Curso de java

TUTORIAL MUY COMPLETO DESDE 0

Desde este código QR podrás acceder a todos los videotutoriales de este curso realizado por Diego Moisset de Espanes.

Pere Manel Verdugo Zamora
pereverdugo@gmail.com | www.peremanelv.com

Contenido

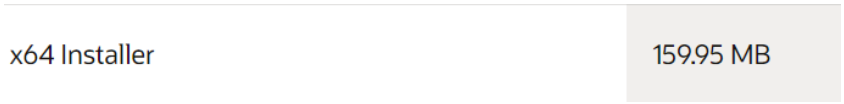
Capítulo 1.- Instalación de Java

Vamos a acceder al siguiente enlace:

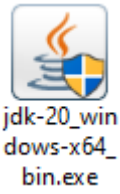
https://www.oracle.com/java/technologies/downloads/#jdk20-windows

Linux macOS Windows

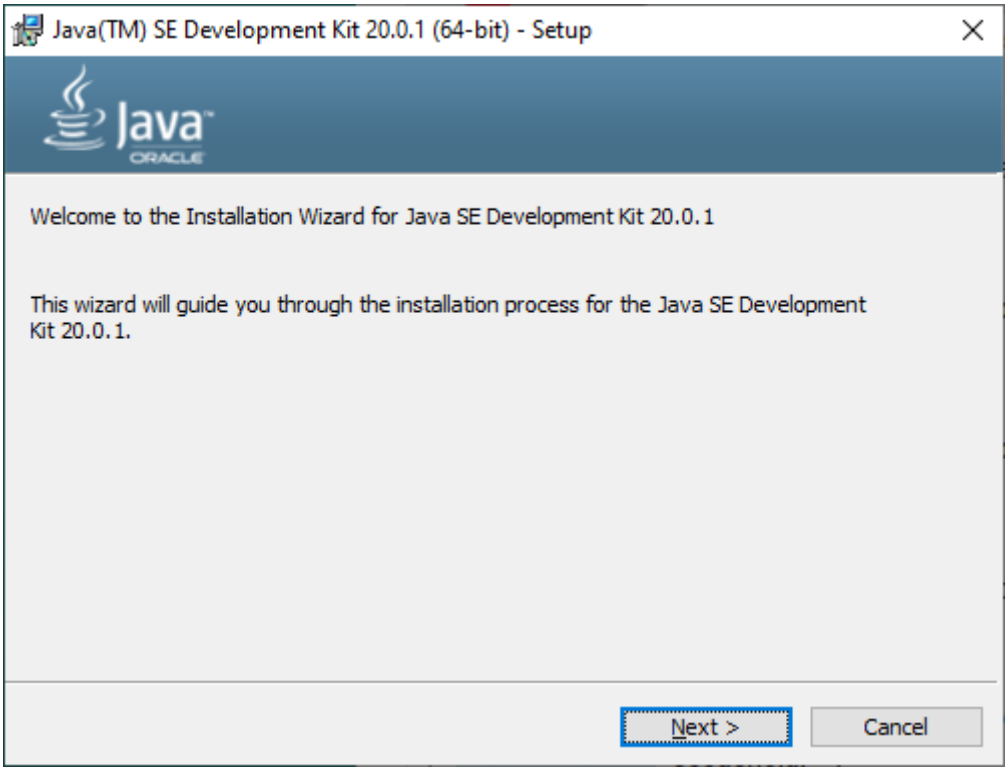
Product/file description	File size	Download
x64 Compressed Archive	180.81 MB	https://download.oracle.com/java/20/latest/jdk-20_windows-x64_bin.zip (sha256)
x64 Installer	159.95 MB	https://download.oracle.com/java/20/latest/jdk-20_windows-x64_bin.exe (sha256)
x64 MSI Installer	158.74 MB	https://download.oracle.com/java/20/latest/jdk-20_windows-x64_bin.msi (sha256)



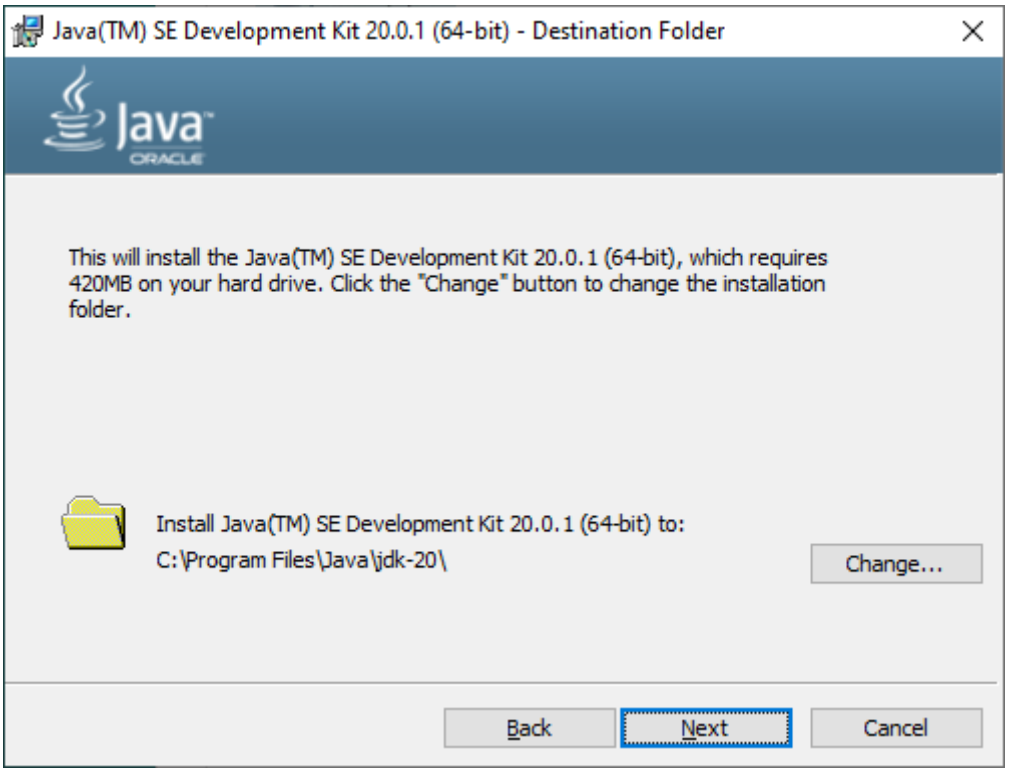
Seleccionaremos x64 Intaller.



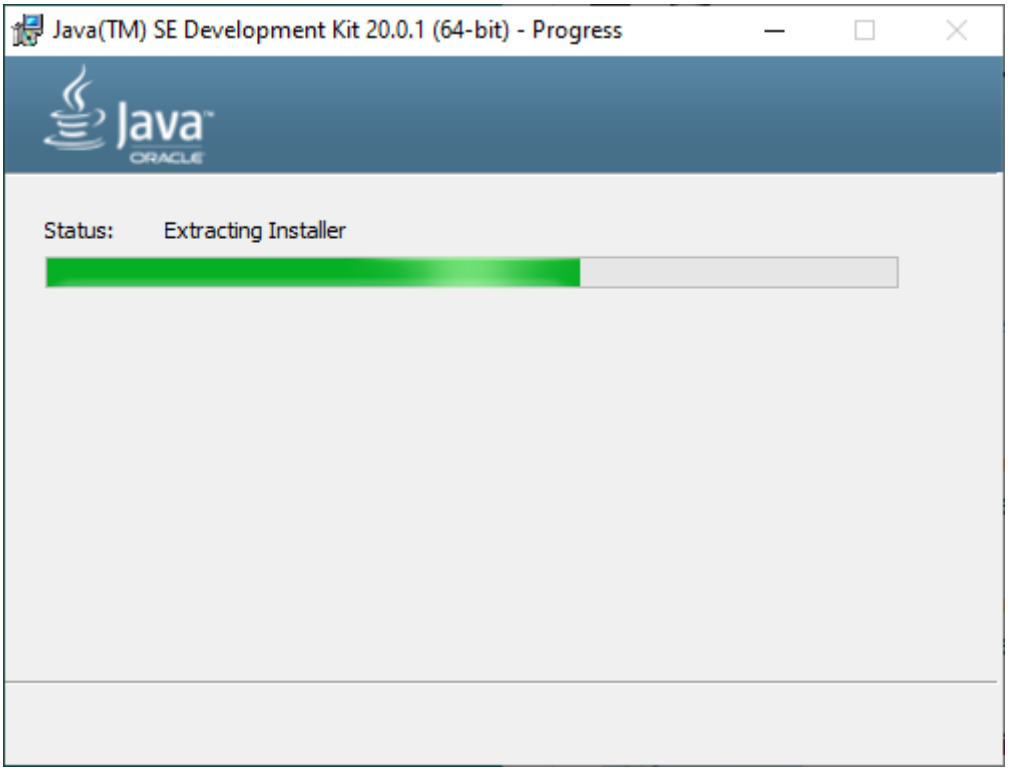
Vamos a proceder a su instalación.



Le damos a Next.



Le damos a Next.

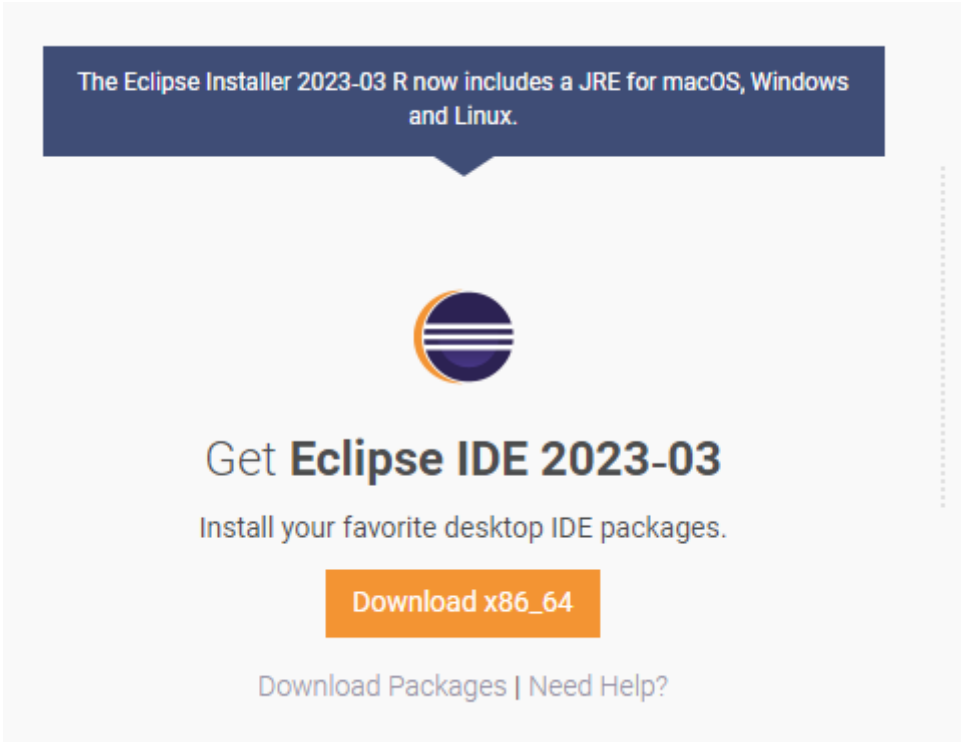


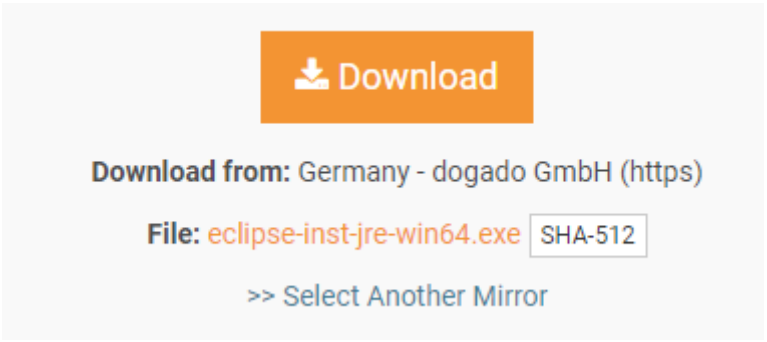
Le damos a Close.

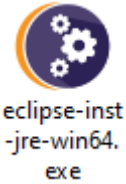
Capítulo 2.- Instalación del editor Eclipse

Vamos al siguiente enlace: https://www.eclipse.org/

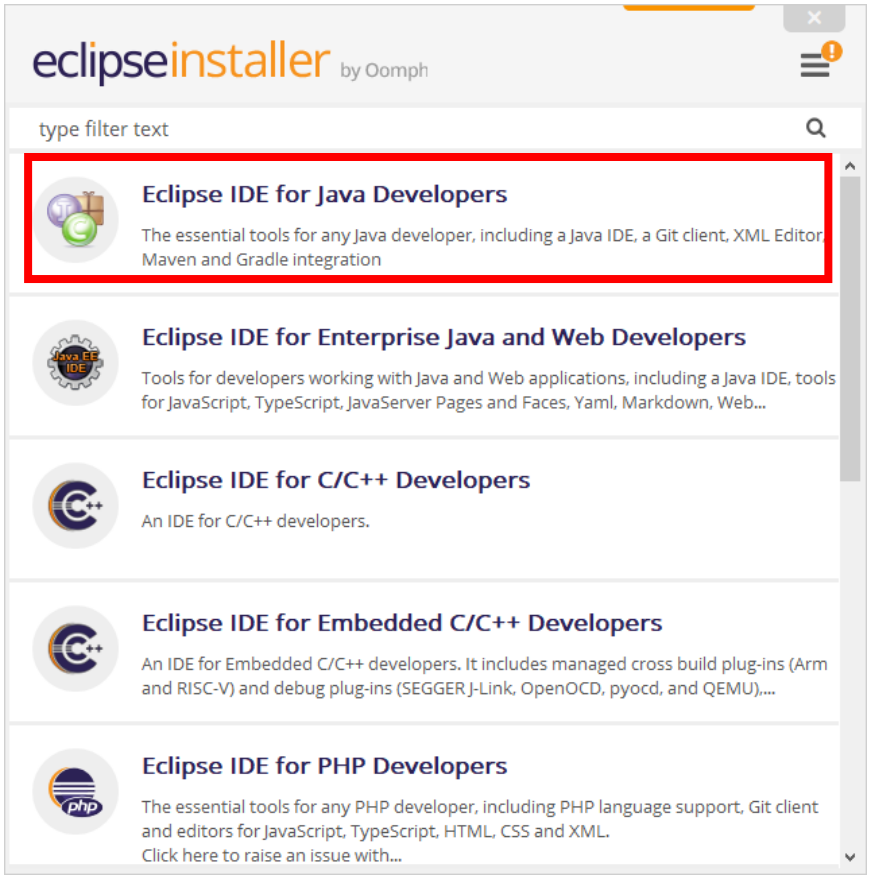
Procedemos a la descarga.



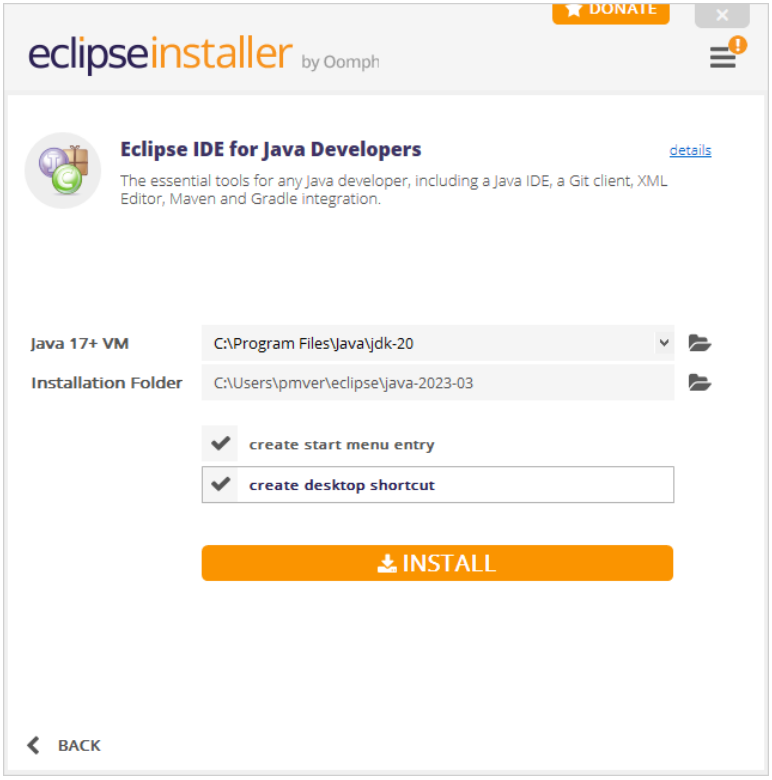




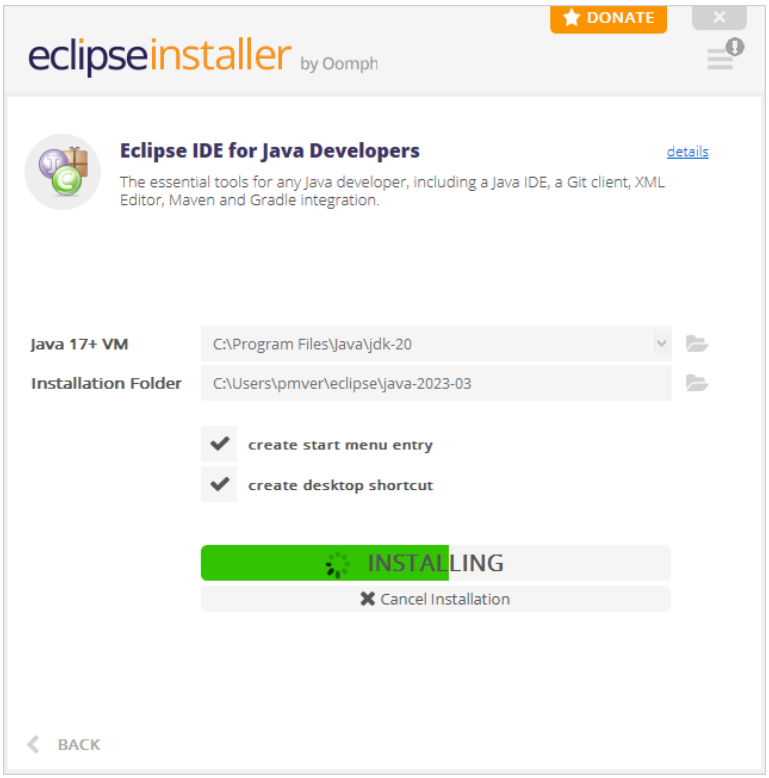
Empezamos con la instalación.



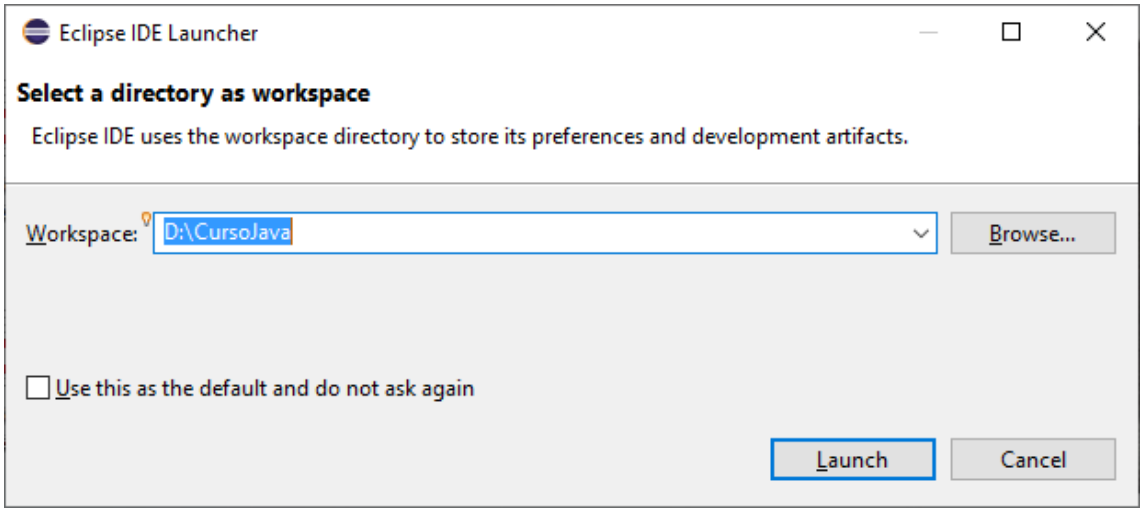
Seleccionaremos Eclipse IDE for Java Developers.



Le damos a INSTALL

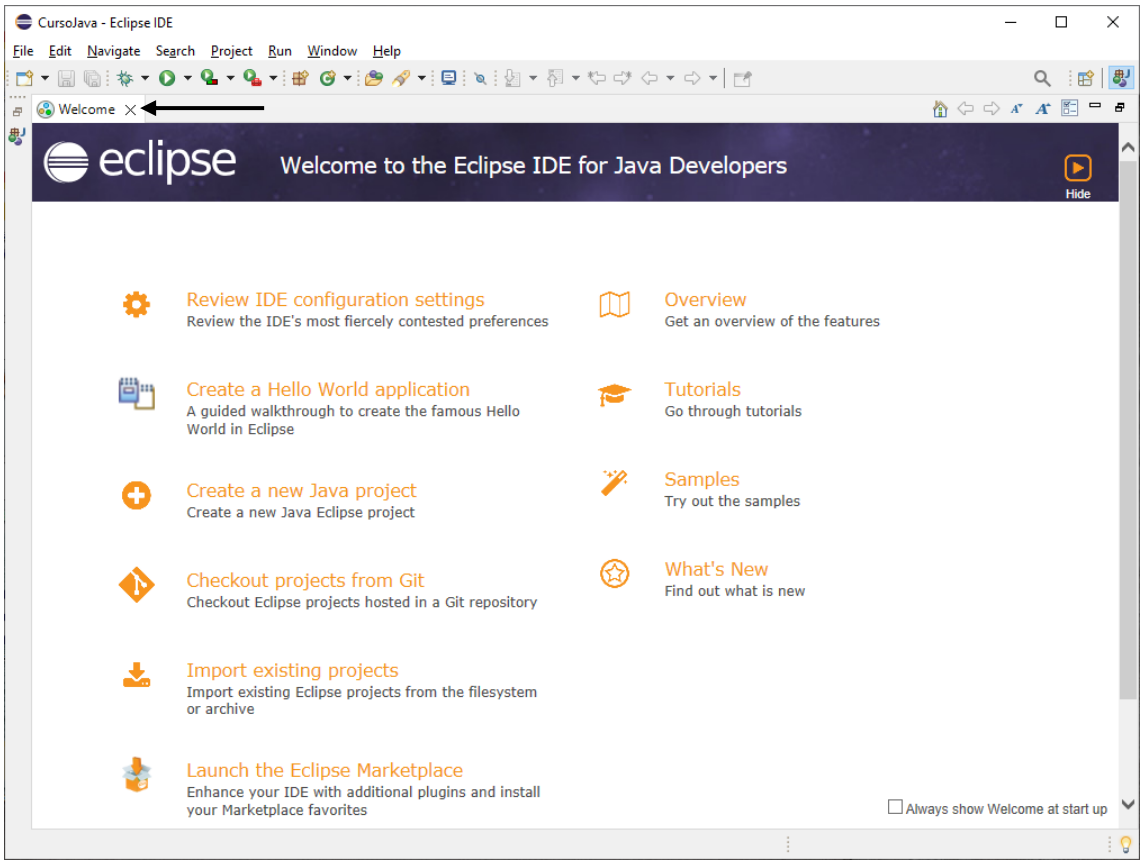


Una vez finalizada la instalación ya tenemos un acceso directo en nuestro escritorio.



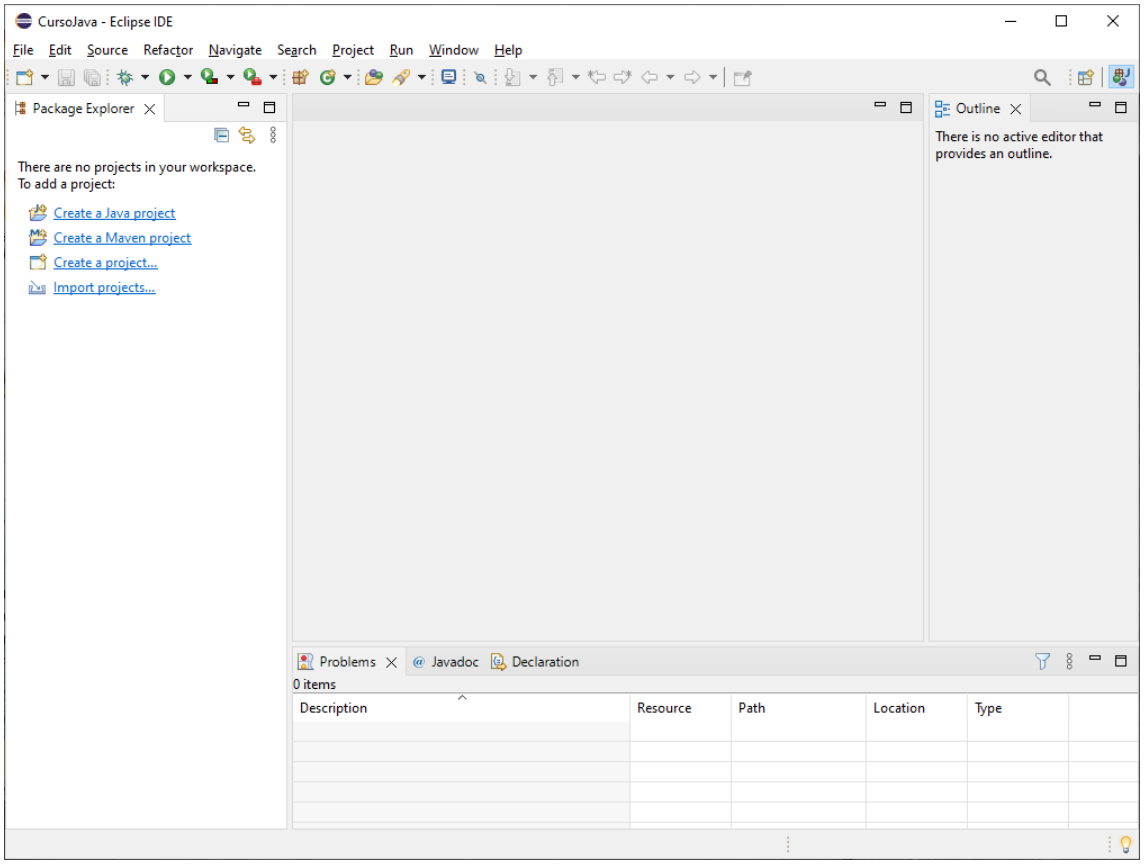
Nos pregunta en que directorio queremos guardar los proyectos.

Seleccionamos el botón Launch.



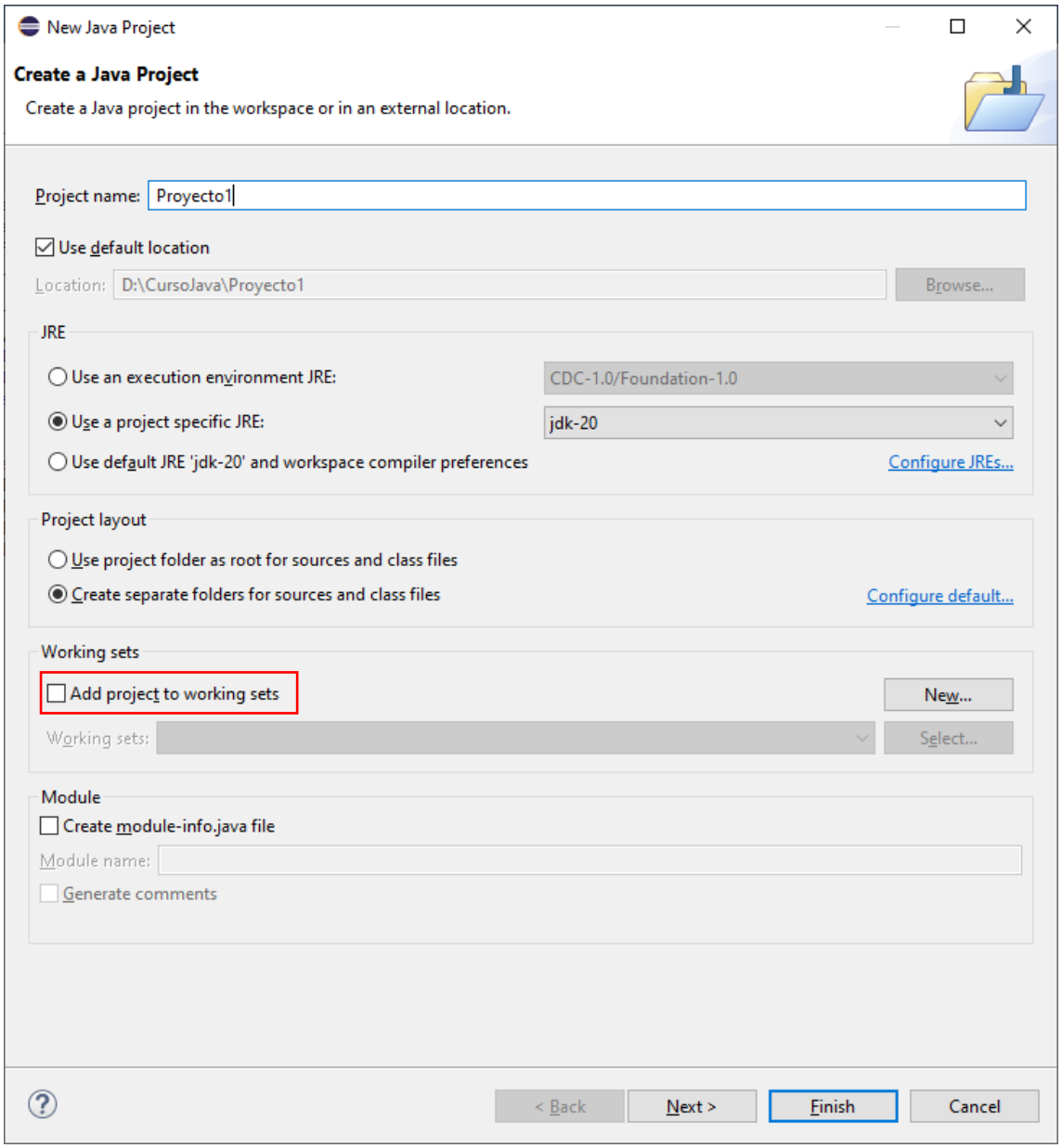
Ya se ha cargado el editor.

Cerramos esta pestaña, ya tenemos el entorno de desarrollo.



Capítulo 3.- Pasos para crear un programa con Eclipse

Para crear un nuevo proyecto del menú File seleccionaremos New y de este Java Proyect.

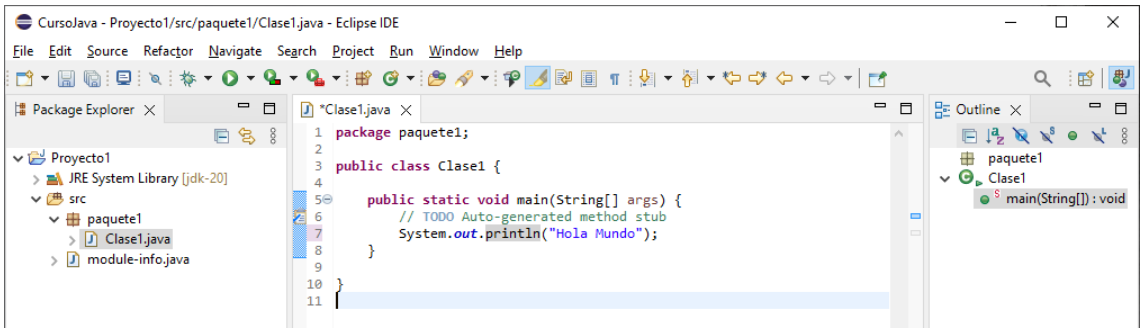


Desactivamos la casilla 'Add project to working sets', seguido del botón Finish.

Del menú File, seleccionamos New y de este Class.

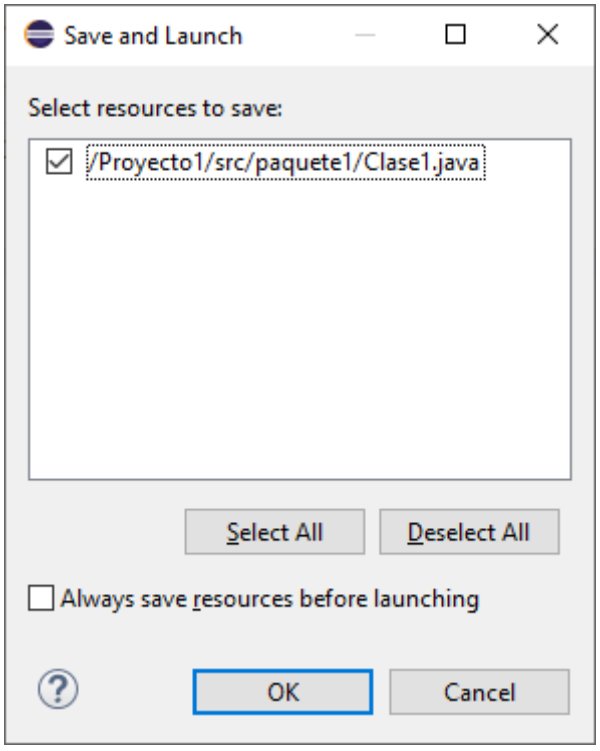
Como nombre de la clase Clase1, y activamos la opción public static void main(String[] args)

Seguido del botón Finsh.

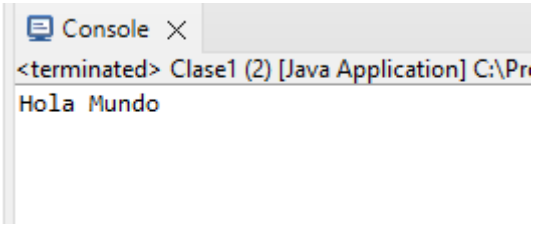



```java
package paquete1;

public class Clase1 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hola Mundo");
    }

}
```

Escribimos la siguiente línea, para ejecutar el programa en la parte superior le daremos a ejecutar.



Le damos a OK.



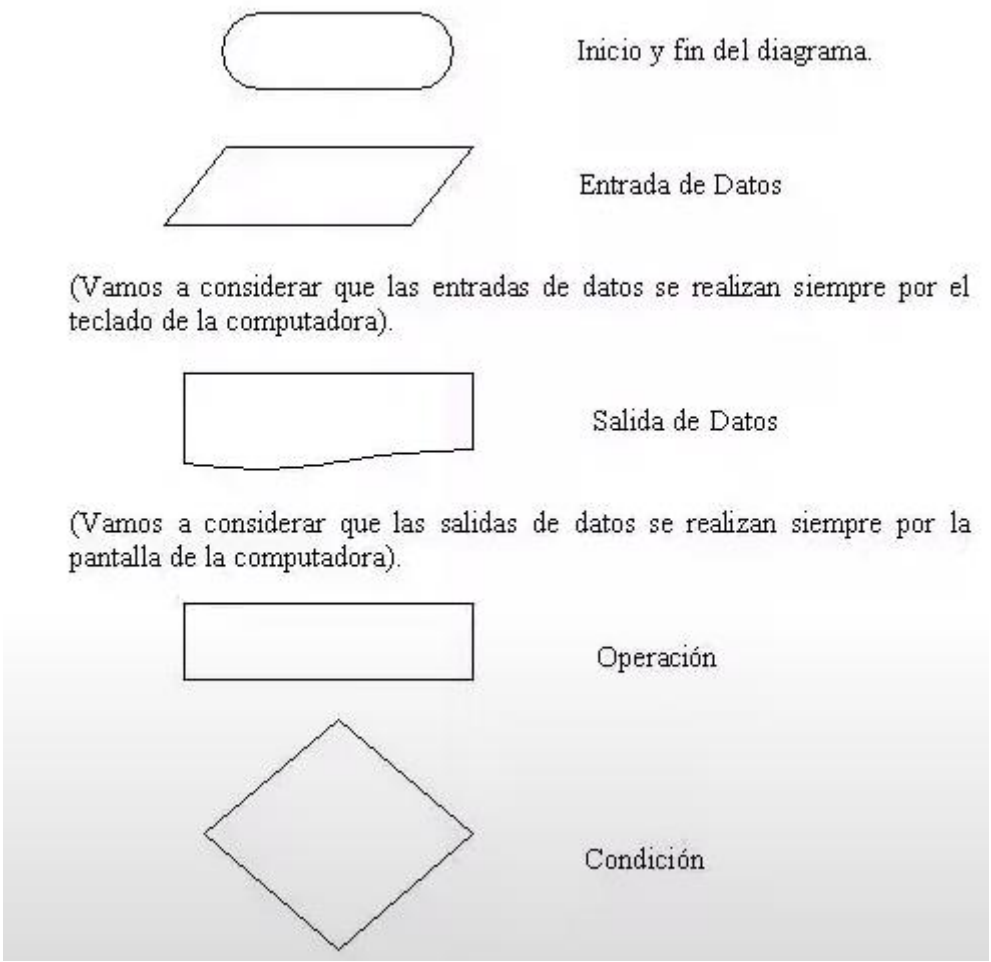
En la parte inferior se muestra la consola con el resultado.

Capítulo 4.- Objetivos del curso y nociones básicas indispensables – diagrama de flujo

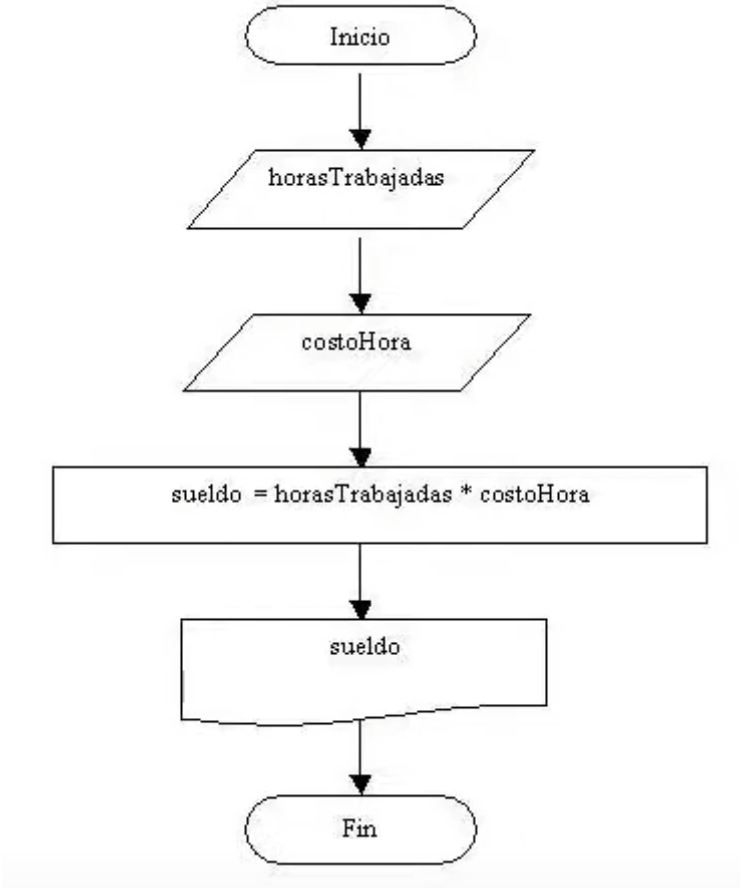
Programa: Conjunto de instrucciones que entiende un ordenador para realizar una actividad.

Algoritmo: Son los pasos a seguir para resolver un problema.

Diagrama de flujo: es la representación gráfica de un ALGORITMO.



Problema: Calcula es sueldo mensual de un operario conociendo la cantidad de horas trabajadas y el pago por hora.



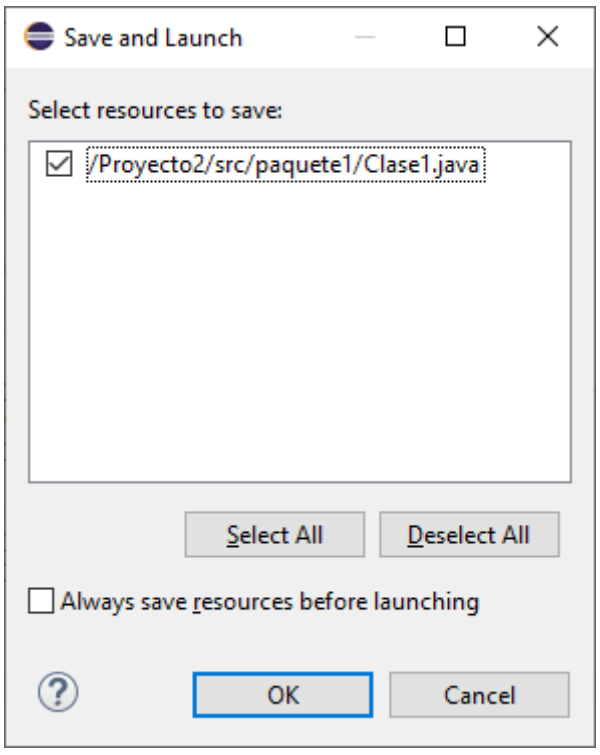
Capítulo 5.- Objetivos del curso y nociones básicas indispensables – codificación

Vamos a realizar el programa comentado en el capítulo anterior, vamos a realizar un nuevo proyecto llamado Proyecto2.

*Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8
9          Scanner teclado=new Scanner(System.in);
10
11         int horasTrabajadas;
12         float precioHora;
13         float sueldo;
14
15         System.out.print("Ingrese las horas trabajadas:");
16         horasTrabajadas=teclado.nextInt();
17         System.out.print("Precio por hora:");
18         precioHora=teclado.nextFloat();
19         sueldo=horasTrabajadas*precioHora;
20         System.out.print("Sueldo del operario:");
21         System.out.println(sueldo);
22     }
23 }
```

Vamos a ejecutar:



Guardamos el proyecto seleccionando OK.

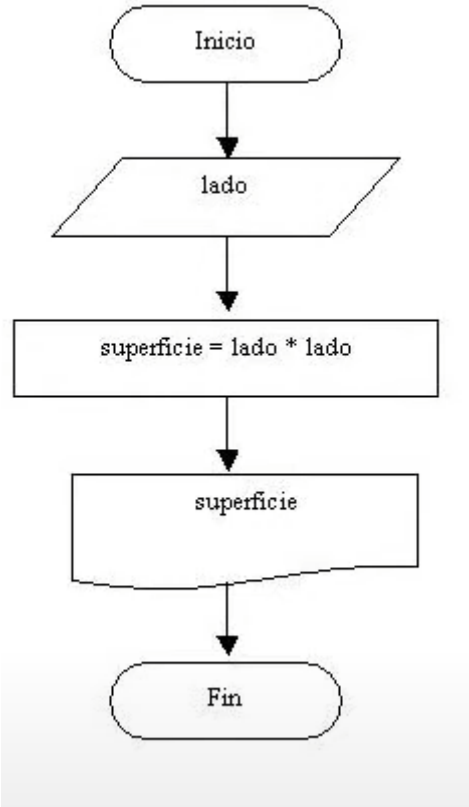
```
Ingrese las horas trabajadas:100
Precio por hora:60
Sueldo del operario:6000.0
```

La diferencia de print y println es que el segundo realiza un salto de línea mientras el primero no.

Capítulo 6.- Errores sintácticos y lógicos

Problema:

Hallar la superficie de una cuadrado conociendo el valor de un lado.



Vamos con un proyecto nuevo.

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8
9          Scanner teclado=new Scanner(System.in);
10
11         int ladoCuadrado;
12         int superficie;
13
14         System.out.print("Ingrese el lado del cuadrado:");
15         ladoCuadrado=teclado.nextInt();
16         superficie=ladoCuadrado*ladoCuadrado;
17         System.out.print("La superficie del cuadrado:");
18         System.out.println(superficie);
19     }
20 }
```

Vamos a ejecutar:

```
Ingrese el lado del cuadrado:5
La superficie del cuadrado:25
```

Los errores sintácticos es cuando escribimos erróneamente una palabra, nos dejamos un punto y como, etc. Estos errores los detecta el editor avisando de un error.

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8
9          Scanner teclado=new Scanner(System.in);
10
11         int ladoCuadrado;
12         int superficie;
13
14         System.out.print("Ingrese el lado del cuadrado:")
15         ladoCuadrado1=teclado.nextInt();
16         superficie=ladoCuadrado*ladoCuadrado;
17         System.out.print("La superficie del cuadrado:");
18         System.out.println(superficie);
19     }
20 }
```

Los errores lógicos son más difíciles de detectar en este caso en la línea 16 la formula no es correcta.

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8
9          Scanner teclado=new Scanner(System.in);
10
11         int ladoCuadrado;
12         int superficie;
13
14         System.out.print("Ingrese el lado del cuadrado:");
15         ladoCuadrado=teclado.nextInt();
16         superficie=ladoCuadrado*ladoCuadrado*ladoCuadrado;
17         System.out.print("La superficie del cuadrado:");
18         System.out.println(superficie);
19     }
20 }
21
```

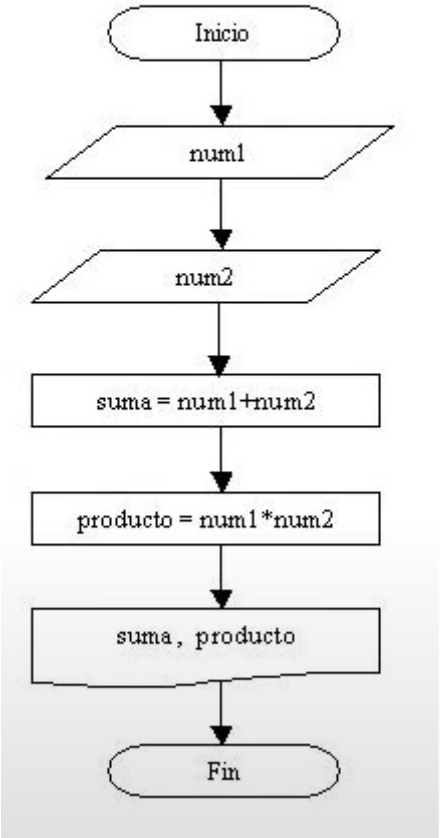
Cuando ejecutemos el resultado no será correcto.

```
Ingrese el lado del cuadrado:10
La superficie del cuadrado:1000
```

Capítulo 7.- Estructura de programación secuencial – 1

Problema:

Realizar la carga de dos números enteros por teclado e imprimir su suma y su producto.



Vamos a programar:

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] arg) {
8
9          Scanner teclado=new Scanner(System.in);
10
11         int num1;
12         int num2;
13         int suma;
14         int producto;
15
```

```
        System.out.print("Ingrese el primer numero:");
        num1=teclado.nextInt();
        System.out.print("Ingrese el segundo numero:");
        num2=teclado.nextInt();
        suma=num1+num2;
        producto=num1*num2;
        System.out.print("La suma es ");
        System.out.println(suma);
        System.out.print("El producto es ");
        System.out.println(producto);
    }
}
```

Si ejecutamos este será el resultado:

```
Ingrese el primer numero:7
Ingrese el segundo numero:5
La suma es 12
El producto es 35
```

También se pueden definir todas las variables en una línea.

```
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] arg) {

        Scanner teclado=new Scanner(System.in);

        int num1, num2, suma, producto;

        System.out.print("Ingrese el primer numero:");
        num1=teclado.nextInt();
        System.out.print("Ingrese el segundo numero:");
        num2=teclado.nextInt();
        suma=num1+num2;
        producto=num1*num2;
        System.out.print("La suma es ");
        System.out.println(suma);
        System.out.print("El producto es ");
        System.out.println(producto);
    }
}
```

Capítulo 8.- Estructura de programación secuencial – 2

Problema:

Realizar la carga del lado de un cuadrado, mostrar por pantalla el perímetro del mismo.

(El perímetro de un cuadrado se calcula multiplicando el valor del lado por cuatro).

Vamos a programar:

Clase1.java

```java
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] args) {

        Scanner teclado=new Scanner(System.in);

        int lado, perimetro;

        System.out.print("Ingrese el lado de un cuadrado:");
        lado=teclado.nextInt();
        perimetro=lado*4;
        System.out.print("El perimetro del ccuadrado es ");
        System.out.println(perimetro);
    }
}
```

Si ejecutamos este será el resultado:

```
Ingrese el lado de un cuadrado:7
El perimetro del ccuadrado es 28
```

Capítulo 9.- Estructura de programación secuencial – 3

Problema:

Escribir un programa en el cual ingresen cuatro números, calcular e informar la suma de los dos primeros y el producto del tercero y el cuarto.

Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8
9          Scanner teclado=new Scanner(System.in);
10
11         int num1, num2, num3, num4;
12         int suma, producto;
13
14         System.out.print("Ingrese el primer numero: ");
15         num1=teclado.nextInt();
16         System.out.print("Ingrese el segundo numero: ");
17         num2=teclado.nextInt();
18         System.out.print("Ingrese el tercero numero: ");
19         num3=teclado.nextInt();
20         System.out.print("Ingrese el cuarto numero: ");
21         num4=teclado.nextInt();
22
23         suma=num1+num2;
24         producto=num3*num4;
25
26         System.out.print("La suma del primer y segundo numero es ");
27         System.out.println(suma);
28         System.out.print("el producto del tercer y cuarto numero es ");
29         System.out.print(producto);
30     }
31 }
```

Si ejecutamos este será el resultado:

```
Ingrese el primer numero: 7
Ingrese el segundo numero: 3
Ingrese el tercero numero: 5
Ingrese el cuarto numero: 4
La suma del primer y segundo numero es 10
el producto del tercer y cuarto numero es 20
```

Si lo necesitas puedes ayudarte con un diagrama de flujo.

Capítulo 10.- Estructura de programación secuencial – 4

Problema:

Realizar un programa que lea cuatro valores numéricos e informar su suma y promedio.

Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8
9          Scanner teclado=new Scanner(System.in);
10
11         int num1, num2, num3, num4;
12         int suma, promedio;
13
14         System.out.print("Ingrese el primer numero: ");
15         num1=teclado.nextInt();
16         System.out.print("Ingrese el segundo numero: ");
17         num2=teclado.nextInt();
18         System.out.print("Ingrese el tercero numero: ");
19         num3=teclado.nextInt();
20         System.out.print("Ingrese el cuarto numero: ");
21         num4=teclado.nextInt();
22
23         suma=num1+num2+num3+num4;
24         promedio=suma/4;
25
26         System.out.print("La suma de los 4 numeros es ");
27         System.out.println(suma);
28         System.out.print("Su prommedio es ");
29         System.out.print(promedio);
30     }
31 }
```

Si ejecutamos este será el resultado:

```
Ingrese el primer numero: 7
Ingrese el segundo numero: 10
Ingrese el tercero numero: 8
Ingrese el cuarto numero: 3
La suma de los 4 numeros es 28
Su prommedio es 7
```

Capítulo 11.- Estructura de programación secuencial – 5

Problema:

Se debe desarrollar un programa que pida el ingreso del precio de un artículo y la cantidad que lleva el cliente.

Mostrar lo que debe abonar el comprador.

Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8
9          Scanner teclado=new Scanner(System.in);
10
11         float precio, importe;
12         int cantidad;
13
14
15         System.out.print("Ingrese el precio del articulo: ");
16         precio=teclado.nextFloat();
17         System.out.print("Ingrese el numero de unidades: ");
18         cantidad=teclado.nextInt();
19
20         importe=precio*cantidad;
21
22         System.out.print("El importe a abonar sera de ");
23         System.out.print(importe);
24
25     }
26 }
```

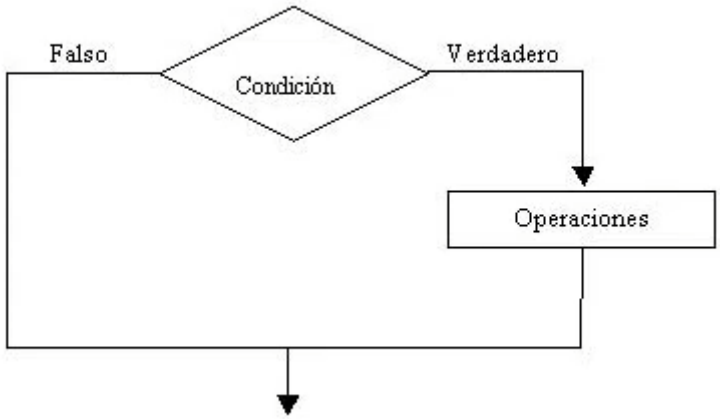
Si ejecutamos este será el resultado:

```
Ingrese el precio del articulo: 25,32
Ingrese el numero de unidades: 5
El importe a abonar sera de 126.6
```

Como separador decimal utilizaremos la coma, ya que en nuestro idioma está configurado de este modo.

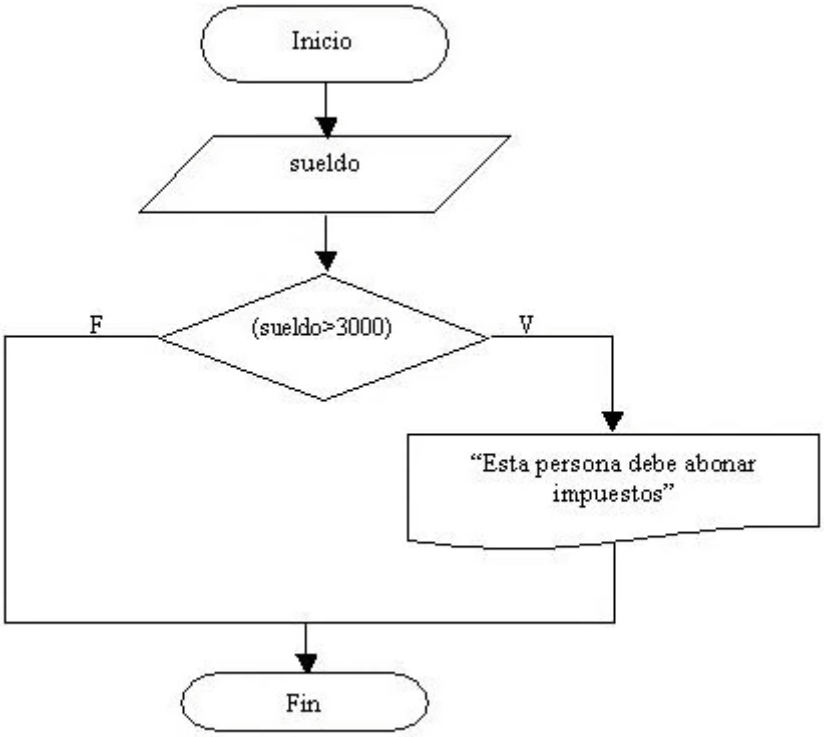
Capítulo 12.- Estructuras condicionales simples y compuestas – 1

Estructuras condicionales simple



Problema:

Ingresar el sueldo de una persona, si supera los 300 pesos mostrar un mensaje en pantalla indicando que debe abonar impuestos.



Vamos a programar:

Clase1.java

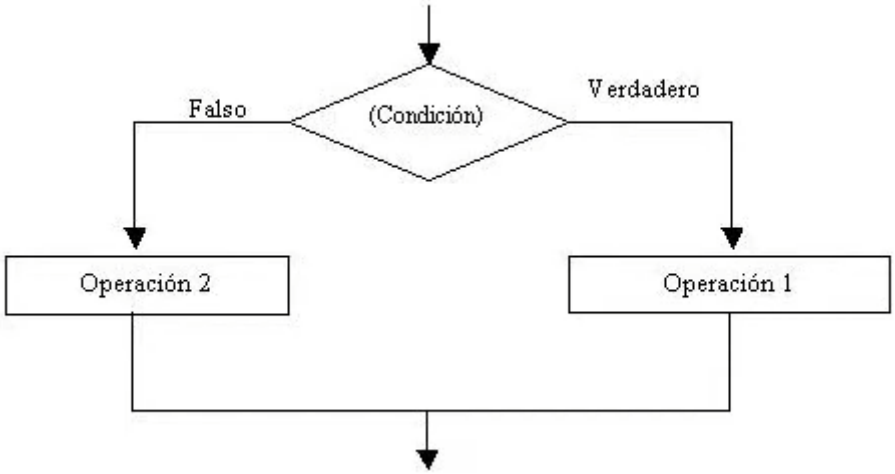
```java
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);

        float sueldo;
        System.out.print("Ingrese sueldo:");
        sueldo=teclado.nextFloat();
        if (sueldo>3000) {
            System.out.print("Esta presona debe abonar impuestos:");
        }
    }
}
```

Si ejecutamos este será el resultado:

```
Ingrese sueldo:5000
Esta presona debe abonar impuestos:
```

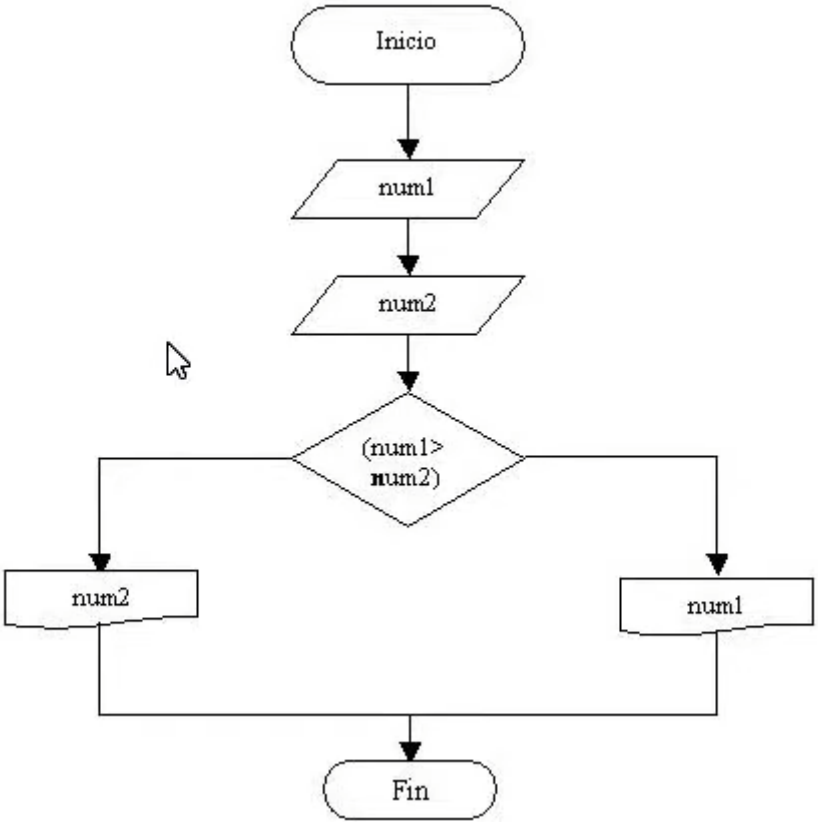
Capítulo 13.- Estructuras condicionales simples y compuestas – 2

Estructura condicional compuesta



Problema:

Realizar un programa que solicite ingresar dos números distintos y muestre por pantalla el mayor de ellos.



Vamos a programar:

Clase1.java

```java
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);

        int num1, num2;
        System.out.print("Ingrese el primer valor:");
        num1=teclado.nextInt();
        System.out.print("Ingrese el segundo valor:");
        num2=teclado.nextInt();
        System.out.print("El valor mayor es ");
        if(num1>num2) {
            System.out.print(num1);
        }else {
            System.out.print(num2);
        }
    }
}
```

Si ejecutamos este será el resultado:

```
Ingrese el primer valor:51
Ingrese el segundo valor:32
El valor mayor es 51
```

Capítulo 14.- Estructuras condicionales simples y compuestas – 3

Problema:

Realizar un programa que lea por teclado dos números, si el primero es mayor al segundo informar su suma y diferencia, en caso contrario informar el producto y la división del primero respecto al segundo.

Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int num1, num2;
11         int suma, diferencia, producto, division;
12
13         System.out.print("Ingrese el primer valor:");
14         num1=teclado.nextInt();
15         System.out.print("Ingrese el segundo valor:");
16         num2=teclado.nextInt();
17
18         if(num1>num2) {
19             suma=num1+num2;
20             diferencia=num1-num2;
21             System.out.print("la suma es ");
22             System.out.println(suma);
23             System.out.print("la diferencia es ");
24             System.out.println(diferencia);
25
26         }else {
27             producto=num1*num2;
28             division=num1/num2;
29             System.out.print("el producto es ");
30             System.out.println(producto);
31             System.out.print("la division es ");
32             System.out.println(division);
33         }
34     }
35 }
```

Vamos a ejecutar introduciendo los valores 10 y 5

```
Ingrese el primer valor:10
Ingrese el segundo valor:5
la suma es 15
la diferencia es 5
```

Ahora los valores 12 y 70.

```
Ingrese el primer valor:12
Ingrese el segundo valor:70
el producto es 840
la division es 0
```

Las variables suma y diferencia se podría definir en el bloque de verdadero y las variables producto y división en el bloque de falso.

Capítulo 15.- Estructuras condicionales simples y compuestas – 4

Problema:

Se ingresan tres notas de un alumno, si el promedio es mayor o igual a siete mostrar un mensaje "Promocionado".

Vamos a programar:

*Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         float nota1, nota2, nota3, promedio;
11
12         System.out.print("Ingrese la primera nota: ");
13         nota1=teclado.nextFloat();
14         System.out.print("Ingrese la segunda nota: ");
15         nota2=teclado.nextFloat();
16         System.out.print("Ingrese la tercera nota: ");
17         nota3=teclado.nextFloat();
18
19         promedio=(nota1+nota2+nota3)/3;
20         System.out.print("Promedio:");
21         System.out.println(promedio);
22         if(promedio>=7) {
23             System.out.print("Este alumno esta promocionado");
24         }
25     }
26 }
```

Si ejecutamos este será el resultado introduciendo las notas de 7, 7 y 6.

```
Ingrese la primera nota: 7
Ingrese la segunda nota: 7
Ingrese la tercera nota: 6
Promedio:6.6666665
```

Vamos a ejecutar de nuevo introduciendo las notas 8, 8 y 7.

```
Ingrese la primera nota: 8
Ingrese la segunda nota: 8
Ingrese la tercera nota: 7
Promedio:7.6666665
Este alumno esta promocionado
```

Capítulo 16.- Estructuras condicionales simples y compuestas – 5

Problema:

Se ingresa por teclado un número positivo de uno o dos dígitos (1..99) mostrar un mensaje indicando si el número tiene uno o dos dígitos.

(Tener en cuenta que la condición debe cumplirse para tener dos dígitos, un número entero)

Vamos a programar:

*Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int numero;
11
12         System.out.print("Ingrese un numero entre 1 y 99:");
13         numero=teclado.nextInt();
14         if(numero>=10) {
15             System.out.print("Este numero tiene 2 digitos");
16         }else {
17             System.out.print("Este numero tiene 1 digito");
18         }
19     }
20 }
```

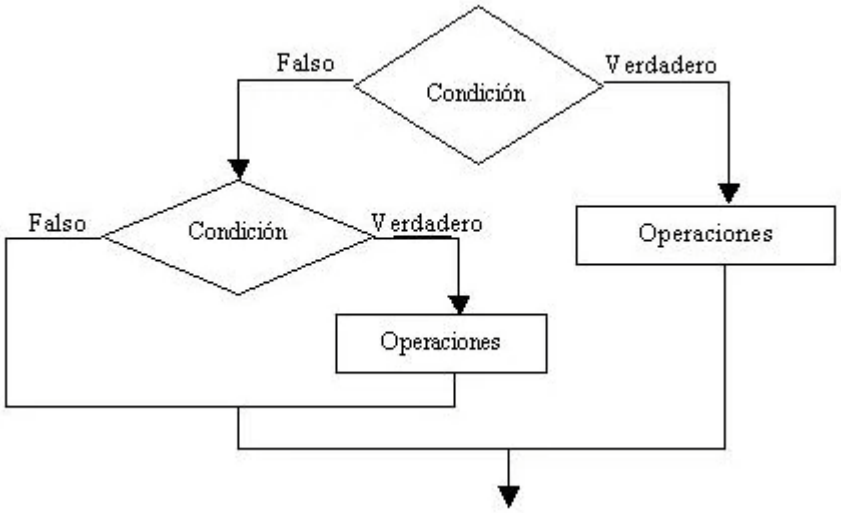
Si ejecutamos y introducimos el valor 12.

```
Ingrese un numero entre 1 y 99:12
Este numero tiene 2 digitos
```

Vamos a ejecutar de nuevo y contestaremos con el valor 7.

```
Ingrese un numero entre 1 y 99:7
Este numero tiene 1 digito
```

Capítulo 17.- Estructuras condicionales anidadas – 1



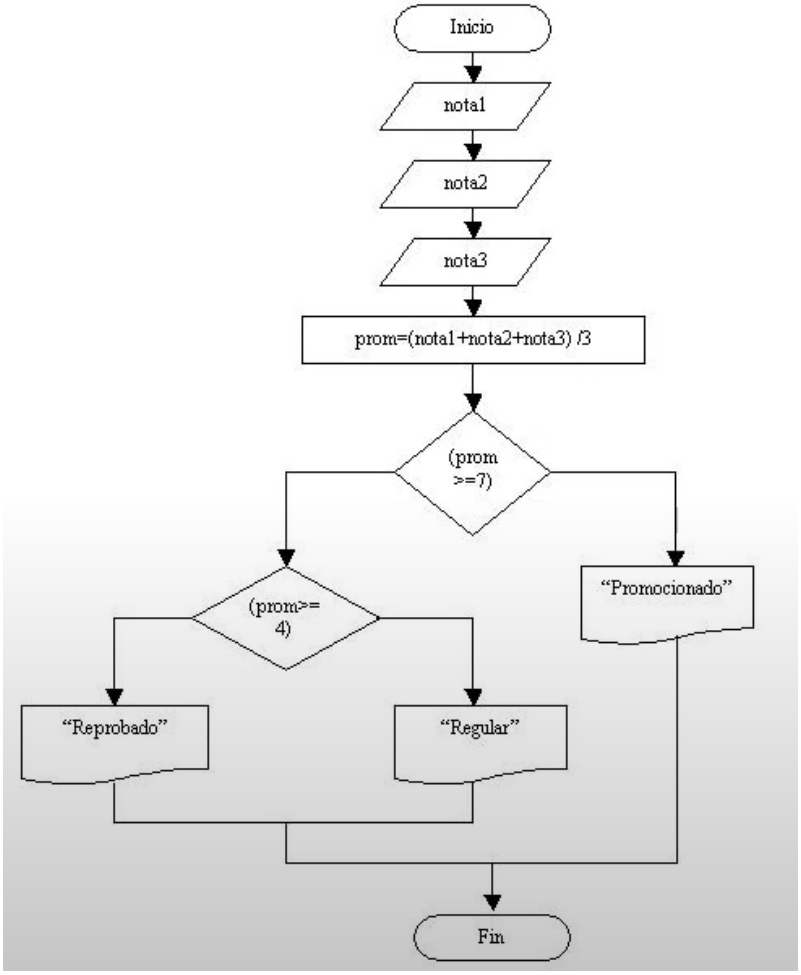
Problema:

Confeccionar un programa que pida por teclado tres notas de un alumno, calcule el promedio e imprima alguno de estos mensajes.

Si el promedio es >= 7 mostrar "Promocionado"

Si el promedio es >=4 y <7 mostrar "Regular".

Si el promedio es <4 mostrar "Reprobado.



Vamos a programar:

*Clase1.java

```java
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);

        float nota1, nota2, nota3;
        float promedio;

        System.out.print("Ingrese la primera nota:");
        nota1=teclado.nextFloat();
        System.out.print("Ingrese la segunda nota:");
        nota2=teclado.nextFloat();
        System.out.print("Ingrese la tercera nota:");
        nota3=teclado.nextFloat();
        promedio=(nota1+nota2+nota3)/3;

        if(promedio>=7) {
            System.out.print("Su calificacion es de Promocionado");
        }else {
            if(promedio>=4) {
                System.out.print("Su calificacion es de Reprobado");
            }else {
                System.out.print("Su calificacion es de Regular");
            }
        }
    }
}
```

Vamos a ejecutar introduciendo los siguientes valores 1, 2 y 4.

```
Ingrese la primera nota:1
Ingrese la segunda nota:2
Ingrese la tercera nota:4
Su calificacion es de Regular
```

Ejecutamos de nuevo con los siguientes valores 6, 7 y 8.

```
Ingrese la primera nota:6
Ingrese la segunda nota:7
Ingrese la tercera nota:8
Su calificacion es de Promocionado
```

Ejecutamos de nuevo con los siguientes valores 5, 5 y 6.

```
Ingrese la primera nota:5
Ingrese la segunda nota:5
Ingrese la tercera nota:6
Su calificacion es de Reprobado
```

Capítulo 18.- Estructuras condicionales anidadas – 2

Problema:

Se carga por teclado tres números distintos. Mostrar en pantalla el mayor de ellos.

Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int numero1, numero2, numero3;
11
12         System.out.print("Ingrese el primer numero:");
13         numero1=teclado.nextInt();
14         System.out.print("Ingrese el segundo numero:");
15         numero2=teclado.nextInt();
16         System.out.print("Ingrese el tercer numro:");
17         numero3=teclado.nextInt();
18         System.out.print("El numero mayor es ");
19         if (numero1>numero2) {
20             if(numero1>numero3) {
21                 System.out.print(numero1);
22             }
23         }
24         else {
25             if (numero2>numero3) {
26                 System.out.print(numero2);
27             }else {
28                 System.out.print(numero3);
29             }
30         }
31     }
32 }
```

Vamos a ejecutar con los valores 1, 2 y 3

```
Ingrese el primer numero:1
Ingrese el segundo numero:2
Ingrese el tercer numro:3
El numero mayor es 3
```

Ejecutamos de nuevo introduciendo los valores 1, 3 y 2

```
Ingrese el primer numero:1
Ingrese el segundo numero:3
Ingrese el tercer numro:2
El numero mayor es 3
```

Ejecutamos de nuevo con los valores 3, 1 y 2

```
Ingrese el primer numero:3
Ingrese el segundo numero:1
Ingrese el tercer numro:2
El numero mayor es 3
```

Capítulo 19.- Estructuras condicionales anidadas – 3

Problema:

Se ingresa por teclado un valor entero, mostrar una leyenda que indique si el número es positivo, nulo o negativo.

Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] arg) {
8          Scanner teclado=new Scanner(System.in);
9
10         int numero;
11
12         System.out.print("Ingrese un numero:");
13         numero=teclado.nextInt();
14
15         System.out.print("El numero es ");
16         if(numero>0) {
17             System.out.print("positivo");
18         }else {
19             if(numero<0) {
20                 System.out.print("negativo");
21             }else {
22                 System.out.print("Nulo");
23             }
24         }
25     }
26 }
```

Si ejecutamos e introducimos el valor 5.

```
Ingrese un numero:5
El numero es positivo
```

Si ejecutamos e introducimos el valor -3.

```
Ingrese un numero:-3
El numero es negativo
```

Si ejecutamos e introducimos el valor 0.

```
Ingrese un numero:0
El numero es Nulo
```

Capítulo 20.- Estructuras condicionales anidadas – 4

Problema:

Confeccionar un programa que permita cargar un numero entero positivo de hasta tres cifras y muestre un mensaje indicando si tiene 1, 2 o 3 cifras. Mostrar un mensaje de error si el número de cifras es mayor.

Vamos a programar:

*Clase1.java

```java
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] args) {
        Scanner teclado= new Scanner(System.in);

        int numero;

        System.out.print("Ingrese un numero:");
        numero=teclado.nextInt();

        if(numero<10) {
            System.out.print("Tiene 1 digito.");
        }else {
            if (numero<100) {
                System.out.print("Tiene 2 digitos");
            }else {
                if(numero<1000) {
                    System.out.print("Tiene 3 digitos");
                }else {
                    System.out.print("Error en la entrada de datos");
                }
            }
        }
    }
}
```

Vamos a ejecutar introduciendo un 7.

```
Ingrese un numero:7
Tiene 1 digito.
```

Vamos a ejecutar introduciendo un 15.

```
Ingrese un numero:15
Tiene 2 digitos
```

Vamos a ejecutar introduciendo 127.

```
Ingrese un numero:127
Tiene 3 digitos
```

Vamos a ejecutar introduciendo 1250.

```
Ingrese un numero:1250
Error en la entrada de datos
```

Capítulo 21.- Estructuras condicionales anidadas – 5

un postulante a un empleo, realiza un test de capacitación, se obtuvo la siguiente información:

Cantidad total de preguntas que se le realizaron y la cantidad de preguntas que contestó correctamente. Se pide confeccionar un programa que ingrese los dos datos por teclado e informe el nivel del mismo según el porcentaje de respuestas correctas que ha obtenido, y sabiendo que:

Nivel máximo: Porcentaje>=90%
Nivel medio: Porcentaje>=75% y <90%
Nivel regular: Porcentaje>=50% y <75%
Fuera de nivel: Porcentaje<50%

Vamos a programar:

*Clase1.java

```java
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);

        int cantpreguntas, cantaciertos, porcentaje;

        System.out.print("Ingrese el numero de preguntas:");
        cantpreguntas=teclado.nextInt();
        System.out.print("Ingrese el numero de aciertos:");
        cantaciertos=teclado.nextInt();

        porcentaje=cantaciertos*100/cantpreguntas;

        System.out.print("Ha obtendido el ");
        if(porcentaje>=90) {
            System.out.print("Nibel maximo");
        }else {
            if(porcentaje>=75) {
                System.out.print("Nivel medio");
            }else {
                if(porcentaje>=50) {
                    System.out.print("Nivel regular");
                }else {
                    System.out.print("Fuera de nivel");
                }
            }
        }
    }
}
```

Este será el resultado si de 100 preguntas ha acertado 80.

```
Ingrese el numero de preguntas:100
Ingrese el numero de aciertos:80
Ha obtendido el Nivel medio
```

Este será el resultado si de 100 preguntas ha acertado 65.

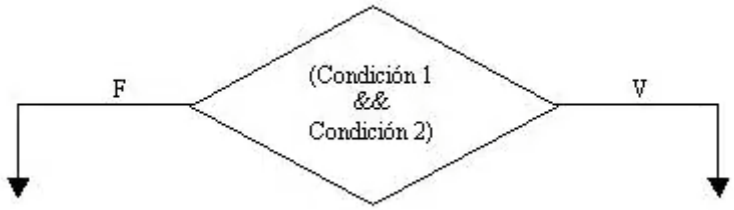
```
Ingrese el numero de preguntas:100
Ingrese el numero de aciertos:65
Ha obtendido el Nivel regular
```

Por último este será el resultado si de 100 preguntas ha acertado 5.

```
Ingrese el numero de preguntas:100
Ingrese el numero de aciertos:5
Ha obtendido el Fuera de nivel
```

Capítulo 22.- Condiciones compuestas con operadores lógicos – 1

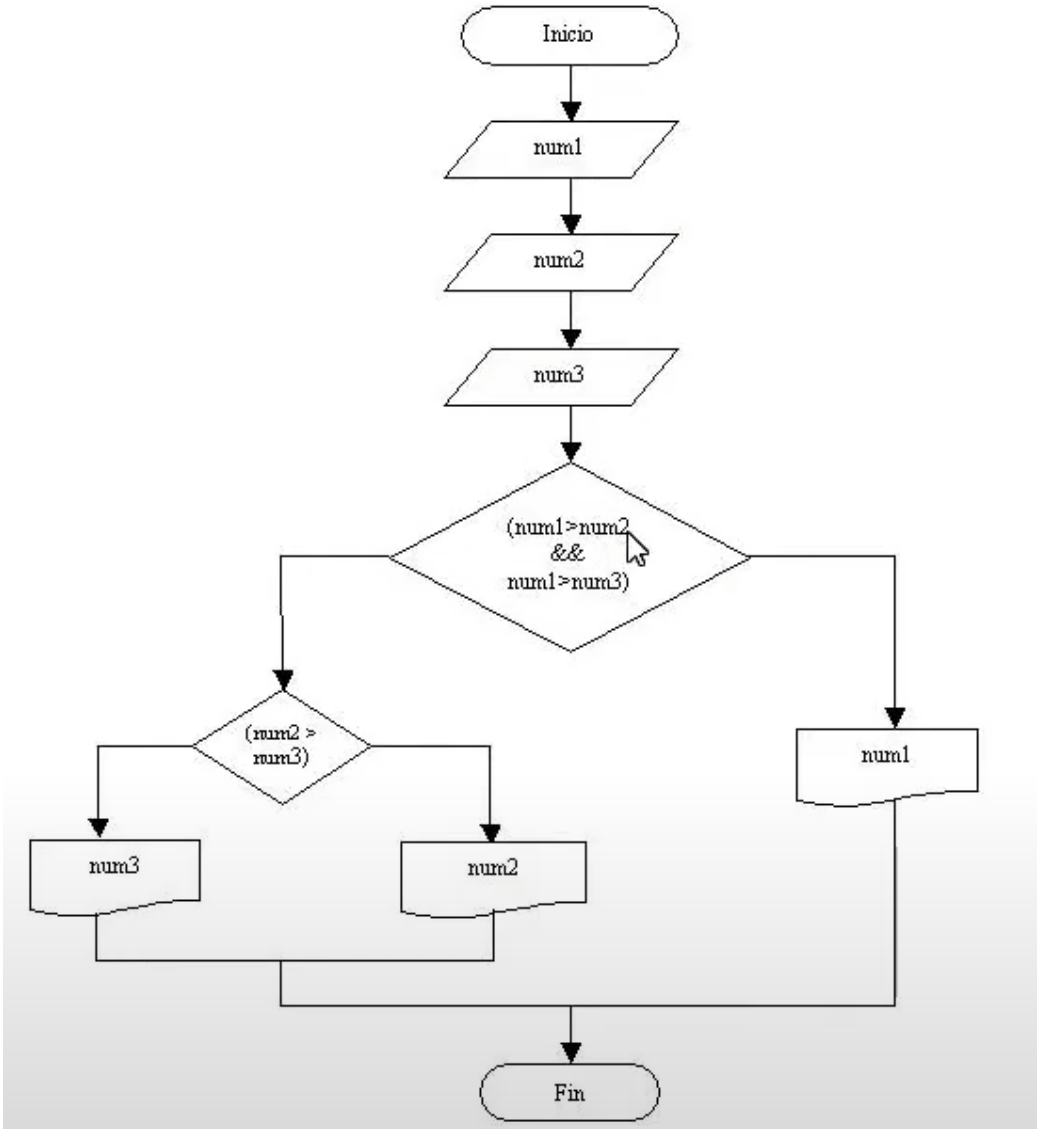
Condiciones compuestas con el operador lógico && (Y)



Operadores relacionales (>, <, >=, <=, ==, !=)
Operadores matemáticos (+, -, *, /, %)
Operadores lógicos (&&)

Problema:

Confeccionar un programa que lea por teclado tres números distintos y nos muestre el mayor.



Vamos a programar:

Clase1.java

```java
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);

        int num1, num2, num3;

        System.out.print("Ingrese el primer numero:");
        num1=teclado.nextInt();
        System.out.print("Ingrese el segundo numero:");
        num2=teclado.nextInt();
        System.out.print("Ingrese el tercer numero:");
        num3=teclado.nextInt();

        System.out.print("El numero mayor es ");
        if (num1>num2 && num1>num3) {
            System.out.print(num1);
        }else {
            if(num2>num3) {
                System.out.print(num2);
            }else {
                System.out.print(num3);
            }
        }
    }
}
```

Vamos a ejecutar introduciendo los valores 1, 2 y 3,

```
Ingrese el primer numero:1
Ingrese el segundo numero:2
Ingrese el tercer numero:3
El numero mayor es 3
```

Vamos a ejecutar introduciendo los valores 1, 3 y 2.

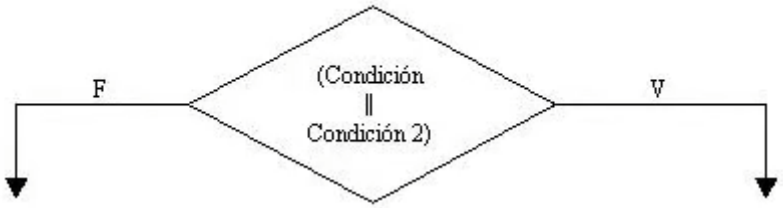
```
Ingrese el primer numero:1
Ingrese el segundo numero:3
Ingrese el tercer numero:2
El numero mayor es 3
```

Vamos a ejecutar introduciendo los valores 3, 1 y 2.

```
Ingrese el primer numero:3
Ingrese el segundo numero:2
Ingrese el tercer numero:1
El numero mayor es 3
```

Capítulo 23.- Condiciones compuestas con operadores lógicos – 2

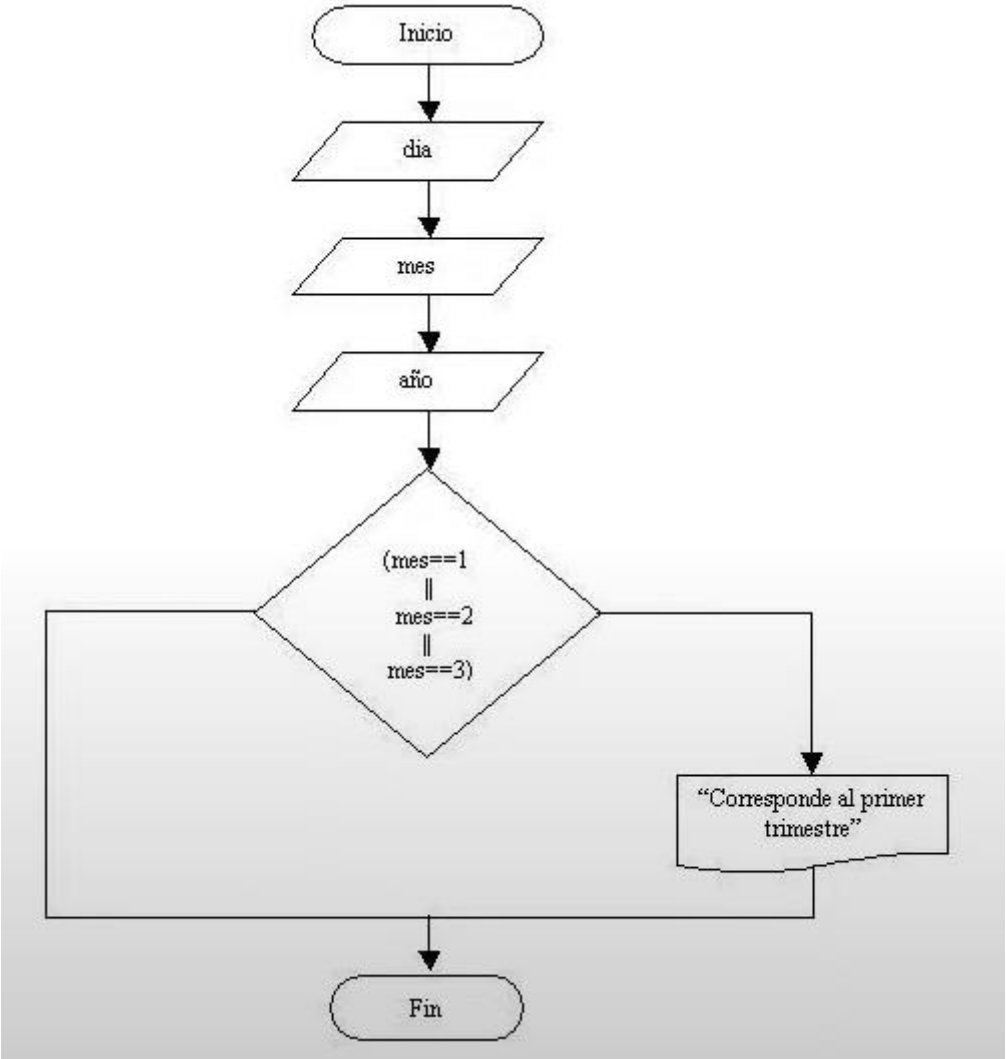
Condiciones compuestas con el operador lógico || (o)



Operadores relacionales (>, <, >=, <=, ==, !=)
Operadores matemáticos (+, -, *, /, %)
Operadores lógicos (&&, ||)

Problema:
Se carga una fecha (día, mes y año) por teclado. Mostrar un mensaje si corresponde al primer trimestre del año (enero, febrero o marzo) Cagar por teclado el valor numérico del día mes y año.

Ejemplo: día:10, mes:1 y año: 2020



Vamos a programar:

Clase1.java

```java
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);

        int dia, mes, anyo;

        System.out.print("Ingrese el dia en numeros:");
        dia=teclado.nextInt();
        System.out.print("Ingrese el mes en numeros:");
        mes=teclado.nextInt();
        System.out.print("Ingrese el anyo:");
        anyo=teclado.nextInt();

        if(mes==1 || mes==2 || mes==3) {
            System.out.print("Corresponde al primer trimestre");
        }
    }
}
```

Vamos a ejecutar contestando por los valores día: 18, mes: 2 y año 2020.

```
Ingrese el dia en numeros:18
Ingrese el mes en numeros:2
Ingrese el anyo:2020
Corresponde al primer trimestre
```

Ejecutamos de nuevo contestado por los valores día: 7, mes: 4 y año 2020.

```
Ingrese el dia en numeros:7
Ingrese el mes en numeros:4
Ingrese el anyo:2020
```

Capítulo 24.- Condiciones compuestas con operadores lógicos – 3

Problema:

Realizar un programa que pida cargar una fecha cualquiera, luego verificar si dicha fecha corresponde a Navidad.

Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int dia, mes, anyo;
11
12         System.out.print("Ingrese el dia en numeros:");
13         dia=teclado.nextInt();
14         System.out.print("Ingrese el mes en numeros:");
15         mes=teclado.nextInt();
16         System.out.print("Ingrese el anyoo:");
17         anyo=teclado.nextInt();
18
19         if(dia==25 && mes==12) {
20             System.out.print("Corresponde a la Navidad");
21         }
22     }
23 }
```

vamos a ejecutar introduciendo en día: 25, mes: 12 y año 2020.

```
Ingrese el dia en numeros:25
Ingrese el mes en numeros:12
Ingrese el anyoo:2020
Corresponde a la Navidad
```

Vamos a ejecutar de nuevo introduciendo en día: 1 mes 12 y año 2020.

```
Ingrese el dia en numeros:1
Ingrese el mes en numeros:12
Ingrese el anyoo:2020
```

Capítulo 25.- Condiciones compuestas con operadores lógicos – 4

Problema:

Se ingresan tres valores por teclado, si todos son iguales se imprime la suma del primero con el segundo y a este resultado se lo multiplica el tercero.

Vamos a programar:

*Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int num1, num2, num3;
11         int suma, producto;
12
13         System.out.print("Ingrese el primer numero:");
14         num1=teclado.nextInt();
15         System.out.print("Ingrese el segundo numero:");
16         num2=teclado.nextInt();
17         System.out.print("Ingrese el tercer numero:");
18         num3=teclado.nextInt();
19
20         if(num1==num2 && num1==num3) {
21             suma=num1+num2;
22             producto=suma*num3;
23             System.out.print("La suma de los dos primeros es ");
24             System.out.println(suma);
25             System.out.print("El resultado es ");
26             System.out.print(producto);
27         }
28     }
29 }
```

Vamos a ejecutar introduciendo los valores 4,4 y 6.

```
Ingrese el primer numero:4
Ingrese el segundo numero:4
Ingrese el tercer numero:6
```

Ejecutamos de nuevo introduciendo los valores 7,7 y 7.

```
Ingrese el primer numero:7
Ingrese el segundo numero:7
Ingrese el tercer numero:7
La suma de los dos primeros es 14
El resultado es 98
```

Capítulo 26.- Condiciones compuestas con operadores lógicos – 5

Problema:

Se ingresan por teclado tres números, si todos los valores ingresados son menores a 10, imprimir en pantalla la leyenda "Todos los números son menores a diez".

Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int num1, num2, num3;
11
12         System.out.print("Ingrese el primer numero:");
13         num1=teclado.nextInt();
14         System.out.print("Ingrese el segundo numero:");
15         num2=teclado.nextInt();
16         System.out.print("Ingrese el tercer numero:");
17         num3=teclado.nextInt();
18
19         if(num1< 10 && num2<10 && num1<10) {
20             System.out.print("Todos los numeros son menores a diez");
21         }
22     }
23 }
```

Si ejecutamos e introducimos los valores 7, 5 y 9.

```
Ingrese el primer numero:7
Ingrese el segundo numero:5
Ingrese el tercer numero:9
Todos los numeros son menores a diez
```

Ejecutamos de nuevo introducimos los valores 9, 11, 5.

```
Ingrese el primer numero:9
Ingrese el segundo numero:11
Ingrese el tercer numero:5
```

Capítulo 27.- Condiciones compuestas con operadores lógicos – 6

Problema:

Se ingresan por teclado tres números, si al menos uno de los valores ingresados es menor a 10 imprimir en pantalla una leyenda "Alguno de los números es menor a diez".

Vamos a programar:

*Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int num1, num2, num3;
11
12         System.out.print("Ingrese el primer numero:");
13         num1=teclado.nextInt();
14         System.out.print("Ingrese el segundo numero:");
15         num2=teclado.nextInt();
16         System.out.print("Ingrese el tercer numero:");
17         num3=teclado.nextInt();
18
19         if(num1< 10 || num2<10 || num1<10) {
20             System.out.print("alguno de los numeros es menor a diez");
21         }
22     }
23 }
```

Vamos a ejecutar introduciendo los valores 45, 77 y 44.

```
Ingrese el primer numero:45
Ingrese el segundo numero:77
Ingrese el tercer numero:44
```

Vamos a ejecutar de nuevo introduciendo los valores 3, 55 y 66.

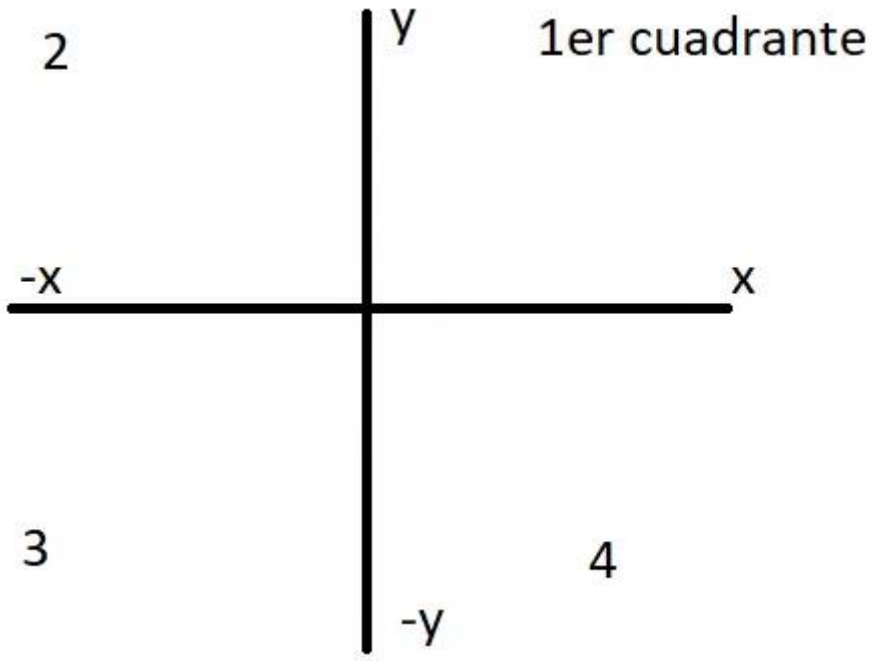
```
Ingrese el primer numero:3
Ingrese el segundo numero:55
Ingrese el tercer numero:66
alguno de los numeros es menor a diez
```

Capítulo 28.- Condiciones compuestas con operadores lógicos – 7

Problema:

Escribir un programa que pida ingresar la coordenada de un punto en el plano, es decir dos valores enteros x e y (distritos de cero).

Posteriormente imprimir en pantalla en que cuadrante se ubica dicho punto. (1º Cuadrante si <>0 Y y>0, 2º Cuadrante: x<0 Y y>0, etc.



Vamos a programar:

*Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int x, y;
11
12         System.out.print("Indique la coordenada x:");
13         x=teclado.nextInt();
14         System.out.print("Indique la coordenada y:");
15         y=teclado.nextInt();
16
17         System.out.print("Esta coordenada pertenece al ");
18         if(x>0 && y>0) {
19             System.out.print("primer cuadrante");
20         }else {
21             if(x<0 && y>0) {
22                 System.out.print("segundo cuadrante");
23             }else {
```

```
24                  if(x<0 && y<0) {
25                      System.out.print("tercer cuadrante");
26                  }else {
27                      if(x>0 && y<0) {
28                          System.out.print("cuarto Cuadrante");
29                      }else {
30                          System.out.print("eje");
31                      }
32                  }
33              }
34          }
35
36      }
37
38  }
```

Vamos a ejecutar:

Si le damos a x un valor positivo y a y un valor positivo:

```
Indique la coordenada x:5
Indique la coordenada y:3
Esta coordenada pertenece al primer cuadrante
```

Si le damos a x un valor negativo y a y un valor positiv0:

```
Indique la coordenada x:-4
Indique la coordenada y:7
Esta coordenada pertenece al segundo cuadrante
```

Si le damos a x un valor negativo y a y un valor negativo:

```
Indique la coordenada x:-4
Indique la coordenada y:-3
Esta coordenada pertenece al tercer cuadrante
```

Si le damos a x un valor positivo y a y un valor negativo:

```
Indique la coordenada x:10
Indique la coordenada y:-13
Esta coordenada pertenece al cuarto Cuadrante
```

Si le damos a x un valor de 0 o a y:

```
Indique la coordenada x:0
Indique la coordenada y:4
Esta coordenada pertenece al eje
```

Capítulo 29.- Condiciones compuestas con operadores lógicos – 8

Problema:

De un operario se conoce su sueldo y los años de antigüedad. Se pide confeccionar un programa que lea los datos de entrada e informe:

a) Si el sueldo es inferior a 500 y su antigüedad es igual o superior a 10 años, otorgarle un aumento del 20%, mostrar el sueldo a pagar.
b) Si el sueldo en inferior a 500 pero su antigüedad es menor a 10 años, otorgarle un aumento de 5%.
c) Si el sueldo es mayor o igual a 500 mostrar el sueldo en pantalla sin cambios.

Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         float sueldo;
11         int antiguedad;
12
13         System.out.print("Indique el sueldo:");
14         sueldo=teclado.nextFloat();
15         System.out.print("Indique la antiguedad:");
16         antiguedad=teclado.nextInt();
17         if(sueldo<500 && antiguedad>=10) {
18             System.out.print("El total sueldo a precibir es de ");
19             System.out.print(sueldo+(sueldo*0.20f));
20         }else {
21             if(sueldo<500 && antiguedad<10) {
22                 System.out.print("El total sueldo a precibir es de ");
23                 System.out.print(sueldo+(sueldo*0.05f));
24             }else {
25                 System.out.print("El total sueldo a precibir es de ");
26                 System.out.print(sueldo);
27             }
28         }
29     }
30 }
```

Vamos a ejecutar introduciendo un sueldo de 400 y una antigüedad de 20 años:

```
Indique el sueldo:400
Indique la antiguedad:20
El total sueldo a precibir es de 480.0
```

Vamos a ejecutar de nuevo introduciendo un sueldo de 400 y una antigüedad de 4 años:

```
Indique el sueldo:400
Indique la antiguedad:5
El total sueldo a precibir es de 420.0
```

Volvemos a ejecutar de nuevo introduciendo un sueldo de 1200 y una antigüedad de 23 años.

```
Indique el sueldo:1200
Indique la antiguedad:23
El total sueldo a precibir es de 1200.0
```

Capítulo 30.- Condiciones compuestas con operadores lógicos – 9

Problema:

Escribir un programa en el cual: dada una lista de tres valores numéricos distintos se calcule e informe su rango de variación (debe mostrar el mayor y el menor de ellos).

Vamos a programar:

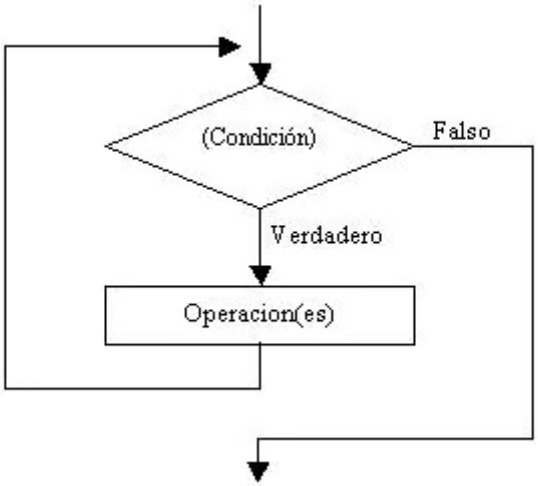
*Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int num1, num2, num3;
11
12         System.out.print("Ingrese el primer numero:");
13         num1=teclado.nextInt();
14         System.out.print("Ingrese el segundo numero:");
15         num2=teclado.nextInt();
16         System.out.print("Ingrese el tercer numero:");
17         num3=teclado.nextInt();
18         if(num1<num2 && num1<num3) {
19             System.out.print(num1);
20         }else {
21             if (num2<num3) {
22                 System.out.print(num2);
23             }else {
24                 System.out.print(num3);
25             }
26
27         System.out.print(" - ");
28         }
29         if(num1>num2 && num1>num3) {
30             System.out.print(num1);
31         }else {
32             if (num2>num3) {
33                 System.out.print(num2);
34             }else {
35                 System.out.print(num3);
36             }
37         }
38
39     }
40 }
```

Vamos a ejecutar introduciendo los valores 88, 10 y 45:

```
Ingrese el primer numero:88
Ingrese el segundo numero:10
Ingrese el tercer numero:45
10 - 88
```

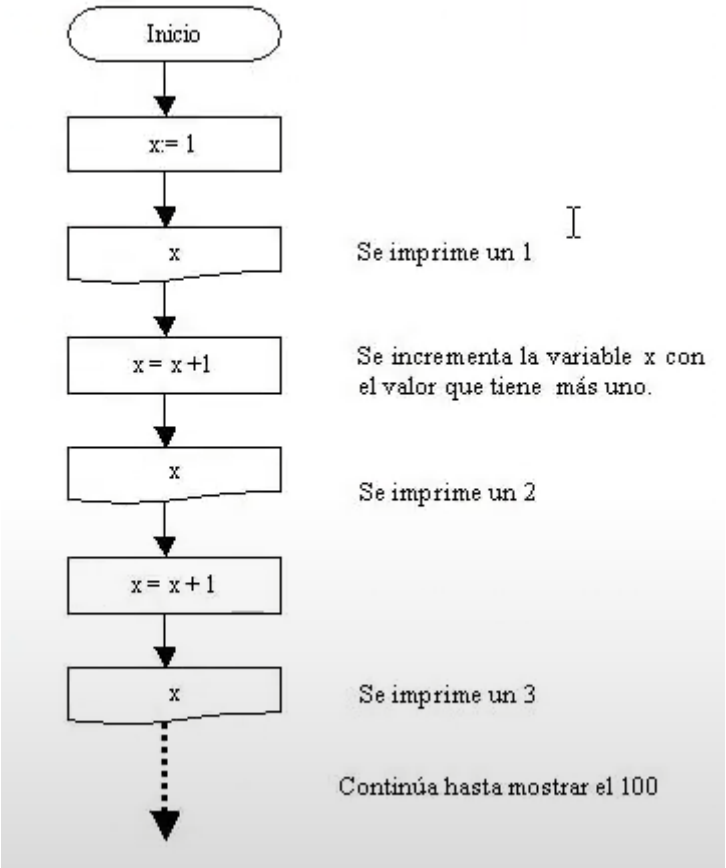
Capítulo 31.- Estructura repetitiva while – 1



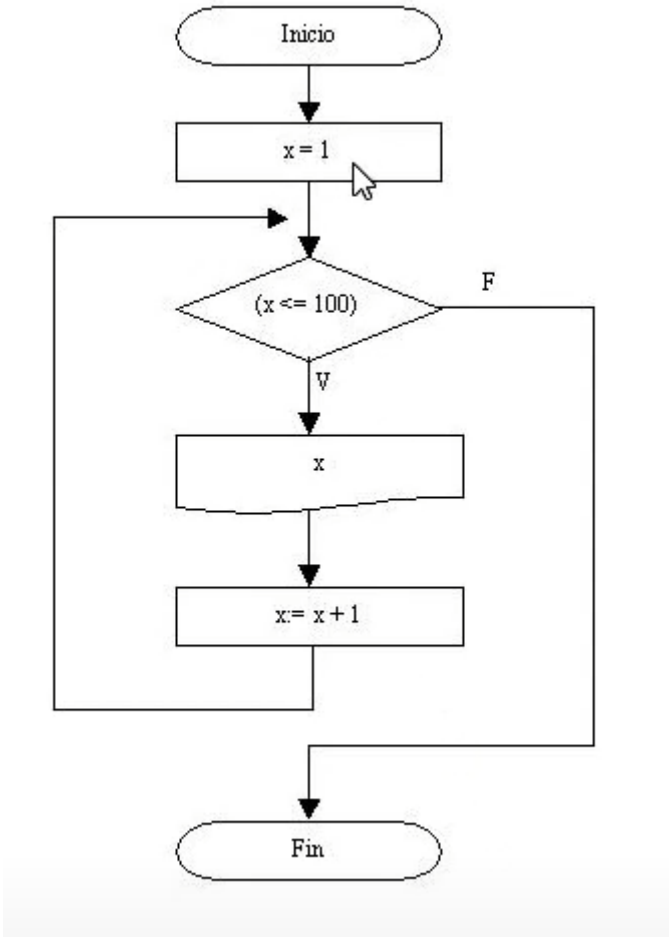
Problema:

Realizar un programa que imprima en pantalla los números del 1 al 100.

- Sin estructura repetitiva:



- Con estructura repetitiva:



Vamos a programar:

*Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8      Scanner teclado=new Scanner(System.in);
9      int x=1;
10     while(x<=100) {
11         System.out.print(x);
12         System.out.print(" - ");
13         if(x%10==0) {
14             System.out.println();
15         }
16         x=x+1;
17     }
18     }
19 }
```

Vamos a ejecutar:

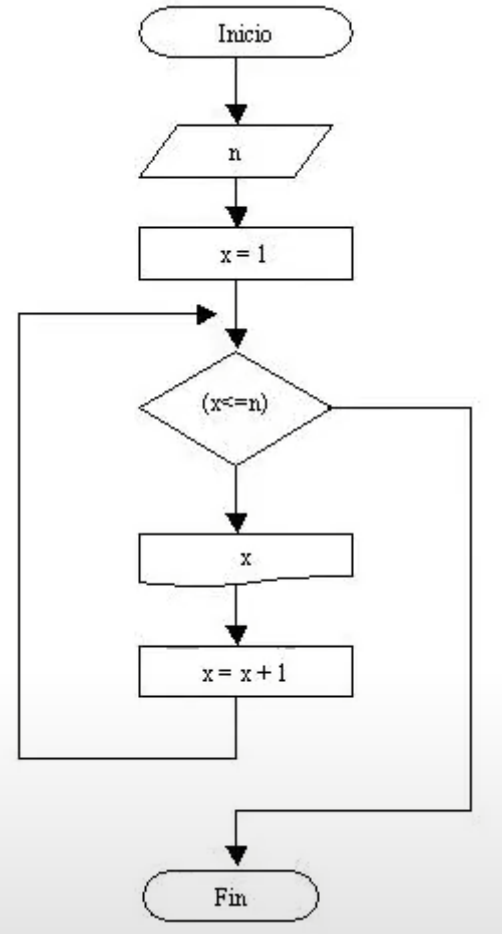
```
1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 -
11 - 12 - 13 - 14 - 15 - 16 - 17 - 18 - 19 - 20 -
21 - 22 - 23 - 24 - 25 - 26 - 27 - 28 - 29 - 30 -
31 - 32 - 33 - 34 - 35 - 36 - 37 - 38 - 39 - 40 -
41 - 42 - 43 - 44 - 45 - 46 - 47 - 48 - 49 - 50 -
51 - 52 - 53 - 54 - 55 - 56 - 57 - 58 - 59 - 60 -
61 - 62 - 63 - 64 - 65 - 66 - 67 - 68 - 69 - 70 -
71 - 72 - 73 - 74 - 75 - 76 - 77 - 78 - 79 - 80 -
81 - 82 - 83 - 84 - 85 - 86 - 87 - 88 - 89 - 90 -
91 - 92 - 93 - 94 - 95 - 96 - 97 - 98 - 99 - 100 -
```

Capítulo 32.- Estructura repetitiva while – 2

Problema:

Escribir un programa que solicite la carga de un valor positivo y nos muestre desde el 1 hasta el valor ingresado de uno en uno.

Ejemplo: Si ingresamos 30 se debe mostrar en pantalla los números del 1 al 30.



Vamos a programar:

*Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int num;
11
12         System.out.print("Ingrese un numero:");
13         num=teclado.nextInt();
14
```

```
15          int x=1;
16          while (x<=num) {
17              System.out.println(x);
18              x=x+1;
19          }
20      }
21  }
```

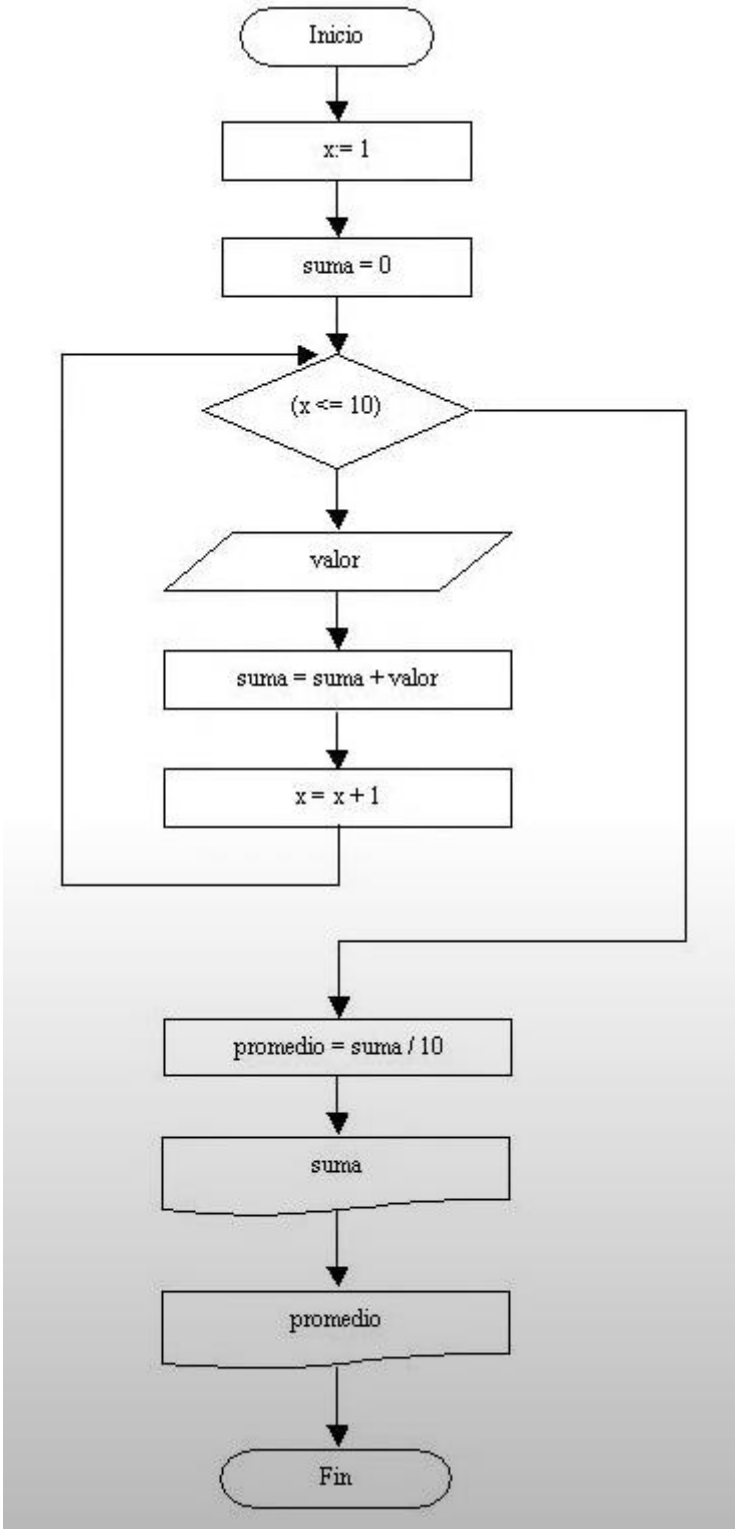
Vamos a ejecutar introduciendo el valor 7.

```
Ingrese un numero:7
1
2
3
4
5
6
7
```

Capítulo 33.- Estructura repetitiva while – 3

Problema:

Desarrollar un programa que permita la carga de 10 valores por teclado y nos muestre posteriormente la suma de los valores ingresados y su promedio.



Vamos a programar:

Clase1.java

```java
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);

        int x, numero, suma;
        float promedio;

        x=1;
        suma=0;

        while (x<=10) {
            System.out.print("Ingrese un numero:");
            numero=teclado.nextInt();
            suma=suma+numero;
            x=x+1;
        }
        promedio=(float)suma/10;

        System.out.print("La suma es ");
        System.out.println(suma);
        System.out.print("El promedio es ");
        System.out.print(promedio);
    }
```

Vamos a ejecutar:

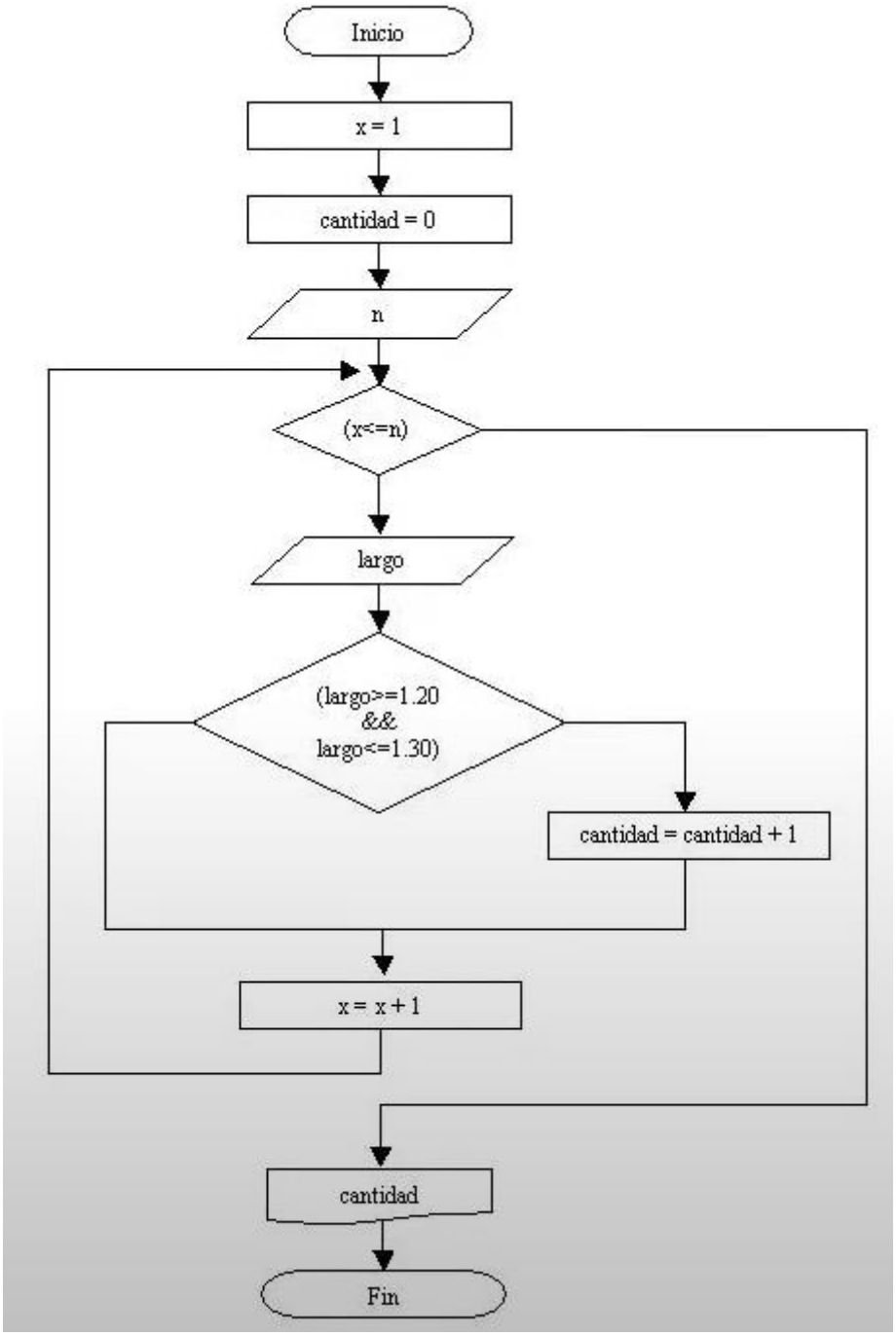
```
Ingrese un numero:4
Ingrese un numero:5
Ingrese un numero:4
Ingrese un numero:5
Ingrese un numero:6
Ingrese un numero:5
Ingrese un numero:6
Ingrese un numero:7
Ingrese un numero:6
Ingrese un numero:7
La suma es 55
El promedio es 5.5
```

Capítulo 34.- Estructura repetitiva while – 4

Problema:

Una planta de fabrica de perfiles de hierro posee un lote de n piezas.

Confeccionar un programa que pida ingresar por teclado la cantidad de piezas a procesar y luego ingrese la longitud de cada perfil; sabiendo que la pieza cuya longitud esté comprendida en el rango de 1,20 y 1,30 son aptas. Imprimir por pantalla la cantidad de aptas que hay en el lote.



Vamos a programar:

Clase1.java

```java
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);
        int piezas, x, aptas;
        float longitud;

        System.out.print("Ingrese la cantidad de piezas:");
        piezas=teclado.nextInt();

        x=1;
        aptas=0;
        while(x<=piezas) {
            System.out.print("Ingrese sus dimensiones:");
            longitud=teclado.nextFloat();
            if(longitud>=1.20 && longitud<=1.30) {
                aptas=aptas+1;
            }
            x=x+1;
        }
        System.out.print("En este lote hay ");
        System.out.print(aptas);
        System.out.print(" aptas.");
    }
}
```

Vamos a ejecutar:

```
Ingrese la cantidad de piezas:5
Ingrese sus dimensiones:1,25
Ingrese sus dimensiones:1,23
Ingrese sus dimensiones:1,45
Ingrese sus dimensiones:1,34
Ingrese sus dimensiones:1,1
En este lote hay 2 aptas.
```

Capítulo 35.- Estructura repetitiva white – 5

Problema:

Escribir un programa que solicite ingresar 10 notas de alumnos y nos informe cuántos tienen notas mayores o iguales a 7 y cuántos menores.

Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         float nota;
11         int x, aptos, noaptos;
12
13         x=1;
14         aptos=0;
15         noaptos=0;
16
17         while(x<=10) {
18             System.out.print("Ingrese una nota:");
19             nota=teclado.nextFloat();
20             if(nota>=7) {
21                 aptos=aptos+1;
22             }else {
23                 noaptos=noaptos+1;
24             }
25             x=x+1;
26         }
27         System.out.print("Calificciones mayores o iguales a 7:");
28         System.out.println(aptos);
29         System.out.print("Calificaciones menores a 7:");
30         System.out.print(noaptos);
31     }
32 }
```

Si ejecutamos este será el resultado:

```
Ingrese una nota:5
Ingrese una nota:3
Ingrese una nota:5
Ingrese una nota:8
Ingrese una nota:9
Ingrese una nota:8
Ingrese una nota:9
Ingrese una nota:10
Ingrese una nota:9
Ingrese una nota:9
Calificciones mayores o iguales a 7:7
Calificaciones menores a 7:3
```

Capítulo 36.- Estructura repetitiva while – 6

Problema:

Se ingresan un conjunto de n alturas de personas por teclado (n se ingresan por teclado). Mostrar la altura promedio de las personas.

Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         float altura, suma, promedio;
11         int n, x;
12
13         System.out.print("Ingrese el numero de alturas a introducir:");
14         n=teclado.nextInt();
15
16         x=1;
17         suma=0;
18         while (x<=n) {
19             System.out.print("Ingrese la algura:");
20             altura=teclado.nextFloat();
21             suma=suma+altura;
22             x=x+1;
23         }
24         promedio=suma/n;
25         System.out.print("El promedio de alturas es ");
26         System.out.print(promedio);
27     }
28 }
```

Vamos a ejecutar:

```
Ingrese el numero de alturas a introducir:5
Ingrese la algura:1,45
Ingrese la algura:1,95
Ingrese la algura:1,65
Ingrese la algura:1,67
Ingrese la algura:1,78
El promedio de alturas es 1.7
```

Capítulo 37.- Estructura repetitiva while – 7

Problema:

En una empresa trabajan n (n ingresar por teclado) empleados cuyos sueldos oscilan entre 100 y 500, realizar un programa que lea los sueldos que cobra cada empleado e informe cuántos empleados cobran entre 100 y 300 y cuantos cobran más de 300. Además el programa deberá informar el importe que gasta la empresa en sueldos al personal.
Vamos a programar:

Clase1.java

```
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);

        int n, sueldo, sueldobajo, sueldoalto, totalsueldo, x;
        System.out.print("Cuantos empleados tiene la empresa:");
        n=teclado.nextInt();
        x=1;
        sueldobajo=0;
        sueldoalto=0;
        totalsueldo=0;

        while(x<=n) {
            System.out.print("Ingrese el sueldo:");
            sueldo=teclado.nextInt();
            totalsueldo=totalsueldo+sueldo;
            if (sueldo>=100 && sueldo<=300) {
                sueldobajo=sueldobajo+1;
            }else {
                sueldoalto=sueldoalto+1;
            }
            x=x+1;
        }
        System.out.print("Empleados que cobran entre 100 y 300 son:");
        System.out.println(sueldobajo);
        System.out.print("Empleados que cobran mas de 300 son:");
        System.out.println(sueldoalto);
        System.out.print("El gasto en suldos de personal es de ");
        System.out.print(totalsueldo);
    }
}
```

Vamos a ejecutar:

```
Cuantos empleados tiene la empresa:4
Ingrese el sueldo:450
Ingrese el sueldo:350
Ingrese el sueldo:100
Ingrese el sueldo:150
Empleados que cobran entre 100 y 300 son:2
Empleados que cobran mas de 300 son:2
El gasto en suldos de personal es de 1050
```

Capítulo 38.- Estructura repetitiva while – 8

Problema:

Realizar un programa que imprima 25 términos de la serie 11 – 22 – 33 – 44, etc. (No se ingresan por teclado).

Vamos a programar:

*Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int x;
11
12         x=1;
13
14         while(x<=25) {
15             System.out.print(x*11);
16             System.out.print(" - ");
17             x=x+1;
18         }
19     }
20 }
```

Vamos a ejecutar:

```
11 - 22 - 33 - 44 - 55 - 66 - 77 - 88 - 99 - 110 - 121 - 132 - 143 - 154 -
165 - 176 - 187 - 198 - 209 - 220 - 231 - 242 - 253 - 264 - 275 -
```

Capítulo 39.- Estructura repetitiva while – 9

Problema:

Mostrar los múltiplos de 8 hasta el valor 500. Debe aparecer en pantalla 8- 16 – 24, etc.

Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int x;
11
12         x=8;
13
14         while(x<=500) {
15             System.out.print(x);
16             System.out.print(" - ");
17             x=x+8;
18         }
19     }
20 }
```

Si ejecutamos este será el resultado:

```
8 - 16 - 24 - 32 - 40 - 48 - 56 - 64 - 72 - 80 - 88 - 96 - 104 - 112 - 120 -
128 - 136 - 144 - 152 - 160 - 168 - 176 - 184 - 192 - 200 - 208 - 216 - 224 -
232 - 240 - 248 - 256 - 264 - 272 - 280 - 288 - 296 - 304 - 312 - 320 - 328 -
336 - 344 - 352 - 360 - 368 - 376 - 384 - 392 - 400 - 408 - 416 - 424 - 432 -
440 - 448 - 456 - 464 - 472 - 480 - 488 - 496 -
```

Capítulo 40.- Estructura repetitiva while – 10

Problema:

Realizar un programa que permita cargar dos listas de 15 valores cada una. Informar con un mensaje cual de las dos listas tiene un valor acumulado mayor (mensaje "Lista 1 mayor", "Lista 2 mayor", "Listas iguales")

Tener en cuenta que pueden haber dos o más estructuras repetitivas en un algoritmo.

Vamos a programar:

Clase1.java

```
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);

        int lista1, lista2;
        int x, numero;
        x=1;
        lista1=0;
        lista2=0;
        System.out.println("Lista 1");
        while (x<=15) {
            System.out.print("Ingrese un numero de la primera lista:");
            numero=teclado.nextInt();
            lista1=lista1+numero;
            x=x+1;
        }
        x=1;
        System.out.println("Lista 2");
        while (x<=15) {
            System.out.print("Ingrese un numero de la segunda lista:");
            numero=teclado.nextInt();
            lista2=lista2+numero;
            x=x+1;
        }
        if (lista1>lista2) {
            System.out.print("Lista 1 mayor");
        }else
            if (lista2>lista1) {
                System.out.print("Lista 2 mayor");
            }else {
                System.out.print("Listas iguales");
            }
    }
}
```

Ahora puedes ejecutar te pedirá 15 valores para la primera lista u después 15 valores más para la segunda lista, y nos dirá que lista es mayor o si son iguales.

Capítulo 41.- Estructura repetitiva while – 11

Problema:

Desarrollar un programa que permita cargar n números enteros y luego nos informe cuántos valores fueron pares y cuántos impares.
Emplear el operador % en la condición de la estructura condicional:
if (valor%2==0) //Si el if da verdadero luego es par.
11%2 es 1 10%2 es 0

Vamos a programar:

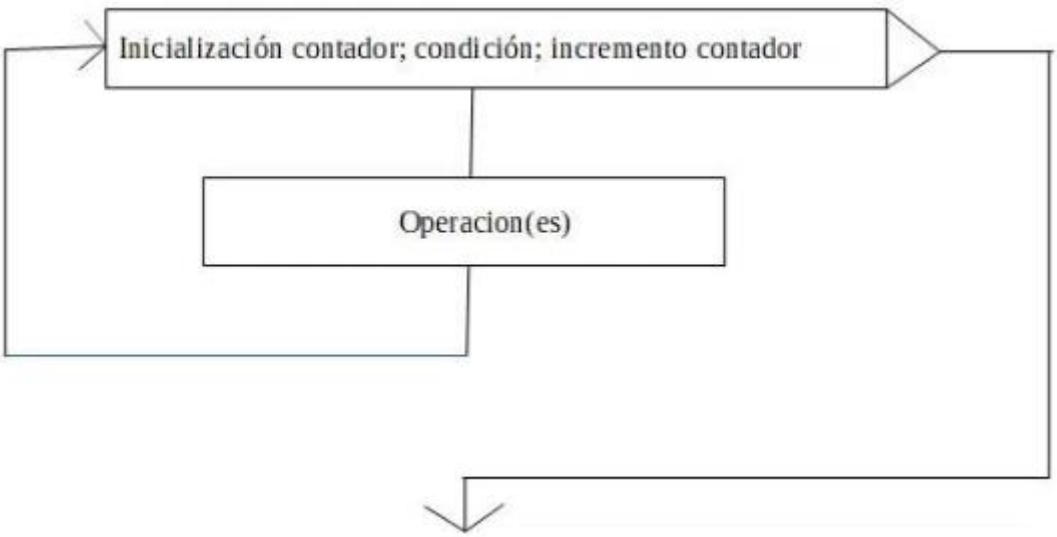
Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int n, x, numero, pares, impares;
11         x=1;
12         pares=0;
13         impares=0;
14         System.out.print("Ingrese el número de enteros:");
15         n=teclado.nextInt();
16
17         while (x<=n) {
18             System.out.print("Ingrese un numero:");
19             numero=teclado.nextInt();
20             if(numero%2==0) {
21                 pares=pares+1;
22             }else {
23                 impares=impares+1;
24             }
25             x=x+1;
26         }
27         System.out.print("La cantidad de pares: ");
28         System.out.println(pares);
29         System.out.print("La cantidad de impares: ");
30         System.out.print(impares);
31     }
32 }
```

Vamos a ejecutar:

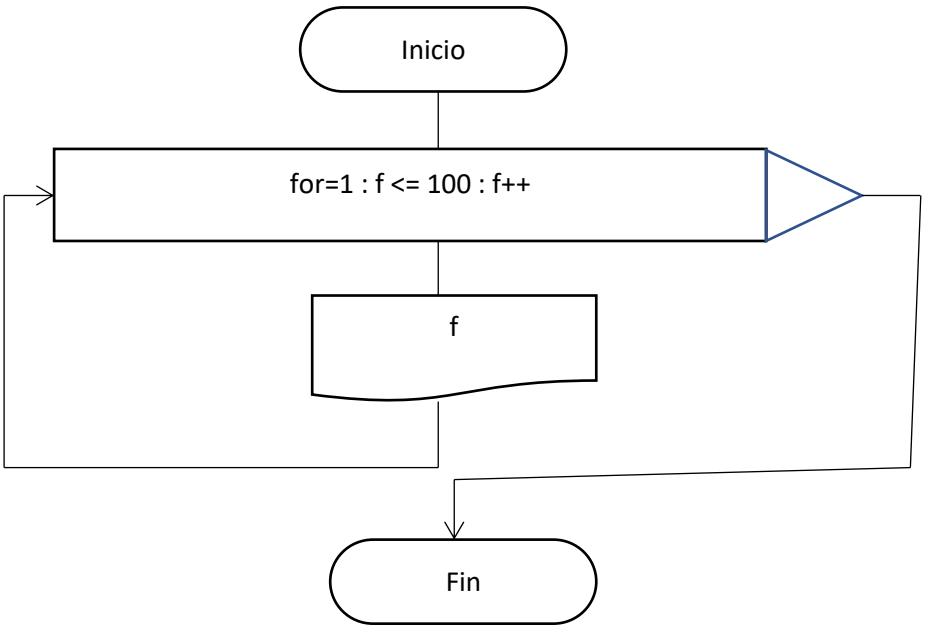
```
Ingrese el n□mero de enteros:5
Ingrese un numero:1
Ingrese un numero:2
Ingrese un numero:3
Ingrese un numero:4
Ingrese un numero:5
La cantidad de pares: 2
La cantidad de impares: 3
```

Capítulo 42.- Estructura repetitiva for – 1



Problema:

Realizar un programa que imprima en pantalla los números del 1 al 100.



Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  public class Clase1 {
4
5      public static void main(String[] args) {
6          for(int f=1;f<=100;f++) {
7              System.out.print(f+" - ");
8          }
9          System.out.println();
10         int x=1;
11         while (x<=100) {
12             System.out.print(x+" - ");
13             x++;
14         }
15     }
16 }
```

Este será el resultado:

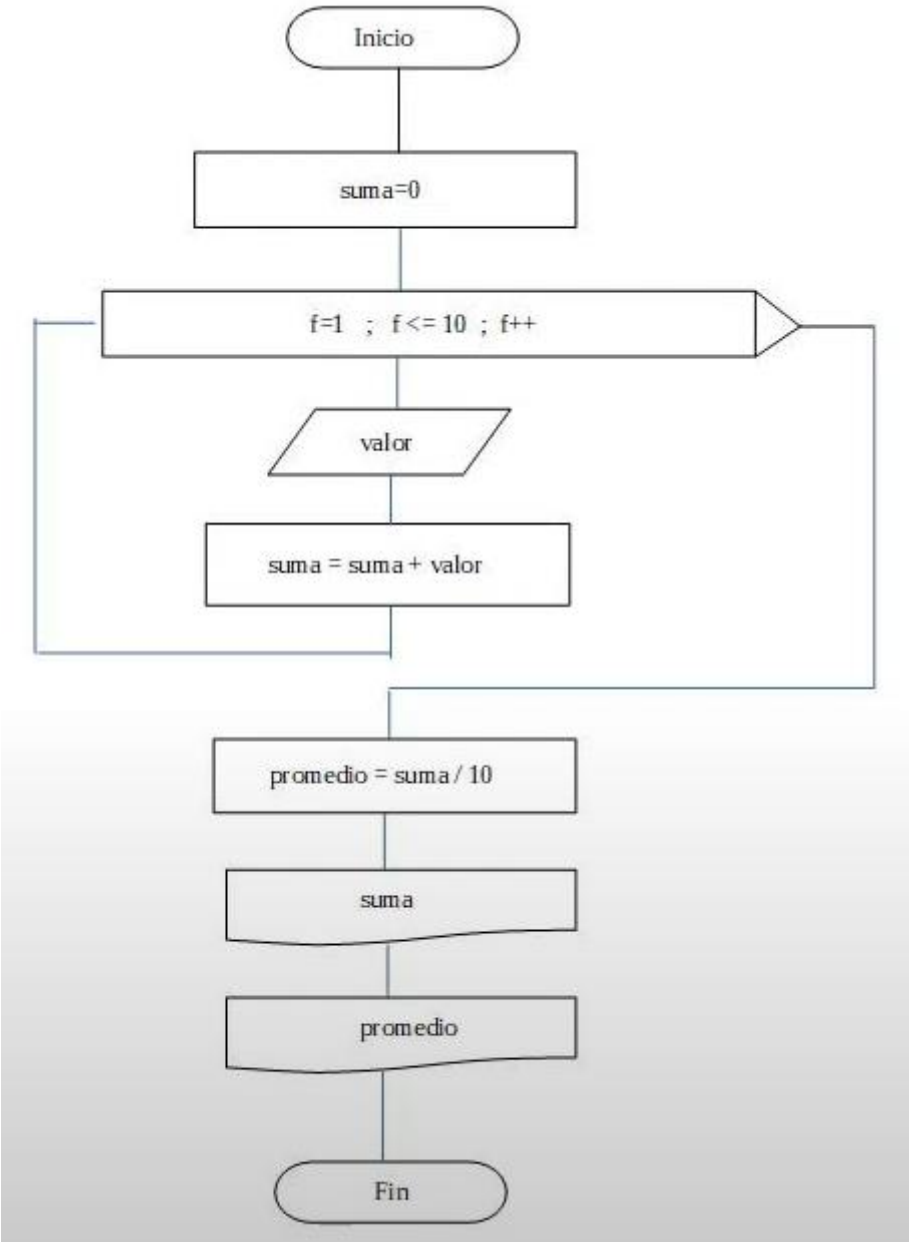
```
1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 12 - 13 - 14 - 15 - 16 - 17 -
18 - 19 - 20 - 21 - 22 - 23 - 24 - 25 - 26 - 27 - 28 - 29 - 30 - 31 - 32 - 33
- 34 - 35 - 36 - 37 - 38 - 39 - 40 - 41 - 42 - 43 - 44 - 45 - 46 - 47 - 48 -
49 - 50 - 51 - 52 - 53 - 54 - 55 - 56 - 57 - 58 - 59 - 60 - 61 - 62 - 63 - 64
- 65 - 66 - 67 - 68 - 69 - 70 - 71 - 72 - 73 - 74 - 75 - 76 - 77 - 78 - 79 -
80 - 81 - 82 - 83 - 84 - 85 - 86 - 87 - 88 - 89 - 90 - 91 - 92 - 93 - 94 - 95
- 96 - 97 - 98 - 99 - 100 -
1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 12 - 13 - 14 - 15 - 16 - 17 -
18 - 19 - 20 - 21 - 22 - 23 - 24 - 25 - 26 - 27 - 28 - 29 - 30 - 31 - 32 - 33
- 34 - 35 - 36 - 37 - 38 - 39 - 40 - 41 - 42 - 43 - 44 - 45 - 46 - 47 - 48 -
49 - 50 - 51 - 52 - 53 - 54 - 55 - 56 - 57 - 58 - 59 - 60 - 61 - 62 - 63 - 64
- 65 - 66 - 67 - 68 - 69 - 70 - 71 - 72 - 73 - 74 - 75 - 76 - 77 - 78 - 79 -
80 - 81 - 82 - 83 - 84 - 85 - 86 - 87 - 88 - 89 - 90 - 91 - 92 - 93 - 94 - 95
- 96 - 97 - 98 - 99 - 100 -
```

El primer recuadro es el ciclo for y en el segundo recuadro se obtiene el mismo resultado con el ciclo while.

Capítulo 43.- Estructura repetitiva for – 2

Problema:

Desarrollar un programa a que permitan la carga de 10 valores por teclado y nos muestre posteriormente la suma de los valores ingresados y su promedio.



Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
```

```
 9          int valor, suma, promedio;
10          suma=0;
11          for(int n=1; n<=10; n++) {
12              System.out.print("Ingrese un valor:");
13              valor=teclado.nextInt();
14              suma=suma+valor;
15          }
16          promedio=suma/10;
17          System.out.println("La suma de los valores es:"+suma);
18          System.out.print("El promedio es:"+promedio);
19      }
20  }
```

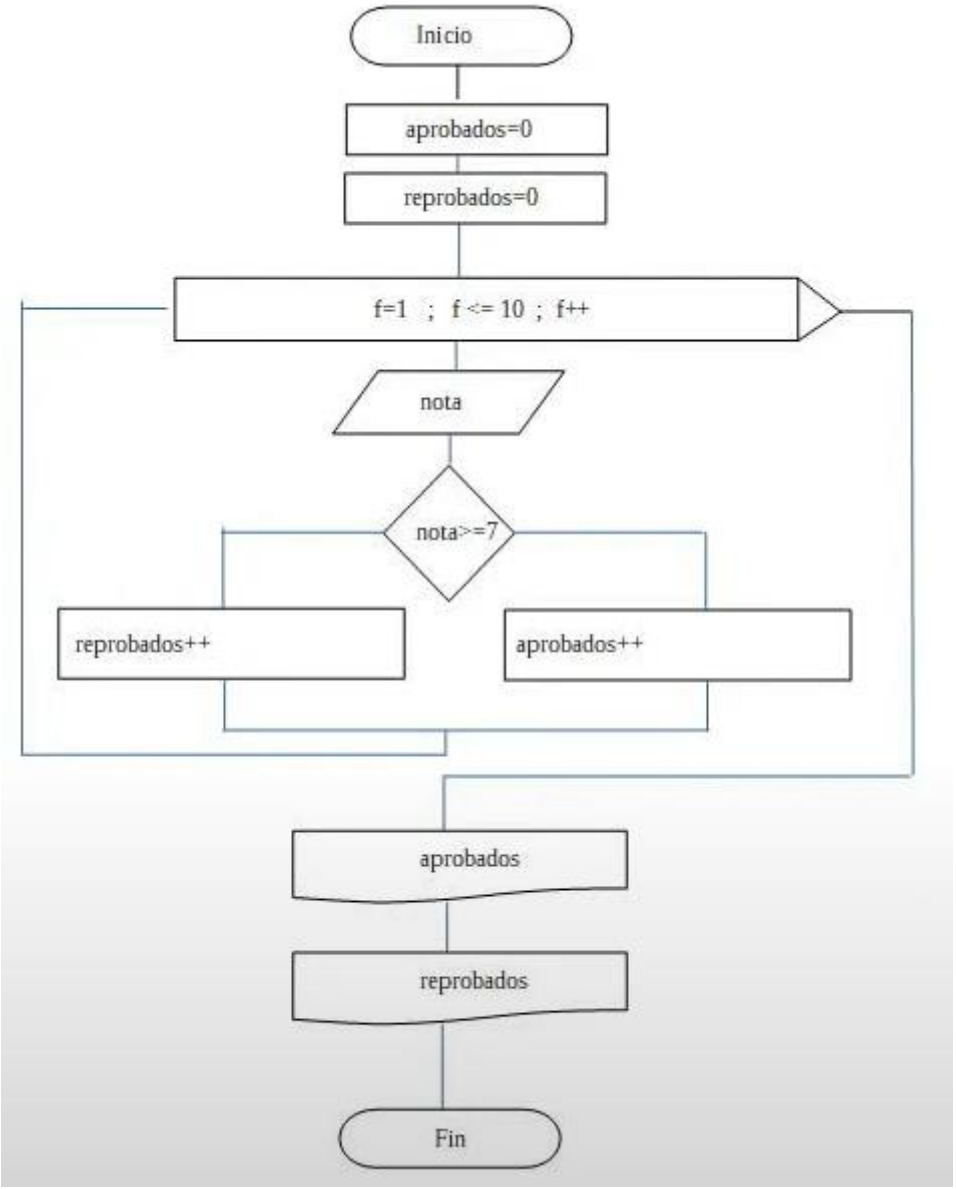
Si ejecutamos este será el resultado:

```
Ingrese un valor:1
Ingrese un valor:2
Ingrese un valor:3
Ingrese un valor:4
Ingrese un valor:5
Ingrese un valor:6
Ingrese un valor:7
Ingrese un valor:8
Ingrese un valor:9
Ingrese un valor:10
La suma de los valores es:55
El promedio es:5
```

Capítulo 44.- Estructura repetitiva for – 3

Problema:

Escribir un programa que lea 10 notas de alumnos y nos informe cuántos tienen notas mayores o iguales a 7 y cuántos menores.



Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
```

```
10          float nota;
11          int apto, noapto;
12
13          apto=0;
14          noapto=0;
15
16          for(int n=1; n<=10; n++) {
17              System.out.print("Ingrese una nota:");
18              nota=teclado.nextFloat();
19
20              if(nota>=7) {
21                  apto++;
22              }else {
23                  noapto++;
24              }
25          }
26          System.out.println("Alumnos que han aprobado:"+apto);
27          System.out.print("Alumnos que no han aprobado:"+noapto);
28      }
29  }
```

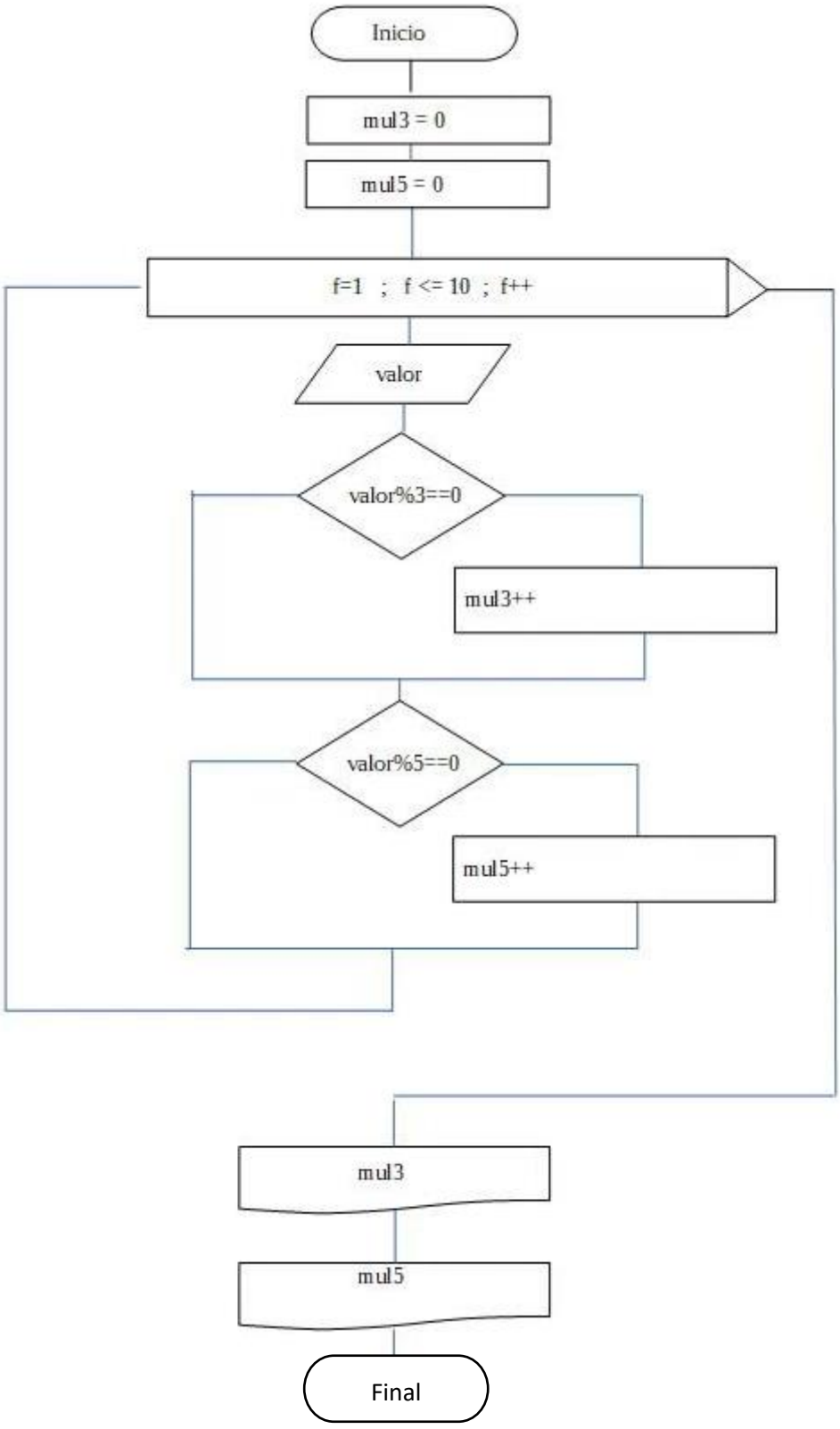
Vamos a ejecutar, este será el resultado:

```
Ingrese una nota:1
Ingrese una nota:2
Ingrese una nota:3
Ingrese una nota:4
Ingrese una nota:5
Ingrese una nota:6
Ingrese una nota:7
Ingrese una nota:8
Ingrese una nota:9
Ingrese una nota:10
Alumnos que han aprobado:4
Alumnos que no han aprobado:6
```

Capítulo 45.- Estructura repetitiva for – 4

Problema:

Escribir un programa que lea 10 números enteros y luego muestre cuántos valores ingresados fueron múltiplos de 3 y cuantos en 5. Debemos tener en cuenta que hay números que son múltiplos de 3 y de 5 a la vez.



Vamos a programar:

Clase1.java

```java
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);

        int numero, mul3, mul5;
        mul3=0;
        mul5=0;
        for (int n=1; n<=10; n++) {
            System.out.print("Ingrese un numero:");
            numero=teclado.nextInt();
            if(numero%3==0) {
                mul3++;
            }
            if(numero%5==0) {
                mul5++;
            }
        }
        System.out.println("Los numeros multiplos de 3 son "+mul3);
        System.out.print("Los numeros multiplos de 5 son "+mul5);
    }
}
```

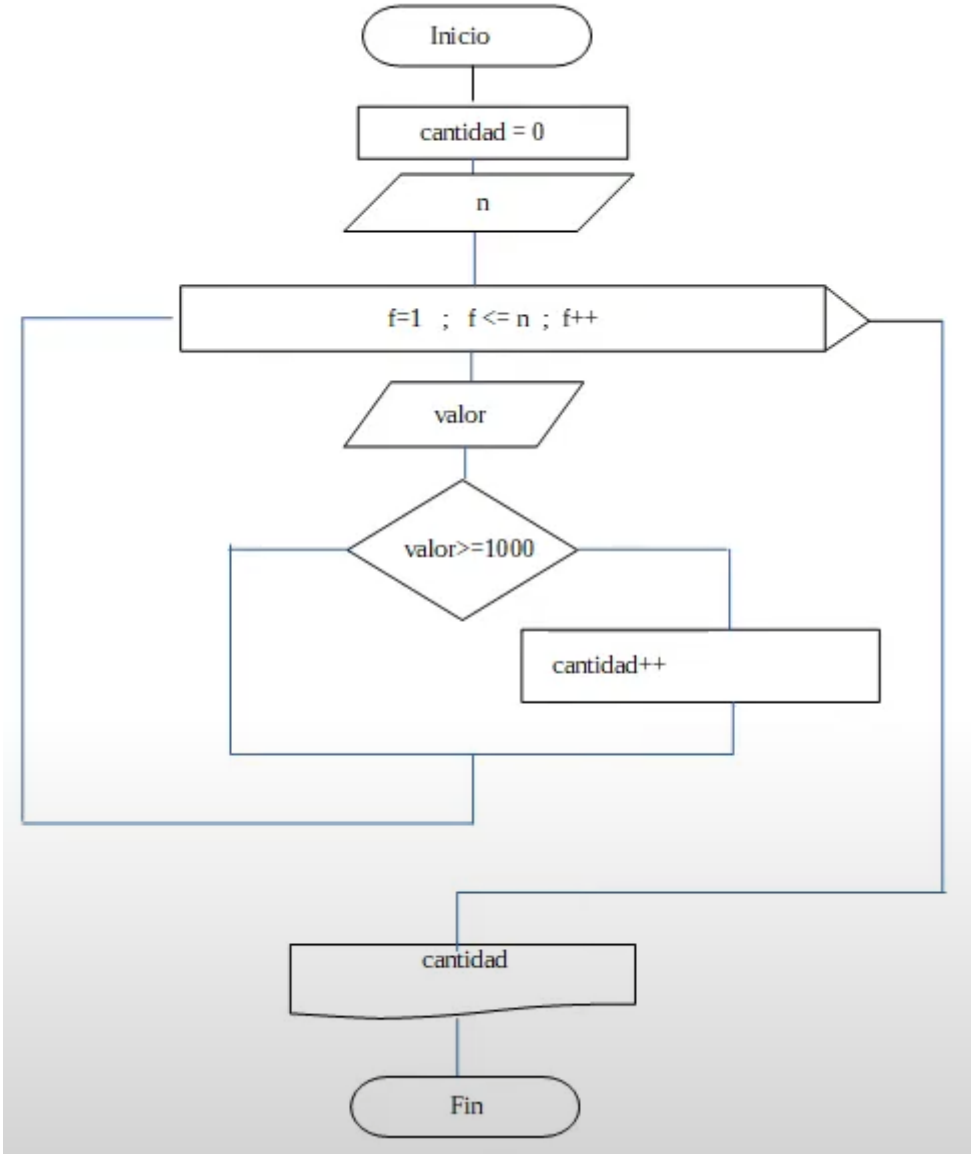
Vamos a ejecutar este será el resultado:

```
Ingrese un numero:1
Ingrese un numero:2
Ingrese un numero:3
Ingrese un numero:4
Ingrese un numero:5
Ingrese un numero:6
Ingrese un numero:7
Ingrese un numero:8
Ingrese un numero:9
Ingrese un numero:15
Los numeros multiplos de 3 son 4
Los numeros multiplos de 5 son 2
```

Capítulo 46.- Estructura repetitiva for – 5

Problema:

Escribir un programa que lea n números enteros (n se ingresa por teclado) y calcule la cantidad de valores mayores o iguales a 1000.



Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int n, valor, cantidad;
11         System.out.print("Cuantos numeros desea ingresar:");
12         n=teclado.nextInt();
13         cantidad=0;
```

```
14          for (int x=1; x<=n; x++) {
15              System.out.print("Ingrese un valor:");
16              valor=teclado.nextInt();
17              if(valor>=1000) {
18                  cantidad++;
19              }
20          }
21          System.out.print("La cantidad de valores igual o mayor a 1000 es "+cantidad);
22      }
23  }
```

Vamos a ejecutar y este será el resultado:

```
Cuantos numeros desea ingresar:5
Ingrese un valor:100
Ingrese un valor:200
Ingrese un valor:1200
Ingrese un valor:1300
Ingrese un valor:1500
La cantidad de valores igual o mayor a 1000 es 3
```

Capítulo 47.- Estructura repetitiva for – 6

Problema:

Confeccionar un programa que lea n pares de datos, cada par de datos corresponde a la medida de la base y la altura de un triángulo. El programa deberá informar:

a) De cada triángulo la media de su base, su altura y su superficie (la superficie se calcula: base * altura / 2).
b) La cantidad de triángulos cuya superficie es mayor a 12.

Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int n, base, altura, cantidad;
11         float superficie, media;
12
13         System.out.print("Cuantos pares de datos desea ingresar:");
14         n=teclado.nextInt();
15         cantidad=0;
16         for (int x=1; x<=n; x++) {
17             System.out.print("Ingrese la base del triangulo:");
18             base=teclado.nextInt();
19             System.out.print("Ingrese la altura del triangulo:");
20             altura=teclado.nextInt();
21             superficie=base*altura/2;
22             System.out.println("Superficie: "+superficie);
23             if (superficie>12) {
24                 cantidad++;
25             }
26         }
27         System.out.println("La cantidad de triangulos con una");
28         System.out.print("superficie mayor a 12 es de "+cantidad);
29     }
30 }
```

Vamos a ejecutar este será el resultado:

```
Cuantos pares de datos desea ingresar:5
Ingrese la base del triangulo:3
Ingrese la altura del triangulo:6
Superficie: 9.0
Ingrese la base del triangulo:10
Ingrese la altura del triangulo:20
Superficie: 100.0
Ingrese la base del triangulo:3
Ingrese la altura del triangulo:4
Superficie: 6.0
Ingrese la base del triangulo:89
Ingrese la altura del triangulo:10
Superficie: 445.0
```

```
Ingrese la base del triangulo:10
Ingrese la altura del triangulo:10
Superficie: 50.0
La cantidad de triangulos con una
superficie mayor a 12 es de 3
```

Capítulo 48.- Estructura repetitiva for – 7

Problema:

Desarrollar un programa que solicite la carga de 10 números e imprima la suma de los últimos 5 valores ingresados.

Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int numero, suma;
11         suma=0;
12
13         for (int x=1; x<=10; x++) {
14             System.out.print("Ingrese un numero:");
15             numero=teclado.nextInt();
16             if (x>5) {
17                 suma=suma+numero;
18             }
19         }
20         System.out.print("La suma de los 5 ultimos numeros es "+suma);
21     }
22 }
```

Vamos a ejecutar y este será el resultado:

```
Ingrese un numero:1
Ingrese un numero:1
Ingrese un numero:1
Ingrese un numero:1
Ingrese un numero:1
Ingrese un numero:1
Ingrese un numero:1
Ingrese un numero:1
Ingrese un numero:1
Ingrese un numero:1
La suma de los 5 ultimos numeros es 5
```

Otra forma de realizarlos es la siguiente:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int numero, suma;
```

```
11          suma=0;
12
13          for (int x=1; x<=5; x++) {
14              System.out.print("Ingrese un numero:");
15              numero=teclado.nextInt();
16
17          }
18          for (int x=1; x<=5; x++) {
19              System.out.print("Ingrese un numero:");
20              numero=teclado.nextInt();
21                  suma=suma+numero;
22          }
23          System.out.print("La suma de los 5 ultimos numeros es "+suma);
24      }
25  }
```

Podrás observar que el resultado será el mismo cuando ejecutemos.

Capítulo 49.- Estructura repetitiva for – 8

Problema:

Desarrollar un programa que muestre la tabla de multiplicar del 5 del 5 al 50).

vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  public class Clase1 {
4
5      public static void main(String[] args) {
6          int tabla5=5;
7          for(int f=1; f<=10; f++) {
8              System.out.println(f+" * 5 = "+tabla5*f);
9          }
10     }
11 }
```

Cuando ejecutemos este será el resultado:

```
1 * 5 = 5
2 * 5 = 10
3 * 5 = 15
4 * 5 = 20
5 * 5 = 25
6 * 5 = 30
7 * 5 = 35
8 * 5 = 40
9 * 5 = 45
10 * 5 = 50
```

Otra forma también de poderlo hacer es:

*Clase1.java

```
1  package paquete1;
2
3  public class Clase1 {
4
5      public static void main(String[] args) {
6          int tabla5=5;
7          for(int f=1; f<=10; f++) {
8              System.out.println(f+" * 5 = "+tabla5);
9              tabla5=tabla5+5;
10         }
11     }
12 }
```

Cuando ejecutemos el resultado será el mismo.

Capítulo 50.- Estructura repetitiva for – 9

Problema:

Confeccionar un programa que permita ingresar un valor de 1 al 10 y nos muestre la tabla de multiplicar del mismo (los primeros 13 términos)

Ejemplo: Si ingreso 3 deberá aparecer en pantalla los valores 3, 6, 9 hasta el 39.

Vamos a programar:

*Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9          int tabla;
10
11         System.out.print("Ingrese un numero entre el 1 y el 10 para mostrar la tabla: ");
12         tabla=teclado.nextInt();
13
14         if (tabla>=1 && tabla<=10) {
15             for(int f=1; f<=10; f++) {
16                 System.out.println(f+" * " + tabla + " = "+tabla*f);
17             }
18         }else {
19             System.out.println("El numero debe estar comprendido entre 1 y 10");
20         }
21     }
22 }
```

Cuando ejecutemos vamos a introducir por la tabla del 11.

```
Ingrese un numero entre el 1 y el 10 para mostrar la tabla: 11
El numero debe estar comprendido entre 1 y 10
```

Ejecutamos de nuevo y consultamos por la tabla del 9.

```
Ingrese un numero entre el 1 y el 10 para mostrar la tabla: 9
1 * 9 = 9
2 * 9 = 18
3 * 9 = 27
4 * 9 = 36
5 * 9 = 45
6 * 9 = 54
7 * 9 = 63
8 * 9 = 72
9 * 9 = 81
10 * 9 = 90
```

Capítulo 51.- Estructura repetitiva for – 10

Problema:

Realizar un programa que lea los datos de n triángulos, e informar:

a) De cada uno de ellos, qué tipo de triangulo es: equilátero (tres lados iguales), isósceles (dos lados iguales) o escaleno (ningún lado igual).
b) Cantidad de triángulos de cada tipo.
c) Tipo de triángulo que posee menor cantidad.

Vamos a programar:

```
package paquete1;

import java.util.Scanner;

public class Clase1 {

	public static void main(String[] args) {
		Scanner teclado=new Scanner(System.in);

		int n, lado1, lado2, lado3;
		int equilatero, isosceles, escaleno;
		equilatero=0;
		isosceles=0;
		escaleno=0;

	System.out.print("Cuantos triangulos va a ingresar:");
	n=teclado.nextInt();

	for(int x=1; x<=n; x++) {
		System.out.print("Ingrese el lado 1:");
		lado1=teclado.nextInt();
		System.out.print("Ingrese el lado 2:");
		lado2=teclado.nextInt();
		System.out.print("Ingrese el lado 3:");
		lado3=teclado.nextInt();
		if(lado1==lado2 && lado1==lado3) {
			equilatero++;
			System.out.println("Este triangulo es equilatero");
		}else {
			if(lado1==lado2 || lado2==lado3 || lado1==lado3) {
				isosceles++;
				System.out.println("Este triangulo es isosceles");
			}else {
				escaleno++;
				System.out.println("Este triangulo es escaleno");
			}
		}
	}
	System.out.println("Triangulos tipo equilatero "+equilatero);
	System.out.println("Triangulos tipo isoscelenes "+isosceles);
	System.out.println("Triangulos tipo escaleno "+escaleno);
	System.out.print("El menor tipo de triangulo es ");
	if(equilatero<isosceles && equilatero<escaleno) {
		System.out.println("equilatero");
	}else {
		if(isosceles<escaleno) {
```

```
                System.out.println("isosceles");
            }else {
                System.out.println("escaleno");
            }
        }
        }
}
```

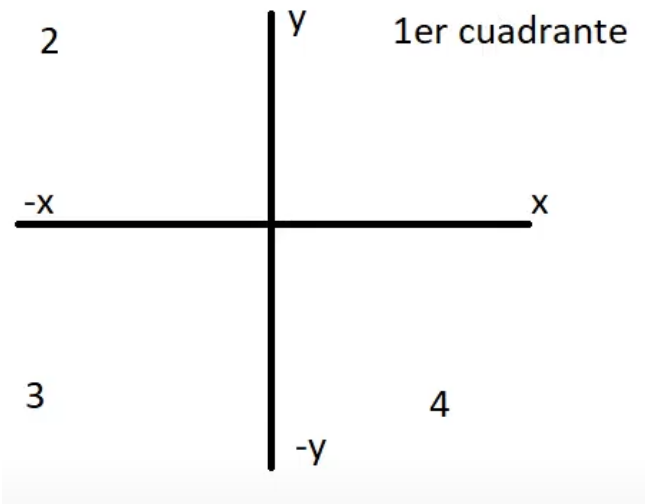
Vamos a ejecutar y veremos el siguiente resultado:

```
Cuantos triangulos va a ingresar:3
Ingrese el lado 1:5
Ingrese el lado 2:5
Ingrese el lado 3:5
Este triangulo es equilatero
Ingrese el lado 1:7
Ingrese el lado 2:7
Ingrese el lado 3:7
Este triangulo es equilatero
Ingrese el lado 1:2
Ingrese el lado 2:2
Ingrese el lado 3:1
Este triangulo es isosceles
Triangulos tipo equilatero 2
Triangulos tipo isoscelenes 1
Triangulos tipo escaleno 0
El menor tipo de triangulo es escaleno
```

Capítulo 52.- Estructura repetitiva for – 11

Problema:

Escribir un programa que pida ingresar coordenadas (x,y) que representan puntos en el plano. Informar cuántos puntos se han ingresado en el primer, segundo, tercer y cuarto cuadrante. Al comenzar el programa se pide que se ingrese la cantidad de puntos a procesar.



Vamos a programar:

```
package paquete1;

import java.util.Scanner;

public class Clase1 {

	public static void main(String[] args) {
		Scanner teclado=new Scanner(System.in);

		int n, x, y;
		int primero, segundo, tercero, cuarto;
		primero=0;
		segundo=0;
		tercero=0;
		cuarto=0;
		System.out.print("Cuantas cooredenadas va a ingresar:");
		n=teclado.nextInt();

		for (int z=1; z<=n; z++) {
			System.out.print("Ingrese la coordenada x:");
			x=teclado.nextInt();
			System.out.print("Ingrese la coordenada y:");
			y=teclado.nextInt();
			if (x>0 && y>0) {
				primero++;
			}else {
				if (x<0 && y>0) {
					segundo++;
				}else {
					if (x<0 && y<0) {
						tercero++;
```

```
                                    }else {
                                          if (x>0 && y<0) {
                                                cuarto++;
                                    }

                                    }
                              }
                        }
                  }
                  System.out.println("Cantidad de puntos ingresados en el primer
cuadrante: "+primero);
                  System.out.println("Cantidad de puntos ingresados en el segundo
cuadrante: "+segundo);
                  System.out.println("Cantidad de puntos ingresados en el tercer
cuadrante: "+tercero);
                  System.out.print("Cantidad de puntos ingresados en el cuarto
cuadrante: "+cuarto);
            }
}
```

Vamos a ejecutar:

```
Cuantas cooredenadas va a ingresar:4
Ingrese la coordenada x:2
Ingrese la coordenada y:3
Ingrese la coordenada x:-3
Ingrese la coordenada y:3
Ingrese la coordenada x:-2
Ingrese la coordenada y:-3
Ingrese la coordenada x:4
Ingrese la coordenada y:-4
Cantidad de puntos ingresados en el primer cuadrante: 1
Cantidad de puntos ingresados en el segundo cuadrante: 1
Cantidad de puntos ingresados en el tercer cuadrante: 1
Cantidad de puntos ingresados en el cuarto cuadrante: 1
```

Capítulo 53.- Estructura repetitiva for – 12

Problema:

Se realiza la carga de 10 valores enteros por teclado. Se desea conocer:

a) La cantidad de valores ingresados negativos.
b) La cantidad de valores ingresados positivos.
c) La cantidad de múltiplos de 15.
d) El valor acumulado de los números ingresados que son pares.

Vamos a programar:

```
package paquete1;

import java.util.Scanner;

public class Clase1 {

	public static void main(String[] args) {
		Scanner teclado=new Scanner(System.in);

		int valor, neg, pos, mult15,sumpares;
		neg=0;
		pos=0;
		mult15=0;
		sumpares=0;
		for (int x=1; x<=10; x++) {
			System.out.print("Ingrese un valor:");
			valor=teclado.nextInt();

			if(valor<0) {
				neg++;
			}
			if(valor>0) {
				pos++;
			}
			if(valor%15==0) {
				mult15++;
			}
			if(valor%2==0) {
				sumpares=sumpares+valor;
			}
		}
		System.out.println("La cantidad de valores negativos son de
"+neg);
		System.out.println("La cantidad de valroes positivos son de
"+pos);
		System.out.println("La cantidad de valores multiplos de 15 son
de "+mult15);
		System.out.print("La suma de los numero pares es de "+sumpares);
	}
}
```

Vamos a ejecutar:

```
Ingrese un valor:1
Ingrese un valor:2
Ingrese un valor:3
Ingrese un valor:-10
Ingrese un valor:15
Ingrese un valor:30
Ingrese un valor:1
Ingrese un valor:2
Ingrese un valor:3
Ingrese un valor:4
La cantidad de valores negativos son de 1
La cantidad de valroes positivos son de 9
La cantidad de valores multiplos de 15 son de 2
La suma de los numero pares es de 28
```

Capítulo 54.- Estructura repetitiva for – 13

Se cuenta con la siguiente información:
Las edades de 50 estudiantes del turno de mañana.
Las edades de 60 estudiantes de turno de tarde.
Las edades de 110 estudiantes del turno de noche.
Las edades de cada estudiante deben ingresar por teclado.

a) Obtener el promedio de las edades de cada turno (tres promedios)
b) Imprimir dichos promedios (promedio de cada turno)
c) Mostrar por pantalla un mensaje que indique cual de los tres turnos tiene un promedio de edades menor.

Vamos a programar:

```
package paquete1;

import java.util.Scanner;

public class Clase1 {

	public static void main(String[] args) {
		Scanner teclado=new Scanner(System.in);

		int ma,ta,no, edad;
		float proma, prota, prono;
		ma=0;
		ta=0;
		no=0;
		System.out.println("Turno de mañana");
		for(int x=1;x<=50;x++) {
			System.out.print("Ingrese la edad:");
			edad=teclado.nextInt();
			ma=ma+edad;
		}
		proma=((float)ma/50);
		System.out.println("El promedio del turno de mañana es "+proma);
		System.out.println("Turno de tarde");
		for(int y=1; y<=60; y++) {
			System.out.print("Ingrese la edad:");
			edad=teclado.nextInt();
			ta=ta+edad;
		}
		prota=((float)ta/60);
		System.out.println("El promedio del turno de mañana es "+prota);
		System.out.println("Turno de noche");
		for(int z=1; z<=110; z++) {
			System.out.print("Ingrese la edad:");
			edad=teclado.nextInt();
			no=no+edad;
		}
		prono=((float)no/110);
		System.out.print("El promedio del turno de noche es "+prono);

		System.out.println("El promedio de edades del turno de mañana es
"+proma);
		System.out.println("El promedio de edades del turno de tarde es
"+prota);
```

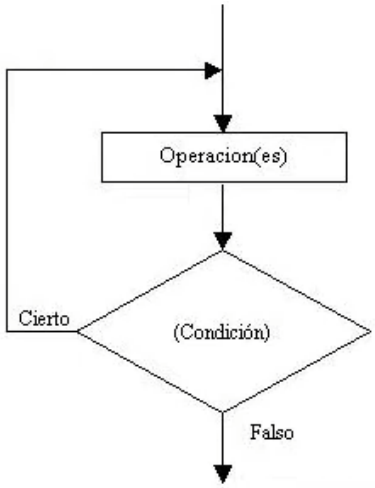
```
            System.out.println("El promedio de edades del turno de noche es
"+prono);

            if(proma<prota && proma<prono) {
                  System.out.println("El promedio de edades del turno mañana
es el menor");
            }else {
                  if(prota<prono) {
                        System.out.println("El promedio de edades del turno
tarde es el menor");
                  }else {
                        System.out.println("El promedio de edades del turno
noche es el menor");
                  }
            }
      }
}
```

Para probar el programa vamos a cambiar el número de alumnos de cada turno:
Mañana=5
Tarde=6
Noche=11

Vamos a ejecutar:

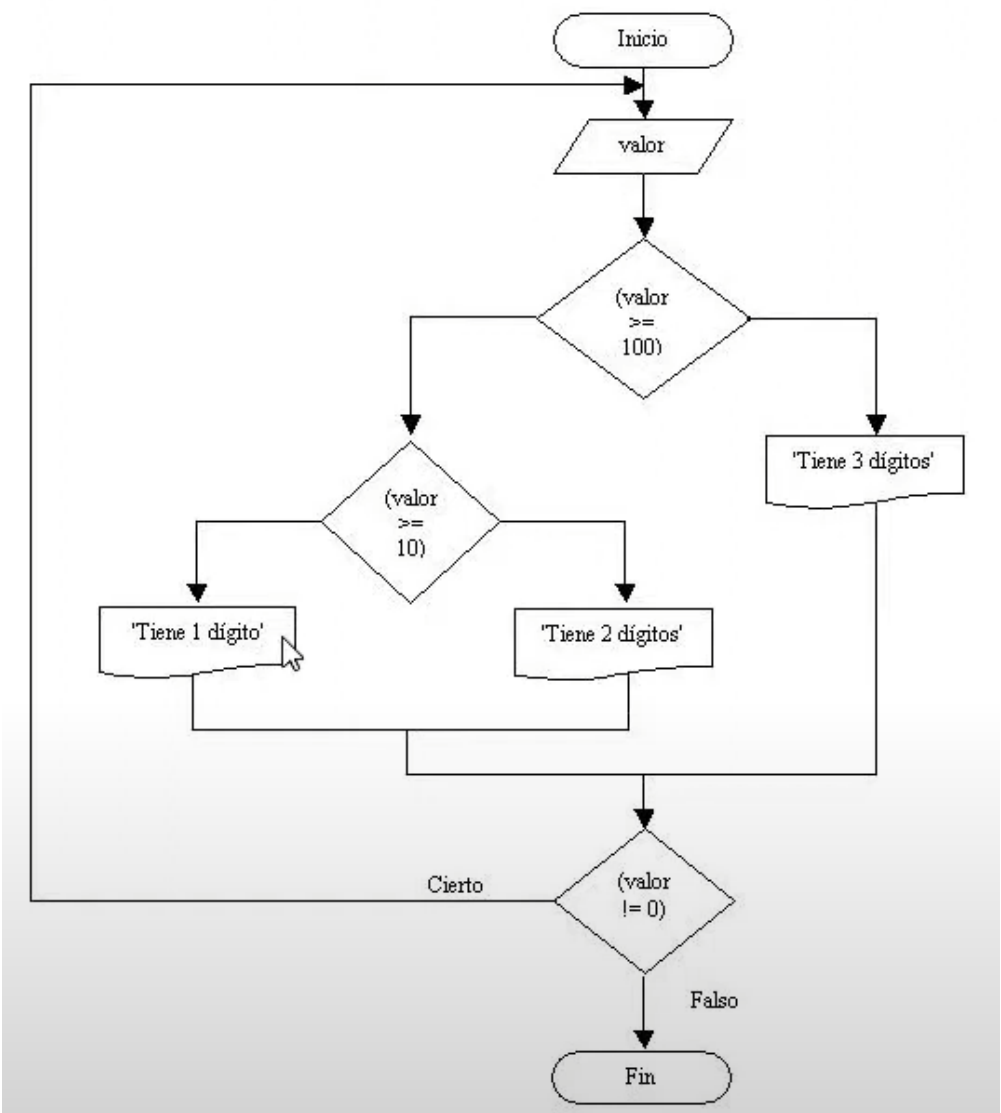
```
Turno de mañana
Ingrese la edad:12
Ingrese la edad:10
Ingrese la edad:13
Ingrese la edad:9
Ingrese la edad:13
El promedio del turno de mañaana es 11.4
Turno de tarde
Ingrese la edad:14
Ingrese la edad:13
Ingrese la edad:16
Ingrese la edad:15
Ingrese la edad:12
Ingrese la edad:13
El promedio del turno de maÃ±ana es 13.833333
Turno de noche
Ingrese la edad:18
Ingrese la edad:16
Ingrese la edad:14
Ingrese la edad:15
Ingrese la edad:21
Ingrese la edad:23
Ingrese la edad:14
Ingrese la edad:17
Ingrese la edad:14
Ingrese la edad:18
Ingrese la edad:19
El promedio del turno de noche es 17.0El promedio de edades del turno de maÃ±ana es 11.4
El promedio de edades del turno de tarde es 13.833333
El promedio de edades del turno de noche es 17.0
El promedio de edades del turno mañana es el menor
```

Capítulo 55.- Estructura repetitiva do while – 1



Problema:

Escribir un programa que solicite la carga de un número entre 0 y 999, y nos muestre un mensaje de cuántos dígitos tiene el mismo. Finalizar el programa cuando se cargue el valor 0.



Vamos a programar:

Clase1.java

```java
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);
        int valor;

        do {
            System.out.print("Ingrese un valor:");
            valor=teclado.nextInt();
            if(valor>=100) {
                System.out.println("Tiene 3 dígitos");
            }else {
                if(valor>=10) {
                    System.out.println("Tiene 2 dígitos");
                }else {
                    System.out.println("Tiene 1 dígito");
                }
            }

        }while(valor!=0);
    }
}
```

Este será el resultado:

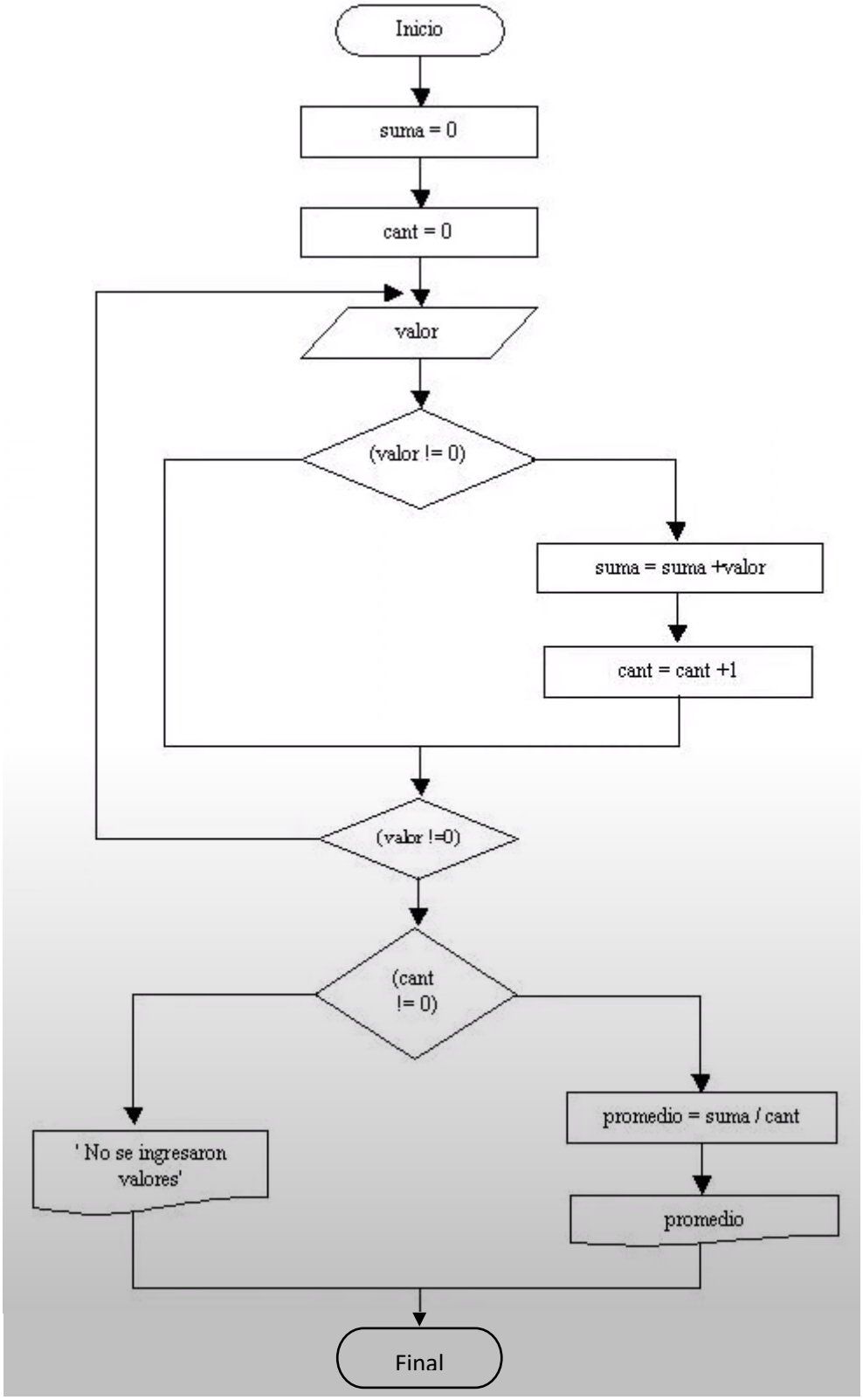
```
Ingrese un valor:700
Tiene 3 dígitos
Ingrese un valor:78
Tiene 2 dígitos
Ingrese un valor:9
Tiene 1 dígito
Ingrese un valor:0
Tiene 1 dígito
```

Capítulo 56.- Estructura repetitiva do while – 2

Problema:

Escribir un programa que solicite la carga de números por teclado, obtener su promedio. Finalizar la carga de valores cuando se cargue el valor 0.

Cuando la finalización depende de algún valor ingresado por el operador conviene el empleo de la estructura do while, por lo menos se cargará un valor (en el caso más extremo se carga 0, que indica la finalización de la carga de valores).



Vamos a programar:

```java
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);
        int valor, promedio;
        int suma=0;
        int cant=0;

        do {
            System.out.print("Ingrese un valor:");
            valor=teclado.nextInt();

            if (valor!=0) {
                suma=suma+valor;
                cant=cant+1;
            }
        }while (valor!=0);

        if (cant!=0) {
            promedio=suma/cant;
            System.out.print("El promedio es "+promedio);
        }else {
            System.out.print("No se ingresaron valores");
        }
    }
}
```

Este será el resultado cuando ejecutemos:

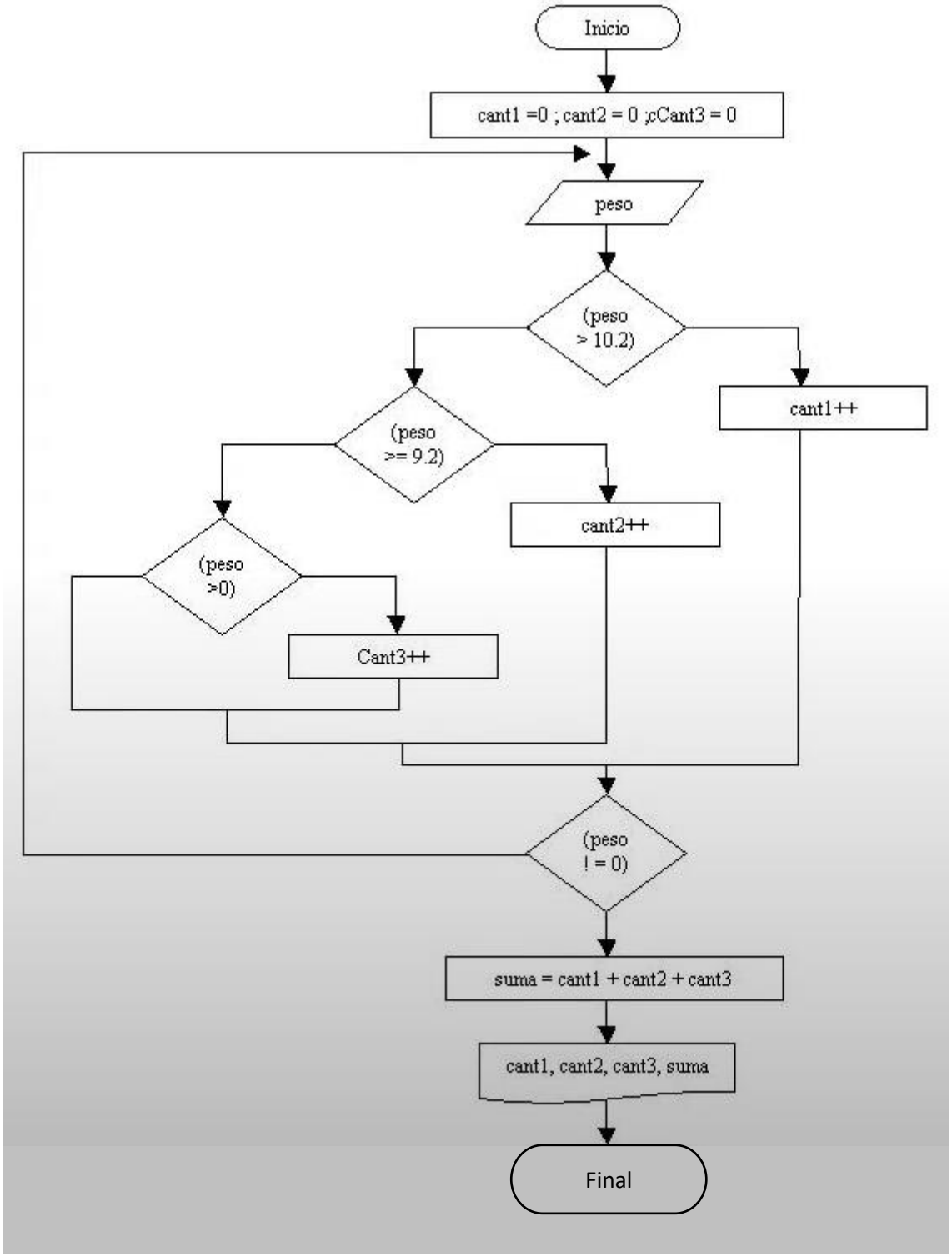
```
Ingrese un valor:10
Ingrese un valor:20
Ingrese un valor:30
Ingrese un valor:40
Ingrese un valor:50
Ingrese un valor:0
El promedio es 30
```

Capítulo 57.- Estructura repetitiva do while – 3

Problema:

Realizar un programa que permita ingresar el peso (en kilogramos) de piezas. El proceso termina cuando ingresamos el valor 0. Se de informar:

a) Cuántas piezas tienen un peso entre 9.8 Kg. y 10.2 Kg.?, cuántas con más de 10.2 Kg.? y cuántas con menos de 9.8 Kg.?
b) La cantidad de piezas procesadas.



Vamos a programar:

*Clase1.java

```java
package paquete1;

import java.util.Scanner;

public class Clase1 {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);
        float peso;

        int cant1=0, cant2=0, cant3=0,suma;
        do {
            System.out.print("Ingrese el peso de la pieza:");
            peso=teclado.nextFloat();

            if(peso>10.2) {
                cant1++;
            }else {
                if(peso>=9.2) {
                    cant2++;
                }else {
                    if (peso>0) {
                        cant3++;
                    }
                }
            }

        }while(peso!=0);
        suma=cant1+cant2+cant3;
        System.out.println("Piezas con un peso superior 12.0 Kg. hay "+cant1);
        System.out.println("Piezas con un peso de entre 9.2 y 12.0 hay "+cant2);
        System.out.println("Piezas con un peso menor a 9.2 hay "+cant2);
        System.out.print("La suma total de piezas es de "+suma);
    }
}
```

Vamos a ejecutar y observaremos el siguiente resultado:

```
Ingrese el peso de la pieza:10,45
Ingrese el peso de la pieza:10,70
Ingrese el peso de la pieza:10,15
Ingrese el peso de la pieza:5
Ingrese el peso de la pieza:0
Piezas con un peso superior 12.0 Kg. hay 2
Piezas con un peso de entre 9.2 y 12.0 hay 1
Piezas con un peso menor a 9.2 hay 1
La suma total de piezas es de 4
```

Capítulo 58.- Estructura repetitiva do while – 4

Problema:

Realizar un programa que acumule (sume) valores ingresados por teclado hasta ingresas el 9999 (no sumar dicho valor, indica que ha finalizado la carga). Imprimir el valor acumulado e informar si dicho valor es cero, mayor de cero o menor a cero.

Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9
10         int valor, suma=0;
11         int cero=0, mayorcero=0, menorcero=0;
12
13         do {
14             System.out.print("Ingrese un valor:");
15             valor=teclado.nextInt();
16             if(valor!=9999) {
17                 suma=suma+valor;
18                 }
19         }while(valor!=9999);
20         System.out.println("La suma de los valores es: "+suma);
21         if(suma>0) {
22             System.out.print("El valor acumulado es positivo");
23         }else {
24             if(suma<0) {
25                 System.out.print("El valor acumulado es negativo");
26             }else {
27                 System.out.print("El valor acumulado es cero");
28             }
29         }
30     }
31 }
```

Este será el resultado cuando lo ejecutemos:

```
Ingrese un valor:1000
Ingrese un valor:500
Ingrese un valor:-200
Ingrese un valor:9999
La suma de los valores es: 1300
El valor acumulado es positivo
```

Capítulo 59.- Estructura repetitiva do while – 5

Problema:

En un banco se procesan datos de las cuentas corrientes de sus clientes. De cada cuenta corriente se conoce: número de cuenta y saldo actual. El ingreso de datos debe finalizar al ingresar un valor negativo en el número de cuenta.

Se pide confeccionar un programa que lea los datos de la cuentas corrientes e informe:

a) De cada cuenta: número de cuenta y estado de la cuenta según su saldo, sabiendo que:

Estado de la cuenta: 'Acreedor' si el saldo es >0.
'Deudor' si el saldo es <0.
'Nulo' si el saldo es =0.

b) La suma total de los saldos acreedores.

Vamos a programar:

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9          int cuenta;
10         float saldo, suma=0;
11
12         do {
13             System.out.print("Ingrese el número de cuenta: ");
14             cuenta=teclado.nextInt();
15             if(cuenta>0) {
16                 System.out.print("Ingrese saldo acutal: ");
17                 saldo=teclado.nextFloat();
18                 if(saldo>0) {
19                     System.out.println("Tiene saldo Acreedor");
20                     suma=suma+saldo;
21                 }else {
22                     if(saldo<0) {
23                         System.out.println("Tiene saldo Deudor");
24                     }else {
25                         System.out.println("Tiene saldo Nulo");
26                     }
27                 }
28
29             }
30         }while(cuenta>0);
31         System.out.print("La suma total de los saldos acreedores es de "+suma);
32     }
33 }
```

Vamos a ejecutar y este será el resultado:

```
Ingrese el número de cuenta: 1
Ingrese saldo acutal: 5000
Tiene saldo Acreedor
```

```
Ingrese el número de cuenta: 2
Ingrese saldo acutal: 8000
Tiene saldo Acreedor
Ingrese el número de cuenta: 3
Ingrese saldo acutal: 0
Tiene saldo Nulo
Ingrese el número de cuenta: 4
Ingrese saldo acutal: -1000
Tiene saldo Deudor
Ingrese el número de cuenta: -1
La suma total de los saldos acreedores es de 13000.0
```

Capítulo 60.- Cadenas de caracteres en Java – 1

Tipos de datos primitivos: int
float

Tipos de datos referencia: String

Problema 1:

Solicitar el ingreso del nombre y la edad de dos personas. Mostrar el nombre de la persona con mayor edad.

*Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  final class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9          String nombre1, nombre2;
10         int edad1, edad2;
11         System.out.print("Ingrese el primer nombre:");
12         nombre1=teclado.next();
13         System.out.print("Ingrese la edad:");
14         edad1=teclado.nextInt();
15         System.out.print("Ingrese el segundo nombre:");
16         nombre2=teclado.next();
17         System.out.print("Ingrese la edad:");
18         edad2=teclado.nextInt();
19         if(edad1>edad2) {
20             System.out.print("La persona de mayor edad es "+nombre1);
21         }else {
22             System.out.print("La persona de mayor edad es "+nombre2);
23         }
24     }
25 }
```

Cuando ejecutemos este será el resultado:

```
Ingrese el primer nombre:Juan
Ingrese la edad:45
Ingrese el segundo nombre:Ana
Ingrese la edad:10
La persona de mayor edad es Juan
```

La variable de tipo String solo almacena una palabra si ponemos espacios en blanco nos dará un error.

Capítulo 61.- Cadenas de caracteres en Java – 2

Problema 2:

Solicitar el ingreso del apellido, nombre y edad de dos personas (el apellido y nombre almacenarlo en una misma variable). Mostrar el nombre de la persona con mayor edad. Realizar la carga del apellido y nombre en una variable de tipo String.

*Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9          String nombre1, nombre2;
10         int edad1, edad2;
11         System.out.print("Ingrese el primer nombre con el apellido:");
12         nombre1=teclado.nextLine();
13         System.out.print("Ingrese la edad:");
14         edad1=teclado.nextInt();
15         System.out.print("Ingrese el segundo nombre con el apellido:");
16         teclado.nextLine();
17         nombre2=teclado.nextLine();
18         System.out.print("Ingrese la edad:");
19         edad2=teclado.nextInt();
20         if(edad1>edad2) {
21             System.out.print("La persona de mayor edad es "+nombre1);
22         }else {
23             System.out.print("La persona de mayor edad es "+nombre2);
24         }
25     }
26 }
```

En la línea 16 con la instrucción 'teclado.nextLine()' lo que hace es borrar el buffer del teclado, si omitimos esta línea, veremos lo que pasa:

```
// teclado.nextLine();

Ingrese el primer nombre con el apellido:Pablo Rodriquez
Ingrese la edad:45
Ingrese el segundo nombre con el apellido:Ingrese la edad:
```

Ahora veremos el resultado con la línea:

```
Ingrese el primer nombre con el apellido:Pabro Rodrigez
Ingrese la edad:32
Ingrese el segundo nombre con el apellido:Ana Durán
Ingrese la edad:45
La persona de mayor edad es Ana Durán
```

Capítulo 62.- Cadenas de caracteres en Java – 3

Problema 3:

Solicitar el ingreso de dos apellidos. Mostrar un mensaje si son iguales o distintos.

Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9          String apellido1, apellido2;
10
11         System.out.print("Ingrese el primer apellido:");
12         apellido1=teclado.nextLine();
13         System.out.print("Ingrese el segundo apellido:");
14         apellido2=teclado.nextLine();
15         if(apellido1.equals(apellido2)) {
16             System.out.print("Los apellidos son iguales");
17         }else {
18             System.out.print("Los apellidos son distintos");
19         }
20     }
21 }
```

Este será el resultado:

```
Ingrese el primer apellido:lopez
Ingrese el segundo apellido:Lopez
Los apellidos son distintos
```

Podréis observar que diferencia las mayúsculas de la minúsculas, para que no las diferencie cambiaremos 'equals' por 'equalsIgnoreCase'.

*Clase1.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clase1 {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9          String apellido1, apellido2;
10
11         System.out.print("Ingrese el primer apellido:");
12         apellido1=teclado.nextLine();
13         System.out.print("Ingrese el segundo apellido:");
14         apellido2=teclado.nextLine();
15         if(apellido1.equalsIgnoreCase(apellido2)) {
16             System.out.print("Los apellidos son iguales");
17         }else {
18             System.out.print("Los apellidos son distintos");
19         }
20     }
21 }
```

Vamos a ejecutar:

```
Ingrese el primer apellido:Lopez
Ingrese el segundo apellido:lopez
Los apellidos son iguales
```

Capítulo 63.- Declaración de una clase y definición de objetos – 1

class [nombre de la clase] {
  [atributos de la clase]
  [métodos de la clase]
  [main]
}

Problema:
Confeccionar una clase que permita cargar el nombre y la edad de una persona. Mostrar los datos cargados. Imprimir un mensaje si es mayor de edad (edad>=18)
Vamos a programar:

Persona.java

```
1  package paquete;
2
3  import java.util.Scanner;
4
5  public class Persona {
6      private String nombre;
7      private int edad;
8      private Scanner teclado;
9
10     public void inicializar() {
11         teclado=new Scanner(System.in) ;
12         System.out.print("Ingrese el nombre:");
13         nombre=teclado.next();
14         System.out.print("Ingrese la edad:");
15         edad=teclado.nextInt();
16     }
17
18     public void imprimir() {
19         System.out.println("Nombre de la persona:"+nombre);
20         System.out.println("Edad:"+edad);
21     }
22
23     public void esMayor() {
24         if(edad>=18) {
25             System.out.println("Es mayor de edad");
26         }else {
27             System.out.println("No es mayor de edad");
28         }
29     }
30
31     public static void main(String[] ar) {
32         Persona persona1=new Persona();
33         persona1.inicializar();
34         persona1.imprimir();
35         persona1.esMayor();
36     }
37 }
```

Vamos a ejecutar:

```
Ingrese el nombre:Juan
Ingrese la edad:45
Nombre de la persona:Juan
Edad:45
Es mayor de edad
```

Capítulo 64.- Declaración de una clase y definición de objetos – 2

class [nombre de la clase] {
  [atributos de la clase]
  [métodos de la clase]
  [main]
}

Problema:

Desarrollar un programa que cargue los lados de un triángulo e implante los siguientes métodos: Inicializar los atributos, imprimir el valor del lado mayor y otro que muestre si es equilátero o no.

Vamos a programar:

*Triangulo.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Triangulo {
6      private int lado1;
7      private int lado2;
8      private int lado3;
9      private Scanner teclado;
10
11     public void inicializar() {
12         teclado=new Scanner(System.in);
13         System.out.print("Ingrese lado 1 del triángulo:");
14         lado1=teclado.nextInt();
15         System.out.print("Ingrese lado 2 del triángulo:");
16         lado2=teclado.nextInt();
17         System.out.print("Ingrese lado 3 del triángulo:");
18         lado3=teclado.nextInt();
19     }
20
21     public void imprimirMayor() {
22         if(lado1>lado2 && lado1>lado3) {
23             System.out.println("El lado mayor es "+lado1);
24         }else {
25             if(lado2>lado3) {
26                 System.out.println("El lado mayor es "+lado2);
27             }else {
28                 System.out.println("El lado mayor es "+lado3);
29             }
30         }
31     }
32
33     public void equilatero() {
34         if(lado1==lado2 && lado1==lado3) {
35             System.out.println("El triángulo es equilátero");
36         }else {
37             System.out.println("El triángulo no es equilátero");
38         }
39     }
40
```

```
39      public static void main(String[] args) {
40          Triangulo triangulo1=new Triangulo();
41          triangulo1.inicializar();
42          triangulo1.imprimirMayor();
43          triangulo1.equilatero();
44      }
45  }
```

Vamos a ejecutar ingresando los tres lado iguales.

```
Ingrese lado 1 del triángulo:40
Ingrese lado 2 del triángulo:40
Ingrese lado 3 del triángulo:40
El lado mayor es 40
El triángulo es equilátero
```

Vamos a ejecutar de nuevo ingresando los lados con distintos valores.

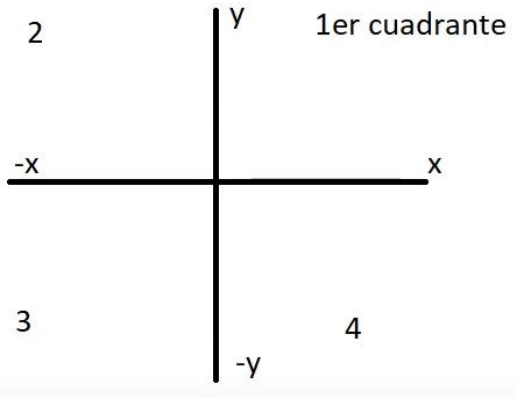
```
Ingrese lado 1 del triángulo:20
Ingrese lado 2 del triángulo:40
Ingrese lado 3 del triángulo:60
El lado mayor es 60
El triángulo no es equilátero
```

Capítulo 65.- Declaración de una clase y definición de objetos – 3

class [nombre de la clase] {
  [atributos de la clase]
  [métodos de la clase]
  [main]
}

Problema:

Desarrollar una clase que presente un punto en el plano y tenga los siguientes métodos: cargar los valores de x e y, imprimir en que cuadrante se encuentra dicho punto (concepto matemático, primer cuadrante si x e y son positivos, si x<0 e y>0 segundo cuadrante, etc.)



Vamos a programar:

*Punto.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Punto {
6      private int x;
7      private int y;
8      private Scanner teclado;
9
10     public void inicializar() {
11         teclado=new Scanner(System.in);
12         System.out.print("Ingrese la coodenada x:");
13         x=teclado.nextInt();
14         System.out.print("Ingrese la coordenada y:");
15         y=teclado.nextInt();
16     }
17
18     public void imprimirCuadrante() {
19         if(x>0 && y>0) {
20             System.out.print("Se encuentra en el primer cudrante");
21         }else {
22             if(x<0 && y>0) {
23                 System.out.print("Se encuentra en el segundo cuadrante");
24             }else {
25                 if(x<0 && y<0) {
26                     System.out.print("Se encuenbtra en el tercer cuadrante");
27                 }else {
28                     if(x>0 && y<0) {
29                         System.out.print("Se encuentra en el cuarto cuadrante");
30                     }else {
31                         System.out.print("No se encuentra en ningún cuadrante");
```

```
32                      }
33                  }
34              }
35          }
36      }
37
38      public static void main(String[] args) {
39          Punto punto1=new Punto();
40          punto1.inicializar();
41          punto1.imprimirCuadrante();
42      }
43  }
```

Vamos a ejecutar:

```
Ingrese la coodenada x:10
Ingrese la coordenada y:10
Se encuentra en el primer cudrante
```

Ejecutamos de nuevo:

```
Ingrese la coodenada x:-10
Ingrese la coordenada y:10
Se encuentra en el segundo cuadrante
```

Ejecutamos de nuevo:

```
Ingrese la coodenada x:-10
Ingrese la coordenada y:-10
Se encuenbtra en el tercer cuadrante
```

Ejecutamos de nuevo:

```
Ingrese la coodenada x:10
Ingrese la coordenada y:-10
Se encuentra en el cuarto cuadrante
```

Ejecutamos de nuevo:

```
Ingrese la coodenada x:0
Ingrese la coordenada y:10
No se encuentra en ningún cuadrante
```

Capítulo 66.- Declaración de una clase y definición de objetos – 4

class [nombre de la clase] {
  [atributos de la clase]
  [métodos de la clase]
  [main]
}

Problema:

Desarrollar una clase que represente un Cuadrado y tenga los siguientes métodos: Cargar el valor de su lado, imprimir su perímetro y su superficie.

Vamos a programar:

Cuadrado.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Cuadrado {
6      private int lado;
7      private int perimetro;
8      private int superficie;
9      private Scanner teclado;
10
11     public void inicializar() {
12         teclado=new Scanner(System.in);
13         System.out.print("Ingrese el lado de un cuadrado:");
14         lado=teclado.nextInt();
15     }
16
17     public void imprimirCuadrado() {
18         perimetro=lado*4;
19         superficie=lado*lado;
20         System.out.println("El perimetro del cuadrado es "+ perimetro);
21         System.out.print("La superficie del cuadrado es "+superficie);
22     }
23
24     public static void main(String[] args) {
25         Cuadrado cuadrado1=new Cuadrado();
26         cuadrado1.inicializar();
27         cuadrado1.imprimirCuadrado();
28     }
29 }
```

Cuando ejecutemos este será el resultado:

```
Ingrese el lado de un cuadrado:10
El perimetro del cuadrado es 40
La superficie del cuadrado es 100
```

Capítulo 67.- Declaración de una clase y definición de objetos – 5

Problema propuesto:

Confeccionar una clase que represente un empleado.
Definir como atributos su nombre y su sueldo.
Confeccionar los métodos para la carga, otro para imprimir su datos y por último uno que imprima un mensaje de si debe pagar impuestos (si sueldo supera a 3000).

Vamos a programar:

*Empleado.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Empleado {
6      private String nombre;
7      private int sueldo;
8      private Scanner teclado;
9
10     public void inicializar() {
11         teclado=new Scanner(System.in);
12         System.out.print("Ingrese el nombre del empleado:");
13         nombre=teclado.next();
14         System.out.print("Ingrese su sueldo:");
15         sueldo=teclado.nextInt();
16     }
17
18     public void imprimirEmpleado() {
19         System.out.println("El nombre del empleado es "+nombre);
20         System.out.println("El sueldo es "+sueldo);
21     }
22
23     public void pagarImpuestos() {
24         if(sueldo>3000) {
25             System.out.print("El empleado debe pagar impuestos");
26         }else {
27             System.out.print("El empelado no debe pagar impuestos");
28         }
29     }
30
31     public static void main(String[] args) {
32         Empleado empleado1=new Empleado();
33         empleado1.inicializar();
34         empleado1.imprimirEmpleado();
35         empleado1.pagarImpuestos();
36     }
37 }
```

Si ejecutamos este será el resultado si cobra más de 3000.

```
Ingrese el nombre del empleado:Juan
Ingrese su sueldo:7000
El nombre del empleado es Juan
El sueldo es 7000
El empleado debe pagar impuestos
```

Este será el resultado si cobra menos de 3000.

```
Ingrese el nombre del empleado:Luis
Ingrese su sueldo:2500
El nombre del empleado es Luis
El sueldo es 2500
El empelado no debe pagar impuestos
```

Capítulo 68.- Declaración de una clase y definición de objetos – 6

Problema propuesto:

Implementar la clase operaciones. Se deben cargar dos valores enteros, calcular su suma, resta, multiplicación y división, cada una en un método, imprimir dichos resultados.

Vamos a programar:

*Operaciones.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Operaciones {
6      private int num1;
7      private int num2;
8      private int suma;
9      private int resta;
10     private int multiplicacion;
11     private float division;
12     private Scanner teclado;
13
14     public void inicializar() {
15         teclado=new Scanner(System.in);
16         System.out.print("Ingrese el primer número:");
17         num1=teclado.nextInt();
18         System.out.print("Ingrese el segundo número:");
19         num2=teclado.nextInt();
20     }
21
22     public void sumar() {
23         suma=num1+num2;
24         System.out.println("La suma de "+num1+"+"+num2+" es igual a "+suma);
25     }
26
27     public void restar() {
28         resta=num1-num2;
29         System.out.println("La resta de "+num1+"-"+num2+" es igual a "+resta);
30     }
31
32     public void multiplicacion() {
33         multiplicacion=num1*num2;
34         System.out.println("La multiplicación de "+num1+"*"+num2+" es igual a "+multiplicacion);
35     }
36
37     public void division() {
38         division=num1/num2;
39         System.out.println("La division de "+num1+"/"+num2+" es igual a "+division);
40     }
41
42     public static void main(String[] args) {
43         Operaciones operacion1=new Operaciones();
44         operacion1.inicializar();
45         operacion1.sumar();
46         operacion1.restar();
47         operacion1.multiplicacion();
48         operacion1.division();
49     }
50 }
```

Si ejecutamos este será el resultado:

```
Ingrese el primer número:10
Ingrese el segundo número:5
La suma de 10+5 es igual a 15
La resta de 10-5 es igual a 5
La multiplicación de 10*5 es igual a 50
La division de 10/5 es igual a 2.0
```

También sería correcto de la siguiente forma:

*Operaciones.java

```java
package paquete1;

import java.util.Scanner;

public class Operaciones {
    private int num1;
    private int num2;
    private Scanner teclado;

    public void inicializar() {
        teclado=new Scanner(System.in);
        System.out.print("Ingrese el primer número:");
        num1=teclado.nextInt();
        System.out.print("Ingrese el segundo número:");
        num2=teclado.nextInt();
    }

    public void sumar() {
        int suma;
        suma=num1+num2;
        System.out.println("La suma de "+num1+"+"+num2+" es igual a "+suma);
    }

    public void restar() {
        int resta;
        resta=num1-num2;
        System.out.println("La resta de "+num1+"-"+num2+" es igual a "+resta);
    }

    public void multiplicacion() {
        int multiplicacion;
        multiplicacion=num1*num2;
        System.out.println("La multiplicación de "+num1+"*"+num2+" es igual a "+multiplicacion);
    }

    public void division() {
        float division;
        division=num1/num2;
        System.out.println("La division de "+num1+"/"+num2+" es igual a "+division);
    }

    public static void main(String[] args) {
        Operaciones operacion1=new Operaciones();
        operacion1.inicializar();
        operacion1.sumar();
        operacion1.restar();
        operacion1.multiplicacion();
        operacion1.division();
    }
}
```

Capítulo 69.- Declaración de métodos – 1

```
public void [nombre del método]() {
   [algoritmo]
}

public void [nombre del método]([parámetro]) {
   [algoritmo]
}
```

Problema:

Confeccionar una clase que permita ingresar valores enteros por teclado y nos muestre la tabla de multiplicar de dicho valor. Finalizar el programa al ingresar el -1.

Vamos a programar:

Tabla.java

```java
package paquete1;

import java.util.Scanner;

public class Tabla {
    private int numTabla;
    private int resultado;
    private Scanner teclado;

    public void inicializar() {
        teclado=new Scanner(System.in);
    }

    public void mostrarTabla() {
        do {
            System.out.print("Ingrese el número de la tabla a consultar:");
            numTabla=teclado.nextInt();
            if(numTabla!=-1) {
                System.out.println("La tabla de multiplicar de "+numTabla);
                System.out.println("-----------------------------------");
                for(int t=1; t<=10; t++) {
                    resultado=numTabla*t;
                    System.out.println(numTabla+" * "+t+" = "+resultado);
                }
            }

        }while(numTabla!=-1);
    }

    public static void main(String[] args) {
        Tabla tabla1=new Tabla();
        tabla1.inicializar();
        tabla1.mostrarTabla();
    }
}
```

Cuando ejecutemos este será el resultado:

```
Ingrese el número de la tabla a consultar:5
La tabla de multiplicar de 5
--------------------------------
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
Ingrese el número de la tabla a consultar:3
La tabla de multiplicar de 3
--------------------------------
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
Ingrese el número de la tabla a consultar:-1
```

Ahora vamos a modificar el código para que admita parámetros:

Tabla1.java

```java
package paquete1;

import java.util.Scanner;

public class Tabla1 {
    private int resultado;
    private Scanner teclado;

    public void inicializar() {
        teclado=new Scanner(System.in);
    }

    public void mostrarTabla(int tabla) {
        System.out.println("La tabla de multiplicar de "+tabla);
        System.out.println("--------------------------------");
        for(int t=1; t<=10; t++) {
            resultado=tabla*t;
            System.out.println(tabla+" * "+t+" = "+resultado);
        }
    }

    public static void main(String[] args) {
        Tabla1 tabla1=new Tabla1();
        tabla1.inicializar();
        tabla1.mostrarTabla(7);
    }
}
```

Este será el resultado:

```
La tabla de multiplicar de 7
---------------------------------
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70
```

Capítulo 70.- Declaración de métodos – 2

public [void/int/float] [nombre del método](parámetros]) {
  [algoritmo]
}

Problema:

Confeccionar una clase que permita ingresar tres valores por teclado. Luego mostrar el mayor y el menor.

Vamos a programar:

*Numeros.java

```java
package paquete1;

import java.util.Scanner;

public class Numeros {
    private int num1;
    private int num2;
    private int num3;
    private Scanner teclado;

    public void inicializar() {
        teclado=new Scanner(System.in);
        System.out.print("Ingrese el primer número:");
        num1=teclado.nextInt();
        System.out.print("Ingrese el segundo número:");
        num2=teclado.nextInt();
        System.out.print("Ingrese el tercer número:");
        num3=teclado.nextInt();
        System.out.println("El mayor de los tres valores es "+mostrarMayor(num1, num2, num3));
        System.out.print("El valor menor de los tres valores es "+mostrarMenor(num1, num2, num3));
    }

    public int mostrarMayor(int v1, int v2, int v3) {
        if(v1>v2 && v1>v3) {
            return v1;
        }else {
            if(v2>v3) {
                return v2;
            }else {
                return v3;
            }
        }
    }

    public int mostrarMenor(int v1, int v2, int v3) {
        if(v1<v2 && v1<v3) {
            return v1;
        }else {
            if(v2<v3) {
                return v2;
            }else {
                return v3;
            }
        }
    }


    public static void main(String[] args) {
        Numeros numero1=new Numeros();
        numero1.inicializar();
    }
}
```

Este será el resultado cuando ejecutemos:

```
Ingrese el primer número:45
Ingrese el segundo número:6
Ingrese el tercer número:2
El mayor de los tres valores es 45
El valor menor de los tres valores es 2
```

Capítulo 71.- Estructura de datos tipo vector – 1

Hemos empleado variables de distinto tipo para el almacenamiento de datos (variables int, float, String).

Un vector es una estructura de datos que permite almacenar un CONJUNTO de datos del MISMO tipo.

Con un único nombre de define un vector y por medio de un subíndice hacemos referencia a cada elemento del mismo (componente).

Problema:

Se desea guardar los sueldos de 5 operarios.

Según lo conocido deberíamos definir 5 variables si queremos tener en un cierto momento los 5 sueldos almacenados en memoria.

Empleando un vector solo se requiere definir un único nombre y acedemos a cada elemento por medio del subíndice.

Vamos a programar:

SueldosEmpleados.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class SueldosEmpleados {
6      private int[] sueldos;
7      private Scanner teclado;
8
9      public void inicializar() {
10         teclado=new Scanner(System.in);
11         sueldos=new int[5];
12         for(int x=0; x<5; x++) {
13             System.out.print("Ingrese el sueldo número "+x+1+":");
14             sueldos[x]=teclado.nextInt();
15         }
16     }
17
18     public void mostrarSueldos() {
19         for(int y=0; y<5; y++) {
20             System.out.println("Sueldo "+ (y+1) +":"+sueldos[y]);
21         }
22     }
23
```

```
24      public static void main(String[] args) {
25          SueldosEmpleados sueldos1=new SueldosEmpleados();
26          sueldos1.inicializar();
27          sueldos1.mostrarSueldos();
28      }
29  }
```

Vamos a ejecutar y este será el resultado:

```
Ingrese el sueldo número 01:1200
Ingrese el sueldo número 11:750
Ingrese el sueldo número 21:820
Ingrese el sueldo número 31:550
Ingrese el sueldo número 41:490
Sueldo 1:1200
Sueldo 2:750
Sueldo 3:820
Sueldo 4:550
Sueldo 5:490
```

Capítulo 72.- Estructura de datos tipo vector – 2

Problema:

Definir un ventor de 5 componentes de tipo float que representan las alturas de 5 personas. Obtener el promedio de las mismas. Contar cuántas personas son más altas que el promedio y cuantas más bajas.

Vamos a programar:

*Altura.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Altura {
6      private float[] altura;
7      private Scanner teclado;
8
9      public void inicializar() {
10         teclado=new Scanner(System.in);
11         altura=new float[5];
12         for (int x=0; x<5; x++) {
13             System.out.print("Ingrese la altura por (ejemplo 1,87) :");
14             altura[x]=teclado.nextFloat();
15         }
16     }
17
18     public float promedio() {
19         float suma=0;
20         for(int y=0; y<5; y++) {
21             suma=suma+altura[y];
22         }
23         return (suma/5);
24     }
25
26     public int contarMasAltas() {
27         int masAltas=0;
28         for(int z=0; z<5; z++) {
29             if(altura[z]>promedio()) {
30                 masAltas++;
31             }
32         }
33         return masAltas;
34     }
35
36     public int contarMasBajas() {
37         int masBajas=0;
38         for(int z=0; z<5; z++) {
39             if(altura[z]<promedio()) {
40                 masBajas++;
41             }
42         }
43         return masBajas;
44     }
45
46     public static void main(String[] args) {
47         Altura altura1=new Altura();
48         altura1.inicializar();
49         System.out.println("El promedio de las alturas es de "+altura1.promedio());
50         System.out.println("Con una altura superior al promedio "+altura1.contarMasAltas());
51         System.out.print("Con una altura inferior al promedio "+altura1.contarMasBajas());
52     }
53 }
```

Vamos a ejecutar:

```
Ingrese la altura por (ejemplo 1,87) :1,78
Ingrese la altura por (ejemplo 1,87) :1,90
Ingrese la altura por (ejemplo 1,87) :1,65
Ingrese la altura por (ejemplo 1,87) :1,72
Ingrese la altura por (ejemplo 1,87) :1,55
El promedio de las alturas es de 1.72
Con una altura superior al promedio 2
Con una altura inferior al promedio 2
```

Otra forma de realizarlo:

AlturaPersonas.java

```java
import java.util.Scanner;

public class AlturaPersonas {
    private Scanner teclado;
    private float []alturas;
    private float promedio;

    public void cargar() {
        teclado=new Scanner(System.in);
        alturas=new float[5];
        for(int f=0;f<5;f++) {
            System.out.print("Ingrese la altura de la persona:");
            alturas[f]=teclado.nextFloat();
        }
    }

    public void calcularPromedio() {
        float suma=0;
        for(int f=0;f<5;f++) {
            suma=suma+alturas[f];
        }
        promedio=suma/5;
        System.out.println("Promedio de alturas:"+promedio);
    }

    public void mayoresMenores() {
        int may=0, men=0;
        for(int f=0;f<5;f++) {
            if (alturas[f]>promedio) {
                may++;
            } else {
                if (alturas[f]<promedio) {
                    men++;
                }
            }
        }
        System.out.println("La cantidad de personas mayor a la altura promedio:"+may);
        System.out.println("La cantidad de personas menores al promedio:"+men);
    }

    public static void main(String[] ar) {
        AlturaPersonas ap=new AlturaPersonas();
        ap.cargar();
        ap.calcularPromedio();
        ap.mayoresMenores();
    }
}
```

El resultado será el mismo.

Capítulo 73.- Estructura de datos tipo vector – 3

Problema:

Una empresa tiene dos turnos (mañana y tarde) en los que trabajan 8 empleados (4 por la mañana y 4 por la tarde).
Confeccionar un programa que permita almacenar los sueldos de los empleados agrupados por turno.
Imprimir los gastos en sueldos de cada turno.

Vamos a programar:

*Empresa.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Empresa {
6      private int[] turnoma;
7      private int[] turnota;
8      Scanner teclado;
9
10     public void cargar() {
11         teclado=new Scanner(System.in);
12         turnoma=new int[4];
13         turnota=new int[4];
14         System.out.println("Turno de mañana");
15         for(int x=0; x<4; x++) {
16             System.out.print("Ingrese el sueldo: ");
17             turnoma[x]=teclado.nextInt();
18         }
19         System.out.println("Turno de tarde");
20         for(int x=0; x<4; x++) {
21             System.out.print("Ingrese el sueldo: ");
22             turnota[x]=teclado.nextInt();
23         }
24     }
25
26     public void imprimir() {
27         int totalma=0;
28         int totalta=0;
29         for(int x=0; x<4; x++) {
30             totalma=totalma+turnoma[x];
31             totalta=totalta+turnota[x];
32         }
33         System.out.println("Los sueldos del turno de mañana son: "+totalma);
34         System.out.print("Los sueldos del tuno de tarde son: "+totalta);
35     }
36
37     public static void main(String[] args) {
38         Empresa empresa1=new Empresa();
39         empresa1.cargar();
40         empresa1.imprimir();
41     }
42 }
```

Este será el resultado cuando ejecutemos:

```
Turno de mañana
Ingrese el sueldo: 3000
Ingrese el sueldo: 4000
Ingrese el sueldo: 2000
Ingrese el sueldo: 1000
Turno de tarde
Ingrese el sueldo: 4000
Ingrese el sueldo: 4000
Ingrese el sueldo: 4000
Ingrese el sueldo: 4000
Los sueldos del turno de mañana son: 10000
Los sueldos del tuno de tarde son: 16000
```

Capítulo 74.- Estructura de datos tipo vector – 4

Problema propuesto:

Desarrollar un programa que permita ingresar un vector de 8 elementos, e informe:
El valor acumulado de todos los elementos del vector.
El valor acumulado de los elementos del vector que sean mayores a 36.
Cantidad de valores mayores a 50.

Vamos a programar:

*Numeros.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Numeros {
6      private int[] num;
7      Scanner teclado;
8
9      public void cargar() {
10         teclado=new Scanner(System.in);
11         num=new int[8];
12         for(int x=0; x<8; x++) {
13             System.out.print("Ingrese un número: ");
14             num[x]=teclado.nextInt();
15         }
16     }
17
18     public void contarElementos() {
19         int sumaTodo=0;
20         int sumaMayor36=0;
21         int cantidadMayor50=0;
22         for(int x=0; x<8; x++) {
23             sumaTodo=sumaTodo+num[x];
24         if(num[x]>36) {
25             sumaMayor36=sumaMayor36+num[x];
26         }
27         if(num[x]>50) {
28             cantidadMayor50++;
29         }
30         }
31         System.out.println("El valor acumulado de todos los elementos se "+sumaTodo);
32         System.out.println("El valor acumulado elementos mayores de 36: "+sumaMayor36);
33         System.out.print("La cantidad de valores mayores de 50: "+cantidadMayor50);
34     }
35
36     public static void main(String[] args) {
37         Numeros numero1=new Numeros();
38         numero1.cargar();
39         numero1.contarElementos();
40     }
41 }
42
```

Vamos a ejecutar y este será el resultado:

```
Ingrese un número: 70
Ingrese un número: 50
Ingrese un número: 80
Ingrese un número: 1
Ingrese un número: 2
Ingrese un número: 3
Ingrese un número: 4
Ingrese un número: 5
```

```
El valor acumulado de todos los elementos se 215
El valor acumulado elementos mayores de 36: 200
La cantidad de valores mayores de 50: 2
```

Capítulo 75.- Estructura de datos tipo vector – 5

Problema propuesto:

Realizar un programa que pida la carga de dos vectores numéricos enteros de 4 elementos. Obtener la suma de los dos vectores, dicho resultado guardarlo en un tercer vector del mismo tamaño. Sumar componente a componente.

Vamos a programar:

*Vectores.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Vectores {
6      Scanner teclado;
7      private int[] vec1;
8      private int[] vec2;
9      private int[] sumavec;
10
11     public void cargar() {
12         teclado=new Scanner(System.in);
13         vec1=new int[4];
14         System.out.println("Introducir los datos en el primer vector");
15         for(int x=0; x<4; x++) {
16             System.out.print("Ingrese un valor:");
17             vec1[x]=teclado.nextInt();
18         }
19         vec2=new int[4];
20         System.out.println("Introducir los datos en el segundo vector");
21         for(int x=0; x<4; x++) {
22             System.out.print("Ingrese un valor:");
23             vec2[x]=teclado.nextInt();
24         }
25     }
26
27     public void sumarVector() {
28         sumavec=new int[4];
29         for(int x=0; x<4; x++) {
30             sumavec[x]=vec1[x]+vec2[x];
31         }
32     }
33
34     public void mostrarVector() {
35         for(int x=0; x<4; x++) {
36             System.out.println(sumavec[x]);
37         }
38     }
39
40     public static void main(String[] args) {
41         Vectores vector1 = new Vectores();
42         vector1.cargar();
43         vector1.sumarVector();
44         vector1.mostrarVector();
45     }
46 }
```

Este será el resultado cuando ejecutemos:

```
Introducir los datos en el primer vector
Ingrese un valor:7
Ingrese un valor:10
Ingrese un valor:12
Ingrese un valor:9
Introducir los datos en el segundo vector
Ingrese un valor:8
Ingrese un valor:4
Ingrese un valor:9
Ingrese un valor:20
15
14
21
29
```

Capítulo 76.- Estructura de datos tipo vector – 6

Problema propuesto:

Se tienen las notas del primer parcial de los alumnos de dos cursos, el curso A y el curso B, cada curso cuenta con 5 alumnos.
Realizar un programa que muestre el curso que obtuvo el mayor promedio general.

Vamos a programar:

Alumnos.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Alumnos {
6      Scanner teclado;
7      private int[] cursoA;
8      private int[] cursoB;
9
10     public void cargar() {
11         teclado=new Scanner(System.in);
12         cursoA=new int[5];
13         cursoB=new int[5];
14         System.out.println("Notas del curso A");
15         for(int x=0; x<5; x++) {
16             System.out.print("Ingrese la nota:");
17             cursoA[x]=teclado.nextInt();
18         }
19         System.out.println("Notas del curso B");
20         for(int y=0; y<5; y++) {
21             System.out.print("Ingrese la nota:");
22             cursoB[y]=teclado.nextInt();
23         }
24     }
25
26     public void promedio() {
27         int sumaA=0;
28         int sumaB=0;
29         float promedioA=0;
30         float promedioB=0;
31         for(int x=0; x<5; x++) {
32             sumaA=sumaA+cursoA[x];
33             sumaB=sumaB+cursoB[x];
34         }
35         promedioA=sumaA/5;
36         promedioB=sumaB/5;
37         if(promedioA>promedioB) {
38             System.out.print("El curso con promedio más alto es Curso A");
39         }else {
40             System.out.print("El curso con promedio más alto es Curso B");
41         }
42     }
43
44     public static void main(String[] args) {
45         Alumnos alumno1=new Alumnos();
46         alumno1.cargar();
47         alumno1.promedio();
48     }
49 }
```

Vamos a ejecutar este será el resultado:

```
Notas del curso A
Ingrese la nota:6
Ingrese la nota:7
Ingrese la nota:8
Ingrese la nota:9
Ingrese la nota:10
Notas del curso B
Ingrese la nota:8
Ingrese la nota:7
Ingrese la nota:6
Ingrese la nota:5
Ingrese la nota:4
El curso con promedio más alto es Curso A
```

Ejecutamos de nuevo:

```
Notas del curso A
Ingrese la nota:6
Ingrese la nota:5
Ingrese la nota:8
Ingrese la nota:4
Ingrese la nota:3
Notas del curso B
Ingrese la nota:7
Ingrese la nota:8
Ingrese la nota:6
Ingrese la nota:9
Ingrese la nota:10
El curso con promedio más alto es Curso B
```

Capítulo 77.- Estructura de datos tipo vector – 7

Problema propuesto:

Cargar un vector de 10 elementos y verificar posteriormente si el mismo está ordenado de menor a mayor.

Vamos a programar:

Ordenado.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Ordenado {
6      Scanner teclado;
7      private int[] numeros;
8
9      public void cargar() {
10         teclado=new Scanner(System.in);
11         numeros=new int[10];
12         for(int x=0; x<10; x++) {
13             System.out.print("Ingrese un número:");
14             numeros[x]=teclado.nextInt();
15         }
16     }
17
18     public void estaOrdenado() {
19         int ord=0;
20         for(int x=1; x<10; x++) {
21             if (numeros[x-1]>numeros[x]) {
22                 ord=1;
23                 break;
24             }
25         }
26         if (ord==1) {
27             System.out.print("Los elementos del vector no están ordenados de menor a mayor");
28         }else {
29             System.out.print("Los elementos del vector si esán ordendos de menor a mayor");
30         }
31     }
32
33     public static void main(String[] args) {
34         Ordenado ordenado1=new Ordenado();
35         ordenado1.cargar();
36         ordenado1.estaOrdenado();
37     }
38 }
```

Vamos a ejecutar introduciendo valores en orden:

```
Ingrese un número:1
Ingrese un número:2
Ingrese un número:3
Ingrese un número:4
Ingrese un número:5
Ingrese un número:6
Ingrese un número:7
Ingrese un número:8
Ingrese un número:9
Ingrese un número:10
Los elementos del vector si esán ordendos de menor a mayor
```

Ahora vamos a introducir valores que no están ordenados de menor a mayor:

```
Ingrese un número:1
Ingrese un número:2
Ingrese un número:4
Ingrese un número:3
Ingrese un número:5
Ingrese un número:7
Ingrese un número:6
Ingrese un número:9
Ingrese un número:10
Ingrese un número:8
Los elementos del vector no están ordenados de menor a mayor
```

Capítulo 78.- Vector (Tamaño de un vector) – 1

Tamaño de un vector – atributo length

Podemos crear un vector pasando una variable en el momento de la creación.
vec=new int[cant]
Luego podemos consultar el tamaño de un vector mediante el atributo length.
System.out.print(vec.length)

Problema:

Se desea almacenar los sueldos de operarios. Cuando se ejecuta el programa se debe pedir la cantidad de sueldos a ingresar. Luego crear un vector con dicho tamaño.

Vamos a programar:

Vector.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Vector {
6      Scanner teclado;
7      private int[] numeros;
8      int numelementos;
9
10     public void cargar(){
11         teclado=new Scanner(System.in);
12         System.out.print("¿Cuantos sueldos tenemos que ingresar:");
13         numelementos=teclado.nextInt();
14         numeros=new int[numelementos];
15         for(int x=0; x<numelementos; x++) {
16             System.out.print("Ingrese el esueldo:");
17             numeros[x]=teclado.nextInt();
18         }
19     }
20
21     public void mostrar() {
22         System.out.println("El número de elementos del vector es de "+numeros.length);
23         System.out.println("Estos son los elementos:");
24         for (int x=0; x<numelementos; x++) {
25             System.out.println(numeros[x]);
26         }
27     }
28
29     public static void main(String[] args) {
30         Vector vector1=new Vector();
31         vector1.cargar();
32         vector1.mostrar();
33     }
34 }
```

Vamos a ejecutar y este será el resultado:

```
¿Cuantos sueldos tenemos que ingresar:3
Ingrese el esueldo:1000
Ingrese el esueldo:2000
Ingrese el esueldo:4000
El número de elementos del vector es de 3
Estos son los elementos:
1000
2000
4000
```

Ejecutamos de nuevo:

```
¿Cuantos sueldos tenemos que ingresar:5
Ingrese el esueldo:1000
Ingrese el esueldo:2000
Ingrese el esueldo:4000
Ingrese el esueldo:1000
Ingrese el esueldo:3000
El número de elementos del vector es de 5
Estos son los elementos:
1000
2000
4000
1000
3000
```

Capítulo 79.- Vector (Tamaño de un vector) – 2

Problema propuesto:

Desarrollar un programa que permita ingresar un vector de n elementos, ingresar n por teclado. Luego imprimir la suma de todos sus elementos.

Vamos a programar:

*Elementos.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Elementos {
6      Scanner teclado;
7      private int[] numeros;
8      int numelementos;
9      int sumar=0;
10
11     public void cargar() {
12         teclado=new Scanner(System.in);
13         System.out.print("Cuantos números vas a ingresar:");
14         numelementos=teclado.nextInt();
15         numeros=new int[numelementos];
16         for(int x=0; x<numelementos; x++) {
17             System.out.print("Ingrese un valor:");
18             numeros[x]=teclado.nextInt();
19         }
20     }
21
22     public int sumar() {
23         for(int x=0; x<numeros.length; x++) {
24             sumar=sumar+numeros[x];
25         }
26         return sumar;
27     }
28
29     public void imprimir() {
30         System.out.println("el número de elemetos es de "+numeros.length);
31         System.out.print("La suma de sus elementos es de "+sumar());
32     }
33
34     public static void main(String[] args) {
35         Elementos elemento1=new Elementos();
36         elemento1.cargar();
37         elemento1.imprimir();
38     }
39 }
```

Vamos a ejecutar y este será el resultado:

```
Cuantos números vas a ingresar:3
Ingrese un valor:10
Ingrese un valor:10
Ingrese un valor:10
el número de elemetos es de 3
La suma de sus elementos es de 30
```

Capítulo 80.- Vectores paralelos

Este concepto se da cuando una relación entre los componentes de igual subíndice (misma posición) de un ventor y otro.

nombres	Juan	Ana	Marcos	Pablo	Laura
edades	12	21	27	14	21

Problema:

Desarrollar un programa que permita cargar 5 nombres de personas y sus edades respectivas. Luego de realizar la carga por teclado de todos los datos imprimir los nombres de las personas mayores de edad (mayores o iguales a 18 años).

Vamos a programar:

*Personas.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Personas {
6      Scanner teclado;
7      private String[] nombre;
8      private int[] edad;
9
10     public void cargar() {
11         teclado=new Scanner(System.in);
12         nombre = new String[5];
13         edad = new int[5];
14         for(int x=0; x<5; x++) {
15             System.out.print("Ingrese el nombre:");
16             nombre[x]=teclado.next();
17             System.out.print("Ingres su edad:");
18             edad[x]=teclado.nextInt();
19         }
20     }
21
22     public void imprimir() {
23         for(int x=0; x<5; x++) {
24             System.out.println("El empleado "+nombre[x]+" iiene una edad de "+edad[x]);
25         }
26     }
27
28     public void mayoresEdad() {
29         System.out.println("Los mayores de edad son");
30         for (int x=0; x<5; x++) {
31             if(edad[x]>=18) {
32                 System.out.println(nombre[x]);
33             }
34         }
35     }
36
37     public static void main(String[] args) {
38         Personas persona1=new Personas();
39         persona1.cargar();
40         persona1.imprimir();
41         persona1.mayoresEdad();
42     }
43 }
```

Vamos a ejecutar y este será el resultado:

```
Ingrese el nombre:Juan
Ingres su edad:12
Ingrese el nombre:Ana
Ingres su edad:21
Ingrese el nombre:Marcos
Ingres su edad:27
Ingrese el nombre:Pablo
Ingres su edad:14
Ingrese el nombre:Laura
Ingres su edad:21
El empleado Juan iiene una edad de 12
El empleado Ana iiene una edad de 21
El empleado Marcos iiene una edad de 27
El empleado Pablo iiene una edad de 14
El empleado Laura iiene una edad de 21
Los mayores de edad son
Ana
Marcos
Laura
```

Capítulo 81.- Vectores (mayor y menor elemento) – 1

es una actividad común la búsqueda del mayor y menor elemento de un vector, lo mismo que su posición.

sueldos

120	750	820	550	490
sueldos[0]	sueldos[1]	sueldos[2]	sueldos[3]	sueldos[4]

Problema:

Confeccionar un programa que permita cargar los nombres de 5 operarios y sus sueldos respectivos.

Mostrar el sueldo mayor y el nombre del operario.

Vamos a programar:

*Empleados.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Empleados {
6      Scanner teclado;
7      private String[] nombre;
8      private int[] sueldo;
9      private int pos;
10
11     public void cargar() {
12         teclado=new Scanner(System.in);
13         nombre=new String[5];
14         sueldo=new int[5];
15         for (int x=0; x<5; x++) {
16             System.out.print("Ingrese el nombre del operio:");
17             nombre[x]=teclado.next();
18             System.out.print("Ingrese su sueldo:");
19             sueldo[x]=teclado.nextInt();
20         }
21     }
22
23     public void buscarMayor() {
24         int mayor=sueldo[0];
25         for(int x=1; x<5; x++) {
26             if(sueldo[x]>mayor) {
27                 pos=x;
28                 mayor=sueldo[x];
29             }
30         }
31         System.out.print("El operario "+nombre[pos]+" tiene el mayor sueldo que es "+sueldo[pos]);
32     }
33
34     public static void main(String[] args) {
35         Empleados empleado1=new Empleados();
36         empleado1.cargar();
37         empleado1.buscarMayor();
38     }
39 }
```

Vamos a ejecutar y este será el resultado:

```
Ingrese el nombre del operio:Juan
Ingrese su sueldo:120
Ingrese el nombre del operio:Luis
Ingrese su sueldo:750
```

```
Ingrese el nombre del operio:Pedro
Ingrese su sueldo:820
Ingrese el nombre del operio:Ana
Ingrese su sueldo:550
Ingrese el nombre del operio:Carlos
Ingrese su sueldo:490
El operario Pedro tiene el mayor sueldo que es 820
```

Capítulo 82.- Vectores (mayor y menor elemento) – 2

Problema propuesto:

Cargar un vector de n elementos, imprimir el menor y un mensaje si se repite dentro del vector.

Vamos a programar:

*Vector.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Vector {
6      Scanner teclado;
7      private int[] numero;
8
9      public void cargar() {
10         teclado=new Scanner(System.in);
11         System.out.print("Cuantos números tiene el vector:");
12         int n=teclado.nextInt();
13         numero=new int[n];
14         for(int x=0; x<n; x++) {
15             System.out.print("Ingrese un número:");
16             numero[x]=teclado.nextInt();
17         }
18     }
19
20     public void numeroMenor() {
21         int menor=numero[0];
22         for(int x=1; x<numero.length; x++) {
23             if (numero[x]<menor){
24                 menor=numero[x];
25             }
26         }
27         System.out.println("El número menor es "+menor);
28         int cuenta=0;
29         for(int x=0; x<numero.length; x++) {
30             if(numero[x]==menor) {
31                 cuenta++;
32             }
33         }
34         if(cuenta>1) {
35             System.out.print("El valor menor se repite "+cuenta+" veces.");
36         }else {
37             System.out.print("El valor menor no se repite.");
38         }
39     }
40
41
42     public static void main(String[] args) {
43         Vector vector1=new Vector();
44         vector1.cargar();
45         vector1.numeroMenor();
46     }
47 }
```

Cuando ejecutemos repitiendo el valor mínimo:

```
Cuantos números tiene el vector:4
Ingrese un número:3
Ingrese un número:3
Ingrese un número:67
Ingrese un número:5
El número menor es 3
El valor menor se repite 2 veces.
```

Vamos a ejecutar de nuevo si repetir el número menor.

```
Cuantos números tiene el vector:7
Ingrese un número:5
Ingrese un número:2
Ingrese un número:8
Ingrese un número:7
Ingrese un número:9
Ingrese un número:4
Ingrese un número:6
El número menor es 2
El valor menor no se repite.
```

Capítulo 83.- Vectores (ordenamiento) – 1

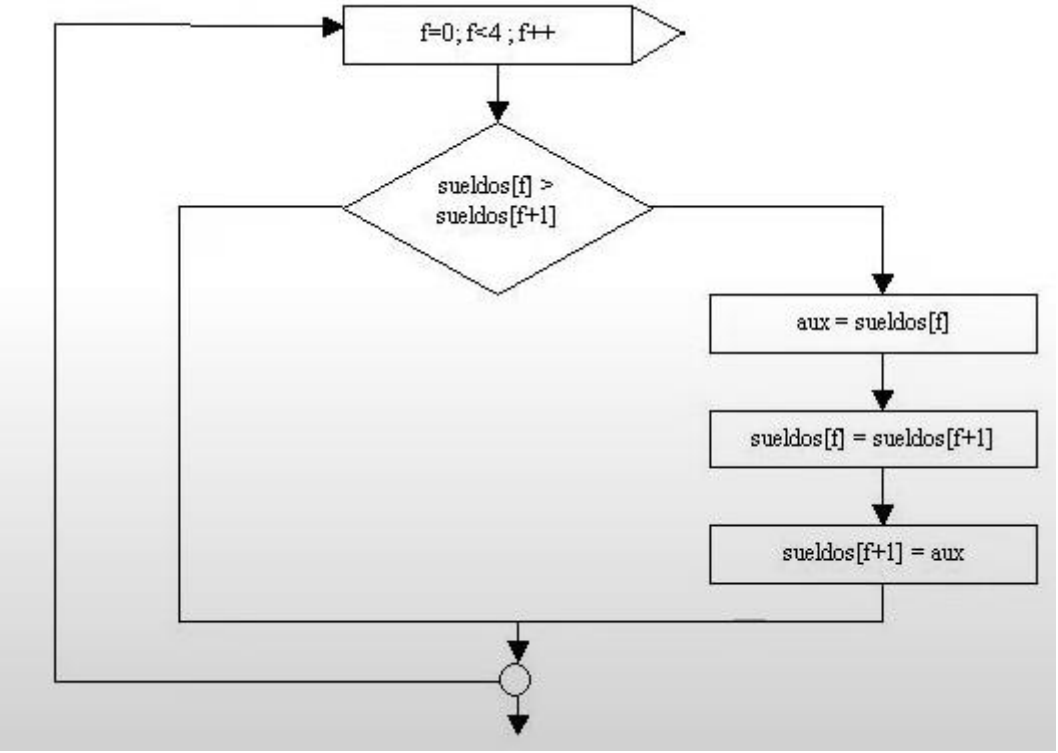
Vectores (ordenamiento)

El ordenamiento en un vector se logra intercambiando los componentes de manera que: vec[0]<=vec[1]<=vec[2] etc.
Si se cumple lo dicho anteriormente decimo que el vector esta ordenado de menor a mayor. Igualmente podemos ordenar un vector de mayor a menor.

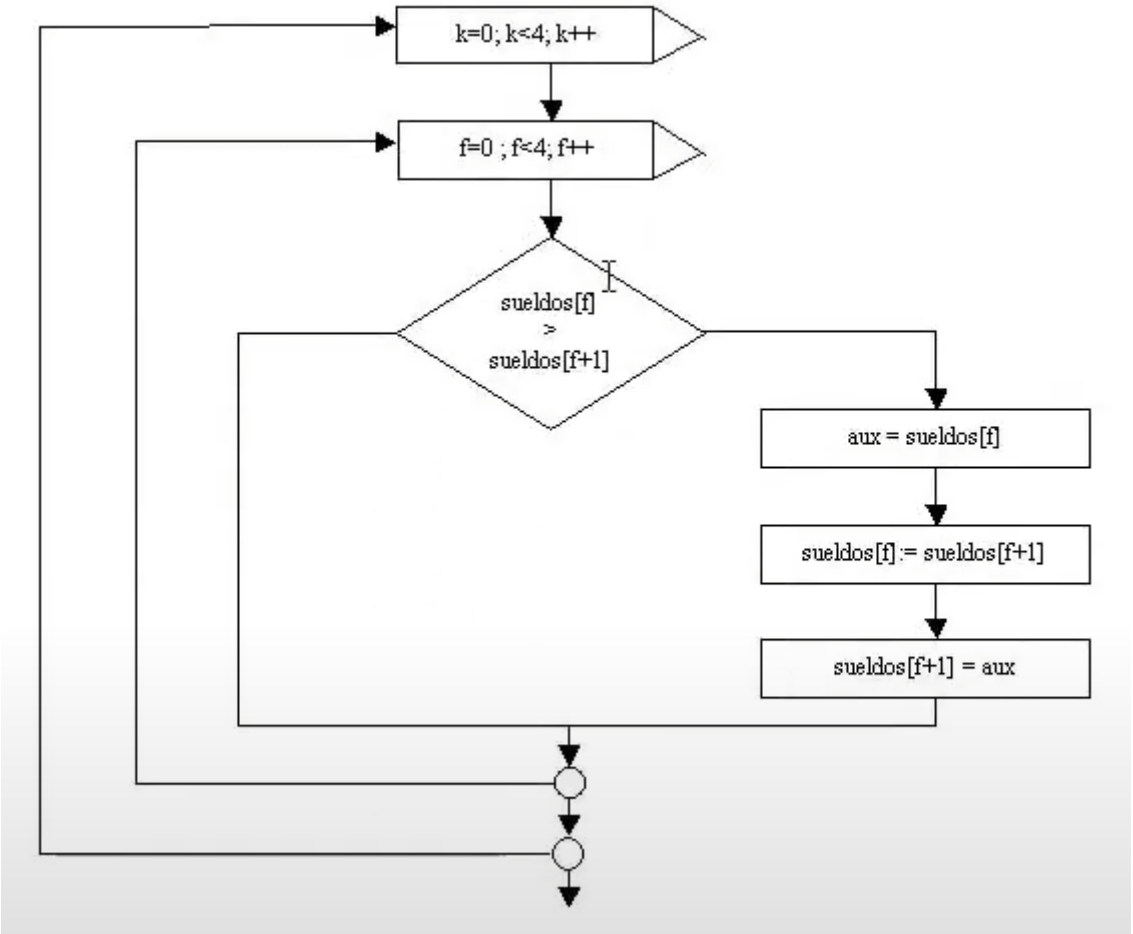
Problema:

Se debe crear un vector donde almacenar 5 sueldos. Ordenar el vector sueldos de menor a mayor.

Primer paso:



Cuando	f = 0	f = 1	f = 2	f = 3
	750	750	750	750
	1200	820	820	820
	820	1200	550	550
	550	550	1200	490
	490	490	490	1200




```
Cuando k = 0
                f = 0           f = 1           f = 2           f = 3
                750             750             750             750
                1200            820             820             820
                820             1200            550             550
                550             550             1200            490
                490             490             490             1200

Cuando k = 1
                f = 0           f = 1           f  = 2          f = 3
                750             750             750             750
                820             550             550             550
                550             820             490             490
                490             490             820             820
                1200            1200            1200            1200

Cuando k = 2
                f = 0           f = 1           f  = 2          f = 3
                550             550             550             550
                750             490             490             490
                490             750             750             750
                820             820             820             820
                1200            1200            1200            1200
```

```
Cuando k = 3
                f = 0           f = 1           f  = 2          f = 3
                490             490             490             490
                550             550             550             550
                750             750             750             750
                820             820             820             820
                1200            1200            1200            1200
```

Vamos a programar:

*Ordenar.java

```java
package paquete1;

import java.util.Scanner;

public class Ordenar {
    Scanner teclado;
    private int[] sueldo;

    public void cargar() {
        teclado=new Scanner(System.in);
        sueldo=new int[5];
        for(int x=0; x<5; x++) {
            System.out.print("Ingrese el sueldo:");
            sueldo[x]=teclado.nextInt();
        }
    }

    public void ordenar() {
        int aux;
        for(int x=0; x<4; x++) {
            for(int y=0; y<4; y++) {
                if(sueldo[y]>sueldo[y+1]) {
                    aux=sueldo[y];
                    sueldo[y]=sueldo[y+1];
                    sueldo[y+1]=aux;
                }
            }
        }
    }

    public void imprimir() {
        System.out.println("Los números ordenados de menor a mayor son:");
        for(int x=0; x<5; x++) {
            System.out.print(sueldo[x]);
            if(x<4) {
                System.out.print(" - ");
            }
        }
    }

    public static void main(String[] args) {
        Ordenar ordena1=new Ordenar();
        ordena1.cargar();
        ordena1.ordenar();
        ordena1.imprimir();
    }
}
```

Cuando ejecutamos este será el resultado:

```
Ingrese el sueldo:750
Ingrese el sueldo:1200
Ingrese el sueldo:820
Ingrese el sueldo:550
Ingrese el sueldo:490
Los números ordenados de menor a mayor son:
490 - 550 - 750 - 820 - 1200
```

Capítulo 84.- Vectores (ordenamiento) – 2

El ordenamiento en un vector se logra intercambiando los componentes de manera que: vec[0]<=vec[1]<=vec[2] etc.
Si se cumple lo dicho anteriormente decimo que el vector esta ordenado de menor a mayor.
Igualmente podemos ordenar un vector de mayor a menor.

Problema:

Definir un vector donde almacenar los nombres de 5 países. Confeccionar el algoritmo de ordenamiento alfabético.

Vamos a programar:

*Paises.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Paises {
6      Scanner teclado;
7      private String[] pais;
8
9      public void cargar() {
10         teclado=new Scanner(System.in);
11         pais=new String[5];
12         for(int x=0; x<5; x++) {
13             System.out.print("Ingrese el país:");
14             pais[x]=teclado.next();
15         }
16     }
17
18     public void ordenar() {
19         String aux;
20         for(int x=0; x<4; x++) {
21             for(int y=0; y<4; y++) {
22                 if(pais[y].compareTo(pais[y+1])>0) {
23                     aux=pais[y];
24                     pais[y]=pais[y+1];
25                     pais[y+1]=aux;
26
27                 }
28             }
29         }
30     }
31
32     public void imprimir() {
33         System.out.println("La lista de paises ordenada alfabéticamente.");
34         for (int x=0; x<5; x++) {
35             System.out.print(pais[x]);
36             if (x<4) {
37                 System.out.print(", ");
38             }
39         }
40     }
41
```

```
42      public static void main(String[] args) {
43          Paises pais1=new Paises();
44          pais1.cargar();
45          pais1.ordenar();
46          pais1.imprimir();
47      }
48  }
```

Vamos a ejecutar este será el resultado:

```
Ingrese el país:Chile
Ingrese el país:Argentina
Ingrese el país:Uruguay
Ingrese el país:Brasil
Ingrese el país:Colombia
La lista de paises ordenada alfabéticamente.
Argentina, Brasil, Chile, Colombia, Uruguay
```

Capítulo 85.- Vectores (ordenamiento) – 3

Problema propuesto:

Cargar un vector de n elementos de tipo entero. Ordenar posteriormente el vector.

Vamos a programar:

*Vector.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Vector {
6      Scanner teclado;
7      private int numero[];
8
9      public void cargar() {
10         teclado=new Scanner(System.in);
11         System.out.print("Cuántos números desea ingresar:");
12         int n=teclado.nextInt();
13         numero=new int[n];
14         for(int x=0; x<n; x++) {
15             System.out.print("Ingrese un número:");
16             numero[x]=teclado.nextInt();
17         }
18     }
19
20     public void ordenar() {
21         int aux;
22         for(int x=0; x<numero.length-1; x++) {
23             for(int y=0; y<numero.length-1; y++) {
24                 if(numero[y]>numero[y+1]) {
25                     aux=numero[y];
26                     numero[y]=numero[y+1];
27                     numero[y+1]=aux;
28                 }
29             }
30         }
31     }
32
33     public void imprimir() {
34         for(int x=0; x<numero.length; x++) {
35             System.out.print(numero[x]);
36             if(x<numero.length-1) {
37                 System.out.print(", ");
38             }else {
39                 System.out.print(".");
40             }
41         }
42     }
43
44     public static void main(String[] args) {
45         Vector vector1=new Vector();
46         vector1.cargar();
47         vector1.ordenar();
48         vector1.imprimir();
49     }
50 }
```

Vamos a ejecutar, este será el resultado:

```
Cuántos números desea ingresar:5
Ingrese un número:10
Ingrese un número:50
Ingrese un número:2
Ingrese un número:789
Ingrese un número:34
2, 10, 34, 50, 789.
```

Capítulo 86.- Vectores (ordenamiento con vectores paralelos) – 1

Cuando se tienen vectores paralelos y se ordena uno de ellos hay que tener la precaución de intercambiar los elementos de los vectores paralelos.

Problema:

Confeccionar un programa que permita cargar los nombres de 5 alumnos y sus notas respectivas.
Luego ordenar las notas de mayor a menor. Imprimir las notas y nos nombres de los alumnos.

Vamos a programar:

Alumnos.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Alumnos {
6      Scanner teclado;
7      private String nombres[];
8      private int notas[];
9
10     public void cargar() {
11         teclado=new Scanner(System.in);
12         nombres=new String[5];
13         notas=new int[5];
14         for(int x=0; x<5; x++) {
15             System.out.print("Ingrese el nombre del alumnos:");
16             nombres[x]=teclado.next();
17             System.out.print("Ingrese su nota:");
18             notas[x]=teclado.nextInt();
19         }
20     }
21
22     public void ordenar() {
23         String auxNombre;
24         int auxNota;
25         for(int x=0; x<4; x++) {
26             for(int y=0; y<4; y++) {
27                 if(notas[y]<notas[y+1]) {
28                     auxNota=notas[y];
29                     notas[y]=notas[y+1];
30                     notas[y+1]=auxNota;
31                     auxNombre=nombres[y];
32                     nombres[y]=nombres[y+1];
33                     nombres[y+1]=auxNombre;
34                 }
35             }
36         }
37     }
38
39     public void imprimir() {
40         for(int x=0; x<5; x++) {
41             System.out.println("El alumno "+nombres[x]+" ha obtenido una nota de "+notas[x]);
42         }
43     }
44
45     public static void main(String[] args) {
46         Alumnos alumno1=new Alumnos();
47         alumno1.cargar();
48         alumno1.ordenar();
49         alumno1.imprimir();
50     }
51 }
```

Vamos a ejecutar este será el resultado:

```
Ingrese el nombre del alumnos:Juan
Ingrese su nota:5
Ingrese el nombre del alumnos:Ana
Ingrese su nota:7
Ingrese el nombre del alumnos:Pedro
Ingrese su nota:9
Ingrese el nombre del alumnos:Carlos
Ingrese su nota:2
Ingrese el nombre del alumnos:Luis
Ingrese su nota:7
El alumno Pedro ha obtenido una nota de 9
El alumno Ana ha obtenido una nota de 7
El alumno Luis ha obtenido una nota de 7
El alumno Juan ha obtenido una nota de 5
El alumno Carlos ha obtenido una nota de 2
```

Capítulo 87.- Vectores (ordenamiento con vectores paralelos) – 2

Problema propuesto:

Cargar en un vector los nombres de 5 países y en otro vector paralelo la cantidad de habitantes del mismo. Ordenar alfabéticamente e imprimir los resultados. Por último ordenar con respecto a la cantidad de habitantes (de mayor a menor) e imprimir nuevamente.

Vamos a programar:

```
package paquete1;

import java.util.Scanner;

public class Paises {
	private Scanner teclado;
	private String[] paises;
	private int [] habitantes;

	public void cargar() {
		teclado=new Scanner(System.in);
		paises=new String[5];
		habitantes=new int[5];
		for(int x=0; x<5; x++) {
			System.out.print("Ingrese un país:");
			paises[x]=teclado.next();
			System.out.print("Ingrese el número de habitantes:");
			habitantes[x]=teclado.nextInt();
		}
	}

	public void ordenaPaises(){
		String aux1;
		int aux2;
		for(int x=0;x<4; x++) {
			for(int y=0; y<4-x; y++) {
				if(paises[y].compareTo(paises[y+1])>0) {
					aux1=paises[y];
					paises[y]=paises[y+1];
					paises[y+1]=aux1;
					aux2=habitantes[y];
					habitantes[y]=habitantes[y+1];
					habitantes[y+1]=aux2;
				}
			}
		}
	}

	public void ordenaHabitantes() {
		String aux1;
		int aux2;
		for(int x=0;x<4; x++) {
			for(int y=0; y<4-x; y++) {
				if(habitantes[y]>habitantes[y+1]) {
					aux2=habitantes[y];
					habitantes[y]=habitantes[y+1];
					habitantes[y+1]=aux2;
					aux1=paises[y];
					paises[y]=paises[y+1];
```

```
                                paises[y+1]=aux1;
                        }
                }
        }
    }

    public void imprimir() {
        System.out.println("--------------------");
        System.out.println("|Relación de paises|");
        System.out.println("--------------------");
        for(int x=0; x<5; x++) {
            System.out.println(paises[x]+": "+habitantes[x]);
        }
        System.out.println("--------------------");
    }

    public static void main(String[] args) {
        Paises pais1=new Paises();
        pais1.cargar();
        System.out.println("Imprimri por orden de entrada");
        pais1.imprimir();
        pais1.ordenaPaises();
        System.out.println("Imprimir por orden de paises");
        pais1.imprimir();
        pais1.ordenaHabitantes();
        System.out.println("Imprimir por número de habitantes");
        pais1.imprimir();
    }
}
```

Este será el resultado cuando ejecutemos:

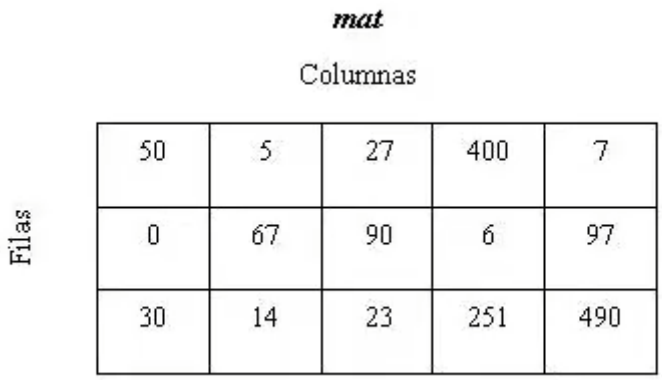
```
Ingrese un país:Chile
Ingrese el número de habitantes:19
Ingrese un país:Argentina
Ingrese el número de habitantes:45
Ingrese un país:Uruguay
Ingrese el número de habitantes:3
Ingrese un país:Brasil
Ingrese el número de habitantes:200
Ingrese un país:Bolivia
Ingrese el número de habitantes:10
Imprimri por orden de entrada
--------------------
|Relación de paises|
--------------------
Chile: 19
Argentina: 45
Uruguay: 3
Brasil: 200
Bolivia: 10
--------------------
```

```
Imprimir por orden de paises
--------------------
|Relación de paises|
--------------------
Argentina: 45
Bolivia: 10
Brasil: 200
Chile: 19
Uruguay: 3
--------------------
Imprimir por número de habitantes
--------------------
|Relación de paises|
--------------------
Uruguay: 3
Bolivia: 10
Chile: 19
Argentina: 45
Brasil: 200
--------------------
```

Capítulo 88.- Estructura de datos tipo matriz – 1

Una matriz es una estructura de datos que permite almacenar un CONJUNTO de datos del mismo tipo.

Con un único nombre se define la matriz y por medio de DOS subíndices hacemos referencia a cada elemento del mismo (componente)



Problema:

Crear una matriz de 3 filas por 5 columnas con elementos de tipo int, cargar sus componentes y luego imprimirlas.

Vamos a programar:

Matriz.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Matriz {
6      private Scanner teclado;
7      private int [][] numero;
8
9      public void cargar() {
10         teclado=new Scanner(System.in);
11         numero=new int[3][5];
12         for(int x=0; x<3; x++) {
13             System.out.println("Ingrese los valores de la fila "+(x+1)+":");
14             for(int y=0; y<5; y++) {
15                 System.out.print("Ingres un valor:");
16                 numero[x][y]=teclado.nextInt();
17             }
18         }
19     }
20
21     public void imprimir() {
22         for(int x=0; x<numero.length; x++) {
23             for(int y=0; y<numero[x].length; y++) {
24                 System.out.print(numero[x][y]);
25                 if (y<numero[x].length-1) {
26                     System.out.print(" - ");
27                 }
28             }
29             System.out.println();
30         }
```

```
31      }
32      public static void main(String[] args) {
33          Matriz matriz1=new Matriz();
34          matriz1.cargar();
35          matriz1.imprimir();
36      }
37  }
```

Vamos a ejecutar este será el resultado:

```
Ingrese los valores de la fila 1:
Ingres un valor:50
Ingres un valor:5
Ingres un valor:27
Ingres un valor:400
Ingres un valor:7
Ingrese los valores de la fila 2:
Ingres un valor:0
Ingres un valor:67
Ingres un valor:90
Ingres un valor:6
Ingres un valor:97
Ingrese los valores de la fila 3:
Ingres un valor:30
Ingres un valor:14
Ingres un valor:23
Ingres un valor:251
Ingres un valor:490
50 - 5 - 27 - 400 - 7
0 - 67 - 90 - 6 - 97
30 - 14 - 23 - 251 - 490
```

Capítulo 89.- Estructura de datos tipo matriz – 2

Problema:

Crear y cargar una matriz de 4 filas por 4 columnas. Imprimir la diagonal principal.

```
x   -   -   -
-   x   -   -
-   -   x   -
-   -   -   x
```

Vamos a programar:

Matriz.java

```
1  package paquete1;
2
3  public class Matriz {
4
5      private String [][] elemento;
6
7      public void cargar() {
8          elemento=new String[4][4];
9          for(int x=0; x<4; x++) {
10             for(int y=0; y<4; y++) {
11                 if(x==y) {
12                     elemento[x][y]="x";
13                 }else {
14                     elemento[x][y]="-";
15                 }
16             }
17         }
18     }
19
20     public void imprimir() {
21         for(int x=0; x<4; x++) {
22             for(int y=0; y<4; y++) {
23                 System.out.print(elemento[x][y]+"   ");;
24             }
25             System.out.println();
26         }
27     }
28
29     public static void main(String[] args) {
30         Matriz matriz1=new Matriz();
31         matriz1.cargar();
32         matriz1.imprimir();
33     }
34 }
```

Vamos a ejecutar este será el resultado:

```
x   -   -   -
-   x   -   -
-   -   x   -
-   -   -   x
```

Ahora queremos que en dicha matriz se puedan guardar valores que introduciremos por teclado y solo queremos que se impriman los que se muestran marcados con una x.

Vamos a programar:

Matriz.java

```java
package paquete1;

import java.util.Scanner;

public class Matriz {
    Scanner teclado;
    private int [][] numero;

    public void cargar() {
        teclado=new Scanner(System.in);
        numero=new int[4][4];
        for(int x=0; x<4; x++) {
            for(int y=0; y<4; y++) {
                System.out.print("Ingrese un número:");
                numero[x][y]=teclado.nextInt();
            }
        }
    }

    public void imprimir() {
        for(int x=0; x<4; x++) {
            for(int y=0; y<4; y++) {
                if(x==y) {
                    System.out.println(numero[x][y]);
                }
            }
        }
    }

    public static void main(String[] args) {
        Matriz matriz1=new Matriz();
        matriz1.cargar();
        matriz1.imprimir();
    }
}
```

Vamos a ejecutar, este será el resultado:

```
Ingrese un número:1
Ingrese un número:2
Ingrese un número:3
Ingrese un número:4
Ingrese un número:5
Ingrese un número:6
Ingrese un número:7
Ingrese un número:8
Ingrese un número:9
Ingrese un número:10
Ingrese un número:11
Ingrese un número:12
Ingrese un número:13
Ingrese un número:14
Ingrese un número:15
Ingrese un número:16
1
6
11
16
```

Capítulo 90.- Estructura de datos tipo matriz – 3

Problema:

Crear y cargar una matriz de 3 filas por 4 columnas. Imprimir la primera fila. Imprimir la última fila e imprimir la primera columna.

Vamos a programar:

```
package paquete1;

import java.util.Scanner;

public class Matriz {
	private Scanner teclado;
	private int [][] numero;

	public void cargar() {
		teclado=new Scanner(System.in);
		numero=new int[3][4];
		for(int x=0; x<3; x++) {
			for(int y=0; y<4; y++) {
				System.out.print("Ingrese un valor:");
				numero[x][y]=teclado.nextInt();
			}
		}
	}

	public void imprimirTodo() {
		System.out.println("Imprimiendo toda la matriz");
		for(int x=0; x<3; x++) {
			for(int y=0; y<4; y++) {
				System.out.print(numero[x][y]+"    ");
			}
			System.out.println();
		}
		System.out.println("--------------------");
	}

	public void imprimirPrimeraFila() {
		System.out.println("Imprimiendo primera fila");
		for(int y=0; y<4; y++) {
			System.out.print(numero[0][y]+"   ");
		}
		System.out.println();
		System.out.println("--------------------");
	}

	public void imprimirUltimaFila() {
		System.out.println("Imprimiendo última fila");
		for(int y=0; y<4; y++) {
			System.out.print(numero[2][y]+"   ");
		}
		System.out.println();
		System.out.println("--------------------");
	}

	public void imprimirPrimeraColumna() {
		System.out.println("Imprimiendo primera columna");
		for(int x=0; x<3; x++) {
```

```
                System.out.println(numero[x][0]);
            }
        }

        public static void main(String[] args) {
            Matriz matriz1=new Matriz();
            matriz1.cargar();
            matriz1.imprimirTodo();
            matriz1.imprimirPrimeraFila();
            matriz1.imprimirUltimaFila();
            matriz1.imprimirPrimeraColumna();
        }
}
```

Vamos a ejecutar y este será el resultado:

```
Ingrese un valor:1
Ingrese un valor:2
Ingrese un valor:3
Ingrese un valor:4
Ingrese un valor:5
Ingrese un valor:6
Ingrese un valor:7
Ingrese un valor:8
Ingrese un valor:9
Ingrese un valor:10
Ingrese un valor:11
Ingrese un valor:12
Imprimiendo toda la matriz
1    2    3    4
5    6    7    8
9    10    11    12
--------------------
Imprimiendo primera fila
1   2   3   4
--------------------
Imprimiendo última fila
9   10   11   12
--------------------
Imprimiendo primera columna
1
5
9
```

Capítulo 91.- Estructura de datos tipo matriz – 4

Problema propuesto:

Crear una matriz de 2 filas y 5 columnas. Realizar la carga de componentes por columna (es decir primero ingresar toda la primera columna, luego la segunda columna y así sucesivamente)

Imprimir luego la matriz:

Vamos a programar:

*Matriz.java

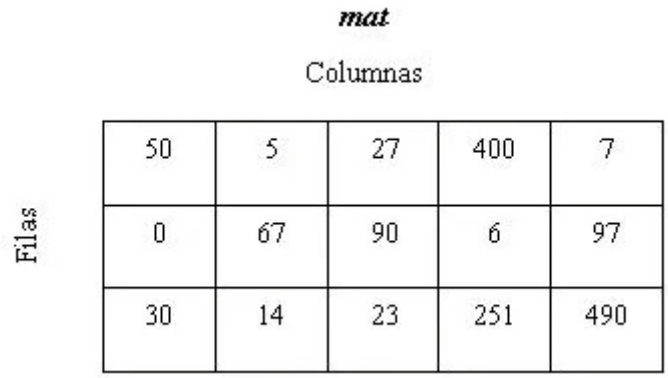
```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Matriz {
6      private Scanner teclado;
7      private int [][] numero;
8
9      public void cargar() {
10         teclado=new Scanner(System.in);
11         numero=new int[2][5];
12         for(int x=0; x<5; x++) {
13             for(int y=0; y<2; y++) {
14                 System.out.print("Ingrese un número:");
15                 numero[y][x]=teclado.nextInt();
16             }
17             imprimir();
18         }
19     }
20
21     public void imprimir() {
22         for(int x=0; x<2; x++) {
23             for(int y=0; y<5; y++) {
24                 System.out.print(numero[x][y]+"    ");
25             }
26             System.out.println();
27         }
28         System.out.println("------------------");
29     }
30
31
32     public static void main(String[] args) {
33         Matriz matriz1=new Matriz();
34         matriz1.cargar();
35     }
36 }
```

Si ejecutamos este será el resultado:

```
Ingrese un número:1
Ingrese un número:2
1   0   0   0   0
2   0   0   0   0
------------------
Ingrese un número:3
Ingrese un número:4
1   3   0   0   0
2   4   0   0   0
------------------
```

```
Ingrese un número:5
Ingrese un número:6
1   3   5   0   0
2   4   6   0   0
------------------
Ingrese un número:7
Ingrese un número:8
1   3   5   7   0
2   4   6   8   0
------------------
Ingrese un número:9
Ingrese un número:10
1   3   5   7   9
2   4   6   8   10
------------------
```

Capítulo 92.- Matrices (cantidad de filas y columnas) – 1



System.out.println("Cantidad de filas de la matriz:" + mat.lenght);

System.out.println("Cantidad de elementos de la primera fila:" + mat[0].lenght);

Problema:

Crear un matriz de n * m fiilas (cargar n y m por teclado) Imprimir la matriz completa y la última fila.

Vamos a programar:

Matriz.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Matriz {
6      private Scanner teclado;
7      private int[][] numero;
8
9      public void cargar() {
10         teclado=new Scanner(System.in);
11         int f, c;
12         System.out.print("Ingrese el número de filas de la matriz:");
13         f=teclado.nextInt();
14         System.out.print("Ingrese el número de columnas de la matriz:");
15         c=teclado.nextInt();
16         numero=new int[f][c];
17         for(int x=0; x<f; x++) {
18             System.out.println("Fila :"+(x+1));
19             for(int y=0; y<c; y++) {
20                 System.out.print("Ingrese el valor para columna "+(y+1)+":");
21                 numero[x][y]=teclado.nextInt();
22             }
23         }
24     }
25
26     public void imprimirMatrizCompleta() {
27         for(int x=0; x<numero.length; x++) {
28             for(int y=0; y<numero[0].length; y++) {
29                 System.out.print(numero[x][y]+"    ");
30             }
31             System.out.println();
32         }
33     }
34
```

```
35      public void imprimirUltimaFila() {
36          System.out.println("Imprimiendo la última fila");
37          for(int x=0; x<numero[0].length; x++) {
38              System.out.print(numero[numero.length-1][x]+"    ");
39          }
40      }
41
42      public static void main(String[] args) {
43          Matriz matriz1=new Matriz();
44          matriz1.cargar();
45          matriz1.imprimirMatrizCompleta();
46          matriz1.imprimirUltimaFila();
47      }
48  }
```

Vamos a ejecutar y este será el resultado:

```
Ingrese el número de filas de la matriz:3
Ingrese el número de columnas de la matriz:5
Fila :1
Ingrese el valor para columna 1:50
Ingrese el valor para columna 2:5
Ingrese el valor para columna 3:27
Ingrese el valor para columna 4:400
Ingrese el valor para columna 5:7
Fila :2
Ingrese el valor para columna 1:0
Ingrese el valor para columna 2:67
Ingrese el valor para columna 3:90
Ingrese el valor para columna 4:6
Ingrese el valor para columna 5:97
Fila :3
Ingrese el valor para columna 1:30
Ingrese el valor para columna 2:14
Ingrese el valor para columna 3:23
Ingrese el valor para columna 4:251
Ingrese el valor para columna 5:490
50   5   27   400   7
0   67   90   6   97
30   14   23   251   490
Imprimiendo la última fila
30   14   23   251   490
```

Capítulo 93.- Matrices (cantidad de filas y columnas) – 2

Problema:

Crear una matriz de n * m (cargar n y m por teclado) imprimir el mayor elemento y la fila y columna donde se almacena.

Vamos a programar:

Matriz.java

```
package paquete1;

import java.util.Scanner;

public class Matriz {
    private Scanner teclado;
    private int[][] numero;

    public void cargar() {
        teclado=new Scanner(System.in);
        int f, c;
        System.out.print("Ingrese el número de filas de la matriz:");
        f=teclado.nextInt();
        System.out.print("Ingrese el número de columnas de la matriz:");
        c=teclado.nextInt();
        numero=new int[f][c];
        for(int x=0; x<f; x++) {
            System.out.println("Fila :"+(x+1));
            for(int y=0; y<c; y++) {
                System.out.print("Ingrese el valor para columna "+(y+1)+":");
                numero[x][y]=teclado.nextInt();
            }
        }
    }

    public void imprimir() {
        int mayor=0;
        int posicionFila=0;
        int posicionColumna=0;
        for(int x=0; x<numero.length; x++) {
            for(int y=0; y<numero[0].length; y++) {
                if(numero[x][y]>mayor) {
                    mayor=numero[x][y];
                    posicionFila=x;
                    posicionColumna=y;
                }
            }
        }
        System.out.println("El valor mayor es "+mayor);
        System.out.print("Se encuentra en la fila "+posicionFila+" y en la columna "+posicionColumna);
    }

    public static void main(String[] args) {
        Matriz matriz1=new Matriz();
        matriz1.cargar();
        matriz1.imprimir();
    }
}
```

Vamos a ejecutar y este será el resultado:

```
Ingrese el número de filas de la matriz:3
Ingrese el número de columnas de la matriz:5
Fila :1
Ingrese el valor para columna 1:50
Ingrese el valor para columna 2:5
Ingrese el valor para columna 3:27
Ingrese el valor para columna 4:400
Ingrese el valor para columna 5:7
```

```
Fila :2
Ingrese el valor para columna 1:0
Ingrese el valor para columna 2:67
Ingrese el valor para columna 3:90
Ingrese el valor para columna 4:6
Ingrese el valor para columna 5:97
Fila :3
Ingrese el valor para columna 1:30
Ingrese el valor para columna 2:14
Ingrese el valor para columna 3:23
Ingrese el valor para columna 4:251
Ingrese el valor para columna 5:490
El valor mayor es 490
Se encuentra en la fila 2 y en la columna 4
```

Capítulo 94.- Matrices (cantidad de filas y columnas – 3

Problema propuesto:

Crear una matriz de n * m filas (cargar n y m por teclado). intercambiar la primera fila con la segunda. Imprimir luego la matriz.

Vamos a programar:

*Matriz.java

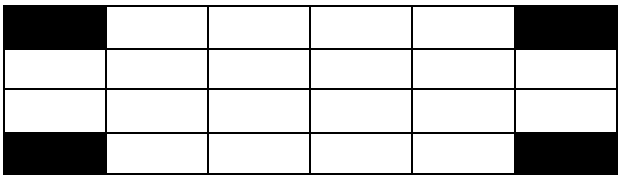
```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Matriz {
6      private Scanner teclado;
7      private int[][] numero;
8
9      public void cargar() {
10         teclado=new Scanner(System.in);
11         int f, c;
12         System.out.print("Ingrese el número de filas de la matriz:");
13         f=teclado.nextInt();
14         System.out.print("Ingrese el número de columnas de la matriz:");
15         c=teclado.nextInt();
16         numero=new int[f][c];
17         for(int x=0; x<f; x++) {
18             System.out.println("Fila :"+(x+1));
19             for(int y=0; y<c; y++) {
20                 System.out.print("Ingrese el valor para columna "+(y+1)+":");
21                 numero[x][y]=teclado.nextInt();
22             }
23         }
24     }
25
26     public void imprimir() {
27
28         for(int x=0; x<numero.length; x++) {
29             for(int y=0; y<numero[0].length; y++) {
30                 System.out.print(numero[x][y]+ " - ");
31             }
32             System.out.println();
33         }
34     }
35
36     public void realizarCambios() {
37         int aux=0;
38         for(int y=0; y<numero[0].length; y++) {
39                 aux=numero[0][y];
40                 numero[0][y]=numero[1][y];
41                 numero[1][y]=aux;
42         }
43     }
44
45     public static void main(String[] args) {
46         Matriz matriz1=new Matriz();
47         matriz1.cargar();
48         System.out.println("Mostrar matriz según el orden de entrada");
49         matriz1.imprimir();
50         matriz1.realizarCambios();
51         System.out.println("Mostrar matriz según cambios realizados");
52         matriz1.imprimir();
53     }
54 }
```

Vamos a ejecutar este será el resultado:

```
Ingrese el número de filas de la matriz:3
Ingrese el número de columnas de la matriz:5
Fila :1
Ingrese el valor para columna 1:50
Ingrese el valor para columna 2:5
Ingrese el valor para columna 3:27
Ingrese el valor para columna 4:400
Ingrese el valor para columna 5:7
Fila :2
Ingrese el valor para columna 1:0
Ingrese el valor para columna 2:67
Ingrese el valor para columna 3:90
Ingrese el valor para columna 4:6
Ingrese el valor para columna 5:97
Fila :3
Ingrese el valor para columna 1:30
Ingrese el valor para columna 2:14
Ingrese el valor para columna 3:23
Ingrese el valor para columna 4:251
Ingrese el valor para columna 5:490
Mostrar matriz según el orden de entrada
50 - 5 - 27 - 400 - 7 -
0 - 67 - 90 - 6 - 97 -
30 - 14 - 23 - 251 - 490 -
Mostrar matriz según cambios realizados
0 - 67 - 90 - 6 - 97 -
50 - 5 - 27 - 400 - 7 -
30 - 14 - 23 - 251 - 490 -
```

Capítulo 95.- Matrices (cantidad de filas y columnas) – 4

Crear una matriz de n * m (cargar por teclado) Imprimir los cuatro valores que se encuentran en los vértices de la misma (mat[0][0], etc.)



Vamos a programar:

Matriz.java

```
package paquete1;

import java.util.Scanner;

public class Matriz {
    private Scanner teclado;
    private int[][] numero;

    public void cargar() {
        teclado=new Scanner(System.in);
        int f, c;
        System.out.print("Ingrese el número de filas de la matriz:");
        f=teclado.nextInt();
        System.out.print("Ingrese el número de columnas de la matriz:");
        c=teclado.nextInt();
        numero=new int[f][c];
        for(int x=0; x<f; x++) {
            System.out.println("Fila :"+(x+1));
            for(int y=0; y<c; y++) {
                System.out.print("Ingrese el valor para columna "+(y+1)+":");
                numero[x][y]=teclado.nextInt();
            }
        }
    }

    public void imprimir() {

        for(int x=0; x<numero.length; x++) {
            for(int y=0; y<numero[0].length; y++) {
                System.out.print(numero[x][y]+ " - ");
            }
            System.out.println();
            System.out.println();
        }
    }

    public void imprimirValoresVertices() {
        System.out.println("Vertice superior izquierdo "+numero[0][0]);
        System.out.println("Vertice superior derechro "+numero[0][numero[0].length-1]);
        System.out.println("Vertice infirior izquierdo "+numero[numero.length-1][0]);
        System.out.println("Vertice inferior derecho "+numero[numero.length-1][numero[0].length-1]);
    }

    public static void main(String[] args) {
        Matriz matriz1=new Matriz();
        matriz1.cargar();
        System.out.println("Mostrar matriz según el orden de entrada");
        matriz1.imprimir();
        matriz1.imprimirValoresVertices();
    }
}
```

Vamos a ejecutar y este será el resultado:

```
Ingrese el número de filas de la matriz:3
Ingrese el número de columnas de la matriz
Fila :1
Ingrese el valor para columna 1:50
Ingrese el valor para columna 2:5
Ingrese el valor para columna 3:27
Ingrese el valor para columna 4:400
Ingrese el valor para columna 5:7
Fila :2
Ingrese el valor para columna 1:0
Ingrese el valor para columna 2:67
Ingrese el valor para columna 3:90
Ingrese el valor para columna 4:6
Ingrese el valor para columna 5:97
Fila :3
Ingrese el valor para columna 1:30
Ingrese el valor para columna 2:14
Ingrese el valor para columna 3:23
Ingrese el valor para columna 4:251
Ingrese el valor para columna 5:490
Mostrar matriz según el orden de entrada
50 - 5 - 27 - 400 - 7 -

0 - 67 - 90 - 6 - 97 -

30 - 14 - 23 - 251 - 490 -

Vertice superior izquierdo 50
Vertice superior derechro 7
Vertice infirior izquierdo 30
Vertice inferior derecho 490
```

Capítulo 96.- Matrices y vectores paralelos – 1

empleados	sueldos			sueldostot
Marcos	540	540	760	
Ana	200	220	250	
Luis	760	760	760	
María	605	799	810	

Problema:

Se tiene la siguiente información:

- Nombre de 4 empleados.
- Ingresos en concepto de sueldo, cobrado por cada empleado, en los últimos 3 meses.

Confeccionar un programa para:

a) Realizar la carga de la información mencionada.
b) Generar un vector que contenga el ingreso acumulado en sueldos en los últimos 3 meses para cada empleado.
c) Mostrar por pantalla el total pagado en sueldos a todos los empleados en los últimos 3 meses.
d) Obtener el nombre del empleado que tuvo mayor ingreso acumulado.

Vamos a programar:

```
package paquete1;

import java.util.Scanner;

public class Matriz {
	private Scanner teclado;
	private String []empleados;
	private int [][]sueldos;
	private int []sueldostot;

	public void cargar() {
		teclado=new Scanner(System.in);
		empleados=new String[4];
		sueldos=new int[4][3];
		sueldostot=new int[4];
		for(int x=0; x<4; x++) {
			System.out.print("Ingrese el nombre del empleado:");
			empleados[x]=teclado.next();
		}for(int x=0; x<4; x++) {
			System.out.println("Ingrese los sueldos de
"+empleados[x]+":");
			for(int y=0; y<3; y++){
				System.out.print("Ingreese el sueldo "+(y+1)+":");
				sueldos[x][y]=teclado.nextInt();
			}
		}
```

```
	}

	public void sumarSueldos() {
		for(int x=0;x<4; x++) {
			int suma=0;
			for(int y=0; y<3; y++) {
				suma=suma+sueldos[x][y];
			}
			sueldostot[x]=suma;
		}
	}

	public void imprimirTotalSueldos() {
		int total=0;
		System.out.println("El total de sueldos por empleado");
		for(int x=0; x<4; x++) {
			total=total+sueldostot[x];
			System.out.println(empleados[x]+"   "+sueldostot[x]);
		}
	}

	public void mayorIngreso() {
		int mayor=0;
		int posicion=0;
		for(int x=0; x<4; x++) {
			if(sueldostot[x]>mayor) {
				mayor=sueldostot[x];
				posicion=x;
			}
		}
		System.out.println("El empleado "+empleados[posicion]+" tiene en
sueldo mayor");
		System.out.print("por un importe de "+sueldostot[posicion]);
	}

	public static void main(String[] args) {
		Matriz matriz1=new Matriz();
		matriz1.cargar();
		matriz1.sumarSueldos();
		matriz1.imprimirTotalSueldos();
		matriz1.mayorIngreso();
	}
}
```

Vamos a ejecutar, este será el resultado:

```
Ingrese el nombre del empleado:Marcos
Ingrese el nombre del empleado:Ana
Ingrese el nombre del empleado:Luis
Ingrese el nombre del empleado:Maria
Ingrese los sueldos de Marcos:
Ingreese el sueldo 1:540
Ingreese el sueldo 2:540
Ingreese el sueldo 3:760
Ingrese los sueldos de Ana:
Ingreese el sueldo 1:200
Ingreese el sueldo 2:220
Ingreese el sueldo 3:250
```

```
Ingrese los sueldos de Luis:
Ingreese el sueldo 1:760
Ingreese el sueldo 2:760
Ingreese el sueldo 3:760
Ingrese los sueldos de Maria:
Ingreese el sueldo 1:605
Ingreese el sueldo 2:779
Ingreese el sueldo 3:810
El total de sueldos por empleado
Marcos   1840
Ana   670
Luis   2280
Maria   2194
El empleado Luis tiene en sueldo mayor
por un importe de 2280
```

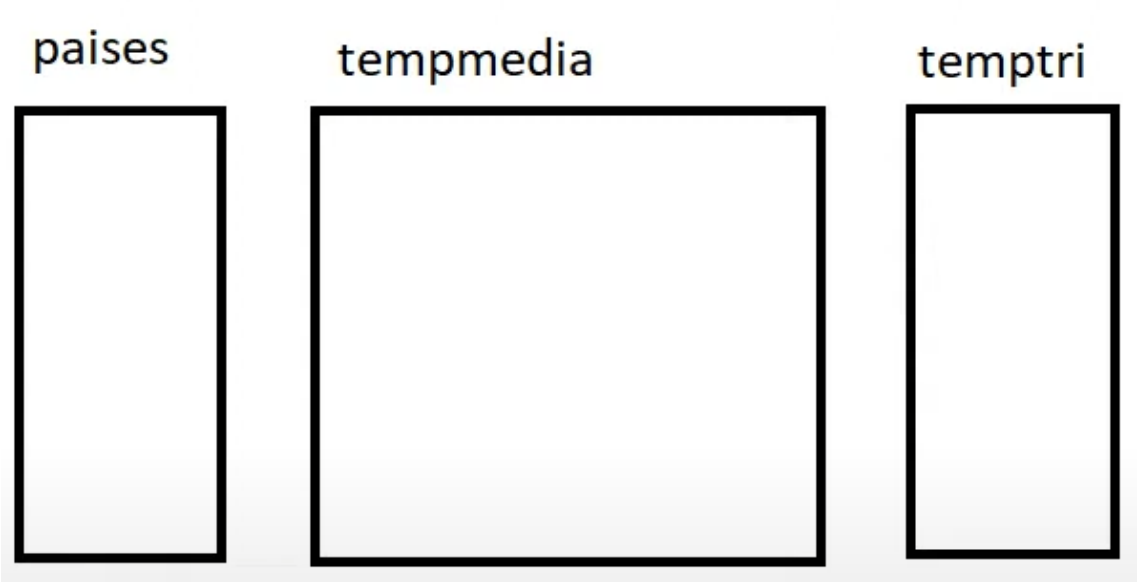
Capítulo 97.- Matrices y vectores paralelos – 2

Problema propuesto

Se desea saber la temperatura media trimestral de cuatro países. Para ello se tiene como dato las temperaturas medias mensuales de dichos países.

Se debe ingresar el nombre del país, y seguidamente las tres temperaturas medias mensuales. Seleccionar las estructuras de datos adecuadas para el almacenamiento de los datos en memoria.

a- Cargar por teclado los nombres de los países y las temperaturas medias mensuales.
b- Imprimir los nombres de los países y las temperaturas medias mensuales.
c- Calcular la temperatura media trimestral de cada país.
d- Imprimir los nombres de los países y las temperaturas medias trimestrales.
e- Imprimir el nombre del país con la temperatura media trimestral mayor.



Vamos a programar:

```
package paquete1;

import java.util.Scanner;

public class Temperaturas {
	private Scanner teclado;
	private String[] paises;
	private int[][] tempmedia;
	private int[] temptri;

	public void cargar() {
		teclado=new Scanner(System.in);
		paises=new String[4];
		tempmedia=new int[4][3];
		temptri=new int[4];
		for(int x=0; x<4; x++) {
			System.out.print("Ingrese el país:");
			paises[x]=teclado.next();
			for(int y=0; y<3; y++) {
				System.out.print("Ingrese la nota media por mes:");
```

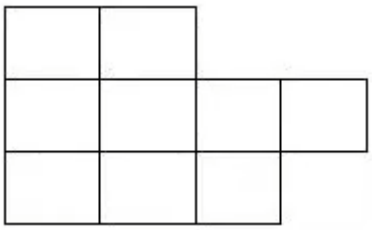
```
                        tempmedia[x][y]=teclado.nextInt();
                }
        }
    }

    public void TempTrimestral() {
        for(int x=0; x<4; x++) {
            int suma=0;
            for(int y=0; y<3; y++) {
                suma=suma+tempmedia[x][y];
            }
            temptri[x]=suma/3;
        }
    }

    public void imprimirMediasTrimestrales() {
        for(int x=0; x<4; x++) {
            System.out.println(paises[x]+"   "+temptri[x]);
        }
    }

    public void temperaturaMayor() {
        float max=0;
        int pos=0;
        for(int x=0; x<4; x++) {
            if(temptri[x]>max) {
                max=temptri[x];
                pos=x;
            }
        }
        System.out.println("El país con mayor temperatura es
"+paises[pos]);
        System.out.print("con una temperatura media de "+temptri[pos]);
    }

    public static void main(String[] args) {
        Temperaturas temperatura1=new Temperaturas();
        temperatura1.cargar();
        temperatura1.TempTrimestral();
        temperatura1.imprimirMediasTrimestrales();
        temperatura1.temperaturaMayor();

    }
}
```

Vamos a ejecutar y este será el resultado:

```
Ingrese el país:Chile
Ingrese la nota media por mes:10
Ingrese la nota media por mes:11
Ingrese la nota media por mes:12
Ingrese el país:Argentina
Ingrese la nota media por mes:15
Ingrese la nota media por mes:16
Ingrese la nota media por mes:17
Ingrese el país:Uruguay
Ingrese la nota media por mes:20
Ingrese la nota media por mes:21
Ingrese la nota media por mes:22
```

```
Ingrese el país:Brasil
Ingrese la nota media por mes:25
Ingrese la nota media por mes:25
Ingrese la nota media por mes:27
Chile   11
Argentina   16
Uruguay   21
Brasil   25
El país con mayor temperatura es Brasil
con una temperatura media de 25
```

Capítulo 98.- Matrices irregulares – 1



Primero creamos la cantidad de filas dejando vacío el espacio que indica la cantidad de columnas:

mat=new int[3][];

Luego debemos ir creando cada fila de la matriz indicando la cantidad de elementos de la respectiva fila:

mat[0]=new int[2];
mat[1]=new int[4];
mat[2]=new int[3]

Problema:

Confeccionaremos un programa que permita crear una matriz irregular y luego imprimir la matriz en forma completa.

Vamos a programar:

Matriz.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Matriz {
6      private Scanner teclado;
7      private int [][] numero;
8
9      public void cargar() {
10         teclado=new Scanner(System.in);
11         int filas=0;
12         int columnas=0;
13         System.out.print("Ingrese el número de filas:");
14         filas=teclado.nextInt();
15         numero=new int[filas][];
16         for(int x=0; x<filas; x++) {
17             System.out.print("Ingrese el número de columnas:");
18             columnas=teclado.nextInt();
19             numero[x]=new int[columnas];
20                 for(int y=0; y<numero[x].length; y++) {
21                     System.out.print("Ingrese un número:");
22                     numero[x][y]=teclado.nextInt();
23                 }
24         }
25     }
```

```
26      public void imprimir() {
27          for (int x=0; x<numero.length; x++) {
28              for(int y=0; y<numero[x].length; y++) {
29                  System.out.print(numero[x][y]+"  ");
30              }
31              System.out.println();
32          }
33      }
34
35      public static void main(String[] args) {
36          Matriz matriz1=new Matriz();
37          matriz1.cargar();
38          matriz1.imprimir();
39      }
40  }
```

Vamos a ejecutar, este será el resultado:

```
Ingrese el número de filas:3
Ingrese el número de columnas:2
Ingrese un número:1
Ingrese un número:2
Ingrese el número de columnas:4
Ingrese un número:1
Ingrese un número:2
Ingrese un número:3
Ingrese un número:4
Ingrese el número de columnas:3
Ingrese un número:1
Ingrese un número:2
Ingrese un número:3
1  2
1  2  3  4
1  2  3
```

Capítulo 99.- Matrices irregulares – 2

Problema propuesto:

Confeccionar una clase para administrar una matriz irregular de 5 filas y 1 columna la primera fila, 2 columnas la segunda fila y así sucesivamente hasta 5 columnas la última fila (crearla sin la intervención del operador).

Realiza la carga por teclado e imprimir posteriormente.

Vamos a programar:

*Matriz.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Matriz {
6      private Scanner teclado;
7      private int [][] numero;
8
9      public void cargar() {
10         teclado=new Scanner(System.in);
11         numero=new int[5][];
12         for(int x=0; x<5; x++) {
13             numero[x]= new int[x+1];
14         }
15         for(int x=0; x<numero.length; x++) {
16             System.out.println("Fila: "+x);
17             for(int y=0; y<numero[x].length; y++) {
18                 System.out.print("Ingrese un valor:");
19                 numero[x][y]=teclado.nextInt();
20             }
21         }
22     }
23
24     public void imprimir() {
25         for(int x=0; x<numero.length; x++) {
26             for(int y=0; y<numero[x].length; y++) {
27                 System.out.print(numero[x][y]+ "  ");
28             }
29             System.out.println();
30         }
31     }
32
33     public static void main(String[] args) {
34         Matriz matriz1 = new Matriz();
35         matriz1.cargar();
36         matriz1.imprimir();
37     }
38 }
```

Vamos a ejecutar, este será el resultado:

```
Fila: 0
Ingrese un valor:1
Fila: 1
Ingrese un valor:1
Ingrese un valor:2
```

```
Ingrese un valor:1
Ingrese un valor:2
Ingrese un valor:3
Fila: 3
Ingrese un valor:1
Ingrese un valor:2
Ingrese un valor:3
Ingrese un valor:4
Fila: 4
Ingrese un valor:1
Ingrese un valor:2
Ingrese un valor:3
Ingrese un valor:4
Ingrese un valor:5
1
1  2
1  2  3
1  2  3  4
1  2  3  4  5
```

Capítulo 100.- Matrices irregulares – 3

Problema propuesto:

Confeccionar una clase para administrar los días que han faltado los 3 empleados de una empresa.

Definir un vector de 3 elementos de tipo String para cargar los nombres y una matriz irregular para cargar los días que han faltado cada empleado (cargar el número de días que faltó).
Cada fila de la matriz representan los días que falto cada empleado.

Mostrar los empleados con la cantidad de inasistencias.

Cuál empleado faltó menos días.

Vamos a programar:

```
package paquete1;

import java.util.Scanner;

public class Matriz {
	private Scanner teclado;
	private String[] nombres;
	private int[][] dias;

	public void cargar() {
		int faltas;
		teclado=new Scanner(System.in);
		nombres=new String[3];
		dias=new int[3][];
		for(int x=0; x<3; x++) {
			System.out.print("Nombre del empleado:");
			nombres[x]=teclado.next();
			System.out.print("Cuantos días a faltado:");
			faltas=teclado.nextInt();
			dias[x]=new int[faltas];
			System.out.println("Las faltas en el trabajo de
"+nombres[x]+":");
			for(int y=0; y<faltas; y++) {
				System.out.print("Día del mes que hiciste falta en
el trabajo:");
				dias[x][y]=teclado.nextInt();
			}
		}
	}

	public void imprimir() {
		for(int x=0; x<nombres.length; x++) {
			System.out.print(nombres[x]+" Ha faltado en el trabajo los
días:");
			for(int y=0; y<dias[x].length; y++) {
				System.out.print(dias[x][y]+ ", ");
			}
			System.out.println();
		}
	}

	public void masFaltas(){
```

```
		int max=0;
		int pos=0;
		for(int x=0; x<3; x++) {
			if(dias[x].length>max) {
				max=dias[x].length;
				pos=x;
			}
		}
		System.out.print("El empleado "+nombres[pos]+" con mayor días de
falta:"+max);
	}

	public static void main(String[] args) {
		Matriz matriz1=new Matriz();
		matriz1.cargar();
		matriz1.imprimir();
		matriz1.masFaltas();
	}
}
```

Si ejecutamos este será el resultado:

```
Nombre del empleado:Juan
Cuantos días a faltado:3
Las faltas en el trabajo de Juan:
Día del mes que hiciste falta en el trabajo:2
Día del mes que hiciste falta en el trabajo:7
Día del mes que hiciste falta en el trabajo:12
Nombre del empleado:Luis
Cuantos días a faltado:10
Las faltas en el trabajo de Luis:
Día del mes que hiciste falta en el trabajo:2
Día del mes que hiciste falta en el trabajo:6
Día del mes que hiciste falta en el trabajo:9
Día del mes que hiciste falta en el trabajo:10
Día del mes que hiciste falta en el trabajo:12
Día del mes que hiciste falta en el trabajo:18
Día del mes que hiciste falta en el trabajo:22
Día del mes que hiciste falta en el trabajo:24
Día del mes que hiciste falta en el trabajo:27
Día del mes que hiciste falta en el trabajo:30
Nombre del empleado:Ana
Cuantos días a faltado:5
Las faltas en el trabajo de Ana:
Día del mes que hiciste falta en el trabajo:4
Día del mes que hiciste falta en el trabajo:8
Día del mes que hiciste falta en el trabajo:12
Día del mes que hiciste falta en el trabajo:22
Día del mes que hiciste falta en el trabajo:27
Juan Ha faltado en el trabajo los días:2, 7, 12,
Luis Ha faltado en el trabajo los días:2, 6, 9, 10, 12, 18, 22, 24, 27, 30,
Ana Ha faltado en el trabajo los días:4, 8, 12, 22, 27,
El empleado Luis con mayor días de falta:10
```

Capítulo 101.- Constructor de la clase – 1

Es un método.

El constructor tiene las siguientes características:

- Tiene el mismo nombre de la clase.
- Es el primer método que se ejecuta.
- Se ejecuta en forma automática.
- No puede retornar datos.
- Se ejecuta una única vez.
- Un constructor tiene por objetivo inicializar atributos.
- Podemos tener varios constructores, pero con distintos tipos o cantidad de parámetros.

Problema:

Se desea guardar los sueldos de 5 operarios en un vector. Realiza la creación y carga del vector en el constructor.

Vamos a programar:

Vector.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Vector {
6      Scanner teclado;
7      int[] sueldo;
8
9      public Vector() {
10         teclado=new Scanner(System.in);
11         sueldo=new int[5];
12         for(int x=0; x<5; x++) {
13             System.out.print("Ingrese el sueldo:");
14             sueldo[x]=teclado.nextInt();
15         }
16     }
17
18     public void imprimir() {
19         for(int x=0; x<sueldo.length; x++) {
20             System.out.println(sueldo[x]);
21         }
22     }
23
24     public static void main(String[] args) {
25         Vector vector1=new Vector();
26         vector1.imprimir();
27     }
28 }
```

El constructor es un método que tiene el mismo nombre que la clase, este se ejecuta automáticamente al instanciar una clase.

Vamos a ejecutar, este será el resultado:

```
Ingrese el sueldo:450
Ingrese el sueldo:830
Ingrese el sueldo:348
Ingrese el sueldo:630
Ingrese el sueldo:560
450
830
348
630
560
```

Capítulo 102.- Constructor de la clase – 2

Problema:

Plantear una clase llamada Alumno y definir como atributos su nombre y su edad. En el constructor realizar la carga de datos. Definir otros dos métodos para imprimir los datos ingresados y un mensaje se es mayor o no de edad (edad>=18)

Vamos a programar:

*Alumnos.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Alumnos {
6      private Scanner teclado;
7      private String[] nombre;
8      private int[] edad;
9
10     public Alumnos() {
11         teclado=new Scanner(System.in);
12         nombre=new String[5];
13         edad=new int[5];
14         for(int x=0; x<5; x++) {
15             System.out.print("Ingrese el nombre del empleado:");
16             nombre[x]=teclado.next();
17             System.out.print("Ingrese su edad:");
18             edad[x]=teclado.nextInt();
19         }
20     }
21
22     public void imprimir() {
23         System.out.println("Relación de nombes y sus edades");
24         for(int x=0; x<nombre.length; x++) {
25             System.out.println(nombre[x]+"    "+edad[x]);
26         }
27     }
28
29     public void mayorEdad() {
30         System.out.println("Relación de alumnos mayores de edad");
31         for(int x=0; x<nombre.length; x++) {
32             if(edad[x]>=18) {
33                 System.out.println(nombre[x]+"       "+edad[x]);
34             }
35         }
36     }
37
38     public static void main(String[] args) {
39         Alumnos alumno1=new Alumnos();
40         alumno1.imprimir();
41         alumno1.mayorEdad();
42     }
43 }
```

Si ejecutamos este será el resultado:

```
Ingrese el nombre del empleado:Carlos
Ingrese su edad:22
Ingrese el nombre del empleado:Ana
Ingrese su edad:16
Ingrese el nombre del empleado:Luis
Ingrese su edad:18
Ingrese el nombre del empleado:Juan
Ingrese su edad:32
Ingrese el nombre del empleado:Pedro
Ingrese su edad:15
Relación de nombes y sus edades
Carlos    22
Ana    16
Luis    18
Juan    32
Pedro    15
Relación de alumnos mayores de edad
Carlos       22
Luis       18
Juan       32
```

Capítulo 103.- Constructor de la clase – 3

Problema:

Plantear una clase TablasMultiplicar. Defina dos constructores uno con un parámetro que lleve un entero indicando que tabla queremos ver y otro con dos enteros que indique el primero que tabla queremos ver y el segundo parámetro indica cuantos términos mostrar.

Si no llega la cantidad de términos a mostrar inicializar en 10 los términos a mostrar.

Vamos a programar:

TablaMultiplicar.java

```
1  package paquete1;
2
3  public class TablaMultiplicar {
4      private int tabla;
5      private int terminos;
6
7      public TablaMultiplicar(int tabla) {
8          this.tabla=tabla;
9          this.terminos=10;
10     }
11
12     public TablaMultiplicar(int tabla, int terminos) {
13         this.tabla=tabla;
14         this.terminos=terminos;
15     }
16
17     public void imprimir() {
18         System.out.println("Tabla de multiplicar del "+tabla);
19         for (int x=0; x<=terminos; x++) {
20             int resultado=x*tabla;
21             System.out.println(tabla+" * "+ x + " = " + resultado);
22         }
23     }
24
25
26     public static void main(String[] args) {
27         TablaMultiplicar tabla1=new TablaMultiplicar(2);
28         tabla1.imprimir();
29         TablaMultiplicar tabla2=new TablaMultiplicar(3, 12);
30         tabla2.imprimir();
31     }
32 }
```

En ese ejemplo tenemos 2 constructores la diferencia son los parámetros de cada constructor, si lo instanciamos con un parámetro se ejecutará el primero en cambio sí lo instanciamos con dos parámetos se ejecutará el segundo.

Vamos a ejecutar este será el resultado:

```
Tabla de multiplicar del 2
2 * 0 = 0
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
Tabla de multiplicar del 3
3 * 0 = 0
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
3 * 11 = 33
3 * 12 = 36
```

El primer constructor imprime la tabla hasta el valor 10, ya que lo tiene como valor por defecto.

El segundo constructor además del parámetro del número de tabla le pasamos hasta que valor queremos que multiplique.

Capítulo 104.- Constructor de la clase – 4

Problema propuesto:

Confeccionar una clase que represente un empleado. Define como atributos su nombre y su sueldo. En el constructor cargar los atributos y luego en otro método imprimir sus datos y por último uno que imprima un mensaje si debe pagar impuestos (si el sueldo supera a 3000).

Vamos a programar:

*Empleado.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Empleado {
6      private Scanner teclado;
7      private String nombre;
8      private int sueldo;
9
10     public Empleado(String nombre, int sueldo) {
11         this.nombre=nombre;
12         this.sueldo=sueldo;
13     }
14
15     public void imprimir() {
16         System.out.println(this.nombre+" con un sueldo de "+this.sueldo);
17     }
18
19     public void pagar() {
20         if(this.sueldo>3000) {
21             System.out.print("Debe pagar impuestos");
22         }else {
23             System.out.print("No debe pagar impuestos");
24         }
25         System.out.println();
26     }
27
28     public static void main(String[] args) {
29         Empleado empleado1 = new Empleado("Juan", 2500);
30         empleado1.imprimir();
31         empleado1.pagar();
32         Empleado empleado2 = new Empleado("Carlos", 4000);
33         empleado2.imprimir();
34         empleado2.pagar();
35     }
36 }
```

Si ejecutamos este será el resultado:

```
Juan con un sueldo de 2500
No debe pagar impuestos
Carlos con un sueldo de 4000
Debe pagar impuestos
```

Otra forma de hacerlo si queremos introducir los datos por teclado:

*Empleado.java

```java
import java.util.Scanner;

public class Empleado {
    private Scanner teclado;
    private String nombre;
    private float sueldo;

    public Empleado() {
        teclado=new Scanner(System.in);
        System.out.print("Ingrese el nombre del empleado:");
        nombre=teclado.next();
        System.out.print("Ingrese el sueldo:");
        sueldo=teclado.nextFloat();
    }

    public void imprimir() {
        System.out.println("Nombre:"+nombre);
        System.out.println("Sueldo:"+sueldo);
    }

    public void pagaImpuestos() {
        if (sueldo>3000) {
            System.out.print("Paga impuestos");
        }
    }


    public static void main(String[] ar) {
        Empleado empleado1=new Empleado();
        empleado1.imprimir();
        empleado1.pagaImpuestos();
    }
}
```

Capítulo 105.- Constructor de la clase – 5

Problema propuesto:

Implementar la clase operaciones. Se debe cargar dos valores enteros en el constructor, calcular su suma, resta, multiplicación y división, cada una en un método. imprimir dichos resultados.

Vamos a programar:

Operaciones.java

```java
package paquete1;

import java.util.Scanner;

public class Operaciones {
    private Scanner teclado;
    private int num1;
    private int num2;

    public Operaciones() {
        teclado=new Scanner(System.in);
        System.out.print("Ingrese un número:");
        num1=teclado.nextInt();
        System.out.print("Ingrese otro número:");
        num2=teclado.nextInt();
    }

    public void suma() {
        int suma=num1+num2;
        System.out.println("La suma es "+suma);
    }

    public void resta() {
        int resta=num1-num2;
        System.out.println("La suma es "+resta);
    }

    public void multiplicacion() {
        int multiplicacion=num1*num2;
        System.out.println("La suma es "+multiplicacion);
    }

    public void division() {
        int division=num1/num2;
        System.out.println("La suma es "+division);
    }

    public static void main(String[] args) {
        Operaciones operacion1=new Operaciones();
        operacion1.suma();
        operacion1.resta();
        operacion1.multiplicacion();
        operacion1.division();
    }
}
```

Si ejecutamos este será el resultado:

```
Ingrese un número:10
Ingrese otro número:5
La suma es 15
La suma es 5
La suma es 50
La suma es 2
```

Capítulo 106.- Clase String – 1

Métodos más comunes:

- boolean equals(String s1)

Retorna true si el contenido del parámetro s1 es exactamente igual a la cadena de caracteres del objeto que llama al método equals.

- boolean equalsIgnoreCase(String s1)

El funcionamiento es casi exactamente igual que el método equals con la diferencia que no tiene en cuenta mayúsculas y minúsculas.

- int compareTo(String s1)

El método retorna un 0 si el contenido de s1 es exactamente igual al String contenido por el objeto que llama al método compareTo. Retorna el valor >0 si el contenido del String que llama al método compareTo es mayor alfabéticamente al parámetro s1.

- char sharAt(int pos)

Retorna el carácter del String, llega al método la posición del carácter a extraer.

- lint length()

Retorna la cantidad de caracteres almacenados en el String.

- String substring(int pos1, int pos2)

Retorna un substring a partir de la posición indicada en el parámetro pos1 hasta la posición pos2 sin incluir dicha posición.

- int indexOf(String s1)

Retorna -1 si el String que le pasamos como parámetro no está contenida en la cadena del objeto que llama al método. En caso que se encuentre contenido el String s1 retorna la posición donde comienza a repetirse.

- String toUpperCase()

Retorna un String con el contenido convertido todo a mayúsculas.

- String toLowerCase()

Retorna un String con el contenido convertido todo a minúsculas.

Vamos a ver un ejemplo:

PruebaString.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class PruebaString {
6
7      public static void main(String[] args) {
8          Scanner teclado=new Scanner(System.in);
9          String cad1;
10         String cad2;
11         System.out.print("Ingrese la primera cadena:");
12         cad1=teclado.nextLine();
```

```
        System.out.print("Ingrese la segunda cadena:");
        cad2=teclado.nextLine();
        if (cad1.equals(cad2)) {
            System.out.println(cad1+" es exactamente igual a "+cad2);
        } else {
            System.out.println(cad1+" no es exactamente igual a "+cad2);
        }
        if (cad1.equalsIgnoreCase(cad2)) {
            System.out.println(cad1+" es igual a "+cad2+" sin tener en cuenta mayúsculas y minúsculas");
        }else {
            System.out.println(cad1+" no es igual a "+cad2+" sin tener en cuenta mayúsculas y minúsculas");
        }
        if (cad1.compareTo(cad2)==0) {
            System.out.println(cad1+" es exactamente igual a "+cad2);
        }else {
            if (cad1.compareTo(cad2)>0) {
                System.out.println(cad1+" es mayor alfabéticamente que "+cad2);
            }else {
                System.out.println(cad2+ " es mayor alfabéticamente que "+cad1);
            }
        }
        char caracter=cad1.charAt(0);
        System.out.println("El primer caracter de "+cad1+" es "+caracter);
        int largo=cad1.length();
        System.out.println("El largo de la cadena "+cad1+" es "+largo);
        System.out.println("El último caracter de "+cad1+ " es "+cad1.charAt(cad1.length()-1));
        String cad3=cad1.substring(0,3);
        System.out.println("Los primeros tres caracteres de "+cad1+ " son "+cad3);
    }
}
```

Si ejecutamos este será el resultado:

```
Ingrese la primera cadena:juan
Ingrese la segunda cadena:ana
juan no es exactamente igual a ana
juan no es igual a ana sin tener en cuenta mayúsculas y minúsculas
juan es mayor alfabéticamente que ana
El primer caracter de juan es j
El largo de la cadena juan es 4
El último caracter de juan es n
Los primeros tres caracteres de juan son jua
```

Capítulo 107.- Clase String – 2

Problema propuesto:

Realiza un clase, que permita cargar una dirección de mail en el constructor, luego en otro método mostrar un mensaje si contiene el carácter '@'.

Vamos a programar:

*Mail.java

```
3  import java.util.Scanner;
4
5  public class Mail {
6      private Scanner teclado;
7      private String email;
8
9      public Mail() {
10         teclado=new Scanner(System.in);
11         System.out.print("Ingrese un correo electónico:");
12         email=teclado.nextLine();
13     }
14
15     public void comprobar() {
16         int caracteres;
17         boolean tiene=false;
18         caracteres=email.length();
19         for(int x=0; x<caracteres; x++) {
20             if(email.charAt(x)=='@') {
21                 tiene=true;
22             }
23         }
24
25         if(tiene) {
26             System.out.print("Este correo contiene el caracter '@'");
27         }else {
28             System.out.print("Este correo no contiene el caracter '@'");
29         }
30
31     }
32
33     public static void main(String[] args) {
34         Mail email1=new Mail();
35         email1.comprobar();
36     }
37 }
```

Vamos a ejecutar introduciendo un correo sin '@'.

```
Ingrese un correo electónico:diego.hotmail.com
Este correo no contiene el caracter '@'
```

Ahora ejecutamos de nuevo introduciendo un correo que contiene '@'.

```
Ingrese un correo electónico:diego@hotmail.com
Este correo contiene el caracter '@'
```

Capítulo 108.- Clase String – 3

Problema propuesto:

Cargar un String por teclado e implementar los siguientes métodos:

a) Imprimir la primera mitad de los caracteres de la cadena.
b) Imprimir el último carácter.
c) Imprimirlo en forma inversa.
d) Imprimir cada carácter del String separado por un guion.
e) Imprimir la cantidad de vocales almacenadas.
f) Implementar un método que verifique si la cadena se lee igual de izquierda a derecha tanto como de derecha a izquierda (ej. neuquen se lee igual en las dos direcciones).

Vamos a programar:

```
package paquete1;

import java.util.Scanner;

public class Cadena {
	private Scanner teclado;
	private String texto;

	public Cadena() {
		teclado=new Scanner(System.in);
		System.out.print("Ingrese un String:");
		texto=teclado.next();
	}

	public void primeraMitad() {
		int longitud=((int)texto.length()/2);
		System.out.println("Imprimir primera mitad del
texto:"+texto.substring(0, longitud));
	}

	public void ultimoCaracter() {
		System.out.println("Imprimir el último caracter del
texto:"+texto.charAt(texto.length()-1));
	}

	public void imprimirInversa() {
		System.out.print("Imprimir de forma inversa:");
		for(int x=texto.length()-1; x>=0; x--) {
			System.out.print(texto.charAt(x));
		}
		System.out.println();
	}

	public void imprimirGuion() {
		for(int x=0; x<texto.length(); x++) {
			System.out.print(texto.charAt(x));
			if (x<texto.length()-1) {
				System.out.print(" - ");
			}
		}
		System.out.println();
	}
```

```
	public void cantidadVocales() {
		int cantidad=0;
		String texto1=texto.toLowerCase();
		for(int x=0; x<texto1.length(); x++) {
			if(texto1.charAt(x)=='a' || texto1.charAt(x)=='e' ||
texto1.charAt(x)=='i' || texto1.charAt(x)=='o' || texto1.charAt(x)=='u') {
				cantidad++;
			}
		}
		System.out.println("La palabra '"+texto+"' tiene "+cantidad+"
vocales");
	}

	public void capicua() {
		boolean capi=true;
		int longitud=((int)texto.length()/2);
		for(int x=0; x<longitud; x++) {
			if(texto.charAt(x)!=texto.charAt(texto.length()-x-1)){
				capi=false;
				break;
			}
		}
		if(capi) {
			System.out.println("La palabar '"+texto+"' es capicula");
		}else {
			System.out.println("La palabar '"+texto+"' no es
capicula");
		}
	}

	public static void main(String[] args) {
		Cadena cadena1=new Cadena();
		cadena1.primeraMitad();
		cadena1.ultimoCaracter();
		cadena1.imprimirInversa();
		cadena1.imprimirGuion();
		cadena1.cantidadVocales();
		cadena1.capicua();
	}
}
```

Vamos a ejecutar este será el resultado:

```
Ingrese un String:neuquen
Imprimir primera mitad del texto:neu
Imprimir el último caracter del texto:n
Imprimir de forma inversa:neuquen
n - e - u - q - u - e - n
La palabra 'neuquen' tiene 4 vocales
La palabar 'neuquen' es capicula
```

Capítulo 109.- Clase String – 4

Problema propuesto:

Desarrollar un programa que solicite la carga de una clave. La clave debe tener dos métodos uno para la carga y otro que muestre si la clave es la correcta (la clave es "123abc")

Vamos a programar:

Clave.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Clave {
6      private Scanner teclado;
7      private String cla;
8
9      public Clave() {
10         teclado=new Scanner(System.in);
11         System.out.print("Ingrese la clave:");
12         cla=teclado.nextLine();
13     }
14
15     public void verificar() {
16         if (cla.equals("123abc")) {
17             System.out.print("La clave es correcta");
18         }else {
19             System.out.print("La clave no es correcta");
20         }
21     }
22
23     public static void main(String[] args) {
24         Clave clave1=new Clave();
25         clave1.verificar();
26     }
27 }
```

Este será el resultado cundo ejecutemos:

```
Ingrese la clave:123abc
La clave es correcta
```

Volvemos a ejecutar con una clave errónea:

```
Ingrese la clave:456def
La clave no es correcta
```

Capítulo 110.- Clase String – 5

Problema propuesto:

Confeccionar un programa que permita cargar los nombres de 5 personas y sus mail, luego implementar los siguientes módulos:

a) Mostrar por pantallas los datos.
b) Consulta el mail ingresando su nombre.
c) Mostrar los mail que no tienen el carácter @.

Vamos a programar:

```
package paquete1;

import java.util.Scanner;

public class Contactos {
	private Scanner teclado;
	private String[] nombre;
	private String[] mail;

	public Contactos() {
		teclado=new Scanner(System.in);
		nombre=new String[5];
		mail=new String[5];
		for(int x=0; x<5; x++) {
			System.out.print("Ingrese el nombre:");
			nombre[x]=teclado.nextLine();
			System.out.print("Ingrese su mail:");
			mail[x]=teclado.nextLine();
		}
	}

	public void imprimir() {
		System.out.println("Listado de contactos");
		for(int x=0; x<5; x++) {
			System.out.println(nombre[x]+"....."+mail[x]);
		}
	}

	public void consulta() {
		String nom;
		System.out.print("Que nombre desea consultar:");
		nom=teclado.nextLine();
		for(int x=0; x<5; x++) {
			if(nombre[x].equals(nom)) {
				System.out.println(nombre[x]+" tiene el mail
"+mail[x]);
			}
		}
	}

	public void errorMail() {
		int cantidad;
		for(int x=0; x<5; x++) {
			cantidad=0;
			for(int y=0; y<mail[x].length(); y++) {
				if(mail[x].charAt(y)=='@') {
```

```
                                cantidad++;
                        }
                    }
                    if(cantidad==0) {
                        System.out.println("Erro en el mail, falta la
'@':"+mail[x]);
                    }
                }
            }

        public static void main(String[] args) {
            Contactos contacto1=new Contactos();
            contacto1.imprimir();
            contacto1.consulta();
            contacto1.errorMail();
        }
}
```

Vamos a ejecutar, este será el resultado:

```
Ingrese el nombre:Ana
Ingrese su mail:ana@gmail.com
Ingrese el nombre:Juan
Ingrese su mail:juan@gmail.com
Ingrese el nombre:Pedro
Ingrese su mail:pedro.gmail.com
Ingrese el nombre:Luis
Ingrese su mail:luis.gmail.com
Ingrese el nombre:Mauricio
Ingrese su mail:mauricio@gmail.com
Listado de contactos
Ana.....ana@gmail.com
Juan.....juan@gmail.com
Pedro.....pedro.gmail.com
Luis.....luis.gmail.com
Mauricio.....mauricio@gmail.com
Que nombre desea consultar:Ana
Ana tiene el mail ana@gmail.com
Erro en el mail, falta la '@':pedro.gmail.com
Erro en el mail, falta la '@':luis.gmail.com
```

Otra forma de realizar el método erroMail()

```
public void errorMail() {
    for(int x=0; x<5; x++) {
        if(mail[x].indexOf("@")==-1) {
            System.out.println("Erro en el mail, falta la '@':"+mail[x]);
        }

    }
}
```

No tenemos necesidad de recorrer carácter a carácter.

Capítulo 111.- Clase String – 6

Problema propuesto:

Codifique un programa que permita cargar una oración por teclado, luego mostrar cada palabra ingresada en un línea distinta.

Por ejemplo si cargo:

La mañana está fría.

Debe aparecer:

La
mañana
está
fría.

Vamos a programar:

*Oracion.java

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Oracion {
6      private Scanner teclado;
7      private String texto;
8
9      public Oracion() {
10         teclado=new Scanner(System.in);
11         System.out.print("Ingrese una oración:");
12         texto=teclado.nextLine();
13     }
14
15     public void imprimir() {
16         for(int x=0; x<texto.length(); x++) {
17             if(texto.charAt(x)!=' ') {
18                 System.out.print(texto.charAt(x));
19             }else {
20                 System.out.println();
21             }
22         }
23     }
24
25     public static void main(String[] args) {
26         Oracion oracion1 = new Oracion();
27         oracion1.imprimir();
28     }
29 }
```

Si ejecutamos, este será el resultado:

```
Ingrese una oración:La mañana está fria.
La
mañana
está
fria.
```

Capítulo 112.- Colaboración de clases – 1

Normalmente un problema resuelto con la metodología de programación orientada a objetos no interviene una sola clase, sino que hay muchas clases que interactúan y se comunican.

Problema:

Un banco tiene 3 clientes que pueden hacer depósitos y extracciones. También el banco requiere que al final del día calcule la cantidad de dinero que hay depositado.

Lo primero que hacemos es identificar las clases:

Podemos identificar la clase Cliente y la clase Banco.

Luego debemos definir los atributos y los métodos de cada clase:

Cliente:

- atributos
  - nombre
  - importe
- métodos
  - constructor
  - depositar
  - extraer
  - retornarImporte

Banco:

- atributos
  - 3 Clientes (3 objetos de la clase Cliente)
- métodos
  - constructor
  - operar
  - depositosTotales

Vamos a programar:

En teste proyecto vamos a crear dos clases, el código de la clase Cliente:

```
1  package paquete1;
2
3  public class Cliente {
4      private String nombre;
5      private int importe;
6
7      public Cliente(String nom) {
8          nombre=nom;
9          importe=0;
10     }
11
12     public void depositar(int i) {
13         importe=importe+i;
14     }
15
16     public void extraer(int i) {
17         importe=importe-i;
18     }
19
```

```
20      public int retornarImporte() {
21          return importe;
22      }
23
24      public void imprimir() {
25          System.out.println(nombre+" Tiene depositado la suma de "+importe);
26      }
27  }
```

En la línea 3 definimos las variables nombre e importe.

En la línea 7 creamos el constructor para añadir en nombre del cliente con un importe de 0.

En la línea 12 creamos un método llamado depositar para cuando el cliente realiza un ingreso.

En la línea 15 creamos un método llamado extraer para cuando el cliente retira dinero.

En la línea 20 creamos un método llamado retornarImporte para saber el saldo de cada cliente.

en la línea 24 creamos un método llamado imprimir imprimimos el nombre del cliente con su saldo en cuenta.

Se puede acceder a la variable importe porque está definida como global, si no fuera global haríamos uso del método:

```
public void imprimir() {
    System.out.println(nombre+" Tiene depositado la suma de "+retornarImporte());
}
```

Ahora vamos con la clase Banco:

```
1  package paquete1;
2
3  public class Banco {
4      private Cliente cliente1, cliente2, cliente3;
5
6      public Banco() {
7          cliente1=new Cliente("Ana");
8          cliente2=new Cliente("Juan");
9          cliente3=new Cliente("Pedro");
10     }
11
12     public void operar() {
13         cliente1.depositar(100);
14         cliente2.depositar(150);
15         cliente3.depositar(300);
16         cliente1.extraer(100);
17     }
18
19     public void depositosTotales() {
20         int total=cliente1.retornarImporte() +
21                 cliente2.retornarImporte() +
22                 cliente3.retornarImporte();
23         System.out.println("El total de dinero en banco:"+total);
24         cliente1.imprimir();
25         cliente2.imprimir();
26         cliente3.imprimir();
27     }
28
```

```
29      public static void main(String[] args) {
30          Banco banco1=new Banco();
31          banco1.operar();
32          banco1.depositosTotales();
33      }
34  }
```

Primero hay que tener en cuenta que esta clase es la que vamos a ejecutar, por este motivo está el codigo que vemos a partir de la línea 29.

En la línea 4 definimos 3 objetos de la clase Cliente.

En la línea 4 en el constructor a cada cliente le asignamos un nombre y con un saldo de 0, como podemos apreciar en la clase Clientes.

```
 7      public Cliente(String nom) {
 8          nombre=nom;
 9          importe=0;
10      }
```

En la línea 12 realizamos operaciones de depositar y extraer de los clientes, según los métodos de la clase Cliente.

```
12      public void depositar(int i) {
13          importe=importe+i;
14      }
15
16      public void extraer(int i) {
17          importe=importe-i;
18      }
```

En la línea 19 con el método depositosTotales, realizamos la suma del dinero depositado de todos los clientes, para saber el dinero que tiene el bando, además de imprimir el saldo que tiene cada cliente utilizando el método imprimir de la clase Cliente.

```
24      public void imprimir() {
25          System.out.println(nombre+" Tiene depositado la suma de "+importe);
26      }
27  }
```

En la línea 24 vamos al bloque principal del programa para realizar las operaciones y mostrar los respectivos saldos de clientes como el dinero total que tiene el banco.

```
29      public static void main(String[] args) {
30          Banco banco1=new Banco();
31          banco1.operar();
32          banco1.depositosTotales();
33      }
```

Cuando ejecutemos este será el resultado:

```
El total de dinero en banco:450
Ana Tiene depositado la suma de 0
Juan Tiene depositado la suma de 150
Pedro Tiene depositado la suma de 300
```

Capítulo 113.- Colaboración de clases – 2

Problema:

Plantear un programa que permita jugar a los dados. Las reglas de juego son: se tiran 3 dados si los tres salen con el mismo valor mostrar un mensaje que "gano", sino "perdió".

Lo primero que hacemos es identificar las clases:

Podemos identificar la clase Dado y la clase JuegoDeDados.

Luego los atributos y los métodos de cada clase:

Dado
- atributos
  - valor
- método
  - tirar
  - imprimir
  - retronarValor

JuegoDeDados
- atributo
  - 3 Dado (3 objetos de la clase Dado)
- métodos
  - constructor
  - jugar

Vamos a programar:

Como en el capitulo anterior vamos a trabajar con dos clases Dado y JuegoDeDados, vamos a analizar la primera clase Dado.

```
1  package paquete1;
2
3  public class Dado {
4      private int valor;
5
6      public void tirar() {
7          valor=(int)(Math.random()*6+1);
8      }
9
10     public void imprimir() {
11         System.out.println("Valor del dado:"+valor);
12     }
13
14     public int retornar() {
15         return valor;
16     }
17 }
```

En la línea 4 creamos una variable global llamada valor.

En la línea 6 creamos el método tirar que genera un valor aleatorio entre 1 y 6.

En la línea 10 creamos el método llamado imprimir que muestra el valor del dado.

En la línea 14 el método retornar el valor del dado.

Ahora vamos a ver la clase JuegoDeDados.

```
1  package paquete1;
2
3  public class JuegoDeDados {
4
5      Dado dado1, dado2, dado3;
6
7      public JuegoDeDados() {
8          dado1=new Dado();
9          dado2=new Dado();
10         dado3=new Dado();
11     }
12
13     public void jugar() {
14         dado1.tirar();
15         dado1.imprimir();
16         dado2.tirar();
17         dado2.imprimir();
18         dado3.tirar();
19         dado3.imprimir();
20         if (dado1.retornar()==dado2.retornar() && dado1.retornar()==dado3.retornar()) {
21             System.out.print("Gano");
22         }else {
23             System.out.print("Perdió");
24         }
25     }
26
27     public static void main(String[] args) {
28         JuegoDeDados juego1=new JuegoDeDados();
29         juego1.jugar();
30     }
31 }
```

En la línea 5 definimos tres objetos del tipo Dado.

En la línea 7 el constructor instanciamos los objetos de tipo Dado.

En la línea 13 con el método jugar llamamos los métodos tirar() e imprimir() de la clase Dado.

```
6      public void tirar() {
7          valor=(int)(Math.random()*6+1);
8      }
9
10     public void imprimir() {
11         System.out.println("Valor del dado:"+valor);
12     }
13
```

En la línea 20 comprobamos si el valor de los datos son iguales para los tres dados, llamando al método retornar() de la clase Dado.

```
14     public int retornar() {
15         return valor;
16     }
17 }
```

En la línea 27 bloque principal instanciamos juego1 de tipo JuegoDeDados y a continuación llamamos al método jugar().

Si ejecutamos este será el resultado:

```
Valor del dado:5
Valor del dado:5
Valor del dado:5
Gano
```

Ejecútalo varias veces hasta que te diga 'Gano'.

Capítulo 114.- Colaboración de clases – 3

Problema propuesto:

Plantear una clase Club y otra clase Socio.

La clase Socio debe tener los siguientes atributos privados: nombre y la antigüedad en el club (en años). En el constructor pedir la carga del nombre y su antigüedad.

La clase Club debe tener como atributo 3 objetos de la clase Socio. Definir una responsabilidad para imprimir el nombre del socio con mayor antigüedad en el club.

Vamos a programar:

Código de la clase Socio:

```
package paquete1;

import java.util.Scanner;

public class Socio {
	private Scanner teclado;
	private String nombre;
	private int antiguedad;

	public Socio(){
		teclado=new Scanner(System.in);
		System.out.print("Ingrese el nombre del socio:");
		nombre=teclado.next();
		System.out.print("Ingrese su antigüedad en años:");
		antiguedad=teclado.nextInt();
	}

	public String nombreSocio() {
		return nombre;
	}

	public int socioMasAnti() {
		return antiguedad;
	}
}
```

Código de la clase Club:

```
package paquete1;

public class Club {

	Socio socio1, socio2, socio3;

	public Club() {
		socio1=new Socio();
		socio2=new Socio();
		socio3=new Socio();
	}

	public void responsable() {
		if(socio1.socioMasAnti()>socio2.socioMasAnti() &&
socio1.socioMasAnti()>socio3.socioMasAnti()) {
```

```
                  System.out.print("El socio responsable será "+
socio1.nombreSocio()+" de "+socio1.socioMasAnti()+" años como socio.");
            }else {
                  if(socio2.socioMasAnti()>socio3.socioMasAnti()) {
                        System.out.print("El socio responsable será "+
socio2.nombreSocio()+" de "+socio2.socioMasAnti()+" años como socio.");
                  }else {
                        System.out.print("El socio responsable será "+
socio3.nombreSocio()+" de "+socio3.socioMasAnti()+" años como socio.");
                  }
            }
      }

      public static void main(String[] args) {
            Club club1 = new Club();
            club1.responsable();
      }
}
```

Si ejecutamos este será el resultado:

```
Ingrese el nombre del socio:Juan
Ingrese su antigüedad en años:35
Ingrese el nombre del socio:Ana
Ingrese su antigüedad en años:10
Ingrese el nombre del socio:Jorge
Ingrese su antigüedad en años:42
El socio responsable será Jorge de 42 años como socio.
```

Capítulo 115.- Herencia – 1

Herencia

La herencia significa que se pueden crear nuevas clases partiendo de las clases existentes, que tendrá todos los atributos y métodos de su 'superclase' o 'clase padre' y además se lo podrán añadir otros atributos y métodos propios.

Clase padre

Clase de la que desciende o deriva una clase. Las clases hijas (descendientes) heredan (incorporan) automáticamente los atributos y métodos de la clase padre.

Subclase

Clase descendiente de otra. Hereda automáticamente los atributos y métodos de su superclase. Es una especialización de otra clase. Admiten la definición de nuevos atributos y métodos para aumentar la especialización de la clase.

Ejemplo:

Clase: Persona
Subclase: Empleado

Otro ejemplo:

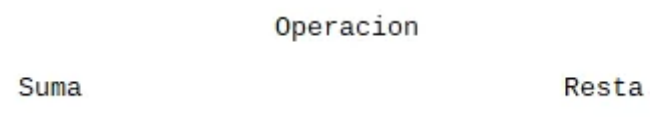
Clase: Vehículo
Subclase: Colectivo – Moto – Auto

Problema:

Ahora planteamos el primer problema utilizando herencia. Supongamos que necesitamos implementar dos clases que llamaremos Suma y Resta. Cada clase se tiene como atributo valor1, valor2 y resultado. Los métodos a definir son cargar1 (que inicializa el atributo valor1), cargar2 (que inicializa el atributo valor2), operar (que en el caso de la clase "Suma" suma los dos atributos y en el caso de la clase "Resta" hace la diferencia entre valor1 y valor2, y otro método mostrarResultado.

Si analizamos ambas clases encontraremos que muchos atributos y métodos son idénticos. En estos casos es bueno definir una clase padre que agrupe dichos atributos y responsabilidades comunes.

La relación de herencia que podemos disponer para este problema es:



Vamos a programar:

Para este proyecto vamos a realizar las clases Operación, Suma, Resta y Principal, este último será el ejecutable.

Vamos a ver el código de la clase Operacion:

```java
package paquete1;

import java.util.Scanner;

public class Operacion {
    protected Scanner teclado;
    protected int valor1;
    protected int valor2;
    protected int resultado;

    public Operacion() {
        teclado=new Scanner(System.in);
    }

    public void cargar1() {
        System.out.print("Ingrese el primer valor:");
        valor1=teclado.nextInt();
    }

    public void cargar2() {
        System.out.print("Ingrese el segundo valor:");
        valor2=teclado.nextInt();
    }

    public void mostrarResultado() {
        System.out.println(resultado);
    }
}
```

El modificador de acceso protected nos permite acceso a los componentes de dicho modificador desde la misma clase, clases del mismo paquete y clases que heredan de ella (incluso en diferentes paquetes).

En las líneas 6 hasta la 7 definimos las variables globales que necesitaremos.
En la línea 11 creamos un método para instanciar teclado como objeto de tipo Scanner.
En la línea 15 creamos el método cargar1 para asignar valor por teclado a la variable valor1.
En la línea 20 creamos el método cargar2 para asignar valor por teclado a la variable valor2.
En la línea 25 creamos el método mostrarResultado que nos mostrará el valor de la variable resultado.

Ahora vamos a ver el código de la clase Suma:

```java
package paquete1;

public class Suma extends Operacion{

    public void operar() {
        resultado=valor1+valor2;
    }
}
```

Con la palabra reservada extends le estamos diciendo que la clase Suma heredará todos los atributos y métodos de la clase Operación.

En la línea 5 creamos un método llamado operar() que realiza la suma de las variables valor1 y valor2 y lo guarda en la variable resultado, las tres variables se crearon en la clase Operación y como clase hija tenemos acceso a ellas.

Ahora vamos a ver el código de la clase Resta:

```
1  package paquete1;
2
3  public class Resta extends Operacion{
4
5      public void operar() {
6          resultado=valor1-valor2;
7      }
8  }
```

Es similar a la clase Suma pero en este caso es para la resta.

Ahora vamos a ver el código de la clase Principal, recuerda que es la que vamos a ejecutar.

```
1  package paquete1;
2
3  public class Principal {
4
5      public static void main(String[] args) {
6          Suma suma1=new Suma();
7          suma1.cargar1();
8          suma1.cargar2();
9          suma1.operar();
10         System.out.print("El resultado de la suma:");
11         suma1.mostrarResultado();
12
13         Resta resta1=new Resta();
14         resta1.cargar1();
15         resta1.cargar2();
16         resta1.operar();
17         System.out.print("El resultado de la resta es:");
18         resta1.mostrarResultado();
19     }
20 }
```

En esta clase instanciamos los objetos suma1 y resta1 con sus respectivas clases.

Los métodos cargar1(), cargar2() y mostrar resultado las tenemos en la clase Padre, en cambio el método operar() las tenemos en las clases Suma y Resta y cada una realiza una operación distinta.

Vamos a ejecutar el programa y este será el resultado:

```
Ingrese el primer valor:10
Ingrese el segundo valor:20
El resultado de la suma:30
Ingrese el primer valor:40
Ingrese el segundo valor:1
El resultado de la resta es:39
```

Capítulo 116.- Herencia – 2

Problema propuesto:

Confeccionar una clase Persona que tenga como atributo el nombre y la edad. Definir como responsabilidades un método que cargue los datos personales y otro que imprima.

Plantear una segunda clase Empleado que herede de la clase Persona. Añadir un atributo sueldo y los métodos de cargar el sueldo e imprimir su sueldo.

Definir un objeto de la clase Persona y llamar a sus métodos. También crear un objeto de la clase empleados y llamar a sus métodos.

Vamos a programar la clase Persona:

```
1  package paquete1;
2
3  import java.util.Scanner;
4
5  public class Persona {
6      protected Scanner teclado;
7      protected String nombre;
8      protected int edad;
9
10     public Persona() {
11         teclado=new Scanner(System.in);
12     }
13
14     public void cargarDatosPersonales() {
15         System.out.print("Ingrese el nombre:");
16         nombre=teclado.next();
17         System.out.print("Ingrese la edad:");
18         edad=teclado.nextInt();
19     }
20
21     public void imprimirDatosPersonales() {
22         System.out.println("Nombre:"+nombre);
23         System.out.println("Edad:"+edad);
24     }
25 }
```

Vamos a programar la clase Empleado:

```
1  package paquete1;
2
3  public class Empleado extends Persona{
4      protected int sueldo;
5
6      public void cargarSueldo() {
7          System.out.print("Ingrese su sueldo:");
8          sueldo=teclado.nextInt();
9      }
10     public void imprimirSueldo() {
11         System.out.println("El sueldo es:"+sueldo);
12     }
13 }
```

Vamos a programar la clase Principal:

```java
package paquete1;

public class Principal {

    public static void main(String[] args) {
        Persona persona1=new Persona();
        persona1.cargarDatosPersonales();
        persona1.imprimirDatosPersonales();

        Empleado empleado1=new Empleado();
        empleado1.cargarDatosPersonales();
        empleado1.cargarSueldo();
        empleado1.imprimirDatosPersonales();
        empleado1.imprimirSueldo();
    }
}
```

Si ejecutamos este serla el resultado:

```
Ingrese el nombre:Diego
Ingrese la edad:44
Nombre:Diego
Edad:44
Ingrese el nombre:Juan
Ingrese la edad:50
Ingrese su sueldo:3000
Nombre:Juan
Edad:50
El sueldo es:3000
```

Capítulo 117.- Interfaces visuales (componentes Swing)

En Java existen varias librerías de clase para implementar interfaces visuales. Utilizaremos los componentes Swing.

Problema:

Confeccionar el programa "Hola Mundo" utilizando una interface gráfica de usuario con los componentes Swing.

Vamos a programar:

*Formulario.java

```
1  import javax.swing.JFrame;
2  import javax.swing.JLabel;
3  public class Formulario extends JFrame{
4      private JLabel label1;
5
6      public Formulario() {
7          setLayout(null);
8          label1=new JLabel("Hola Mundo.");
9          label1.setBounds(10,20, 200, 30);
10         add(label1);
11     }
12
13     public static void main(String[] args) {
14         Formulario formulario1=new Formulario();
15         formulario1.setBounds(10,20,400,300);
16         formulario1.setVisible(true);
17         formulario1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18     }
19 }
```

Lo primero que hacemos es importar los paquetes JFrame y JLabel, si tenemos la necesidad de importar más paquetes lo podemos simplificar con la siguiente línea:

import javax.swing.*;

En la línea 9 con setBounds ubicamos la etiqueta en la posición 10 pixeles en la columna y 20 pixeles en la fila, 200 pixeles de ancho y 30 pixeles de alto.

En la línea 14 creamos un objeto de tipo Formulario.

En la línea 15 le decimos donde queremos ubicar la ventana y sus dimensiones en nuestro escritorio de Windows.

En la línea 16 hacemos visible la ventana.

En la línea 17 nos permitirá cerrar la ventana presionando la x.

Para agregar elementos en el formulario lo haremos desde el constructor.

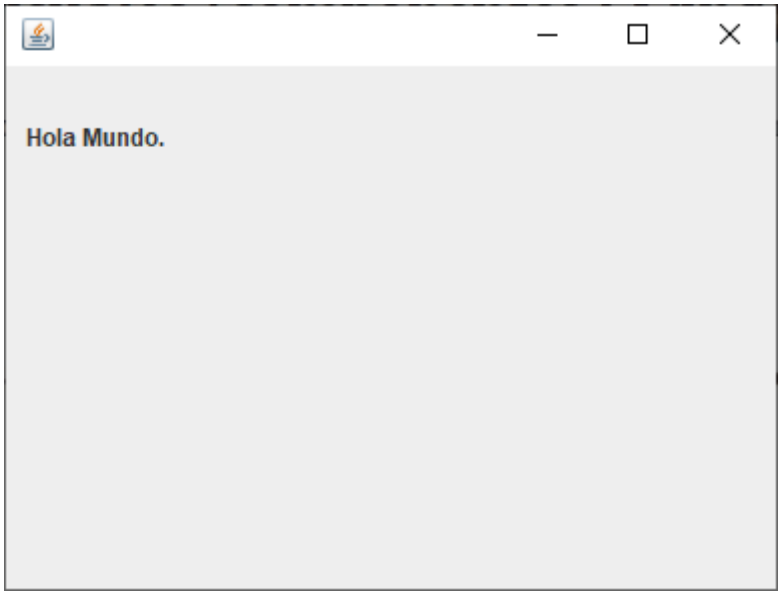
En la línea 7 nos permite ubicar los objetos correctamente si no lo ponemos ocupará toda la ventana.

En la línea 8 creamos un objeto de tipo JLabel con el texto "Hola Mundo."

En la línea 9 definimos sus coordenadas, así como su ancho y alto en pixeles.

En la línea 10 agregamos la etiqueta a nuestro formulario.

Si ejecutamos este será el resultado.



Capitulo 118.- Swing – Jframe

Problema propuesto:

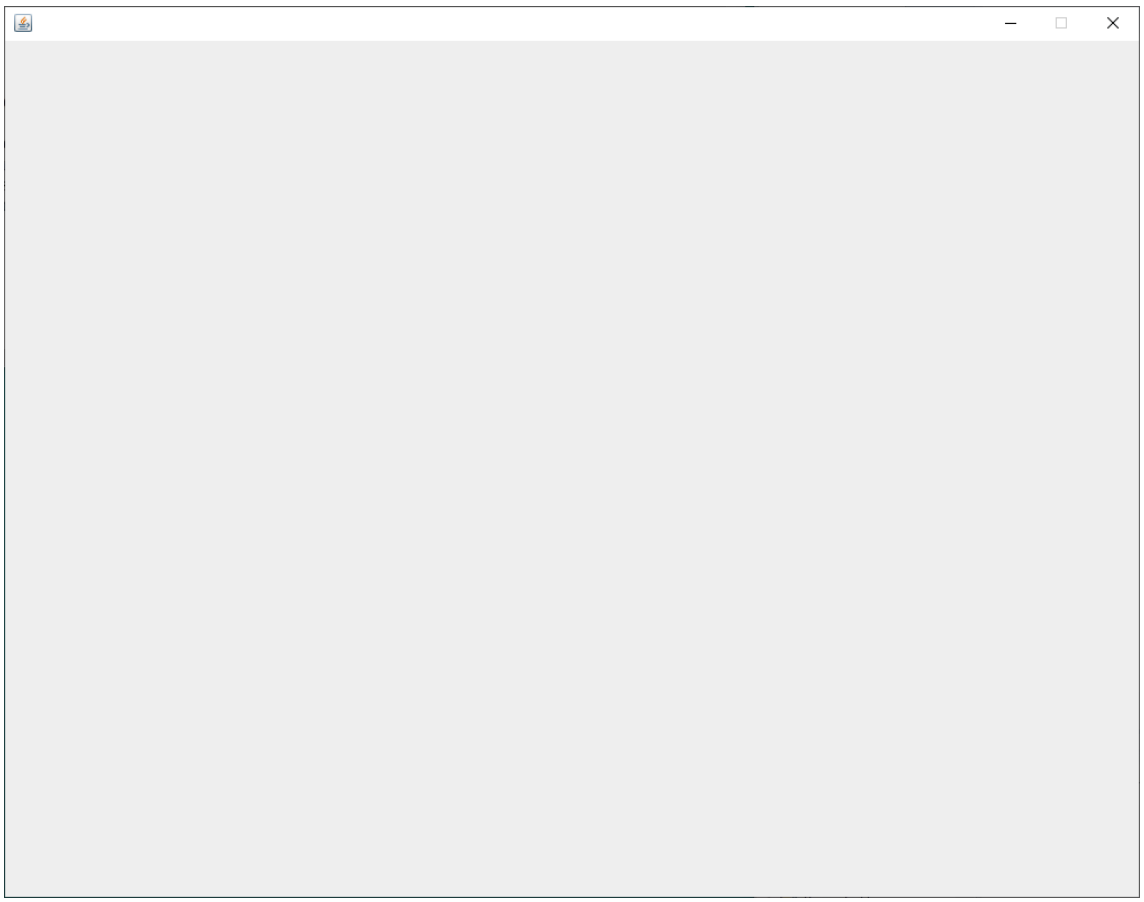
Crear una ventana de 1024 píxeles por 800 píxeles. Luego no permitir que el operador modifique el tamaño de la ventana. Sabiendo que hacemos visible al JFrame llamando al método setVisible pasando el valor true, existe otro método llamado setResizable que también requiere como parámetro el valor true o false.

Vamos a programar.

*Formulario.java

```
1  import javax.swing.JFrame;
2  public class Formulario extends JFrame{
3
4      public static void main(String[] args) {
5          Formulario formulario1=new Formulario();
6          formulario1.setBounds(10,20,1024,800);
7          formulario1.setVisible(true);
8          formulario1.setResizable(false);
9          formulario1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
10     }
11 }
```

Si ejecutamos este será el resultado:



Capítulo 119.- Swing – Jlabel – 1

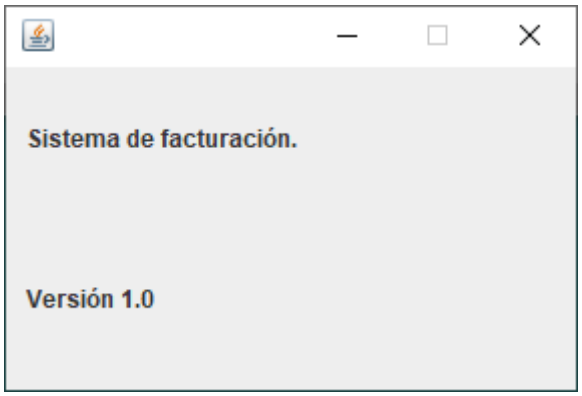
Confeccionar una ventana que muestre el nombre de un programa en la parte superior y su número de versión en la parte inferior (Utilizar dos JLabel). No permitir modificar el tamaño de la ventana en tiempo de ejecución.

Vamos a programar:

Formulario.java

```
1  import javax.swing.JFrame;
2  import javax.swing.JLabel;
3
4  public class Formulario extends JFrame{
5      private JLabel label1, label2;
6
7      public Formulario() {
8          setLayout(null);
9          label1=new JLabel("Sistema de facturación.");
10         label1.setBounds(10,20,300,30);
11         add(label1);
12         label2=new JLabel("Versión 1.0");
13         label2.setBounds(10,100, 100, 30);
14         add(label2);
15     }
16
17     public static void main(String[] args) {
18         Formulario formulario1=new Formulario();
19         formulario1.setBounds(0,0,300,200);
20         formulario1.setResizable(false);
21         formulario1.setVisible(true);
22         formulario1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23     }
24 }
```

Si ejecutamos este será el resultado:



Capítulo 120.- Swing – Jlabel – 2

Problema propuesto:

Crear tres objetos de la clase JLabel, ubicarlos uno debajo del otro y mostrar nombres de colores.

Vamos a programar:

Formulario.java

```
1  import javax.swing.JFrame;
2  import javax.swing.JLabel;
3
4  public class Formulario extends JFrame{
5      private JLabel label1, label2, label3;
6
7      public Formulario() {
8          setLayout(null);
9          label1=new JLabel("Rojo");
10         label1.setBounds(125,25, 100,30);
11         add(label1);
12         label2=new JLabel("Verde");
13         label2.setBounds(125,50, 100,30);
14         add(label2);
15         label3=new JLabel("Azul");
16         label3.setBounds(125,75, 100,30);
17         add(label3);
18     }
19
20     public static void main(String[] args) {
21         Formulario formulario1=new Formulario();
22         formulario1.setBounds(0,0, 300,200);
23         formulario1.setVisible(true);
24         formulario1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25     }
26 }
```

Si ejecutamos este será el resultado:

Capítulo 121.- Swing – JButton – 1

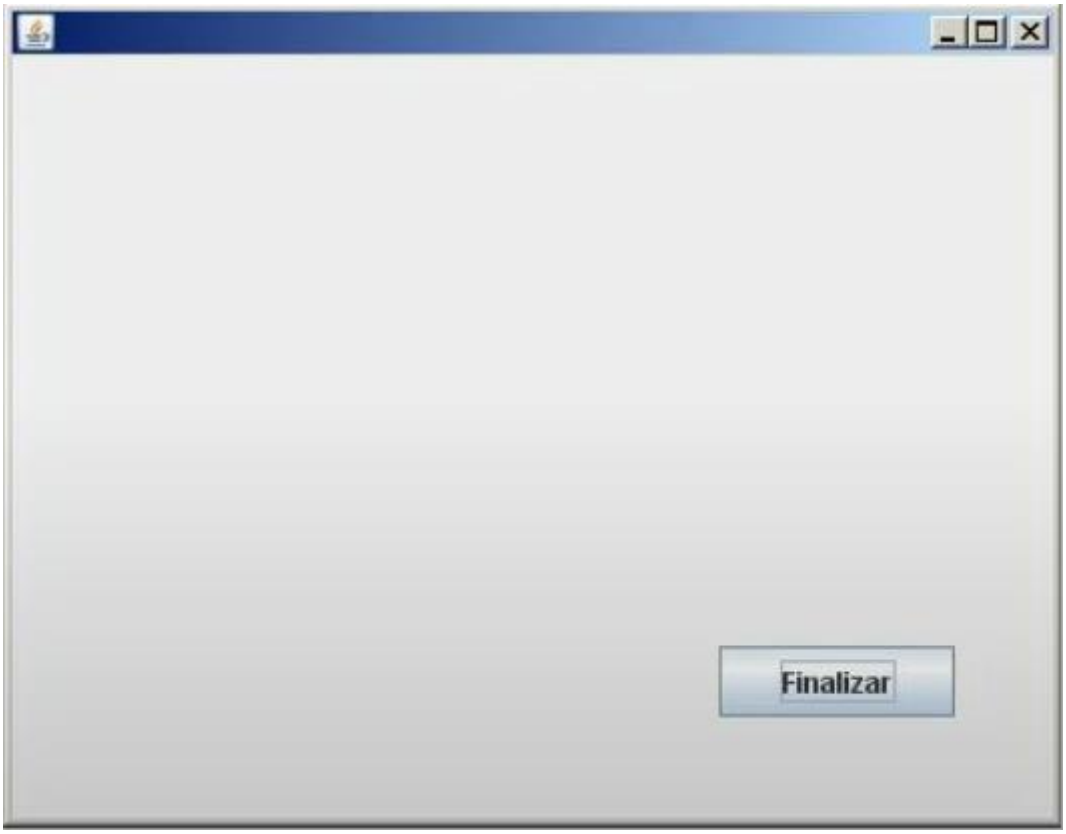
El tercer control visual de uso muy común es el que provee la clase JButton. Este control visual muestra un botón.

Ahora veremos la captura de eventos con los controles visuales. Uno de los eventos más comunes es cuando hacemos clic sobre un botón.

Java implementa el concepto de interfaces para poder llamar a métodos de una clase existente a una clase desarrollada por nosotros.

Problema:

Confeccionar una ventana que muestre un botón. Cuando se presione finalizar la ejecución del programa Java.



Vamos a programar:

*Formulario.java

```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6
7  public class Formulario extends JFrame implements ActionListener{
8      JButton boton1;
9
```

```
10      public Formulario() {
11          setLayout(null);
12          boton1=new JButton("Finalizar");
13          boton1.setBounds(300,250,100,30);
14          add(boton1);
15          boton1.addActionListener(this);
16      }
17
18      public static void main(String[] args) {
19          Formulario formulario1=new Formulario();
20          formulario1.setBounds(0,0,450,350);
21          formulario1.setVisible(true);
22          formulario1.setDefaultCloseOperation(EXIT_ON_CLOSE);
23      }
24
25      @Override
26      public void actionPerformed(ActionEvent e) {
27          if (e.getSource()==boton1) {
28              System.exit(0);
29          }
30      }
31  }
```

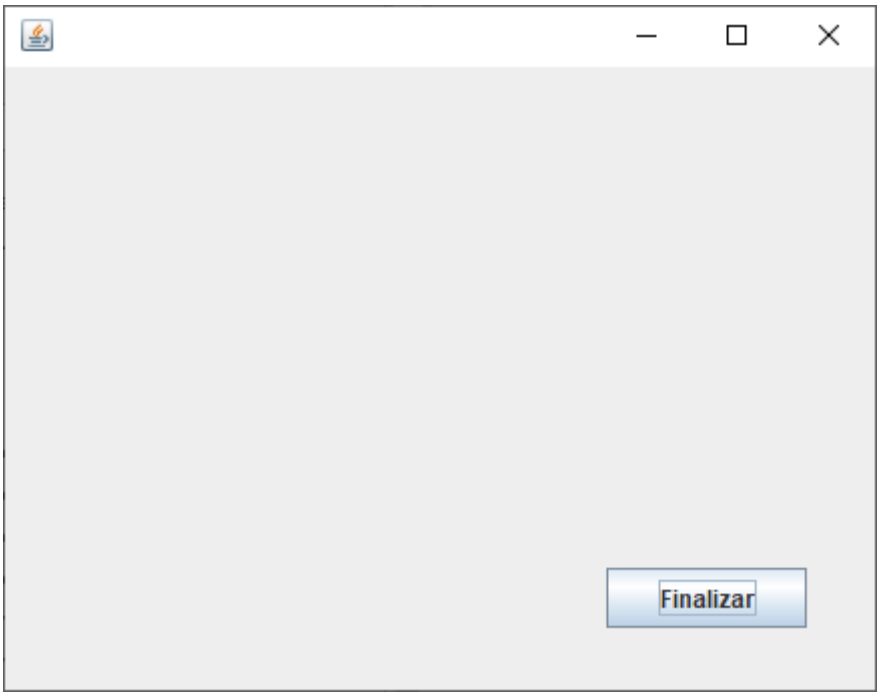
En la línea 1 y 2 importamos la acción de los eventos y que esté pendientes al mismo.

En la línea 15 le decimos que el botón tiene que estar pendiente a una acción, esta acción la tenemos a partir de la línea 25.

En la línea 7 tenemos que implementar ActionListener, además de añadir la extensión JFrame.

Cuando hagamos clic sobre el botón se cerrará la aplicación.

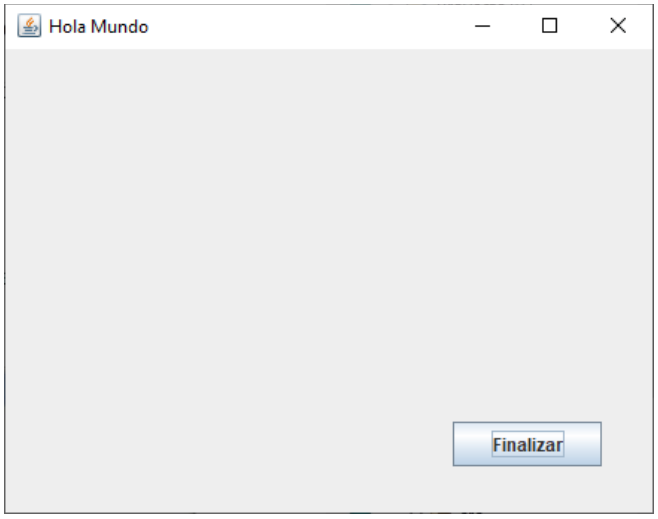
Vamos a ejecutar, este será el resultado:



Te propongo que modifiques el método actionPerformed:

```
25      @Override
26      public void actionPerformed(ActionEvent e) {
27          if (e.getSource()==boton1) {
28              setTitle("Hola Mundo");
29          }
30      }
```

Ahora queremos que cuando presionemos el botón se muestre 'Hola Mundo." en el titulo de la ventana:




```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6
7  public class Formulario extends JFrame implements ActionListener{
8      JButton boton1;
9
10     public Formulario() {
11         setLayout(null);
12         boton1=new JButton("Finalizar");
13         boton1.setBounds(300,250,100,30);
14         add(boton1);
15         boton1.addActionListener(this);
16     }
17
18     public static void main(String[] args) {
19         Formulario formulario1=new Formulario();
20         formulario1.setBounds(0,0,450,350);
21         formulario1.setVisible(true);
22         formulario1.setDefaultCloseOperation(EXIT_ON_CLOSE);
23     }
24
25     @Override
26     public void actionPerformed(ActionEvent e) {
27         if (e.getSource()==boton1) {
28             System.exit(0);
29         }
30     }
31 }
```

Al implementar ActionListener estamos obligados a añadir el método que está seleccionado.

Capítulo 122.- Swing – JButton – 2

Problema:

Confeccionar una ventana que contenga tres objetos de la clase JButton con las siguientes etiquetas "1", "2" y "3".

Al presionarse cambiar el título del JFrame indicando cuál botón se presionó.

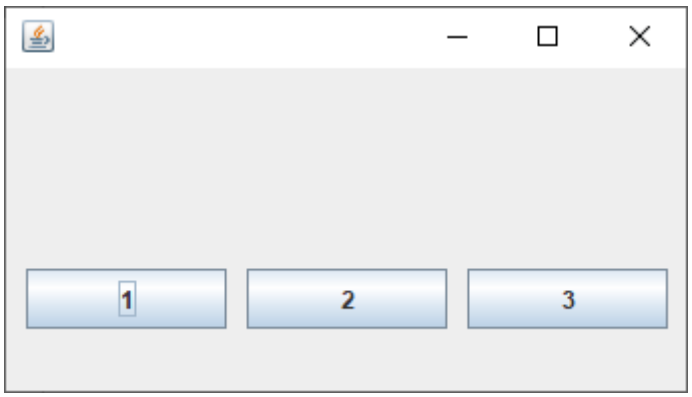
Vamos a programar:

Ventana.java

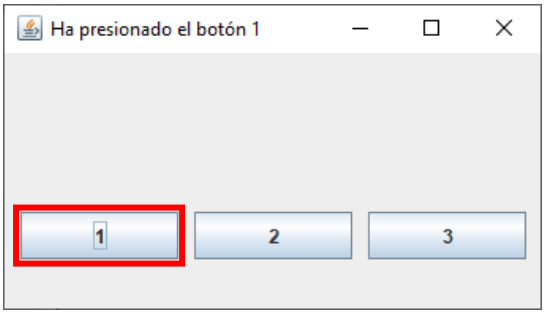
```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6
7  public class Ventana extends JFrame implements ActionListener{
8      JButton boton1, boton2, boton3;
9
10     public Ventana() {
11         setLayout(null);
12         boton1=new JButton("1");
13         boton1.setBounds(10,100,100,30);
14         add(boton1);
15         boton1.addActionListener(this);
16
17         boton2=new JButton("2");
18         boton2.setBounds(120,100,100,30);
19         add(boton2);
20         boton2.addActionListener(this);
21
22         boton3=new JButton("3");
23         boton3.setBounds(230,100,100,30);
24         add(boton3);
25         boton3.addActionListener(this);
26     }
27
28     public static void main(String[] args) {
29         Ventana ventana1=new Ventana();
30         ventana1.setBounds(0,0,355,200);
31         ventana1.setVisible(true);
32         ventana1.setDefaultCloseOperation(EXIT_ON_CLOSE);
33
34     }
```

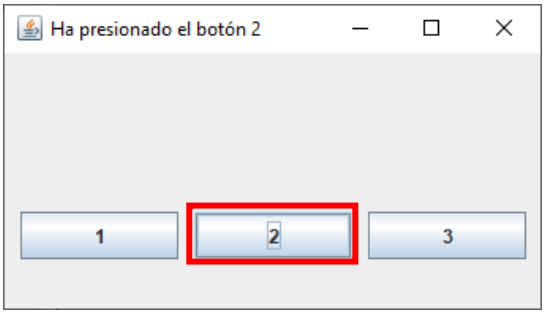
```
35
36      @Override
37      public void actionPerformed(ActionEvent e) {
38          if(e.getSource()==boton1) {
39              setTitle("Ha presionado el botón 1");
40          }
41          if(e.getSource()==boton2) {
42              setTitle("Ha presionado el botón 2");
43          }
44          if(e.getSource()==boton3) {
45              setTitle("Ha presionado el botón 3");
46          }
47      }
48  }
```

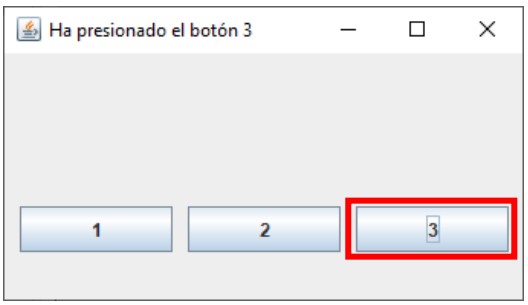
Si ejecutamos este será el resultado:



Vamos a presionar los botones:







e.

Capítulo 123.- Swing – Jbutton – 3

Problema propuesto:

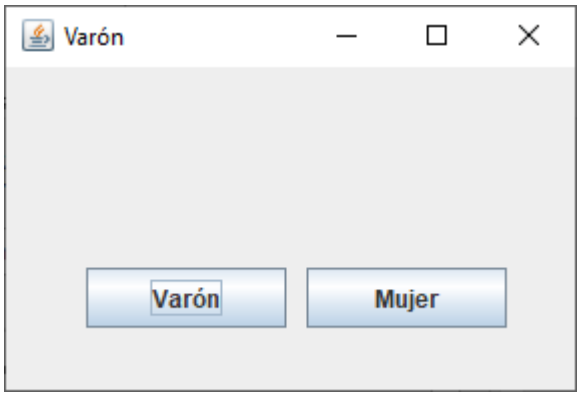
Disponer dos objetos de la clase JButton con las etiquetas: "varón" y "mujer", al presionarse mostrar en la barra de títulos del JFrame la etiqueta del botón presionado.

Vamos a programar.

*Formulario.java

```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6
7  public class Formulario extends JFrame implements ActionListener{
8      JButton boton1, boton2;
9
10     public Formulario() {
11         setLayout(null);
12         boton1=new JButton("Varón");
13         boton1.setBounds(40,100, 100,30);
14         add(boton1);
15         boton1.addActionListener(this);
16
17         boton2=new JButton("Mujer");
18         boton2.setBounds(150,100, 100,30);
19         add(boton2);
20         boton2.addActionListener(this);
21     }
22
23     public static void main(String[] args) {
24         Formulario formulario1=new Formulario();
25         formulario1.setBounds(100,100, 300, 200);
26         formulario1.setVisible(true);
27         formulario1.setDefaultCloseOperation(EXIT_ON_CLOSE);
28     }
29
30     @Override
31     public void actionPerformed(ActionEvent e) {
32         if(e.getSource()==boton1) {
33             setTitle("Varón");
34         }
35         if(e.getSource()==boton2) {
36             setTitle("Mujer");
37         }
38     }
39 }
```

Vamos a ejecutar, este será el resultado:



Capítulo 124.- Swing – JtextField – 1

Así como podríamos decir que el control JLabel remplaza a la salida estándar System.out.print, el control JTextField cumple la función de la clase Scanner para la entrada de datos.

El control JTextField permite al operador del programa ingresar una cadena de caracteres por teclado.

Problema:

Confeccionar un programa que permita ingresar el nombre del usuario y cuando presione un botón mostrar el valor ingresado en la barra de títulos del JFrame.

Vamos a programar:

*Formulario.java

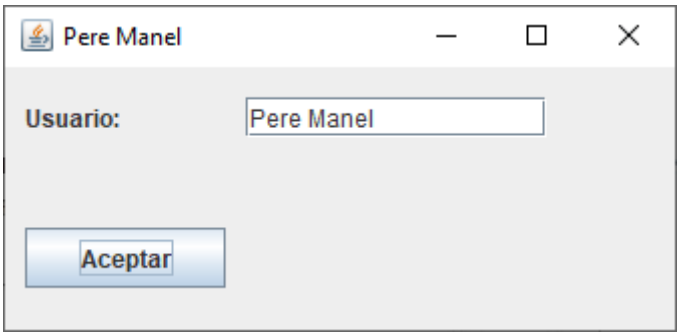
```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6  import javax.swing.JLabel;
7  import javax.swing.JTextField;
8
9  public class Formulario extends JFrame implements ActionListener{
10     private JTextField textfield1;
11     private JLabel label1;
12     private JButton boton1;
13
14     public Formulario() {
15         setLayout(null);
16         label1=new JLabel("Usuario:");
17         label1.setBounds(10,10,100,30);
18         add(label1);
19         textfield1=new JTextField();
20         textfield1.setBounds(120,15,150,20);
21         add(textfield1);
22         boton1=new JButton("Aceptar");
23         boton1.setBounds(10,80, 100, 30);
24         add(boton1);
25         boton1.addActionListener(this);
26     }
27
28 public static void main(String[]ar) {
29     Formulario f=new Formulario();
30     f.setBounds(0,0,350,170);
31     f.setVisible(true);
32     f.setDefaultCloseOperation(EXIT_ON_CLOSE);
33 }
```

```
34
35
36  @Override
37  public void actionPerformed(ActionEvent e) {
38      if (e.getSource()==boton1) {
39          String cadena=textfield1.getText();
40          setTitle(cadena);
41      }
42  }
```

También podemos realizar el método actionPerformed sin utilizar ninguna variable.

```
36  @Override
37  public void actionPerformed(ActionEvent e) {
38      if (e.getSource()==boton1) {
39          setTitle(textfield1.getText());
40      }
41  }
```

Cuando ejecutemos este será el resultado:



Capítulo 125.- Swing – JtextField – 2

Problema:

Confeccionar un programa que permita ingresar dos números en controles de tipo JTextField, luego sumar los dos valores ingresados y mostrar la suma en la barra de título del control JFrame.

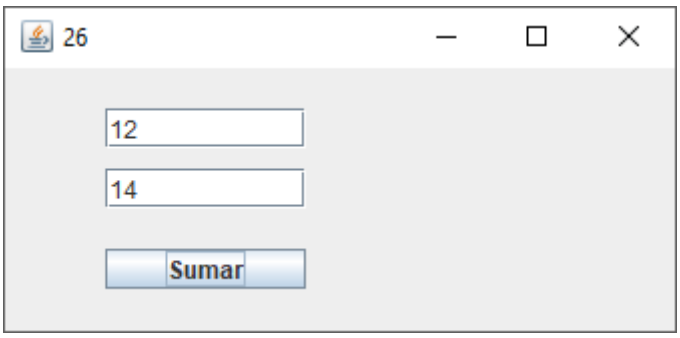
Vamos a programar:

*Formulario.java

```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3
4  import javax.swing.JFrame;
5  import javax.swing.JTextField;
6  import javax.swing.JButton;
7
8  public class Formulario extends JFrame implements ActionListener{
9      private JTextField textfield1, textfield2;
10     private JButton boton1;
11
12     public Formulario() {
13         setLayout(null);
14         textfield1=new JTextField();
15         textfield1.setBounds(50,20,100,20);
16         add(textfield1);
17         textfield2=new JTextField();
18         textfield2.setBounds(50,50,100,20);
19         add(textfield2);
20         boton1=new JButton("Sumar");
21         boton1.setBounds(50,90, 100,20);
22         add(boton1);
23         boton1.addActionListener(this);
24     }
25
26 public static void main(String[] args) {
27     Formulario f=new Formulario();
28     f.setBounds(0,0,350,170);
29     f.setVisible(true);
30     f.setDefaultCloseOperation(EXIT_ON_CLOSE);
31 }
32
```

```
33  @Override
34  public void actionPerformed(ActionEvent e) {
35      if(e.getSource()==boton1) {
36          int num1=Integer.parseInt((textfield1.getText()));
37          int num2=Integer.parseInt((textfield2.getText()));
38          int suma=num1+num2;
39          String cadenaSuma=String.valueOf(suma);
40          setTitle(cadenaSuma);
41      }
42  }
43  }
```

Ejecutamos, este será el resultado:



Capítulo 126.- Swing – JtextField – 3

Problema propuesto:

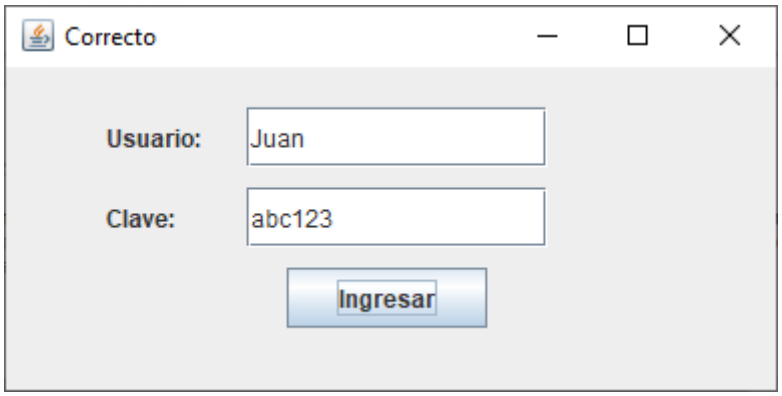
Ingresar el nombre de usuario y la clave en controles de tipo JtextField. Si se ingresa las cadena (usuario: "Juan", clave: "abc123") luego mostrar en el título del JFrame el mensaje "Correcto" en caso contrario mostrar el mensaje "Incorrecto".

Vamos a programar:

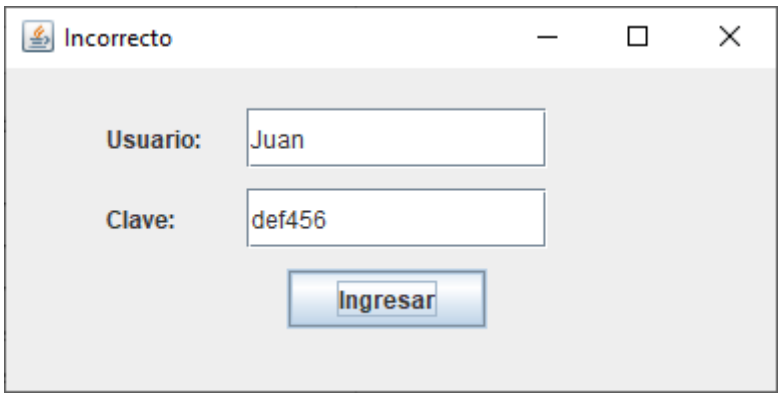
*Usuario.java

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;

public class Usuario extends JFrame implements ActionListener{
    private JLabel label1, label2;
    private JTextField textfield1, textfield2;
    private JButton boton1;

    public Usuario() {
        setLayout(null);
        label1=new JLabel("Usuario:");
        label1.setBounds(50,20, 200, 30);
        add(label1);
        label2=new JLabel("Clave:");
        label2.setBounds(50,60, 200, 30);
        add(label2);
        textfield1=new JTextField();
        textfield1.setBounds(120, 20, 150, 30);
        add(textfield1);
        textfield2=new JTextField();
        textfield2.setBounds(120, 60, 150, 30);
        add(textfield2);
        boton1=new JButton("Ingresar");
        boton1.setBounds(140, 100, 100, 30);
        add(boton1);
        boton1.addActionListener(this);
    }

    public static void main(String[] args) {
        Usuario usuario1=new Usuario();
        usuario1.setBounds(100,100, 400,200);
        usuario1.setVisible(true);
        usuario1.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==boton1) {
            if(textfield1.getText().equals("Juan") && textfield2.getText().equals("abc123")) {
                setTitle("Correcto");
            }else {
                setTitle("Incorrecto");
            }
        }
    }
}
```

Vamos a ejecutar introduciendo nombre de usuario y contraseña correcta.



Ahora vamos a introducir una contraseña incorrecta.

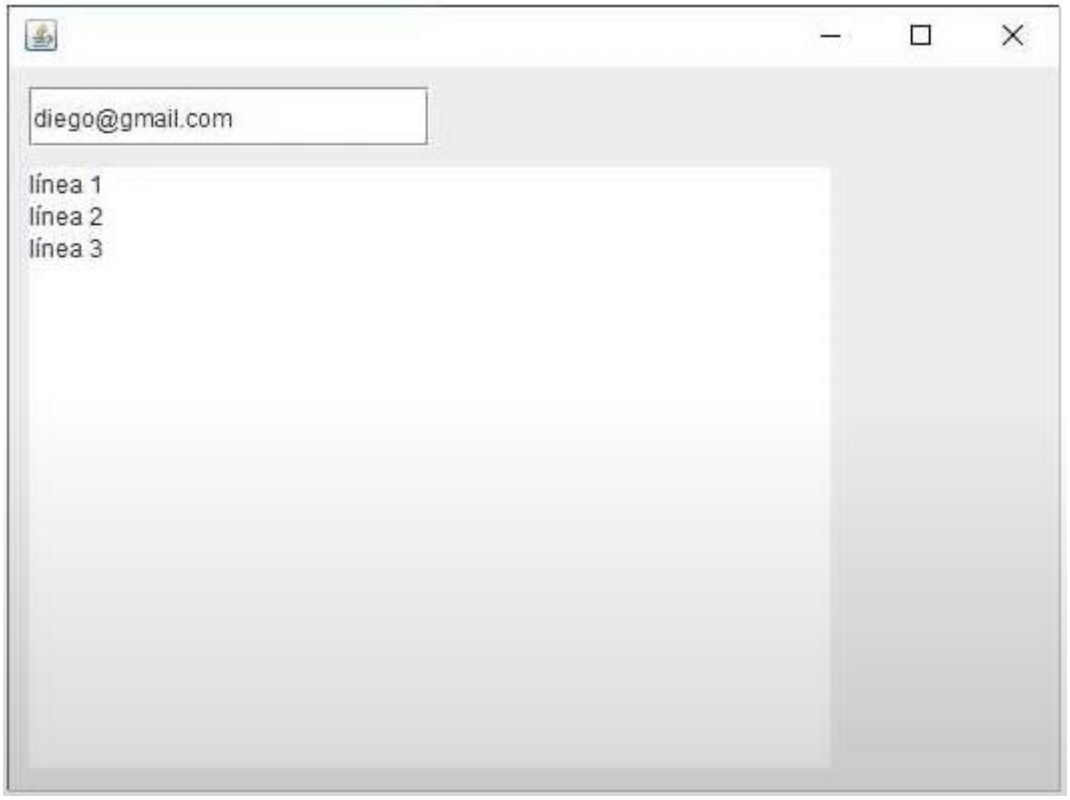


Capítulo 127.- Swing – JTextArea – 1

El control de tipo JTextArea permite ingresar múltiples líneas, a diferencia del control de tipo JTextField.

Problema:

Confeccionar un programa que permita ingresar u mail en un control de tipo JtextField y el cuerpo del mail en un control de tipo JTextArea.



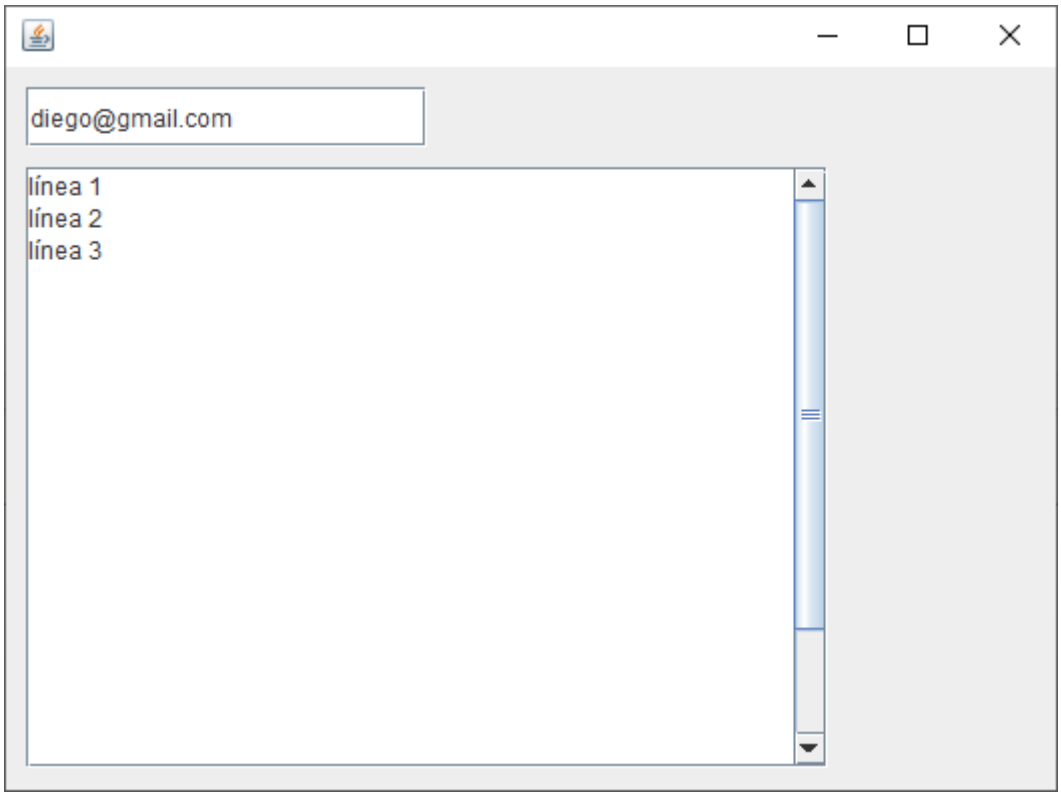
Vamos a programar:

*Formulario.java

```
1  import javax.swing.JFrame;
2  import javax.swing.JScrollPane;
3  import javax.swing.JTextArea;
4  import javax.swing.JTextField;
5  public class Formulario extends JFrame {
6      private JTextField tf1;
7      private JTextArea ta;
8      private JScrollPane sp1;
9
10     public Formulario() {
11         setLayout(null);
12         tf1=new JTextField();
13         tf1.setBounds(10,10,200,30);
14         add(tf1);
15         ta=new JTextArea();
16         sp1=new JScrollPane(ta);
17         sp1.setBounds(10,50,400,300);
18         add(sp1);
19     }
```

```
20
21      public static void main(String[] args) {
22          Formulario f=new Formulario();
23          f.setBounds(0,0,540,400);
24          f.setVisible(true);
25          f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
26      }
27  }
```

Ejecutamos y este será el resultado:



Capítulo 128.- Swing – JtextArea – 2

Problema:

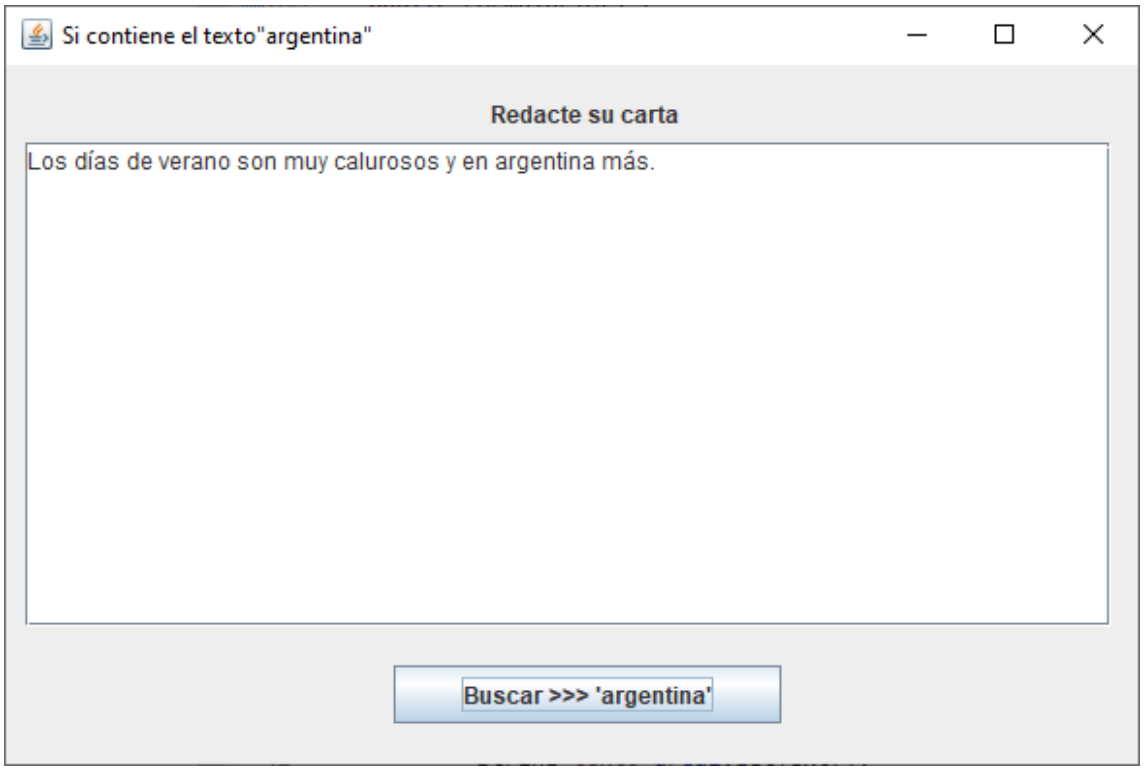
Confeccionar un programa que permita ingresar en un control de tipo JTextArea una carta. Luego al presionar un botón mostrar un mensaje si la carta contiene el string "argentina".

Vamos a programar:

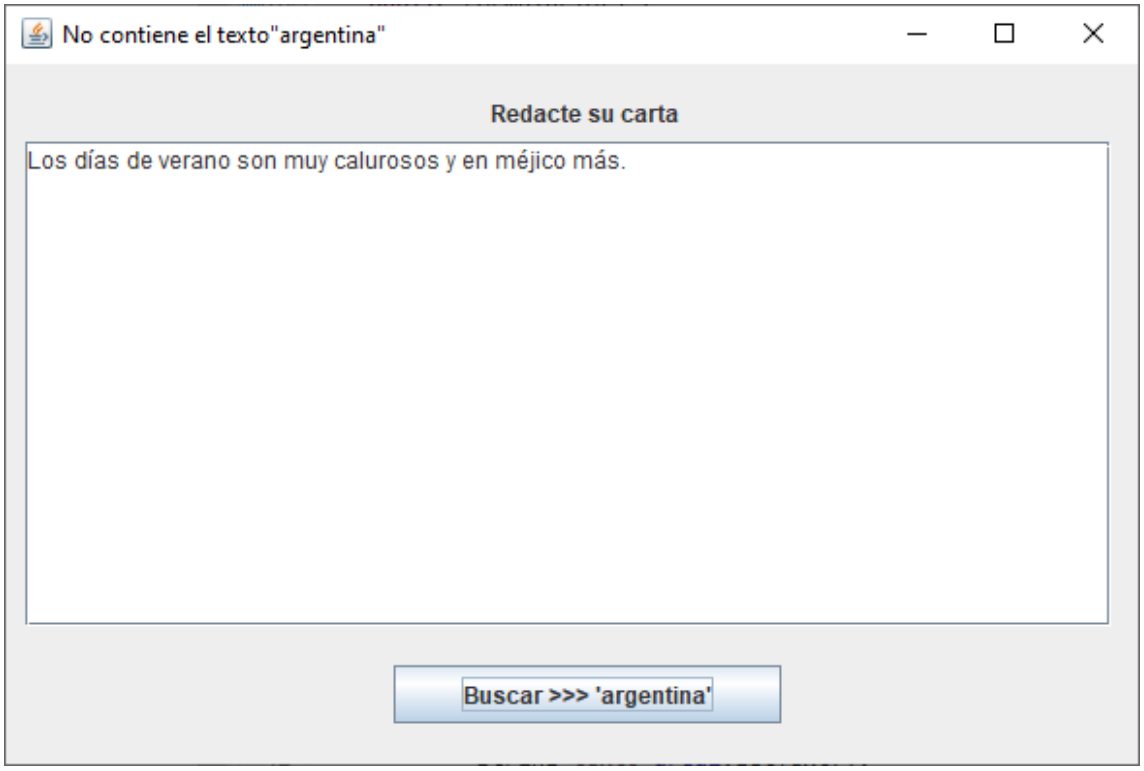
Formulario.java

```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3
4  import javax.swing.JFrame;
5  import javax.swing.JLabel;
6  import javax.swing.JTextArea;
7  import javax.swing.JScrollPane;
8  import javax.swing.JButton;
9
10 public class Formulario extends JFrame implements ActionListener{
11     private JLabel label1;
12     private JTextArea area1;
13     private JScrollPane sp1;
14     private JButton boton1;
15
16     public Formulario() {
17         setLayout(null);
18         label1=new JLabel("Redacte su carta");
19         label1.setBounds(250, 10, 300,30);
20         add(label1);
21         area1=new JTextArea();
22         sp1=new JScrollPane(area1);
23         sp1.setBounds(10,40, 560, 250);
24         add(sp1);
25         boton1=new JButton("Buscar >>> 'argentina'");
26         boton1.setBounds(200, 310, 200,30);
27         add(boton1);
28         boton1.addActionListener(this);
29     }
30
31     public static void main(String[] args) {
32         Formulario f=new Formulario();
33         f.setBounds(0,0, 600,400);
34         f.setVisible(true);
35         f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
36
37     }
38
39     @Override
40     public void actionPerformed(ActionEvent e) {
41         if(e.getSource()==boton1) {
42             String texto=area1.getText();
43             if (texto.indexOf("argentina")!=-1) {
44                 setTitle("Si contiene el texto\"argentina\"");
45             }else {
46                 setTitle("No contiene el texto\"argentina\"");
47             }
48         }
49     }
50 }
```

Si ejecutamos este será el resultado si encuentra 'argentina'.



Si no tiene la palabra 'argentina'.



Capítulo 129.- Swing – JTextArea – 3

Problema propuesto:

Disponer dos controles de tipo JTextArea, luego al presionar un botón verificar si tienen exactamente el mismo contenido.

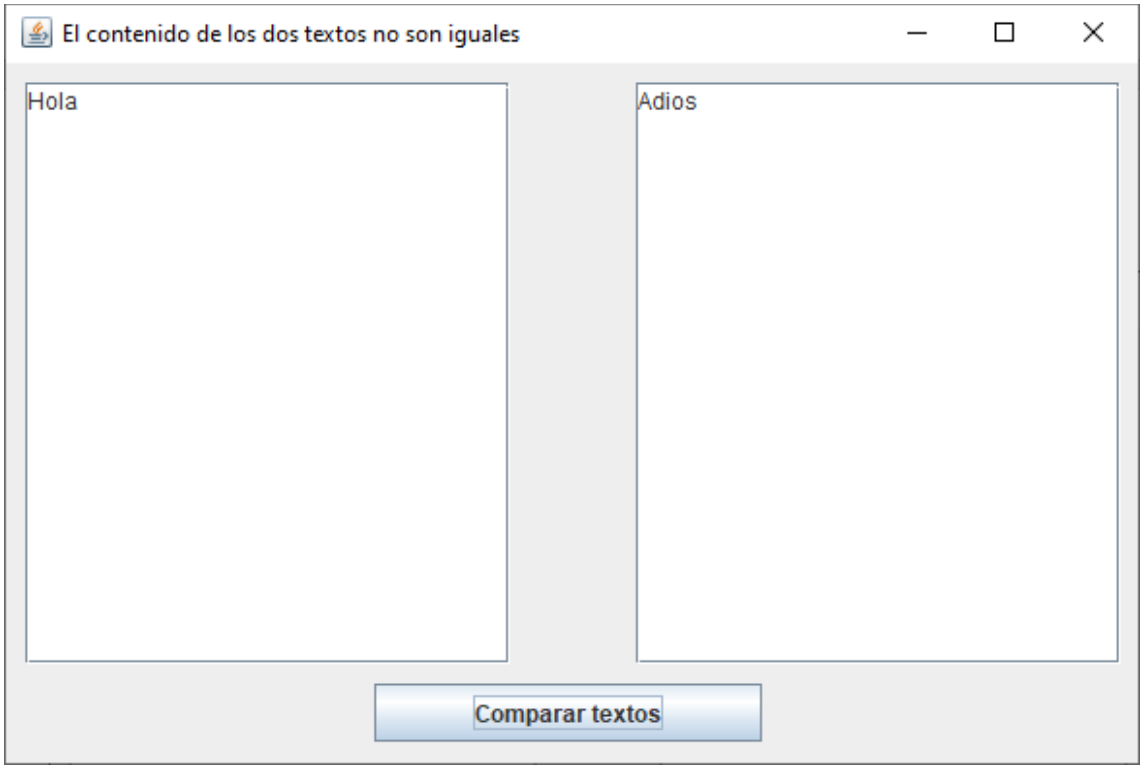
Vamos a programar:

*Formulario.java

```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3
4  import javax.swing.JFrame;
5  import javax.swing.JTextArea;
6  import javax.swing.JScrollPane;
7  import javax.swing.JButton;
8  public class Formulario extends JFrame implements ActionListener{
9      private JTextArea area1, area2;
10     private JScrollPane barra1, barra2;
11     private JButton boton1;
12
13     public Formulario() {
14         setLayout(null);
15         area1=new JTextArea();
16         barra1=new JScrollPane(area1);
17         barra1.setBounds(10,10, 250, 300);
18         add(barra1);
19         area2=new JTextArea();
20         barra2=new JScrollPane(area2);
21         barra2.setBounds(325,10, 250, 300);
22         add(barra2);
23         boton1=new JButton("Comparar textos");
24         boton1.setBounds(190, 320, 200,30);
25         add(boton1);
26         boton1.addActionListener(this);
27     }
28
29     public static void main(String[] args) {
30         Formulario f1=new Formulario();
31         f1.setBounds(100,100, 600,400);
32         f1.setVisible(true);
33         f1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
34
35     }
36
37     @Override
38     public void actionPerformed(ActionEvent e) {
39         if(e.getSource()==boton1) {
40             if(area1.getText().equals(area2.getText())) {
41                 setTitle("El contenido de los dos textos son iguales");
42             }else {
43                 setTitle("El contenido de los dos textos no son iguales");
44             }
45         }
46     }
47 }
```

Vamos a ejecutar y este será el resultado con dos textos iguales:

Vamos a ejecutar de nuevo pero con dos textos distintos:



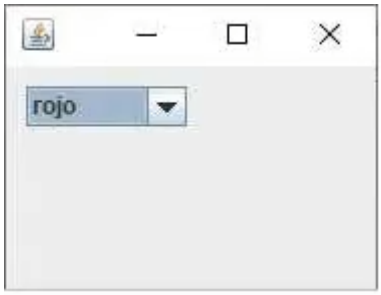
Capítulo 130.- Swing – JComboBox – 1

El control JComboBox permite seleccionar un String u otro tipo de dato.

Un evento muy útil con este control es cuando el operador selecciona un Item de la lista. Para capturar la selección de un item debemos implementar la interface ItemListener que contiene un método llamado itemStateChanged,

Problema:

Cargar en un JComboBox los nombres de varios colores. Al seleccionar alguno mostrar en la barra de título del JFrame el String seleccionado.



Vamos a programar:

Formulario.java

```
1  import java.awt.event.ItemEvent;
2  import java.awt.event.ItemListener;
3
4  import javax.swing.JComboBox;
5  import javax.swing.JFrame;
6
7  public class Formulario extends JFrame implements ItemListener{
8      private JComboBox<String> combo1;
9
10     public Formulario() {
11         setLayout(null);
12         combo1=new JComboBox<String>();
13         combo1.setBounds(10,10,80,20);
14         add(combo1);
15         combo1.addItem("rojo");
16         combo1.addItem("verde");
17         combo1.addItem("azul");
18         combo1.addItem("amarillo");
19         combo1.addItem("negro");
20         combo1.addItemListener(this);
21     }
22
23     public static void main(String[] args) {
24         Formulario f=new Formulario();
25         f.setBounds(0,0,300,200);
26         f.setVisible(true);
27         f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
28     }
29
```

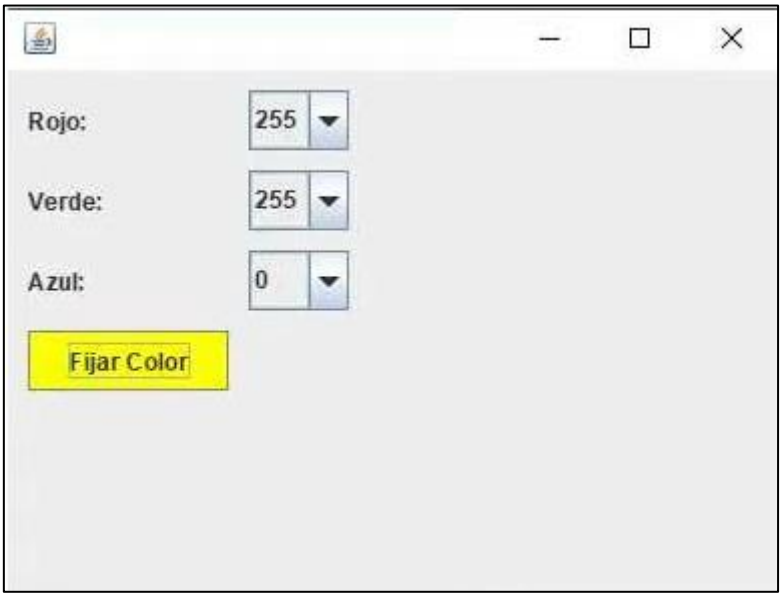
```
30      @Override
31      public void itemStateChanged(ItemEvent e) {
32          if(e.getSource()==combo1) {
33              String texto=combo1.getSelectedItem().toString();
34              setTitle(texto);
35          }
36      }
37  }
```

Si ejecutamos este será el resultado:

Capítulo 131.- Swing – JComboBox – 2

Problema:

Disponer tres controles de tipo JComboBox con valores entre 0 y 255 (cada uno representa la cantidad de tojo, verde y azul). Luego un botón pintar el mismo con el color que se genera combinando los valores de los JComboBox.

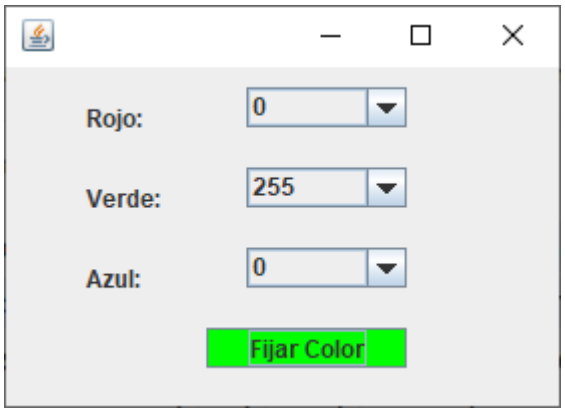


Vamos a programar:

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
// Para utilizar la clase Color
import java.awt.*;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JComboBox;
import javax.swing.JButton;

public class Formulario extends JFrame implements ActionListener{
	private JLabel label1, label2, label3;
	private JComboBox<String> combo1, combo2, combo3;
	private JButton boton1;

	public Formulario() {
		setLayout(null);
		label1=new JLabel("Rojo:");
		label1.setBounds(40,10,100,30);
		add(label1);
		combo1=new JComboBox<String>();
		combo1.setBounds(120,10,80,20);
		add(combo1);
		for(int x=0; x<=255; x++) {
			combo1.addItem(String.valueOf(x));
		}

		label2=new JLabel("Verde:");
		label2.setBounds(40,50,100,30);
		add(label2);
```

```
		combo2=new JComboBox<String>();
		combo2.setBounds(120,50,80,20);
		add(combo2);
		for(int x=0; x<=255; x++) {
			combo2.addItem(String.valueOf(x));
		}

		label3=new JLabel("Azul:");
		label3.setBounds(40,90,100,30);
		add(label3);
		combo3=new JComboBox<String>();
		combo3.setBounds(120,90,80,20);
		add(combo3);
		for(int x=0; x<=255; x++) {
			combo3.addItem(String.valueOf(x));
		}

		boton1=new JButton("Fijar Color");
		boton1.setBounds(100, 130, 100,20);
		add(boton1);
		boton1.addActionListener(this);
	}

	public static void main(String[] args) {
		Formulario f=new Formulario();
		f.setBounds(0,0,400,300);
		f.setVisible(true);
		f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		if(e.getSource()==boton1) {
			int
rojo=Integer.parseInt(combo1.getSelectedItem().toString());
			int
verde=Integer.parseInt(combo2.getSelectedItem().toString());
			int
azul=Integer.parseInt(combo3.getSelectedItem().toString());
			Color color1=new Color(rojo, verde, azul);
			boton1.setBackground(color1);
		}
	}
}
```

Si ejecutamos este será el resultado:



Capítulo 132.- Swing – JcomboBox – 3

Problema propuesto:

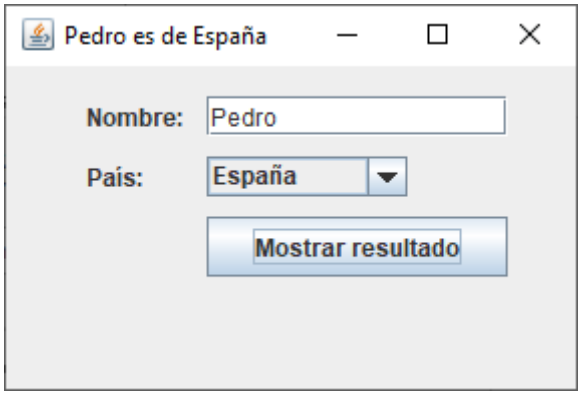
Solicitar el ingreso del nombre de una persona y seleccionar de un control JComboBox un país. Al presionar un botón mostrar en la barra de título del JFrame el nombre ingresado y el país seleccionado.

Vamos a programar:

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JComboBox;
import javax.swing.JButton;

public class Formulario extends JFrame implements ActionListener{
	private JLabel label1, label2;
	private JTextField texto1;
	private JComboBox<String> combo1;
	private JButton boton1;

	public Formulario() {
		setLayout(null);
		label1=new JLabel("Nombre:");
		label1.setBounds(40,10, 100, 30);
		add(label1);
		texto1=new JTextField();
		texto1.setBounds(100,15, 150,20);
		add(texto1);
		label2=new JLabel("País:");
		label2.setBounds(40, 40, 100, 30);
		add(label2);
		combo1=new JComboBox<String>();
		combo1.setBounds(100,45, 100,20);
		combo1.addItem("España");
		combo1.addItem("Francia");
		combo1.addItem("Alemania");
		combo1.addItem("Italia");
		combo1.addItem("Holanda");
		combo1.addItem("Polonia");
		add(combo1);
		boton1=new JButton("Mostrar resultado");
		boton1.setBounds(100,75,150,30);
		add(boton1);
		boton1.addActionListener(this);
	}

	public static void main(String[] args) {
		Formulario f=new Formulario();
		f.setBounds(100,100, 300,200);
		f.setVisible(true);
		f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}

	@Override
```

```
	public void actionPerformed(ActionEvent e) {
		if(e.getSource()==boton1) {
			String nombre=texto1.getText();
			String pais=combo1.getSelectedItem().toString();
			String mensaje=nombre+" es de "+pais;
			setTitle(mensaje);
		}
	}
}
```

Si ejecutamos este será el resultado:



Capítulo 133.- Swing – JMenuBar, JMenu, JMenuItem – 1

Cuando necesiamos implementar un menú horizontal en la parte superior de un JFrame requerimos de un objeto de la clase JMenuBar, uno o más objetos de la clase JMenu y por último objetos de la clase JMenuItem.

Para la captura de eventos debemos implementar la interface ActionListener y asociarlo a los controles de tipo JMenuItem, el mismo se dispara al presionar con el mouse el JMenuItem.

Problema:

Confeccionaremos un menú de opciones que contenga tres opciones que permita cambiar de color el fondo del JFrame a los colores: rojo, verde y azul.

Vamos a programar:

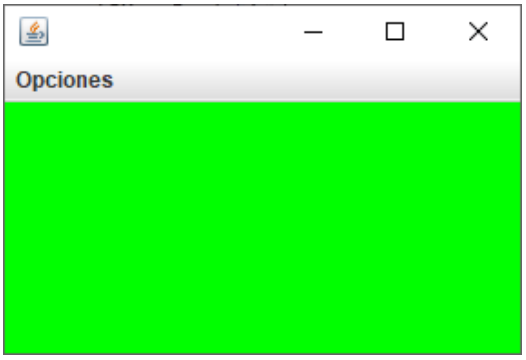
```
import java.awt.Color;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;

public class Formulario extends JFrame implements ActionListener{
	private JMenuBar mb;
	private JMenu menu1;
	private JMenuItem mi1, mi2, mi3;

	public Formulario() {
		setLayout(null);
		mb=new JMenuBar();
		setJMenuBar(mb);
		menu1=new JMenu("Opciones");
		mb.add(menu1);
		mi1=new JMenuItem("Rojo");
		menu1.add(mi1);
		mi2=new JMenuItem("Verde");
		menu1.add(mi2);
		mi3=new JMenuItem("Azul");
		menu1.add(mi3);
		mi1.addActionListener(this);
		mi2.addActionListener(this);
```

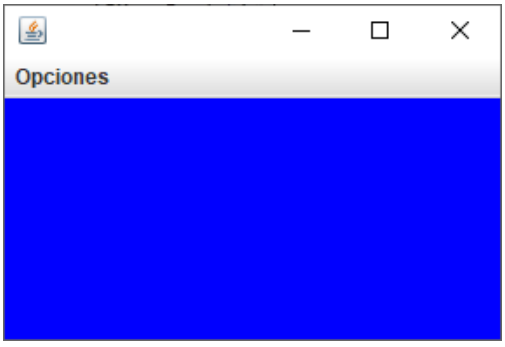
```
            mi3.addActionListener(this);
        }

        public static void main(String[] args) {
            Formulario f=new Formulario();
            f.setBounds(10,20,300,200);
            f.setVisible(true);
            f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            Container c=getContentPane();
            if (e.getSource()==mi1) {
                c.setBackground(new Color(255, 0, 0));
            }
            if (e.getSource()==mi2) {
                c.setBackground(new Color(0, 255, 0));
            }
            if (e.getSource()==mi3) {
                c.setBackground(new Color(0, 0, 255));
            }
        }
}
```

Creamos un objeto que contendrá el atributo getContentPane(), para poder cambiar el color de fondo del formulario.

Ahora la c en un objeto que podremos utilizar para cambiar el color de fondo.

Vamos a ejecutar, este será el resultado:





Si queremos que al inicio aparezca por defecto de color amarillo podremos:

```
33    public static void main(String[] args) {
34        Formulario f=new Formulario();
35        f.setBounds(10,20,300,200);
36        f.getContentPane().setBackground(new Color(255, 255, 0));
37        f.setVisible(true);
38        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
39    }
```

Capítulo 134.- Swing – JmenuBar, JMenu, JMenuItem – 2

Problema:

Confeccionaremos un menú de opciones que contenga además del JMenu de la barra otros dos objetos de la clase JMenu que dependan del primero.

Uno debe mostrar dos JMenuItem que permita modificar el tamaño del JFrame con las siguientes dimensiones (640*480 y 1024*768) y el segundo también debe mostrar dos JMenuItem que permita cambiar el color de fondo.

Vamos a programar:

```
import java.awt.Color;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;

public class Formulario extends JFrame implements ActionListener{
	private JMenuBar mb;
	private JMenu menu1, menu2, menu3;
	private JMenuItem mi1, mi2, mi3, mi4;

	public Formulario() {
		setLayout(null);
		mb=new JMenuBar();
		setJMenuBar(mb);
		menu1=new JMenu("Opciones");
		mb.add(menu1);
		menu2=new JMenu("Tamaño de la ventana");
		menu1.add(menu2);
		mi1=new JMenuItem("640*480");
		menu2.add(mi1);
		mi2=new JMenuItem("1024*768");
		menu2.add(mi2);
		menu3=new JMenu("Color de fondo");
		menu1.add(menu3);
		mi3=new JMenuItem("Rojo");
		menu3.add(mi3);
		mi4=new JMenuItem("Verde");
		menu3.add(mi4);
```

```
		mi1.addActionListener(this);
		mi2.addActionListener(this);
		mi3.addActionListener(this);
		mi4.addActionListener(this);

	}

	public static void main(String[] args) {
		Formulario f=new Formulario();
		f.setBounds(10,20,300,200);
		f.setVisible(true);
		f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

	}

	@Override
	public void actionPerformed(ActionEvent e) {
		if(e.getSource()==mi1) {
			setSize(640,480);
		}
		if(e.getSource()==mi2) {
			setSize(1024,768);
		}
		if(e.getSource()==mi3) {
			getContentPane().setBackground(new Color(255,0,0));
		}
		if(e.getSource()==mi4) {
			getContentPane().setBackground(new Color(0,255,0));
		}
	}
}
```

Si ejecutas podrás cambiar a color rojo o verde y modificar el tamaño a 640*480 o 1024*678.

Capítulo 135.- Swing – JMenuBar, JMenu, JMenuItem – 3

Problema propuesto:

Mediante dos controles de tipo JtextField permitir el ingreso de dos números. Crear un menú que contenga una opción que redimensione el JFrame con los valores ingresados por teclado.

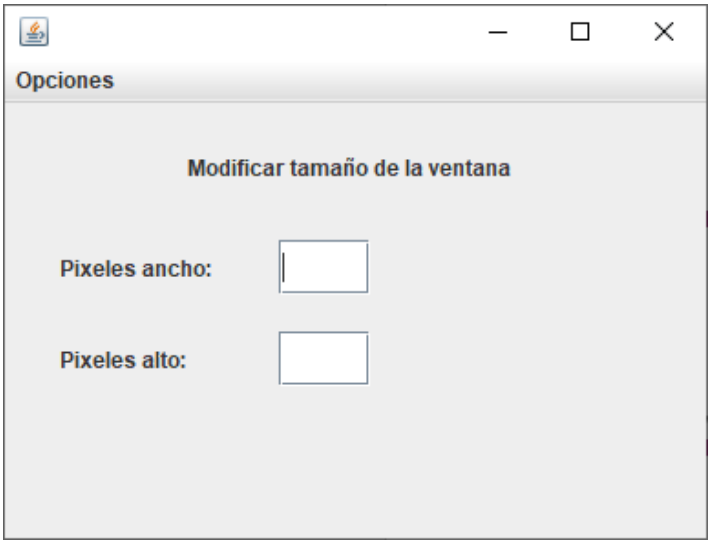
Finalmente disponer otra opción que finalice el programa (Finalizamos un programa java llamando el método exit de la clase System: System.exit(0)).

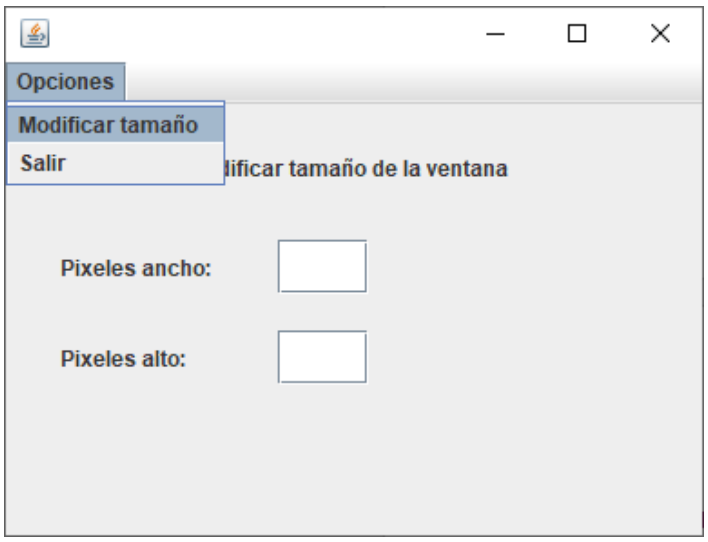
Vamos a programar:

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;

public class Formulario extends JFrame implements ActionListener{

	private JLabel label1, label2, label3;
	private JTextField texto1, texto2;
	private JMenuBar mb;
	private JMenu menu1;
	private JMenuItem mi1, mi2;

	public Formulario() {
		setLayout(null);
		label1=new JLabel("Modificar tamaño de la ventana");
		label1.setBounds(100,20, 200,30);
		add(label1);
		label2=new JLabel("Pixeles ancho:");
		label2.setBounds(30, 75, 200, 30);
		add(label2);
		texto1=new JTextField();
		texto1.setBounds(150, 75, 50, 30);
		add(texto1);
		label3=new JLabel("Pixeles alto:");
		label3.setBounds(30, 125, 200, 30);
		add(label3);
		texto2=new JTextField();
		texto2.setBounds(150, 125, 50, 30);
		add(texto2);

		mb=new JMenuBar();
		setJMenuBar(mb);
		menu1=new JMenu("Opciones");
		mb.add(menu1);
		mi1=new JMenuItem("Modificar tamaño");
		menu1.add(mi1);
		mi2=new JMenuItem("Salir");
		menu1.add(mi2);
		mi1.addActionListener(this);
		mi2.addActionListener(this);
	}
```

```
	public static void main(String[] args) {
		Formulario f=new Formulario();
		f.setBounds(10,20,400,300);
		f.setVisible(true);
		f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		if(e.getSource()==mi1) {
			int medidaH=Integer.parseInt(texto1.getText());
			int medidaV=Integer.parseInt(texto1.getText());
			setSize(medidaH,medidaV);
		}
		if(e.getSource()==mi2) {
			System.exit(0);
		}
	}
}
```

Este será el resultado:



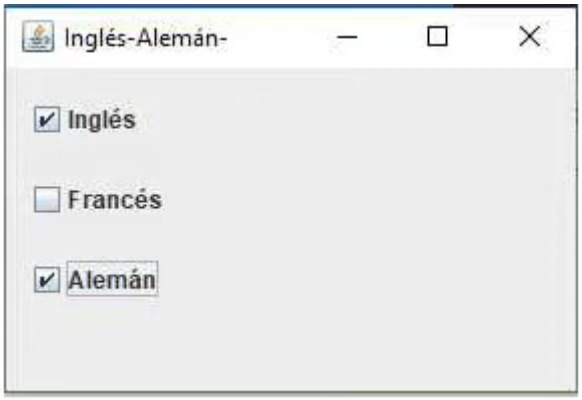


Capítulo 136.- Swing – JCheckBox – 1

El control JChechBox permite implementar un cuadro de selcción.

Problema:

Confeccionar un programa que muestre 3 objetos de la clase JCheckBox con etiquetas de tres idiomas. Cuando se lo selecciona mostrar en el título del JFrame todos los JCheckBox seleccionados hasta el momento.



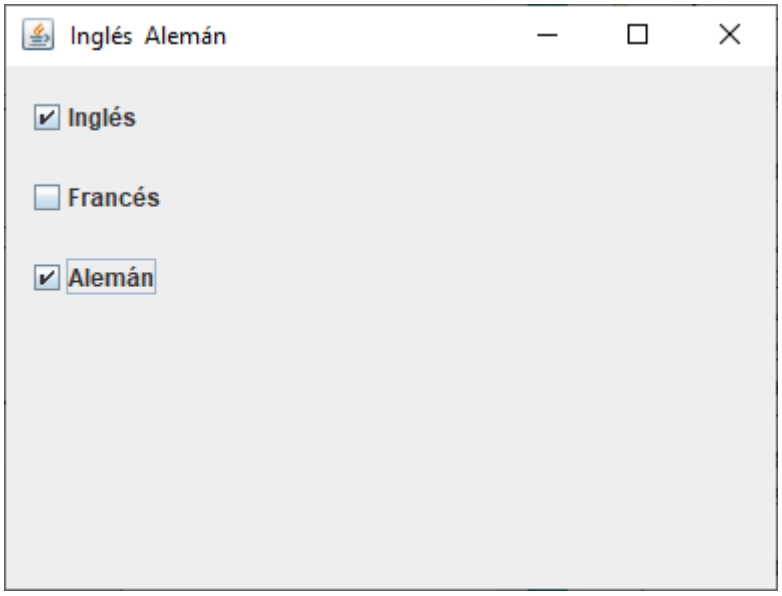
Vamos a programar:

Formulario.java

```
1  import javax.swing.JFrame;
2  import javax.swing.event.ChangeEvent;
3  import javax.swing.event.ChangeListener;
4  import javax.swing.JCheckBox;
5
6  public class Formulario extends JFrame implements ChangeListener{
7      private JCheckBox check1, check2, check3;
8
9      public Formulario() {
10         setLayout(null);
11         check1=new JCheckBox("Inglés");
12         check1.setBounds(10,10,150,30);
13         add(check1);
14         check2=new JCheckBox("Francés");
15         check2.setBounds(10,50,150,30);
16         add(check2);
17         check3=new JCheckBox("Alemán");
18         check3.setBounds(10,90,150,30);
19         add(check3);
20         check1.addChangeListener(this);
21         check2.addChangeListener(this);
22         check3.addChangeListener(this);
23     }
24
25     public static void main(String[] args) {
26         Formulario f=new Formulario();
27         f.setBounds(0, 0, 400, 300);
28         f.setVisible(true);
29         f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
30     }
31
```

```
32      @Override
33      public void stateChanged(ChangeEvent e) {
34          String cad="";
35          if(check1.isSelected()) {
36              cad+=" Inglés ";
37          }
38          if(check2.isSelected()) {
39              cad+=" Francés ";
40          }
41          if(check3.isSelected()) {
42              cad+=" Alemán ";
43          }
44          setTitle(cad);
45      }
46  }
```

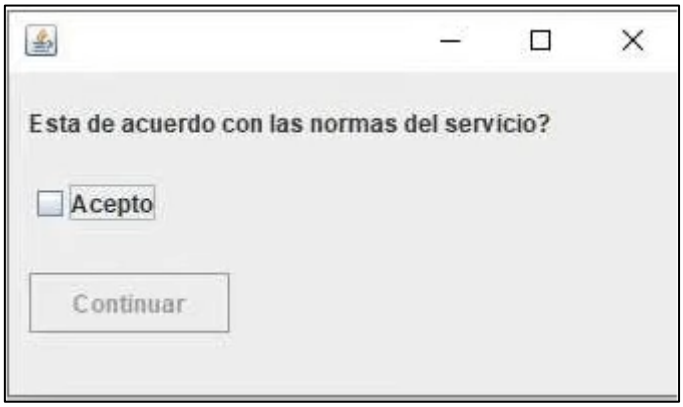
Vamos a ejecutar, este será el resultado:



Capítulo 137.- Swing – JCheckBox – 2

Problema:

Disponer un control JLabel que muestre el siguiente mensaje: "Está de acuerdo con las normas del servicio?", luego en JCheckBox y finalmente un objeto de tipo JButton desactivado. Cuando se tilde el JCheckBox debemos activar el botón.

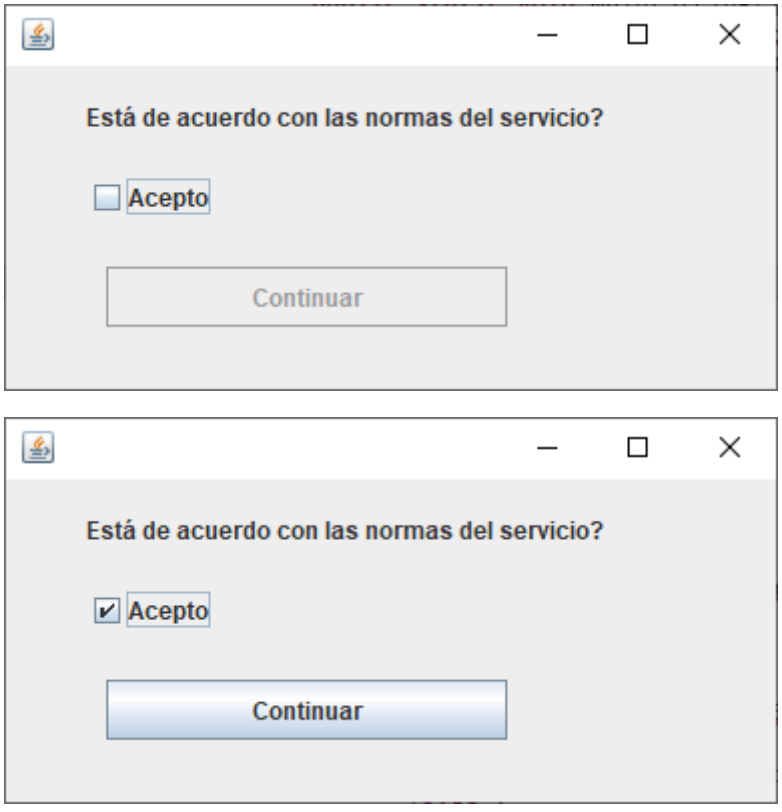


Vamos a programar:

```
import javax.swing.JFrame;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JLabel;
import javax.swing.JCheckBox;
import javax.swing.JButton;
public class Formulario extends JFrame implements ChangeListener,
ActionListener{
	private JLabel label1;
	private JCheckBox chek1;
	private JButton boton1;

	public Formulario() {
		setLayout(null);
		label1=new JLabel("Está de acuerdo con las normas del
servicio?");
		label1.setBounds(40,10, 300,30);
		add(label1);
		chek1=new JCheckBox("Acepto");
		chek1.setBounds(40,50, 200, 30);
		add(chek1);
		boton1=new JButton("Continuar");
		boton1.setBounds(50,100, 200, 30);
		boton1.setEnabled(false);
		add(boton1);
		chek1.addChangeListener(this);
		boton1.addActionListener(this);
	}

	public static void main(String[] args) {
		Formulario f=new Formulario();
		f.setBounds(50,50, 400,200);
```

```
            f.setVisible(true);
            f.setDefaultCloseOperation(EXIT_ON_CLOSE);
        }

        @Override
        public void stateChanged(ChangeEvent e) {
            if(chek1.isSelected()) {
                boton1.setEnabled(true);
            }else {
                boton1.setEnabled(false);
            }
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            if(e.getSource()==boton1) {
                System.exit(0);
            }
        }
}
```

Si ejecutamos este será el resultado:



Capítulo 138.- Swing – JCheckBox – 3

Problema propuesto:

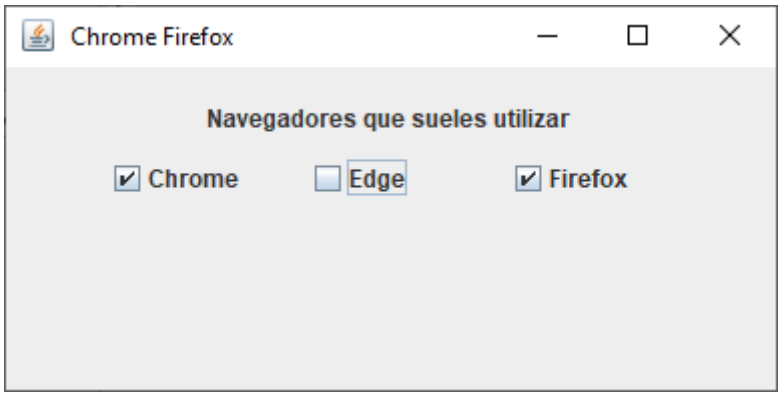
Disponer tres objetos de la clase JCheckBox con nombres de navegadores web. Cuando presione un botón mostrar en el título del JFrame los programas seleccionados.

Vamos a programar:

*Formulario.java

```
1  import javax.swing.JFrame;
2  import javax.swing.JLabel;
3  import javax.swing.JCheckBox;
4  import javax.swing.event.ChangeEvent;
5  import javax.swing.event.ChangeListener;
6
7  public class Formulario extends JFrame implements ChangeListener{
8      private JCheckBox chec1, chec2, chec3;
9      private JLabel label1;
10
11     public Formulario() {
12         setLayout(null);
13         label1=new JLabel("Navegadores que sueles utilizar");
14         label1.setBounds(100,10, 200,30);
15         add(label1);
16         chec1=new JCheckBox("Chrome");
17         chec1.setBounds(50,40, 100, 30);
18         add(chec1);
19         chec2=new JCheckBox("Edge");
20         chec2.setBounds(150,40, 100, 30);
21         add(chec2);
22         chec3=new JCheckBox("Firefox");
23         chec3.setBounds(250,40, 100, 30);
24         add(chec3);
25         chec1.addChangeListener(this);
26         chec2.addChangeListener(this);
27         chec3.addChangeListener(this);
28     }
29
30     public static void main(String[] args) {
31         Formulario f=new Formulario();
32         f.setBounds(100,100,400,200);
33         f.setVisible(true);
34         f.setDefaultCloseOperation(EXIT_ON_CLOSE);
35     }
36
37     @Override
38     public void stateChanged(ChangeEvent e) {
39         String navegadores="";
40         if(chec1.isSelected()) {
41             navegadores=navegadores+" Chrome";
42         }
43         if(chec2.isSelected()) {
44             navegadores=navegadores+" Edge";
45         }
46         if(chec3.isSelected()) {
47             navegadores=navegadores+" Firefox";
48         }
49         setTitle(navegadores);
50     }
51 }
```

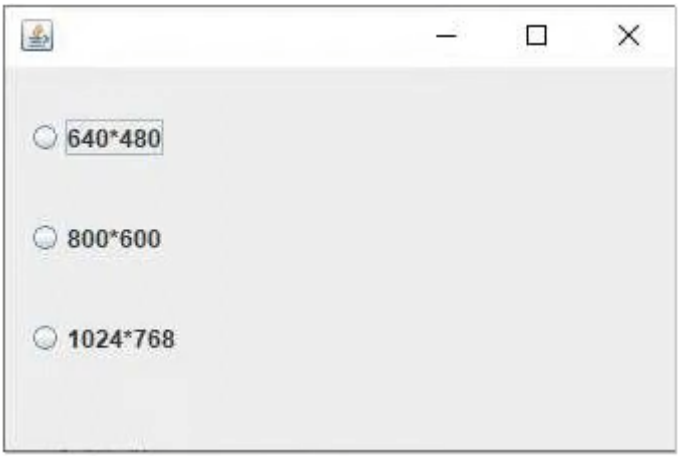
Si ejecutamos este será el resultado:



Capítulo 139.- Swing – JRadioButton – 1

Otro control visual muy común es el JRadioButton que normalmente se muestran en conjunto de JRadioButton y permite la selección de solo uno de ellos. Se los debe agrupar para que actúen en conjunto, es decir cuando se selecciona uno automáticamente se deben deseleccionar los otros.

Problema:



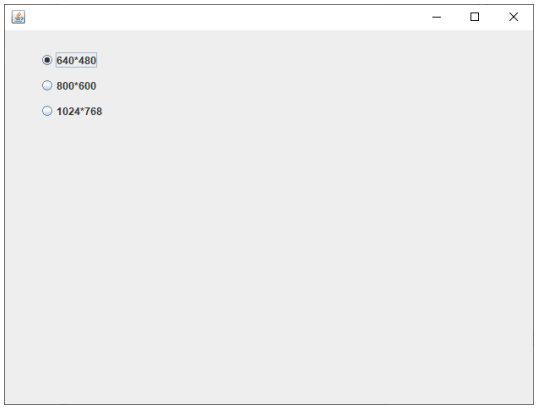
Vamos a programar:

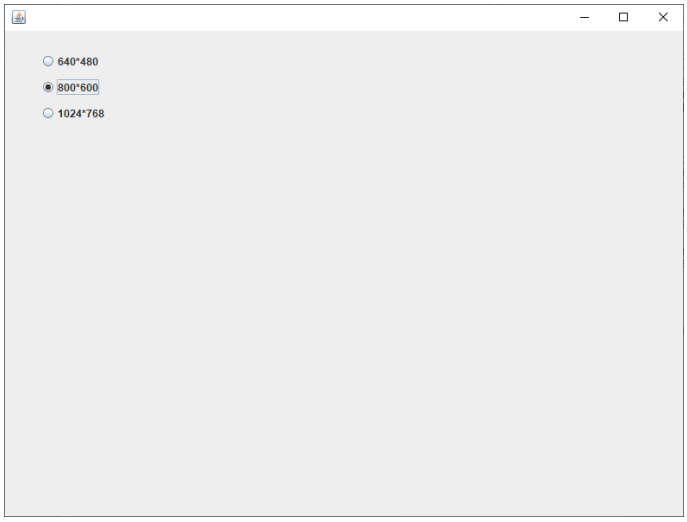
Formulario.java

```
1  import javax.swing.JFrame;
2  import javax.swing.JRadioButton;
3  import javax.swing.ButtonGroup;
4  import javax.swing.event.ChangeEvent;
5  import javax.swing.event.ChangeListener;
6
7  public class Formulario extends JFrame implements ChangeListener{
8      private JRadioButton radio1, radio2, radio3;
9      private ButtonGroup bg;
10
11     public Formulario() {
12         setLayout(null);
13         radio1=new JRadioButton("640*480");
14         radio1.setBounds(40,20, 200, 30);
15         add(radio1);
16         radio2=new JRadioButton("800*600");
17         radio2.setBounds(40,50, 200, 30);
18         add(radio2);
19         radio3=new JRadioButton("1024*768");
20         radio3.setBounds(40,80, 200, 30);
21         add(radio3);
22         bg=new ButtonGroup();
23         bg.add(radio1);
24         bg.add(radio2);
25         bg.add(radio3);
26         radio1.addChangeListener(this);
27         radio2.addChangeListener(this);
28         radio3.addChangeListener(this);
29     }
30
```

```
31      public static void main(String[] args) {
32          Formulario f=new Formulario();
33          f.setBounds(100,100, 400, 200);
34          f.setVisible(true);
35          f.setDefaultCloseOperation(EXIT_ON_CLOSE);
36      }
37
38      @Override
39      public void stateChanged(ChangeEvent e) {
40          if(radio1.isSelected()) {
41              setSize(640, 480);
42          }
43          if(radio2.isSelected()) {
44              setSize(800, 600);
45          }
46          if(radio3.isSelected()) {
47              setSize(1024, 768);
48          }
49      }
50  }
```

Vamos a ejecutar, este será el resultado:





Capítulo 140.- Swing – JRadioButton – 2

Problema propuesto:

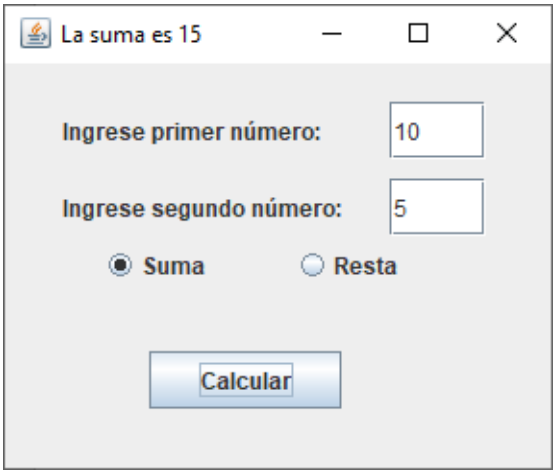
Permitir el ingreso de dos números en controles de tipo JTextFields y mediante dos controles de tipo JRadioButon permitir seleccionar si queremos sumarlos o restarlos. Al presionar un botón mostrar en el título del JFrame el resultado de la operación.

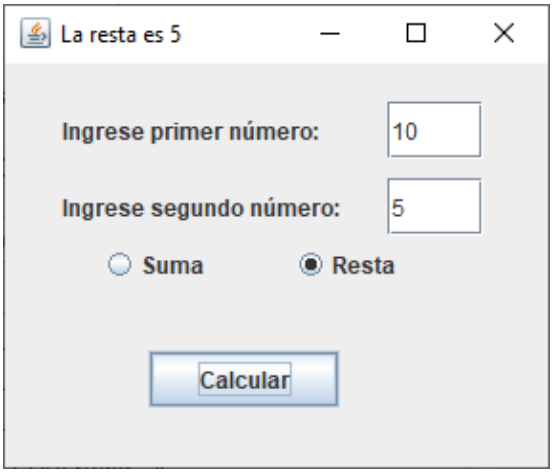
Vamos a programar:

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JRadioButton;
import javax.swing.ButtonGroup;
import javax.swing.JButton;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Formulario extends JFrame implements ActionListener {
	JLabel label1, label2;
	JTextField texto1, texto2;
	JRadioButton suma, resta;
	ButtonGroup bg;
	JButton calcular;

	public Formulario() {
		setLayout(null);
		label1=new JLabel("Ingrese primer número:");
		label1.setBounds(30,20, 300,30);
		add(label1);
		texto1=new JTextField();
		texto1.setBounds(200,20,50,30);
		add(texto1);
		label2=new JLabel("Ingrese segundo número:");
		label2.setBounds(30,60,300,30);
		add(label2);
		texto2=new JTextField();
		texto2.setBounds(200,60,50,30);
		add(texto2);
		suma=new JRadioButton("Suma");
		suma.setBounds(50,90,100,30);
		suma.setSelected(true);
		add(suma);
		resta=new JRadioButton("Resta");
		resta.setBounds(150,90,100,30);
		add(resta);
		bg=new ButtonGroup();
		bg.add(suma);
		bg.add(resta);
		calcular=new JButton("Calcular");
		calcular.setBounds(75, 150, 100,30);
		add(calcular);
		calcular.addActionListener(this);
	}

	public static void main(String[] args) {
		Formulario f=new Formulario();
```

```
            f.setBounds(100,100,300,250);
            f.setVisible(true);
            f.setDefaultCloseOperation(EXIT_ON_CLOSE);
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            if(e.getSource()==calcular) {
                if(suma.isSelected()) {
                    int num1=Integer.parseInt(texto1.getText());
                    int num2=Integer.parseInt(texto2.getText());
                    int sum=num1+num2;
                    String cadenaSuma="La suma es "+sum;
                    setTitle(cadenaSuma);
                }
                if(resta.isSelected()) {
                    int num1=Integer.parseInt((texto1.getText()));
                    int num2=Integer.parseInt((texto2.getText()));
                    int res=num1-num2;
                    String cadenaResta="La resta es "+res;
                    setTitle(cadenaResta);
                }
            }
        }
}
```

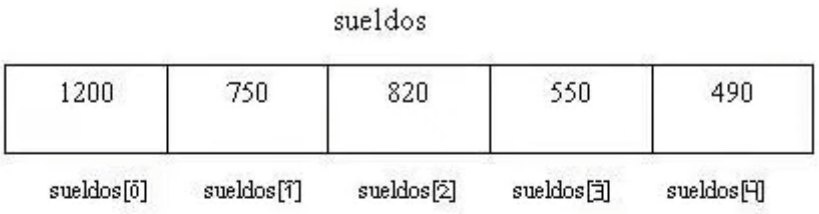
Si ejecutamos este será el resultado:

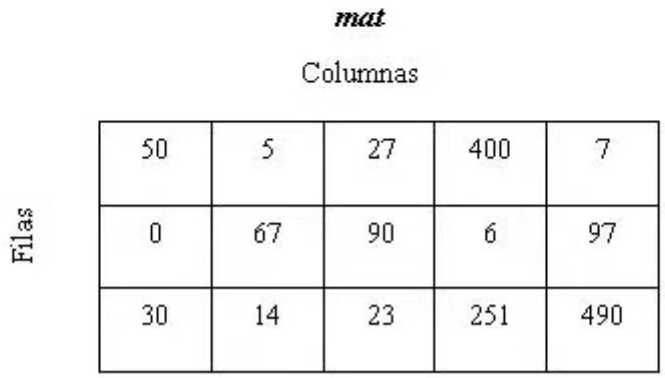




Capítulo 141.- Estructuras dinámicas: Listas

Estructura estática: Vectores y Matrices:

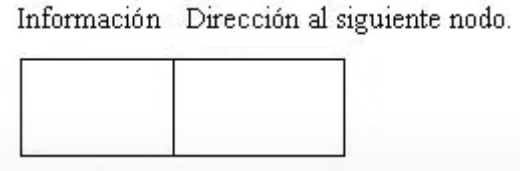




Estructura dinámica: Listas:

Una lista es un conjunto de nodos, cada uno de los cuales tiene dos campos: uno de información y un apuntador al siguiente nodo de la lista. Además un apuntador externo señala el primir nodo de la lista.

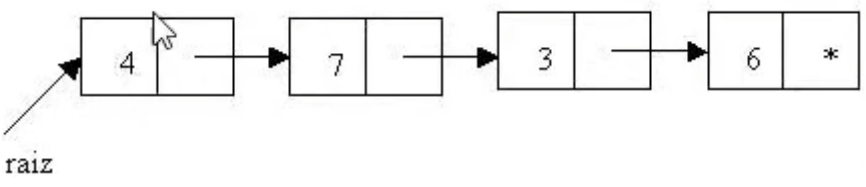
Representación gráfica de un nodo.



La información puede ser cualquier tipo de dato simple, estructura de datos o inclusive uno o más objetos.

La dirección al siguiente nodo es un puntero.

Representación gráfica de una lista:



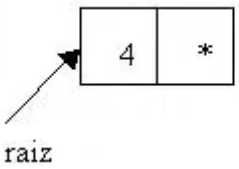
Como decíamos, una lista es una secuencia de nodos (en este caso cuatro nodos). La información de los nodos en este caso es un entero y siempre contiene un puntero que guarda la dirección del siguiente nodo.

Raiz es otro puntero externo a la lista que contiene la dirección del primer nodo.

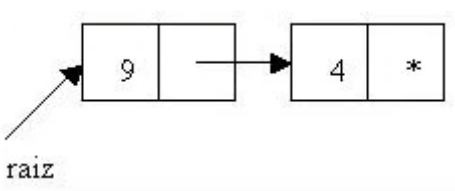
El estado de una lista vacía durante la ejecución del programa:

De esta forma representamos gráficamente una lista vacía.

Si insertamos un nodo en la lista luego:



Si insertamos otro nodo al principio con el valor 9 tenemos:



Lo mismo podemos borrar nodos de cualquier parte de la lista.

Esto nos trae a la mente el primer problema planteado: el desarrollo del procesador de textos.

Podríamos utilizar una lista que inicialmente estuviera vacía e introdujéramos un nuevo nodo con cada línea que escribiera el operador. Con esta estructura haremos un uso muy eficiente de la memoria.

Tipo de listas.

- Lista tipo pila.
- Lista tipo cola.
- Listas genéricas.

Una lista se comporta como una pila si las inserciones y extracciones las hacemos por un mismo lado de la lista. También se las llama listas LIFO (Last In First Out) - último en entrar primero en salir).

Una lista se comporta como una cola si las inserciones las hacemos al final y las extracciones las hacemos por el frente de la lista. También se las llama listas FIFO (First In First Out – primero en entrar primero en salir).

Una lista se comporta como genérica cuando las inserciones y extracciones se realizan en cualquier parte de la lista.

Podemos en algún momento insertar un nodo en medio de la lista, en otro momento al final, borrar uno del frente, borrar uno del fondo o uno interior, etc.

Capítulo 142.- Estructuras dinámicas Listas tipo Pila

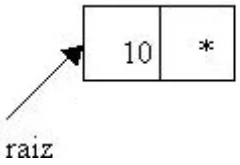
Una lista se comporta como una pila si las inserciones y extracciones las hacemos por un mismo lado de la lista. También se las llama listas LIFO (Last In First OUT – último en entrar primero en salir).

Importante: Una pila al ser una lista puede almacenar en el campo de la información cualquier tipo de valor (int, char, float, vectores, un objeto, etc).

Para estudiar el mecanismo de utilización de una pila supondremos que en el campo de información almacena un entero (para una fácil interpretación y codificación).

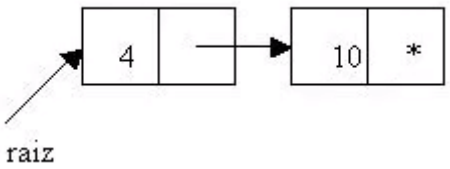
Inicialmente la PILA está vacía y decimos que el puntero raíz apunta a null (Si apunta a null decimos que no tiene una dirección en memoria):

Insertamos un valor entero en la pila: insertar(10)



Luego al realzar la inserción la lista tipo pila queda de esta manera: un nodo con el valor de 10 y raíz apunta a dicho nodo. El puntero del nodo apunta a null ya que no hay otro nodo después de este.

Insertamos luego el valor 4: insertar(4)

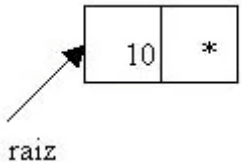


Ahora el primer nodo de la pila es el que almacena el valor cuatro. Raíz apunta a dicho nodo.

Recordemos que raíz es el puntero externo a la lista que almacena la dirección del primer nodo. El nodo que acabamos de insertar en el campo puntero guarda la dirección del nodo que almacena el valor 10.

Ahora qué sucede so extraemos un nodo de la pila. ¿Cuál se extrae? Como sabemos en una pila se extrae el último en entrar.

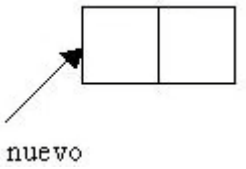
Al extraer de la pila tenemos: extraer()

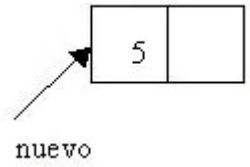


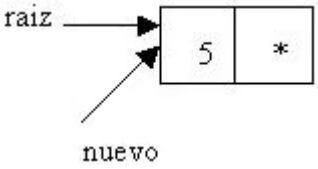
La pila ha quedado con un nodo.

Hay que tener cuidado que si se extrae un nuevo nodo la pila quedará vacía y no se podrá extraer otros valores (avisar que la pila está vacía).

Insertar en una pila vacía:







Pila.java

```java
public class Pila {
    class Nodo{
        int info;
        Nodo sig;
    }

    private Nodo raiz;

    public Pila() {
        raiz=null;
    }

    public void insertar(int x) {
        Nodo nuevo=new Nodo();
        nuevo.info=x;
        if(raiz==null) {
            raiz=nuevo;
            nuevo.sig=null;
        }else {
            nuevo.sig=raiz;
            raiz=nuevo;
        }
    }
    public void imprimir() {
        Nodo reco=raiz;
        while(reco!=null) {
            System.out.print(reco.info+"-");
            reco=reco.sig;
        }
        System.out.println();
    }

```

```
33      public static void main(String[] args) {
34          Pila pila1=new Pila();
35          pila1.insertar(5);
36          pila1.insertar(40);
37          pila1.insertar(45);
38          pila1.imprimir();
39      }
40  }
```

Si ejecutamos este será el resultado:

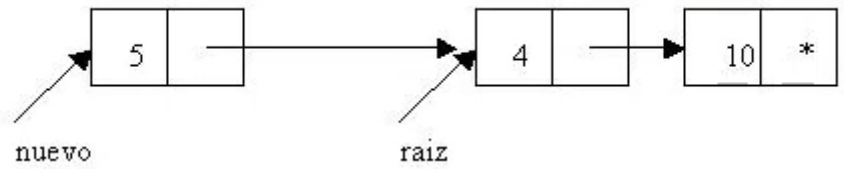
```
45-40-5-
```

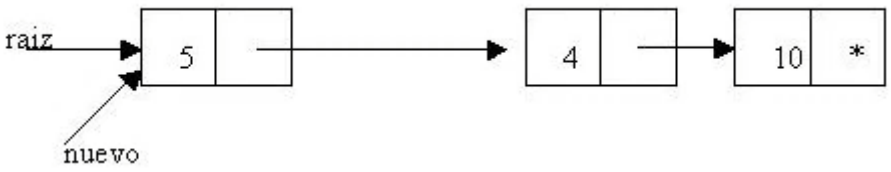
El último a insertar es el primero que está en la lista.

```
33      public static void main(String[] args) {
34          Pila pila1=new Pila();
35          pila1.insertar(5);
36          pila1.insertar(40);
37          pila1.insertar(45);
38          pila1.imprimir();
39          pila1.insertar(200);
40          pila1.imprimir();
41      }
42  }
```

Añadimos estas dos líneas y ejecutamos de nuevo.

```
45-40-5-
200-45-40-5-
```





Recorrer una lista para imprimirla:

```
reco=raiz;
```

Extraer un nodo:

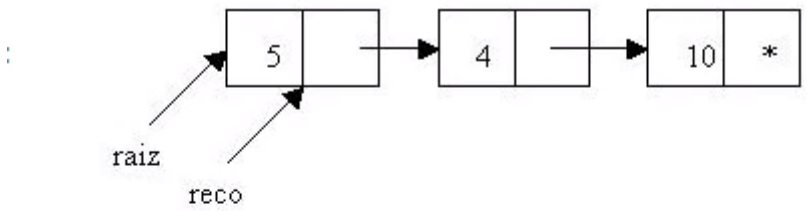
```
33      public int extraer() {
34          if(raiz==null) {
35              return Integer.MAX_VALUE;
36          }else {
37              int informacion=raiz.info;
38              raiz=raiz.sig;
39              return informacion;
40          }
41      }
```

Agregamos estas dos líneas:

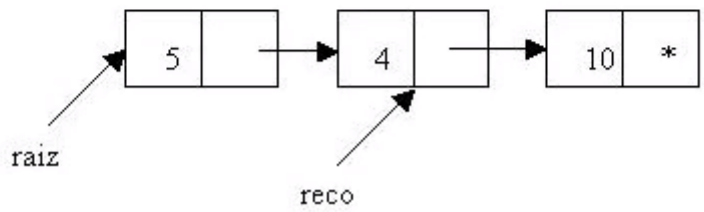
```
43      public static void main(String[] args) {
44          Pila pila1=new Pila();
45          pila1.insertar(5);
46          pila1.insertar(40);
47          pila1.insertar(45);
48          pila1.imprimir();
49          pila1.insertar(200);
50          pila1.imprimir();
51          System.out.println("Extraemos el primer nodo de la lista "+pila1.extraer());
52          pila1.imprimir();
53      }
54  }
```

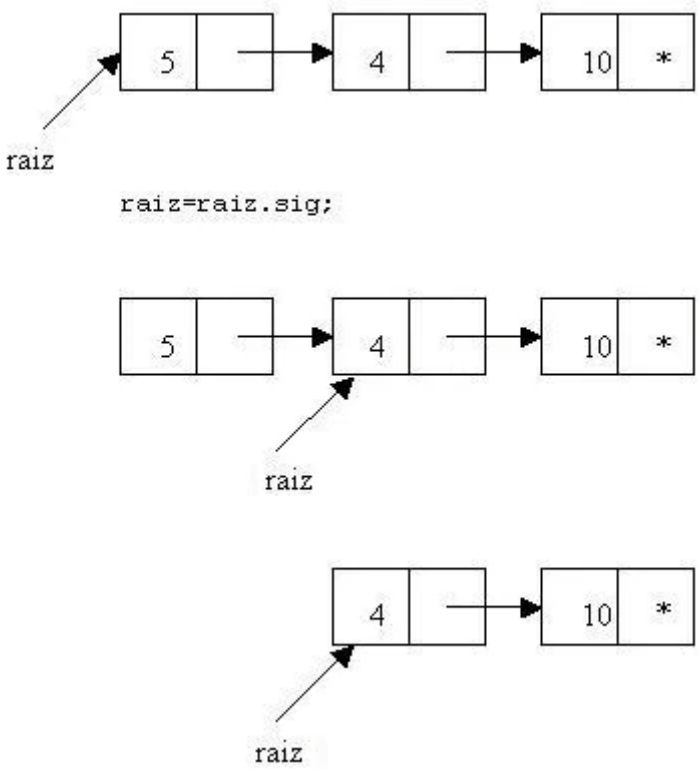
Ejecutamos de nuevo:

```
45-40-5-
200-45-40-5-
Extraemos el primer nodo de la lista 200
45-40-5-
```




```
reco=reco.sig;
```





Problema:

Agregar a la clase Pila un método que retorne la cantidad de nodos y otro que indique si está vacía.

```
44      public int cantidad() {
45          int cant=0;
46          Nodo reco=raiz;
47          while(reco!=null) {
48              reco=reco.sig;
49              cant++;
50          }
51          return cant;
52      }
```

Agregamos un línea al método main.

```
55      public static void main(String[] args) {
56          Pila pila1=new Pila();
57          pila1.insertar(5);
58          pila1.insertar(40);
59          pila1.insertar(45);
60          pila1.imprimir();
61          pila1.insertar(200);
62          pila1.imprimir();
63          System.out.println("Extraemos el primer nodo de la lista "+pila1.extraer());
64          pila1.imprimir();
65          System.out.println("Cantidad de nodos de la lista tipo pila:"+pila1.cantidad());
66      }
67  }
```

Ejecutamos de nuevo:

```
45-40-5-
200-45-40-5-
Extraemos el primer nodo de la lista 200
45-40-5-
Cantidad de nodos de la lista tipo pila:3
```

Comprobar si la pila está vacía.

```
    public boolean vacia() {
        if(raiz==null) {
            return true;
        }else {
            return false;
        }
    }
```

```
    public static void main(String[] args) {
        Pila pila1=new Pila();
        pila1.insertar(5);
        pila1.insertar(40);
        pila1.insertar(45);
        pila1.imprimir();
        pila1.insertar(200);
        pila1.imprimir();
        System.out.println("Extraemos el primer nodo de la lista "+pila1.extraer());
        pila1.imprimir();
        System.out.println("Cantidad de nodos de la lista tipo pila:"+pila1.cantidad());
        if (pila1.vacia()) {
            System.out.println("Pila vacía.");
        }else {
            System.out.println("La pila no está vacía.");
        }
    }
}
```

Ejecutamos de nuevo:

```
45-40-5-
200-45-40-5-
Extraemos el primer nodo de la lista 200
45-40-5-
Cantidad de nodos de la lista tipo pila:3
La pila no está vacía.
```

Agregar un método en la clase Pila que retorne la información del primer nodo de la Pila sin borrarlo.

```
    public int retornar() {
        if(raiz==null) {
            return Integer.MAX_VALUE;
        }else {
            return raiz.info;
        }
    }
```

```
70      public static void main(String[] args) {
71          Pila pila1=new Pila();
72          pila1.insertar(5);
73          pila1.insertar(40);
74          pila1.insertar(45);
75          pila1.imprimir();
76          pila1.insertar(200);
77          System.out.println("El primer elemento de la pila:"+pila1.retornar());
78          pila1.imprimir();
79          System.out.println("Extraemos el primer nodo de la lista "+pila1.extraer());
80          pila1.imprimir();
81          System.out.println("Cantidad de nodos de la lista tipo pila:"+pila1.cantidad());
82          if (pila1.vacia()) {
83              System.out.println("Pila vacía.");
84          }else {
85              System.out.println("La pila no está vacía.");
86          }
87      }
```

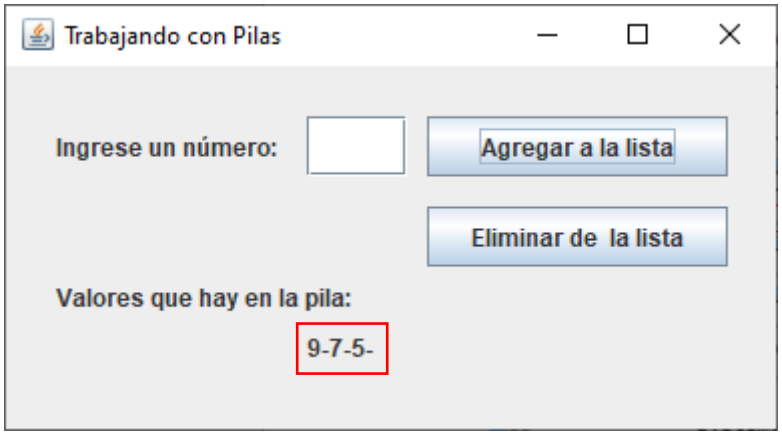
Ejecutamos de nuevo:

```
45-40-5-
El primer elemento de la pila:200
200-45-40-5-
Extraemos el primer nodo de la lista 200
45-40-5-
Cantidad de nodos de la lista tipo pila:3
La pila no está vacía.
```

Problema:

Partiendo con parte del código de la clase Pila, vamos a realizar un formulario donde podremos agregar números de tipo entero en la pila, así como su eliminación.

Este tiene que ser el formulario:



La lista que se vaya creando tiene que aparecer en la parte inferior.

Este tiene que ser el código de la clase Pila:

```
public class Pila {
	class Nodo{
		int info;
		Nodo sig;
	}

	private Nodo raiz;

	public Pila() {
```

```
            raiz=null;
        }

        public void insertar(int x) {
            Nodo nuevo=new Nodo();
            nuevo.info=x;
            if(raiz==null) {
                raiz=nuevo;
                nuevo.sig=null;
            }else {
                nuevo.sig=raiz;
                raiz=nuevo;
            }
        }

        public int extraer() {
            if(raiz==null) {
                return Integer.MAX_VALUE;
            }else {
                int informacion=raiz.info;
                raiz=raiz.sig;
                return informacion;
            }
        }


        public String retornar() {
            String texto="";
            if(raiz==null) {
                return "[ ]";
            }else {
                Nodo reco=raiz;
                while(reco!=null) {
                    texto=texto+String.valueOf(reco.info)+"-";
                    reco=reco.sig;
                }
                return texto;
            }
        }
}
```

Este es el código de la clase Formulario:

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Formulario extends JFrame implements ActionListener{
        JLabel label1, label2, label3;
        JTextField numero;
        JButton boton1, boton2;
        Pila pila1;

        public Formulario() {
            setLayout(null);
```

```
		pila1=new Pila();
		label1=new JLabel("Ingrese un número:");
		label1.setBounds(25,25,200,30);
		add(label1);
		numero=new JTextField();
		numero.setBounds(150,25,50,30);
		add(numero);
		boton1=new JButton("Agregar a la lista");
		boton1.setBounds(210,25,150,30);
		add(boton1);
		boton2=new JButton("Eliminar de  la lista");
		boton2.setBounds(210,70,150,30);
		add(boton2);
		label2=new JLabel("Valores que hay en la pila:");
		label2.setBounds(25, 100, 200,30);
		add(label2);
		label3=new JLabel("");
		label3.setBounds(150, 125, 200, 30);
		add(label3);
		boton1.addActionListener(this);
		boton2.addActionListener(this);
	}


	public static void main(String[] args) {
		Formulario f=new Formulario();
		f.setBounds(100,100, 400,220);
		f.setTitle("Trabajando con Pilas");
		f.setVisible(true);
		f.setDefaultCloseOperation(EXIT_ON_CLOSE);
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		if(e.getSource()==boton1) {

			int num=Integer.parseInt(numero.getText());
			pila1.insertar(num);
			label3.setText(pila1.retornar());
			numero.setText("");
		}
		if(e.getSource()==boton2) {
			pila1.extraer();
			label3.setText(pila1.retornar());
		}
	}
}
```

Capítulo 143.- Estructuras dinámicas: Lista tipo Pila – Problema de aplicación

Implementar un algoritmo que analice si una expresión está correctamente codificada, es decir que los paréntesis, corchetes y llaves estén abiertos y cerrados en un orden lógico y bien balanceados.

Se debe desarrollar una clase que tenga las siguientes responsabilidades (clase Formula):

- Ingresar kuna fórmula que contenga paréntesis, corchetes y llaves.
- Validar que los ( ) [ ] y { } estén correctamente balanceados.

Para la solución de este problema la clase Formula tendrá un atributo de la clase Pila.

Veamos cómo nos puede ayudar el empleo de una pila para solucionar este problema.

Primero cargaremos la fórmula en un JTextField.

Ejemplo de fórmula correcta: (2+[3-12]*{8/3})

Ejemplo de fórmula incorrecta: (2+]3-12]*{8/3})

Primero declaramos y definimos la clase Pila. Almacenamos en cada nodo un carácter y llamamos al campo de información símbolo.

La clase Pila.

```
public class Pila {

       class Nodo {
           char simbolo;
           Nodo sig;
       }

    private Nodo raiz;

    Pila () {
        raiz=null;
    }

    public void insertar(char x) {
       Nodo nuevo;
        nuevo = new Nodo();
        nuevo.simbolo = x;
        if (raiz==null)
        {
            nuevo.sig = null;
            raiz = nuevo;
        }
        else
        {
            nuevo.sig = raiz;
            raiz = nuevo;
        }
    }

    public char extraer ()
    {
        if (raiz!=null)
```

```
        {
            char informacion = raiz.simbolo;
            raiz = raiz.sig;
            return informacion;
        }
        else
        {
            return Character.MAX_VALUE;
        }
    }

    public boolean vacia() {
        if (raiz==null) {
            return true;
        } else {
            return false;
        }
    }
}
```

La clase Formula:

```
import javax.swing.*;
import java.awt.event.*;
public class Formula extends JFrame implements ActionListener {
    private JTextField tf1;
    private JButton boton1;
    public Formula() {
        setLayout(null);
        tf1=new JTextField("{2*(4-5)-{3*4}-[4-5]}");
        tf1.setBounds(10,10,230,30);
        add(tf1);
        boton1=new JButton("Verificar fórmula.");
        boton1.setBounds(10,70,180,30);
        add(boton1);
        boton1.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==boton1) {
           if (balanceada()) {
               setTitle("Está correctamente balanceada.");
           } else {
               setTitle("No está correctamente balanceada.");
           }
        }
    }

    public boolean balanceada() {
        Pila pila1;
      pila1 = new Pila ();
      String cadena=tf1.getText();
      for (int f = 0 ; f < cadena.length() ; f++)
      {
          if (cadena.charAt(f) == '(' || cadena.charAt(f) == '[' ||
cadena.charAt(f) == '{') {
             pila1.insertar(cadena.charAt(f));
           } else {
```

```
            if (cadena.charAt(f)==')') {
                if (pila1.extraer()!='(') {
                    return false;
                }
            } else {
                if (cadena.charAt(f)==']') {
                    if (pila1.extraer()!='[') {
                        return false;
                    }
                } else {
                    if (cadena.charAt(f)=='}') {
                        if (pila1.extraer()!='{') {
                            return false;
                        }
                    }
                }
              }
           }
         }
       if (pila1.vacia()) {
           return true;
       } else {
           return false;
       }
    }

    public static void main(String[] ar) {
        Formula formula1=new Formula();
        formula1.setBounds(0,0,360,160);
        formula1.setVisible(true);
        formula1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Planteo del problema:

Este práctico tiene por objetivo mostrar la importancia de las pilas en la Ciencia de la informática y más precisamente en la programación de software de bajo nivel.

Todo copilador o intérprete de un lenguaje tiene un módulo dedicado a analizar si una expresión está correctamente codificada, es decir que los paréntesis estén abiertos y cerrados en un orden lógico y bien balanceados.

Se debe desarrollar una clase que tenga las siguientes responsabilidades (clase Formula):

- Ingresar una fórmula que contenga paréntesis, corchetes y llaves.
- Validar que los ( ) [ ] y { } estén correctamente balanceados.

Para la solución de este problema la clase formula tendrá un atributo de la clase Pila.

Veamos cómo nos puede ayudar el empleo de una pila para solucionar este problema.

Primero cargaremos la formula en un JTextField.

Ejemplo de la fórmula: (2+[3-12]*{8/3}).

El algoritmo de validación es el siguiente:

Analizamos carácter a carácter la presencia de los paréntesis, corchetes y llaves.

Si vemos símbolos de apertura los almacenamos en la pila.

Si vienen símbolos de cerrado los extraemos de la pila y verificamos si está el mismo símbolo pero de apertura: en caso negativo podemos inferir que la fórmula no está correctamente balanceada.

Si al finalizar el análisis del último carácter de la fórmula la pila está vacía podemos concluir que está correctamente balanceada.

Ejemplos de fórmulas no balanceadas:

}(2+[3-12]*{8/3})

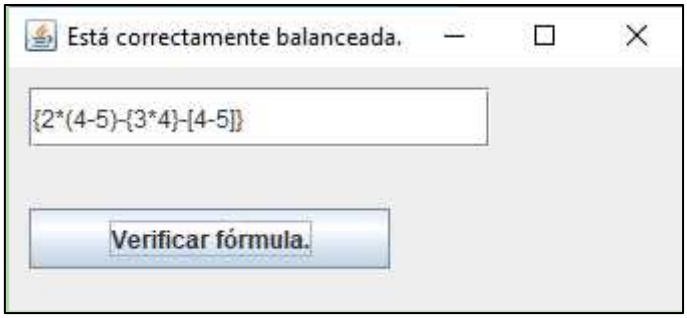
Incorrecta: llega una } de cerrado y la pila está vacía.

{[2+4}]

Incorrecta: llega una llave } y en el tope de la pila hay un corchete [.

{[2+4}

Incorrecta: al finalizar el análisis del último carácter en la pila queda pendiente una llave {.



Primero declaramos y definimos la clase Pila. Almacenamos en cada nodo un carácter y llamamos al campo de información símbolo.

No es necesario implementar los métodos imprimir, cantidad, etc. Poque no se requieren para este problema.

La clase Formula tiene como atributos:

```
private JTextField tf1;
private JButton boton1;
```

En el constructor creamos los dos objetos y los ubicamos:

```
setLayout(null);
tf1=new JTextField("{2*(4-5)-{3*4}-[4-5]}");
tf1.setBounds(10,10,230,30);
add(tf1);
boton1=new JButton("Verificar fórmula.");
boton1.setBounds(10,70,180,30);
 add(boton1);
 boton1.addActionListener(this);
```

En el méodo actionPerformed llamamos al método balanceada que debe retornar si la fórmula están correctos los paréntesis, corchetes y llaves.

```
if (e.getSource()==boton1) {
      if (balanceada()) {
            setTitle("Está correctamente balanceada.");
      } else {
            setTitle("No está correctamente balanceada.");
      }
}
```

El método más importante es el balanceada.

En este analizamos la fórmula para verificar si está correctamente balanceada.

En este método es donde está gran parte del algoritmo de este problema. Retornar true en caso de ser correcta y false en caso contrario.

Definimos una pila y extraemos el contenido del JTextField:

```
Pila pila1;
pila1 = new Pila ();
String cadena=tf1.getText();
```

El for se repite tantas veces como caracteres tenga el JTextField.

Se debe procesar sólo los símbolos ([{ y)] }.

Si el símbolo es un ([{ de apertura procedemos a cargarlo en la pila:

```
if (cadena.charAt(f) == '(' || cadena.charAt(f) == '[' || cadena.charAt(f) == '{') {
      pila1.insertar(cadena.charAt(f));
```

En caso de ser un) cerrado debemos extraer un carácter de la pila y verificar si no coincide con el paréntesis de apertura (la fórmula es incorrecta):

```
if (cadena.charAt(f)==')') {
   if (pila1.extraer()!='(') {
      return false;
   }
```

El mismo proceso es para los símbolos] }.

Al finalizar el análisis de toda la cadena se la pila está vacía podemos afirmar que la formula está correctamente balanceada, en caso contrario quiere decir que faltan símbolos de cerrado y es incorrecta:

```
if (pila1.vacia()) {
      return true;
} else {
      return false;
}
```

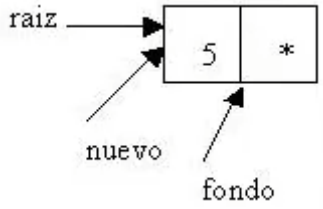
Es importante entender que la clase Formula utiliza un objeto de la clase Pila para resolver el algoritmo de verificar el balanceo de la formula, pero no accede directamente a los nodos de la lista.

Capítulo 144.- Estructuras dinámicas: Lista tipo Cola

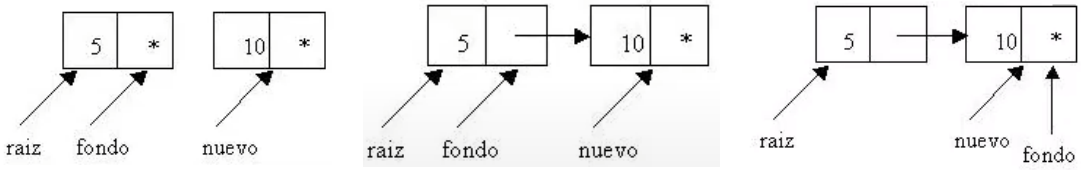
Una lista se comporta como una cola si las inserciones las hacemos al final y las extracciones las hacemos por el frente de la lista. También se las llama FIFO (First In First OUT -primero de entrar primero en salir).

Método insertar:

Si la lista está vacía:



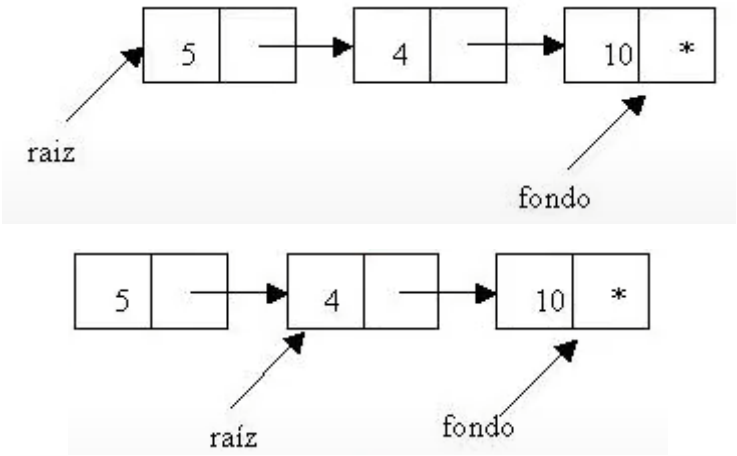
Si la lista no está vacía:



Método extraer:

Si hay un solo nodo:

en caso de haber 2 o más nodos debemos avanzar el puntero raíz al siguiente nodo:



Este será el código:

```
public class Cola {

    class Nodo {
        int info;
        Nodo sig;
    }

    private Nodo raiz,fondo;

    Cola() {
        raiz=null;
        fondo=null;
    }

    boolean vacia (){
        if (raiz == null)
            return true;
        else
            return false;
    }

    void insertar (int info)
    {
        Nodo nuevo;
        nuevo = new Nodo ();
        nuevo.info = info;
        nuevo.sig = null;
        if (vacia ()) {
            raiz = nuevo;
            fondo = nuevo;
        } else {
            fondo.sig = nuevo;
            fondo = nuevo;
        }
    }

    int extraer ()
    {
        if (!vacia ())
        {
            int informacion = raiz.info;
            if (raiz == fondo){
                raiz = null;
                fondo = null;
            } else {
                raiz = raiz.sig;
            }
            return informacion;
        } else
            return Integer.MAX_VALUE;
    }

    public void imprimir() {
        Nodo reco=raiz;
        System.out.println("Listado de todos los elementos de la cola.");
        while (reco!=null) {
            System.out.print(reco.info+"-");
```

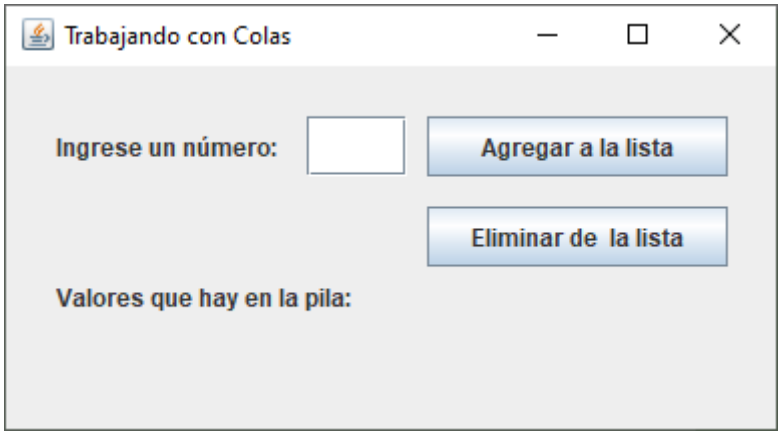
```
            reco=reco.sig;
        }
        System.out.println();
    }

    public static void main(String[] ar) {
        Cola cola1=new Cola();
        cola1.insertar(5);
        cola1.insertar(10);
        cola1.insertar(50);
        cola1.imprimir();
        System.out.println("Extraemos uno de la cola:"+cola1.extraer());
        cola1.imprimir();
    }
}
```

Cuando ejecutemos este será el resultado:

```
Listado de todos los elementos de la cola.
5-10-50-
Extraemos uno de la cola:5
Listado de todos los elementos de la cola.
10-50-
```

Problema:

Partiendo del siguiente formulario vamos a realizar la simulación de colas como hicimos en un capítulo anterior, este tiene que ser el formulario:



El código de la clase Cola.

```
public class Cola {

    class Nodo {
        int info;
        Nodo sig;
    }

    private Nodo raiz,fondo;

    Cola() {
        raiz=null;
        fondo=null;
    }
```

```
    boolean vacia (){
        if (raiz == null)
            return true;
        else
            return false;
    }

    void insertar (int info)
    {
        Nodo nuevo;
        nuevo = new Nodo ();
        nuevo.info = info;
        nuevo.sig = null;
        if (vacia ()) {
            raiz = nuevo;
            fondo = nuevo;
        } else {
            fondo.sig = nuevo;
            fondo = nuevo;
        }
    }

    int extraer ()
    {
        if (!vacia ())
        {
            int informacion = raiz.info;
            if (raiz == fondo){
                raiz = null;
                fondo = null;
            } else {
                raiz = raiz.sig;
            }
            return informacion;
        } else
            return Integer.MAX_VALUE;
    }

    public String retornar() {
             String texto="";
             if(raiz==null) {
                    return "[ ]";
             }else {
                    Nodo reco=raiz;
                    while(reco!=null) {
                           texto=texto+String.valueOf(reco.info)+"-";
                           reco=reco.sig;
                    }
                    return texto;
             }
       }
}
```

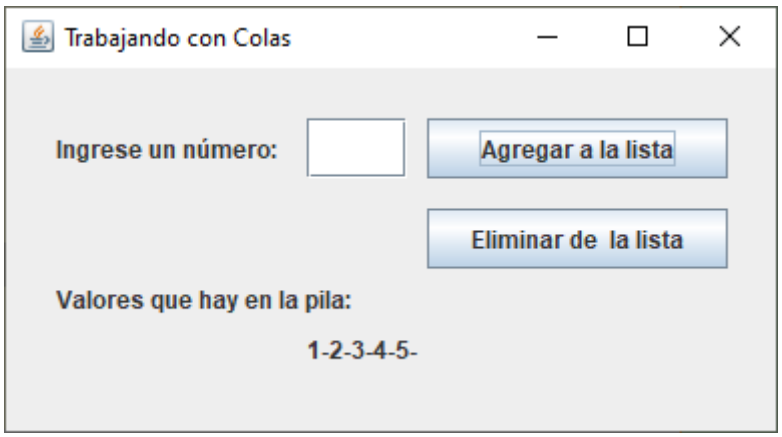
Este es el código de la clase Formulario:

```
import javax.swing.JFrame;
import javax.swing.JLabel;
```

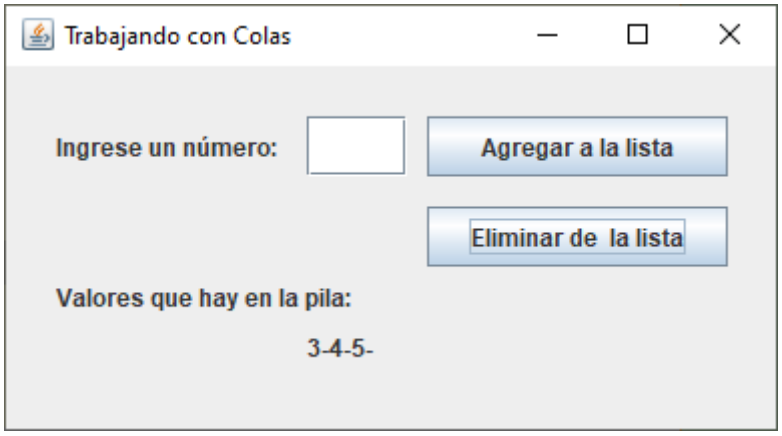
```
import javax.swing.JTextField;
import javax.swing.JButton;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Formulario extends JFrame implements ActionListener{
	JLabel label1, label2, label3;
	JTextField numero;
	JButton boton1, boton2;
	Cola cola1;

	public Formulario() {
		setLayout(null);
		cola1=new Cola();
		label1=new JLabel("Ingrese un número:");
		label1.setBounds(25,25,200,30);
		add(label1);
		numero=new JTextField();
		numero.setBounds(150,25,50,30);
		add(numero);
		boton1=new JButton("Agregar a la lista");
		boton1.setBounds(210,25,150,30);
		add(boton1);
		boton2=new JButton("Eliminar de  la lista");
		boton2.setBounds(210,70,150,30);
		add(boton2);
		label2=new JLabel("Valores que hay en la pila:");
		label2.setBounds(25, 100, 200,30);
		add(label2);
		label3=new JLabel("");
		label3.setBounds(150, 125, 200, 30);
		add(label3);
		boton1.addActionListener(this);
		boton2.addActionListener(this);
	}


	public static void main(String[] args) {
		Formulario f=new Formulario();
		f.setBounds(100,100, 400,220);
		f.setTitle("Trabajando con Colas");
		f.setVisible(true);
		f.setDefaultCloseOperation(EXIT_ON_CLOSE);
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		if(e.getSource()==boton1) {

			int num=Integer.parseInt(numero.getText());
			cola1.insertar(num);
			label3.setText(cola1.retornar());
			numero.setText("");
		}
		if(e.getSource()==boton2) {
			cola1.extraer();
			label3.setText(cola1.retornar());
		}
```

```
        }
}
```

Este será el resultado de añadir los elementos 1, 2, 3, 4 y 5.



Ahora este será el resultado después de eliminar dos elementos.



Capítulo 145.- Estructuras dinámicas: Lista tipo Cola – Problema de aplicación – 1

Desarrollar un programa para la simulación de un cajero automático.

Se cuenta con la siguiente información:

- Llegan clientes a la puerta del cajero cada 2 ó 3 minutos.
- Cada cliente tarda entre 2 y 4 minutos en ser atendido.

Obtener la siguiente información:

1- Cantidad de clientes que se atienden en 10 horas.
2- Cantidad de clientes que hay en cola después de 10 horas.
3- Hora de llegada del primer cliente que no es atendido luego de 10 horas (es decir la persona que esta primera en la cola cuando se cumplen 10 horas)

Código de la clase Cola:

```
public class Cola {

    class Nodo {
        int info;
        Nodo sig;
    }

    Nodo raiz,fondo;

    Cola() {
        raiz=null;
        fondo=null;
    }

    boolean vacia (){
        if (raiz == null)
            return true;
        else
            return false;
    }

    void insertar (int info)
    {
        Nodo nuevo;
        nuevo = new Nodo ();
        nuevo.info = info;
        nuevo.sig = null;
        if (vacia ()) {
            raiz = nuevo;
            fondo = nuevo;
        } else {
            fondo.sig = nuevo;
            fondo = nuevo;
        }
    }

    int extraer ()
    {
        if (!vacia ())
```

```
        {
            int informacion = raiz.info;
            if (raiz == fondo){
                raiz = null;
                fondo = null;
            } else {
                raiz = raiz.sig;
            }
            return informacion;
        } else
            return Integer.MAX_VALUE;
    }

    public void imprimir() {
        Nodo reco=raiz;
        System.out.println("Listado de todos los elementos de la cola.");
        while (reco!=null) {
            System.out.print(reco.info+"-");
            reco=reco.sig;
        }
        System.out.println();
    }

    public int cantidad() {
        int cant=0;
        Nodo reco=raiz;
        while (reco!=null) {
            cant++;
            reco=reco.sig;
        }
        return cant;
    }
}
```

Código de la clase Cajero (Es el ejecutable):

```
import javax.swing.*;
import java.awt.event.*;
public class Cajero extends JFrame implements ActionListener{
    private JLabel l1,l2,l3;
    private JButton boton1;
    public Cajero() {
        setLayout(null);
        boton1=new JButton("Activar Simulación");
        boton1.setBounds(10,10,180,30);
        add(boton1);
        boton1.addActionListener(this);
        l1=new JLabel("Atendidos:");
        l1.setBounds(10,50,300,30);
        add(l1);
        l2=new JLabel("En cola:");
        l2.setBounds(10,90,300,30);
        add(l2);
        l3=new JLabel("Minuto de llegada:");
        l3.setBounds(10,130,400,30);
        add(l3);
    }
```

```
    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==boton1) {
            simulacion();
        }
    }

    public void simulacion () {
        int estado = 0;
        int llegada = 2 + (int) (Math.random () * 2);
        int salida = -1;
        int cantAtendidas = 0;
        Cola cola = new Cola ();
        for (int minuto = 0 ; minuto < 600 ; minuto++) {
            if (llegada == minuto)
            {
                if (estado==0) {
                    estado=1;
                    salida=minuto+2+(int)(Math.random()*3);
                } else {
                    cola.insertar(minuto);
                }
                llegada=minuto+2+(int)(Math.random()*2);
            }
            if (salida == minuto)
            {
                estado=0;
                cantAtendidas++;
                if (!cola.vacia()) {
                    cola.extraer();
                    estado=1;
                    salida=minuto+2+(int)(Math.random()*3);
                }
            }
        }
        l1.setText("Atendidos: "+String.valueOf(cantAtendidas));
        l2.setText("En cola "+String.valueOf(cola.cantidad()));
        l3.setText("Minuto llegada: "+String.valueOf(cola.extraer()));
    }

    public static void main(String[] ar) {
        Cajero cajero1=new Cajero();
        cajero1.setBounds(0,0,340,250);
        cajero1.setVisible(true);
        cajero1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Planteo del problema:

Este práctico tiene por objeto mostrar la importancia de las colas en las Ciencias de la informática y más precisamente en las simulaciones.

Las simulaciones permiten analizar situaciones de la realidad sin la necesidad de ejecutarlas realmente. Tiene el beneficio que su costo es muy inferior a hacer pruebas en la realidad.

Desarrollar un programa para la simulación de un cajero automático se cuenta con la siguiente información:

- Llegan clientes a la puerta del cajero cada 2 o 3 minutos.
- Cada cliente tarda entre 2 y 4 minutos para ser atendido.

Obtener la siguiente información:

1- Cantidad de clientes que se atienden en 10 horas.
2- Cantidad de clientes que hay en cola después de 10 horas.
3- Hora de llegada del primer cliente que no es atendido luego de 10 horas (es decir la persona que está primera en la cola cuando se cumplen 10 horas).

La clase Cola colabora con la clase Cajero. En la clase cola debemos definir como mínimo los métodos de insertar, extraer, vacía y cantidad.

La clase Cajero define tres objetos de la clase JLabel para mostrar los resultados de la simulación.

El método más importante es el de simulación, veamos las distintas partes de dicho método:

```
int estado = 0;
int llegada = 2 + (int) (Math.random () * 2);
int salida = -1;
int cantAtendidas = 0;
Cola cola = new Cola ();
```

La variable estado almacena un cero si el cajero está libre y un uno cuando está ocupado.

La variable llegada almacena en que minuto llegará el próximo cliente (debemos generar un valor entre 2 y 3).

La variable salida almacenará en que minuto terminará el cliente de ser atendido (como en principio el cajero está vacío inicializaremos esta variable con -1.

Llevamos un contador para saber la cantidad de personas atendidas (cantAtendidas).

Luego definimos un objeto de tipo Cola para poder almacenar las personas que llegan al cajero y se lo encuentran ocupado.

Disponemos un for que se repita 600 veces (600 minutos o lo que es lo mismo 10 horas).

Dentro del for hay dos if fundamentales que verifican que sucede cuando llega una persona o cuando una persona se retira:

```
if (llegada == minuto)
{
   .....
}
if (salida == minuto)
{
   .....
}
```

Cuando llega una persona al cajero primero verifica si el cajero está desocupado:

```
if (llegada == minuto)
{
      if (estado==0) {
```

Si está desocupado lo ocupamos cambiando el valor de la variable estado y generando en que minuto esta persona dejará el cajero (un valor aleatorio entre 2 y 4 minutos):

```
estado=1;
salida=minuto+2+(int)(Math.random()*3);
```

Si el cajero está ocupado procedemos a cargar dicha persona en la cola (insertamos el minuto que llega):

```
} else {
   cola.insertar(minuto);
 }
```

Luego generamos el próximo minuto que llegará otra persona:

```
llegada=minuto+2+(int)(Math.random()*2);
```

El otro if importante es ver que sucede cuando sale la persona del cajero:

```
if (salida == minuto) {
```

Si sale una persona del cajero cambiamos el valor de la variable estado, incrementamos en una el contador cantAtendidos y si la cola no está vacía extraemos una persona, cambiamos a uno la variable estado y generamos en que minuto dejará esta persona el cajero:

```
estado=0;
cantAtendidas++;
if (!cola.vacia()) {
   cola.extraer();
   estado=1;
   salida=minuto+2+(int)(Math.random()*3);
}
}
```

Fuera del for Actualizamos las tres JLabel:

```
l1.setText("Atendidos:"+String.valueOf(cantAtendidas));
l2.setText("En cola"+String.valueOf(cola.cantidad()));
l3.setText("Minuto llegada:"+String.valueOf(cola.extraer()));
```

Capítulo 146.- Estructuras dinámicas: Lista tipo Cola – Problema de aplicación – 2

Un supermercado tiene tres cajas para la atención de los clientes.

Las cajeras tardan entre 7 y 11 minutos para la atención de cada cliente.

Los clientes llegan a la zona de cajas cada 2 ó 3 minutos. (Cuando el cliente llega, si todas las cajas tienen 6 personas, el cliente se marcha se marcha del supermercado).

Cuando el cliente llega a la zona de cajas elige la caja con una cola menor.

Realizar una simulación durante 8 horas y obtener la siguiente información:

a- Cantidad de clientes atendidos por cada caja.
b- Cantidad de clientes que se marcharon sin hacer la compras.
c- Tiempo promedio en cola.

Código de la clase Cola:

```
public class Cola {

   class Nodo {
      int info;
      Nodo sig;
   }

   Nodo raiz,fondo;

   Cola() {
      raiz=null;
      fondo=null;
   }

   boolean vacia (){
      if (raiz == null)
         return true;
      else
         return false;
   }

   void insertar (int info)
   {
      Nodo nuevo;
      nuevo = new Nodo ();
      nuevo.info = info;
      nuevo.sig = null;
      if (vacia ()) {
         raiz = nuevo;
         fondo = nuevo;
      } else {
```

```
            fondo.sig = nuevo;
            fondo = nuevo;
        }
    }

    int extraer ()
    {
        if (!vacia ())
        {
            int informacion = raiz.info;
            if (raiz == fondo){
                raiz = null;
                fondo = null;
            } else {
                raiz = raiz.sig;
            }
            return informacion;
        } else
            return Integer.MAX_VALUE;
    }

    public void imprimir() {
        Nodo reco=raiz;
        System.out.println("Listado de todos los elementos de la cola.");
        while (reco!=null) {
            System.out.print(reco.info+"-");
            reco=reco.sig;
        }
        System.out.println();
    }

    public int cantidad() {
        int cant=0;
        Nodo reco=raiz;
        while (reco!=null) {
            cant++;
            reco=reco.sig;
        }
        return cant;
    }
}
```

Código de la clase Supermercado:

```
import javax.swing.*;

import java.awt.event.*;
public class Supermercado extends JFrame implements ActionListener {
   private JButton boton1;
   private JLabel l1,l2,l3;
   public Supermercado() {
      setLayout(null);
      boton1=new JButton("Activar Simulación");
      boton1.setBounds(10,10,180,30);
      add(boton1);
      boton1.addActionListener(this);
      l1=new JLabel("Clientes atendidos por caja:");
      l1.setBounds(10,50,400,30);
      add(l1);
      l2=new JLabel("Se marchan sin hacer compras:");
      l2.setBounds(10,90,400,30);
      add(l2);
      l3=new JLabel("Tiempo promedio en cola:");
      l3.setBounds(10,130,400,30);
      add(l3);
   }
   public void actionPerformed(ActionEvent e) {
      if (e.getSource()==boton1) {
         simulacion();
      }
   }

   public void simulacion () {
      int estado1=0,estado2=0,estado3=0;
      int marchan=0;
      int llegada = 2 + (int) (Math.random () * 2);
      int salida1=-1,salida2=-1,salida3=-1;
      int cantAte1=0,cantAte2=0,cantAte3=0;
      int tiempoEnCola=0;
      int cantidadEnCola=0;
      Cola cola1 = new Cola ();
      Cola cola2 = new Cola ();
      Cola cola3 = new Cola ();
      for (int minuto = 0 ; minuto < 600 ; minuto++) {
         if (llegada == minuto) {
            if (estado1==0) {
               estado1=1;
               salida1=minuto+7+(int)(Math.random()*5);
            } else {
               if (estado2==0) {
```

```
                estado2=1;
                salida2=minuto+7+(int)(Math.random()*5);
              } else {
                if (estado3==0) {
                  estado3=1;
                  salida3=minuto+7+(int)(Math.random()*5);
                } else {
                  if (cola1.cantidad()==6 && cola2.cantidad()==6 && cola3.cantidad()==6) {
                    marchan++;
                  } else {
                    if (cola1.cantidad()<=cola2.cantidad() && cola1.cantidad()<=cola3.cantidad())
{
                      cola1.insertar(minuto);
                    } else {
                      if (cola2.cantidad()<=cola3.cantidad()) {
                        cola2.insertar(minuto);
                      } else {
                        cola3.insertar(minuto);
                      }
                    }
                  }
                }
              }

            }
            llegada=minuto+ 2+ (int) (Math.random () * 2);
          }
          if (salida1 == minuto) {
            cantAte1++;
            estado1=0;
            if(!cola1.vacia()) {
              estado1=1;
              int m=cola1.extraer();
              salida1=minuto+7+(int)(Math.random()*5);
              tiempoEnCola=tiempoEnCola+(minuto-m);
              cantidadEnCola++;
            }
          }
          if (salida2 == minuto) {
            cantAte2++;
            estado2=0;
            if(!cola2.vacia()) {
              estado2=1;
              int m=cola2.extraer();
              salida2=minuto+7+(int)(Math.random()*5);
              tiempoEnCola=tiempoEnCola+(minuto-m);
              cantidadEnCola++;
            }
```

```
        }
        if (salida3 == minuto) {
          cantAte3++;
          estado3=0;
          if(!cola3.vacia()) {
            estado3=1;
            int m=cola3.extraer();
            salida3=minuto+7+(int)(Math.random()*5);
            tiempoEnCola=tiempoEnCola+(minuto-m);
            cantidadEnCola++;
          }
        }
      }
      l1.setText("Clientes atendidos por caja: caja1="+cantAte1+"  caja2="+cantAte2+"
caja3="+cantAte3);
      l2.setText("Se marchan sin hacer compras:"+marchan);
      if (cantidadEnCola>0) {
        int tiempoPromedio=tiempoEnCola/cantidadEnCola;
        l3.setText("Tiempo promedio en cola:"+tiempoPromedio);
      }
    }

    public static void main(String[] ar) {
      Supermercado super1=new Supermercado();
      super1.setBounds(0,0,390,250);
      super1.setVisible(true);
      super1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Problema propuesto:

Queremos hacer una comparativa trabajando con Pilas y Colas y observaremos el resultado en los dos caso, en primer lugar no encontramos un una máquina que entre 2 y 3 minutos deposita en la pila un plato que será recogida por otra máquina para su lavado entre 3 y 4 minutos.

Durante una jornada laboral de 8 horas queremos saber cuantos platos se quedan pendientes de lavar y el número del minuto de dichos platos de cuando fueron depositados en la pila.

Código de clase Pila.

```
public class Pila {

	class Nodo{
		int minuto;
		Nodo sig;
	}

	private Nodo raiz;

	public Pila(){
		raiz=null;
	}

	public void insertar(int min) {
		Nodo nuevo;
		nuevo = new Nodo();
		nuevo.minuto=min;
		if(raiz==null) {
			nuevo.sig=null;
			raiz=nuevo;
		}else {
			nuevo.sig=raiz;
			raiz=nuevo;
		}
	}

	public int extraer() {
		if(raiz!=null) {
			int min=raiz.minuto;
			raiz=raiz.sig;
			return min;
		}else {
			return Integer.MAX_VALUE;
		}
	}

	public boolean vacia() {
		if(raiz==null) {
			return true;
		}else {
			return false;
		}
	}

	public String imprimir() {
		String platos="";
		Nodo reco=raiz;
		int cont=0;
```

```
            while(reco!=null) {
                  platos=platos+(String.valueOf(reco.minuto)+" - ");
                  reco=reco.sig;
                  cont++;
                  if (cont==10) {
                        platos=platos+"\n";
                        cont=0;
                  }
            }
            return platos;
      }

      public int cantidad() {
            int cant=0;
            Nodo reco=raiz;
            while(reco!=null) {
                  reco=reco.sig;
                  cant++;
            }
            return cant;
      }
}
```

A continuación el código de la clase Platos, esta clase es la que a continuación ejecutaremos ya que contiene el método main.

```
1  public class Platos {
2
3      public void simulacion() {
4          int estado=0;
5          int entrada=2+(int)(Math.random()*2);
6          int salida=-1;
7          Pila pila1=new Pila();
8          for(int minuto=0; minuto<480; minuto++) {
9              if(entrada==minuto) {
10                 if(estado==0) {
11                     estado=1;
12                     salida=minuto+3+(int)(Math.random()*2);
13                 }else {
14                     pila1.insertar(minuto);
15                 }
16                 entrada=minuto+2+(int)(Math.random()*2);
17             }
18             if(salida==minuto) {
19                 estado=0;
20                 if(!pila1.vacia()) {
21                     estado=1;
22                     pila1.extraer();}
23                     salida=minuto+3+(int)(Math.random()*2);
24                 }
25             }
26
27
28         System.out.println("Cantidad de platos en la pila "+pila1.cantidad());
29         System.out.println("Número de platos: \n"+ pila1.imprimir());
30     }
31
32
```

```
33      public static void main(String[] args) {
34          Platos plato1=new Platos();
35          plato1.simulacion();
36      }
37  }
```

Cuando ejecutemos este será el resultado:

```
Cantidad de platos en la pila 57
Número de platos:
476 - 469 - 462 - 449 - 445 - 441 - 437 - 429 - 420 - 411 -
406 - 396 - 392 - 382 - 375 - 358 - 351 - 347 - 337 - 328 -
321 - 317 - 303 - 294 - 290 - 286 - 276 - 270 - 262 - 255 -
238 - 225 - 217 - 210 - 206 - 196 - 186 - 182 - 172 - 168 -
161 - 151 - 145 - 126 - 121 - 114 - 107 - 99 - 91 - 87 -
77 - 66 - 59 - 49 - 38 - 27 - 14 -
```

La cantidad de platos que quedan en la pila son de 57 y con sus respectivos minutos de cuando fueron depositados en la Pila.

Ahora vamos a crear la clase Cola para realizar el mismo ejemplo que la clase pila pero en este caso las botellas las depositamos en una Cinta ya que no se pueden apilar.

Código de la clase Cola:

```
public class Cola {

       class Nodo{
              int minuto;
              Nodo sig;
       }

       private Nodo raiz, fondo;

       public Cola(){
              raiz=null;
              fondo=null;
       }

    boolean vacia (){
        if (raiz == null)
            return true;
        else
            return false;
    }

       public void insertar(int min) {
              Nodo nuevo;
              nuevo = new Nodo();
              nuevo.minuto=min;
              nuevo.sig=null;
              if(vacia()) {
                     raiz=nuevo;
                     fondo=nuevo;
              }else {
                     fondo.sig=nuevo;
                     fondo=nuevo;
              }
       }
```

```
	public int extraer() {
		if(!vacia()) {
			int min=raiz.minuto;
			if (raiz==fondo) {
				raiz=null;
				fondo=null;
			}else {
				raiz=raiz.sig;
			}
			return min;
		}else
			return Integer.MAX_VALUE;
		}


	public String imprimir() {
		String botellas="";
		Nodo reco=raiz;
		int cont=0;
		while(reco!=null) {
			botellas=botellas+(String.valueOf(reco.minuto)+" - ");
			reco=reco.sig;
			cont++;
			if (cont==10) {
				botellas=botellas+"\n";
				cont=0;
			}
		}
		return botellas;
	}

	public int cantidad() {
		int cant=0;
		Nodo reco=raiz;
		while(reco!=null) {
			reco=reco.sig;
			cant++;
		}
		return cant;
	}

}
```

A continuación vamos a realizar la clase Botella, esta contiene el método main, por este motivo será la clase que a continuación ejecutaremos:

```
1  public class Botellas {
2
3      public void simulacion() {
4          int estado=0;
5          int entrada=2+(int)(Math.random()*2);
6          int salida=-1;
7          Cola cola1=new Cola();
8          for(int minuto=0; minuto<480; minuto++) {
9              if(entrada==minuto) {
10                 if(estado==0) {
11                     estado=1;
12                     salida=minuto+3+(int)(Math.random()*2);
13                 }else {
```

```
                    cola1.insertar(minuto);
                }
                entrada=minuto+2+(int)(Math.random()*2);
            }
            if(salida==minuto) {
                estado=0;
                if(!cola1.vacia()) {
                    cola1.extraer();
                    estado=1;
                    salida=minuto+3+(int)(Math.random()*2);
                }
            }
        }
        System.out.println("Cantidad de botellas en la cinta "+cola1.cantidad());
        System.out.println("Número de botellas: \n"+ cola1.imprimir());
    }


    public static void main(String[] args) {
        Botellas botella1=new Botellas();
        botella1.simulacion();
    }
}
```

cuando ejecutemos este será el resultado:

```
Cantidad de botellas en la cinta 53
Número de botellas:
350 - 353 - 356 - 358 - 360 - 362 - 364 - 366 - 369 - 372 -
374 - 376 - 379 - 381 - 383 - 386 - 388 - 390 - 393 - 395 -
397 - 399 - 402 - 404 - 407 - 410 - 413 - 415 - 418 - 421 -
423 - 426 - 428 - 431 - 433 - 435 - 438 - 440 - 443 - 445 -
448 - 451 - 454 - 457 - 460 - 462 - 464 - 467 - 470 - 472 -
475 - 477 - 479 -
```

Podemos observar que el número de botellas que se quedan en la cinta es bastante similar, pero las botellas por el minuto que fueron depositadas son distintas, esto es debido a que el sistema de carga de una Pila es distinta a la de la Cola.

Capítulo 147.- Estructura dinámica: Listas genéricas – 1

```
2  public class ListasGenericas {
3      class Nodo{
4          int info;
5          Nodo sig;
6      }
7      private Nodo raiz;
8
```

El método vacía debe retornar true si está vacía y false si no lo está.

boolen vacia();

```
9      public boolean vacia() {
10         if (raiz==null) {
11             return true;
12         }else{
13             return false;
14         }
15     }
```

El método main:

```
public static void main(String[] args) {
    ListasGenericas lg=new ListasGenericas();
    if (lg.vacia()) {
        System.out.println("Lista vacia");
    }
}
```

Si ejecutamos este será el resultado:

```
Lista vacia
```

Retorna la cantidad de nodos de la lista.

int cantidad()

```
17     public int cantidad() {
18         int cant=0;
19         Nodo reco=raiz;
20         while (reco!=null) {
21             cant++;
22             reco=reco.sig;
23         }
24         return cant;
25     }
```

Insertar un nodo en la posición (pos) y con la información que hay en el parámetro x.

void insertar(int pos, int x)

```
27     public void insertar(int pos, int x) {
28         if(pos<=cantidad()+1) {
29             Nodo nuevo=new Nodo();
30             nuevo.info=x;
31             if(pos==1) {
32                 nuevo.sig=raiz;
33                 raiz=nuevo;
```

```
            }else {
                if(pos==cantidad()+1) {
                    Nodo reco=raiz;
                    while(reco.sig!=null) {
                        reco=reco.sig;
                    }
                    reco.sig=nuevo;
                }else {
                    Nodo reco=raiz;
                    for(int f=1; f<=pos-2; f++) {
                        reco=reco.sig;
                    }
                    Nodo siguiente=reco.sig;
                    reco.sig=nuevo;
                    nuevo.sig=siguiente;
                }
            }
        }
    }
```

Modificamos el método main:

```
    public static void main(String[] args) {
        ListasGenericas lg=new ListasGenericas();
        if (lg.vacia()) {
            System.out.println("Lista vacia");
            lg.insertar(1, 100);
        }
    }
```

Ahora vamos a realizar el método imprimir:

```
    public void imprimir() {
        Nodo reco=raiz;
        while(reco!=null) {
            System.out.print(reco.info+"-");
            reco=reco.sig;
        }
        System.out.println();
    }
```

Modificamos el método main:

```
    public static void main(String[] args) {
        ListasGenericas lg=new ListasGenericas();
        if (lg.vacia()) {
            System.out.println("Lista vacia");
            lg.insertar(1, 100);
            lg.insertar(2, 200);
            lg.insertar(3, 300);
            lg.insertar(2,50);
            lg.imprimir();
        }
    }
```

Ejecutamos:

```
Lista vacia
100-50-200-300-
```

Extrae la información del nodo de la posición indicada (pos). Se debe eliminar el nodo.

int extraer(int pos)

```
54      public int extraer(int pos) {
55          if (pos>cantidad()){
56              return Integer.MAX_VALUE;
57          }else {
58              if(pos==1) {
59                  int informacion=raiz.info;
60                  raiz=raiz.sig;
61                  return informacion;
62              }else {
63                  Nodo reco=raiz;
64                  for(int f=1; f<pos; f++) {
65                      reco=reco.sig;
66                  }
67                  Nodo prox=reco.sig;
68                  int informacion=prox.info;
69                  reco.sig=prox.sig;
70                  return informacion;
71              }
72          }
73      }
```

Modificamos el código del método main:

```
86      public static void main(String[] args) {
87          ListasGenericas lg=new ListasGenericas();
88          if (lg.vacia()) {
89              System.out.println("Lista vacia");
90              lg.insertar(1, 100);
91              lg.insertar(2, 200);
92              lg.insertar(3, 300);
93              lg.insertar(2,50);
94              lg.insertar(1, 500);
95              lg.imprimir();
96              System.out.println("Extraemos el primero de la lista: "+lg.extraer(1));
97              lg.imprimir();
98          }
99      }
```

Podemos consultar por los distintos elementos de la lista modificando el número de su posición, el 2 para el valor 100, el 3 para el valor 50 y así sucesivamente.

Ejecutamos:

```
Lista vacia
500-100-50-200-300-
Extraemos el primero de la lista: 500
100-50-200-300-
```

Borrar el nodo de la posición (pos).

void borrar (int pos)

```
75      public void borrar(int pos) {
76          if (pos>cantidad())
77              return;
78          else {
79              if(pos==1) {
80                  raiz=raiz.sig;
81              }else {
82                  Nodo reco=raiz;
83                  for(int f=1; f<pos-2; f++) {
84                      reco=reco.sig;
```

```
                }
                Nodo prox=reco.sig;
                reco.sig=prox.sig;
            }
        }
    }
```

```
    public static void main(String[] args) {
        ListasGenericas lg=new ListasGenericas();
        if (lg.vacia()) {
            System.out.println("Lista vacia");
            lg.insertar(1, 100);
            lg.insertar(2, 200);
            lg.insertar(3, 300);
            lg.insertar(2,50);
            lg.insertar(1, 500);
            lg.imprimir();
            System.out.println("Extraemos el primero de la lista: "+lg.extraer(1));
            lg.imprimir();lg.borrar(1);
            System.out.println("Después de borrar el primero de la lista");
            lg.imprimir();
        }
    }
```

Ejecutamos:

```
Lista vacia
500-100-50-200-300-
Extraemos el primero de la lista: 500
100-50-200-300-
Después de borrar el primero de la lista
50-200-300-
```

Intercambiar las informaciones de los nodos de las posiciones pos1 y pos2.

void intercambiar(int pos1, int pos2)

```
    public void intercambiar(int pos1, int pos2) {
        if(pos1<=cantidad() && pos2>=cantidad()) {
            Nodo reco1=raiz;
            for(int f=1; f<pos1; f++) {
                reco1=reco1.sig;
            }
            Nodo reco2=raiz;
            for(int f=1; f<pos2; f++) {
                reco2=reco2.sig;
            }
            int aux=reco1.info;
            reco1.info=reco2.info;
            reco2.info=aux;
        }
    }
```

Vamos a modificar el método main:

```
    public static void main(String[] args) {
        ListasGenericas lg=new ListasGenericas();
        if (lg.vacia()) {
            System.out.println("Lista vacia");
            lg.insertar(1, 100);
            lg.insertar(2, 200);
            lg.insertar(3, 300);
            lg.insertar(2,50);
            lg.insertar(1, 500);
```

```
            lg.imprimir();
            System.out.println("Extraemos el primero de la lista: "+lg.extraer(1));
            lg.imprimir();lg.borrar(1);
            System.out.println("Después de borrar el primero de la lista");
            lg.imprimir();
            lg.intercambiar(1, 3);
            lg.imprimir();
        }
    }
```

Vamos a ejecutar:

```
Lista vacia
500-100-50-200-300-
Extraemos el primero de la lista: 500
100-50-200-300-
Después de borrar el primero de la lista
50-200-300-
300-200-50-
```

Retornar el nodo con mayor información

int posMayor()

```
    public int mayor() {
        if(vacia()) {
            return Integer.MAX_VALUE;
        }else {
            int may=raiz.info;
            Nodo reco=raiz.sig;
            while(reco!=null) {
                if(reco.info>may) {
                    may=reco.info;
                }
                reco=reco.sig;
            }
            return may;
        }
    }
```

Modificamos el método main:

```
    public static void main(String[] args) {
        ListasGenericas lg=new ListasGenericas();
        if (lg.vacia()) {
            System.out.println("Lista vacia");
            lg.insertar(1, 100);
            lg.insertar(2, 200);
            lg.insertar(3, 300);
            lg.insertar(2,50);
            lg.insertar(1, 500);
            lg.imprimir();
            System.out.println("Extraemos el primero de la lista: "+lg.extraer(1));
            lg.imprimir();lg.borrar(1);
            System.out.println("Después de borrar el primero de la lista");
            lg.imprimir();
            lg.intercambiar(1, 3);
            lg.imprimir();
            System.out.println("El mayor de la lista "+lg.mayor());
        }
    }
```

Ejecutamos:

```
Lista vacia
500-100-50-200-300-
Extraemos el primero de la lista: 500
100-50-200-300-
Después de borrar el primero de la lista
50-200-300-
300-200-50-
El mayor de la lista 300
```

Retorna la posición del nodo con mayor información.

int posMayor()

```
124     public int posMayor() {
125         if(vacia()) {
126             return Integer.MAX_VALUE;
127         }else {
128             int may=raiz.info;
129             Nodo reco=raiz.sig;
130             int pos=1;
131             int c=1;
132             while(reco!=null) {
133                 c++;
134                 if (reco.info>may) {
135                     may=reco.info;
136                     pos=c;
137                 }
138                 reco=reco.sig;
139             }
140             return pos;
141         }
142     }
```

Modificamos el método main:

```
155     public static void main(String[] args) {
156         ListasGenericas lg=new ListasGenericas();
157         if (lg.vacia()) {
158             System.out.println("Lista vacia");
159             lg.insertar(1, 100);
160             lg.insertar(2, 200);
161             lg.insertar(3, 300);
162             lg.insertar(2,50);
163             lg.insertar(1, 500);
164             lg.imprimir();
165             System.out.println("Extraemos el primero de la lista: "+lg.extraer(1));
166             lg.imprimir();lg.borrar(1);
167             System.out.println("Después de borrar el primero de la lista");
168             lg.imprimir();
169             lg.intercambiar(1, 3);
170             lg.imprimir();
171             System.out.println("El mayor de la lista "+lg.mayor());
172             System.out.println("La posición con el valor mayore es "+lg.posMayor());
173         }
174     }
```

Ejecutamos:

```
Lista vacia
500-100-50-200-300-
Extraemos el primero de la lista: 500
100-50-200-300-
Después de borrar el primero de la lista
50-200-300-
```

```
300-200-50-
El mayor de la lista 300
La posición con el valor mayore es 1
```

Debe retornar true si la lista está ordenada de menor a mayor, false en caso contrario.

boolean ordenada()

```
144    public boolean ordenada() {
145        if(cantidad()>1) {
146            Nodo reco1=raiz;
147            Nodo reco2=raiz.sig;
148            while(reco2!=null) {
149                if(reco2.info<reco1.info) {
150                    return false;
151                }
152                reco1=reco1.sig;
153                reco2=reco2.sig;
154            }
155        }
156        return true;
157    }
```

Modificamos el método main:

```
170    public static void main(String[] args) {
171        ListasGenericas lg=new ListasGenericas();
172        if (lg.vacia()) {
173            System.out.println("Lista vacia");
174            lg.insertar(1, 100);
175            lg.insertar(2, 200);
176            lg.insertar(3, 300);
177            lg.insertar(2,50);
178            lg.insertar(1, 500);
179            lg.imprimir();
180            System.out.println("Extraemos el primero de la lista: "+lg.extraer(1));
181            lg.imprimir();lg.borrar(1);
182            System.out.println("Después de borrar el primero de la lista");
183            lg.imprimir();
184            lg.intercambiar(1, 3);
185            lg.imprimir();
186            System.out.println("El mayor de la lista "+lg.mayor());
187            System.out.println("La posición con el valor mayore es "+lg.posMayor());
188            if (lg.ordenada()) {
189                System.out.println("La lista está ordenada");
190            }else {
191                System.out.println("La lista no está ordenada");
192            }
193        }
194    }
```

Ejecutamos:

```
Lista vacia
500-100-50-200-300-
Extraemos el primero de la lista: 500
100-50-200-300-
Después de borrar el primero de la lista
50-200-300-
300-200-50-
El mayor de la lista 300
La posición con el valor mayore es 1
La lista no está ordenada
```

Debe retornar true si existe la información que lleva en el parámetro, false en caso contrario.

boolean existe(int info)

```
159     public boolean existe(int info) {
160         Nodo reco=raiz;
161         while(reco!=null) {
162             if(reco.info==info) {
163                 return true;
164             }
165             reco=reco.sig;
166         }
167         return false;
168     }
```

Modificamos el método main:

```
181     public static void main(String[] args) {
182         ListasGenericas lg=new ListasGenericas();
183         if (lg.vacia()) {
184             System.out.println("Lista vacia");
185             lg.insertar(1, 100);
186             lg.insertar(2, 200);
187             lg.insertar(3, 300);
188             lg.insertar(2,50);
189             lg.insertar(1, 500);
190             lg.imprimir();
191             System.out.println("Extraemos el primero de la lista: "+lg.extraer(1));
192             lg.imprimir();lg.borrar(1);
193             System.out.println("Después de borrar el primero de la lista");
194             lg.imprimir();
195             lg.intercambiar(1, 3);
196             lg.imprimir();
197             System.out.println("El mayor de la lista "+lg.mayor());
198             System.out.println("La posición con el valor mayore es "+lg.posMayor());
199             if (lg.ordenada()) {
200                 System.out.println("La lista está ordenada");
201             }else {
202                 System.out.println("La lista no está ordenada");
203             }
204             if(lg.existe(300)) {
205                 System.out.println("La lista contiene el 300");
206             }else {
207                 System.out.println("La lista no contiene el 300");
208             }
209         }
210     }
```

Ejecutamos:

```
Lista vacia
500-100-50-200-300-
Extraemos el primero de la lista: 500
100-50-200-300-
Después de borrar el primero de la lista
50-200-300-
300-200-50-
El mayor de la lista 300
La posición con el valor mayore es 1
La lista no está ordenada
La lista contiene el 300
```

Código completo:

```
public class ListasGenericas {
	class Nodo{
		int info;
		Nodo sig;
	}
	private Nodo raiz;

	public boolean vacia() {
		if (raiz==null) {
			return true;
		}else{
			return false;
		}
	}

	public int cantidad() {
		int cant=0;
		Nodo reco=raiz;
		while (reco!=null) {
			cant++;
			reco=reco.sig;
		}
		return cant;
	}

	public void insertar(int pos, int x) {
		if(pos<=cantidad()+1) {
			Nodo nuevo=new Nodo();
			nuevo.info=x;
			if(pos==1) {
				nuevo.sig=raiz;
				raiz=nuevo;
			}else {
				if(pos==cantidad()+1) {
					Nodo reco=raiz;
					while(reco.sig!=null) {
						reco=reco.sig;
					}
					reco.sig=nuevo;
				}else {
					Nodo reco=raiz;
					for(int f=1; f<=pos-2; f++) {
						reco=reco.sig;
					}
					Nodo siguiente=reco.sig;
					reco.sig=nuevo;
					nuevo.sig=siguiente;
				}
			}
		}
	}

	public int extraer(int pos) {
		if (pos>cantidad()){
			return Integer.MAX_VALUE;
		}else {
			if(pos==1) {
```

```
                int informacion=raiz.info;
                raiz=raiz.sig;
                return informacion;
            }else {
                Nodo reco=raiz;
                for(int f=1; f<pos; f++) {
                    reco=reco.sig;
                }
                Nodo prox=reco.sig;
                int informacion=prox.info;
                reco.sig=prox.sig;
                return informacion;
            }
        }
    }

    public void borrar(int pos) {
        if (pos>cantidad())
            return;
        else {
            if(pos==1) {
                raiz=raiz.sig;
            }else {
                Nodo reco=raiz;
                for(int f=1; f<pos-2; f++) {
                    reco=reco.sig;
                }
                Nodo prox=reco.sig;
                reco.sig=prox.sig;
            }
        }
    }

    public void intercambiar(int pos1, int pos2) {
        if(pos1<=cantidad() && pos2>=cantidad()) {
            Nodo reco1=raiz;
            for(int f=1; f<pos1; f++) {
                reco1=reco1.sig;
            }
            Nodo reco2=raiz;
            for(int f=1; f<pos2; f++) {
                reco2=reco2.sig;
            }
            int aux=reco1.info;
            reco1.info=reco2.info;
            reco2.info=aux;
        }
    }

    public int mayor() {
        if(vacia()) {
            return Integer.MAX_VALUE;
        }else {
            int may=raiz.info;
            Nodo reco=raiz.sig;
            while(reco!=null) {
                if(reco.info>may) {
                    may=reco.info;
                }
```

```
                reco=reco.sig;
            }
            return may;
        }
    }

    public int posMayor() {
        if(vacia()) {
            return Integer.MAX_VALUE;
        }else {
            int may=raiz.info;
            Nodo reco=raiz.sig;
            int pos=1;
            int c=1;
            while(reco!=null) {
                c++;
                if (reco.info>may) {
                    may=reco.info;
                    pos=c;
                }
                reco=reco.sig;
            }
            return pos;
        }
    }

    public boolean ordenada() {
        if(cantidad()>1) {
            Nodo reco1=raiz;
            Nodo reco2=raiz.sig;
            while(reco2!=null) {
                if(reco2.info<reco1.info) {
                    return false;
                }
                reco1=reco1.sig;
                reco2=reco2.sig;
            }
        }
        return true;
    }

    public boolean existe(int info) {
        Nodo reco=raiz;
        while(reco!=null) {
            if(reco.info==info) {
                return true;
            }
            reco=reco.sig;
        }
        return false;
    }

    public void imprimir() {
        Nodo reco=raiz;
        while(reco!=null) {
            System.out.print(reco.info+"-");
            reco=reco.sig;
        }
        System.out.println();
```

```
	}


	public static void main(String[] args) {
		ListasGenericas lg=new ListasGenericas();
		if (lg.vacia()) {
			System.out.println("Lista vacia");
			lg.insertar(1, 100);
			lg.insertar(2, 200);
			lg.insertar(3, 300);
			lg.insertar(2,50);
			lg.insertar(1, 500);
			lg.imprimir();
			System.out.println("Extraemos el primero de la lista:
"+lg.extraer(1));
			lg.imprimir();lg.borrar(1);
			System.out.println("Después de borrar el primero de la
lista");
			lg.imprimir();
			lg.intercambiar(1, 3);
			lg.imprimir();
			System.out.println("El mayor de la lista "+lg.mayor());
			System.out.println("La posición con el valor mayore es
"+lg.posMayor());
			if (lg.ordenada()) {
				System.out.println("La lista está ordenada");
			}else {
				System.out.println("La lista no está ordenada");
			}
			if(lg.existe(300)) {
				System.out.println("La lista contiene el 300");
			}else {
				System.out.println("La lista no contiene el 300");
			}
		}
	}
}
```

Capítulo 148.- Estructuras dinámicas: Listas genéricas – 2

Problema propuesto:

Plantear una clase para administrar una lista genérica implementando los siguientes métodos:

a) Insertar un nodo al principio de la lista.
b) Insertar un nodo al final de la lista.
c) Insertar un nodo en la segunda posición. Si la lista está vacía no se inserta el nodo.
d) Insertar un nodo en la ante última posición.
e) Borrar el primer nodo.
f) Borrar el segundo nodo.
g) Borrar el último nodo.
h) Borrar el nodo con información mayor.

El código:

```
public class ListaGenerica {
	class Nodo{
		int info;
		Nodo sig;
	}
	private Nodo raiz;

	public void imprimir() {
		Nodo reco=raiz;
		while(reco!=null){
			System.out.print(reco.info+"-");
			reco=reco.sig;
		}
		System.out.println();
	}

	public void insertarPrincipio(int x) {
		Nodo nuevo=new Nodo();
		nuevo.info=x;
		nuevo.sig=raiz;
		raiz=nuevo;
	}
	public int cantidad() {
		int cant=0;
		Nodo reco=raiz;
		while (reco!=null) {
			cant++;
			reco=reco.sig;
		}
		return cant;
	}

	public void insertarFinal(int x) {
		Nodo nuevo = new Nodo();
		nuevo.info=x;
		if (raiz==null) {
			raiz=nuevo;
		}else {
			Nodo reco=raiz;
			while(reco.sig!=null) {
				reco=reco.sig;
```

```
                }
                reco.sig=nuevo;
        }
}

public void insertarSegunda(int x) {
        if(raiz!=null) {
                Nodo nuevo = new Nodo();
                nuevo.info=x;
                if(raiz.sig==null) {
                        raiz.sig=nuevo;
                }else {
                        Nodo segundo=raiz.sig;
                        raiz.sig=nuevo;
                        nuevo.sig=segundo;
                }
        }
}

public void insertarAnteUltima(int x) {
        if(raiz!=null) {
                Nodo nuevo=new Nodo();
                nuevo.info=x;
                if(raiz.sig==null) {
                        nuevo.sig=raiz;
                }else {
                        Nodo atras=raiz;
                        Nodo reco=raiz.sig;
                        while(reco.sig!=null) {
                                atras=reco;
                                reco=reco.sig;
                        }
                        nuevo.sig=atras.sig;
                        atras.sig=nuevo;
                }
        }
}

public void borrarPrimer() {
        if(raiz!=null) {
                raiz=raiz.sig;
        }
}

public void borrarSegundo() {
        if(raiz!=null) {
                if(raiz.sig!=null) {
                        Nodo tercero=raiz.sig.sig;
                        raiz.sig=tercero;
                }
        }
}

public void borrarUltimo() {
        if(raiz!=null) {
                if(raiz.sig==null) {
                        raiz=null;
                }else {
                        Nodo reco=raiz.sig;
```

```
                    Nodo atras=raiz;
                    while(reco.sig!=null) {
                          atras=reco;
                          reco=reco.sig;
                    }
                    atras.sig=null;
             }
       }
}

public void borrarMayor() {
       if(raiz!=null) {
             int mayor=raiz.info;
             Nodo reco=raiz;
             while (reco!=null) {
                    if(reco.info>mayor) {
                          mayor=reco.info;
                    }
                    reco=reco.sig;
             }
             reco=raiz;
             Nodo atras=raiz;
             while (reco!=null) {
                    if(reco.info==mayor) {
                          if(reco==raiz) {
                                raiz=raiz.sig;
                                atras=raiz;
                                reco=raiz;
                          }else {
                                atras.sig=reco.sig;
                                reco=reco.sig;
                          }
                    }else {
                          atras=reco;
                          reco=reco.sig;
                    }
             }
       }
}

public static void main(String[] args) {
       ListaGenerica lg=new ListaGenerica();
       lg.insertarPrincipio(10);
       lg.insertarPrincipio(5);
       lg.insertarPrincipio(2);
       lg.imprimir();
       lg.insertarFinal(100);
       lg.imprimir();
       lg.insertarSegunda(320);
       lg.imprimir();
       lg.insertarAnteUltima(7000);
       lg.imprimir();
       lg.borrarPrimer();
       lg.imprimir();
       lg.borrarSegundo();
       lg.imprimir();
       lg.borrarUltimo();
       lg.imprimir();
       lg.borrarMayor();
```

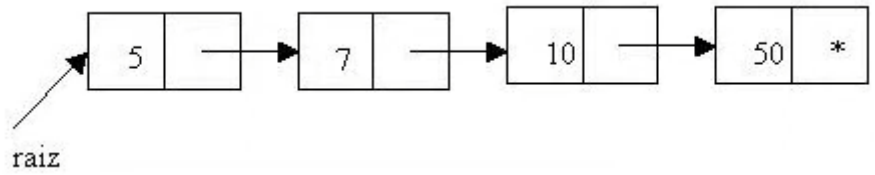
```
            lg.imprimir();
        }
}
```

Ejecutamos:

```
2-5-10-
2-5-10-100-
2-320-5-10-100-
2-320-5-10-7000-100-
320-5-10-7000-100-
320-10-7000-100-
320-10-7000-
320-10-
```

Capítulo 149.- Estructuras dinámicas: Listas genéricas ordenadas

Una lista genérica es ordenada si cuando insertamos información en la lista queda ordenada respecto al campo info. (sea de menor a mayor o a la inversa).




```
public class ListaGenericaOrdenada {
	class Nodo{
		int info;
		Nodo sig;
	}
	private Nodo raiz;

	public void insertar(int x) {
		Nodo nuevo=new Nodo();
		nuevo.info=x;
		if(raiz==null) {
			raiz=nuevo;
		}else {
			if(x<raiz.info) {
				nuevo.sig=raiz;
				raiz=nuevo;
			}else {
				Nodo reco=raiz;
				Nodo atras=raiz;
				while (x>=reco.info && reco.sig!=null) {
					atras=reco;
					reco=reco.sig;
				}
				if(x>=reco.info) {
					reco.sig=nuevo;
				}else {
					nuevo.sig=reco;
					atras.sig=nuevo;
				}
			}
		}
	}

	public void imprimir() {
		Nodo reco=raiz;
		while (reco!=null) {
			System.out.print(reco.info+"-");
			reco=reco.sig;
		}
		System.out.println();
	}

	public static void main(String[] args) {
		ListaGenericaOrdenada lgo=new ListaGenericaOrdenada();
		lgo.insertar(5);
		lgo.insertar(50);
		lgo.insertar(10);
```

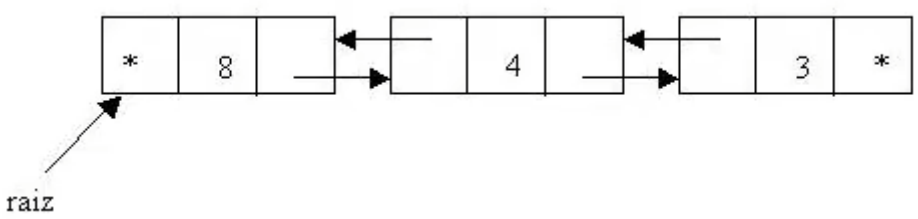
```
            lgo.insertar(7);
            lgo.imprimir();
            lgo.insertar(400);
            lgo.imprimir();
        }
}
```

Si ejecutamos este será el resultado:

```
5-7-10-50-
5-7-10-50-400-
```

Capítulo 150.- Estructuras dinámicas: Listas genéricas doblemente encadenadas – 1

A las listas vistas hasta el momento podemos recorrerlas solamente en una dirección (Listas simplemente encadenadas). Hay problemas donde se requiere recorrer la lista en ambas direcciones, en estos casos el empleo de listas doblemente encadenadas es recomendable.



Retorna la cantidad de nodos de la lista.

int cantidad()

Inserta un nodo en la posición (pos) y con la información que hay en el parámetro x.

void insertar(int posx, int x)

Extrae la información del nodo de la posición indicada (pos). Se debe eliminar el nodo.

int extraer(int pos)

```
public class ListaGenericaDoble {
	class Nodo{
		int info;
		Nodo sig, ant;
	}

	private Nodo raiz;

	public void insertarPrimero(int x) {
		Nodo nuevo=new Nodo();
		nuevo.info=x;
		if(raiz==null) {
			raiz=nuevo;
		}else {
			nuevo.sig=raiz;
			raiz.ant=nuevo;
			raiz=nuevo;
		}
	}

	public void insertarFinal(int x) {
		Nodo nuevo=new Nodo();
		nuevo.info=x;
		if(raiz==null) {
			raiz=nuevo;
		}else {
			Nodo reco=raiz;
			while(reco.sig!=null) {
				reco=reco.sig;
			}
			reco.sig=nuevo;
			nuevo.ant=reco;
```

```
        }
}

public void imprimir() {
        Nodo reco=raiz;
        while(reco!=null) {
                System.out.print(reco.info+"-");
                reco=reco.sig;
        }
        System.out.println();
}

public void imprimirInversa() {
        Nodo reco=raiz;
        while(reco.sig!=null) {
                reco=reco.sig;
        }
        while(reco!=null) {
                System.out.print(reco.info+"-");
                reco=reco.ant;
        }
        System.out.println();
}

public int cantidad() {
        Nodo reco=raiz;
        int cant=0;
        while(reco!=null) {
                cant++;
                reco=reco.sig;
        }
        return cant;
}

public void insertar(int pos, int x) {
        if(pos<=cantidad()+1) {
                Nodo nuevo=new Nodo();
                nuevo.info=x;
                if(pos==1) {
                        nuevo.sig=raiz;
                        if(raiz!=null) {
                                raiz.ant=nuevo;
                        }
                        raiz=nuevo;
                }else {
                        if(pos==cantidad()+1) {
                                Nodo reco=raiz;
                                while(reco.sig!=null) {
                                        reco=reco.sig;
                                }
                                reco.sig=nuevo;
                                nuevo.ant=reco;
                        }else {
                                Nodo reco=raiz;
                                for(int f=1; f<=pos-2; f++) {
                                        reco=reco.sig;
                                }
                                Nodo siguiente=reco.sig;
                                reco.sig=nuevo;
```

```
                        nuevo.ant=reco;
                        nuevo.sig=siguiente;
                        siguiente.ant=nuevo;
                    }
                }
            }

            }

            public int extraer(int pos) {
                if(pos<=cantidad()) {
                    if(pos==1) {
                        int informacion=raiz.info;
                        raiz=raiz.sig;
                        if(raiz!=null) {
                            raiz.ant=null;
                        }
                        return informacion;
                    }else {
                        Nodo reco=raiz;
                        for(int f=1; f<=pos-2; f++) {
                            reco=reco.sig;
                        }
                        Nodo prox=reco.sig;
                        reco.sig=prox.sig;
                        Nodo siguiente=prox.sig;
                        if(siguiente!=null) {
                            siguiente.ant=reco;
                        }
                        return prox.info;
                    }
                }else {
                    return Integer.MAX_VALUE;
                }
            }


        public static void main(String[] args) {
            ListaGenericaDoble lg=new ListaGenericaDoble();
            lg.insertarPrimero(10);
            lg.insertarPrimero(5);
            lg.imprimir();
            lg.insertarPrimero(20);
            lg.imprimir();
            lg.insertarFinal(200);
            lg.imprimir();
            lg.imprimirInversa();
            System.out.println("La cantidad de nodos: "+lg.cantidad());
            lg.insertar(1, 3000);
            lg.imprimir();
            lg.insertar(2,  2000);
            lg.imprimir();
            System.out.println("Extraemos el de la tercera posición
"+lg.extraer(3));
            lg.imprimir();
        }
}
```

Si ejecutamos este será el resultado:

```
5-10-
20-5-10-
20-5-10-200-
200-10-5-20-
La cantidad de nodos: 4
3000-20-5-10-200-
3000-2000-20-5-10-200-
Extraemos el de la tercera posición 20
3000-2000-5-10-200-
```

Capítulo 151.- Estructuras dinámica: Listas genéricas doblemente encadenadas – 2

Problema propuesto:

Plantear una clase para administrar una lista genérica doblemente encadenada implementando los siguientes métodos:

a) Insertar un nodo al principio de la lista.
b) Insertar un nodo al final de la lista.
c) Insertar un nodo en la segunda posición. Si la lista está vacía no se inserta el nodo.
d) Insertar un nodo en la anteúltima posición.
e) Borrar el primer nodo.
f) Borrar el segundo nodo.
g) Borrar el último nodo.
h) Borrar el nodo con información mayor.

```
public class ListaGenericaDoble {

  class Nodo {
     int info;
     Nodo ant,sig;
  }

  private Nodo raiz;

  public ListaGenericaDoble () {
     raiz=null;
  }

  void insertarPrimero(int x)
  {
       Nodo nuevo = new Nodo ();
     nuevo.info = x;
     nuevo.sig=raiz;
     if (raiz!=null)
        raiz.ant=nuevo;
     raiz=nuevo;
  }

  public void insertarUtlimo(int x) {
       Nodo nuevo = new Nodo ();
     nuevo.info = x;
     if (raiz==null)
        raiz=nuevo;
     else {
        Nodo reco=raiz;
        while (reco.sig!=null) {
```

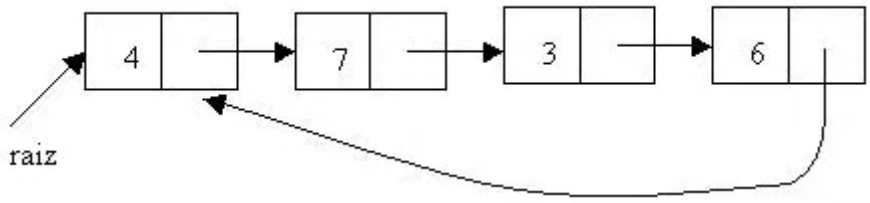
```
            reco=reco.sig;
        }
        reco.sig=nuevo;
        nuevo.ant=reco;
    }
}

public void insertarSegundo(int x) {
    if (raiz!=null) {
         Nodo nuevo = new Nodo ();
        nuevo.info = x;
        if (raiz.sig==null) {
            //Hay un solo nodo.
            raiz.sig=nuevo;
            nuevo.ant=raiz;
        } else {
            Nodo tercero=raiz.sig;
            nuevo.sig=tercero;
            tercero.ant=nuevo;
            raiz.sig=nuevo;
            nuevo.ant=raiz;
        }
    }
}

public void insertarAnteUltimo(int x) {
    if (raiz!=null) {
         Nodo nuevo = new Nodo ();
        nuevo.info = x;
        if (raiz.sig==null) {
            //Hay un solo nodo.
            nuevo.sig=raiz;
            raiz=nuevo;
        } else {
            Nodo reco=raiz;
            while (reco.sig!=null) {
                reco=reco.sig;
            }
            Nodo anterior=reco.ant;
            nuevo.sig=reco;
            nuevo.ant=anterior;
            anterior.sig=nuevo;
            reco.ant=nuevo;
        }
    }
}

public void borrarPrimero() {
```

```
    if (raiz!=null) {
       raiz=raiz.sig;
       if (raiz!=null) {
          raiz.ant=null;
       }
    }
}

public void borrarSegundo() {
    if (raiz!=null) {
       if (raiz.sig!=null) {
          Nodo tercero=raiz.sig;
          tercero=tercero.sig;
          raiz.sig=tercero;
          if (tercero!=null)
             tercero.ant=raiz;
       }
    }
}

public void borrarUltimo () {
    if (raiz!=null) {
       if (raiz.sig==null) {
          raiz=null;
       } else {
          Nodo reco=raiz;
          while(reco.sig!=null) {
             reco=reco.sig;
          }
          reco=reco.ant;
          reco.sig=null;
       }
    }

}
public void imprimir () {
    Nodo reco = raiz;
    while (reco != null) {
       System.out.print (reco.info + "-");
       reco = reco.sig;
    }
    System.out.println();
}

public void borrarMayor() {
    if (raiz!=null) {
       Nodo reco=raiz;
       int may=raiz.info;
```

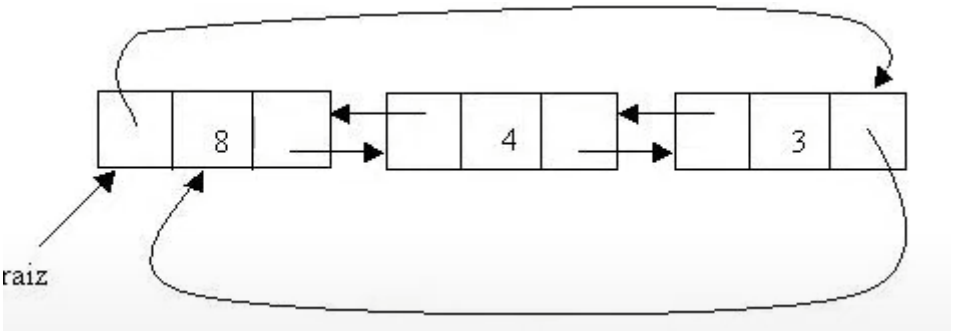
```
        while (reco!=null) {
            if (reco.info>may) {
                may=reco.info;
            }
            reco=reco.sig;
        }
        reco=raiz;
        while (reco!=null) {
            if (reco.info==may) {
                if (reco==raiz) {
                    raiz=raiz.sig;
                    if (raiz!=null)
                        raiz.ant=null;
                    reco=raiz;
                } else {
                    Nodo atras=reco.ant;
                    atras.sig=reco.sig;
                    reco=reco.sig;
                    if (reco!=null)
                        reco.ant=atras;
                }
            } else {
                reco=reco.sig;
            }
        }
    }
}
```

Capítulo 152.- Estructuras dinámicas: Listas genéricas circulares

Una lista circular simplemente encadena, la podemos representar gráficamente:



Una lista circular puede también ser doblemente encadenada:



Implementar los métodos siguientes con una lista doblemente encadenada circular:

public void insertarPrimero(int x)
public boolean vaica()
public void imprimir()
public void insertarUltimo(int x)
public int cantidad()
public void borrar(int pos)

Este será el Código:

```
public class ListaDobleCircular {
	class Nodo{
		int info;
		Nodo sig, ant;
	}

	private Nodo raiz;

	public ListaDobleCircular() {
		raiz=null; //Se inicializa por defecto.
	}

	public void insertarPrimero(int x) {
		Nodo nuevo=new Nodo();
		nuevo.info=x;
		if(raiz==null) {
			raiz=nuevo; //raiz apunte al nuevo nodo.
			raiz.sig=raiz; //el primer nodo apunta a si mismo.
			raiz.ant=raiz; // también apunta a si mismo.
		}else {
			Nodo ultimo=raiz.ant; //el nodo último apunta al nodo
anterior.
```

```
                raiz.ant=nuevo;
                nuevo.sig=raiz;
                nuevo.ant=ultimo;
                ultimo.sig=nuevo;
                raiz=nuevo;
        }
    }

    public void imprimir() {
        if(raiz!=null) {
                Nodo reco=raiz;
                do {
                        System.out.print(reco.info+" - ");
                        reco=reco.sig;
                }while(reco!=raiz);
                System.out.println();
        }
    }

    public boolean vacio() {
        return raiz==null;
    }

    public void insertarUltimo(int x) {
        Nodo nuevo=new Nodo();
        nuevo.info=x;
        if(raiz==null) {
                raiz=nuevo; //raiz apunte al nuevo nodo.
                raiz.sig=raiz; //el primer nodo apunta a si mismo.
                raiz.ant=raiz; // también apunta a si mismo.
        }else {
                Nodo ultimo=raiz.ant; //el nodo último apunta al nodo
anterior.
                raiz.ant=nuevo;
                nuevo.sig=raiz;
                nuevo.ant=ultimo;
                ultimo.sig=nuevo;
        }
    }

    public int cantidad() {
        int cant=0;
        Nodo reco=raiz;
        if (raiz!=null) {
                do  {
                        cant++;
                        reco=reco.sig;
                } while (reco!=raiz);
                return cant;
        }else {
                return 0;
        }
    }

    public void borrar(int pos) {
        if(pos<=cantidad()) {
                if(pos==1) {
                        if(cantidad()==1) {
                                raiz=null;
```

```
                        }else {
                                Nodo ultimo=raiz.ant;
                                raiz=raiz.sig;
                                ultimo.sig=raiz;
                                raiz.ant=ultimo;
                        }

                }else {
                        Nodo reco=raiz;
                        for(int f=1; f<=pos-1; f++) {
                                reco=reco.sig;
                        }
                        Nodo anterior=reco.ant;
                        reco=reco.sig;
                        anterior.sig=reco;
                        reco.ant=anterior;
                }
        }
    }

    public static void main(String[] args) {
            ListaDobleCircular ldc=new ListaDobleCircular();
            if (ldc.vacio()) {
                    System.out.println("La lista circular está vacía");
            }else {
                    System.out.println("La lista circular no esá vacía");
            }
            ldc.insertarPrimero(20);
            ldc.insertarPrimero(10);
            ldc.insertarPrimero(5);
            ldc.imprimir();
            if (ldc.vacio()) {
                    System.out.println("La lista circular está vacía");
            }else {
                    System.out.println("La lista circular no está vacía");
            }
            ldc.insertarUltimo(500);
            ldc.imprimir();
            System.out.println("La lista circular tiene "+ldc.cantidad()+"
elementos");
            ldc.borrar(1);
            ldc.imprimir();
            ldc.borrar(2);
            ldc.imprimir();
    }
}
```

Si ejecutamos este será el resultado:

```
La lista circular está vacía
5 - 10 - 20 -
La lista circular no está vacía
5 - 10 - 20 - 500 -
La lista circular tiene 4 elementos
10 - 20 - 500 -
10 - 500 -
```

Capítulo 153.- Recursividad: Conceptos básicos – 1

En java los métodos pueden llamarse a sí mismos. Si dentro de un método existe la llamada a sí mismo decimos que el método es recursivo.

Cuando un método se llama a sí mismo, se asigna espacio en la pila para las nuevas variables locales y parámetros.

```
1  public class Recursividad {
2
3      public void imprimir() {
4          imprimir();
5      }
6
7      public static void main(String[] args) {
8          Recursividad r=new Recursividad();
9          r.imprimir();
10     }
11 }
```

En este ejemplo como el método que ejecutamos se vuelve a llamar a si mismo esto provoca un error de desbordamiento ya que nunca termina.

```
1  public class Recursividad {
2
3      public void imprimir(int x) {
4          imprimir(x-1);
5      }
6
7      public static void main(String[] args) {
8          Recursividad r=new Recursividad();
9          r.imprimir(5);
10     }
11 }
```

Con estas modificaciones también nos genera un error.

```
1  public class Recursividad {
2
3      public void imprimir(int x) {
4          System.out.println(x);
5          imprimir(x-1);
6      }
7
8      public static void main(String[] args) {
9          Recursividad r=new Recursividad();
10         r.imprimir(5);
11     }
12 }
```

Si ejecutamos antes de generar un error el valor de la x será de -17338.

. . .

```
-17365
-17366
-17367
-17368
```

Antes se han realizado más de 17.368 llamadas recursivas.

```java
public class Recursividad {

    public void imprimir(int x) {
        if(x>0) {
            System.out.println(x);
            imprimir(x-1);
        }
    }

    public static void main(String[] args) {
        Recursividad r=new Recursividad();
        r.imprimir(5);
    }
}
```

Para evitar el error de desbordamiento por recursividad esta tienen que estar controlada por un condicional que evite cuando no se cumpla de llamarse a si misma.

Vamos a ejecutar:

```
5
4
3
2
1
```

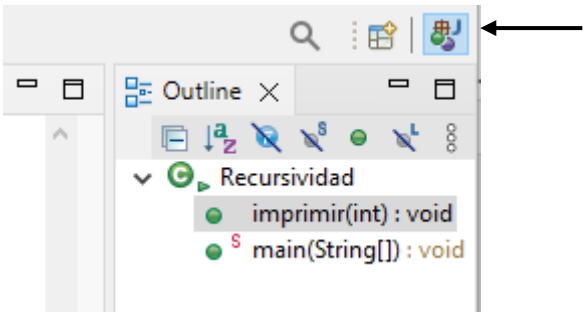
```java
public class Recursividad {

    public void imprimir(int x) {
        if(x>0) {
            imprimir(x-1);
            System.out.println(x);

        }
    }

    public static void main(String[] args) {
        Recursividad r=new Recursividad();
        r.imprimir(5);
    }
}
```

Que pasaría si cambiamos el orden en la parte de la condición llamando al método antes de imprimir.

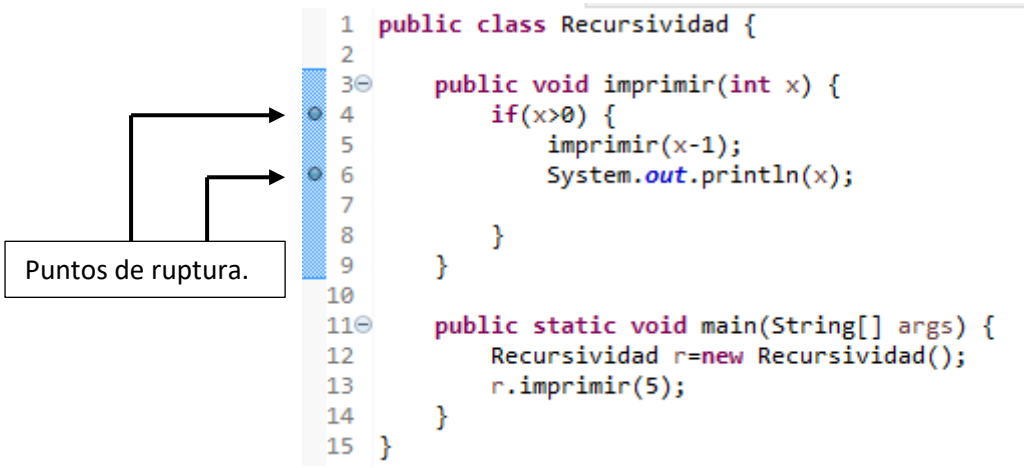
```
1
2
3
4
5
```

Antes se imprimía de mayor a menor y ahora se muestra de menor a mayor.

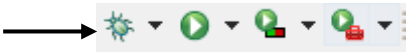
Llamamos al método nuevamente con el valor de x-1, esto hace que se vuelva a ejecutar el método sin pasar por la línea de impresión, cuando la x tenga el valor de 0 ya no se llamará nuevamente el método pero se irá a la línea siguiente después del método ya que tiene que terminar de ejecutar cada uno de los métodos que fueron llamados hasta terminar, de este modo el último método x era igual a 1, el penúltimo x era igual a 2 y así sucesivamente hasta terminar todos los métodos que fueron llamados.

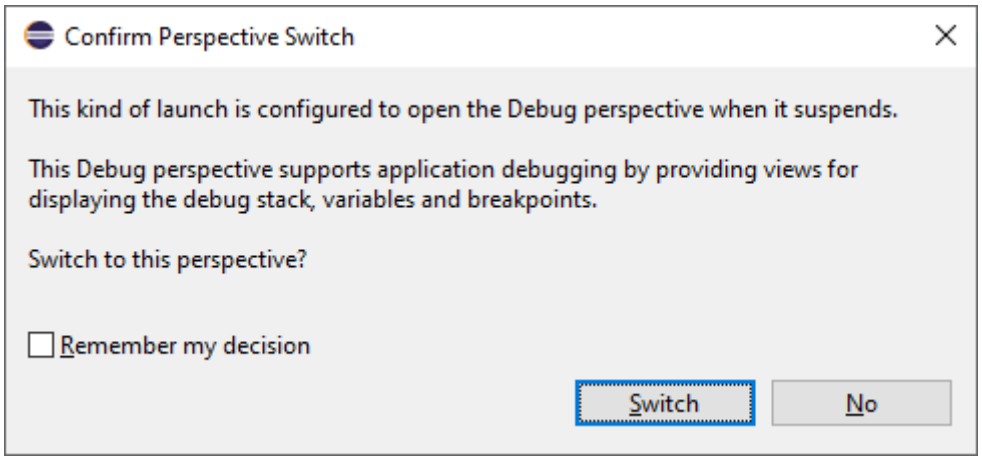


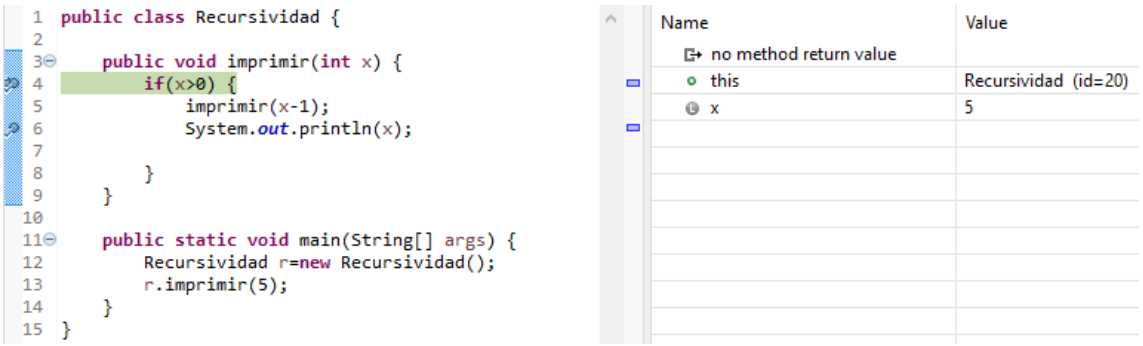
Con el estado de vista de depuración veremos como se ejecuta el programa, para ello haremos doble clic en las líneas que queremos que muestre dicho estado de depuración.



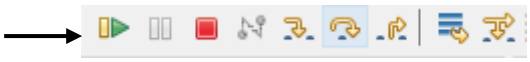
Ahora para ejecutarse paso a paso, seleccionaremos el siguiente botón:







En los puntos donde se detiene el programa, muestra el valor que tiene x, pasa ir al siguiente punto seleccionaremos el botón:



Le iremos dando hasta que termine de ejecutar el programa.

Si hacemos doble clic sobre los puntos de ruptura, estos se quitarán.

```
1  public class Recursividad {
2
3      public void imprimir(int x) {
4          if(x>0) {
5              imprimir(x-1);
6              System.out.println(x);
7
8          }
9      }
10
11     public static void main(String[] args) {
12         Recursividad r=new Recursividad();
13         r.imprimir(5);
14     }
15 }
```

En resumen podemos ver que si llamamos 5 veces al mismo método, este tendrá que imprimir el contenido del valor de x pero en modo inverso.

Capítulo 154.- Recursividad: Conceptos básicos – 2

Problema:

Otro problema típico que se presenta para analizar la recursividad es el obtener el factorial de un número.

Recordar que el factorial de un número es el resultado que se obtiene de multiplicar dicho número por el anterior y así sucesivamente hasta llegar a uno.

Ej. el factorial de 4 es 4 * 3 *2 * 1 es decir 24.

```
1  public class Factorial {
2
3      int y=1;
4
5      public int factor(int fact) {
6          if (fact>0) {
7              int valor=fact*factor(fact-1);
8                  return valor;
9          }else {
10             return 1;
11         }
12     }
13
14     public static void main(String[] args) {
15         Factorial fac=new Factorial();
16         System.out.println("El factorial de 4 es "+fac.factor(4));
17     }
18 }
```

Si ejecutamos este será el resultado:

```
El factorial de 4 es 24
```

Capítulo 155.- Recursividad: Conceptos básicos – 3

Implementar un método recursivo para ordenar los elementos de un vector.

```
1
2  public class Recursividad {
3      private int[] vec= {200,45,33,44};
4
5      public void ordenar() {
6          ordenar(vec, vec.length);
7      }
8
9      private void ordenar(int[] v, int cant) {
10         if(cant>1) {
11             for(int f=0; f<cant-1; f++) {
12                 if(v[f]>v[f+1]) {
13                     int aux=v[f];
14                     v[f]=v[f+1];
15                     v[f+1]=aux;
16                 }
17             }
18             ordenar(v, cant-1);
19         }
20     }
21
22     public void imprimir() {
23         for(int f=0; f<vec.length; f++) {
24             System.out.print(vec[f]+"-");
25         }
26         System.out.println();
27     }
28
29     public static void main(String[] args) {
30         Recursividad r=new Recursividad();
31         r.imprimir();
32         r.ordenar();
33         r.imprimir();
34         }
35 }
```

Si ejecutamos este será el resultado:

```
200-45-33-44-
33-44-45-200-
```

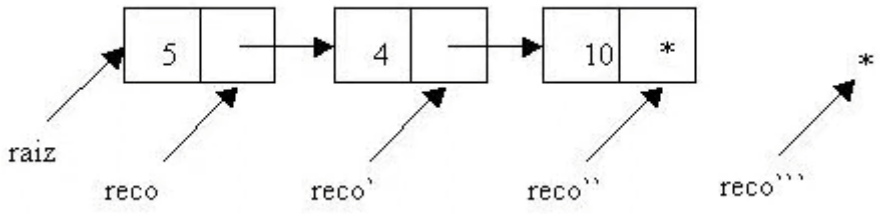
Capítulo 156.- Recursividad: Problema donde conviene aplicar la recursividad – 1

Problema:

Imprimir la información de una lista simplemente encadenada de atrás para adelante.

El empleo de estructuras repetitivas para resolver este problema es bastante engorroso y lento (debemos avanzar hasta el último nodo e imprimir, luego avanzar desde el principio has el antepenúltimo nodo y así sucesivamente).

El empleo de la recursividad para este problema hace más sencillo su solución.




```
1  public class Recursividad {
2
3      class Nodo{
4          int info;
5          Nodo sig;
6      }
7
8      private Nodo raiz;
9
10     public void insertarPrimero(int x) {
11         Nodo nuevo=new Nodo();
12         nuevo.info=x;
13         nuevo.sig=raiz;
14         raiz=nuevo;
15     }
16
17     public void imprimir() {
18         Nodo reco=raiz;
19         while(reco!=null) {
20             System.out.print(reco.info+"-");
21             reco=reco.sig;
22         }
23         System.out.println();
24     }
25
26     private void imprimirInversa(Nodo reco) {
27         if(reco!=null) {
28             imprimirInversa(reco.sig);
29             System.out.print(reco.info+"-");
30         }
31 }
32
33     public void imprimirInversa() {
34         imprimirInversa(raiz);
35         System.out.println();
36     }
37
```

```
38      public static void main(String[] args) {
39          Recursividad r=new Recursividad();
40          r.insertarPrimero(10);
41          r.insertarPrimero(4);
42          r.insertarPrimero(5);
43          r.imprimir();
44          r.imprimirInversa();
45      }
46  }
```

Si ejecutamos este será el resultado:

```
5-4-10-
10-4-5-
```

Que pasará si antes de llamar al método recursivo mostramos el nodo por pantalla:

```
26      private void imprimirInversa(Nodo reco) {
27          if(reco!=null) {
28              System.out.print(reco.info+"-");
29              imprimirInversa(reco.sig);
30          }
31  }
```

Si ejecutamos este sería el resultado:

```
5-4-10-
5-4-10-
```

Capítulo 157.- Recursividad: Problemas donde conviene aplicar la recursividad – 2

Problema:

Recuperar un árbol de directorios en forma recursiva.

```
1  import java.io.File;
2  public class Recursividad {
3
4      public void leer(String directorio) {
5          File ar=new File(directorio);
6          String[] dir=ar.list();
7          for(int f=0; f<dir.length; f++) {
8              File ar2=new File(directorio+dir[f]);
9              if(ar2.isFile()) {
10                 System.out.println("Archivo:"+dir[f]);
11             }
12             if(ar2.isDirectory()) {
13                 System.out.println("Directorio:"+dir[f].toUpperCase());
14                 leer(directorio+dir[f]+"\\");
15             }
16         }
17     }
18
19     public static void main(String[] args) {
20         Recursividad rec=new Recursividad();
21         rec.leer("D:\\CursoJava\\");
22     }
23 }
```

Si ejecutamos este será el resultado:

```
Directorio:TRABAJANDOPILAS
Archivo:.classpath
Archivo:.project
Directorio:.SETTINGS
Archivo:org.eclipse.core.resources.prefs
Archivo:org.eclipse.jdt.core.prefs
Directorio:BIN
Archivo:Formulario.class
Archivo:Pila$Nodo.class
Archivo:Pila.class
Directorio:SRC
Archivo:Formulario.java
Archivo:Pila.java
```

Para no tener un error intentando leer una carpeta protegida:

```
1  import java.io.File;
2  public class Recursividad {
3
4      public void leer(String directorio) {
5          File ar=new File(directorio);
6          String[] dir=ar.list();
7          if(dir!=null) {
8              for(int f=0; f<dir.length; f++) {
9                  File ar2=new File(directorio+dir[f]);
10                 if(ar2.isFile()) {
11                     System.out.println("Archivo:"+dir[f]);
12                 }
```

```
13                  if(ar2.isDirectory()) {
14                      System.out.println("Directorio:"+dir[f].toUpperCase());
15                      leer(directorio+dir[f]+"\\");
16                  }
17              }
18          }
19      }
20
21      public static void main(String[] args) {
22          Recursividad rec=new Recursividad();
23          rec.leer("D:\\CursoJava\\");
24      }
25  }
```

Capítulo 158.- Recursividad: Problemas donde conviene aplicar la recursividad – 3

Problema:

Desarrollar un programa que permita recorrer un laberinto e indique si tiene salida o no.

Para resolver este problema al laberinto lo representamos con una matriz de 10 x 10 JLabel.

El valor:

"0" Representa pasillo.
"1" Representa pared.
"9" Persona
"s" Salida

A la salida ubicada en la parte de la fila 9 y columna 9 de la matriz. La persona comienza a recorrer el laberinto en la fila 0 columna 0. Los ceros y unos disponerlos en forma aleatoria (con la función random).

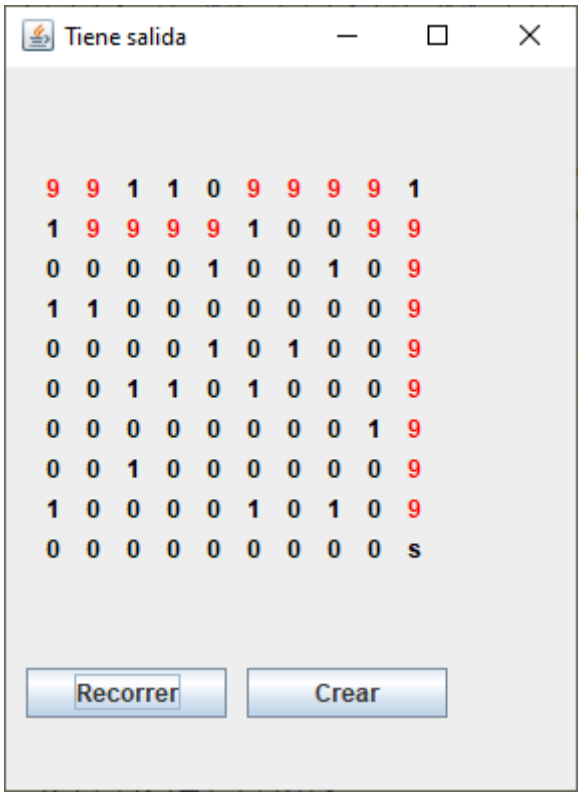
Vamos a programar:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class Laberinto extends JFrame implements ActionListener{
	JLabel [][] l;
	JButton b1;
	JButton b2;
	boolean salida;
	Laberinto()
	{
		setLayout(null);
		l=new JLabel[10][10];
		for(int f=0; f<10; f++) {
			for(int c=0; c<10; c++) {
				l[f][c]=new JLabel();
				l[f][c].setBounds(20+c*20, 50+f*20, 20, 20);
				add(l[f][c]);
			}
		}
		b1=new JButton("Recorrer");
		b1.setBounds(10,300,100,25);
		add(b1);
		b1.addActionListener(this);
		b2=new JButton("Crear");
		b2.setBounds(120,300,100,25);
		add(b2);
		b2.addActionListener(this);
		crear();
	}

	public void crear()
	{
		for(int f=0; f<10; f++) {
			for(int c=0; c<10; c++) {
				int a=(int)(Math.random()*4);
```

```
                        l[f][c].setForeground(Color.black);
                        if(a==0) {
                              l[f][c].setText("1");
                        }else {
                              l[f][c].setText("0");
                        }
                  }
            }
            l[9][9].setText("s");
            l[0][0].setText("0");
      }

      public void recorrer(int fil, int col)
      {
            if(fil>=0 && fil<10 && col>=0 && col<10 && salida==false) {
                  if(l[fil][col].getText().equals("s"))
                        salida=true;
            else
                  if(l[fil][col].getText().equals("0")) {
                        l[fil][col].setText("9");
                        l[fil][col].setForeground(Color.red);
                        recorrer(fil,col+1);
                        recorrer(fil+1,col);
                        recorrer(fil-1,col+1);
                        recorrer(fil,col-1);
                  }
            }
      }

      public static void main(String[] args) {
            Laberinto l=new Laberinto();
            l.setBounds(0,0,300,400);
            l.setVisible(true);
            l.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      }

      @Override
      public void actionPerformed(ActionEvent e) {
            if(e.getSource()==b1) {
                  salida=false;
                  recorrer(0,0);
                  if (salida)
                        setTitle("Tiene salida");
                  else
                        setTitle("No tiene salida");
            }
            if(e.getSource()==b2)
                  crear();
      }
}
```

Si ejecutamos este será el resultado:

Cuando tiene salida:



Cuando no tiene salida:

Capítulo 159.- Recursividad: Problemas donde conviene aplicar la recursividad – 4

Problema:

Desarrollar el juego del Buscaminas. Definir una matriz de 10*10 de JButton y disponer una 'b' para las bombas (10 diez) en cero en los botones que no tienen bombas en su perímetro, un 1 si tiene una bomba en su perímetro y así sucesivamente. Cuando se presiona un botón si hay un cero proceder en forma recursiva a destapar los botones que se encuentran a sus lados. Disponer el mismo color de frente y fondo de los botones para que el jugador no pueda ver si hay bombas o no.

Vamos a programar:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class Buscaminas extends JFrame implements ActionListener{
	JButton [] [] bot;
	JButton b1;
	Buscaminas(){
		setLayout(null);
		bot=new JButton [10][10];
		for(int f=0; f<10; f++) {
			for(int c=0; c<10; c++) {
				bot[f][c] = new JButton("0");
				bot[f][c].setBounds(20+c*41, 50+f*41, 41, 41);
				bot[f][c].setBackground(Color.lightGray);
				bot[f][c].setForeground(Color.lightGray);
				bot[f][c].addActionListener(this);
				add(bot[f][c]);
			}
		}
		b1=new JButton("Reiniciar");
		b1.setBounds(20,470,100,30);
		add(b1);
		b1.addActionListener(this);
		disponerBombas();
		contarBombasPerimetro();
	}

	void disponerBombas() {
		int cantidad=10;
		do {
			int fila=(int)(Math.random()*10);
			int columna=(int)(Math.random()*10);
			if(bot[fila][columna].getText().equals("b")==false) {
				bot[fila][columna].setText("b");
				cantidad--;
			}

		}while(cantidad!=0);
	}

	void contarBombasPerimetro() {
		for(int f=0; f<10; f++) {
			for(int c=0; c<10; c++) {
				if(bot[f][c].getText().equals("0")==true) {
```

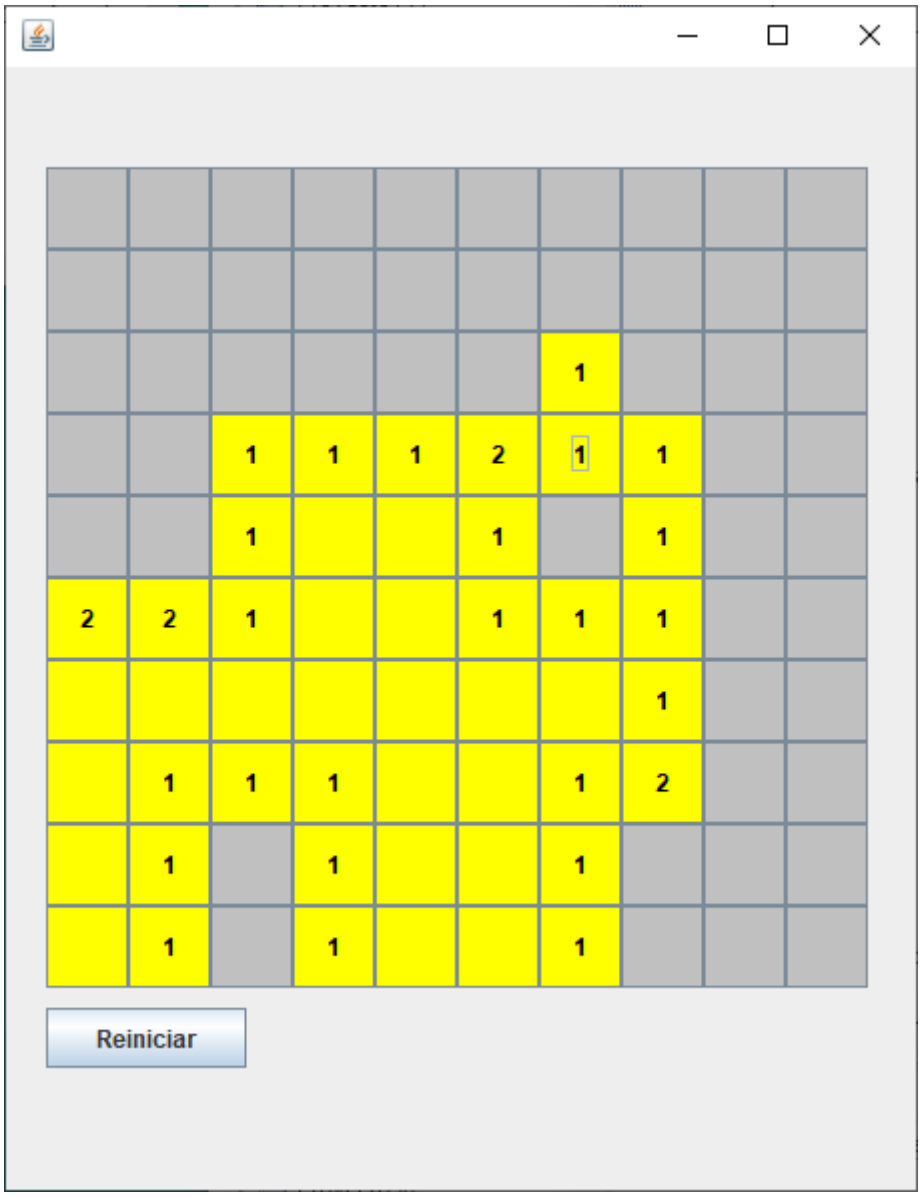
```
                        int cant=contarCoordenadas(f,c);
                        bot[f][c].setText(String.valueOf(cant));
                  }
            }
      }
}

int contarCoordenadas(int fila, int columna) {
      int total=0;
      if(fila-1>=0 && columna-1>=0) {
            if(bot[fila-1][columna-1].getText().equals("b")==true)
                  total++;
      }
      if (fila - 1 >= 0)
  {
      if (bot [fila - 1] [columna].getText ().equals ("b") == true)
      total++;
  }
  if (fila - 1 >= 0 && columna + 1 < 10)
  {
      if (bot [fila - 1] [columna + 1].getText ().equals ("b") == true)
      total++;
  }

  if (columna + 1 < 10)
  {
      if (bot [fila] [columna + 1].getText ().equals ("b") == true)
      total++;
  }
  if (fila + 1 < 10 && columna + 1 < 10)
  {
      if (bot [fila + 1] [columna + 1].getText ().equals ("b") == true)
      total++;
  }

  if (fila + 1 < 10)
  {
      if (bot [fila + 1] [columna].getText ().equals ("b") == true)
      total++;
  }
  if (fila + 1 < 10 && columna - 1 >= 0)
  {
      if (bot [fila + 1] [columna - 1].getText ().equals ("b") == true)
      total++;
  }
  if (columna - 1 >= 0)
  {
      if (bot [fila] [columna - 1].getText ().equals ("b") == true)
      total++;
  }
  return total;
}

void desactivarJuego() {
      for(int f=0; f<10; f++) {
            for(int c=0; c<10; c++) {
                  bot[f][c].setEnabled(false);
            }
      }
```

```
    }

    void reiniciar() {
          setTitle("");
          for (int f=0; f<10; f++) {
                for(int c=0; c<10; c++) {
                      bot[f][c].setText("0");
                      bot[f][c].setEnabled(true);
                      bot[f][c].setBackground(Color.lightGray);
                      bot[f][c].setForeground(Color.lightGray);
                }
          }
          disponerBombas();
          contarBombasPerimetro();
    }

    void verificarTriunfo ()
{
     int cant = 0;
     for (int f = 0 ; f < 10 ; f++)
     {
         for (int c = 0 ; c < 10 ; c++)
         {
             Color col = bot [f] [c].getBackground ();
             if (col == Color.yellow)
                 cant++;
         }
     }
     if (cant == 90)
     {
         setTitle ("Ganooooooooo");
         desactivarJuego ();
     }
}

   void recorrer (int fil, int col)
   {
       if (fil >= 0 && fil < 10 && col >= 0 && col < 10)
       {
           if (bot [fil] [col].getText ().equals ("0"))
           {
               bot [fil] [col].setText (" ");
               bot [fil] [col].setBackground (Color.yellow);
               recorrer (fil, col + 1);
               recorrer (fil, col - 1);
               recorrer (fil + 1, col);
               recorrer (fil - 1, col);
               recorrer (fil - 1, col - 1);
               recorrer (fil - 1, col + 1);
               recorrer (fil + 1, col + 1);
               recorrer (fil + 1, col - 1);
           }
           else
               if (bot [fil] [col].getText ().equals ("1") == true ||
                   bot [fil] [col].getText ().equals ("2") == true ||
                   bot [fil] [col].getText ().equals ("3") == true ||
                   bot [fil] [col].getText ().equals ("4") == true ||
                   bot [fil] [col].getText ().equals ("5") == true ||
                   bot [fil] [col].getText ().equals ("6") == true ||
```

La recursividad se encuentra en el método recorrer.

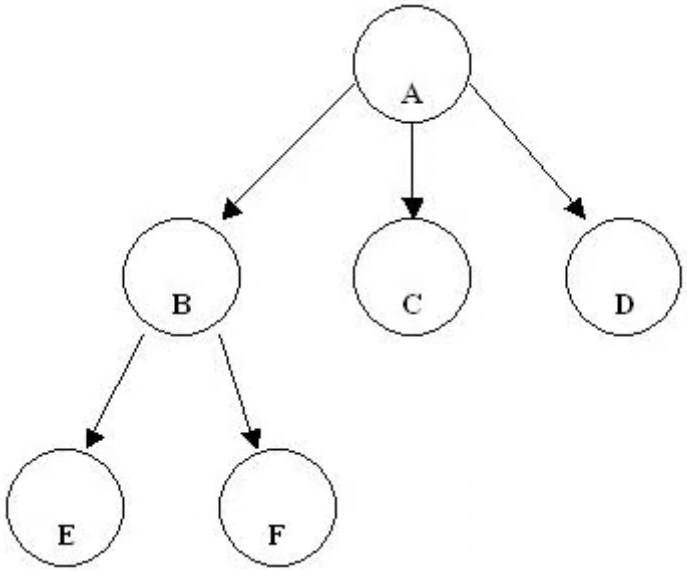
```
                    bot [fil] [col].getText ().equals ("7") == true ||
                    bot [fil] [col].getText ().equals ("8") == true)
                {
                    bot [fil] [col].setBackground (Color.yellow);
                    bot [fil] [col].setForeground (Color.black);
                }
            }
        }

    public static void main(String[] args) {
          Buscaminas m=new Buscaminas();
          m.setBounds(0,0,470,600);
          m.setVisible(true);
          m.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
           if (e.getSource () == b1)
           {
               reiniciar ();
           }
           for (int f = 0 ; f < 10 ; f++)
           {
               for (int c = 0 ; c < 10 ; c++)
               {
               if (e.getSource () == bot [f] [c])
               {
                   if (bot [f] [c].getText ().equals ("b") == true)
                   {
                       setTitle ("Boooooooooooooomm");
                       desactivarJuego ();
                   }
                   else
                   if (bot [f] [c].getText ().equals ("0") == true)
                   {
                      recorrer (f, c);
                   }
                   else
                       if (bot [f] [c].getText ().equals ("1") == true ||
                           bot [f] [c].getText ().equals ("2") == true ||
                           bot [f] [c].getText ().equals ("3") == true ||
                           bot [f] [c].getText ().equals ("4") == true ||
                           bot [f] [c].getText ().equals ("5") == true ||
                           bot [f] [c].getText ().equals ("6") == true ||
                           bot [f] [c].getText ().equals ("7") == true ||
                           bot [f] [c].getText ().equals ("8") == true)
                       {
                           bot [f] [c].setBackground (Color.yellow);
                           bot [f] [c].setForeground (Color.black);
                       }
               }
               }
           }
          verificarTriunfo ();
    }
}
```

Si ejecutamos este será el resultado:


1
1 1 1 2 1 1
1 1 1
2 2 1 1 1 1
1
1 1 1 1 2
1 1 1
1 1 1
Reiniciar

Capítulo 160.- Estructuras dinámicas: Conceptos de árboles

Igual que la lista, el árbol es una estructura de datos. Son muy eficientes para la búsqueda de información. Los árboles soportan estructuras no lineales.



Algunos conceptos de la estructura de datos tipo árbol:

Nodo hoja: Es un nodo sin descendientes (Noto terminal)
Ej. Nodos E-F-C y D.

Nodo interior: Es un nodo que no es hoja.
Ej. Nodos A y B.

Nivel de un árbol: el nodo A está en el nivel 1 y sus descendientes directos están en el nivel 2 y así sucesivamente.
El nivel del árbol esta dado por el nodo de máximo nivel.
Ej. Este árbol es de nivel 3.

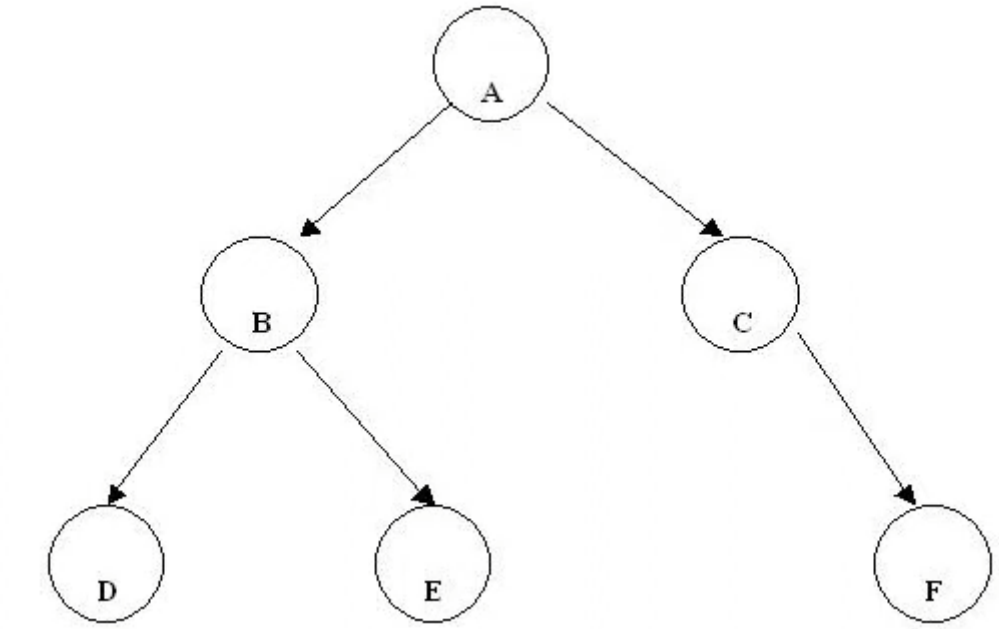
Grado de un nodo: es el número de nodos hijos que tiene dicho nodo (solo se tiene en cuenta los nodos interiores).
Ej. El nodo A tiene grado 3.
El nodo B tiene grado 2.
Los otros nodos no tienen grado porque no tienen descendientes.

Grado de un árbol: Es el máximo de los grados de todos los nodos de un árbol.
Ej. El grado del árbol es 3.

Longitud de camino del nodo x: Al número de arcos que deben ser recorridos para llegar a un nodo x, partiendo de la raíz.

La raíz tiene longitud de camino 1, sus descendientes directos tienen longitud de camino 2, etc.
En forma general un nodo en el nivel i tiene longitud de camino i.

Árbol binario: Un árbol binario si cada nodo tiene como máximo 2 descendientes.



Para cada nodo está definido el subárbol izquierdo y el derecho.

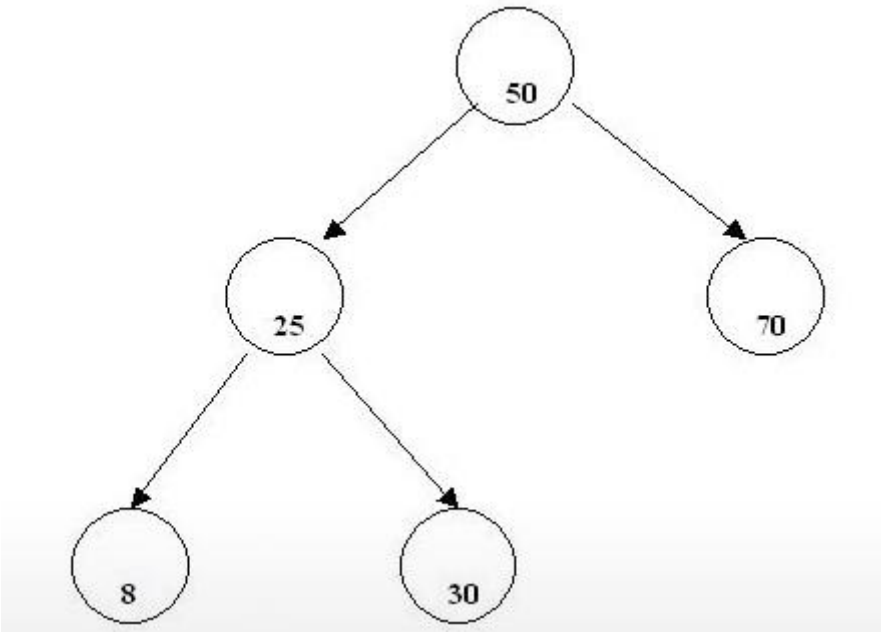
Para el nodo A el subárbol izquierdo está constituido por los nodos B, D y E. Y el subárbol derecho está formado por los nodos C y F.

Lo mismo para el nodo B tiene el subárbol izquierdo con un nodo (D) y un nodo en el subárbol derecho (E).

El nodo D tiene ambos subárboles vacíos.

El nodo C tiene el subárbol izquierdo vacío y el subárbol derecho con el nodo (F).

Árbol binario ordenado: Si para cada nodo del árbol, los nodos ubicados a la izquierda son inferiores al que consideramos raíz para ese momento y los nodos ubicados a la derecha son mayores que la raíz.



Ej. Analicemos si se trata de un árbol binario ordenado:

Para el nodo que tiene 50:

Los nodos del subárbol izquierdo son todos menores a 50? 8, 25, 30 Si.

Los nodos del subárbol derecho son todos mayores a 50? 70 Si.

Para el nodo que tiene el 25:

Los nodos del subárbol izquierdo son todos menores a 25? 8 Si.

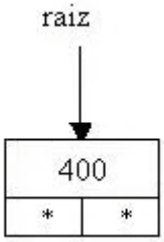
Los nodos del subárbol derecho son todos mayores a 25? 30 Si.

No hace falta analizar los nodos hoja. Si todas las respuestas son afirmativas podemos luego decir que se trata de un árbol binario ordenado.

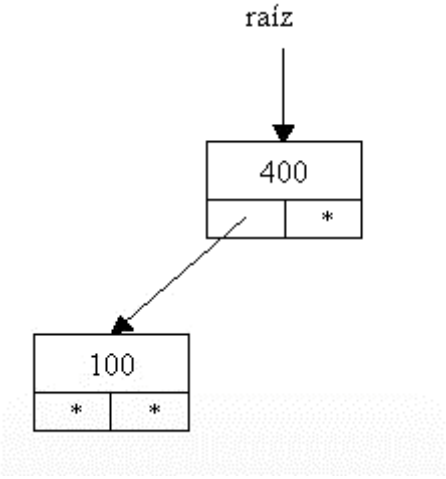
Capítulo 161.- Estructuras dinámicas: Inserción de nodos y recorrido de un árbol binario.

Para administrar un árbol binario ordenado debemos tener especial cuidado en la inserción. Inicialmente el árbol está vacío, es decir raíz apunta a null:

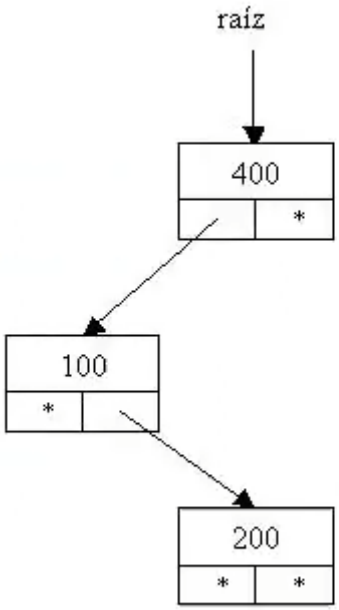
Insertamos el 400.



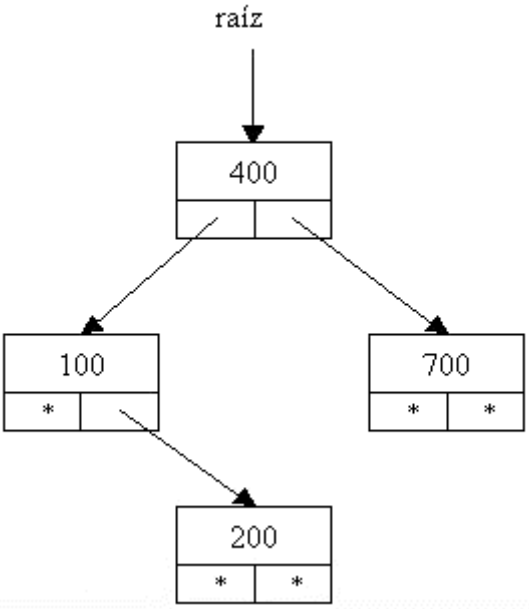
Insertamos el valor 100. Debemos analizar si raíz es distinto a null verificando si 100 es mayor o menor a la información del nodo apuntando por raíz, en este caso es menor y como el subárbol izquierdo es null debemos insertarlo allí.



Insertamos el 200. Hay que tener en cuenta que siempre comenzamos las comparaciones a partir de raíz. El 200 es menor que 400, descendemos por el subárbol izquierdo. Luego analizamos y vemos que el 200 es mayor a 100, debemos avanzar por la derecha. Como el subárbol derecho es null lo insertamos en dicha posición.



Si insertamos el 700 y el árbol será:




```
1  public class ArbolBinarioOrd {
2      class Nodo{
3          int info;
4          Nodo izq, der;
5      }
6
7      private Nodo raiz;
8
9      void insertar(int x) {
10         Nodo nuevo=new Nodo();
11         nuevo.info=x;
12         if(raiz==null) {
13             raiz=nuevo;
14         }else {
```

```
            Nodo anterior=null;
            Nodo reco=raiz;
            while(reco!=null) {
                anterior=reco;
                if(x<reco.info) {
                    reco=reco.izq;
                }else {
                    reco=reco.der;
                }
            }
            if(x<anterior.info) {
                anterior.izq=nuevo;
            }else {
                anterior.der=nuevo;
            }
        }
    }

    public static void main(String[] args) {
        ArbolBinarioOrd arbol=new ArbolBinarioOrd();
        arbol.insertar(400);
        arbol.insertar(100);
        arbol.insertar(200);
        arbol.insertar(700);
    }
}
```

Capítulo 162.- Estructuras dinámicas: Implementación en Java de un árbol binario ordenado – 1

Recorrer: Pasar a través del árbol enumerando cada uno de sus nodos una vez.

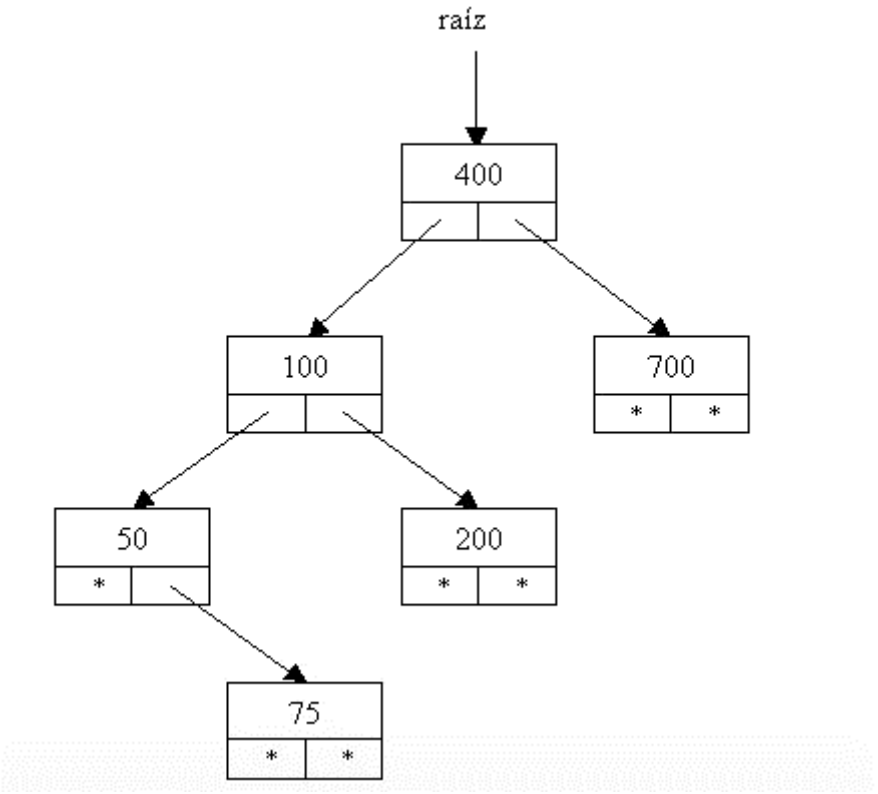
Visitar: Realizar algún procesamiento del nodo.

Los árboles pueden ser recorridos en varios órdenes:

Pre-orden:
- Visitar la raíz.
- Recorrer el subárbol izquierdo en pre-orden.
- Recorrer el subárbol derecho en pre-orden.

Entre-orden:
- Recorrer el subárbol izquierdo en entre-orden.
- Visitar la raíz.
- Recorrer el subárbol derecho en estre-orden.

Post_orden:
- Recorrer el subárbol izquierdo en post-orden.
- Recorrer el subárbol derecho en post-orden.
- Visitar la raiz.



Ejemplo utilizando Pre-orden:

Vamos a programar modificando el código anterior:

```
public class ArbolBinarioOrd {
	class Nodo{
		int info;
		Nodo izq, der;
```

```
	}

	private Nodo raiz;

	void insertar(int x) {
		Nodo nuevo=new Nodo();
		nuevo.info=x;
		if(raiz==null) {
			raiz=nuevo;
		}else {
			Nodo anterior=null;
			Nodo reco=raiz;
			while(reco!=null) {
				anterior=reco;
				if(x<reco.info) {
					reco=reco.izq;
				}else {
					reco=reco.der;
				}
			}
			if(x<anterior.info) {
				anterior.izq=nuevo;
			}else {
				anterior.der=nuevo;
			}
		}
	}

	private void recorrerPre(Nodo reco) {
		if(reco!=null) {
			System.out.print(reco.info+"-");
			recorrerPre(reco.izq);
			recorrerPre(reco.der);
		}
	}

	public void recorrerPre() {
		recorrerPre(raiz);
		System.out.println();
	}

	public static void main(String[] args) {
		ArbolBinarioOrd arbol=new ArbolBinarioOrd();
		arbol.insertar(400);
		arbol.insertar(100);
		arbol.insertar(200);
		arbol.insertar(700);
		arbol.insertar(50);
		arbol.insertar(75);
		arbol.recorrerPre();
	}
}
```

Si ejecutamos:

```
400-100-50-75-200-700-
```

Ejemplo utilizando Entre-orden:

Vamos a programar:

```
public class ArbolBinarioEntre {

	class Nodo{
		int info;
		Nodo izq, der;
	}

	private Nodo raiz;

	void insertar(int x) {
		Nodo nuevo=new Nodo();
		nuevo.info=x;
		if(raiz==null) {
			raiz=nuevo;
		}else {
			Nodo anterior=null;
			Nodo reco=raiz;
			while(reco!=null) {
				anterior=reco;
				if(x<reco.info) {
					reco=reco.izq;
				}else {
					reco=reco.der;
				}
			}
			if(x<anterior.info) {
				anterior.izq=nuevo;
			}else {
				anterior.der=nuevo;
			}
		}
	}

	private void recorrerEntre(Nodo reco) {
		if(reco!=null) {
			recorrerEntre(reco.izq);
			System.out.print(reco.info+"-");
			recorrerEntre(reco.der);
		}
	}

	public void recorrerEntre() {
		recorrerEntre(raiz);
		System.out.println();
	}

	public static void main(String[] args) {
		ArbolBinarioEntre arbol=new ArbolBinarioEntre();
		arbol.insertar(400);
		arbol.insertar(100);
		arbol.insertar(200);
		arbol.insertar(700);
		arbol.insertar(50);
		arbol.insertar(75);
		arbol.recorrerEntre();
```

Hemos cambiado el orden de las líneas.

```
	}
}
```

Si ejecutamos podremos observar que la información aparece ordenada:

```
50-75-100-200-400-700-
```

Ejemplo utilizando Post-orden:

Vamos a programar:

```
public class ArbolBinarioPost {
	class Nodo{
		int info;
		Nodo izq, der;
	}

	private Nodo raiz;

	void insertar(int x) {
		Nodo nuevo=new Nodo();
		nuevo.info=x;
		if(raiz==null) {
			raiz=nuevo;
		}else {
			Nodo anterior=null;
			Nodo reco=raiz;
			while(reco!=null) {
				anterior=reco;
				if(x<reco.info) {
					reco=reco.izq;
				}else {
					reco=reco.der;
				}
			}
			if(x<anterior.info) {
				anterior.izq=nuevo;
			}else {
				anterior.der=nuevo;
			}
		}
	}

	private void recorrerPost(Nodo reco) {
		if(reco!=null) {
			recorrerPost(reco.izq);
			recorrerPost(reco.der);
			System.out.print(reco.info+"-");
		}
	}

	public void recorrerPost() {
		recorrerPost(raiz);
		System.out.println();
	}

	public static void main(String[] args) {
		ArbolBinarioPost arbol=new ArbolBinarioPost();
		arbol.insertar(400);
```

Hemos cambiado el orden de las líneas.

```
            arbol.insertar(100);
            arbol.insertar(200);
            arbol.insertar(700);
            arbol.insertar(50);
            arbol.insertar(75);
            arbol.recorrerPost();
        }
}
```

Si ejecutamos este será el resultado:

```
75-50-200-100-700-400-
```

Capítulo 163.- Estructuras dinámicas: Implementación en Java de un árbol binario ordenado – 2

Confeccionar una clase que permita insertar un entero en un árbol binario ordenado verificando que no se encuentre previamente dicho número.

Desarrollar los siguientes métodos:

1. Retornar la cantidad de nodos del árbol.
2. Retornar la cantidad de nodos hojas del árbol.
3. Imprimir en entre orden.
4. Imprimir en entre orden junto al nivel donde se encuentra dicho nodo.
5. Retornar la altura del del árbol.
6. Imprimir el mayor valor del árbol.
7. Borrar el nodo menor del árbol.

Vamos a programar.

```
public class ArbolBinarioOrd {
	// Creamos la extrucutra Nodo
	class Nodo{
		int info;
		Nodo izq, der;
	}
	// Cramos puntero fuera del arbol
	private Nodo raiz;
	private int cantidad;
	private int altura;
	private int valor;

	// Método existe
	public boolean existe(int x) {
		Nodo reco=raiz;
		while(reco!=null) {
			if (reco.info==x) {
				return true;
			}else {
				if(x>reco.info) {
					reco=reco.der;
				}else {
					reco=reco.izq;
				}
			}
		}
		return false;
	}

	// Método insertar
	public void insertar(int x) {
		if(!existe(x)) {
			Nodo nuevo=new Nodo();
			nuevo.info=x;
			if (raiz==null) {
				raiz=nuevo;
			}else {
				Nodo anterior=null;
				Nodo reco=raiz;
```

```
                while(reco!=null) {
                        anterior=reco;
                        if(x>reco.info) {
                                reco=reco.der;
                        }else {
                                reco=reco.izq;
                        }
                }
                if(x>anterior.info) {
                        anterior.der=nuevo;
                }else {
                        anterior.izq=nuevo;
                }
        }
    }
}

// El método cursivo.
private void recorrerEntre(Nodo reco) {
    if(reco!=null) {
        recorrerEntre(reco.izq);
        System.out.print(reco.info+"-");
        recorrerEntre(reco.der);
    }
}

// Imprimir en etre orden
public void recorrerEntre() {
    recorrerEntre(raiz);
    System.out.println();
}

// El método recursivo.
private void retornarCantidad(Nodo reco) {
    if(reco!=null) {
        cantidad++;
        retornarCantidad(reco.izq);
        retornarCantidad(reco.der);
    }
}

// Retornar la cantidad de nodos
public int retornarCantidad() {
    cantidad=0;
    retornarCantidad(raiz);
    return cantidad;
}

// El método recursivo
private void retornarCantidadHojas(Nodo reco) {
    if (reco!=null) {
        if (reco.izq==null && reco.der==null) {
            cantidad++;
        }
        retornarCantidadHojas(reco.izq);
        retornarCantidadHojas(reco.der);
    }
}
```

```
//Retornar cantidad de nodos hoja
public int retornarCantidadHojas() {
	cantidad=0;
	retornarCantidadHojas(raiz);
	return cantidad;
}

// El método recursivo.
private void imprimirEntreConNivel(Nodo reco, int nivel) {
	if (reco!=null) {
		imprimirEntreConNivel(reco.izq, nivel+1);
		System.out.print(reco.info+" ("+nivel+") - ");
		imprimirEntreConNivel(reco.der, nivel+1);
	}
}

// Imprimir en etre orden
public void imprimirEntreConNivel() {
	imprimirEntreConNivel(raiz,1);
	System.out.println();
}

//El método recursivo.
private void retornarAltura(Nodo reco, int nivel) {
	if(reco!=null) {
		if(nivel>altura) {
			altura=nivel;
		}
		retornarAltura(reco.izq, nivel+1);
		retornarAltura(reco.der, nivel+1);
	}
}

// Altura del árbol
public int retornarAltura() {
	altura=0;
	retornarAltura(raiz,1);
	return altura;
}

//Mayor valor del arbol.
public void mayorValor() {
	if (raiz!=null) {
		Nodo reco=raiz;
		while(reco.der!=null) {
			reco=reco.der;
		}
		System.out.println("Mayor del árbol: "+reco.info);
	}
}

//Borrar el menor del arbol.
public void borrarMenor() {
	if (raiz!=null) {
		if(raiz.izq==null) {
			raiz=raiz.der;
		}else {
			Nodo atras=raiz;
			Nodo reco=raiz.izq;
```

```
                        while(reco.izq!=null) {
                                atras=reco;
                                reco=reco.izq;
                        }
                        atras.izq=reco.der;

                }
        }
    }

    public static void main(String[] args) {
            ArbolBinarioOrd arbol=new ArbolBinarioOrd();
            arbol.insertar(400);
            arbol.insertar(200);
            arbol.insertar(100);
            arbol.insertar(250);
            arbol.insertar(700);
            arbol.insertar(700);
            arbol.recorrerEntre();
            System.out.println("La cantidad de nodos del árbol:
"+arbol.retornarCantidad());
            System.out.println("La cantidad de nodos hojas del árbol:
"+arbol.retornarCantidadHojas());
            arbol.imprimirEntreConNivel();
            System.out.println("La altura del árbol tiene un nivel de
"+arbol.retornarAltura());
            arbol.mayorValor();
            arbol.borrarMenor();
            arbol.recorrerEntre();
    }
}
```

Si ejecutamos este será el resultado:

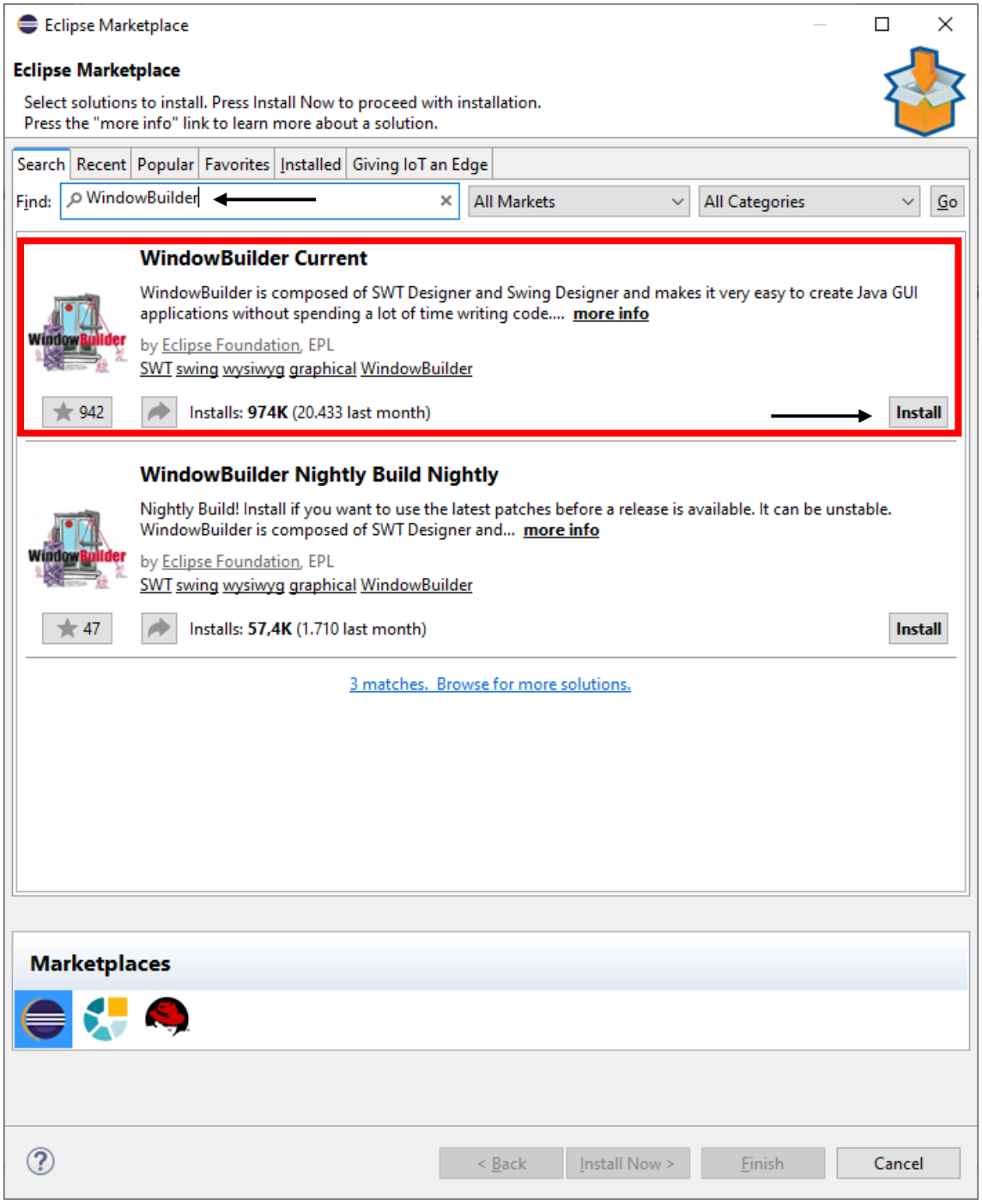
```
100-200-250-400-700-
La cantidad de nodos del árbol: 5
La cantidad de nodos hojas del árbol: 3
100 (3) - 200 (2) - 250 (3) - 400 (1) - 700 (2) -
La altura del árbol tiene un nivel de 3
Mayor del árbol: 700
200-250-400-700-
```

Capítulo 164.- Plug-in WindowBuilder para crear interfaces visuales – 1

Es una herramienta que nos permite generar los formularios en forma visual, arrastrando los objetos.

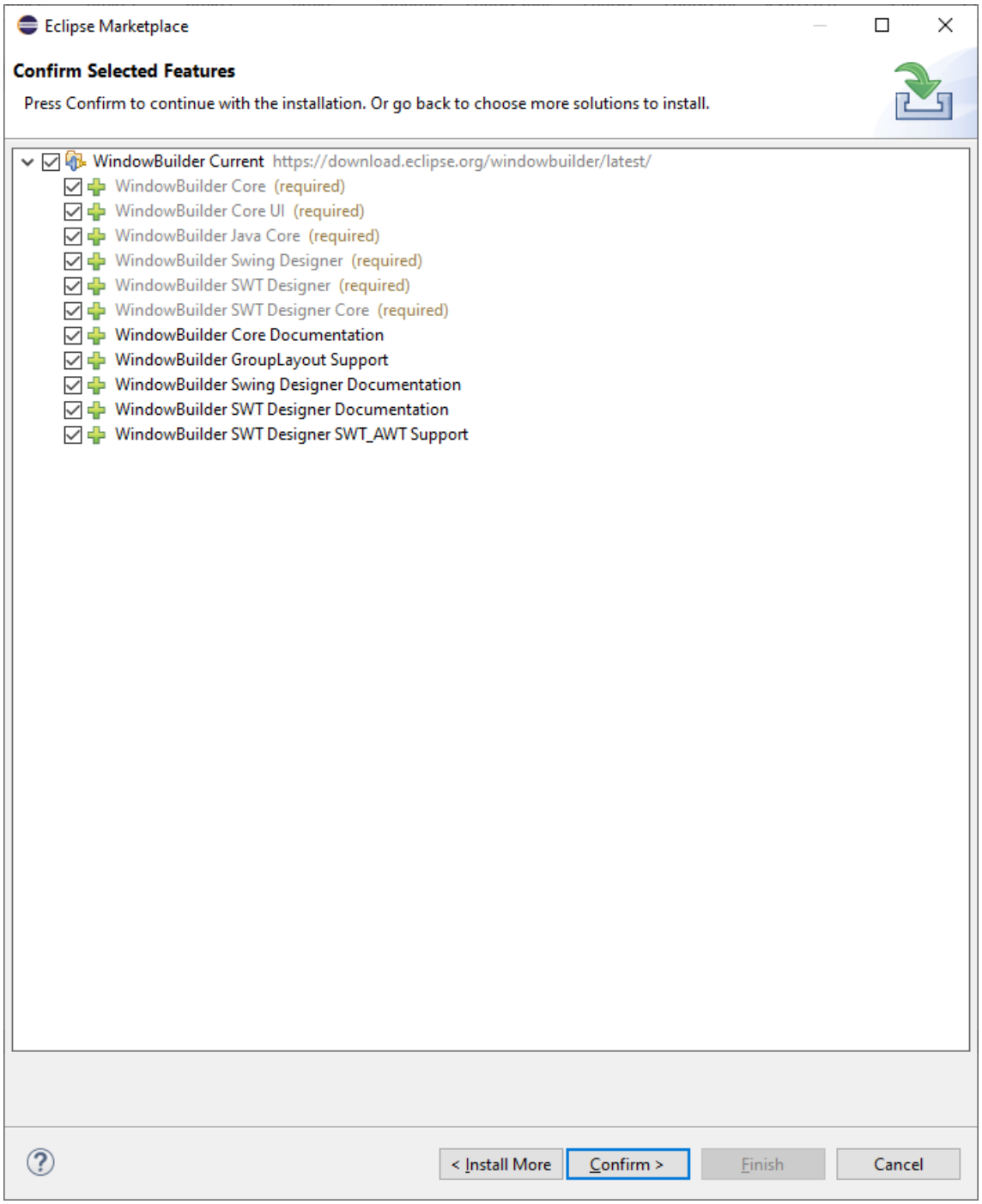
Vamos a ver como instalar WindowBuilder en la versión para Eclipse:

Del menú seleccionaremos Help y de este Eclipse Markeplace…

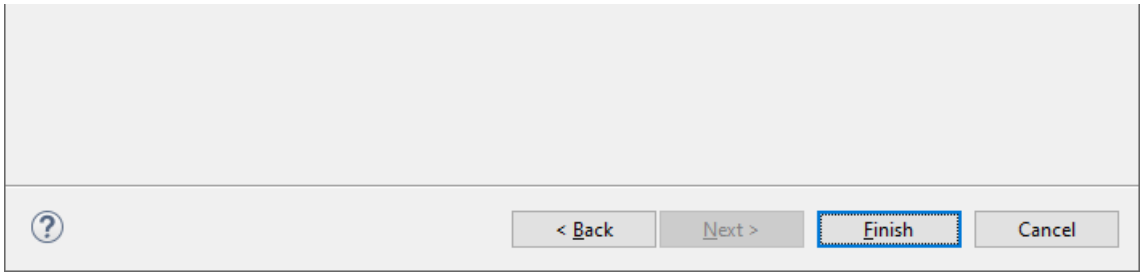


Buscaremos por WindowBuilder y instalaremos WindowBuilder Current.

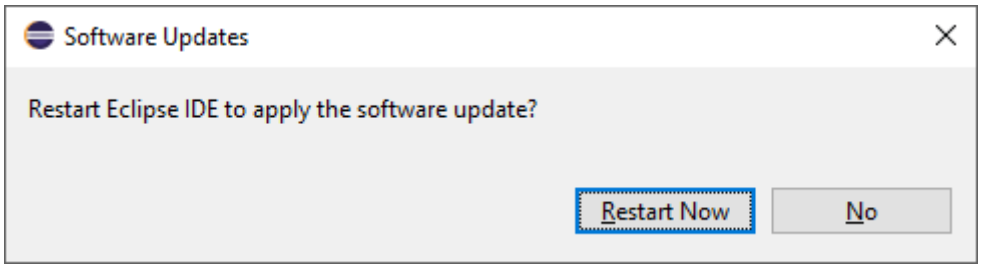
Le damos al botón Install.



Vamos a instalar todo lo que nos propone y le damos al botón Confirm.



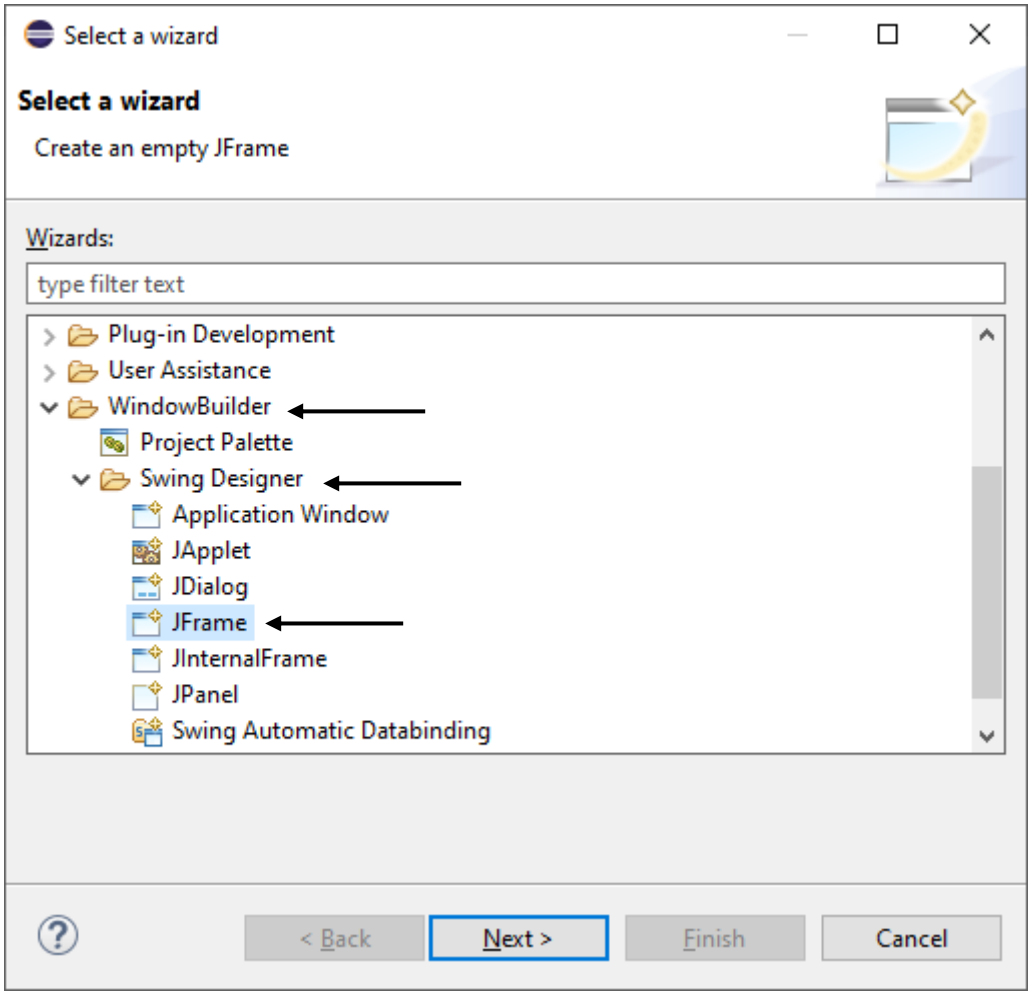
Por último le damos a Finish.



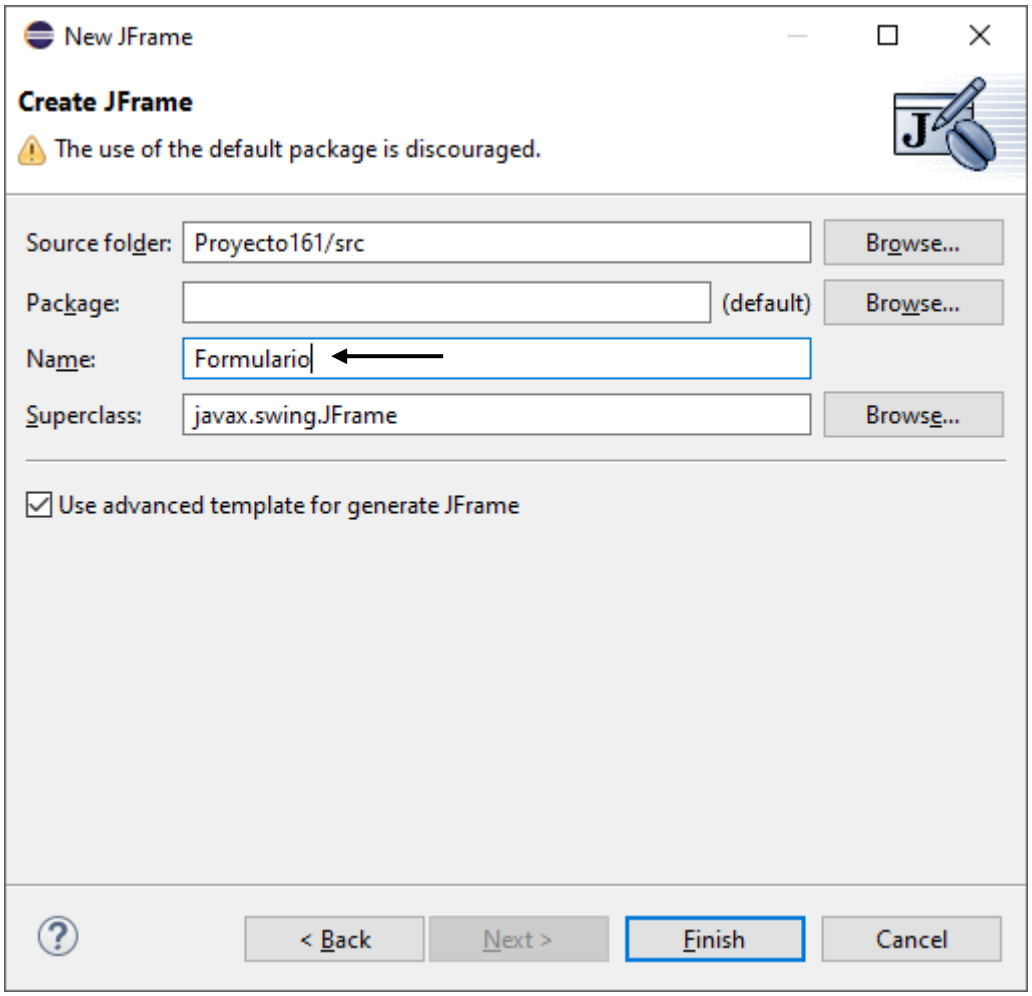
Nos pide que tenemos que reiniciar el eclipse, le daremos a Restart Now.

Vamos a crear un nuevo proyecto.

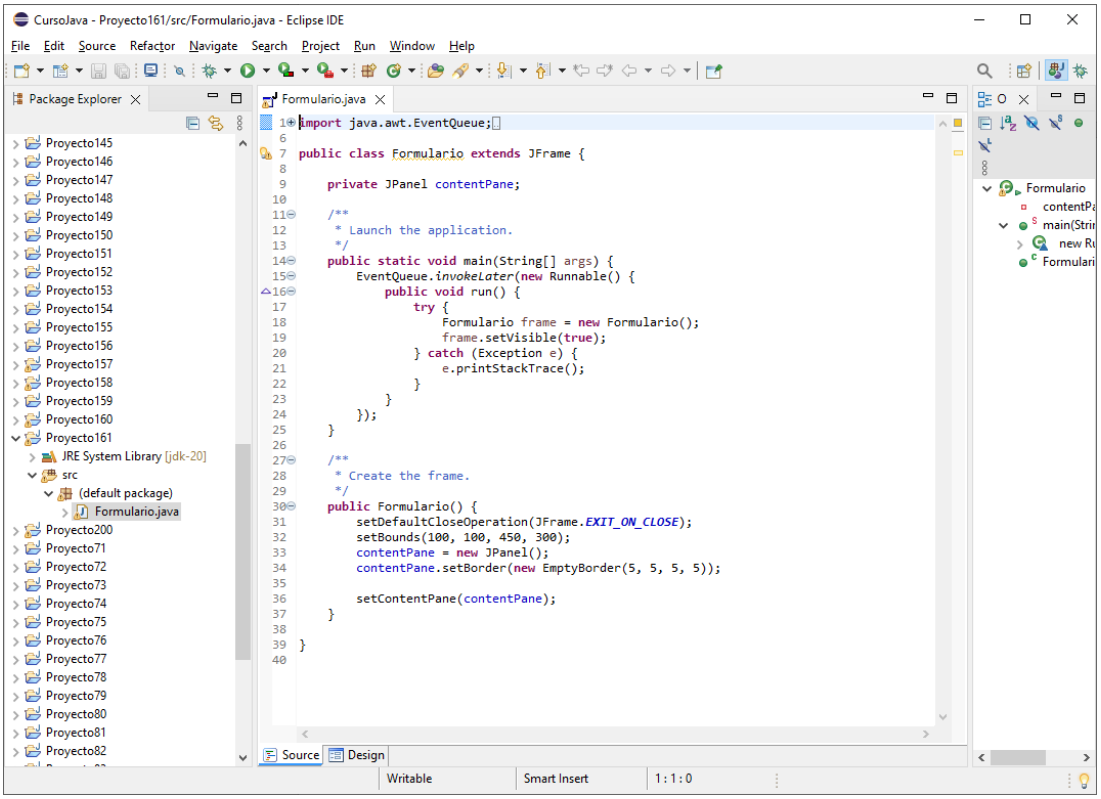
Ahora del menú File seleccionaremos New y de este Other.



Seleccionamos WindowBuilder, seguido de la carpeta Swing Designer y a continuación JFrame, seguido del botón Next.

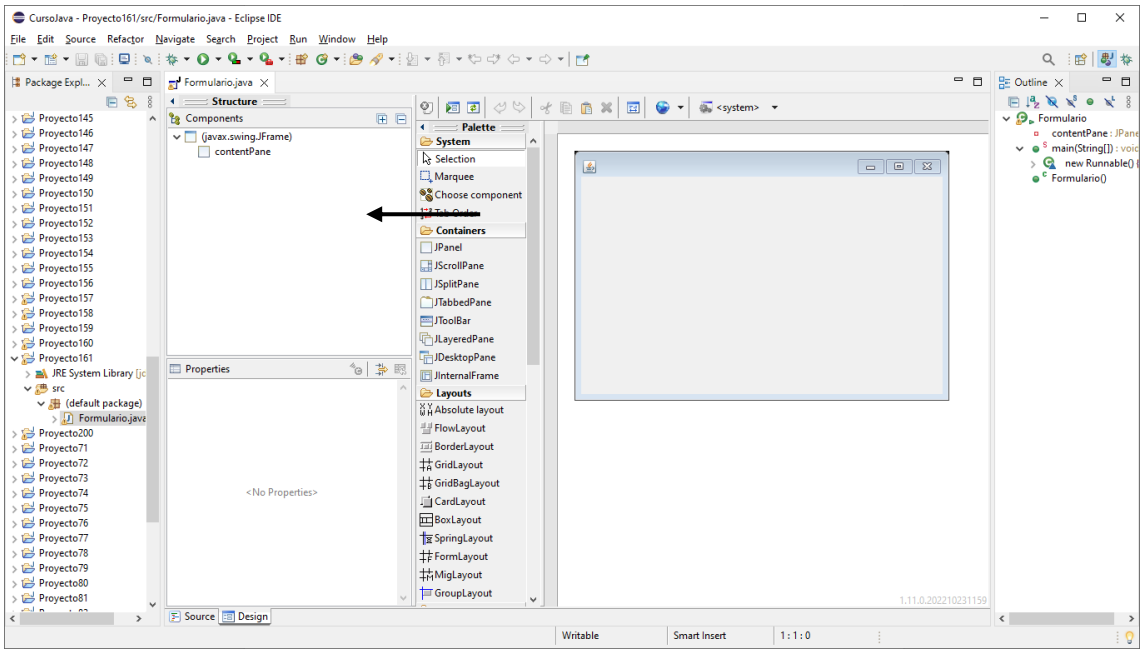


Como nombre ponemos Formulario seguido del botón Finish.

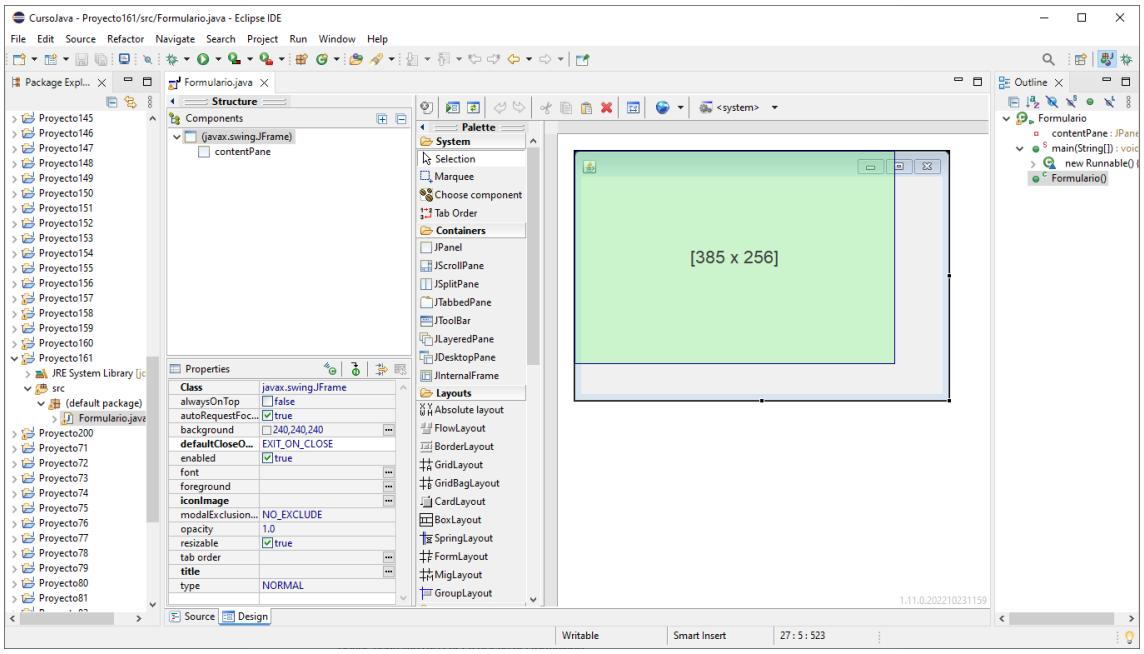


Ya nos ha generado un pequeño código.

En la parte inferior observamos una nueva pestaña, llamada Design.

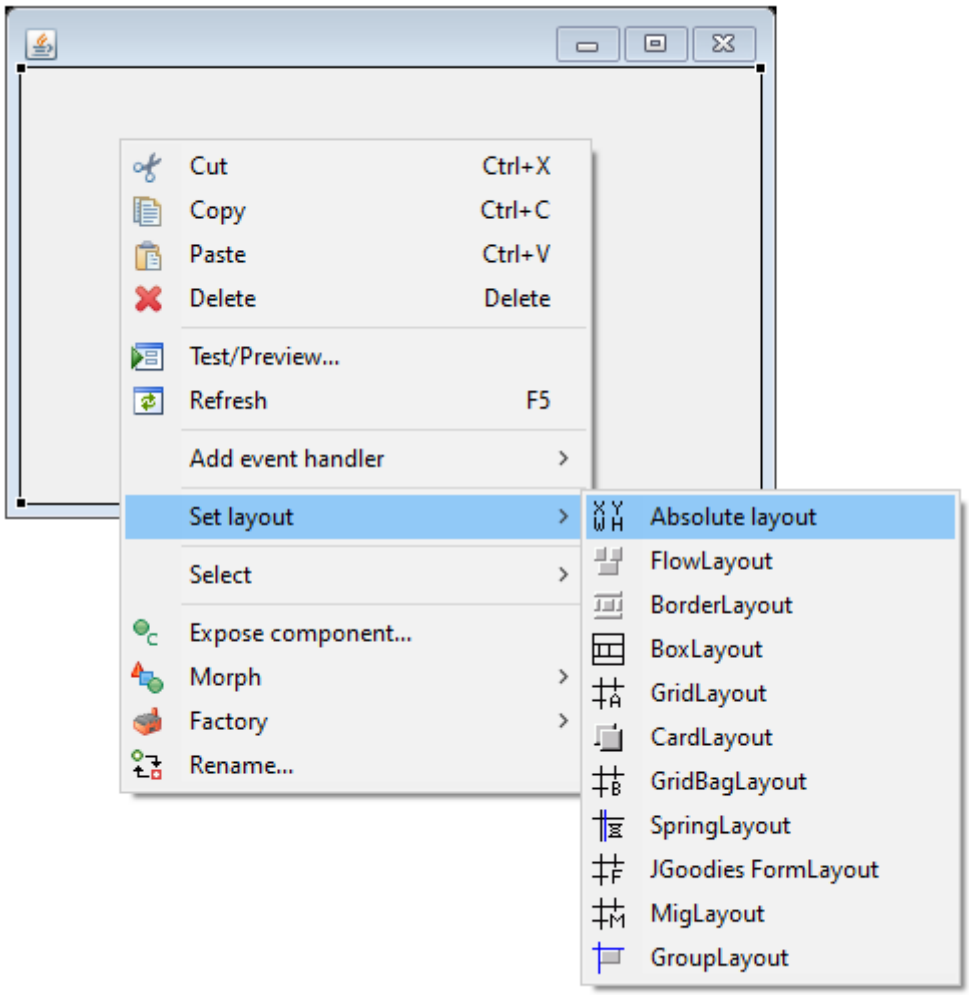


Donde podremos acceder a diseño del formulario.



Si modificamos el tamaño del formulario, este también se modificara en el código del constructor.

```
public Formulario() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 385, 256);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

    setContentPane(contentPane);
}
```

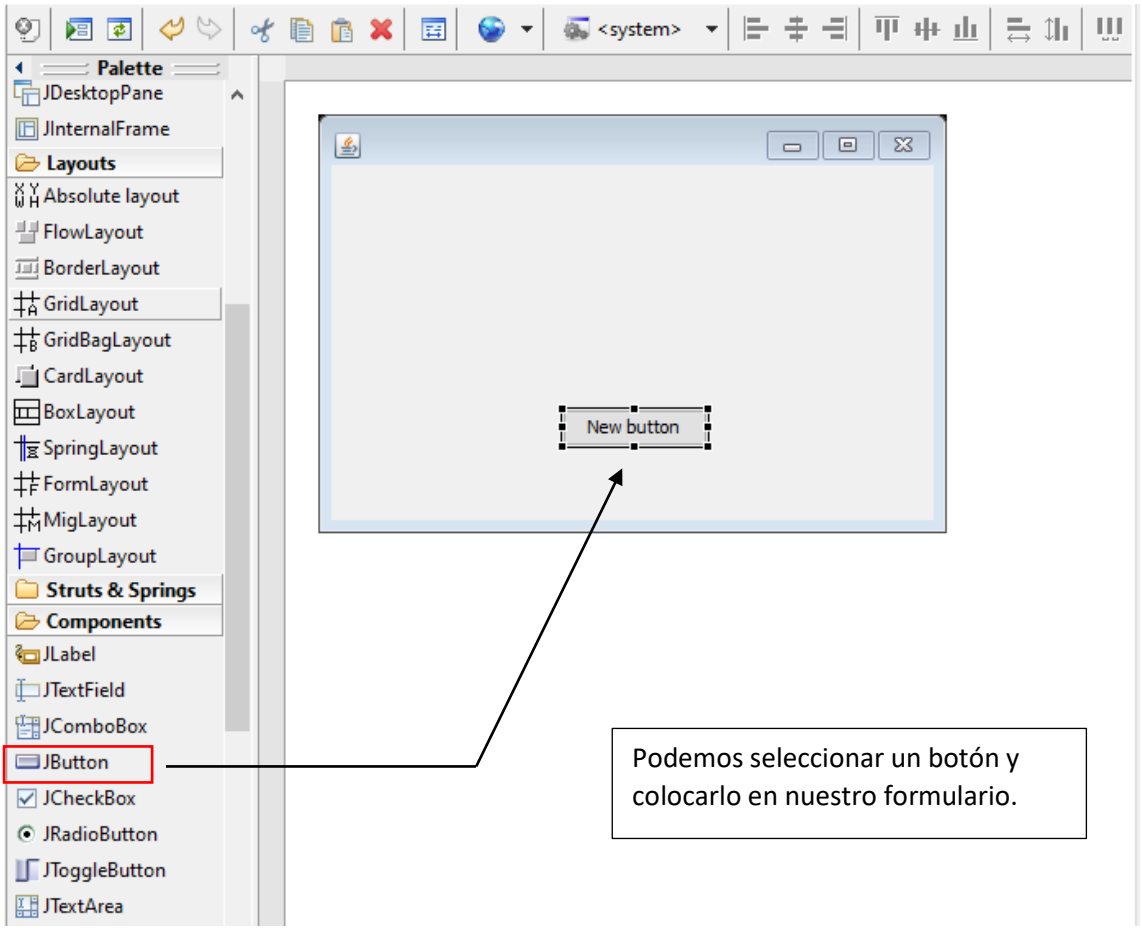


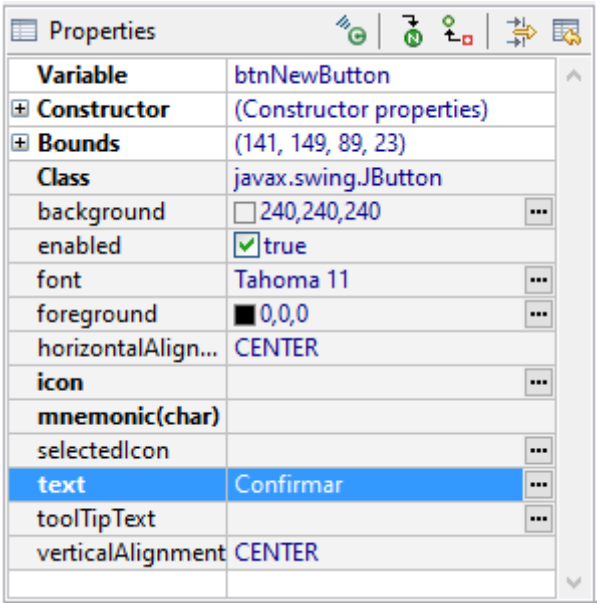
Si seleccionamos con el botón derecho del ratón sobre el formulario y del menú seleccionamos Set layout y de este Absolute layout, obsvaremos que este cambio quedará reflejado en nuestro código.

```
public Formulario() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 384, 256);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
```

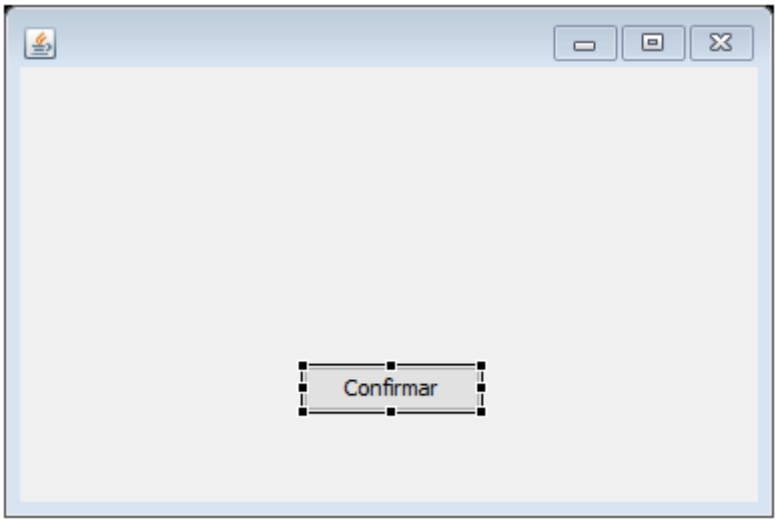
```
        setContentPane(contentPane);
        contentPane.setLayout(null);
    }
```

De este modo podemos posicionar nuestros componentes absolutamente. Para esto, usábamos el método setBounds().





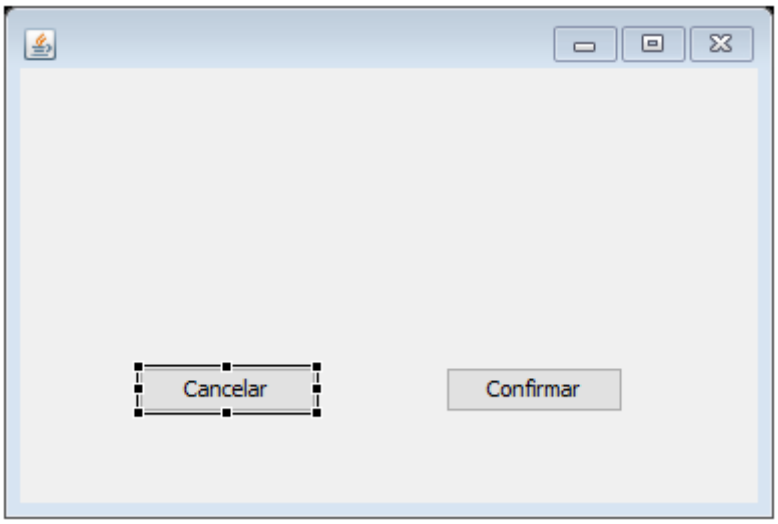
En la ventana de propiedades podemos cambiar el texto del botón.



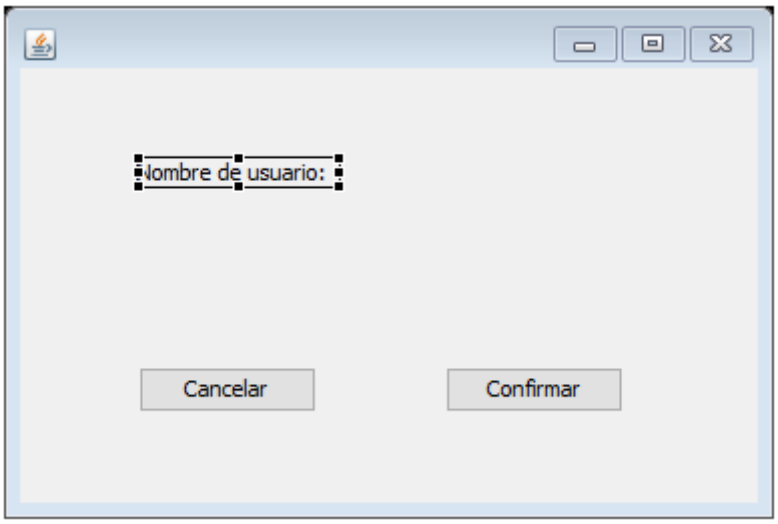
También se ha modificado el código:

```
public Formulario() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 384, 256);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

    setContentPane(contentPane);
    contentPane.setLayout(null);

    JButton btnNewButton = new JButton("Confirmar");
    btnNewButton.setBounds(141, 149, 89, 23);
    contentPane.add(btnNewButton);
}
```

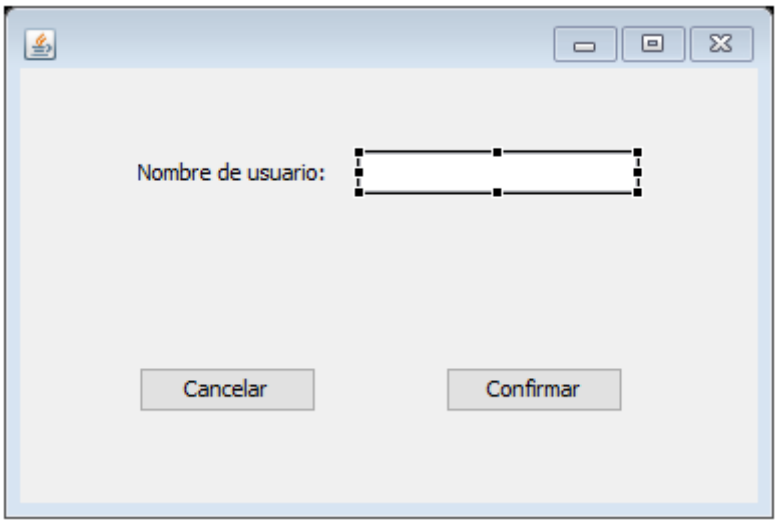
Ahora vamos a agregar otro botón que ponga Cancelar.



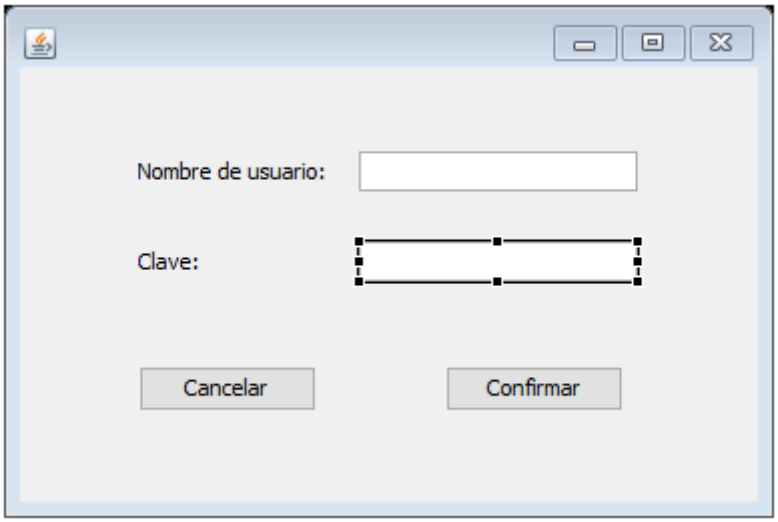
Vamos a agregar una etiqueta que ponga nombre de usuario:



Vamos a añadir una caja de texto:

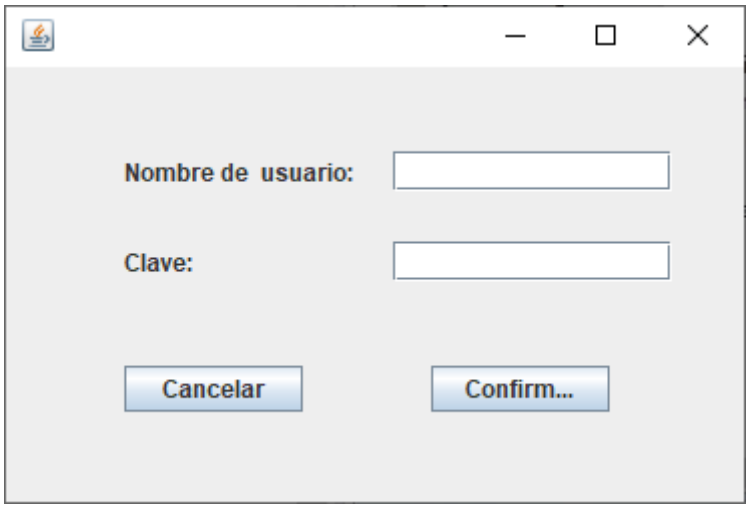


Vamos a agregar una nueva etiqueta que ponga clave y una nueva caja de texto.



Se pueden modificar la posición de cualquier objeto con mucha facilidad.

Si ejecutamos como siempre podemos ver que nuestra interface gráfica ya está creada.



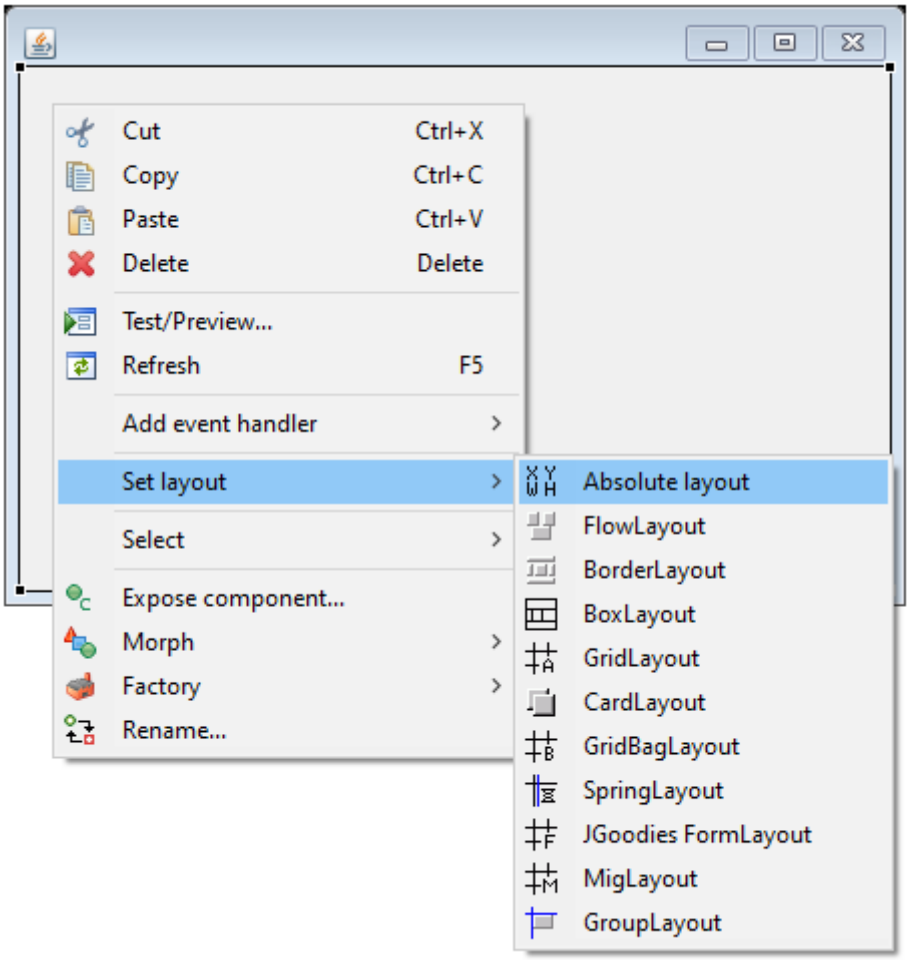
Nos quedaría pendiente la programación lógica para que este funcione.

Capítulo 165.- Plug-in WindowBuilder para crear interfaces visuales – 2

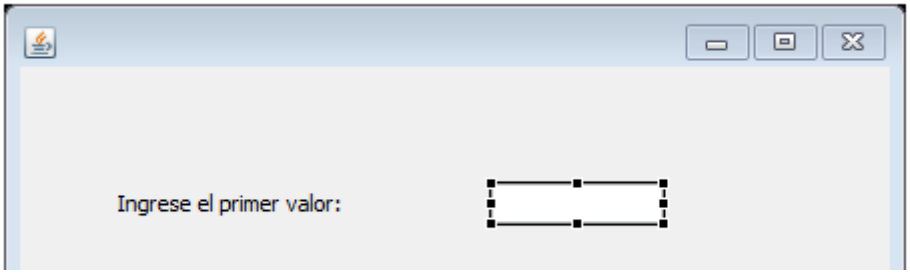
Captura de eventos.

Vamos a realizar un nuevo proyecto utilizando como en el ejemplo anterior la clase WindowBuilder.

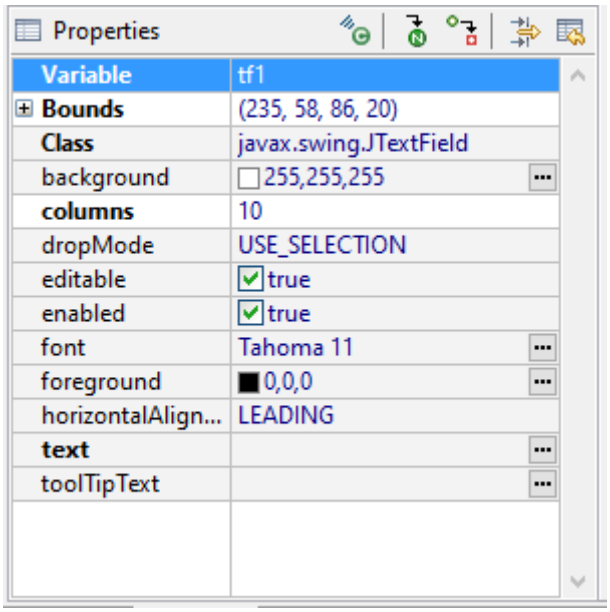
En este ejemplo vamos a realizar un interfaz gráfico que se puedan introducir dos botones y muestre su suma.



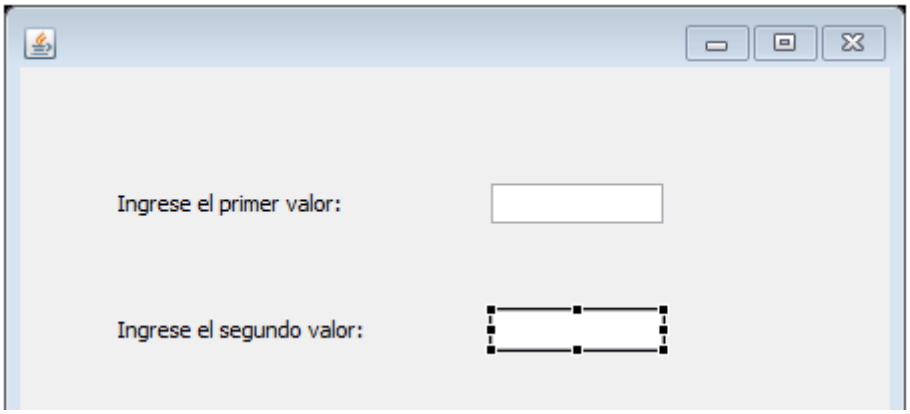
Selecciona Absolute layout para tener libertad de movimiento con los objetos que vayamos a añadir en nuestro formulario.



Ahora quiero cambiar el nombre del JTextField.

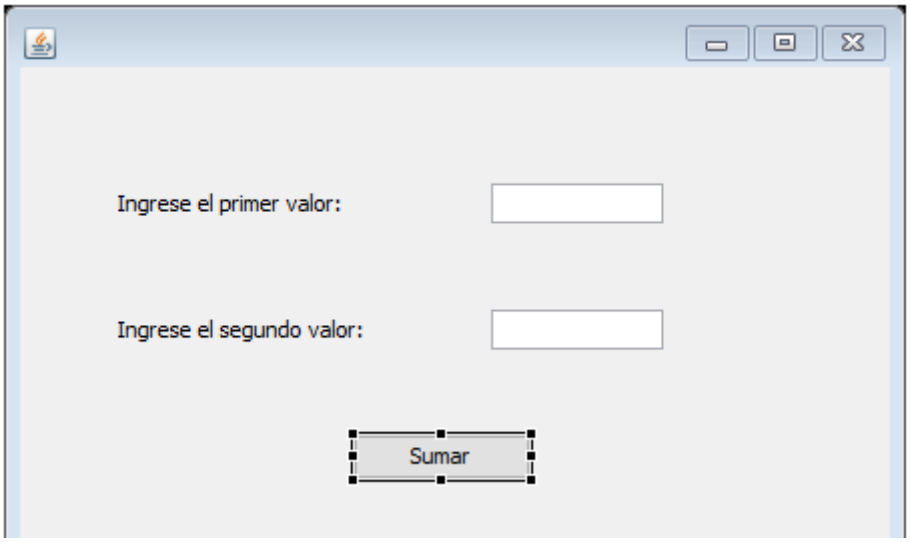


En propiedades en el apartado Variable le podemos cambiar el nombre a tf1.



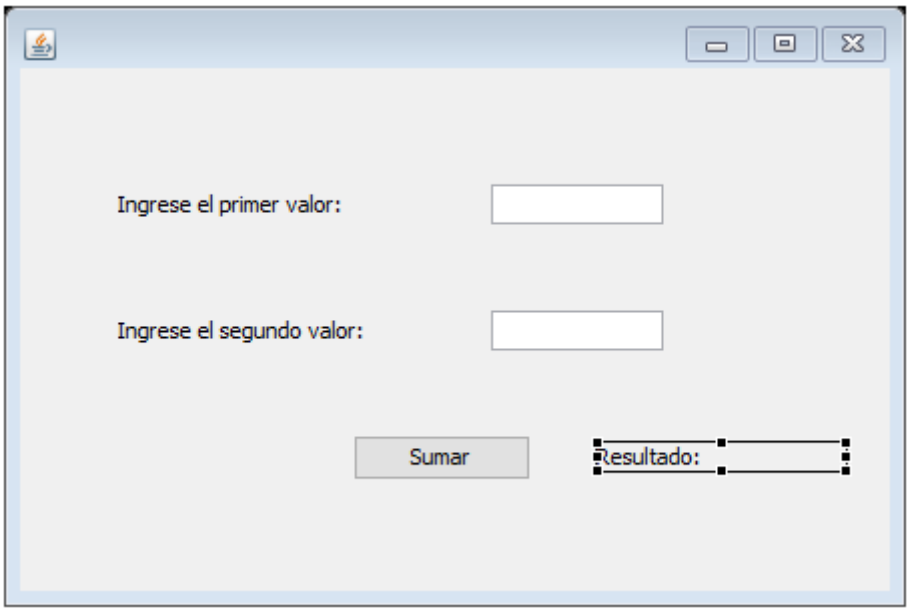
En el segundo JTextField la variable tiene que ser tf2.

Vamos a agregar un botón que diga Sumar.



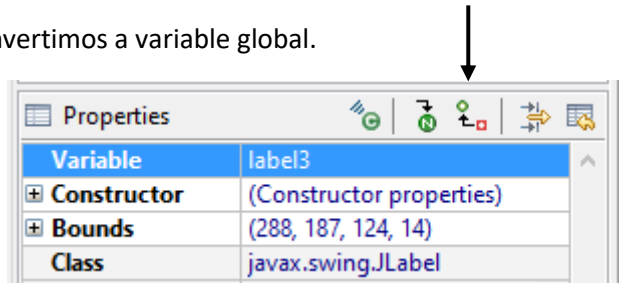
En Propiedades en el apartado text cambiaremos el texto del botón a Sumar.

ahora vamos a agregar un JLabel que diga Resultado: .



Como variable le pondremos label3.

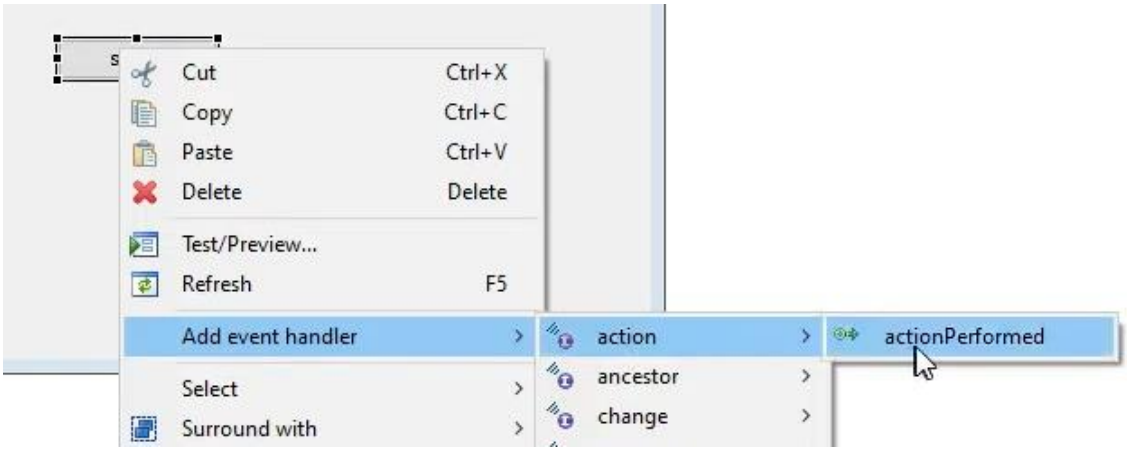
A continuación la convertimos a variable global.




```
public class Formulario extends JFrame {

    private JPanel contentPane;
    private JTextField tf1;
    private JTextField tf2;
    private JLabel label3;
```

Ya la tenemos como variable global.

Ahora para definir el evento seleccionaremos con el botón derecho del ratón el botón, del menú que aparecerá seleccionaremos Add evento handler de este action y por último actionPerformed.



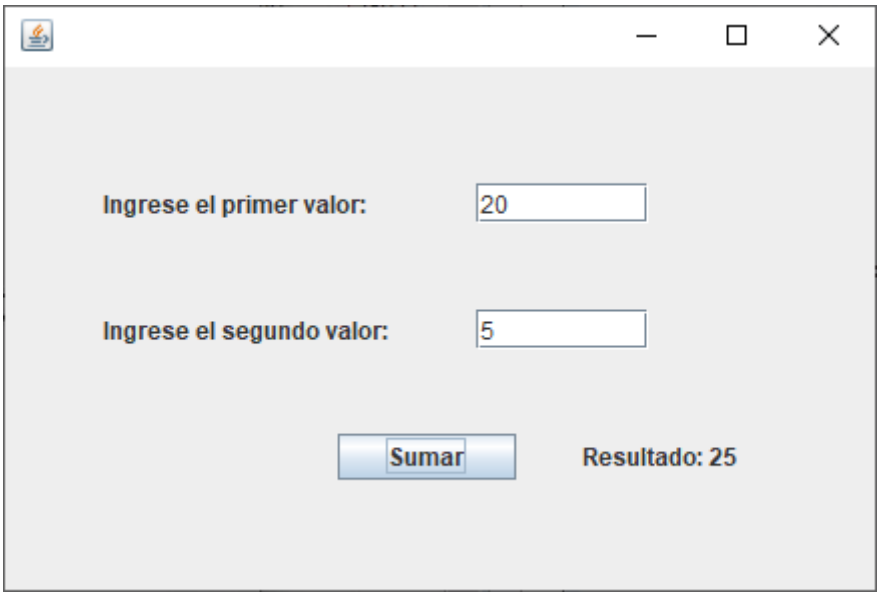
A generado el siguiente código:

```
JButton btnNewButton = new JButton("Sumar");
btnNewButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
    }
```

Ahora tenemos que añadir el código.

```
JButton btnNewButton = new JButton("Sumar");
btnNewButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String s1=tf1.getText();
        String s2=tf2.getText();
        int v1=Integer.parseInt(s1);
        int v2=Integer.parseInt(s2);
        int suma=v1+v2;
        label3.setText("Resultado: "+suma);
    }
```

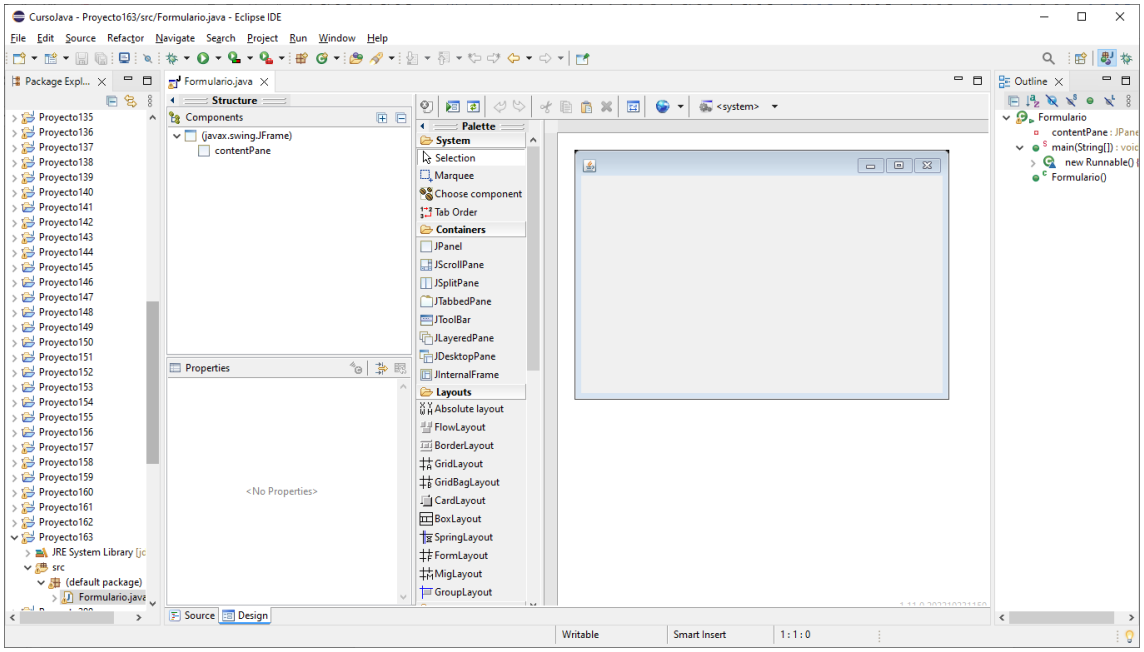
Ahora vamos a ejecutar:



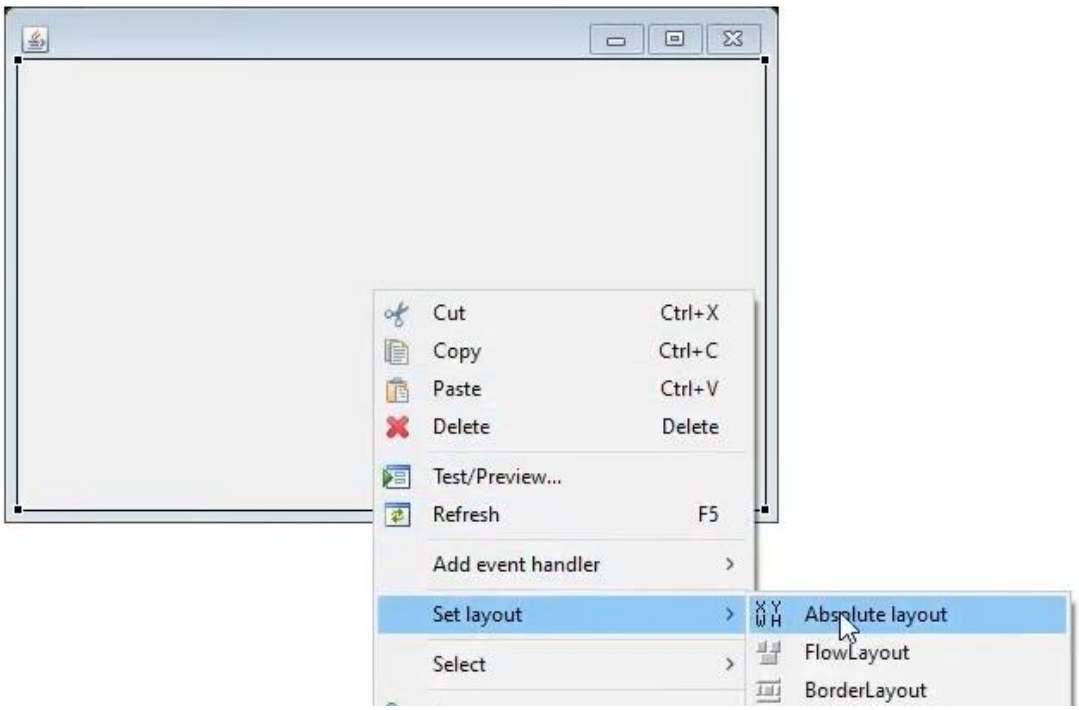
Capítulo 166.- Plug-in WindowBuilder para crear interfaces visuales – 3

En este capítulo vamos a crear un menú de opciones.

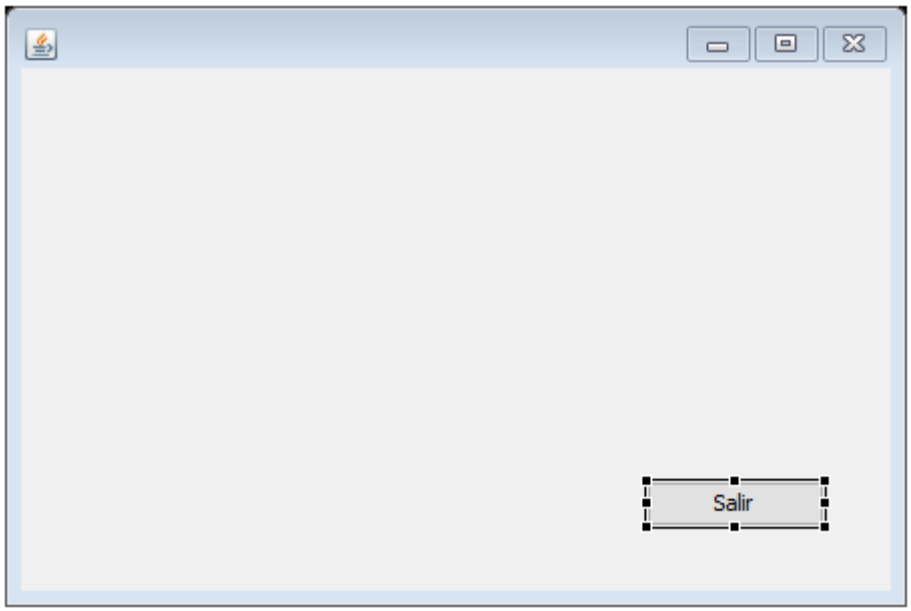
Vamos a crear un nuevo proyecto:



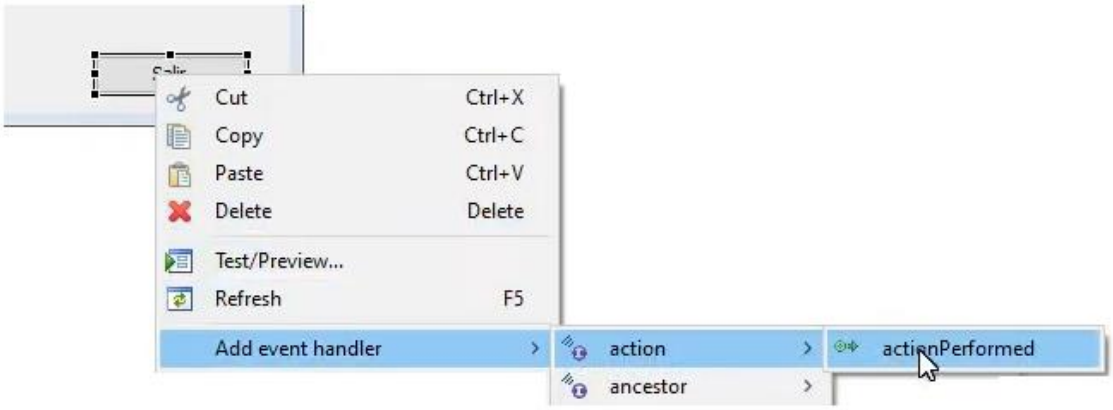
Vamos a cambiar el layaut del JPanel.



Vamos a agregar un botón que diga Salir.



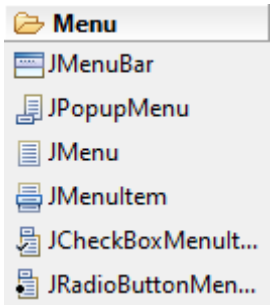
Seleccionaremos el botón con el botón derecho del ratón.



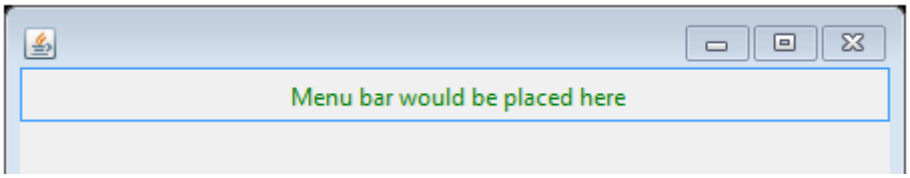
Agregamos el siguiente código:

```
JButton btnNewButton = new JButton("Salir");
btnNewButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
});
```

Si lo ejecutamos podremos finalizar haciendo clic sobre el botón.
Ahora vamos a realizar un menú de opciones con los siguientes componentes:



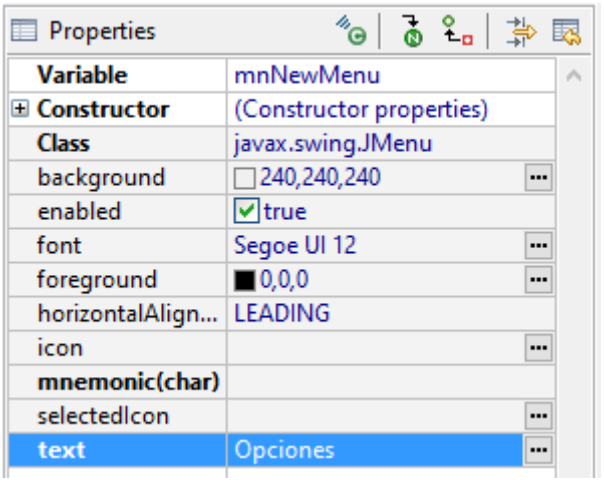
Primero vamos a seleccionar un solo JMenurBar.



Lo soltaremos cuando se muestre la barra de Menu bar.

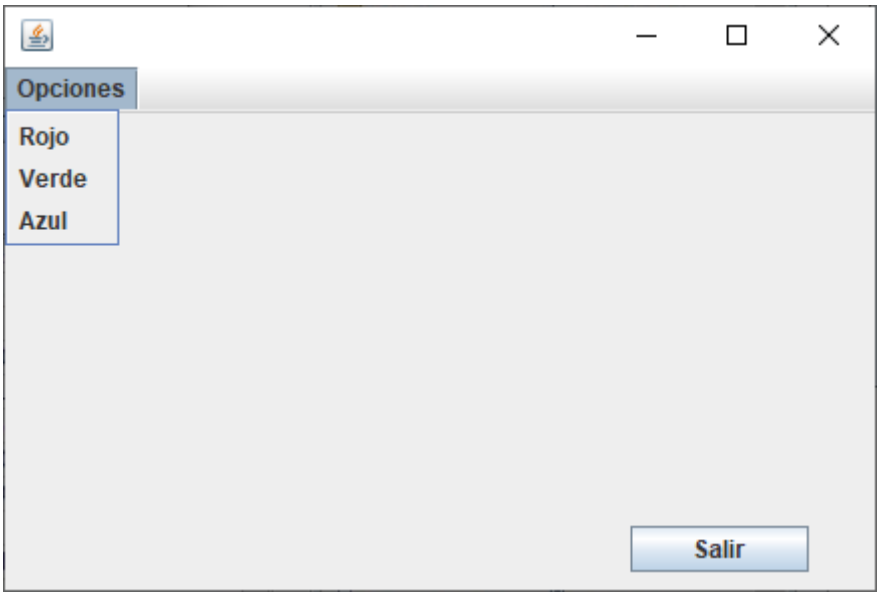
ahora seleccionaremos el JMenu y lo arrastraremos.

En propiedades en text cambiamos el nombre.

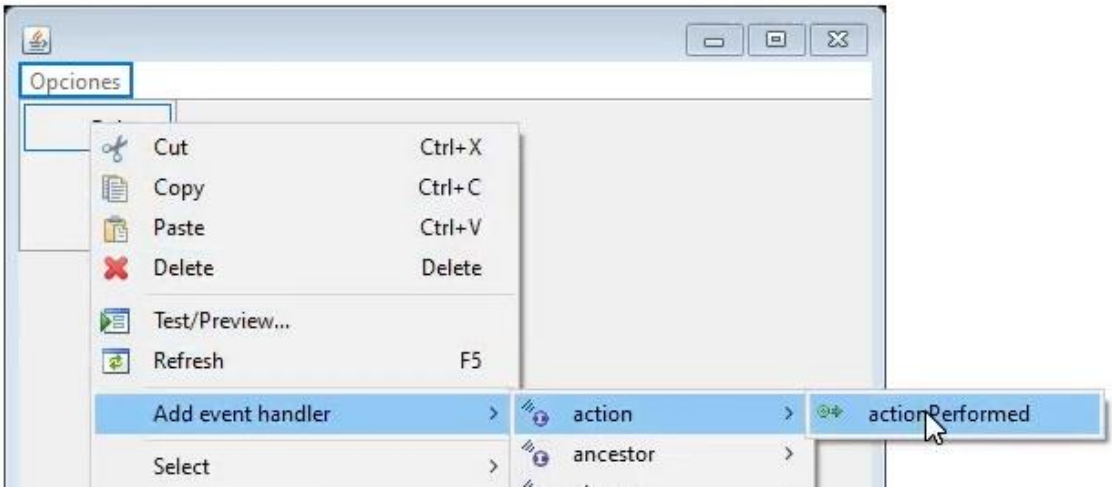


Ahora vamos a agregar 3 JMenuItem en Add ítems here que tienen que poner Rojo, Verde y Azul.

Si ejecutamos veremos que ya tenemos los menús.



Ahora vamos a crear los eventos.



Hay que hacerlo con cada una de las opciones.

```
JMenuItem mntmNewMenuItem = new JMenuItem("Rojo");
mntmNewMenuItem.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
    }
});
mnNewMenu.add(mntmNewMenuItem);

JMenuItem mntmNewMenuItem_1 = new JMenuItem("Verde");
mntmNewMenuItem_1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
    }
});
mnNewMenu.add(mntmNewMenuItem_1);
```

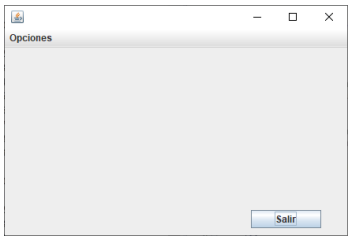
```
JMenuItem mntmNewMenuItem_2 = new JMenuItem("Azul");
mntmNewMenuItem_2.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
    }
});
```

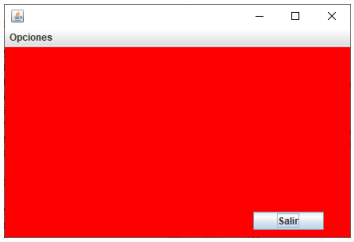
Ha generado el correspondiente código.

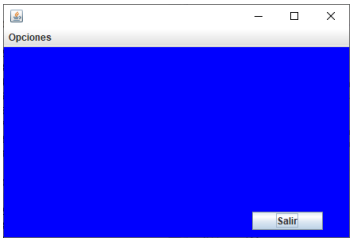
Le decimos el color que le tiene que asignar a JPane.

```
JMenuItem mntmNewMenuItem = new JMenuItem("Rojo");
mntmNewMenuItem.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        contentPane.setBackground(Color.red);
    }
});
mnNewMenu.add(mntmNewMenuItem);

JMenuItem mntmNewMenuItem_1 = new JMenuItem("Verde");
mntmNewMenuItem_1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        contentPane.setBackground(Color.green);
    }
});
mnNewMenu.add(mntmNewMenuItem_1);

JMenuItem mntmNewMenuItem_2 = new JMenuItem("Azul");
mntmNewMenuItem_2.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        contentPane.setBackground(Color.blue);
    }
});
```

Vamos a ejecutar:







Capítulo 167.- Plug-in WindowBuilder problemas resueltos – 1

Problema:

Desarrollar un programa que muestre el tablero de un ascensor:

Este será el código:

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JButton;
import javax.swing.JLabel;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.SwingConstants;

public class Formulario extends JFrame {

	private JPanel contentPane;
	private JLabel lblPiso;
	private JLabel lblDireccion;
	private int piso = 1;
	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Formulario frame = new Formulario();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
```

```
		});
	}

	/**
	 * Create the frame.
	 */
	public Formulario() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 450, 300);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

		setContentPane(contentPane);
		contentPane.setLayout(null);

		JButton btn4 = new JButton("4");
		btn4.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				int level=4;
				if (level>piso) {
					lblDireccion.setText("Sube");
				}else {
					if(level==piso) {
						lblDireccion.setText("Piso actual");
					}else {
						lblDireccion.setText("baja");
					}
				}
				lblPiso.setText("4");
				piso=4;
			}
		});
		btn4.setBounds(39, 11, 53, 49);
		contentPane.add(btn4);

		JButton btn3 = new JButton("3");
		btn3.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				int level=3;
				if (level>piso) {
					lblDireccion.setText("Sube");
				}else {
					if(level==piso) {
						lblDireccion.setText("Piso actual");
					}else {
						lblDireccion.setText("baja");
					}
				}
				lblPiso.setText("3");
				piso=3;
			}
		});
		btn3.setBounds(39, 71, 53, 49);
		contentPane.add(btn3);

		JButton btn2 = new JButton("2");
		btn2.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				int level=2;
```

```
                        if (level>piso) {
                                lblDireccion.setText("Sube");
                        }else {
                                if(level==piso) {
                                        lblDireccion.setText("Piso actual");
                                }else {
                                        lblDireccion.setText("baja");
                                }
                        }
                        lblPiso.setText("2");
                        piso=2;
                }
        });
        btn2.setBounds(39, 130, 53, 47);
        contentPane.add(btn2);

        JButton btn1 = new JButton("1");
        btn1.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                        int level=1;
                        if (level>piso) {
                                lblDireccion.setText("Sube");
                        }else {
                                if(level==piso) {
                                        lblDireccion.setText("Piso actual");
                                }else {
                                        lblDireccion.setText("baja");
                                }
                        }
                        lblPiso.setText("1");
                        piso=1;
                }
        });
        btn1.setBounds(39, 188, 53, 49);
        contentPane.add(btn1);

        JLabel lblNewLabel = new JLabel("Piso: ");
        lblNewLabel.setBounds(128, 82, 46, 14);
        contentPane.add(lblNewLabel);

        JLabel lblNewLabel_1 = new JLabel("Direcci\u00F3n: ");
        lblNewLabel_1.setBounds(128, 118, 68, 14);
        contentPane.add(lblNewLabel_1);

        lblPiso = new JLabel("1");
        lblPiso.setHorizontalAlignment(SwingConstants.RIGHT);
        lblPiso.setBounds(208, 82, 46, 14);
        contentPane.add(lblPiso);

        lblDireccion = new JLabel("");
        lblDireccion.setBounds(206, 118, 115, 14);
        contentPane.add(lblDireccion);
    }
}
```

Capítulo 168.- Plug-in WindowBuilder problemas resueltos – 2

Problema:

Desarrolla un programa que muestre un panel para extracción de una bebida:

Por un lado disponer tres objetos de la clase JRadioButton (llamarlos radio1, radio2 y radio3), configurar el primero para que aparezca seleccionado (propiedad "selected").

Disponer de dos objetos de la clase JComboBox (llamarlos comboPesos y comboCentavos).

En el JComboBox pesos inicializar la propiedad model con los valores del 0 al 5 (hay que cargar un valor por cada línea en el diálogo que aparece).

En forma similar el segundo JComboBox cargamos los valores: 0, 10, 20, 30 etc. hasta 90.

Se sabe que:

Bebida A tiene un costo de 0 pesos 80 centavos.
Bebida B tiene un costo de 1 peso 20 centavos.
Bebida C tiene un costo de 3 pesos 10 centavos.

Cuando se presiona el botón extraer mostrar en la label de resultado el texto "Correcto" o "Incorrecto" dependiendo de la bebida seleccionada y la cantidad de pesos y centavos seleccionados.

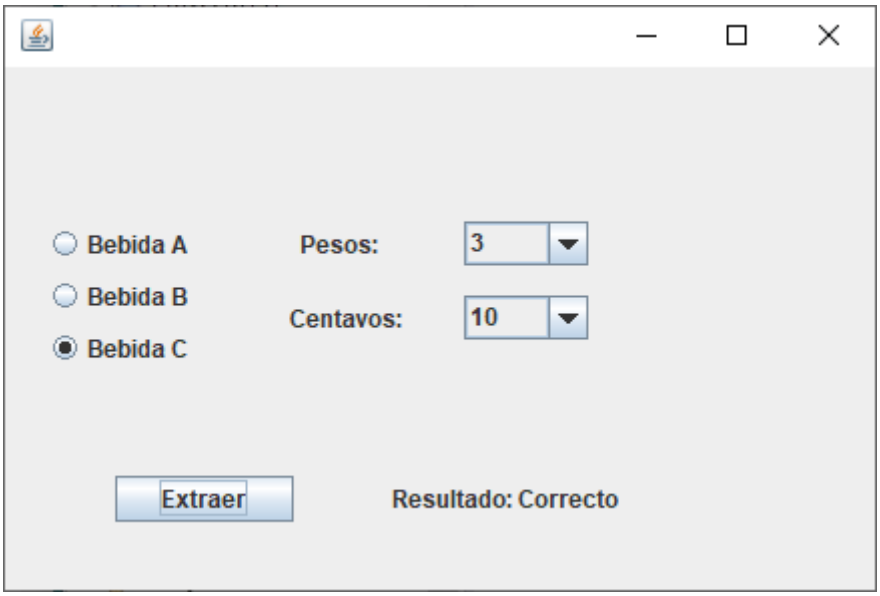
Este será el código:

```
import java.awt.EventQueue;

import javax.swing.ButtonGroup;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JRadioButton;
import javax.swing.JLabel;
import javax.swing.JComboBox;
import javax.swing.DefaultComboBoxModel;
```

```
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class Formulario extends JFrame {

	private JPanel contentPane;
	private ButtonGroup bg;
	private JLabel labelResultado;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Formulario frame = new Formulario();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public Formulario() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 450, 300);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

		setContentPane(contentPane);
		contentPane.setLayout(null);

		JRadioButton radio1 = new JRadioButton("Bebida A");
		radio1.setSelected(true);
		radio1.setBounds(20, 77, 109, 23);
		contentPane.add(radio1);

		JRadioButton radio2 = new JRadioButton("Bebida B");
		radio2.setBounds(20, 103, 109, 23);
		contentPane.add(radio2);

		JRadioButton radio3 = new JRadioButton("Bebida C");
		radio3.setBounds(20, 129, 109, 23);
		contentPane.add(radio3);

		bg=new ButtonGroup();
		bg.add(radio1);
		bg.add(radio2);
		bg.add(radio3);

		JLabel lblNewLabel = new JLabel("Pesos:");
		lblNewLabel.setBounds(147, 81, 75, 14);
		contentPane.add(lblNewLabel);
```

```
		JLabel lblNewLabel_1 = new JLabel("Centavos:");
		lblNewLabel_1.setBounds(142, 118, 80, 14);
		contentPane.add(lblNewLabel_1);

		JComboBox comboPesos = new JComboBox();
		comboPesos.setModel(new DefaultComboBoxModel(new String[] {"0",
"1", "2", "3", "4", "5"}));
		comboPesos.setBounds(229, 77, 62, 22);
		contentPane.add(comboPesos);

		JComboBox comboCentavos = new JComboBox();
		comboCentavos.setModel(new DefaultComboBoxModel(new String[]
{"0", "10", "20", "30", "40", "50", "60", "70", "80", "90"}));
		comboCentavos.setBounds(229, 114, 62, 22);
		contentPane.add(comboCentavos);

		JButton btnNewButton = new JButton("Extraer");
		btnNewButton.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				float precio=0;
				int
pesos=Integer.parseInt(comboPesos.getSelectedItem().toString());
				float
centavos=Integer.parseInt(comboCentavos.getSelectedItem().toString());
				precio=pesos+(centavos/100);
				if(radio1.isSelected() && precio==0.8f) {
					labelResultado.setText("Resultado:
Correcto");
				}else {
					if(radio2.isSelected() && precio==1.20f) {
						labelResultado.setText("Resultado:
Correcto");
					}else {
						if(radio3.isSelected()&&
precio==3.10f) {

	labelResultado.setText("Resultado: Correcto");
						}else {

	labelResultado.setText("Resultado: Incorrecto");
						}
					}
				}

			}
		});
		btnNewButton.setBounds(55, 204, 89, 23);
		contentPane.add(btnNewButton);

		labelResultado = new JLabel("Resultado: ");
		labelResultado.setBounds(193, 208, 195, 14);
		contentPane.add(labelResultado);
	}
}
```

Este será el resultado:


Bebida A
Bebida B
Bebida C
Pesos:
3
Centavos:
10
Extraer
Resultado: Correcto

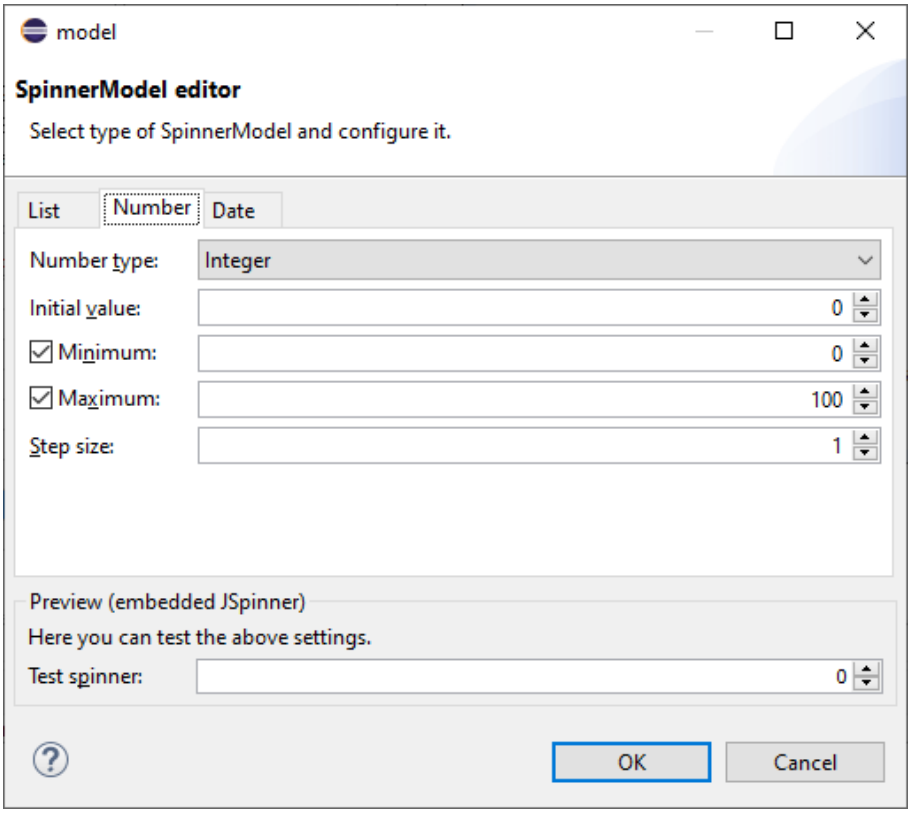
Capítulo 169.- Plug-in WindowBuilder problemas resueltos – 3

Un embalse debe manejar la cantidad de mts^3 de agua que pasa por cada compuerta. Por cada compuerta puede pasar un caudal de 100 mts^3 x seg.

Cuando presionamos el botón "Actualizar caudal" mostramos el nivel de caudal actual y un mensaje que indica si el caudal es Bajo (o a 100 mts^3 x seg.) , Medio (>100 -200 mts^3 x seg.) o Alto (>200 mts^3 seg.)

Para la selección del caudal de cada compuerta utilizar componentes de tipo JSpinner.

Para controlar el valor mínimo y máximo:



Vamos a programar:

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JSpinner;
import javax.swing.JList;
import javax.swing.SpinnerNumberModel;
import javax.swing.JLabel;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class Formulario extends JFrame {

	private JPanel contentPane;
	private JLabel lblResultado;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Formulario frame = new Formulario();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public Formulario() {
		setTitle("Enblase");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 450, 300);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

		setContentPane(contentPane);
		contentPane.setLayout(null);

		JSpinner spinner1 = new JSpinner();
		spinner1.setModel(new SpinnerNumberModel(0, 0, 100, 1));
		spinner1.setBounds(45, 64, 54, 20);
		contentPane.add(spinner1);

		JSpinner spinner2 = new JSpinner();
		spinner2.setModel(new SpinnerNumberModel(0, 0, 100, 1));
		spinner2.setBounds(45, 107, 54, 20);
		contentPane.add(spinner2);
```

```
		JSpinner spinner3 = new JSpinner();
		spinner3.setModel(new SpinnerNumberModel(0, 0, 100, 1));
		spinner3.setBounds(45, 148, 54, 20);
		contentPane.add(spinner3);

		JLabel lblNewLabel = new JLabel("Compuerta 1");
		lblNewLabel.setBounds(135, 67, 117, 14);
		contentPane.add(lblNewLabel);

		JLabel lblNewLabel_1 = new JLabel("Compuerta 2");
		lblNewLabel_1.setBounds(135, 110, 92, 14);
		contentPane.add(lblNewLabel_1);

		JLabel lblNewLabel_2 = new JLabel("Compuerta 3");
		lblNewLabel_2.setBounds(135, 151, 80, 14);
		contentPane.add(lblNewLabel_2);

		JButton btnNewButton = new JButton("Actualizar caudal");
		btnNewButton.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				int
caudal1=Integer.parseInt(spinner1.getValue().toString());
				int
caudal2=Integer.parseInt(spinner2.getValue().toString());
				int
caudal3=Integer.parseInt(spinner3.getValue().toString());
				int caudalTotal=caudal1+caudal2+caudal3;
				if (caudalTotal<=100) {
					lblResultado.setText("Resultado: Caudal
bajo");
				}else {
					if(caudalTotal>100 && caudalTotal<=200) {
						lblResultado.setText("Resultado:
Caudal medio");
					}else {
						lblResultado.setText("Resultado:
Caudal alto");
					}
				}
			}
		});
		btnNewButton.setBounds(26, 207, 161, 23);
		contentPane.add(btnNewButton);

		lblResultado = new JLabel("Resultado: ");
		lblResultado.setBounds(213, 211, 188, 14);
		contentPane.add(lblResultado);
	}
}
```

Este será el resultado:


Enblase
55
Compuerta 1
83
Compuerta 2
20
Compuerta 3
Actualizar caudal
Resultado: Caudal medio

Capítulo 170.- Plug-in WindowBuilder problemas resueltos – 4

Problema propuesto:

Implementar un programa para la extracción de dinero de un cajero automático.

Se debe poder fijar la cantidad de dinero a extraer:

Disponer de control de tipo JComboBox (disponer de los valores 0, 50, 100, 150, etc. hasta 500).

Por otro lado poder seleccionar el tipo de cuenta (almacenar en otro JComboBox los textos "Caja de Ahorro" y "Cuenta Corriente".

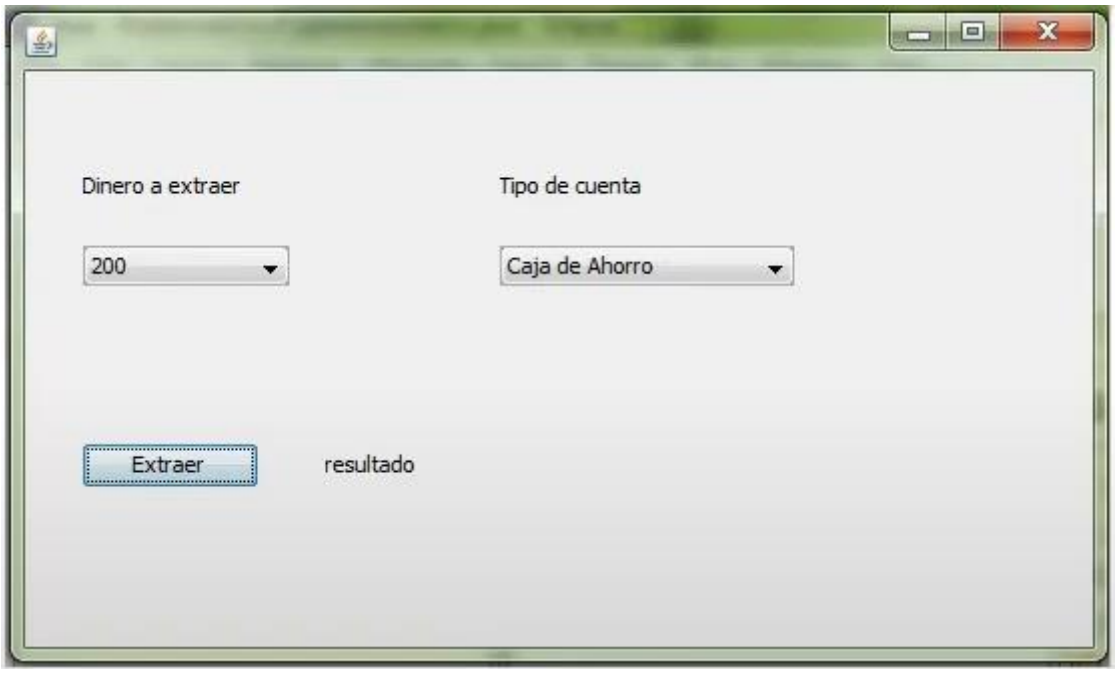
Se debe tener en cuenta que:

De Caja de Ahorro se puede extraer hasta 200.

De Cuenta Corriente se puede extraer hasta 500.

Al presionar el botón extraer mostrar en un label el texto "correcto· si para el tipo de cuenta el importe está permitido.

Inicialmente el cajero tiene almacenado 3000 pesos. Restar en cada extracción el importe respectivo y mostrar el mensaje "fuera de servicio" cuando se intente extraer más del dinero que hay en el cajero.



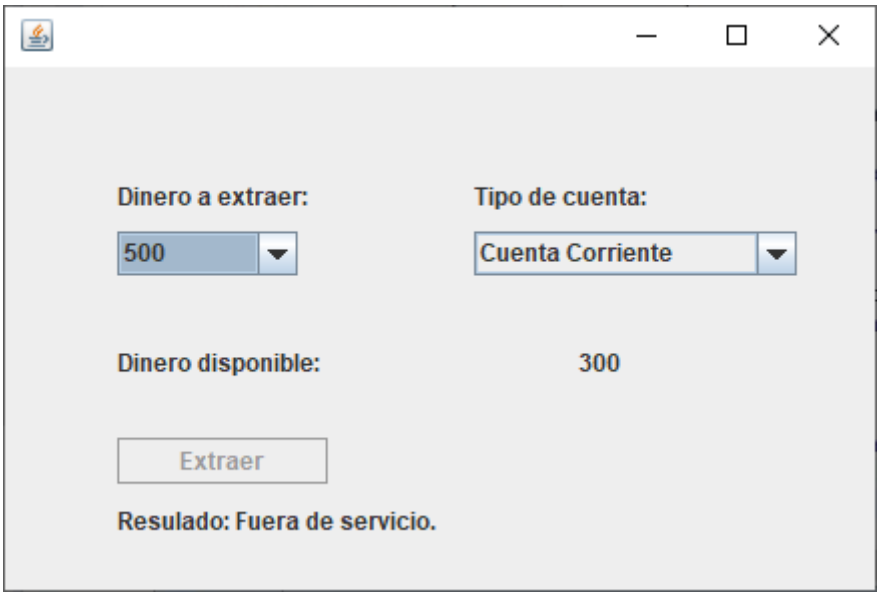
Vamos a programar:

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JComboBox;
import javax.swing.DefaultComboBoxModel;
import javax.swing.JButton;
```

```
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.SwingConstants;

public class Formulario extends JFrame {

	private JPanel contentPane;
	private int saldo=3000;
	private JLabel lblResultado;
	private JLabel lblSaldo;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Formulario frame = new Formulario();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public Formulario() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 450, 300);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

		setContentPane(contentPane);
		contentPane.setLayout(null);

		JLabel lblNewLabel = new JLabel("Dinero a extraer:");
		lblNewLabel.setBounds(56, 57, 118, 14);
		contentPane.add(lblNewLabel);

		JLabel lblNewLabel_1 = new JLabel("Tipo de cuenta:");
		lblNewLabel_1.setBounds(234, 57, 118, 14);
		contentPane.add(lblNewLabel_1);

		JComboBox comboDinero = new JComboBox();
		comboDinero.setModel(new DefaultComboBoxModel(new String[] {"0",
"50", "100", "150", "200", "250", "300", "350", "400", "450", "500"}));
		comboDinero.setBounds(56, 82, 90, 22);
		contentPane.add(comboDinero);

		JComboBox comboCuenta = new JComboBox();
		comboCuenta.setModel(new DefaultComboBoxModel(new String[]
{"Caja de Ahorro", "Cuenta Corriente"}));
		comboCuenta.setBounds(234, 82, 161, 22);
		contentPane.add(comboCuenta);
```

```
		JButton btnExtraer = new JButton("Extraer");
		btnExtraer.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				int
dinero=Integer.parseInt(comboDinero.getSelectedItem().toString());
				String
entidad=comboCuenta.getSelectedItem().toString();
				if (dinero>saldo) {
					lblResultado.setText("Resulado: Fuera de
servicio.");
					btnExtraer.setEnabled(false);
				}else {
					if(entidad.equals("Caja de Ahorro") &&
dinero<=200) {
						saldo=saldo-dinero;
						lblResultado.setText("Resulado:
Operación realizada con exito.");
					}else {
						if(entidad.equals("Cuenta Corriente")
&& dinero<=500) {
							saldo=saldo-dinero;
							lblResultado.setText("Resulado:
Operación realizada con exito.");
						}else {
							lblResultado.setText("Resulado:
Operación no permitida.");
						}
					}
				}
				lblSaldo.setText(Integer.toString(saldo));
			}

	});
		btnExtraer.setBounds(56, 185, 105, 23);
		contentPane.add(btnExtraer);

		lblResultado = new JLabel("Resultado: ");
		lblResultado.setBounds(56, 219, 339, 14);
		contentPane.add(lblResultado);
		lblResultado.setText("Resultado:");

		JLabel lblNewLabel_2 = new JLabel("Dinero disponible:");
		lblNewLabel_2.setBounds(56, 140, 118, 14);
		contentPane.add(lblNewLabel_2);

		lblSaldo = new JLabel("3000");
		lblSaldo.setHorizontalAlignment(SwingConstants.RIGHT);
		lblSaldo.setBounds(217, 140, 90, 14);
		contentPane.add(lblSaldo);
	}
}
```

Si ejecutamos este será el resultado:


Dinero a extraer:
500
Tipo de cuenta:
Cuenta Corriente
Dinero disponible:
300
Extraer
Resulado: Fuera de servicio.

Capítulo 171.- Clase Graphics y sus métodos – 1

Java proporciona la clase Graphics, que permite dibujar elipses, cuadrados, líneas, mostrar texto y también tiene muchos otros métodos de dibujo. Para cualquier programador, es esencial el entendimiento de la clase Graphics, antes de adentrarse en el dibujo en Java.

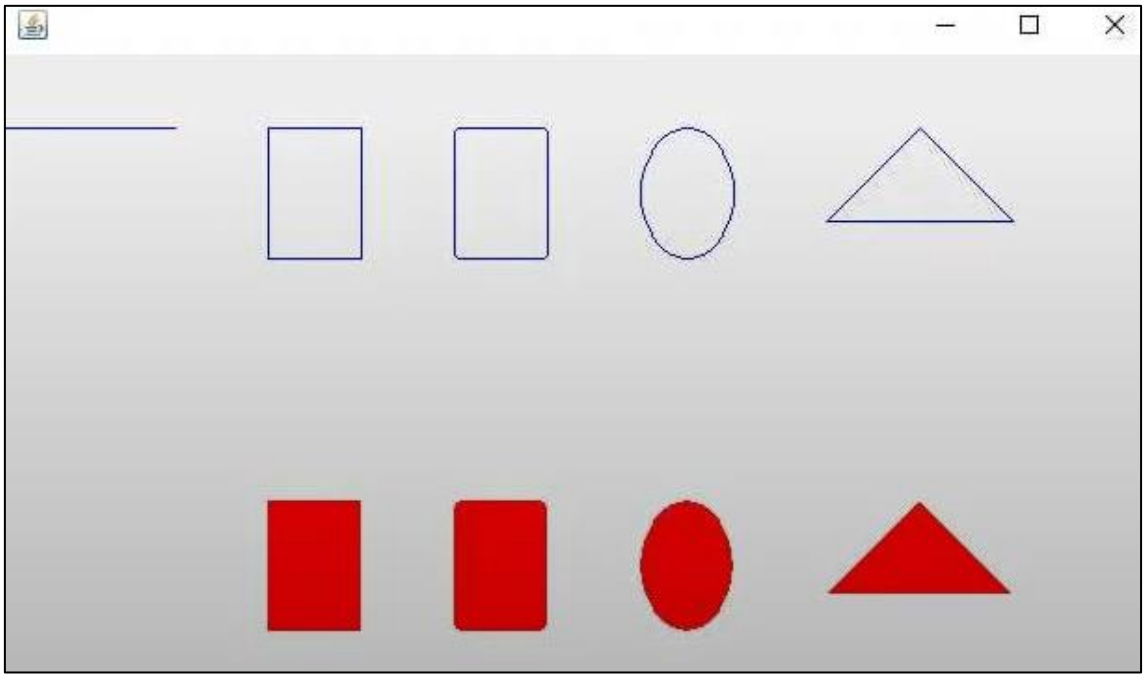
Para poder pintar. un programa necesita un contexto válido, representado por una instancia de la clase Graphics. Pero esta clase no se pueden instanciar directamente; así que debemos crear un componente y pasarlo al programa como un argumento al método Paint().

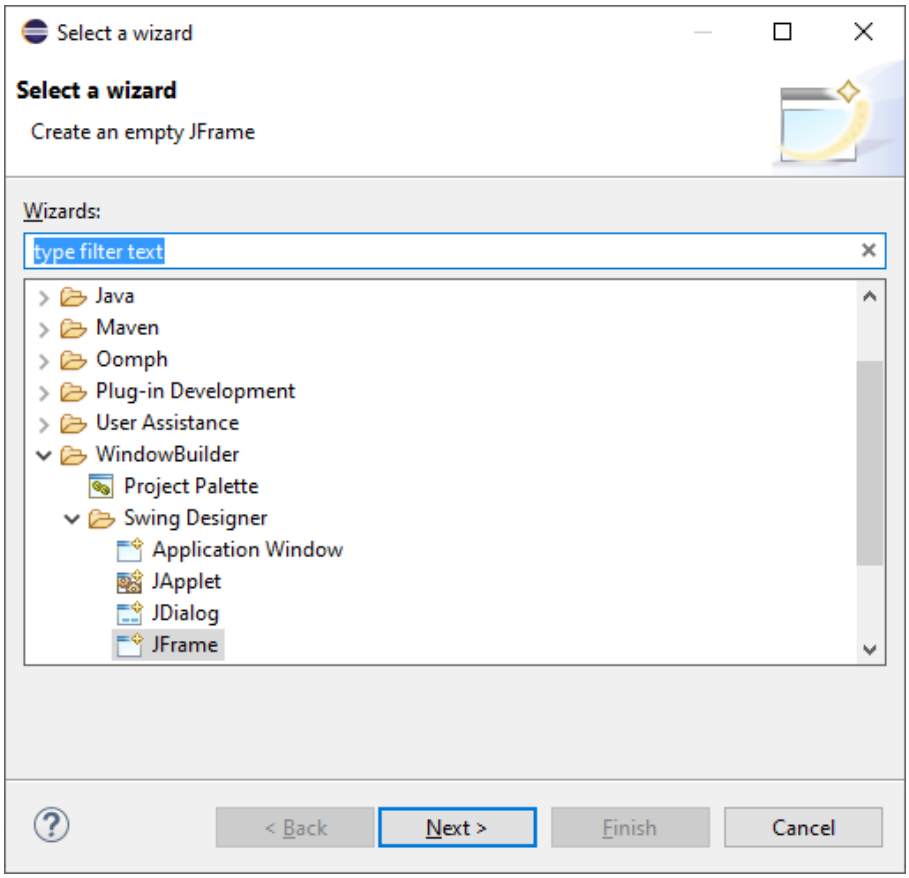
El único argumento del método Paint() es un objeto de esta clase. La clase Graphics dispone de métodos para soportar tres categorías de operaciones gráficas:

1) Dibujo de primitivas gráficas.
2) Dibujo de texto.
3) Presentación de imágenes en formatos *.gif ky *.jpg.

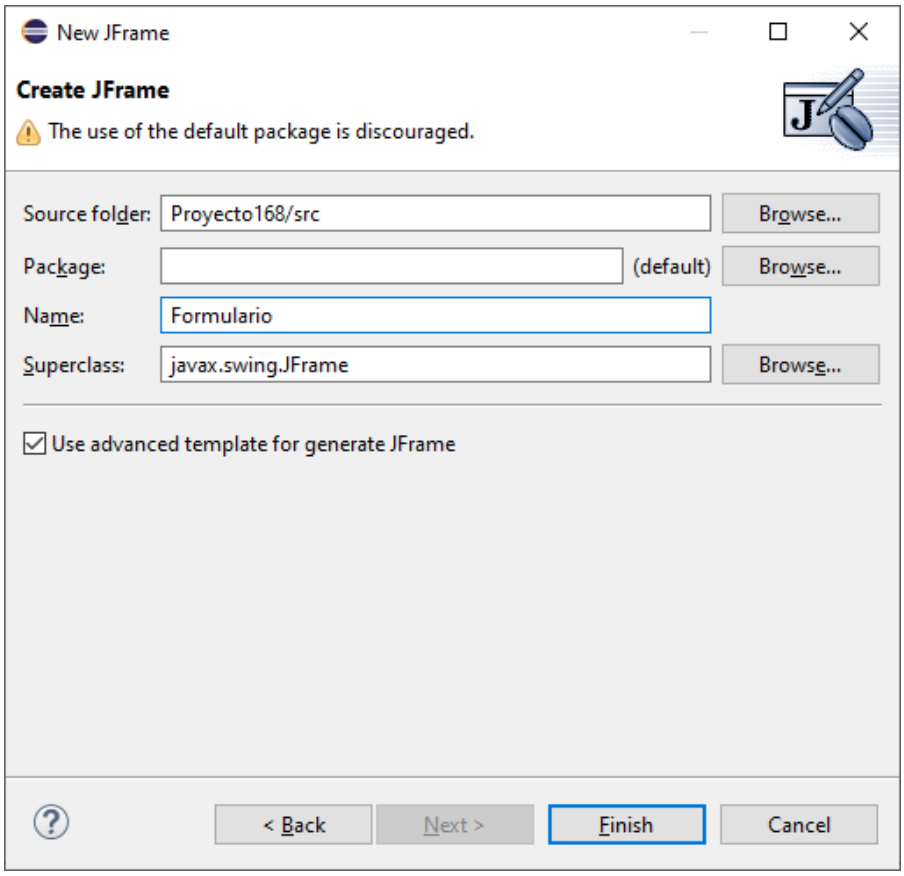
Problema:

Crear una aplicación que utilice las primitivas principales de la clase Graphics:





Seguimos con la librería WindowBuilder y JFrame.



Sobrescribimos el siguiente método:

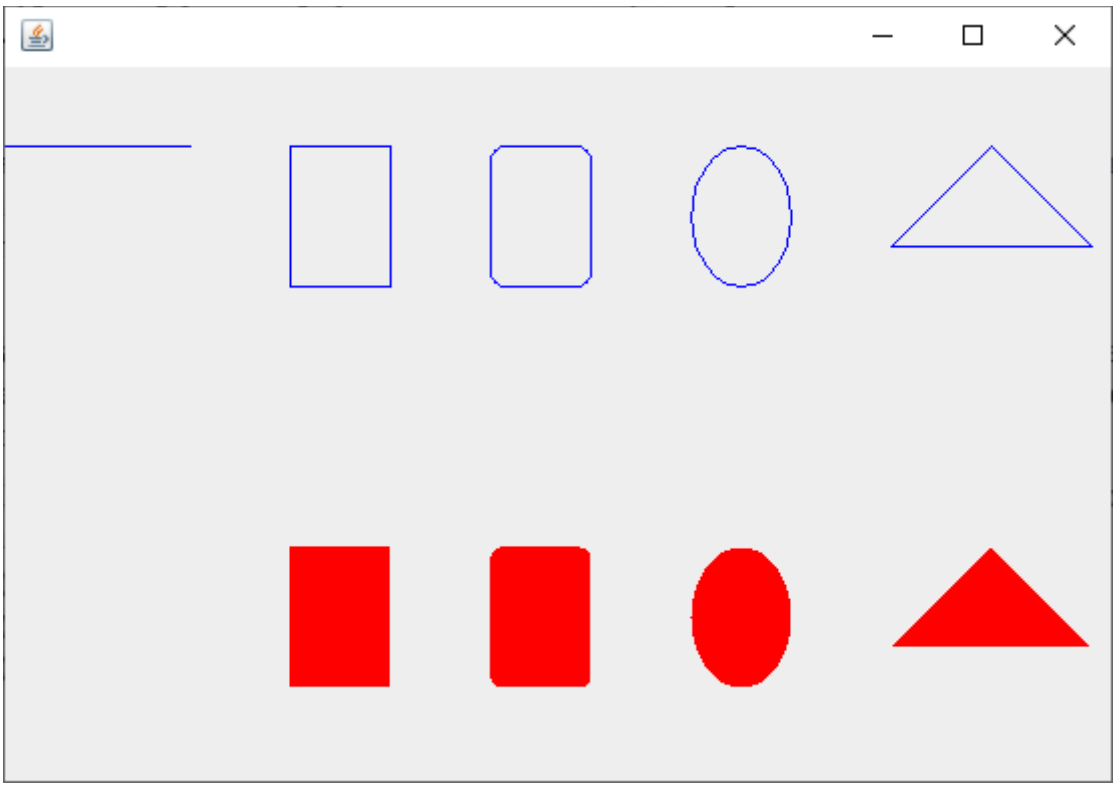
```
@Override
public void paint(Graphics g) {
    super.paint(g);

    g.setColor(Color.blue);
    g.drawLine(0, 70, 100, 70);
    g.drawRect(150, 70, 50, 70);
    g.drawRoundRect(250,70,50,70,15,15);
    g.drawOval(350, 70, 50, 70);
    int[] vx= {500,550,450};
    int[] vy= {70,120,120};
    g.drawPolygon(vx,vy,3);

    g.setColor(Color.red);
    g.fillRect(150, 270, 50, 70);
    g.fillRoundRect(250,270,50,70,10,10);
    g.fillOval(350, 270, 50, 70);
    int[] vx2= {500, 550, 450};
    int[] vy2= {270,320,320};
    g.fillPolygon(vx2, vy2, 3);
}
```

Así queda todo el código:

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Graphics;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

public class Formulario extends JFrame {

	private JPanel contentPane;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Formulario frame = new Formulario();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public Formulario() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 568, 395);
```

```
            contentPane = new JPanel();
            contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

            setContentPane(contentPane);
        }
        @Override
        public void paint(Graphics g) {
            super.paint(g);

            g.setColor(Color.blue);
            g.drawLine(0, 70, 100, 70);
            g.drawRect(150, 70, 50, 70);
            g.drawRoundRect(250,70,50,70,15,15);
            g.drawOval(350, 70, 50, 70);
            int[] vx= {500,550,450};
            int[] vy= {70,120,120};
            g.drawPolygon(vx,vy,3);

            g.setColor(Color.red);
            g.fillRect(150, 270, 50, 70);
            g.fillRoundRect(250,270,50,70,10,10);
            g.fillOval(350, 270, 50, 70);
            int[] vx2= {500, 550, 450};
            int[] vy2= {270,320,320};
            g.fillPolygon(vx2, vy2, 3);
        }

}
```

Al ejecutar este será el resultado:



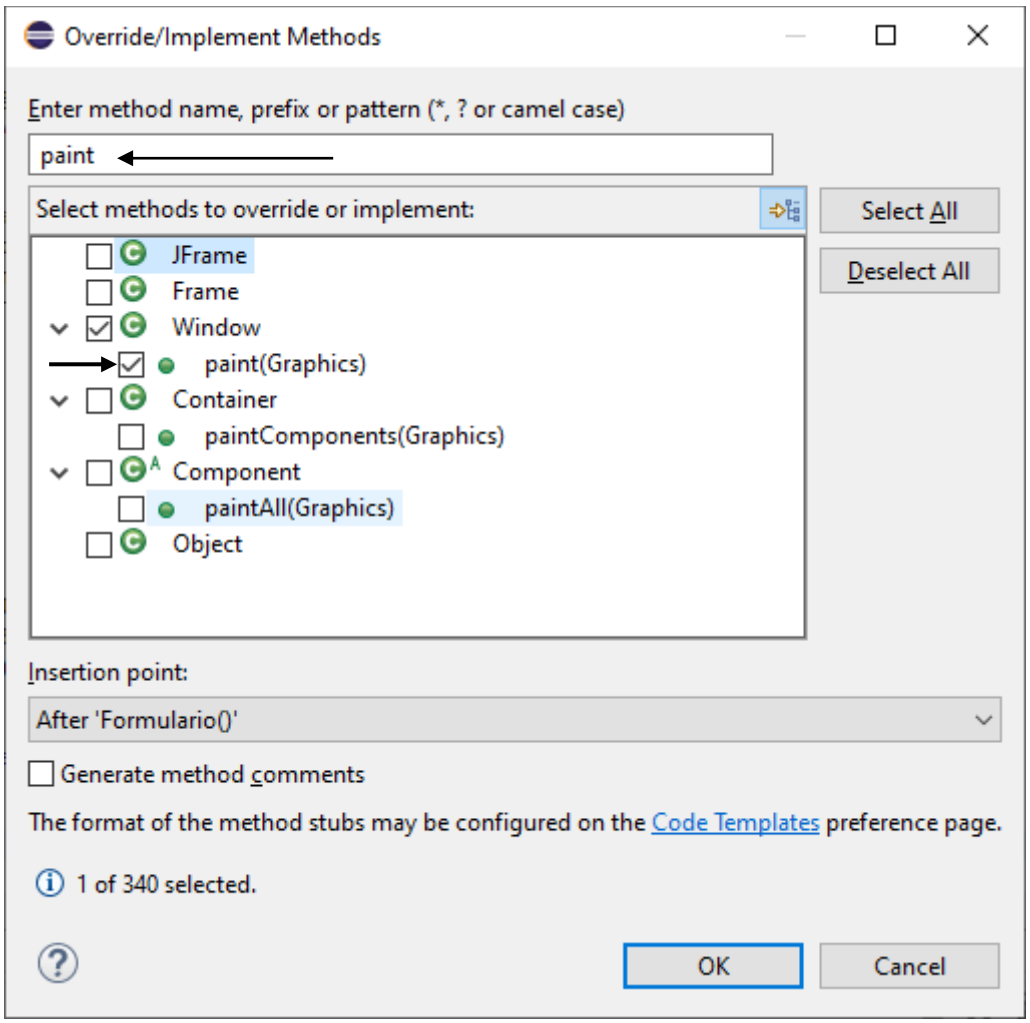
Capítulo 172.- Clase Graphics y sus métodos – 2

Problema:

Crear una aplicación que utilice las primitivas drawString en Java:

Seguimos trabajando con WindowBuilder.

Como en el capítulo anterior tenemos que sobrescribir el método paint, otra forma de hacerlo es seleccionando del menú Souces la opción Override/Implement Methods.



Lo buscamos y lo seleccionamos y por último seleccionamos el botón Ok.

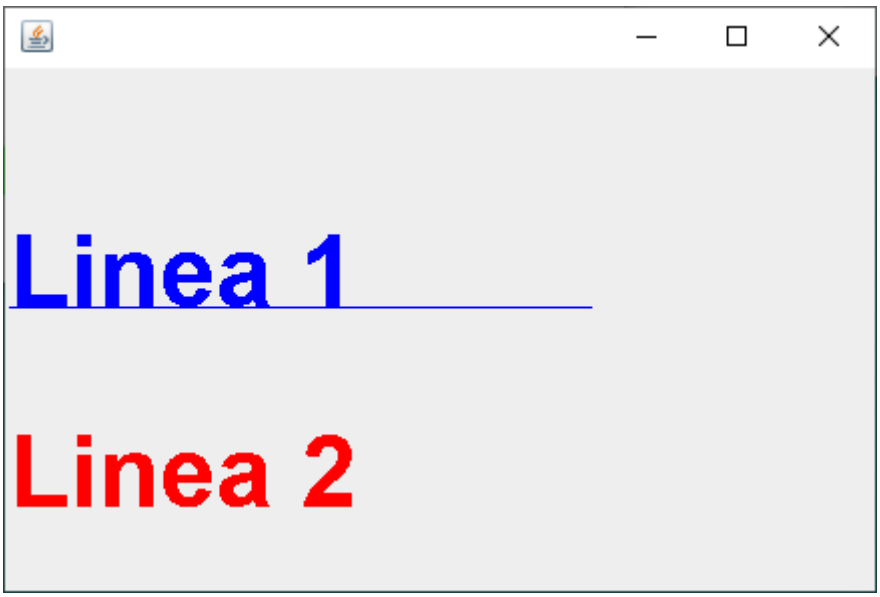
```
@Override
public void paint(Graphics g) {
    // TODO Auto-generated method stub
    super.paint(g);
}
```

Vamos a escribir el siguiente código:

```
@Override
public void paint(Graphics g) {
    // TODO Auto-generated method stub
    super.paint(g);
    g.setColor(Color.blue);
```

```
        g.setFont(new Font("Arial", Font.BOLD, 50));
        g.drawLine(10,150,300,150);
        g.drawString("Linea 1", 10, 150);
        g.setColor(Color.red);
        g.drawString("Linea 2", 10, 250);
    }
```

Si ejecutamos este será el resultado:



Capítulo 173.- Clase Graphics y sus métodos – 3

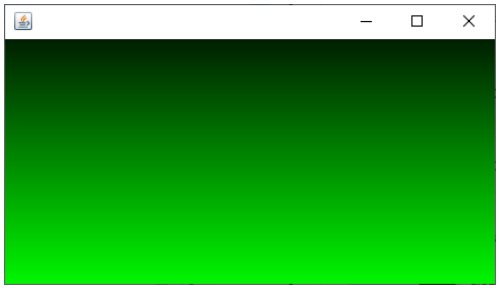
Problema:

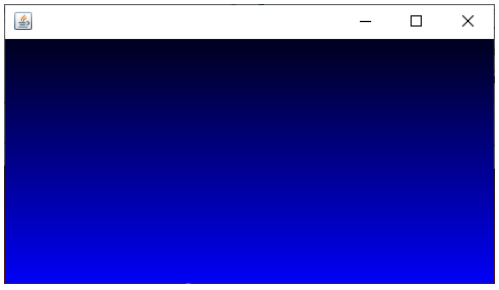
Crear una aplicación que dibuje 255 líneas creando un color distinto para cada una de ellas:

Este será el código:

```
@Override
public void paint(Graphics g) {
    // TODO Auto-generated method stub
    super.paint(g);
    for(int rojo=0; rojo<=255; rojo++) {
        Color color1=new Color(rojo, 0, 0);
        g.setColor(color1);
        g.drawLine(0,rojo, 450, rojo);
    }
}
```

Realiza las siguientes prácticas:





Capítulo 174.- Clase Graphics y sus métodos – 4

Problema:

Crear un aplicación que muestre un archivo jpg dentro de un JFrame.

Tenemos que guardar la imagen en la carpeta donde se encuentra nuestro proyecto:

.settings	27/05/2023 9:08	Carpeta de archivos	
bin	27/05/2023 9:08	Carpeta de archivos	
src	27/05/2023 9:21	Carpeta de archivos	
.classpath	27/05/2023 9:08	Archivo CLASSPATH	1 KB
.project	27/05/2023 9:08	Archivo PROJECT	1 KB
imagen1.jpg	27/05/2023 9:17	Archivo JPG	6 KB

Escribimos el código:

```
@Override
public void paint(Graphics g) {
    // TODO Auto-generated method stub
    super.paint(g);

    Toolkit t=Toolkit.getDefaultToolkit();
    Image imagen=t.getImage("imagen1.jpg");
    g.drawImage(imagen, 75, 50, this);
}
```

Si ejecutamos este será el resultado:

Para cambiar el tamaño de la imagen.

```
@Override
public void paint(Graphics g) {
    // TODO Auto-generated method stub
    super.paint(g);

    Toolkit t=Toolkit.getDefaultToolkit();
    Image imagen=t.getImage("imagen1.jpg");
    g.drawImage(imagen, 75, 50, 140, 80, this);
}
```

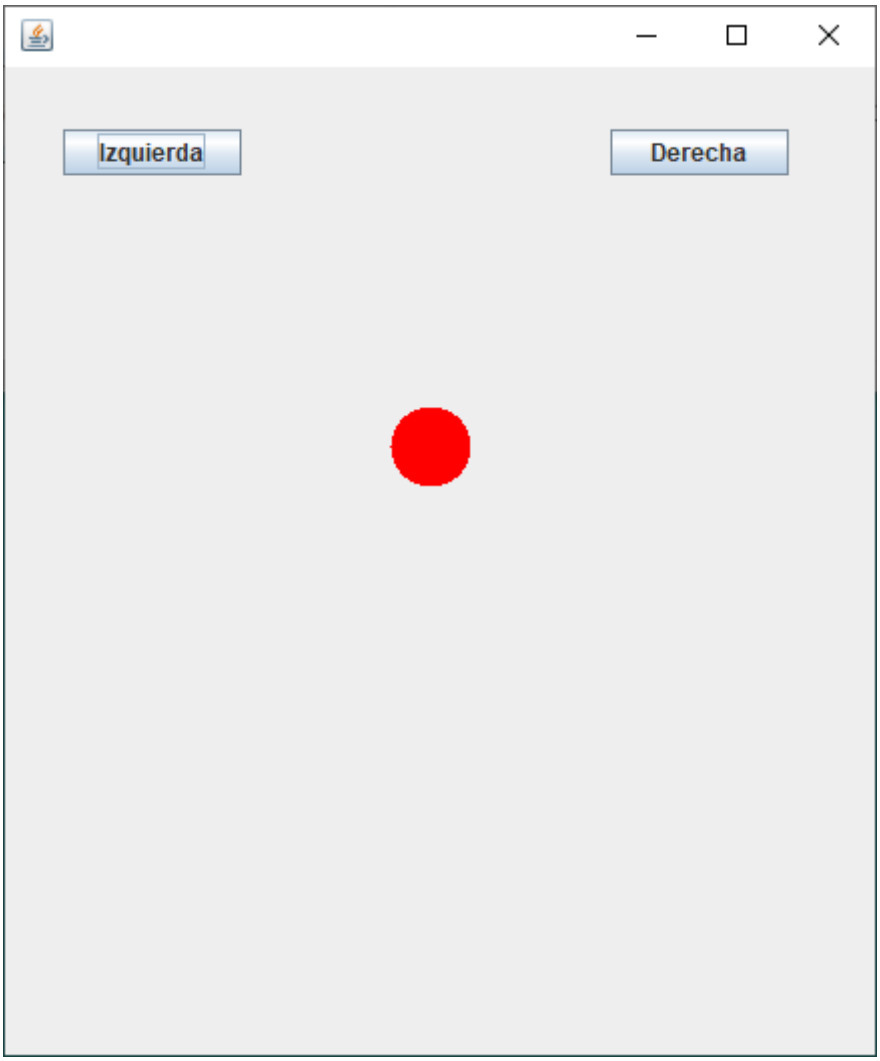
Si ejecutamos este será el resultado:

Capítulo 175.- Clase Graphics y sus métodos – 5

Problema:

Crear un aplicación que muestre un círculo en medio de la pantalla y mediante dos botones permitir que se desplace a izquierda o derecha.

Realizaremos el siguiente diseño:

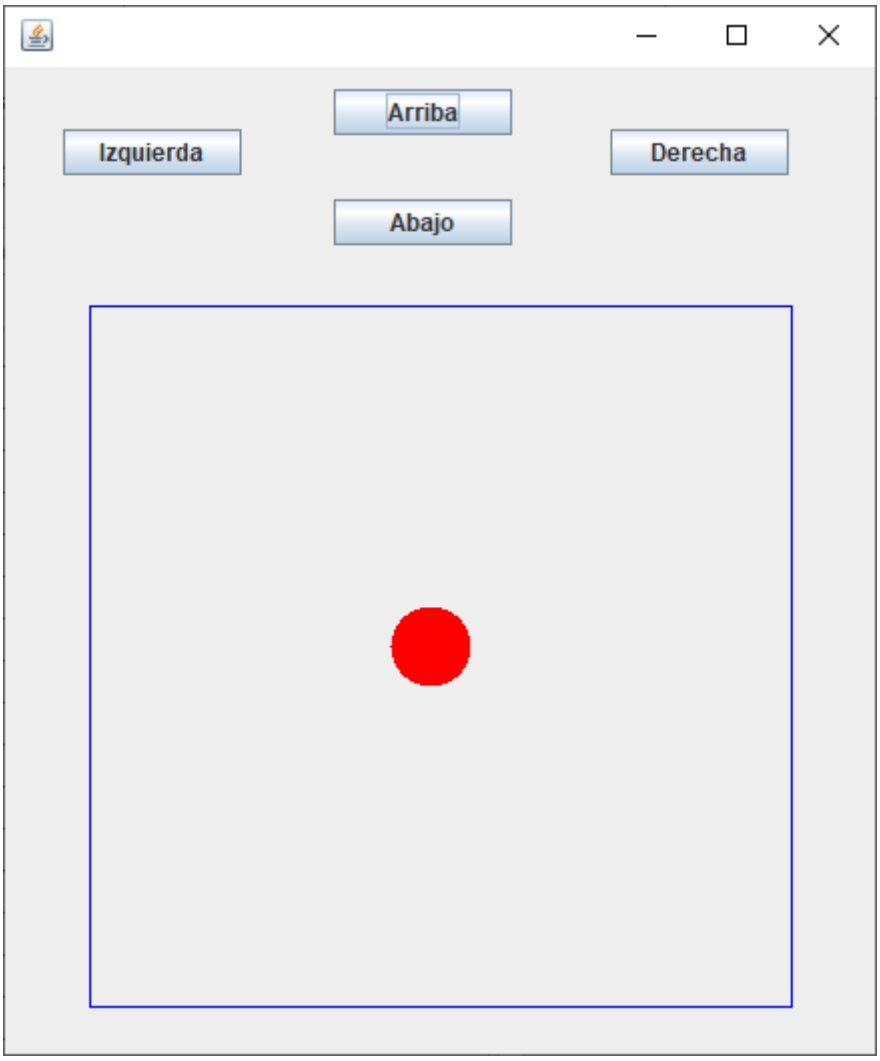


Este será el código:

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Graphics;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class Formulario extends JFrame {

	private JPanel contentPane;
```

```
	private int columna=200;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Formulario frame = new Formulario();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public Formulario() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 450, 532);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

		setContentPane(contentPane);
		contentPane.setLayout(null);

		JButton btnNewButton = new JButton("Izquierda");
		btnNewButton.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				columna=columna-10;
				repaint();
			}
		});
		btnNewButton.setBounds(29, 31, 89, 23);
		contentPane.add(btnNewButton);

		JButton btnNewButton_1 = new JButton("Derecha");
		btnNewButton_1.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				columna=columna+10;
				repaint();
			}
		});
		btnNewButton_1.setBounds(302, 31, 89, 23);
		contentPane.add(btnNewButton_1);
	}

	@Override
	public void paint(Graphics g) {
		// TODO Auto-generated method stub
		super.paint(g);
		g.setColor(Color.red);
		g.fillOval(columna,  200,  40,  40);
	}
}
```

Ahora con los conocimientos adquiridos vamos a realizar modificaciones con dos botones más arriba y abajo para que se pueda mover en todas las direcciones y un cuadrado para limitar su movimiento.



Este será el código:

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Graphics;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class Formulario extends JFrame {

	private JPanel contentPane;
	private int columna=200;
	private int fila=300;

	/**
```

```
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Formulario frame = new Formulario();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public Formulario() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 450, 532);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

		setContentPane(contentPane);
		contentPane.setLayout(null);

		JButton btnNewButton = new JButton("Izquierda");
		btnNewButton.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				if(columna>50) {
					columna=columna-10;
					repaint();
				}
			}
		});
		btnNewButton.setBounds(29, 31, 89, 23);
		contentPane.add(btnNewButton);

		JButton btnNewButton_1 = new JButton("Derecha");
		btnNewButton_1.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				if(columna<360) {
					columna=columna+10;
					repaint();
				}
			}
		});
		btnNewButton_1.setBounds(302, 31, 89, 23);
		contentPane.add(btnNewButton_1);

		JButton btnNewButton_2 = new JButton("Arriba");
		btnNewButton_2.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				if (fila>150) {
					fila=fila-10;
					repaint();
				}
			}
```

```
		});
		btnNewButton_2.setBounds(164, 11, 89, 23);
		contentPane.add(btnNewButton_2);

		JButton btnNewButton_3 = new JButton("Abajo");
		btnNewButton_3.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				if(fila<460) {
					fila=fila+10;
					repaint();
				}
			}
		});
		btnNewButton_3.setBounds(164, 66, 89, 23);
		contentPane.add(btnNewButton_3);
	}

	@Override
	public void paint(Graphics g) {
		// TODO Auto-generated method stub
		super.paint(g);
		g.setColor(Color.red);
		g.fillOval(columna,  fila,  40,  40);
		g.setColor(Color.blue);
		g.drawRect(50,150, 350, 350);
	}
}
```

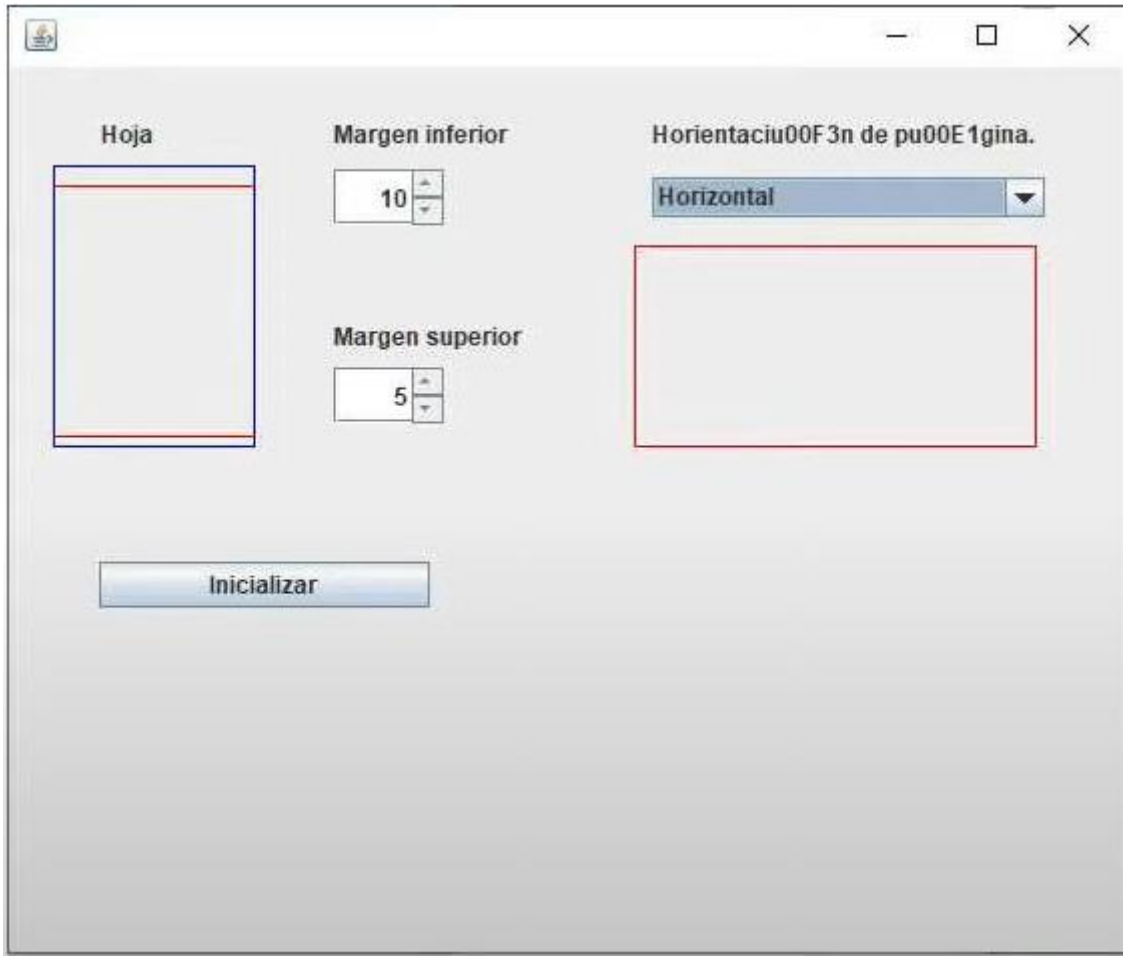
Capítulo 176.- Clase Graphics y sus métodos – 6

Se debe desarrollar una pantalla para configurar ciertas características de un procesador de textos. Debe aparecer y poderse seleccionarse los márgenes superior e inferior de la página.

Los márgenes puede ir en el rango de 0 a 10. Desplazar las líneas a medida que modificamos los márgenes.

Por otro lado tenemos la orientación de página. La misma se administra a través de un JComboBox que tiene dos posibles (Horizontal y Vertical). Cuando está seleccionado en el JComboBox el String horizontal dibuja un rectángulo con una base mayor a la altura, y cuando está seleccionado el String Vertical dibujar un rectángulo con una base menor.

Cuando se presiona el botón inicializar la configuración de márgenes se inicializan con 0 y se selecciona orientación horizontal.



Este será el código:

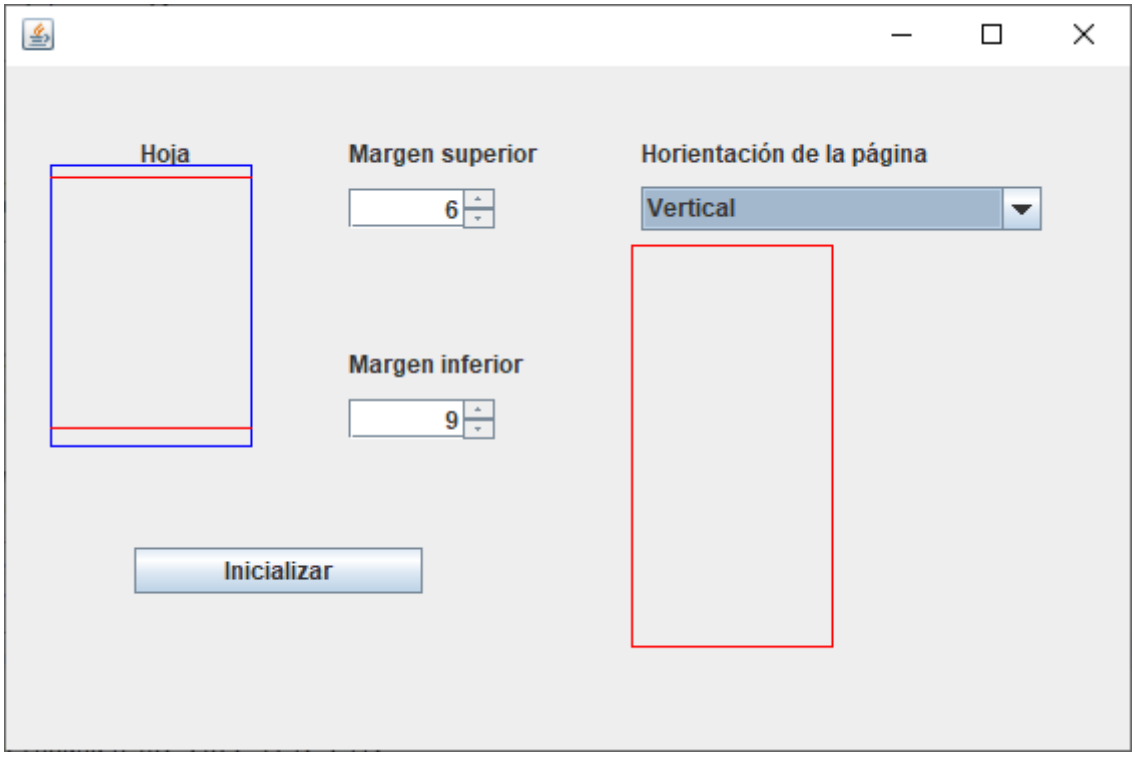
```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Graphics;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JSpinner;
```

```
import javax.swing.SpinnerNumberModel;
import javax.swing.JComboBox;
import javax.swing.DefaultComboBoxModel;
import javax.swing.JButton;
import javax.swing.event.ChangeListener;
import javax.swing.event.ChangeEvent;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class Formulario extends JFrame {

	private JPanel contentPane;
	private JSpinner spinner1;
	private JSpinner spinner2;
	private JComboBox combo1;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Formulario frame = new Formulario();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public Formulario() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 577, 380);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

		setContentPane(contentPane);
		contentPane.setLayout(null);

		JLabel lblNewLabel = new JLabel("Hoja");
		lblNewLabel.setBounds(67, 36, 46, 14);
		contentPane.add(lblNewLabel);

		JLabel lblNewLabel_1 = new JLabel("Margen inferior");
		lblNewLabel_1.setBounds(171, 141, 102, 14);
		contentPane.add(lblNewLabel_1);

		JLabel lblNewLabel_2 = new JLabel("Margen superior");
		lblNewLabel_2.setBounds(171, 36, 102, 14);
		contentPane.add(lblNewLabel_2);

		spinner1 = new JSpinner();
```

```
		spinner1.addChangeListener(new ChangeListener() {
			public void stateChanged(ChangeEvent e) {
				repaint();
			}
		});
		spinner1.setModel(new SpinnerNumberModel(0, 0, 10, 1));
		spinner1.setBounds(171, 61, 73, 20);
		contentPane.add(spinner1);

		spinner2 = new JSpinner();
		spinner2.addChangeListener(new ChangeListener() {
			public void stateChanged(ChangeEvent e) {
				repaint();
			}
		});
		spinner2.setModel(new SpinnerNumberModel(0, 0, 10, 1));
		spinner2.setBounds(171, 166, 73, 20);
		contentPane.add(spinner2);

		JLabel lblNewLabel_3 = new JLabel("Horientaci\u00F3n de la
p\u00E1gina");
		lblNewLabel_3.setBounds(317, 36, 161, 14);
		contentPane.add(lblNewLabel_3);

		combo1 = new JComboBox();
		combo1.addItemListener(new ItemListener() {
			public void itemStateChanged(ItemEvent e) {
				repaint();
			}
		});
		combo1.setModel(new DefaultComboBoxModel(new String[]
{"Horizontal", "Vertical"}));
		combo1.setBounds(317, 60, 200, 22);
		contentPane.add(combo1);

		JButton btnNewButton = new JButton("Inicializar");
		btnNewButton.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				spinner1.setValue(0);
				spinner2.setValue(0);
				combo1.setSelectedIndex(0);
			}
		});
		btnNewButton.setBounds(64, 240, 144, 23);
		contentPane.add(btnNewButton);
	}

	@Override
	public void paint(Graphics g) {
		// TODO Auto-generated method stub
		super.paint(g);
		g.setColor(Color.blue);
		g.drawRect(30,80,100,140);
		int ms=Integer.parseInt(spinner1.getValue().toString());
		int mi=Integer.parseInt(spinner2.getValue().toString());
		g.setColor(Color.red);
		g.drawLine(30, 80+ms, 130, 80+ms);
		g.drawLine(30, 220-mi, 130, 220-mi);
		String direccion=combo1.getSelectedItem().toString();
```

```
                if (direccion.equals("Horizontal")) {
                        g.drawRect(320, 120, 200, 100);
                }else {
                        g.drawRect(320, 120, 100, 200);
                }
        }
}
```

Cuando ejecutemos este será el resultado:



Evento del spinner: change / stateChanged.

Evento del combobox: Item / itemStateChanged.

Evento del botón: action / actionPerformed.

Capítulo 177.- Clase Graphics y sus métodos – 7

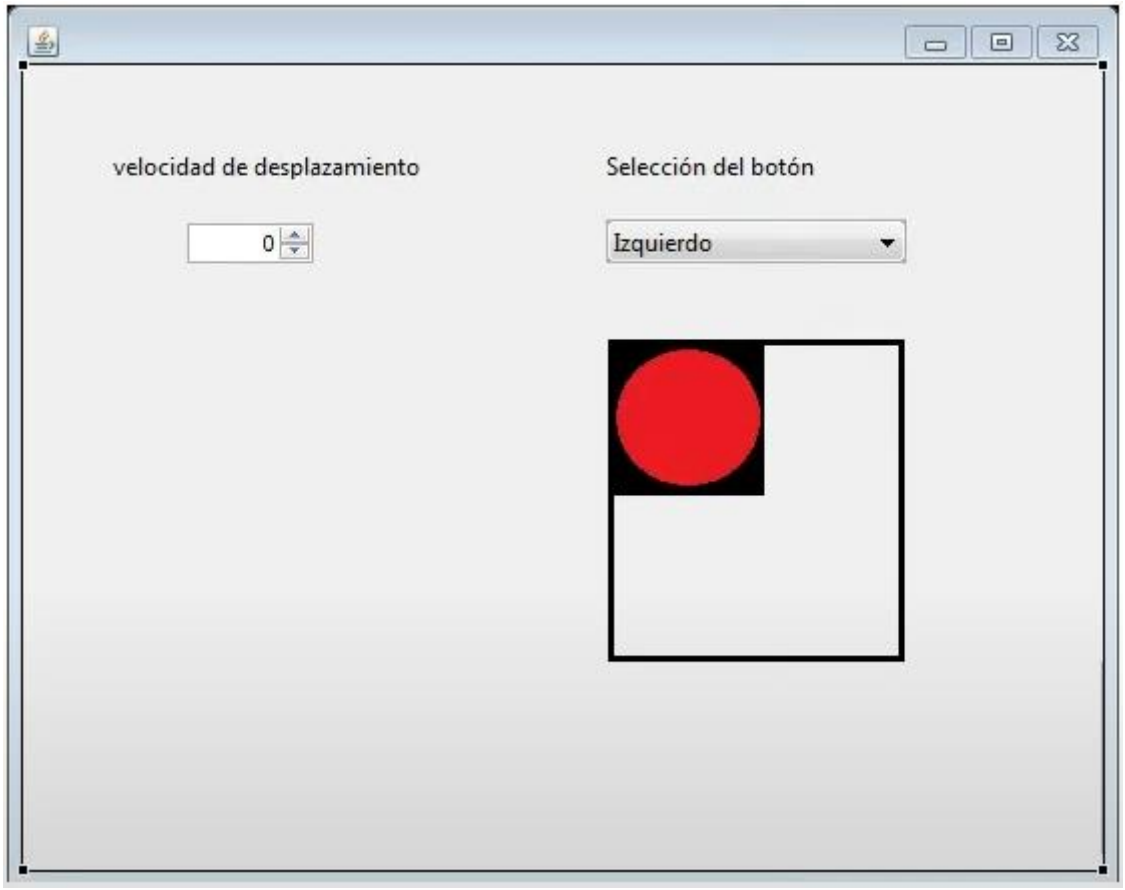
Problema propuesto:

Confeccionar un programa que permita configurar las características del mouse.

Por un lado debemos seleccionar la velocidad de desplazamiento de la flecha del mouse. Disponer un JSpinner para poder seleccionar los valores 0, 25, 50, 75 y 100.

Por otro lado debemos poder seleccionar cual de los dos botones del mouse será el principal, tenemos para esta función un JComboBox con dos opciones: izquierdo o derecho.

Cuando se selecciona el botón (cambio en el JComboBox) actualizar el gráfico mostrando el botón del mouse seleccionado (graficar en el método Paint el mouse en pantalla).



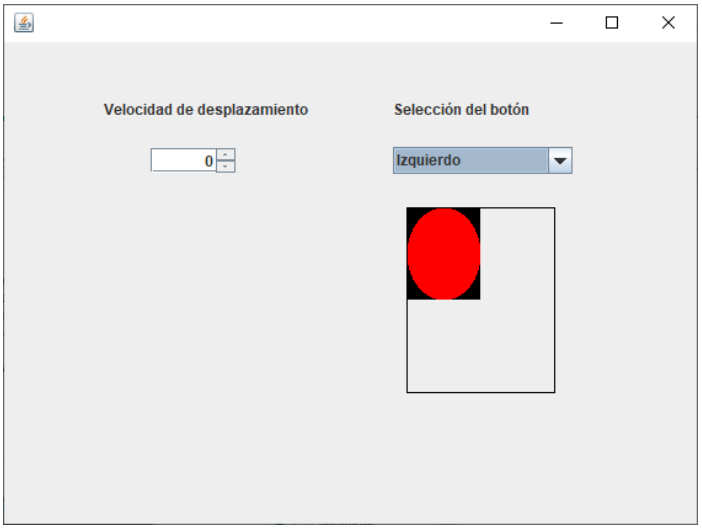
Vamos a programar:

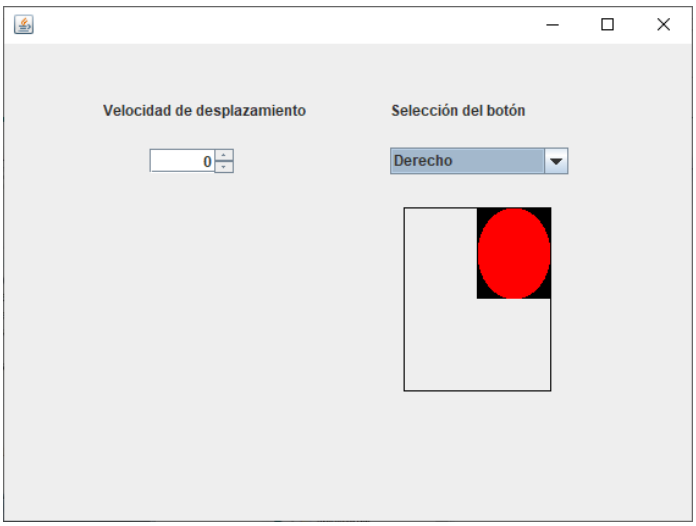
```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Graphics;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JSpinner;
import javax.swing.SpinnerNumberModel;
import javax.swing.JComboBox;
import javax.swing.DefaultComboBoxModel;
import java.awt.event.ItemListener;
```

```
import java.awt.event.ItemEvent;

public class Formulario extends JFrame {

	private JPanel contentPane;
	private JComboBox combo1;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Formulario frame = new Formulario();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public Formulario() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 579, 430);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

		setContentPane(contentPane);
		contentPane.setLayout(null);

		JLabel lblNewLabel = new JLabel("Velocidad de desplazamiento");
		lblNewLabel.setBounds(81, 47, 177, 14);
		contentPane.add(lblNewLabel);

		JLabel lblNewLabel_1 = new JLabel("Selecci\u00F3n del
bot\u00F3n");
		lblNewLabel_1.setBounds(316, 47, 146, 14);
		contentPane.add(lblNewLabel_1);

		JSpinner spinner1 = new JSpinner();
		spinner1.setModel(new SpinnerNumberModel(0, 0, 100, 25));
		spinner1.setBounds(119, 86, 69, 20);
		contentPane.add(spinner1);

		combo1 = new JComboBox();
		combo1.addItemListener(new ItemListener() {
			public void itemStateChanged(ItemEvent e) {
				repaint();
			}
		});
		combo1.setModel(new DefaultComboBoxModel(new String[]
{"Izquierdo", "Derecho"}));
		combo1.setBounds(316, 85, 146, 22);
		contentPane.add(combo1);
```

```
	}

	@Override
	public void paint(Graphics g) {
		// TODO Auto-generated method stub
		super.paint(g);
		g.setColor(Color.black);
		g.drawRect(335, 165, 120, 150);
		if(combo1.getSelectedItem().toString().equals("Izquierdo")) {
			g.fillRect(335,  165,  60, 75);
			g.setColor(Color.red);
			g.fillOval(335,165,60,75);
		}else {
			g.fillRect(395,  165,  60, 75);
			g.setColor(Color.red);
			g.fillOval(395,165,60,75);
		}
	}
}
```

Este será el resultado:





Capítulo 178.- clase Graphics y sus métodos – 8

En una aduana hay una máquina que sortea las personas cuyo equipaje serán revisados.

La persona seleccionada se viene del interior del país o del exterior (a través de un JComoboBox), y por otro lado selecciona la cantidad de bultos (JSpinner).

Luego presiona el botón sortear y aparece al lado de este botón un círculo rojo o verde. (En caso de ser rojo se tiene que revisar su equipaje, en caso de ser verde no se revisa).

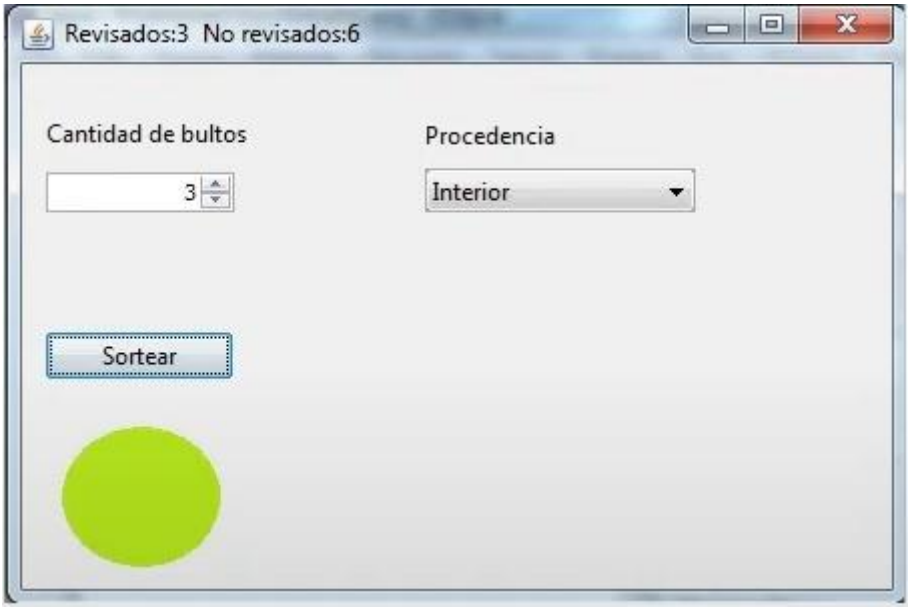
Para el sorteo generar un valor aleatorio entre 1 y 3. Si se genera un 1 se revisa, si se genera un 2 0 un 3 no se revisa.

Validar que también este seleccionado un valor distinto a cero en bultos (los valores pueden ir de 0 a 10).

Si la cantidad de bultos supera a 5 se revisa siempre sus bultos (es decir que aparece un círculo rojo).

Luego de sortear fijar en cero la cantidad de bultos.

Mostrar en el título del JFrame la cantidad de bultos revisados y no revisados hasta el momento.



Este será el código:

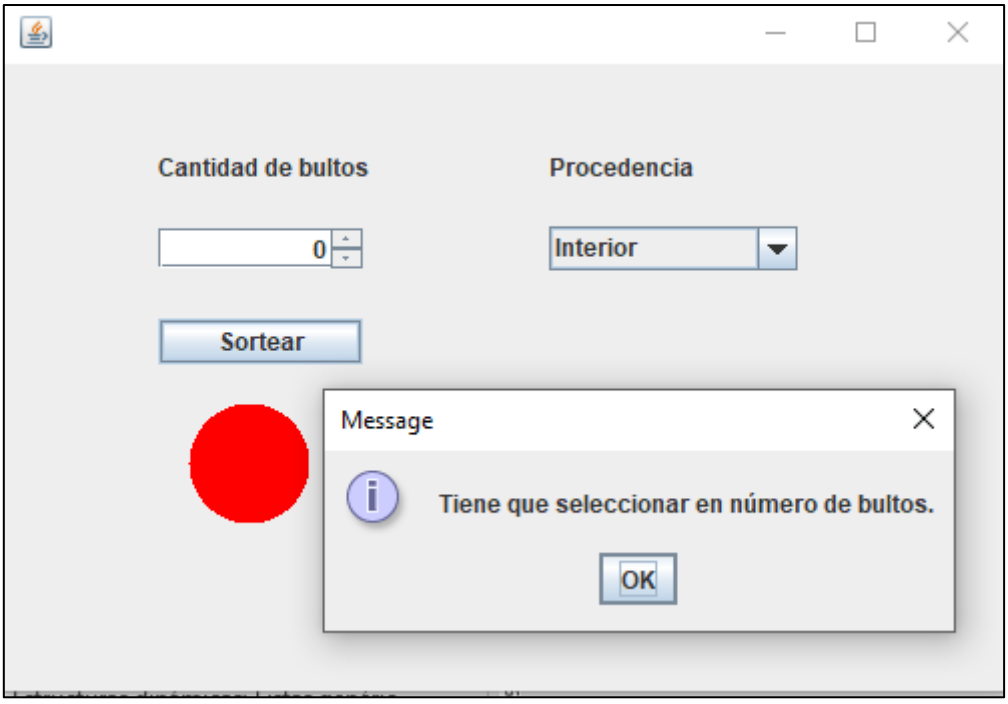
```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Graphics;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JSpinner;
import javax.swing.SpinnerNumberModel;
import javax.swing.JComboBox;
import javax.swing.DefaultComboBoxModel;
import javax.swing.JButton;
```

```
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class Formulario extends JFrame {

	private JPanel contentPane;
	private int sorteo;
	private JSpinner spinner;
	private int revi=0;
	private int noRevi=0;
	private int bultos;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Formulario frame = new Formulario();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public Formulario() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 515, 352);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

		setContentPane(contentPane);
		contentPane.setLayout(null);

		JLabel lblNewLabel = new JLabel("Cantidad de bultos");
		lblNewLabel.setBounds(77, 44, 139, 14);
		contentPane.add(lblNewLabel);

		JLabel lblNewLabel_1 = new JLabel("Procedencia");
		lblNewLabel_1.setBounds(272, 44, 124, 14);
		contentPane.add(lblNewLabel_1);

		spinner = new JSpinner();
		spinner.setModel(new SpinnerNumberModel(0, 0, 10, 1));
		spinner.setBounds(77, 82, 102, 20);
		contentPane.add(spinner);

		JComboBox comboBox = new JComboBox();
		comboBox.setModel(new DefaultComboBoxModel(new String[]
{"Interior", "Exterior"}));
		comboBox.setBounds(272, 81, 124, 22);
		contentPane.add(comboBox);
```

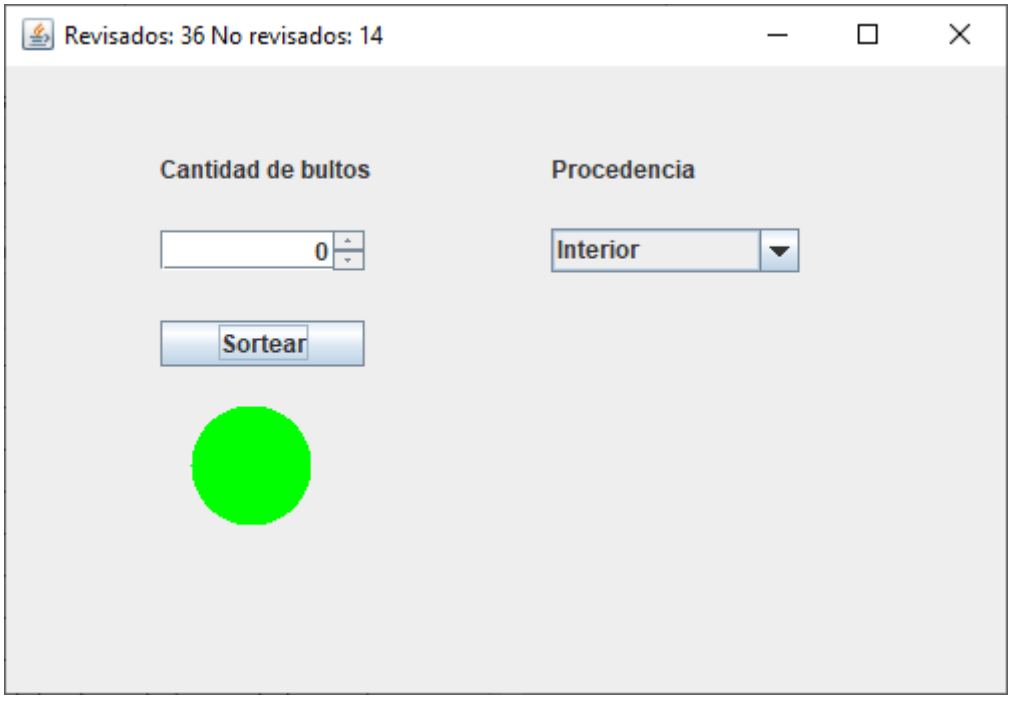
```
		JButton button = new JButton("Sortear");
		button.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				sorteo=1+(int)(Math.random()*3);

	bultos=Integer.parseInt(spinner.getValue().toString());
				if(bultos==0) {
					JOptionPane.showMessageDialog(null, "Tiene
que seleccionar en número de bultos.");
				}else {
					if(bultos>5) {

	revi=revi+(Integer.parseInt(spinner.getValue().toString()));
						repaint();
					}else {
						if(sorteo==1) {

	revi=revi+(Integer.parseInt(spinner.getValue().toString()));
							repaint();
						}else {

	noRevi=noRevi+(Integer.parseInt(spinner.getValue().toString()));
							repaint();
						}
					}
					spinner.setValue(0);
					String mensaje="Revisados: "+revi+" No
revisados: "+noRevi;
					setTitle(mensaje);
				}
				System.out.println(bultos);
			}
		});
		button.setBounds(77, 127, 102, 23);
		contentPane.add(button);
	}

	@Override
	public void paint(Graphics g) {
		// TODO Auto-generated method stub
		super.paint(g);
		g.setColor(Color.green);
		g.fillOval(100,200,60,60);
		if(bultos>5) {
			g.setColor(Color.green);
			g.fillOval(100,200,60,60);
		}else {
			if(sorteo==1) {
				g.setColor(Color.green);
				g.fillOval(100,200,60,60);
			}else {
				g.setColor(Color.red);
				g.fillOval(100,200,60,60);
			}
		}
	}
}
```

Si ejecutamos este será el resultado:

Si no introducimos ningún bulto.



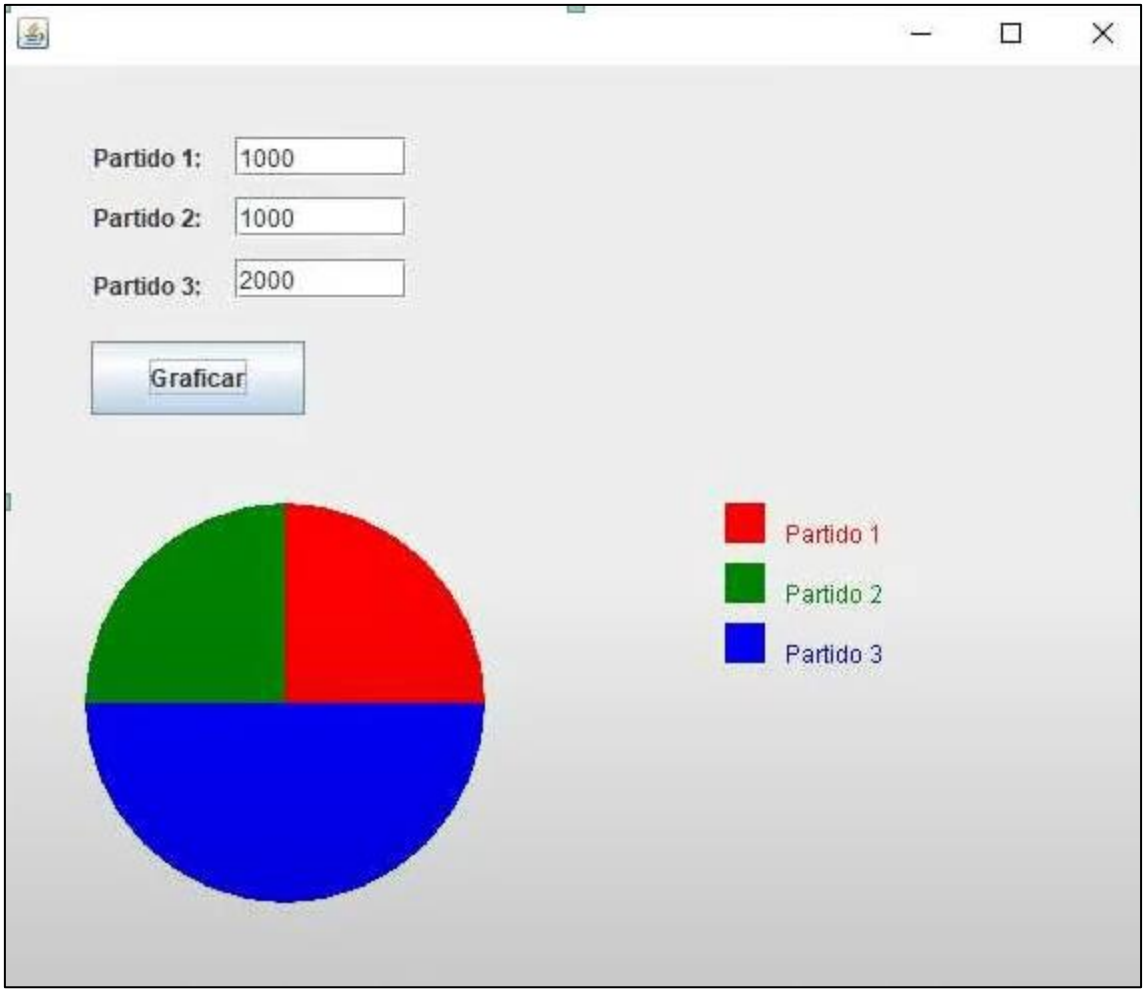
Vamos a ejecutar varias veces introduciendo diferentes unidades en los bultos.



Capítulo 179.- Gráficos estadísticos – 1

Problema:

Crear una aplicación que solicite el ingreso de tres valores por teclado que representan las cantidades de botos obtenidas por tres particos políticos. Luego mostrar un gráfico de tartas:



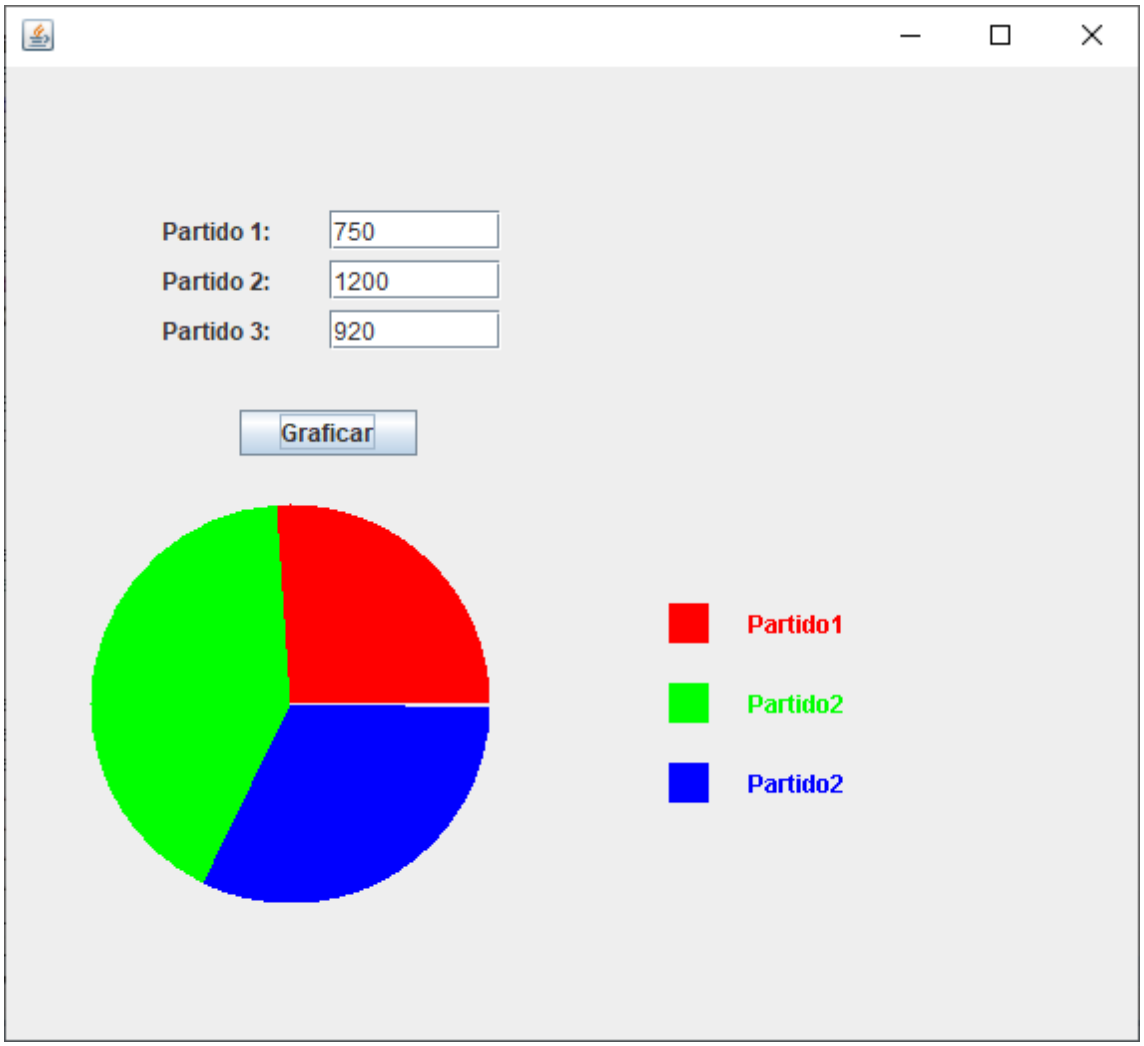
Este es el código:

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Font;
import java.awt.Graphics;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class GraficoTarta extends JFrame {

	private JPanel contentPane;
	private JTextField tf1;
```

```
private JTextField tf2;
private JTextField tf3;
private boolean grafica=false;

/**
 * Launch the application.
 */
public static void main(String[] args) {
	EventQueue.invokeLater(new Runnable() {
		public void run() {
			try {
				GraficoTarta frame = new GraficoTarta();
				frame.setVisible(true);
			} catch (Exception e) {
				e.printStackTrace();
			}
		}
	});
}

/**
 * Create the frame.
 */
public GraficoTarta() {
	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	setBounds(100, 100, 583, 527);
	contentPane = new JPanel();
	contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

	setContentPane(contentPane);
	contentPane.setLayout(null);

	JLabel lblNewLabel = new JLabel("Partido 1:");
	lblNewLabel.setBounds(78, 75, 74, 14);
	contentPane.add(lblNewLabel);

	JLabel lblNewLabel_1 = new JLabel("Partido 2:");
	lblNewLabel_1.setBounds(78, 100, 82, 14);
	contentPane.add(lblNewLabel_1);

	JLabel lblNewLabel_2 = new JLabel("Partido 3:");
	lblNewLabel_2.setBounds(78, 125, 74, 14);
	contentPane.add(lblNewLabel_2);

	tf1 = new JTextField();
	tf1.setBounds(162, 72, 86, 20);
	contentPane.add(tf1);
	tf1.setColumns(10);

	tf2 = new JTextField();
	tf2.setBounds(162, 97, 86, 20);
	contentPane.add(tf2);
	tf2.setColumns(10);

	tf3 = new JTextField();
	tf3.setBounds(162, 122, 86, 20);
	contentPane.add(tf3);
	tf3.setColumns(10);
```

```
		JButton btnNewButton = new JButton("Graficar");
		btnNewButton.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				grafica=true;
				repaint();
			}
		});
		btnNewButton.setBounds(117, 172, 89, 23);
		contentPane.add(btnNewButton);
	}

	@Override
	public void paint(Graphics g) {
		// TODO Auto-generated method stub
		super.paint(g);
		if (grafica) {
			int v1=Integer.parseInt(tf1.getText());
			int v2=Integer.parseInt(tf2.getText());
			int v3=Integer.parseInt(tf3.getText());
			int suma=v1+v2+v3;
			int grados1=v1*360/suma;
			int grados2=v2*360/suma;
			int grados3=v3*360/suma;

			g.setColor(Color.red);
			g.fillArc(50,  250, 200, 200, 0, grados1);
			g.setColor(Color.green);
			g.fillArc(50, 250, 200, 200, grados1, grados2);
			g.setColor(Color.blue);
			g.fillArc(50, 250, 200, 200, grados1+grados2, grados3);
			g.setColor(Color.red);
			g.setFont(new Font("Arial", Font.BOLD, 12));
			g.fillRect(340, 300, 20, 20);
			g.drawString("Partido1", 380, 315);
			g.setColor(Color.green);
			g.fillRect(340, 340, 20, 20);
			g.drawString("Partido2", 380, 355);
			g.setColor(Color.blue);
			g.fillRect(340, 380, 20, 20);
			g.drawString("Partido2", 380, 395);
		}
	}
}
```

Vamos a ejecutar introduciendo otros valores:



Capítulo 180.- Gráficos estadísticos – 2

Problema:

Crear una aplicación que solicite en ingreso de tres valores por teclado que representen las cantidades de votos obtenidos por tres partidos políticos. Luego mostrar un gráfico de barras horizontales:

Este será el código:

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Font;
import java.awt.Graphics;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class Grafico extends JFrame {

	private JPanel contentPane;
	private JTextField tf1;
	private JTextField tf2;
	private JTextField tf3;
	private int v1, v2, v3, max;
	private int por1, por2, por3;

	/**
	 * Launch the application.
```

```
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Grafico frame = new Grafico();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public Grafico() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 507, 432);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

		setContentPane(contentPane);
		contentPane.setLayout(null);

		JLabel lblNewLabel = new JLabel("Partido 1:");
		lblNewLabel.setBounds(69, 58, 66, 14);
		contentPane.add(lblNewLabel);

		JLabel lblNewLabel_1 = new JLabel("Partido 2:");
		lblNewLabel_1.setBounds(69, 83, 78, 14);
		contentPane.add(lblNewLabel_1);

		JLabel lblNewLabel_2 = new JLabel("Partido 3:");
		lblNewLabel_2.setBounds(69, 108, 78, 14);
		contentPane.add(lblNewLabel_2);

		tf1 = new JTextField();
		tf1.setBounds(139, 55, 86, 20);
		contentPane.add(tf1);
		tf1.setColumns(10);

		tf2 = new JTextField();
		tf2.setBounds(139, 83, 86, 20);
		contentPane.add(tf2);
		tf2.setColumns(10);

		tf3 = new JTextField();
		tf3.setBounds(139, 108, 86, 20);
		contentPane.add(tf3);
		tf3.setColumns(10);

		JButton btnNewButton = new JButton("Graficar");
		btnNewButton.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				v1=Integer.parseInt(tf1.getText());
				v2=Integer.parseInt(tf2.getText());
				v3=Integer.parseInt(tf3.getText());
```

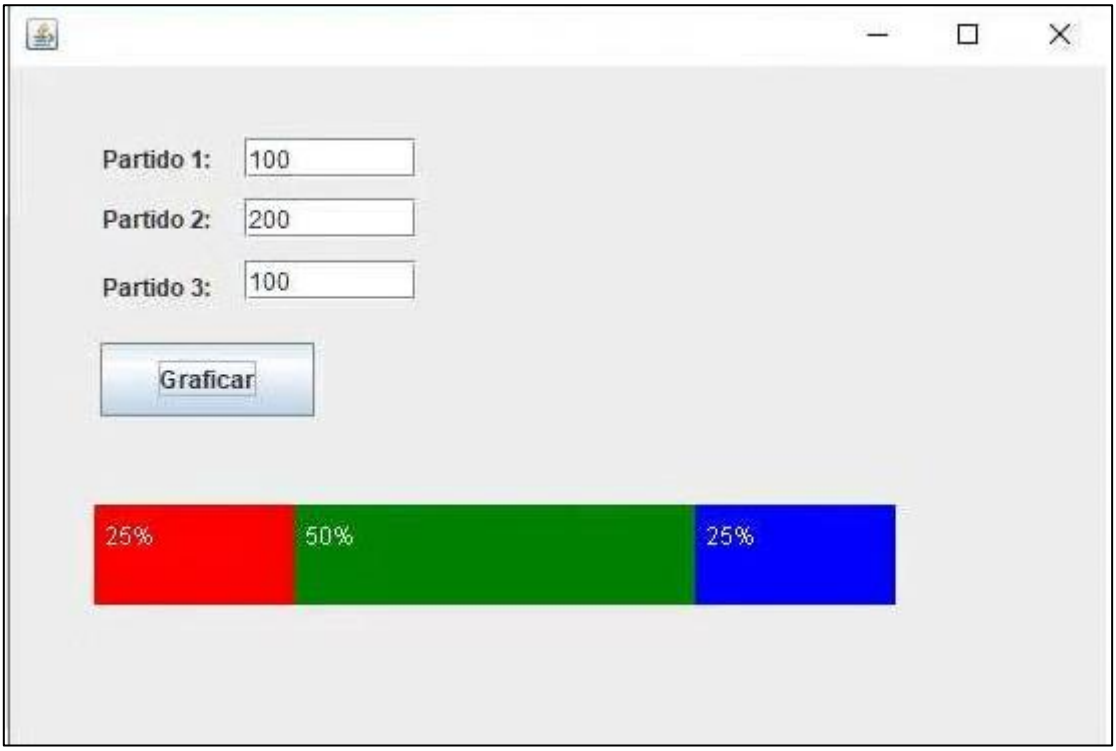
```
                        if(v1>v2 && v1>v3) {
                                max=v1;
                        }else {
                                if(v2>v3) {
                                        max=v2;
                                }else {
                                        max=v3;
                                }
                        }
                        por1=(v1*100/max);
                        por2=(v2*100/max);
                        por3=(v3*100/max);
                        repaint();
                }

        });
        btnNewButton.setBounds(99, 150, 89, 23);
        contentPane.add(btnNewButton);
    }

    @Override
    public void paint(Graphics g) {
        // TODO Auto-generated method stub
        super.paint(g);

        g.setFont(new Font("Arial", Font.BOLD, 12));
        g.setColor(Color.red);
        g.drawString("Partido 1:",50,250);
        g.fillRect(120,230, 300*por1/100, 40);
        g.setColor(Color.green);
        g.drawString("Partido 2:",50,300);
        g.fillRect(120,285, 300*por2/100, 40);
        g.setColor(Color.blue);
        g.drawString("Partido 3:",50,350);
        g.fillRect(120,340, 300*por3/100, 40);
    }
}
```

Ahora realiza la correspondiente modificación:

Capítulo 181.- Gráficos estadísticos – 3

Problema propuesto:

Implementar un gráfico estadístico de tipo "Barra Porcentual":



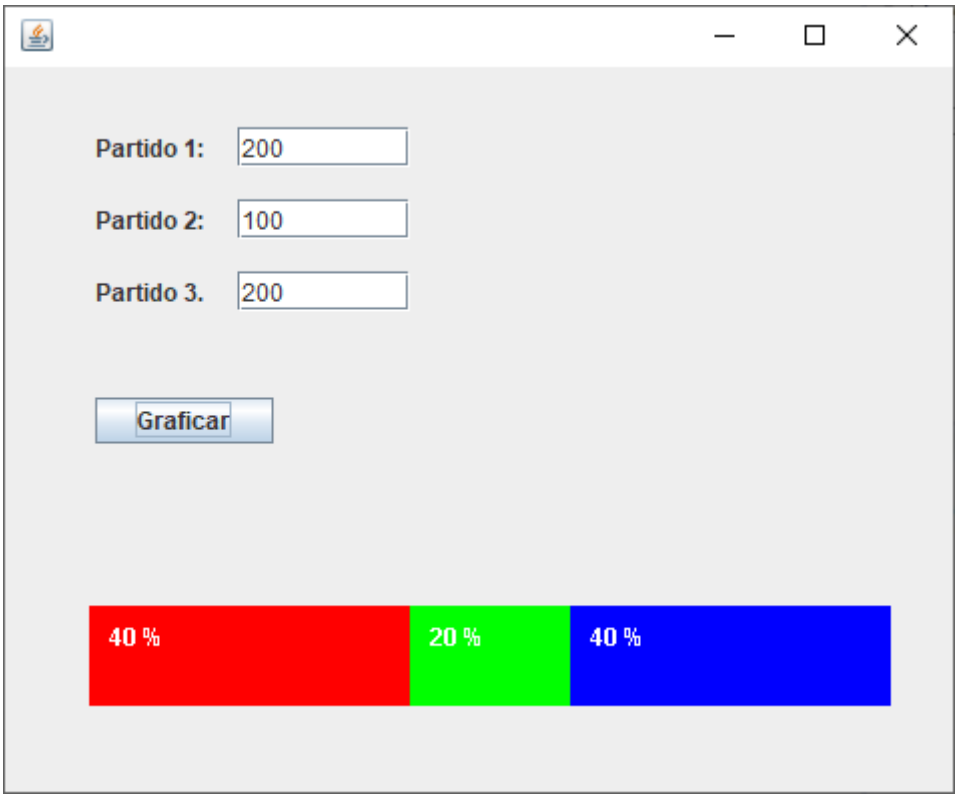
Este será el código:

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Font;
import java.awt.Graphics;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class GraficoBarraPorcentual extends JFrame {

	private JPanel contentPane;
	private int v1, v2, v3;
	private int suma;
	private int por1, por2, por3;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
```

```
                        try {
                                GraficoBarraPorcentual frame = new
GraficoBarraPorcentual();
                                frame.setVisible(true);
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                }
        });
    }

    /**
     * Create the frame.
     */
    public GraficoBarraPorcentual() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 489, 401);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblNewLabel = new JLabel("Partido 1:");
        lblNewLabel.setBounds(45, 33, 91, 14);
        contentPane.add(lblNewLabel);

        JLabel lblNewLabel_1 = new JLabel("Partido 2:");
        lblNewLabel_1.setBounds(45, 69, 74, 14);
        contentPane.add(lblNewLabel_1);

        JLabel lblNewLabel_2 = new JLabel("Partido 3.");
        lblNewLabel_2.setBounds(45, 105, 91, 14);
        contentPane.add(lblNewLabel_2);

        JTextField tf1 = new JTextField();
        tf1.setBounds(116, 30, 86, 20);
        contentPane.add(tf1);
        tf1.setColumns(10);

        JTextField tf2 = new JTextField();
        tf2.setBounds(116, 66, 86, 20);
        contentPane.add(tf2);
        tf2.setColumns(10);

        JTextField tf3 = new JTextField();
        tf3.setBounds(116, 102, 86, 20);
        contentPane.add(tf3);
        tf3.setColumns(10);

        JButton btnNewButton = new JButton("Graficar");
        btnNewButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                v1=Integer.parseInt(tf1.getText());
                v2=Integer.parseInt(tf2.getText());
                v3=Integer.parseInt(tf3.getText());
                suma=v1+v2+v3;
                por1=v1*100/suma;
                por2=v2*100/suma;
```

```
                        por3=v3*100/suma;
                        repaint();
                    }
                });
                btnNewButton.setBounds(45, 165, 89, 23);
                contentPane.add(btnNewButton);
        }

        @Override
        public void paint(Graphics g) {
                // TODO Auto-generated method stub
                super.paint(g);
                g.setColor(Color.red);
                g.fillRect(50,300,400*por1/100,50);
                g.setColor(Color.green);
                g.fillRect(50+(400*por1/100),300,400*por2/100,50);
                g.setColor(Color.blue);
                g.fillRect(50+(400*por1/100)+(400*por2/100),300,400*por3/100,
50);
                g.setColor(Color.white);
                g.setFont(new Font("Arial", Font.BOLD, 12));
                String men1=por1+" %";
                g.drawString(men1, 60, 320);
                String men2=por2+" %";
                g.drawString(men2, 60+(400*por1/100),320);
                String men3=por3+" %";
                g.drawString(men3, 60+(400*por1/100)+(400*por2/100),320);
        }
}
```

Vamos a ejecutar con otros valores:



Capítulo 182.- Estructura selectiva switch – 1

La estructura selectiva switch nos puede remplazar en algunas circunstancias una serie de if anidados, como veremos no es en todos los casos, solo en los casos que se verifican igualdades con ciertos valores.

Problema:

Simular el tirado de un dado 1000 veces. Mostrar luego la cantidad de veces que salieron cada valor. Resolverlo empelando tanto el if como con switch.

Ejemplo con if:

```
public class TirandoDados {

    public static void main(String[] args) {
        int lado1=0;
        int lado2=0;
        int lado3=0;
        int lado4=0;
        int lado5=0;
        int lado6=0;
        for(int f=1; f<=1000; f++) {
            int dado=1+(int)(Math.random()*6);
            if(dado==1) {
                lado1++;
            }else if (dado==2){
                lado2++;
            }else if (dado==3) {
                lado3++;
            }else if (dado==4) {
                lado4++;
            }else if (dado==5) {
                lado5++;
            }else if (dado==6) {
                lado6++;
            }
        }
        System.out.println("Cantidad de veces que ha salido el 1: "+lado1);
        System.out.println("Cantidad de veces que ha salido el 2: "+lado2);
        System.out.println("Cantidad de veces que ha salido el 3: "+lado3);
        System.out.println("Cantidad de veces que ha salido el 4: "+lado4);
        System.out.println("Cantidad de veces que ha salido el 5: "+lado5);
        System.out.println("Cantidad de veces que ha salido el 6: "+lado6);
    }
}
```

Si ejecutamos este será el resultado:

```
Cantidad de veces que ha salido el 1: 169
Cantidad de veces que ha salido el 2: 174
Cantidad de veces que ha salido el 3: 161
Cantidad de veces que ha salido el 4: 156
Cantidad de veces que ha salido el 5: 167
Cantidad de veces que ha salido el 6: 173
```

Ejemplo con switch:

```
public class TirandoDados {

    public static void main(String[] args) {
        int lado1=0;
        int lado2=0;
        int lado3=0;
        int lado4=0;
        int lado5=0;
        int lado6=0;
        for(int f=1; f<=1000; f++) {
            int dado=1+(int)(Math.random()*6);
            switch (dado) {
            case 1: lado1++;
                    break;
            case 2: lado2++;
                    break;
            case 3: lado3++;
                    break;
            case 4: lado4++;
                    break;
            case 5: lado5++;
                    break;
            case 6: lado6++;
                    break;
            }
        }
        System.out.println("Cantidad de veces que ha salido el 1: "+lado1);
        System.out.println("Cantidad de veces que ha salido el 2: "+lado2);
        System.out.println("Cantidad de veces que ha salido el 3: "+lado3);
        System.out.println("Cantidad de veces que ha salido el 4: "+lado4);
        System.out.println("Cantidad de veces que ha salido el 5: "+lado5);
        System.out.println("Cantidad de veces que ha salido el 6: "+lado6);
    }
}
```

Este será el resultado:

```
Cantidad de veces que ha salido el 1: 182
Cantidad de veces que ha salido el 2: 166
Cantidad de veces que ha salido el 3: 172
Cantidad de veces que ha salido el 4: 144
Cantidad de veces que ha salido el 5: 174
Cantidad de veces que ha salido el 6: 162
```

Te propongo el siguiente programa utilizando switch:

Este es el código:

```
import java.awt.EventQueue;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.Toolkit;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class Dado extends JFrame {

	private JPanel contentPane;
	private int lado1;
	private int lado2;
	private int lado3;
	private int lado4;
	private int lado5;
	private int lado6;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Dado frame = new Dado();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}
```

```
/**
 * Create the frame.
 */
public Dado() {
	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	setBounds(100, 100, 628, 274);
	contentPane = new JPanel();
	contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

	setContentPane(contentPane);
	contentPane.setLayout(null);

	JButton btnNewButton = new JButton("Tirar mil veces un dado");
	btnNewButton.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e) {
			lado1=0;
			lado2=0;
			lado3=0;
			lado4=0;
			lado5=0;
			lado6=0;
			for(int f=1; f<=1000; f++) {
				int dado=1+(int)(Math.random()*6);
				switch (dado) {
				case 1: lado1++;
					break;
				case 2: lado2++;
					break;
				case 3: lado3++;
					break;
				case 4: lado4++;
					break;
				case 5: lado5++;
					break;
				case 6: lado6++;
					break;
				}
			}
			repaint();
		}
	});
	btnNewButton.setBounds(226, 201, 202, 23);
	contentPane.add(btnNewButton);
}

@Override
public void paint(Graphics g) {
	// TODO Auto-generated method stub
	super.paint(g);
	Toolkit t=Toolkit.getDefaultToolkit();
	Image image1=t.getImage("cara1.jpg");
	g.drawImage(image1, 50, 50,this);
	Image image2=t.getImage("cara2.jpg");
	g.drawImage(image2, 140, 50,this);
	Image image3=t.getImage("cara3.jpg");
	g.drawImage(image3, 230, 50,this);
	Image image4=t.getImage("cara4.jpg");
	g.drawImage(image4, 320, 50,this);
	Image image5=t.getImage("cara5.jpg");
```

```
            g.drawImage(image5, 410, 50,this);
            Image image6=t.getImage("cara6.jpg");
            g.drawImage(image6, 500, 50,this);

            g.setFont(new Font("Courier", Font.BOLD, 20));
            g.drawString(String.valueOf(lado1), 70, 150);
            g.drawString(String.valueOf(lado2), 160, 150);
            g.drawString(String.valueOf(lado3), 250, 150);
            g.drawString(String.valueOf(lado4), 340, 150);
            g.drawString(String.valueOf(lado5), 430, 150);
            g.drawString(String.valueOf(lado6), 520, 150);
        }
}
```

Capítulo 183.- Estructura selectiva switch – 2

Así como una instrucción if tiene en forma opcional el bloque else, el comando switch tiene en forma opcional el bloque 'default'.

Problema:

Generar 100 valores aleatorios comprendidos entre 1 y 10. Contar cuantos se generaron con los valores 1, 5, 10 y los que no son ni 1, 5, 10.

Este será el código:

```
public class PruebaSwitch {

    public static void main(String[] args) {
        int cont1=0;
        int cont5=0;
        int cont10=0;
        int otros=0;
        for (int f=1; f<=100; f++) {
            int valor=1+(int)(Math.random()*10);
            switch (valor){
            case 1: cont1++;
                    break;
            case 5: cont5++;
                    break;
            case 10: cont10++;
                    break;
            default:otros++;
            }
        }
        System.out.println("El valor 1: "+cont1);
        System.out.println("El valor 5: "+cont5);
        System.out.println("El valor 10: "+cont10);
        System.out.println("El resto de valores: "+otros);
    }
}
```

Si ejecutamos este puede ser uno de los resultados:

```
El valor 1: 9
El valor 5: 15
El valor 10: 10
El resto de valores: 66
```

Capítulo 184.- Estructura selectiva switch – 3

Problema:

Simular el tirado de un dado mis veces. Mostrar luego la cantidad de veces que salieron valores pares e impares.

Este será el resultado:

```
public class PruebaSwitch {

    public static void main(String[] ar) {
        int pares=0;
        int impares=0;
        for(int f=1;f<=1000;f++) {
            int dado=1+(int)(Math.random()*6);
            switch (dado) {
            case 1:impares++;
            case 2:pares++;
            case 3:impares++;
            case 4:pares++;
            case 5:impares++;
            case 6:pares++;
            }
        }
        System.out.println("Cantidad de veces que salieron pares:"+pares);
        System.out.println("Cantidad de veces que salieron impares:"+impares);
    }
}
```

Otra posible solución:

```
public class Dados {

    public static void main(String[] args) {
        int pares=0;
        int impares=0;
        for(int f=1; f<=1000; f++) {
            int dado=1+(int)(Math.random()*6);
            int valor=dado%2;
            switch(valor){
            case 0: pares++;
                    break;
            case 1: impares++;
                    break;
            }
        }
        System.out.println("Cantidad de veces que salieron pares: "+pares);
        System.out.println("Cantidad de veces que salieron impares: "+impares);
    }
}
```

Este puede ser un resultado:

```
Cantidad de veces que salieron pares: 487
Cantidad de veces que salieron impares: 513
```

Capítulo 185.- Estructura selectiva switch – 4

Problema:

Almacenar en un vector los nombres del días que trabaja un empleado:

String[] trabajo = {"lunes", "miércoles", "sábado", "domingo"};

Contar la cantidad de días que pertenecen a días laborables y fin de semana.

```
public class PruebaSwitch {

    public static void main(String[] args) {
        String[] trabajo = {"lunes", "miércoles", "sábado", "domingo"};
        int laborables=0;
        int finsemana=0;
        for (int f=0; f<trabajo.length; f++){
            switch(trabajo[f]) {
            case "lunes":
            case "martes":
            case "miércoles":
            case "jueves":
            case "viernes": laborables++;
                            break;
            case "sábado":
            case "domingo": finsemana++;
                            break;

            }
        }
        System.out.println("Cantidad de días que trabajo en días hábiles:"+laborables);
        System.out.println("Cantidad de días que trabajo en días fin de semana:"+finsemana);
    }
}
```

Si ejecutamos este será el resultado:

```
Cantidad de días que trabajo en días hábiles:2
Cantidad de días que trabajo en días fin de semana:2
```

Capítulo 186.- Estructura repetitiva 'for' adaptada para recorrer colecciones – 1

Estructura for clásica:

Cuando tenemos que iterar por los elementos de una colección (hasta ahora hemos visto los arreglos) accedemos a los mismos por medio de un subíndice que normalmente corresponde al contador de un for.

```
int[] vec = { 10, 20, 30 };


for (int f = 0; f < vec.length; f++) {
	System.out.println(vec[f]);
}
```

Estructura for para colecciones:

```
for (int elemento : vec) {
	System.out.println(elemento);
}
```

Veamos un ejemplo:

```
public class PruebaFor {

    public static void main(String[] args) {
        int[] vec= {10, 20, 30};

        int suma=0;
        for(int elemento: vec) {
            System.out.print(elemento+"-");
            suma=suma+elemento;
        }
        System.out.println();
        System.out.println("La suma de todos sus elementos es: "+suma);
    }
}
```

Si ejecutamos este será el resultado:

```
10-20-30-
La suma de todos sus elementos es: 60
```

Capítulo 187.- Estructura repetitiva 'for' adaptada para recorrer colecciones – 2

Problema:

Se desea almacenar los sueldos de los operarios. Cuando se ejecuta el programa se debe pedir la cantidad de sueldo a ingresar. Luego crear un arreglo con dicho tamaño.

Imprimir todos los sueldos ingresados y mostrar el mayor de ellos.

```
1  import java.util.Scanner;
2  public class Vectores {
3      private Scanner teclado;
4      private int [] sueldos;
5
6      public Vectores() {
7          teclado=new Scanner(System.in);
8          System.out.print("Ingrese el número de salarios:" );
9          int numeroSalarios=teclado.nextInt();
10         sueldos= new int [numeroSalarios];
11         for(int x=0; x<numeroSalarios; x++) {
12             System.out.print("Ingrese el sueldo número "+(x+1)+":");
13             sueldos[x]=teclado.nextInt();
14         }
15     }
16
17     public void imprimir() {
18         System.out.println("Listado de sueldos");
19         for(int elemento: sueldos) {
20             System.out.print(elemento+"-");
21         }
22         System.out.println();
23     }
24
25     public void sueldoMayor() {
26         int max=0;
27         for(int elemento: sueldos) {
28             System.out.print(elemento+"-");
29             if (elemento>max) {
30                 max=elemento;
31             }
32         }
33         System.out.println("El sueldo mayor es "+max);
34     }
35
36     public static void main(String[] args) {
37         Vectores vectores1 =new Vectores();
38         vectores1.imprimir();
39         vectores1.sueldoMayor();
40     }
41 }
```

Si ejecutamos este será el resultado:

```
Ingrese el número de salarios:5
Ingrese el sueldo número 1:1200
Ingrese el sueldo número 2:1400
Ingrese el sueldo número 3:850
Ingrese el sueldo número 4:970
Ingrese el sueldo número 5:1150
Listado de sueldos
1200-1400-850-970-1150-
1200-1400-850-970-1150-El sueldo mayor es 1400
```

Capítulo 188.- estructura repetitiva 'for' adaptada para recorrer colecciones – 3

Recorrer una matiz

También podemos utilizar la nueva sintaxis para recorrer un arreglo de dos dimensiones.

Problema

Crear una matriz de n * m filas (cargar n y m por teclado) imprimir la matriz completa utilizando la estructura 'for' que recorre colecciones.

```
import java.util.Scanner;
public class RecorridoMatriz {
    private int [][] mat;
    private Scanner teclado;

    public RecorridoMatriz() {
        teclado=new Scanner(System.in);
        System.out.print("Cuantas filas tiene la matriz:");
        int filas=teclado.nextInt();
        System.out.print("Cuantas columnas tiene la matriz");
        int columnas=teclado.nextInt();
        mat=new int[filas][columnas];
        for(int f=0; f<mat.length; f++) {
            for(int c=0; c<mat[f].length; c++) {
                System.out.print("Ingrese componente:");
                mat[f][c]=teclado.nextInt();
            }
        }
    }

    public void imprimir() {
        for (int[] fila :mat) {
            for(int elemento :fila) {
                System.out.print(elemento+"-");
            }
            System.out.println();
        }
    }


    public static void main(String[] args) {
        RecorridoMatriz reco=new RecorridoMatriz();
        reco.imprimir();
    }
}
```

Si ejecutamos este será el resultado:

```
Cuantas filas tiene la matriz:3
Cuantas columnas tiene la matriz4
Ingrese componente:1
Ingrese componente:2
Ingrese componente:3
Ingrese componente:4
Ingrese componente:5
Ingrese componente:6
Ingrese componente:7
Ingrese componente:8
Ingrese componente:9
```

```
Ingrese componente:10
Ingrese componente:11
Ingrese componente:12
1-2-3-4-
5-6-7-8-
9-10-11-12-
```

Capítulo 189.- Estructura repetitiva 'for' adaptada para recorrer colecciones – 4

Recorrer un arreglo con elementos de tipo objeto.

Problema

Crear un proyecto y dentro del mismo crear dos clases. La primera clase se debe llamar 'Carta' y definir los atributos palo y numero. Por otro lado declarar la clase llamada 'Mazo' que contenga un arreglo de 6 elementos de tipo 'Carta'.

Imprimir todas las cartas.

Imprimir una carta al azar entre las 6 cartas.

La clase Carta:

```
public class Carta {
    private int numero;
    private String palo;

    public Carta(int numero, String palo) {
        this.numero=numero;
        this.palo=palo;
    }

    public void imprimir() {
        System.out.println(numero+" "+palo);
    }
}
```

La clase Mazo:

```
public class Mazo {
    private Carta[] cartas;

    public Mazo() {
        cartas=new Carta[6];
        cartas[0]=new Carta(1, "Trebol");
        cartas[1]=new Carta(2, "Trebol");
        cartas[2]=new Carta(3, "Trebol");
        cartas[3]=new Carta(4, "Trebol");
        cartas[4]=new Carta(5, "Trebol");
        cartas[5]=new Carta(6, "Trebol");
    }

    public void imprimir() {
        System.out.println("Listado completo de mazo de cartas.");
        for(Carta carta: cartas) {
            carta.imprimir();
        }
    }

    public void imprimirUnaAlAzar() {
        System.out.println("Una carta elegida al azar.");
        cartas[(int)(Math.random()*6)].imprimir();
    }
```

```
    public static void main(String[] args) {
        Mazo m=new Mazo();
        m.imprimir();
        m.imprimirUnaAlAzar();

    }
}
```

Si ejecutamos este puede ser uno de los resultados:

```
Listado completo de mazo de cartas.
1 Trebol
2 Trebol
3 Trebol
4 Trebol
5 Trebol
6 Trebol
Una carta elegida al azar.
3 Trebol
```

Capítulo 190.- Atributos estáticos de una clase

En los primeros conceptos de POO (Programación Orientada a Objetos) dijimos que los atributos que se definen en una clase reservan espacio en forma independiente para cada instancia de la misma.

A diferencia de los anteriores los atributos estáticos tienen un comportamiento muy distinto a los atributos vistos hasta el momento. Un atributo estático reserva espacio para el mismo indistintamente que definamos un objeto de dicha clase. En caso de crear varios objetos de dicha clase todas las instancias acceden al mismo atributo estático.

```
public class Matematica {
    public static float PI = 3.1416f;
}



public class Prueba {
    public static void main(String[] ar) {
        System.out.println(Matematica.PI);
    }
}
```

Problema

Definir un atributo estático que almacene la cantidad de objetos creados en dicha clase. Definir la misma clase Persona con los atributos nombre, edad y el atributo estático cantidad:

```
public class Persona {
    private String nombre;
    private int edad;
    // La variable cantidad será compartida con todas las instancias.
    public static int cantidad;

    public Persona(String nombre, int edad) {
        this.nombre=nombre;
        this.edad=edad;
        cantidad++;
    }

    public void imprimir() {
        System.out.println("Nombre: "+nombre+" Edad: "+edad);
    }

    public static void main(String[] args) {
        System.out.println("Cantidad es igual a "+Persona.cantidad);
        Persona persona1=new Persona("Juan", 21);
        persona1.imprimir();
        System.out.println("Cantidad es igual a "+Persona.cantidad);
        System.out.println("Cantidad recuperado por mediación de una instancia "+persona1.cantidad);
        Persona persona2=new Persona("Ana", 25);
        persona2.imprimir();
        System.out.println("Cantidad es igual a "+Persona.cantidad);
        Persona persona3=new Persona("Luis", 18);
        System.out.println("Cantidad recuperado por mediación de una instancia "+persona1.cantidad);
    }
}
```

Vamos a ejecutar:

```
Cantidad es igual a 0
Nombre: Juan Edad: 21
Cantidad es igual a 1
Cantidad recuperado por mediación de una instancia 1
Nombre: Ana Edad: 25
Cantidad es igual a 2
Cantidad recuperado por mediación de una instancia 3
```

Capítulo 191.- Métodos estáticos de una clase – 1

Así como una clase puede tener atributos estáticos, Java también permite definir métodos estáticos que se crean independientemente de la definición de objetos. Un método estático puede llamarse sin tener que crear un objeto de dicha clase.

Igual que los atributos estáticos, un método estático tiene ciertas restricciones:

- No puede acceder a los atributos de la clase (salvo que sean estáticos)
- No puede utilizar el operador this, ya que este método se puede llamar sin tener que crear un objeto de la clase.
- Puede llamar a otro método siempre y cuando sea estático.
- Un método estático es lo más parecido a lo que son las funciones en los lenguajes estructurados (con la diferencia que se encuentran encapsulado en una clase.

Si recordamos cada vez que creamos un programa en Java debemos especificar el método main:

```
public static void main(String[] args)
```

El método main es estático para que la máquina virtual de Java pueda llamarlo directamente sin tener que crear un objeto de la clase que lo contiene.

Problema

Implementar una clase llamada Operación. Definir dos métodos estáticos que permitan sumar y restar dos valores enteros.
Creamos una clase llamada Operación, esta no contiene el método main:

```
public class Operacion {

    public static int sumar(int x1, int x2) {
        int s=x1+x2;
        return s;
    }

    public static int restar(int x1, int x2) {
        int r=x1-x2;
        return r;
    }

}
```

Creamos una segunda clase llamada PruebaOperación, este contiene el método main:

```
public class PruebaOperacion {

    public static void main(String[] args) {
        System.out.print("La suma de 2+4 es:");
        System.out.println(Operacion.sumar(2, 4));
        System.out.print("La resta de 6-2 es:");
        System.out.println(Operacion.restar(6, 2));
    }
}
```

Ejecutamos y este será el resultado:

```
La suma de 2+4 es:6
La resta de 6-2 es:4
```

En ningún momento hay que instanciar un objeto de la clase Operación.

Capítulo 192.- Métodos estáticos de una clase – 2

Problema:

Declarar una clase Persona con los atributos nombre y edad. Definir un método estático que reciba como parámetro dos objetos de la clase Persona y me retorne la que tiene edad mayor, si son iguales retorne cualquiera de los dos.

Creamos una clase llamada Persona, esta no contiene el método main:

```
public class Persona {
    private String nombre;
    private int edad;

    public Persona(String nombre, int edad) {
        this.nombre=nombre;
        this.edad=edad;
    }

    public void imprimir() {
        System.out.println(nombre+" de "+edad+" años.");
    }

    public static Persona Mayor(Persona per1, Persona per2) {
        if(per1.edad>per2.edad) {
            return per1;
        }else {
            return per2;
        }
    }
}
```

Creamos una clase llamada PruebaPersona, esta si contiene el método main:

```
public class PruebaPersona {

    public static void main(String[] args) {
        Persona per1=new Persona("Juan", 25);
        Persona per2=new Persona("Luis", 18);
        per1.imprimir();
        per2.imprimir();
        Persona personaMayor=Persona.Mayor(per1, per2);
        System.out.print("La persona mayor es ");
        personaMayor.imprimir();
    }
}
```

Si ejecutamos este será el resultado:

```
Juan de 25 años.
Luis de 18 años.
La persona mayor es Juan de 25 años.
```

Esta sería otra posible solución:

```
public class PruebaPersona {

    public static void main(String[] args) {
        Persona per1=new Persona("Juan", 25);
        Persona per2=new Persona("Luis", 18);
        per1.imprimir();
        per2.imprimir();
        System.out.print("La persona mayor es ");
        Persona.Mayor(per1, per2).imprimir();
    }
}
```

Capítulo 193.- Métodos estáticos de una clase – 3

La clase Math tiene una gran cantidad de métodos estáticos:

```
 sin, cos, tan, asin, acos, atan, toRadians, toDegrees, exp, log, log10, sqrt,
cbrt,IEEEremainder, ceil, floor, rint, atan2, pow, round, random, addExact,
subtractExact,multiplyExact, incrementExact, decrementExact, negateExact,
toIntExact, multiplyFull,multiplyHigh, floorDiv, floorMod, abs, max, min, fma,
ulp, signum, sinh, cosh, tanh,hypot, expm1, log1p, copySign, getExponent,
nextAfter, nextUp, nextDown, scalb,
powerOfTwoD, powerOfTwoF
```

La clase String tiene los métodos estáticos:

```
format, valueOf, copyValueOf, join.
```

La clase Integer tiene los métodos estáticos:

```
 parseInt, parseUnsignedInt, hashCode, getInteger, decode, compare,
compareUnsigned, toUnsignedLong, divideUnsigned, remainderUnsigned,
highestOneBit, lowestOneBit, numberOfLeadingZeros, numberOfTrailingZeros,
bitCount, rotateLeft, rotateRight, reverse, signum, reverseBytes,
sum, max, min, toString, toStringUTF16, toUnsignedString,
toHexString, toOctalString, toBinaryString, toUnsignedString0
```

Vamos a realizar el siguiente ejemplo:

```
public class MetodosEstaticos {

	public static void main(String[] args) {
		String cadena1="255";
		// Convertir en número entero
		int x1=Integer.parseInt(cadena1);
		System.out.println(x1);
		// Convetir en formato binario
		System.out.println(Integer.toBinaryString(x1));
		// Convertir en formato Hexadecimal
		System.out.println(Integer.toHexString(x1));
		// Convertir en formato octal
		System.out.println(Integer.toOctalString(x1));

		// Convertirlo en cadena
		String cadena2=String.valueOf(x1);
		System.out.println(cadena2);
		// Valor float con 2 decimales y int imprimir
		float valor1=10.3556f;
		int x2=300;
		System.out.println(String.format("La variable valor1 tiene %.2f
y x2 tiene %d", valor1, x2));
	}
}
```

Este será el resultado:

```
255
11111111
ff
377
255
La variable valor1 tiene 10,36 y x2 tiene 300
```

Capítulo 194.- Definición de constantes en Java mediante la palabra clave final – 1

Hasta ahora hemos visto que cuando definimos un atributo su valor puede ser cambiado en cualquier momento. Hay situaciones que ciertos datos a almacenar se los debe inicializar y nunca más van a cambiar, en estos casos debemos emplear contantes.

Problema

Implementar una clase llamada Circulo. Definir una constante donde se debe almacenar el valor PI (relación entre la longitud de una circunferencia y diámetro). Además definir otro atributo donde almacenar el radio del círculo.

Al constructor debe llegar el radio y otro método debe retornar el perímetro.

```
public class Circulo {
    // Creamos una constante llamada PI
    private final double PI=3.1416;
    private double radio;

    public Circulo(double radio) {
        this.radio=radio;
    }

    public double perimetro() {
        return 2*PI*radio;
    }

    public static void main(String[] args) {
        Circulo circulo1=new Circulo(5);
        System.out.println("El perímetro de 5 es "+ circulo1.perimetro());
    }
}
```

Si ejecutamos este será el resultado:

```
El perímetro de 5 es 31.416
```

Capítulo 195.- Definición de constantes en Java mediante la palabra clave final – 2

Problema

Implementar una clase llamada CajaDeAhorro. Se debe almacenar el número de documento del titular y el importe depositado. Una vez que se carga el documento no permitir su cambio.

Plantear dos constructores uno que lleve el documento del titular y el importe a depositar, y un segundo constructor que solo lleve el documento.

```
public class CajaDeAhorro {
    private final String documento;
    private int importe;

    public CajaDeAhorro(String documento, int importe) {
        this.documento=documento;
        this.importe=importe;
    }

    public CajaDeAhorro(String documento) {
        this.documento=documento;
    }

    public void imprimir() {
        System.out.println("Documento: "+documento+"  Saldo: "+importe);
    }

    public static void main(String[] args) {
        CajaDeAhorro caja1=new CajaDeAhorro("12345678", 100);
        CajaDeAhorro caja2=new CajaDeAhorro("11111111");
        caja1.imprimir();
        caja2.imprimir();
    }
}
```

Si ejecutamos este será el resultado:

```
Documento: 12345678  Saldo: 100
Documento: 11111111  Saldo: 0
```

Si en otra parte del programa intentamos cambiar el valor del documento nos mostrará un menaje de error.

Capítulo 196.- Definición de constantes en Java mediante la palabra clave final – 3

Podemos definir una constante con el modificador 'static', recordemos que con este modificador hace que se reserve espacio para la constante de forma independiente a que se creen o no instancias de dicha clase.

Si definimos una constante con el modificador 'static' estamos obligados a inicializar su valor cuando lo declaramos y no podemos hacerlo en el constructor.

Problema

Plantear una clase Persona con los atributos nombre y edad. Implementar un método que retorne si es mayor de edad (almacenar en una constante estática el valor 18 que representa la mayoría de edad).

```
public class Persona {
    private String nombre;
    private int edad;
    private static final int mayor=18;

    public Persona(String nombre, int edad) {
        this.nombre=nombre;
        this.edad=edad;
    }

    public boolean mayor() {
        if(edad>mayor) {
            return true;
        }else {
            return false;
        }
    }

    public static void main(String[] args) {
        Persona persona1 = new Persona("Juan", 16);
        Persona persona2 = new Persona("Luis", 25);
        if(persona1.mayor()) {
            System.out.println("La persona: "+persona1.nombre+" es mayor de edad.");
        }else{
            System.out.println("La persona: "+persona1.nombre+" no si es mayor de edad.");
        }
        if(persona2.mayor()) {
            System.out.println("La persona: "+persona2.nombre+" es mayor de edad.");
        }else{
            System.out.println("La persona: "+persona2.nombre+" no si es mayor de edad.");
        }
    }
}
```

Si ejecutamos este será el resultado:

```
La persona: Juan no si es mayor de edad.
La persona: Luis es mayor de edad.
```

Capítulo 197.- Definición de constantes en Java mediante la palabra clave final – 4

Existen clases en los paquetes que suministra Java que definen constantes.

Clase: Math, Integer, Color, etc.

```
public class Constantes {

    public static void main(String[] args) {
        System.out.println(Math.PI);
    }
}
```

Manteniendo pulsada al tecla Ctrol hacemos clic sobre PI.

```
 /**
  * The {@code double} value that is closer than any other to
  * <i>pi</i> (&pi;), the ratio of the circumference of a circle to
  * its diameter.
  */
 public static final double PI = 3.141592653589793;
```

Podemos observar que esta constante es de tipo double.
Si la ejecutamos observaremos el valor que tiene asignado:

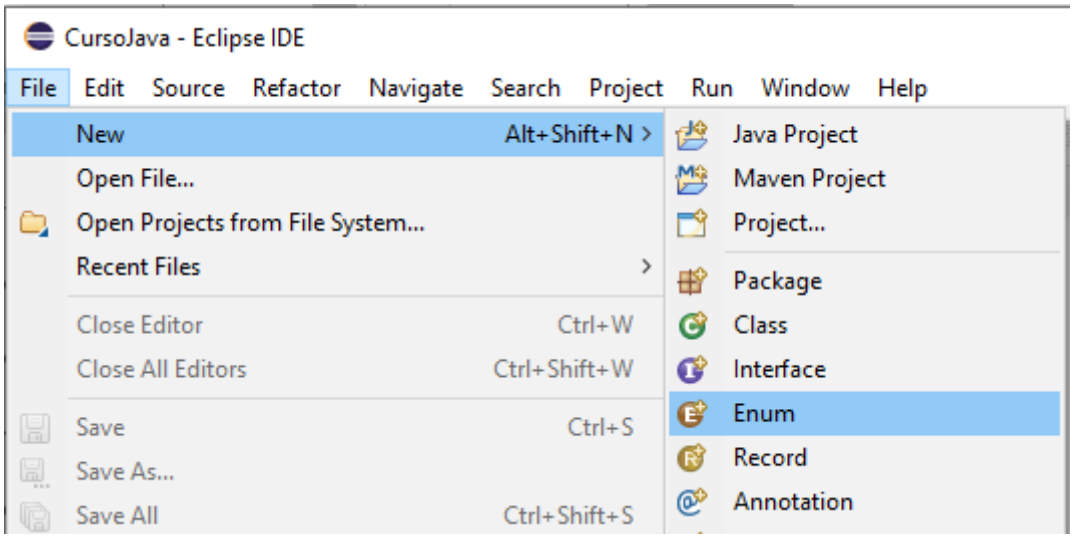
```
3.141592653589793
import java.awt.Color;

public class Constantes {

    public static void main(String[] args) {
        // El valor de PI
        System.out.println(Math.PI);
        // La base de los logaritmos naturales.
        System.out.println(Math.E);
        // El valor máximo de los enteros.
        System.out.println(Integer.MAX_VALUE);
        // El valor mínimo de los enteros.
        System.out.println(Integer.MIN_VALUE);
        // La cantidad de bytes que requiere una variable de tipo Integer.
        System.out.println(Integer.BYTES);
        // Lo que requiere una variable de tipo Integer.
        System.out.println(Integer.SIZE);
        //Para inicizlizar un color en este caso el rojo.
        System.out.println(Color.RED);
    }
}
```

Cuando ejecutemos este será el resultado:

```
3.141592653589793
2.718281828459045
2147483647
-2147483648
4
32
java.awt.Color[r=255,g=0,b=0]
```

Capítulo 198.- Declaración de tipos enum – 1

Básicamente en un conjunto de constantes que se las asocia a un tipo 'enum'. En lugar de class utilizamos la palabra clave 'enum' y entre llaves definimos las contantes.



En lugar del Class seleccionamos Enum.

*Operaciones.java

```
1
2 public enum Operaciones {
3     SUMA, RESTA, MULTIPLICACION, DIVISION
4 }
```

Ahora vamos a crear la clase con su método main:

```
public class PruebaEnum {

    public static void main(String[] args) {
        int valor1=10;
        int valor2=5;
        Operaciones operacion;
        operacion=Operaciones.SUMA;
        if (operacion==Operaciones.SUMA) {
            int suma=valor1+valor2;
            System.out.println("La suma es: "+suma);
        }
        operacion=Operaciones.RESTA;
        if (operacion==Operaciones.RESTA) {
            int resta=valor1-valor2;
            System.out.println("La resta es: "+resta);
        }
        operacion=Operaciones.MULTIPLICACION;
        if (operacion==Operaciones.MULTIPLICACION) {
            int multiplicacion=valor1*valor2;
            System.out.println("La multiplicación es: "+multiplicacion);
        }
```

```
        operacion=Operaciones.DIVISION;
        if (operacion==Operaciones.DIVISION) {
            int division=valor1/valor2;
            System.out.println("La división es: "+division);
        }
    }
}
```

Si ejecutamos este será el resultado:

```
La suma es: 15
La resta es: 5
La multiplicación es: 50
La división es: 2
```

Capítulo 199.- Declaración de tipos enum – 2

El lenguaje Java permite declarar tipos enum dentro de una clase y agregarles el modificador de acceso: public, private, protected.

Problema

Crear un proyecto y dentro del mismo crear dos clases. La primera clase se debe llamar 'Carta', declarar un enum que represente los cuatro palos del mazo y dos atributos uno del tipo de dato enum y el número de carta. Por otro lado declarar una clase llamada 'Mazo' que contenga un arreglo de 8 elementos de tipo 'Carta'. Extraer al azar una carta, mostrarla y según el tipo de palo mostrar la cantidad de puntos que gana.

La clase Carta:

```
public class Carta {
    public enum Palo{
        TREBOL, DIAMANTE, CORAZON, PICA
    }
    private int numero;
    private Palo palo;

    public Carta(int numero, Palo palo) {
        this.numero=numero;
        this.palo=palo;
    }

    public void imprimir() {
        System.out.println(numero+" "+palo.toString().toLowerCase());
    }

    public Palo retornarPalo() {
        return palo;
    }
}
```

La clase Mazo que contiene el método main:

```
public class Mazo {
    private Carta[] cartas;

    public Mazo() {
        cartas = new Carta[8];
        cartas[0]=new Carta(1, Carta.Palo.TREBOL);
        cartas[1]=new Carta(2, Carta.Palo.TREBOL);
        cartas[2]=new Carta(1, Carta.Palo.DIAMANTE);
        cartas[3]=new Carta(2, Carta.Palo.DIAMANTE);
        cartas[4]=new Carta(1, Carta.Palo.PICA);
        cartas[5]=new Carta(2, Carta.Palo.PICA);
        cartas[6]=new Carta(1, Carta.Palo.CORAZON);
        cartas[7]=new Carta(2, Carta.Palo.CORAZON);
    }

    public void imprimir() {
        System.out.println("Listado completo del mazo de cartas");
        for(Carta carta: cartas) {
            carta.imprimir();
        }
    }
```

```
    public void sacarCartaAzar() {
        System.out.println("Una de las cartas al azar");
        Carta carta=cartas[(int)(Math.random()*8)];
        carta.imprimir();
        switch(carta.retornarPalo()) {
        case CORAZON: System.out.println("Gano 4 puntos");
                    break;
        case DIAMANTE: System.out.println("Gano 3 puntos");
                    break;
        case PICA: System.out.println("Gano 2 puntos");
                    break;
        case TREBOL: System.out.println("Gano 1 punto");
                    break;
        }
    }

    public static void main(String[] arg) {
        Mazo mazo1=new Mazo();
        mazo1.imprimir();
        mazo1.sacarCartaAzar();
    }
}
```

Si ejecutamos este será el resultado:

```
Listado completo del mazo de cartas
1 trebol
2 trebol
1 diamante
2 diamante
1 pica
2 pica
1 corazon
2 corazon
Una de las cartas al azar
1 diamante
Gano 3 puntos
```

Capítulo 200.- Declaración de tipos enum – 3

En algunas situaciones podemos crear tipos enum que además de las constantes tengan asociado otros valores cada constante mediante atributos.

Cuando se declara un enum podemos definir constructores y otros métodos a la misma.

Problema

Declarar un tipo enum llamado 'Mes', asociar para cada constante el número del mes que corresponde.

```
public class PruebaEnum {

    public static void main(String[] args) {
        Mes mes1=Mes.AGOSTO;
        System.out.println(mes1+ " " + mes1.retornarNumero());
        Mes mes2=Mes.ENERO;
        System.out.println(mes2.retornarNumero());
        // Retorna todos los valores unumerados en un vector.
        Mes[] meses=Mes.values();

        System.out.println("Recorremos el arreglo con el for");
        for(int f=0; f<12; f++) {
            System.out.println(meses[f]+" "+meses[f].retornarNumero());
        }

        System.out.println("Recorremos el arreglo con el for adaptado para colecciones");
        for(Mes mes: Mes.values()) {
            System.out.println(mes+" "+mes.retornarNumero());
        }
    }
}
```

Si ejecutamos este será el resultado:

```
AGOSTO 8
1
Recorremos el arreglo con el for
ENERO 1
FEBRERO 2
MARZO 3
ABRIL 4
MAYO 5
JUNIO 6
JULIO 7
AGOSTO 8
SEPTIEMRE 9
OCTUBRE 10
NOVIEMBRE 11
DICIEMBRE 12
Recorremos el arreglo con el for adaptado para colecciones
ENERO 1
FEBRERO 2
MARZO 3
ABRIL 4
MAYO 5
JUNIO 6
JULIO 7
AGOSTO 8
SEPTIEMRE 9
OCTUBRE 10
NOVIEMBRE 11
DICIEMBRE 12
```

Capítulo 201.- Tipos de datos primitivos y clases de envoltura en Java

Recordamos que los tipos de datos primitivos son aquellos que almacena directamente el valor, a diferencia de los tipos de datos referencia que almacena la dirección de memoria donde se almacena el dato (los objetos son tipo de datos referencia).

Los tipos de datos primitivos los podemos agrupar en:

Tipos enteros

Según el valor entero máximo a almacenar podemos elegir entre:

- byte: -128 a 127
- short: -32768 a 32767
- int: –2147483648 a 2147483647
- long: –9223372036854775808 a 9223372036854775807

Imaginemos que tenemos que definir un array de 10000 valores enteros. La elección del tipo entero va a ser importante, teniendo en cuenta que un byte ocupa un byte, un short 2 bytes, un int ocupa 4 bytes y un long ocupa 8 bytes.

Si vamos a almacenar en el array edades por ejemplo, lo más conveniente será utilizar el tipo byte y no long.

Tipos reales

- float
- double

Tipo lógico

- boolean: puede almacenar solo alguno de dos valores (true/false)

Tipo carácter

- char: puede almacenar un único carácter Unicode de 16 bits.

Para cada uno de los tipos de datos primitivos existen una clase de envoltura asociada:

Tipo primitivo	Clase de envoltura
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

Problema:

Imprimir los valores máximos y mínimos que pueden almacenar cada tipo de dato primitivo numérico. Emplear las clases de envoltura para re459uperar dichos valores.

```
public class TiposPrimitivos {

    public static void main(String[] args) {
        System.out.println("Máximo y minimo valor que puede almacenar un tipo de dato byte");
        System.out.println(Byte.MIN_VALUE+" / "+Byte.MAX_VALUE);

        System.out.println("Máximo y minimo valor que puede almacenar un tipo de dato short");
        System.out.println(Short.MIN_VALUE+" / "+Short.MAX_VALUE);

        System.out.println("Máximo y minimo valor que puede almacenar un tipo de dato int");
        System.out.println(Integer.MIN_VALUE+" / "+Integer.MAX_VALUE);

        System.out.println("Máximo y minimo valor que puede almacenar un tipo de dato long");
        System.out.println(Long.MIN_VALUE+" / "+Long.MAX_VALUE);

        System.out.println("Máximo y minimo valor que puede almacenar un tipo de dato float");
        System.out.println(Float.MIN_VALUE+" / "+Float.MAX_VALUE);

        System.out.println("Máximo y minimo valor que puede almacenar un tipo de dato double");
        System.out.println(Double.MIN_VALUE+" / "+Double.MAX_VALUE);

        System.out.println("Máximo y minimo valor que puede almacenar un tipo de dato char");
        System.out.println(Character.MIN_VALUE+" / "+Character.MAX_VALUE);

        byte edad=12;
        short largo=20000;
        int premio=1_000_000; // Para la ayudar a leer un número, lo guiones no se muestran.
        long distancia=1_000_000_000_000_0001; // al final se agrega la letra l.
    }
}
```

Si ejecutamos este será el resultado:

```
Máximo y minimo valor que puede almacenar un tipo de dato byte
-128 / 127
Máximo y minimo valor que puede almacenar un tipo de dato short
-32768 / 32767
Máximo y minimo valor que puede almacenar un tipo de dato int
-2147483648 / 2147483647
Máximo y minimo valor que puede almacenar un tipo de dato long
-9223372036854775808 / 9223372036854775807
Máximo y minimo valor que puede almacenar un tipo de dato float
1.4E-45 / 3.4028235E38
Máximo y minimo valor que puede almacenar un tipo de dato double
4.9E-324 / 1.7976931348623157E308
Máximo y minimo valor que puede almacenar un tipo de dato char
  / ?
```

Capítulo 202.- Clases genéricas

Java permite crear clases que administren distintos tipos de datos.

Las clases genéricas se utilizan mucho para la administración de colecciones de datos (pilas, colas, listas genéricas, árboles, etc.)

Las clases genéricas nos evitan duplicar clases que administran tipos de datos distintos pero implementan algoritmos similares, son una herramienta fundamental para reutilizar código.

Vimos en conceptos anteriores como implementar estructuras dinámicas de tipo pila y dependiendo del problema definíamos una pila de enteros o de String o de cualquier otro tipo de datos.

Si no existiera el concepto genérico en Java deberíamos implementar una clase por cada tipo de datos que administra una pila.

Problema

Implementar una clase Pila que administre cualquier tipo de dato mediante el concepto de genéricos.

Crear luego 4 objetos de la clase Pila y almacenar en la primera pila objetos de la clase Persona, en la segunda objetos de la clase Carta, en la tercera objetos de la clase String y finalmente una cuarta objetos de la clase Integer.

Creamos la clase Carta:

```
public class Carta {
    private int numero;
    private String palo;

    public Carta(int numero, String palo) {
        this.numero=numero;
        this.palo=palo;
    }

    public void imprimir() {
        System.out.println("Num. carta: "+numero+", Palo: "+palo);
    }

}
```

Creamos la clase Persona:

```
public class Persona {
    private String nombre;
    private int edad;

    public Persona(String nombre, int edad) {
        this.nombre=nombre;
        this.edad=edad;
    }

    public void imprimir() {
        System.out.println("Nombe: "+nombre+", Edad: "+edad);
    }
}
```

Creamos la clase Pila:

```
// Creamos la clase genérica Pila.
public class Pila <E> {
    class Nodo {
        E info;
        Nodo sig;
    }

    private Nodo raiz;

    public void insertar(E x) {
        Nodo nuevo=new Nodo();
        nuevo.info=x;
        nuevo.sig=raiz;
        raiz=nuevo;
    }

    public E extraer() {
        if(raiz==null) {
            return null;
        }else {
            E informacion=raiz.info;
            raiz=raiz.sig;
            return informacion;
        }
    }
}
```

Creamos la clase PruebaGenerico que contiene el método main:

```
public class PruebaGenerico {

    public static void main(String[] args) {
        Pila<Persona> pila1=new Pila<Persona>();
        pila1.insertar(new Persona("Juan", 22));
        pila1.insertar(new Persona("Luis", 38));
        pila1.insertar(new Persona("Ana", 45));
        Persona ultimaPersona=pila1.extraer();
        System.out.print("Extraemos el prime relemento de la pila Persona: ");
        ultimaPersona.imprimir();

        Pila<Carta> pila2=new Pila<Carta>();
        pila2.insertar(new Carta(1, "Corazones"));
        pila2.insertar(new Carta(2, "Trebol"));
        pila2.insertar(new Carta(3, "Diamantes"));
        pila2.insertar(new Carta(4, "Rombos"));
        Carta ultimaCarta=pila2.extraer();
        System.out.print("Extraemos el primer elemento de la pila Carta: ");
        ultimaCarta.imprimir();

        Pila<String> pila3=new Pila<String>();
        pila3.insertar(new String("Melocotón"));
        pila3.insertar(new String("Manzana"));
        pila3.insertar(new String("Piña"));
        pila3.insertar(new String("Albaricoque"));
        String ultimoString=pila3.extraer();
        System.out.print("Extraemos el primer elemento de la pila String: ");
        System.out.println(ultimoString);
```

```
        Pila<Integer> pila4=new Pila<Integer>();
        pila4.insertar(1);
        pila4.insertar(2);
        pila4.insertar(3);
        pila4.insertar(4);
        Integer UltimoInteger=pila4.extraer();
        System.out.print("Extraemos el primer elemento de la pila Integer: ");
        System.out.println(UltimoInteger);
    }
}
```

Si ejecutamos este será el resultado:

```
Extraemos el prime relemento de la pila Persona: Nombe: Ana, Edad: 45
Extraemos el primer elemento de la pila Carta: Num. carta: 4, Palo: Rombos
Extraemos el primer elemento de la pila String: Albaricoque
Extraemos el primer elemento de la pila Integer: 4
```

Capítulo 203.- Colecciones: Java API

Hemos visto en conceptos anteriores como administrar distintas estructuras de datos estáticos (vectores, matrices) y dinámicas (listas y árboles).

Aprendimos a crear clases en Java para administrar listas de tipo pila, cola y genéricas. Desarrollamos todos los algoritmos internos para su administración utilizando punteros.

Veremos ahora que el API Java nos provee un conjunto de clases e interfaces que nos facilitan la creación de pilas, colas, listas genéricas etc.

En muchas situaciones el empleo de esta librería de clases e interfaces nos reducen el tiempo de desarrollo de un programa.

Para trabajar con estas clases e interfaces debemos importarlas del paquete 'java.util' donde se encuentran las mismas.

Todas estas clases e interfaces están implementadas con el concepto de genéricos para poder almacenar cualquier tipo de dato.

Las colecciones fundamentales que podemos hacer uso en nuestros proyectos son:

- Stack: Implementar el concepto de una pila (LIFO – Last In First Out – Último de entrar primero en salir).
- Queue: Implementa el concepto de cola (FIFO – First In First Out – Primero de entrar primero en salir).
- PriorityQueue: Implementa el concepto de una cola por prioridad (por ejemplo si son números los organiza en la cola de menor a mayor).
- ArrayList: Implementa el concepto de un arreglo dinámico que puede crecer o decrecer.
- LinkedList: Implementa el concepto de una lista genérica.
- HashSet, TreeSet y LinkedHashSet: Implementa el concepto de lista sin valores repetidos.

Capítulo 204.- Colecciones: Stack – 1

Veremos con un ejemplo los métodos disponibles en la clase Stack.

También se llama lista LIFO (Last In First Out – último en entrar primero en salir)

Problema

Definir un objeto de la clase Stack y llamar a los métodos principales.

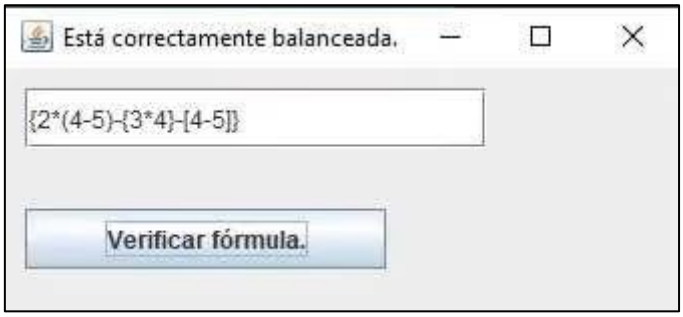
```
import java.util.Stack;
public class PruebaStack {

    public static void main(String[] args) {
        Stack<String> pila1=new Stack<String>();
        System.out.println("Insertamos 3 elementos en la pila");
        pila1.push("Juan");
        pila1.push("Ana");
        pila1.push("Luis");
        System.out.println("Cantidad de elementos de la pila: "+pila1.size());
        System.out.println("Extraer unelemento de la pila:"+pila1.pop());
        System.out.println("Cantidad de elementos de la pila: "+pila1.size());
        System.out.println("Consultar el primer elemento sin extraerlo: "+pila1.peek());
        System.out.println("Cantidad de elementos de la pila: "+pila1.size());
        System.out.println("Extraemos uno a uno los elementos de la pila");
        while (!pila1.isEmpty()) {
            System.out.println(pila1.pop());
        }

        Stack<Integer> pila2=new Stack<Integer>();
        pila2.push(20);
        pila2.push(60);
        pila2.push(120);
        System.out.println("Cantidad de elementos de la pila: "+pila2.size());
        pila2.clear(); // BElimina todos los elementos
        System.out.println("Cantidad de elementos de la pila: "+pila2.size());
    }
}
```

Si ejecutamos este será el resultado:

```
Insertamos 3 elementos en la pila
Cantidad de elementos de la pila: 3
Extraer unelemento de la pila:Luis
Cantidad de elementos de la pila: 2
Consultar el primer elemento sin extraerlo: Ana
Cantidad de elementos de la pila: 2
Extraemos uno a uno los elementos de la pila
Ana
Juan
Cantidad de elementos de la pila: 3
Cantidad de elementos de la pila: 0
```

Capítulo 205.- Colecciones: Stack – 2

Problema



Se debe desarrollar una clase que tenga las siguientes responsabilidades (clase Formula):

- Ingresar una fórmula que contenga paréntesis, corchetes y llaves.
- Validad que los ( ) [ ] y { } estén correctamente balanceados.

Para la solución de este problema la clase Formula empleará la clase Staack.

Primero cargamos la fórmula en un JTextField.

Ejemplo de fórmula: (2+[3-2]*{8/3})

El algoritmo de validación es el siguiente:

Analizamos carácter a carácter la presenciad de paréntesis, corchetes y llaves.

Si vienen símbolos de apertura los almacenamos en la pila.

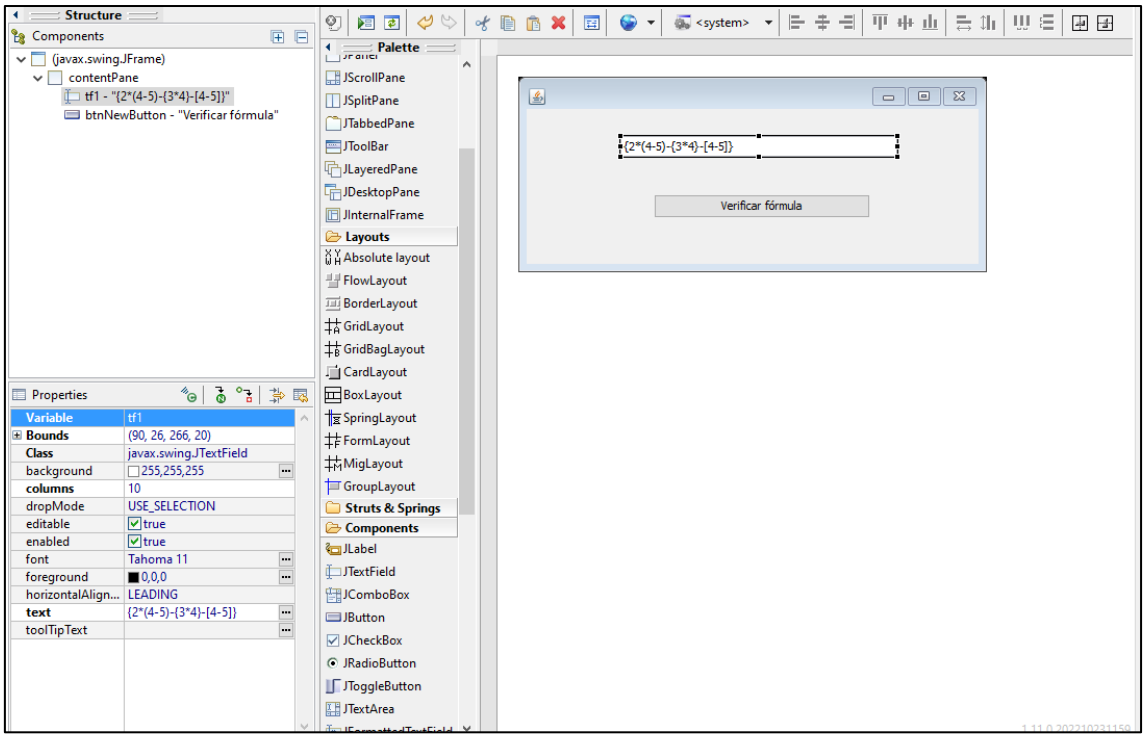
Si vienen símbolos de cerrados extraemos de la pila y verificamos si está el mismo símbolo pero de apertura: en caso negativo podemos inferir que la fórmula no está correctamente balanceada.

Si al finalizar el análisis del último carácter de la fórmula la pila está vacía podemos concluir que está correctamente balanceada.

Ejemplo de fórmulas no balanceadas:

```
}(2+[3-12]*{8/3})
Incorrecta: llega una } de cerrado y la pila está vacía.
{[2+4}]
Incorrecta: llega una llave } y en el tope de la pila hay un corchete [.
{[2+4]
Incorrecta: al finalizar el análisis del último caracter en la pila queda
pendiente una llave {.
```

Utilizaremos a WindowBuilder para crear la interfaz, le llamaremos Formula.



Vamos al actionPerformed del botón.

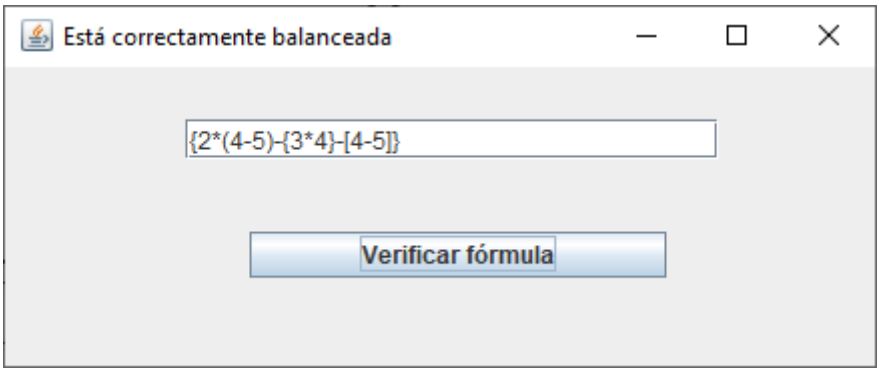
Este será el código:

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

import java.util.Stack;

public class Formula extends JFrame {

    private JPanel contentPane;
    private JTextField tf1;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Formula frame = new Formula();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
```

```
32          });
33      }
34
35      /**
36       * Create the frame.
37       */
38      public Formula() {
39          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
40          setBounds(100, 100, 450, 188);
41          contentPane = new JPanel();
42          contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
43
44          setContentPane(contentPane);
45          contentPane.setLayout(null);
46
47          tf1 = new JTextField();
48          tf1.setText("{2*(4-5)-{3*4}-[4-5]}");
49          tf1.setBounds(90, 26, 266, 20);
50          contentPane.add(tf1);
51          tf1.setColumns(10);
52
53          JButton btnNewButton = new JButton("Verificar f\u00F3rmula");
54          btnNewButton.addActionListener(new ActionListener() {
55              public void actionPerformed(ActionEvent e) {
56                  if (balanceada()) {
57                      setTitle("Está correctamente balanceada");
58                  }else {
59                      setTitle("No está correctamente balanceada");
60                  }
61              }
62          });
63          btnNewButton.setBounds(122, 82, 208, 23);
64          contentPane.add(btnNewButton);
65      }
66      public boolean balanceada() {
67          Stack<Character> pila1=new Stack<Character>();
68          String cadena=tf1.getText();
69          for(int f=0; f<cadena.length(); f++) {
70              if(cadena.charAt(f)=='(' || cadena.charAt(f)=='[' || cadena.charAt(f)=='{') {
71                  pila1.push(cadena.charAt(f));
72              }else {
73                  if(cadena.charAt(f)==')' || cadena.charAt(f)==']' || cadena.charAt(f)=='}') {
74                      if(pila1.isEmpty()) {
75                          return false;
76                      }else {
77                          if(cadena.charAt(f)==')' && pila1.pop()!='(') {
78                              return false;
79                          }else {
80                              if(cadena.charAt(f)==']' && pila1.pop()!='[') {
81                                  return false;
82                              }else {
83                                  if(cadena.charAt(f)=='}' && pila1.pop()!='{') {
84                                      return false;
85                                      }
86
87                                  }
88
89                          }
90                      }
91                  }
92              }
93          }
```

```
94          if (pila1.isEmpty()) {
95              return true;
96          }else {
97              return false;
98          }
99      }
100 }
```

Si ejecutamos este será el resultado:



Capítulo 206.- Colecciones: Queue y PriorityQueue – 1

Una lista se comporta como una cola si las inserciones las hacemos al final y las extracciones las hacemos por el frente de la lista. También se las llama listas FIFO (First In First Out – primero en entrar primero en salir).

Confeccionaremos un programa que permita utilizar la interfaz Queue y mediante la clase LinkedList administre la lista tipo cola.

```
import java.util.Queue;
import java.util.LinkedList;
public class PruebaQueue {

    public static void main(String[] args) {
        Queue<String> cola1=new LinkedList<String>();
        System.out.println("Insertamos 3 elementos en la cola");
        cola1.add("Juan");
        cola1.add("Ana");
        cola1.add("Luis");
        System.out.println("Cantidad de elementos de la cola: "+cola1.size());
        System.out.println("Extraemos un elemento de la cola: "+cola1.poll());
        System.out.println("Cantidad de elementos de la cola: "+cola1.size());
        System.out.println("Consultar el primer elemento de la cola sin extraerlo: "+cola1.peek());
        System.out.println("Extraer todos los elementos de la cola ");
        while(!cola1.isEmpty()) {
            System.out.println("Eliminado el elemento: "+cola1.poll());
        }
        Queue<Integer> cola2=new LinkedList<Integer>();
        cola2.add(10);
        cola2.add(5);
        cola2.add(120);
        System.out.println("Recorrer toda la cola");
        for(Integer elemento: cola2) {
            System.out.print(elemento+"-");
        }
        System.out.println();
        System.out.println("Eliminar todos los elementos de la cola");
        cola2.clear();
        System.out.println("Cantidad de elementos de la cola: "+cola2.size());
    }
}
```

Si ejecutamos este será el resultado:

```
Insertamos 3 elementos en la cola
Cantidad de elementos de la cola: 3
Extraemos un elemento de la cola: Juan
Cantidad de elementos de la cola: 2
Consultar el primer elemento de la cola sin extraerlo: Ana
Extraer todos los elementos de la cola
Eliminado el elemento: Ana
Eliminado el elemento: Luis
Recorrer toda la cola
10-5-120-
Eliminar todos los elementos de la cola
Cantidad de elementos de la cola: 0
```

Capítulo 207.- Colecciones: Queue y PriorityQueue – 2

Una variante de una cola clásica la implementa la clase PriorityQueue. Cuando se agregan elementos a la cola se organiza según su valor, por ejemplo si es un número se ingresan de menor a mayor.

```
import java.util.PriorityQueue;
public class PruebaPriorityQueue {

    public static void main(String[] args) {
        PriorityQueue<Integer> cola1=new PriorityQueue<Integer>();
        cola1.add(70);
        cola1.add(120);
        cola1.add(6);
        System.out.println("Primero en la cola: "+cola1.peek());
        System.out.println("Recuperar todos los elementos de la cola");
        while(!cola1.isEmpty()) {
            System.out.print(cola1.poll()+"-");
        }
        System.out.println();

        PriorityQueue<String> cola2=new PriorityQueue<String>();
        cola2.add("Juan");
        cola2.add("Ana");
        cola2.add("Luis");
        cola2.add("Carlos");
        System.out.println("Recuperar todos los elementos de la cola");
        while(!cola2.isEmpty()) {
            System.out.print(cola2.poll()+"-");
        }
        System.out.println();
    }
}
```

Si ejecutamos este será el resultado:

```
Primero en la cola: 6
Recuperar todos los elementos de la cola
6-70-120-
Recuperar todos los elementos de la cola
Ana-Carlos-Juan-Luis-
```

Capítulo 208.- Colecciones: Queue y PriorityQueue – 3

Problema de aplicación de una cola

Cuando implementamos desde cero el concepto de una cola con el lenguaje Java desarrollamos una serie de ejercicios de aplicación. Los volvemos a codificar pero ahora utilizando la interfaz Queue y la clase LinkedList.

Planteo del problema:

Este práctico tiene por objetivo mostrar la importancia de las colas en las Ciencias de la computación y más precisamente en las simulaciones.

Las simulaciones permiten analizar situaciones de la realidad sin la necesidad de ejecutarlas realmente. Tiene el beneficio que su coste es muy inferior a hacer pruebas en realidad.

Desarrollar un programa para la simulación de un cajero automático.

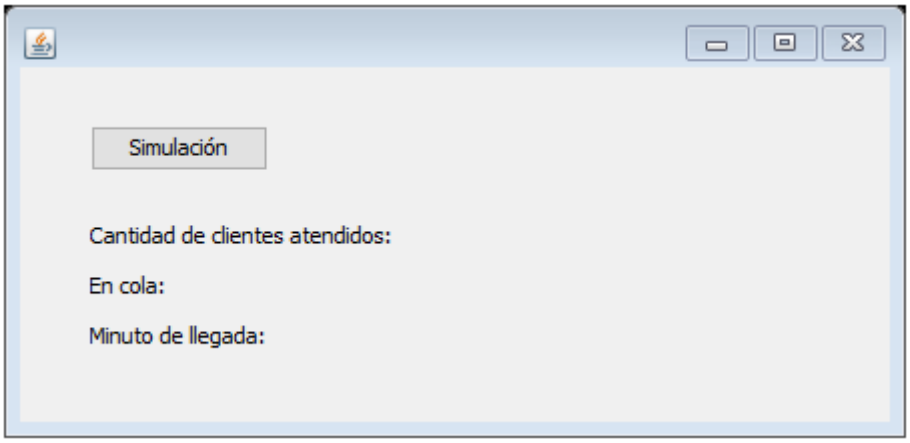
Se cuenta con la siguiente información:

- Llegan clientes que se atienden en 10 horas.
- Cada cliente tarda entre 2 y 4 minutos en ser atendido.

Obtener la siguiente información:

1- Cantidad de clientes que se atienden en 10 horas.
2- Cantidad de clientes que hay en la cola después de 10 horas.
3- Hora de llegada del primer cliente que no es atendido liego de 10 horas (es decir la persona que está primera en la cola cuando se cumplen 10 horas).

Vamos a hacer uso del WindowBuilder:



Las etiquetas las hemos renombrado l1, l2 y l3.

Las hemos pasado de local a global.

Importamos los siguientes métodos:

```
import java.util.Queue;
import java.util.LinkedList;
```

código del botón:

```
JButton btnNewButton = new JButton("Simulaci\u00F3n");
btnNewButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        simular();
    }

    private void simular() {
        int estado=0;
        int llegada=2+(int)(Math.random()*2);
        int salida=-1; // No ha llegado
        int cantAtendidos=0;
        Queue<Integer> cola=new LinkedList<Integer>();
        // Simulamos las 10 horas.
        for(int minuto=0; minuto<600; minuto++) {
            if(llegada==minuto) {
                if(estado==0) {
                    estado=1;
                    salida=minuto+2+(int)(Math.random()*3);
                }else {
                    cola.add(minuto);
                }
                llegada=minuto+2+(int)(Math.random()*2);
            }

            if(salida==minuto) {
                estado=0;
                cantAtendidos++;
                if(!cola.isEmpty()) {
                    cola.poll();
                    estado=1;
                    salida=minuto+2+(int)(Math.random()*3);
                }
            }
        }
        l1.setText("Cantidad de clientes atendidos: "+cantAtendidos);
        l2.setText("En cola: "+cola.size());
        l3.setText("Minuto de llegada: "+cola.peek());
    }
});
btnNewButton.setBounds(35, 29, 138, 23);
contentPane.add(btnNewButton);
```

Ejecutamos este será el resultado:

Capítulo 209.- Colecciones: Queue y PriorityQueue – 4

Problema de aplicación de una cola

Un supermercado tiene tres cajas para la atención de los clientes.

Las cajeras tardan entre 7 y 11 minutos para la atención de cada cliente.

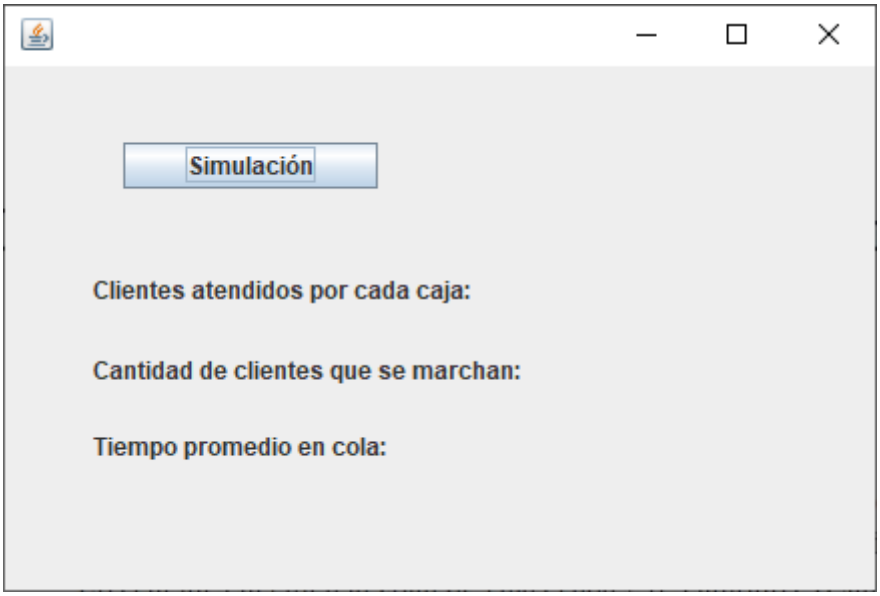
Los clientes llegan a la zona de cajas cada 2 ó 3 minutos. (Cuando el cliente llega, si todas las cajas tienen s6 personas, el cliente se marcha del supermercado).

Cuando el cliente llega a la zona de cajas elige la caja con una cola menor.

Realiza la simulación durante 10 horas y obtener la siguiente información:
a – Cantidad de clientes atendidos por cada caja.
b – Cantidad de clientes que se marcharon sin hacer la compra.
c – Tiempo promedio en cola.

Crearemos la interfaz con WindowBuilder.

Crearemos un botón y tres etiquetas que pasaremos de local a global.



Vamos a programar el botón 'Simulación'.

Este será el código completo:

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JButton;
import javax.swing.JLabel;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

import java.util.Queue;
import java.util.LinkedList;

public class Supermercado extends JFrame {
```

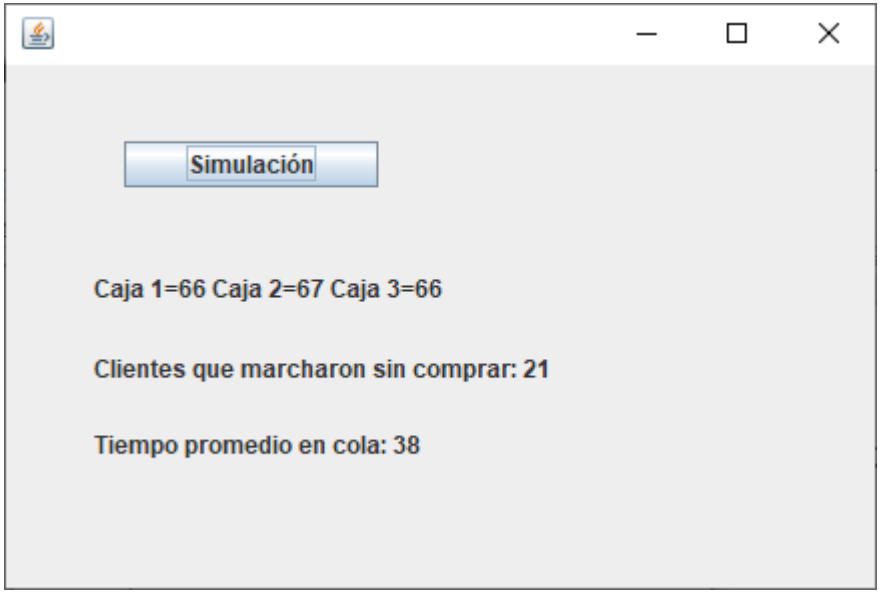
```
private JPanel contentPane;
private JLabel l1;
private JLabel l2;
private JLabel l3;

/**
 * Launch the application.
 */
public static void main(String[] args) {
	EventQueue.invokeLater(new Runnable() {
		public void run() {
			try {
				Supermercado frame = new Supermercado();
				frame.setVisible(true);
			} catch (Exception e) {
				e.printStackTrace();
			}
		}
	});
}

/**
 * Create the frame.
 */
public Supermercado() {
	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	setBounds(100, 100, 450, 300);
	contentPane = new JPanel();
	contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

	setContentPane(contentPane);
	contentPane.setLayout(null);

	JButton btnNewButton = new JButton("Simulaci\u00F3n");
	btnNewButton.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e) {
			simualacion();
		}
	});
	btnNewButton.setBounds(59, 38, 127, 23);
	contentPane.add(btnNewButton);

	l1 = new JLabel("Clientes atendidos por cada caja:");
	l1.setBounds(44, 104, 296, 14);
	contentPane.add(l1);

	l2 = new JLabel("Cantidad de clientes que se marchan:");
	l2.setBounds(44, 144, 320, 14);
	contentPane.add(l2);

	l3 = new JLabel("Tiempo promedio en cola:");
	l3.setBounds(44, 182, 320, 14);
	contentPane.add(l3);
}
```

```
protected void simualacion() {
	int estado1=0, estado2=0, estado3=0;
	int marchar=0;
	int llegada=2+(int)(Math.random()*2);
	int salida1=-1, salida2=-1, salida3=-1;
	int cantAte1=0, cantAte2=0, cantAte3=0;
	int tiempoEnCola=0;
	int cantidadEnCola=0;
	Queue<Integer> cola1=new LinkedList<Integer>();
	Queue<Integer> cola2=new LinkedList<Integer>();
	Queue<Integer> cola3=new LinkedList<Integer>();
	for(int minuto=0; minuto<600; minuto++) {
		if(llegada==minuto) {
			if(estado1==0) {
				estado1=1;
				salida1=minuto+7+(int)(Math.random()*5);
			}else {
				if(estado2==0) {
					estado2=1;
					salida2=minuto+7+(int)(Math.random()*5);
				}else {
					if(estado3==0) {
						estado3=1;
						salida3=minuto+7+(int)(Math.random()*5);
					}else {
						if(cola1.size()==6 && cola2.size()==6 && cola3.size()==6) {
							marchar++;
						}else {
							if(cola1.size()<cola2.size() && cola1.size()<cola3.size()) {
								cola1.add(minuto);
							}else {
								if(cola2.size()<cola3.size()) {
									cola2.add(minuto);
								}else {
									cola3.add(minuto);
								}
```

```
                                }
                            }
                        }
                    }
                }
                llegada=minuto+2+(int)(Math.random()*2);
            }
            if (salida1==minuto) {
                cantAte1++;
                estado1=0;
                if(!cola1.isEmpty()) {
                    estado1=1;
                    int m=cola1.poll();
                    salida1=minuto+7+(int)(Math.random()*5);
                    tiempoEnCola=tiempoEnCola+(minuto-m);
                    cantidadEnCola++;
                }
            }
            if (salida2==minuto) {
                cantAte2++;
                estado2=0;
                if(!cola2.isEmpty()) {
                    estado2=1;
                    int m=cola2.poll();
                    salida2=minuto+7+(int)(Math.random()*5);
                    tiempoEnCola=tiempoEnCola+(minuto-m);
                    cantidadEnCola++;
                }

            }
            if (salida3==minuto) {
                cantAte3++;
                estado3=0;
                if(!cola3.isEmpty()) {
                    estado3=1;
                    int m=cola3.poll();
```

```
                        salida3=minuto+7+(int)(Math.random()*5);
                        tiempoEnCola=tiempoEnCola+(minuto-m);
                        cantidadEnCola++;
                    }

                }
            }
            l1.setText("Caja 1="+cantAte1+" Caja 2="+cantAte2+" Caja 3="+cantAte3);
            l2.setText("Clientes que marcharon sin comprar: "+marchar);
            int tiempoPromedio=tiempoEnCola/cantidadEnCola;
            l3.setText("Tiempo promedio en cola: "+tiempoPromedio);
            }
    }
```

Este será el resultado:



Capítulo 210.- Colecciones: LinkedList – 1

La clase LinkedList implementa la lógica para trabajar con listas genéricas, es decir podemos insertar y extraer elementos de cualquier parte de la lista.

Confeccionaremos un programa para llamar los principales métodos de esta clase.

```
import java.util.LinkedList;
public class PruebaLinkedListt {

    // Recorrer la lista.
    public static void imprimir(LinkedList<String> lista) {
        for(String elemento: lista) {
            System.out.print(elemento+" ");
        }
        System.out.println();
    }

    public static void main(String[] args) {
        LinkedList<String> lista1=new LinkedList<String>();
        // Añadimos 3 elementos.
        lista1.add("Juan");
        lista1.add("Luis");
        lista1.add("Carlos");
        imprimir(lista1);
        lista1.add(1, "Ana");
        imprimir(lista1);
        lista1.remove(0);
        imprimir(lista1);
        // Elimina la primera coincidencia.
        lista1.remove("Carlos");
        imprimir(lista1);
        System.out.println("Cantidad de elmentos: "+lista1.size());
        // Para ver una lista si contiene cierto valor
        if(lista1.contains("Ana")) {
            System.out.println("La lista tiene al cadena 'Ana'");
        }else {
            System.out.println("La lista no tiene al cadena 'Ana'");
        }
        System.out.println(lista1.get(0));
        System.out.println("Cantidad de elmentos: "+lista1.size());
        // Eliminar toda la lista
        lista1.clear();
        if(lista1.isEmpty()) {
            System.out.println("La lista esta vacía");
        }else {
            System.out.println("La lista no está vacía");
        }

    }
}
```

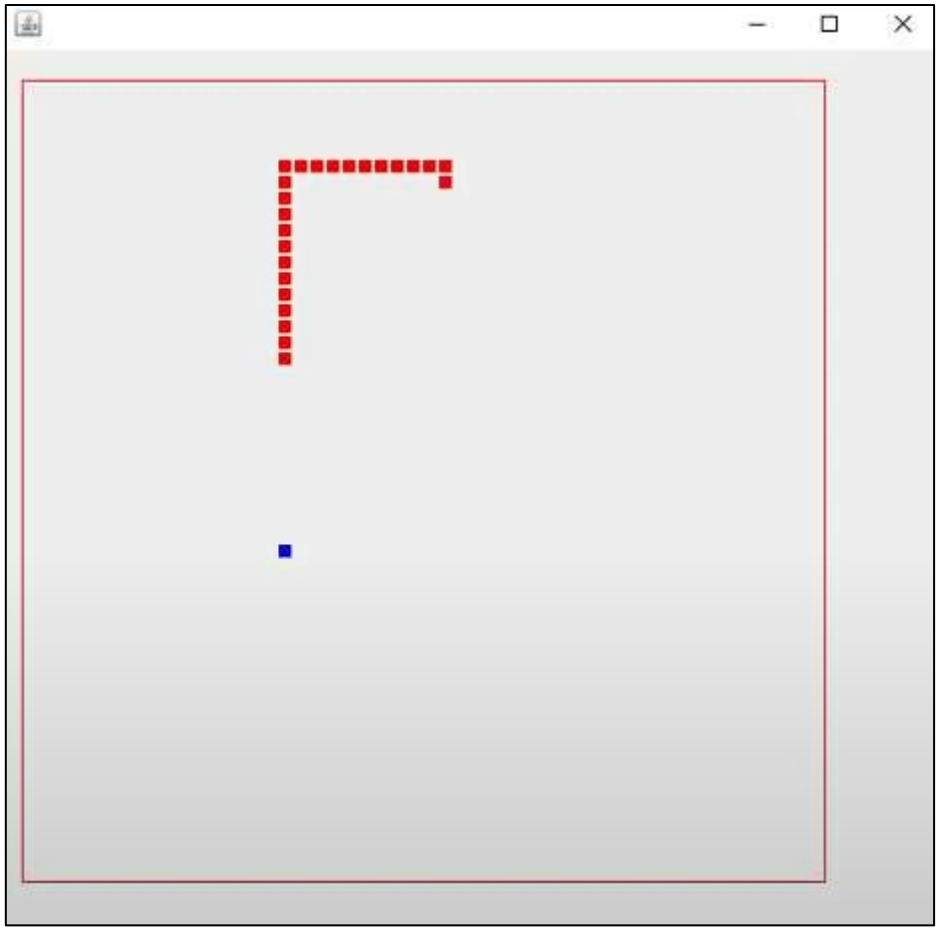
Si ejecutamos este será el resultado:

```
Juan Luis Carlos
Juan Ana Luis Carlos
Ana Luis Carlos
Ana Luis
Cantidad de elmentos: 2
La lista tiene al cadena 'Ana'
Ana
Cantidad de elmentos: 2
La lista esta vacía
```

Capítulo 211.- Colecciones: LinkedList – 2

Problema de aplicación

Confeccionar el juego de la serpiente (snake) utilizando un objeto de la clase LinkedList para representar cada trozo de la misma.



Este es el código:

```
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JFrame;

import java.util.LinkedList;

public class Vibora extends JFrame implements KeyListener, Runnable{
	private enum Direccion{
		IZQUIERDA, DERECHA, SUBE, BAJA
	}

	class Punto{
		int x, y;
		public Punto(int x, int y) {
			this.x=x;
```

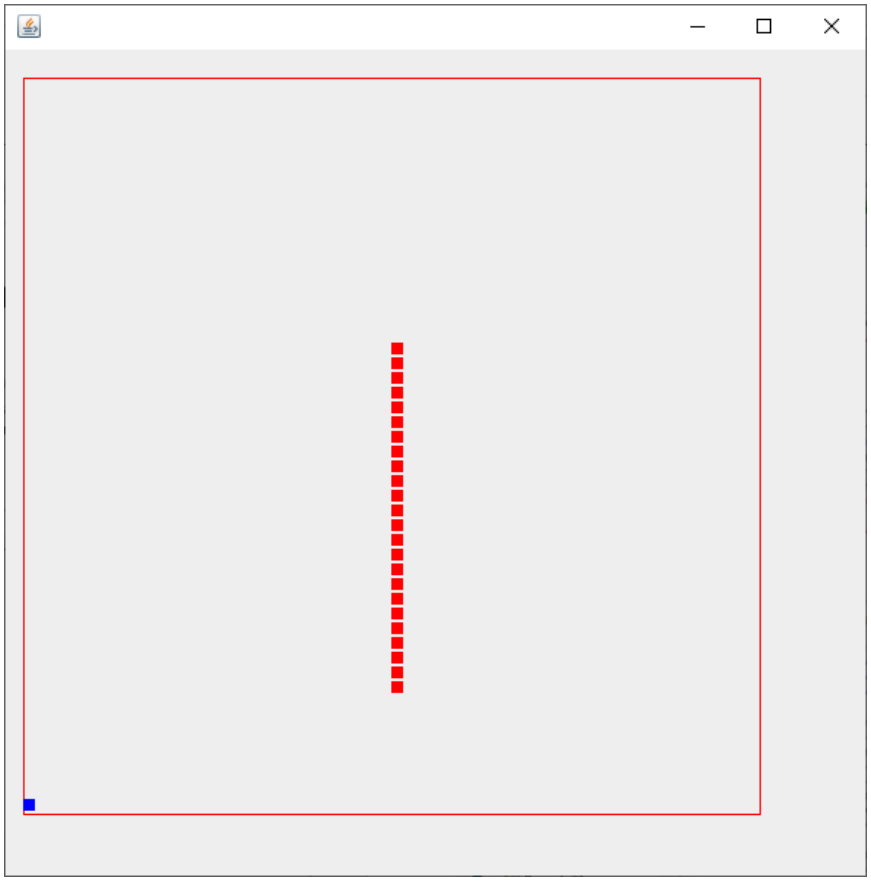
```
		this.y=y;
		}
}

private LinkedList<Punto> lista=new LinkedList<Punto>();
// Columna y fila donde se encuentra la cabeza de la vibora
private int columna, fila;
private int colfruta, filfruta; // Donde está la fruta
private boolean activo=true; // Lo pondremos en falso cuando finalice el juego
private Direccion direccion=Direccion.DERECHA;
private Thread hilo; // Hilo para el programa.
private int crecimiento=0; // Indica la cantidad de cuadradiros que debe crecer
private Image imagen; // Para evitar el parpadeo en el paint.
private Graphics bufferGraphics; // dibujar en memoria

public Vibora() {
		addKeyListener(this); // Escuchar los eventos del teclado
		lista.add(new Punto(4,25));
		lista.add(new Punto(3,25));
		lista.add(new Punto(2,25));
		lista.add(new Punto(1,25));
		// La columna y fila de la vibora
		columna=4;
		fila=25;
		// Coordenada de la fruta
		colfruta=(int)(Math.random()*50);
		filfruta=(int)(Math.random()*50);
		// Creamos el hilo
		hilo=new Thread(this);
		hilo.start();
}

@Override
public void run() {
		while (activo){
			try {
```

```
			Thread.sleep(100);
			switch(direccion) {
			case DERECHA:
				columna++;
				break;
			case IZQUIERDA:
				columna--;
				break;
			case SUBE:
				fila--;
				break;
			case BAJA:
				fila++;
				break;
			}

			repaint();
			sePisa();
			// Insertamos la coordenada de la cabeza en la lista.
			lista.addFirst(new Punto(columna, fila));
			if(this.verificarComeFruta()==false && crecimiento==0) {
				// Si no estamos en la coordenada de la fruta y debe crecer la vibora
				// borramos ultimo nodo de la lista para que siga teniendo la misma
				// cantidad de nodos.
				lista.remove(lista.size()-1);
			} else {
				// Si crecimiento es mayor a 0 es que debemos hacer crecer
				// la vibora
				if(crecimiento>0) {
					crecimiento--;
				}
			}
			verificarFin();

		} catch (InterruptedException e) {
			e.printStackTrace();
```

```
			}
		}

	}

	private void sePisa() {
		for(Punto p: lista) {
			if(p.x==columna && p.y==fila) {
				activo=false;
				setTitle("Perdiste");
			}
		}
	}

	private void verificarFin() {
		if (columna<0 || columna>=50 || fila<0 || fila>=50) {
			activo=false;
			setTitle("Perdiste");
		}
	}

	private boolean verificarComeFruta() {
		if (columna==colfruta && fila==filfruta) {
			crecimiento=10;
			// Coordenada de la fruta
			colfruta=(int)(Math.random()*50);
			filfruta=(int)(Math.random()*50);
			return true;
		}else {
			return false;
		}
	}

	public void paint(Graphics g) {
		super.paint(g);
		if (!lista.isEmpty()) {
```

```
            if (imagen==null) {
                  imagen=createImage(this.getSize().width, this.getSize().height);
                  bufferGraphics=imagen.getGraphics();
            }
            // Borramos la imagen de memoria
            bufferGraphics.clearRect(0, 0, getSize().width, getSize().height);
            // Dibujamos el cuadrado
            bufferGraphics.setColor(Color.red);
            bufferGraphics.drawRect(20, 50, 500, 500);
            // Dibujamos la vibora
            for(Punto punto: lista) {
                  bufferGraphics.fillRect(punto.x*10+20, 50+punto.y*10, 8,8);
            }
            // Dibujamos la fruta
            bufferGraphics.setColor(Color.blue);
            bufferGraphics.fillRect(colfruta*10+20, filfruta*10+50, 8, 8);
            g.drawImage(imagen, 0,0, this);
      }
}



public static void main(String[] args) {
      Vibora vibora1=new Vibora();
      vibora1.setBounds(0,0,600,600);
      vibora1.setVisible(true);
      vibora1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

@Override
public void keyTyped(KeyEvent e) {
      // TODO Auto-generated method stub

}

@Override
```

```
	public void keyPressed(KeyEvent e) {
		// TODO Auto-generated method stub
		if(e.getKeyCode()==KeyEvent.VK_RIGHT) {
			direccion=Direccion.DERECHA;
		}
		if(e.getKeyCode()==KeyEvent.VK_LEFT) {
			direccion=Direccion.IZQUIERDA;
		}
		if(e.getKeyCode()==KeyEvent.VK_UP) {
			direccion=Direccion.SUBE;
		}
		if(e.getKeyCode()==KeyEvent.VK_DOWN) {
			direccion=Direccion.BAJA;
		}
	}

	@Override
	public void keyReleased(KeyEvent e) {
		// TODO Auto-generated method stub

	}
}
```



Capítulo 212.- Colecciones: ArrayList – 1

Problema de aplicación

La clase ArrayList implementa la lógica para trabajar con listas genéricas, es decir podemos insertar y extraer elementos de cualquier parte de la lista.

La diferencia del ArrayList con la clase LinkedList es la implementación interna de los algoritmos.

La clase LinkedList emplea una lista doblemente encadenda y la clase ArrayList utiliza un arreglo que se redimensiona en forma automática según se efectúa inserciones y extracciones de datos.

La principal ventaja de emplear la clase ArrayList es que el acceso a un elemento de la lista es inmediato mediante el método 'get', en cambio la implementación del método 'get' es la clase LinkedList requiere recorrer en forma secuencial la lista hasta llegar a la posición a buscar.

Si la lista no va ha tener grandes cambios es inserciones y extracciones durante la ejecución del programa es más común utilizar la clase ArrayList en lugar de LinkedList.

Confeccionaremos el mismo programa que hicimos en la clase LinkeList pero empleando ahora la clase ArrayList.

```
import java.util.ArrayList;
public class PruebaArraList {

    public static void imprimir(ArrayList<String> lista) {
        for (String elemento: lista) {
            System.out.print(elemento+" ");
        }
        System.out.println();
    }

    public static void main(String[] args) {
        ArrayList<String> lista1=new ArrayList<String>();
        lista1.add("Juan");
        lista1.add("Luis");
        lista1.add("Carlos");
        imprimir(lista1);
        lista1.add(1, "Ana");
        imprimir(lista1);
        lista1.remove(0);
        imprimir(lista1);
        lista1.remove("Carlos");
        imprimir(lista1);
        System.out.println("La cantidad de elementos son "+lista1.size());
        if(lista1.contains("Ana")) {
            System.out.println("Contiene el elemento 'Ana'");
        }else {
            System.out.println("No contiene el elemento 'Ana'");
        }
        System.out.println("El segundo elemento del ArrayList es '"+lista1.get(1)+"'");
        System.out.println("La cantidad de elementos son "+lista1.size());
        lista1.clear();
        if(lista1.isEmpty()) {
            System.out.println("La lista está vacía");
        }else {
            System.out.println("La lista no está vacía");
        }
    }
}
```

Si ejecutamos este será el resultado:

```
Juan Luis Carlos
Juan Ana Luis Carlos
Ana Luis Carlos
Ana Luis
La cantidad de elementos son 2
Contiene el elemento 'Ana'
El segundo elemento del ArrayList es 'Luis'
La cantidad de elementos son 2
La lista está vacía
```

Capítulo 213.- Colecciones: ArrayList – 2

Problema

Crear un proyecto y dentro del mismo crear dos clases. La primera clase se debe llamar 'Carta', con dos atributos el palo y el número de carta. Por otro lado declarar una clase llamada 'Mazo' que contenga un ArrayList de tipo 'Carta'. Imprimir las cartas de forma ordenada según como se insertaron y luego mezclar y volver a imprimir.

Creamos la clase Carta:

```
public class Carta {
    public enum Palo{
        TREBOL, DIAMENTE, CORAZON, PICA
    }

    private Palo palo;
    private int numero;

    public Carta(int numero, Palo palo) {
        this.numero=numero;
        this.palo=palo;
    }

    public void imprimir() {
        System.out.println(numero+ " de "+palo.toString().toLowerCase());
    }
}
```

Creamos la clase Mazo con el método main:

```
import java.util.ArrayList;
import java.util.Collections;

public class Mazo {
    private ArrayList<Carta> cartas;

    public Mazo() {
        cartas=new ArrayList<Carta>(8);
        cartas.add(new Carta(1, Carta.Palo.TREBOL));
        cartas.add(new Carta(2, Carta.Palo.TREBOL));
        cartas.add(new Carta(1, Carta.Palo.DIAMENTE));
        cartas.add(new Carta(2, Carta.Palo.DIAMENTE));
        cartas.add(new Carta(1, Carta.Palo.PICA));
        cartas.add(new Carta(2, Carta.Palo.PICA));
        cartas.add(new Carta(1, Carta.Palo.CORAZON));
        cartas.add(new Carta(2, Carta.Palo.CORAZON));
    }

    public void imprimir() {
        for(Carta carta: cartas) {
            carta.imprimir();
        }
    }

    public void barajar() {
        Collections.shuffle(cartas);
    }
```

```
    public static void main(String[] args) {
        Mazo mazo1=new Mazo();
        mazo1.imprimir();
        System.out.println("Barajamos las cartas");
        mazo1.barajar();
        System.out.println("Mostrar de nuevos las cartas");
        mazo1.imprimir();

    }
}
```

Una vez ejecutado este será el resultado:

```
1 de trebol
2 de trebol
1 de diamente
2 de diamente
1 de pica
2 de pica
1 de corazon
2 de corazon
Barajamos las cartas
Mostrar de nuevos las cartas
2 de diamente
1 de diamente
1 de corazon
2 de corazon
2 de pica
2 de trebol
1 de pica
1 de trebol
```

Capítulo 214.- Colecciones: HashSet, TreeSet y LinkedHashSet – 1

La diferencia fundamental entre las clases HashSet, TreeSet, LinkedHashSet con respecto a las listas ArrayList y LinkedList es que no pueden haber elementos repetidos en las colecciones que implementa la interfaz Set.

A su vez se han creado estas tres clases que tienen pequeñas diferencias entre una y otras.

- HashSet: El conjunto de datos no se almacena en un orden espedificado, si bien se garantiza que no hay duplicados.
- TreeSet: Los elementos del conjunto se almacenan de menor a mayor.
- LinkedHashSet: Los elementos del conjunto se encuentran en el orden que se insertan, similar a una lista pero sin dejar de ingresar valores repetidos.

Métodos más comunes

Los métodos más comunes que tiene esta clase son:

- size: Retorna la cantidad de elementos del conjunto.
- clear: Elimina todos los elementos.
- remove: Elimina el elemento si existe en el conjunto.

  lista1.remove(20);

- isEmpty: Nos informa si la lista está vacía.
- contains: Le pasamos como parámetro el dato a buscar en el conjutno:

  if (conjunto1.contains(20))
  …

```
import java.util.Set;
import java.util.HashSet;
import java.util.TreeSet;
import java.util.LinkedHashSet;

public class PruebaSet {

    public static void main(String[] args) {
        Set<Integer> conjunto1=new HashSet<Integer>();
        conjunto1.add(20);
        conjunto1.add(10);
        conjunto1.add(1);
        conjunto1.add(5);
        // Intentamos añadir de nuevo el valor 20 no se añadirá
        conjunto1.add(20);
        for(Integer elemento: conjunto1) {
            System.out.print(elemento+"-");
        }
        System.out.println();

        Set<Integer> conjunto2=new TreeSet<Integer>();
        conjunto2.add(20);
        conjunto2.add(10);
        conjunto2.add(1);
        conjunto2.add(5);
        // Intentamos añadir de nuevo el valor 20 no se añadirá
        conjunto2.add(20);
```

```
        for(Integer elemento: conjunto2) {
            System.out.print(elemento+"-");
        }
        System.out.println();

        Set<Integer> conjunto3=new LinkedHashSet<Integer>();
        conjunto3.add(20);
        conjunto3.add(10);
        conjunto3.add(1);
        conjunto3.add(5);
        // Intentamos añadir de nuevo el valor 20 no se añadirá
        conjunto3.add(20);
        for(Integer elemento: conjunto3) {
            System.out.print(elemento+"-");
        }
        System.out.println();
    }
}
```

Si ejecutamos este será el resultado:

```
1-20-5-10-
1-5-10-20-
20-10-1-5-
```

Capítulo 215.- Colecciones: HastSet, TreeSet y LinkedHashSet – 2

Problema

Generar una lista de 10 valores enteros comprendidos entre 1 y 100. Validar que no se repitan, para esto utilizar la ayuda de una de las colecciones de conjuntos vistos en este concepto.

```
import java.util.Set;
import java.util.TreeSet;

public class Colecciones {

    public static void main(String[] args) {
        Set<Integer> conjunto1=new TreeSet<Integer>();
        while (conjunto1.size()<10) {
            int valor=1+(int)(Math.random()*100);
            conjunto1.add(valor);
        }
        for(Integer valor: conjunto1) {
            System.out.print(valor+" - ");
        }
        System.out.println();
    }
}
```

Este será el resultado:

```
4 - 7 - 13 - 15 - 28 - 31 - 53 - 64 - 69 - 83 -
```

Si utilizamos LinkedHashSet:

```
import java.util.Set;
import java.util.LinkedHashSet;

public class Colecciones {

    public static void main(String[] args) {
        Set<Integer> conjunto1=new LinkedHashSet<Integer>();
        while (conjunto1.size()<10) {
            int valor=1+(int)(Math.random()*100);
            conjunto1.add(valor);
        }
        for(Integer valor: conjunto1) {
            System.out.print(valor+" - ");
        }
        System.out.println();
    }
}
```

Este será el resultado:

```
75 - 38 - 61 - 79 - 85 - 56 - 73 - 58 - 3 - 59 -
```

Capítulo 216.- Colecciones: HashMap, TreeMap y LinkedHashMap

Estas clases: HashMap, TreeMap y LinkedHashMap nos permiten almacenar elementos asociados a cada clave un valor.

Para cada clave tenemos un valor asociado. Podemos después buscar fácilmente un valor para una determinada clave. Las claves en el diccionario no pueden repetirse.

Algunos ejemplos donde podríamos usar un Mapa:

- Guardar en la clave las extensiones de los archivos y en el valor los nombres de archivos que lo pueden abrir.
- En una agenda podemos guardar como 'clave' la fecha y hora y las actividades en el 'valor'.
- Un diccionario de sinónimos, en la clave almacenar una palabra y en el 'valor' sus sinónimos.

Problema

Almacenar un diccionario las palabras en castellano como 'clave' y las traducciones de las mismas en el 'valor'. Probar métodos más significativos de la clase HashMap.

```
import java.util.Map;
import java.util.HashMap;
public class PruebaHashMap {

    public static void main(String[] args) {
        Map<String, String> mapa1=new HashMap<String, String>();
        mapa1.put("rojo", "red");
        mapa1.put("verde", "green");
        mapa1.put("azul", "blue");
        mapa1.put("blanco", "white");
        System.out.println("Listado completo de valroes");
        for(String valor:mapa1.values()) {
            System.out.print(valor+ " ");
        }
        System.out.println();
        System.out.println("Listado completo de las claves");
        for(String clave:mapa1.keySet()) {
            System.out.print(clave+ " ");
        }
        System.out.println();
        // Si no existe retorna null
        System.out.println("La traduccion de 'rojo' es "+mapa1.get("rojo"));

        if(mapa1.containsKey("negro")) {
            System.out.println("Tiene la palabra clave 'negro' ");
        }else {
            System.out.println("No tiene la palabra clave 'negro' ");
        }
        System.out.print("La traducción de 'negro' es ");
        System.out.println(mapa1.getOrDefault("negro","No existe la traducción de negro"));

        mapa1.remove("azul");
        System.out.println("Listado completo de las claves");
        for(String clave:mapa1.keySet()) {
            System.out.print(clave+ " ");
        }
        System.out.println();
    }
}
```

Si ejecutamos este será el resultado:

```
Listado completo de valroes
red white green blue
Listado completo de las claves
rojo blanco verde azul
La traduccion de 'rojo' es red
No tiene la palabra clave 'negro'
La traducción de 'negro' es No existe la traducción de negro
Listado completo de las claves
rojo blanco verde
```

Vamos a cambiar MashMap por TreeMap

```
import java.util.Map;
import java.util.TreeMap;
public class PruebaHashMap {

    public static void main(String[] args) {
        Map<String, String> mapa1=new TreeMap<String, String>();
        mapa1.put("rojo", "red");
```

Cuando ejecutemos observaremos que las claves aparecen ordenadas alfabéticamente.

```
blue white red green
Listado completo de las claves
azul blanco rojo verde
La traduccion de 'rojo' es red
No tiene la palabra clave 'negro'
La traducción de 'negro' es No existe la traducción de negro
Listado completo de las claves
blanco rojo verde
```

Ahora la cambiamos por LinkedHashMap.

```
public static void main(String[] args) {
    Map<String, String> mapa1=new LinkedHashMap<String, String>();
    mapa1.put("rojo", "red");
```

Cuando lo ejecutemos las claves se mostrarán en el orden que fueron ingresados.

```
Listado completo de valroes
red green blue white
Listado completo de las claves
rojo verde azul blanco
La traduccion de 'rojo' es red
No tiene la palabra clave 'negro'
La traducción de 'negro' es No existe la traducción de negro
Listado completo de las claves
rojo verde blanco
```

Capítulo 217.- Creación de paquetes (package)

En programas pequeños o cuando estamos aprendiendo a programar en Java es común definir la clase o conjunto de clases del proyecto en el paquete por defecto (defaul package), es así como lo hemos hecho en este curso.

Cuando disponemos de clases en el paquete por defecto no pueden ser reutilizadas en otro proyectos.

Las paquetes son un mecanismo de Java que nos permite agrupar un conjunto de clases relacionadas con un nombre de paquete, luego dicho paquete puede compilarse y ser reutilizado en otros proyectos que desarrollemos nosotros u otras empresas.

Para definir una clase dentro de un determinado paquete debemos agregar en la primera línea de nuestro código fuente la palabra clave 'package' y seguidamente el nombre de paquete.

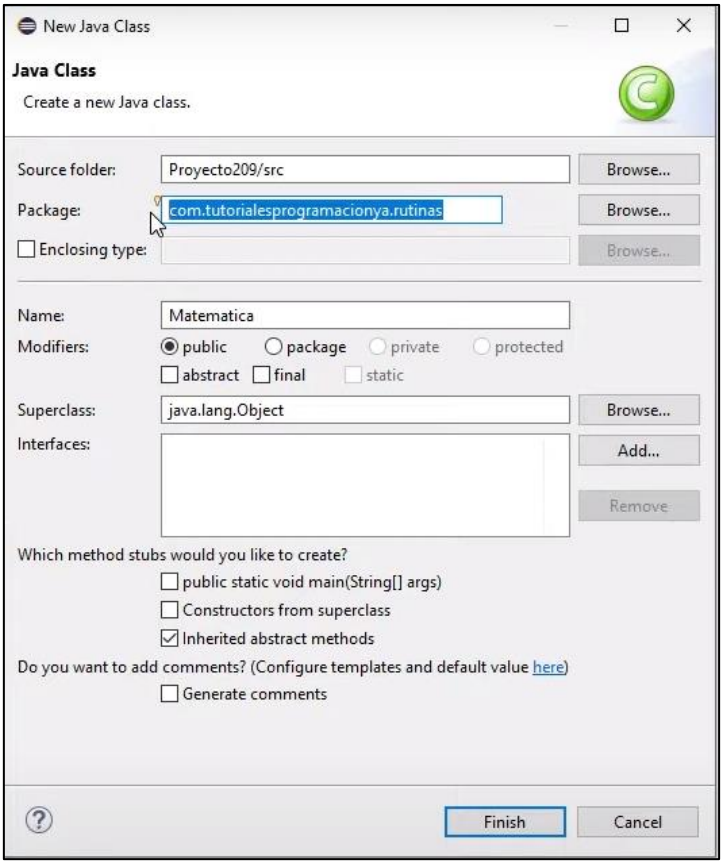
Por convención se propone que todo paquete comience por URL de la empresa (en forma invertida) que lo desarrolla y seguidamente otros nombres que nos den la idea de que clases agrupa dicho paquete.

Problema

Crear dos clases llamadas 'Matematica' y 'Cadena', definir una serie de métodos estáticos en cada una de ellas.

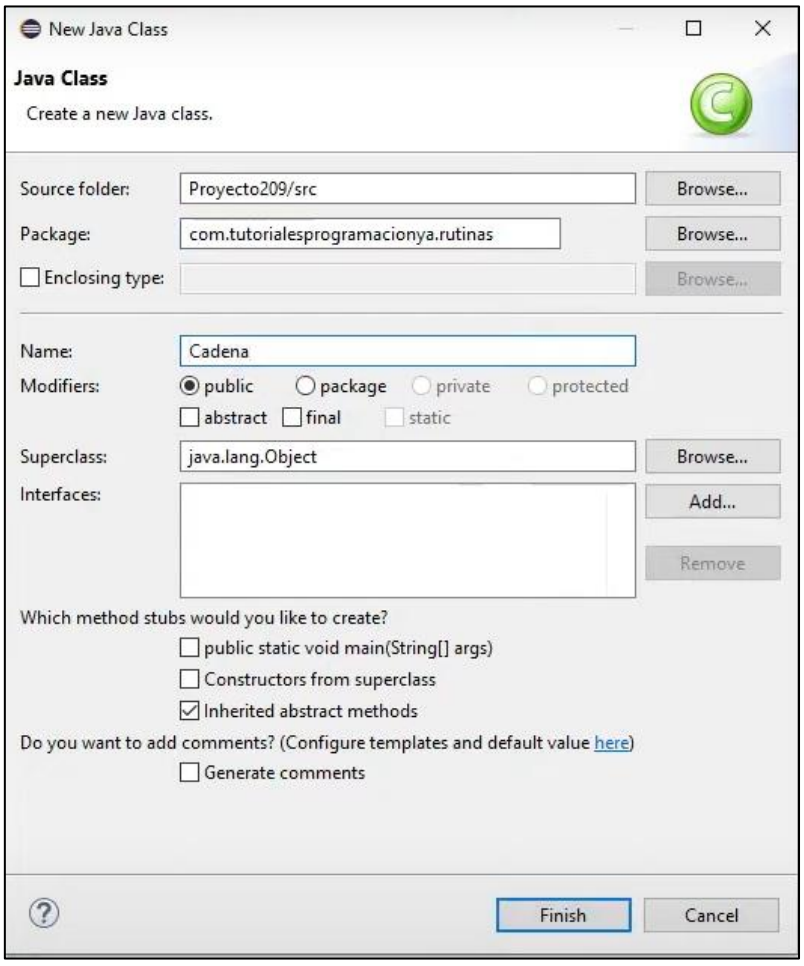
Disponer de dos clases en el paquete: com.tutorialesprogramacionya.rutinas

Crear un segundo paquete llamado 'com.tutorialesprogramacionya.calculadora' en el mismo proyecto y acceder a las clases del primer paquete.



Escribimos el siguiente código:

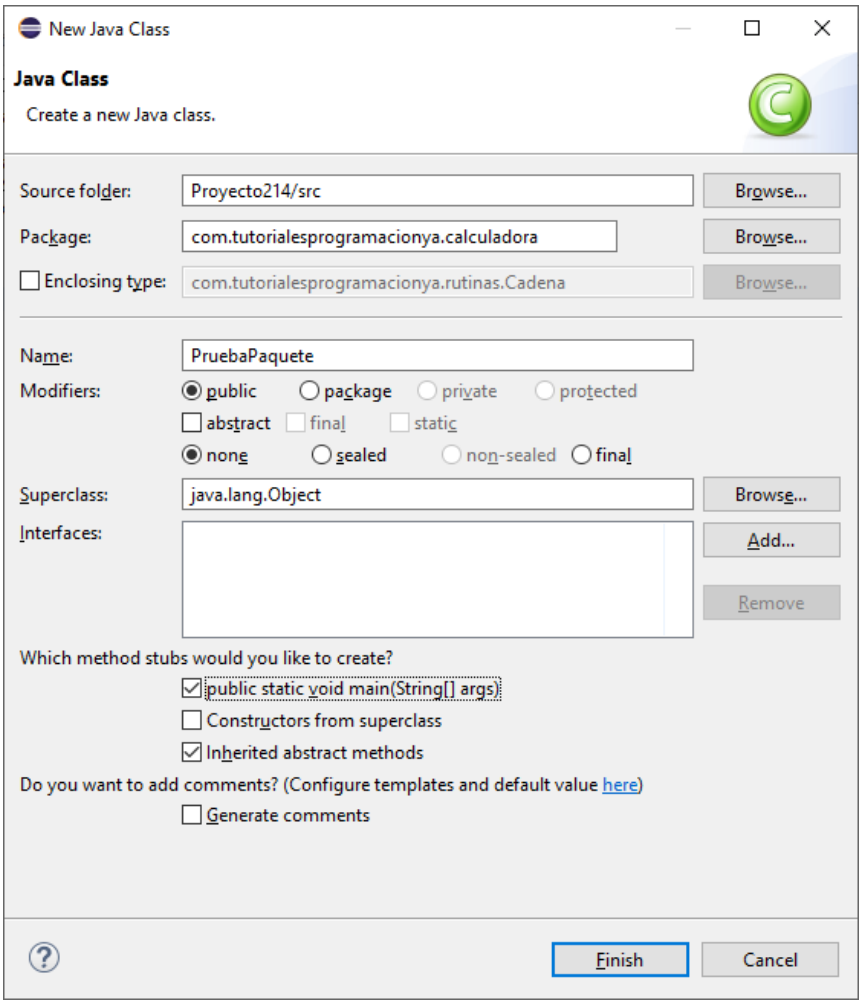
```
package com.tutorialesprogramacionya.rutinas;

public class Matematica {
    public static int sumar(int v1, int v2) {
        return v1+v2;
    }

    public static int restar(int v1, int v2) {
        return v1-v2;
    }
}
```



Este será el código:

```
package com.tutorialesprogramacionya.rutinas;

public class Cadena {
    public static String mayuscula(String s) {
        return s.toUpperCase();
    }
}
```

Vamos a realizar una tercera clase que se guardará en otro paquete.



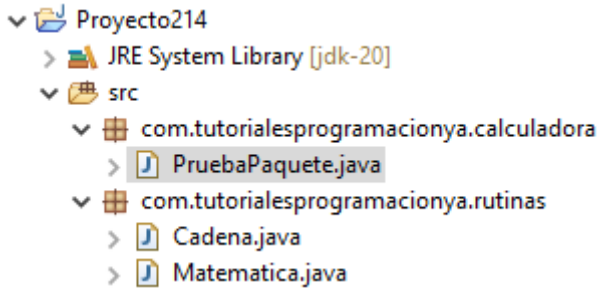
Esta clase tiene el método main.

```
package com.tutorialesprogramacionya.calculadora;

import com.tutorialesprogramacionya.rutinas.Cadena;
import com.tutorialesprogramacionya.rutinas.Matematica;

public class PruebaPaquete {

    public static void main(String[] args) {
        int x1=10;
        int x2=20;
        int suma=Matematica.sumar(x1, x2);
        System.out.println("La suma es "+suma);

        int resta=Matematica.restar(x2, x1);
        System.out.println("La resta es "+resta);

        System.out.println("La palabra 'codorilo' en mayúsculas "+Cadena.mayuscula("cocodrilo"));
    }
}
```

Si ejecutamos este será el resultado:

```
La suma es 30
La resta es 10
La palabra 'codorilo' en mayúsculas COCODRILO
```

Este es la estructura de como se han guardado las clases:



Si queremos importar todas las clases de un paquete:

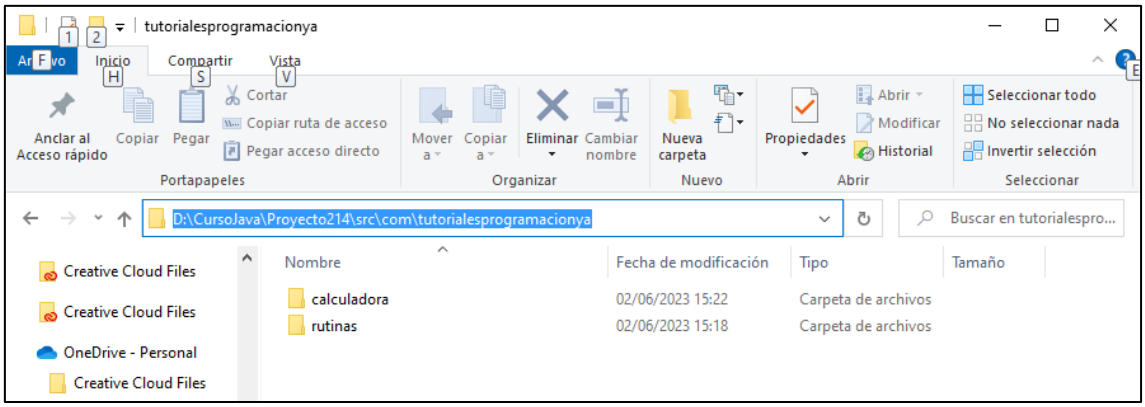
En lugar de escribir:

```
import com.tutorialesprogramacionya.rutinas.Cadena;
import com.tutorialesprogramacionya.rutinas.Matematica;
```

Podemos escribir:

```
import com.tutorialesprogramacionya.rutinas.*;
```

Con el asterisco '*' incluimos todas las clases del paquete.



D:\CursoJava\Proyecto214\src\com\tutorialesprogramacionya

En la carpeta de nuestro proyecto entramos en la carpeta src, en ella encontramos la carpeta com, entramos en ella, encontramos la carpeta tutorialesprogramacionya entramos en ella y podemos observar como vemos las dos carpetas calculadora y rutinas que es donde se encuentran las correspondientes clases.

Recordemos algunos paquetes que hemos utilizado:

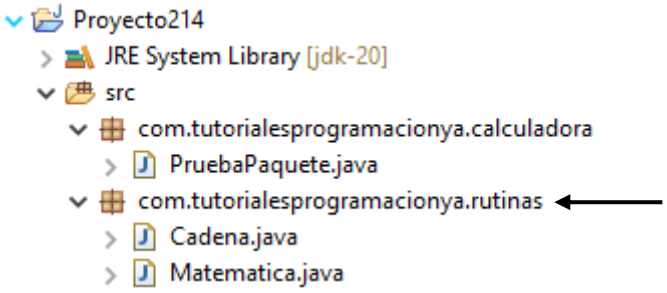
- Paquete java.util donde se implementan las clases, interfaces, enumeraciones, etc. para administrar colecciones vistas en los capítulos anteriores como son las clases Arraylist, LinkedList, Stack, etc.
- Paquete javax.swing donde se declaran entre otras las clases JFrame, JButton, JLabel, etc.
- Paquete java.lang que es el único paquete de Java que no requiere efectuar el import ya que el compilador lo hace en forma automática. Este paquete agrupa las clases de uso más común como pueden ser: Objetc, String, System, Integer, Double, etc.

Capítulo 218.- Generar un archivo Jar de un paquete

Ahora veremos los pasos para generar un archivo Jar para reutilizar un paquete en múltiplos proyectos.

Un archivo Jar (**J**ava **A**rchive) agrupa generalmente un conjunto de archivos .class y otros recursos como texto, imágenes, etc. Para ser reutilizados en otros proyectos.

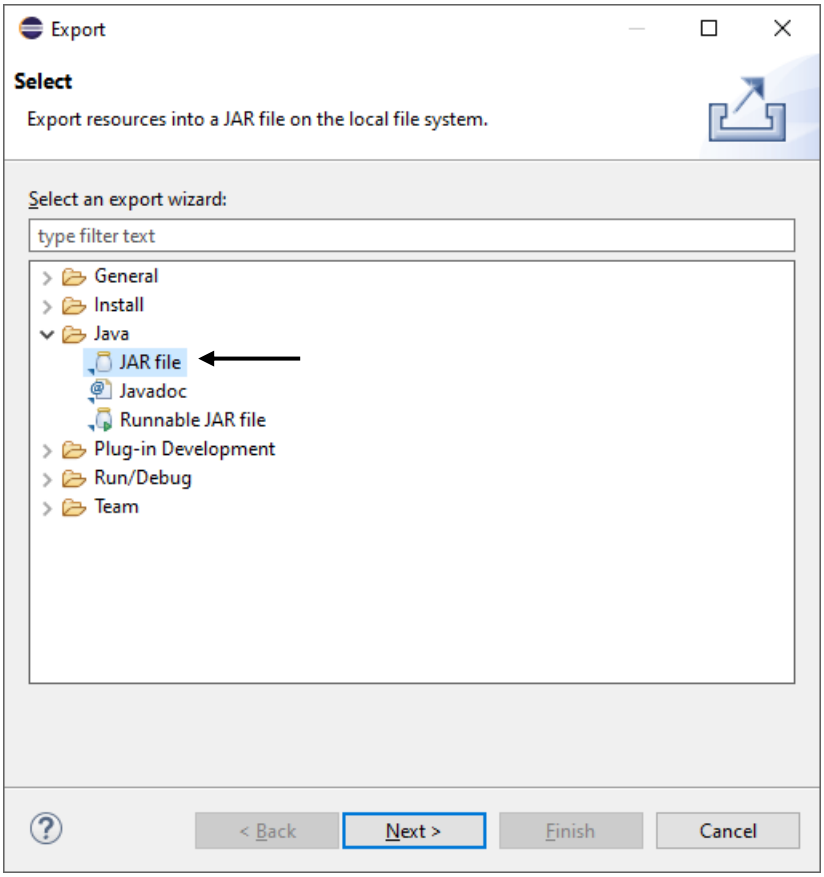
El contenido del archivo Jar se encuentra comprimido en un formato zip.



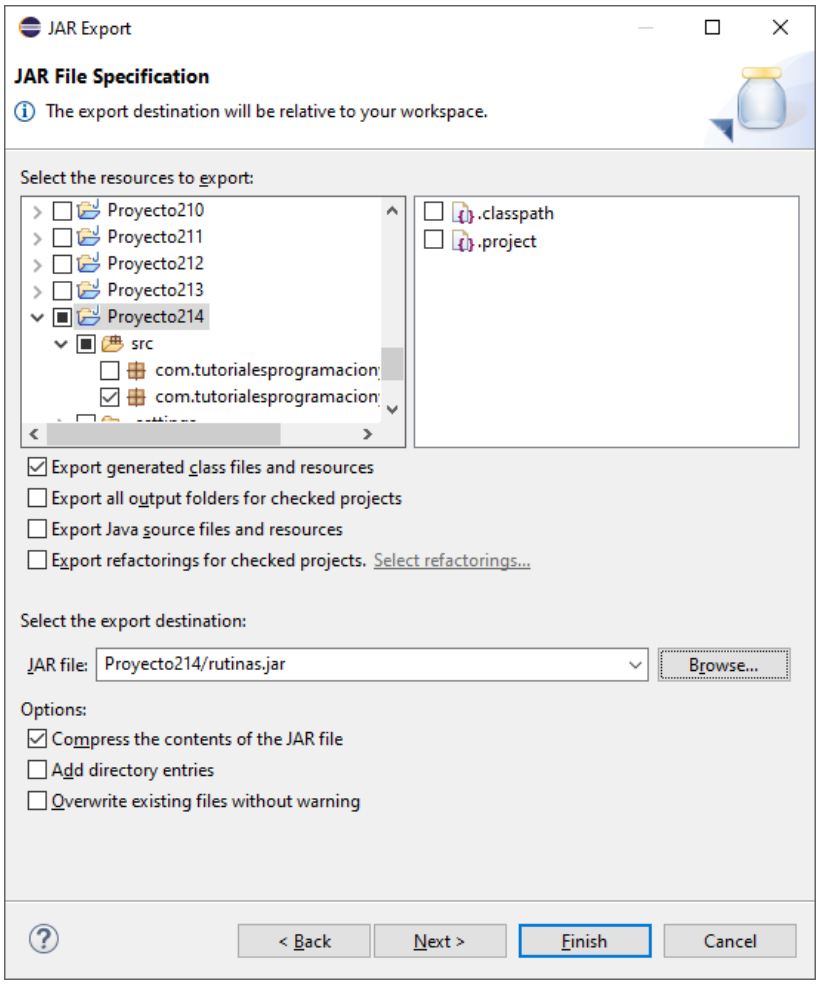
Siguiendo con el proyecto anterior, donde aprendimos a crear paquetes.

Hacemos botón derecho sobre el paquete que queremos exportar.

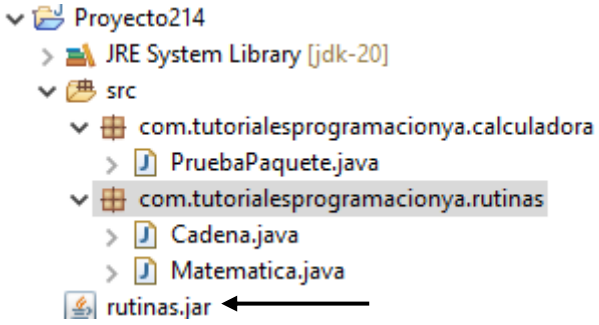
Del menú seleccionaremos Export…



Seleccionaremos JAR file, seguido del botón Next.

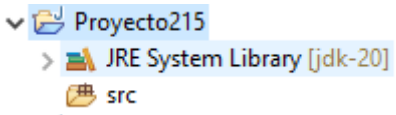


Lo guardaremos en el mismo proyecto con el nombre rutinas.jar, seguido del botón Finish.



Ya tenemos las clases sumar y restar de la clase Matematicas.java. en el archivo rutinas.jar.

Ahora vamos a crear un nuevo proyecto.

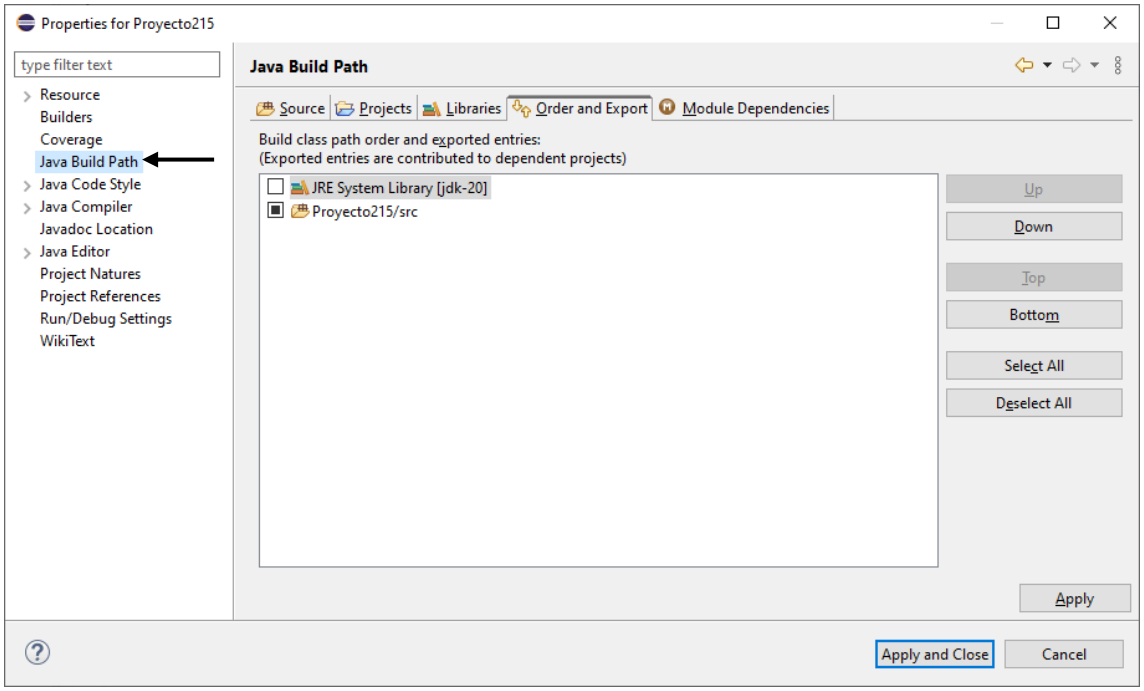


Vamos a crear una clase.

```
1
2  public class PruebaJarExterno {
3
4      public static void main(String[] arg) {
5          System.out.println(Matematicas.)
6      }
7
8  }
```

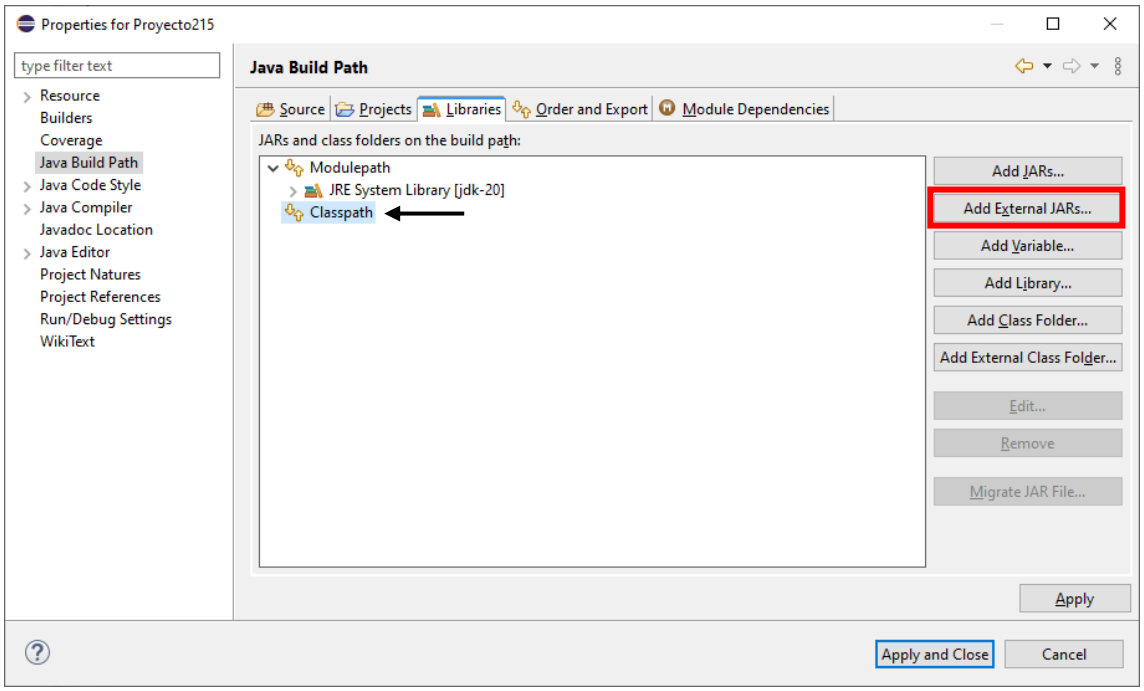
No tenemos acceso a las clase Matematicas.

Realizaremos botón derecho sobre el nombre del proyecto y seleccionaremos Properties,

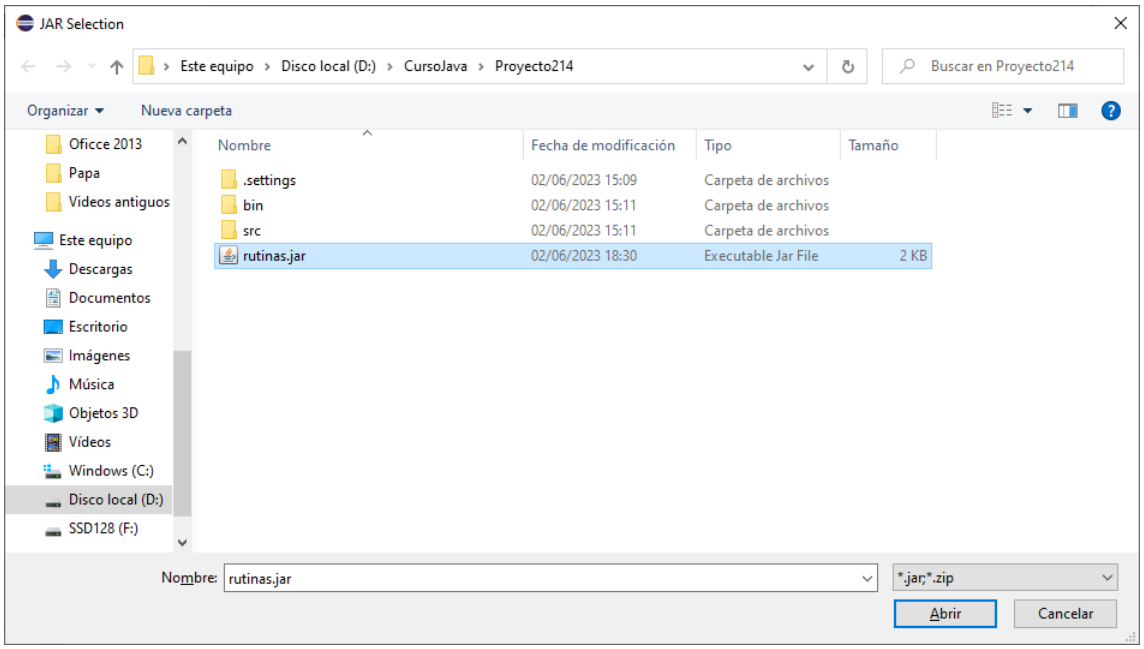


Seleccionaremos Java Build Path.

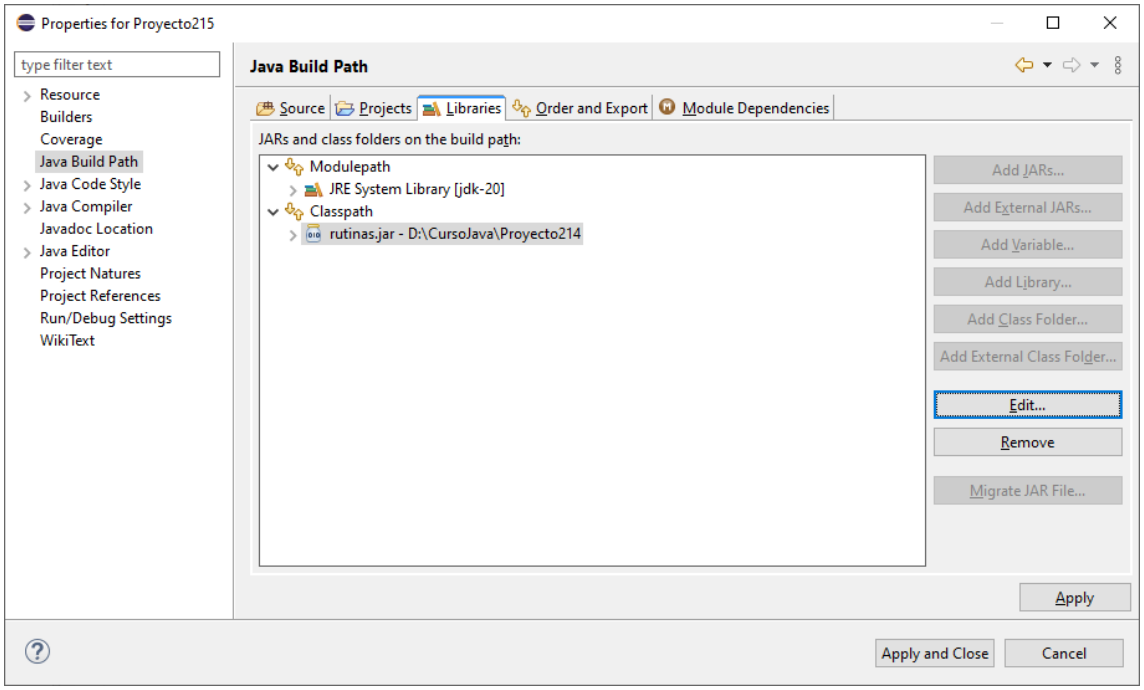
Seleccionamos Libraries.



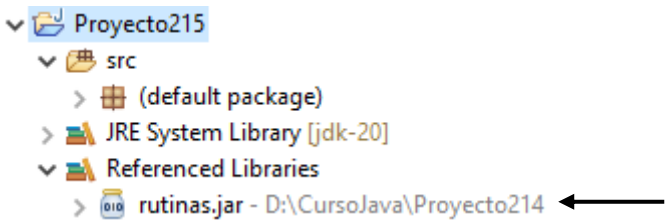
Seleccionaremos Classpath y el botón Add Extgernal JARs…



Seleccionamos el archivo rutinas.jar y le damos al botón Abrir.



Le damos al botón Apply and Close.



Se ha creado un enlace a las librerías rutinas.jar, que se encuentra en el Proyecto214.

Vamos a seguir con la clase que tenemos en el nuevo proyecto.

```
1  import com.tutorialesprogramacionya.rutinas.Matematica;
2  public class PruebaJarExterno {
3
4      public static void main(String[] arg) {
5          System.out.println(Matematica.sumar(7,12));
6      }
7  }
```

Importamos las clase Matematica del paquete com.tutorialesprogramacionya.rutinas.

Si ejecutamos este será el resultado:

```
19
```

También podemos llamar al método restar.

```
import com.tutorialesprogramacionya.rutinas.Matematica;
public class PruebaJarExterno {

    public static void main(String[] arg) {
        System.out.println(Matematica.sumar(7,12));
        System.out.println(Matematica.restar(20, 10));
    }
}
```

Si ejecutamos este será el resultado:

```
19
10
```

De esta forma podemos ejecutar estos métodos en nuevos proyectos.

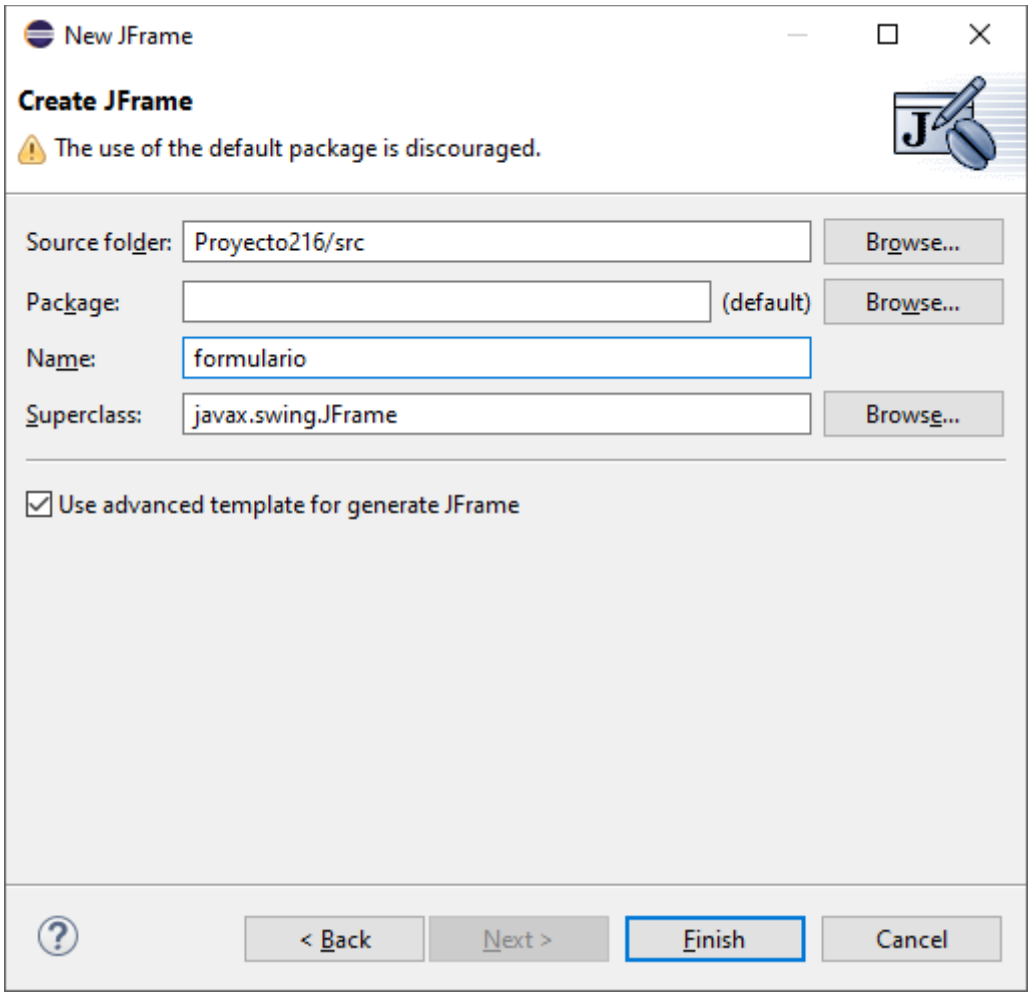
Capítulo 219.- Generar un archivo Jar ejecutable

Cuando tenemos que distribuir la aplicación no se instala Eclipse para ejecutarlo, en su lugar se genera un archivo Jar ejecutable.

Veamos los pasos para generar un archivo ejecutable.

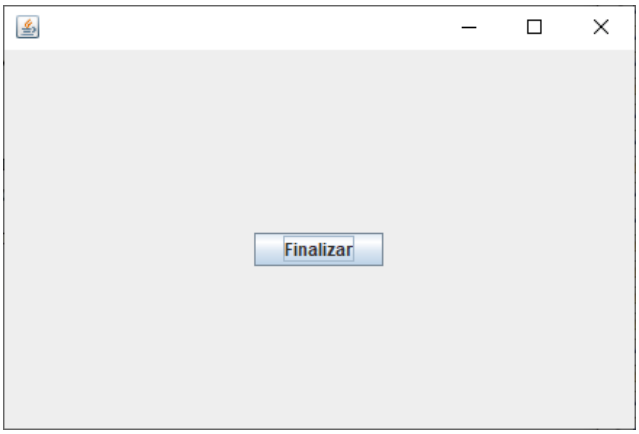
Para que sea ejecutable debe haber una clase en el proyecto que contenga la función 'main'.

Vamos a crear una clase utilizando el WindowBuilder.

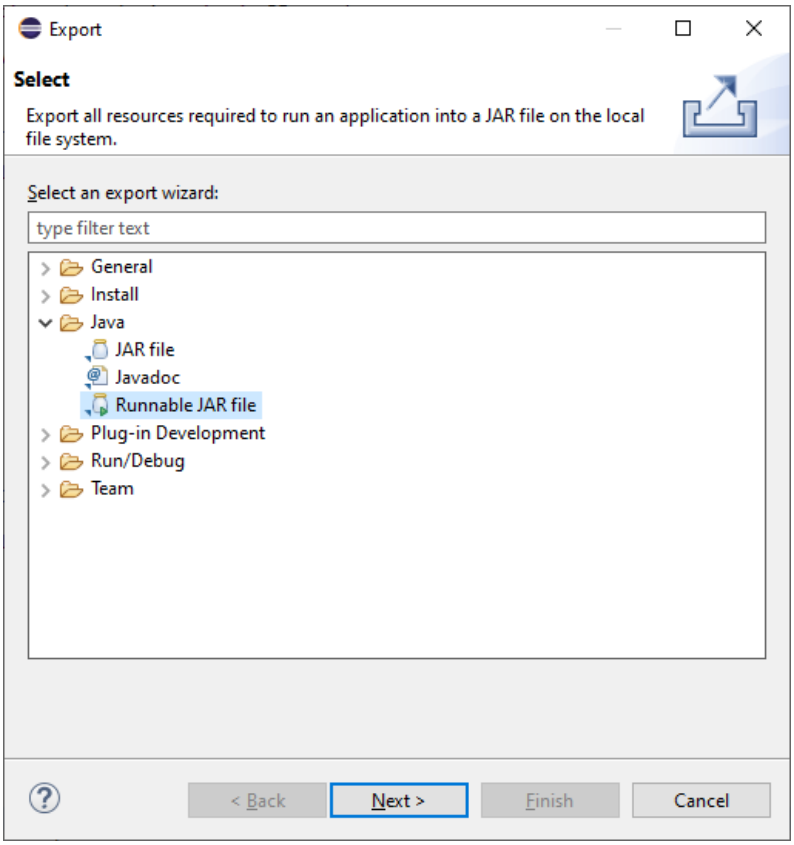



```
public formulario() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

    setContentPane(contentPane);
    contentPane.setLayout(null);

    JButton btnNewButton = new JButton("Finalizar");
    btnNewButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            System.exit(0);
        }
    });
    btnNewButton.setBounds(172, 126, 89, 23);
    contentPane.add(btnNewButton);
}
```

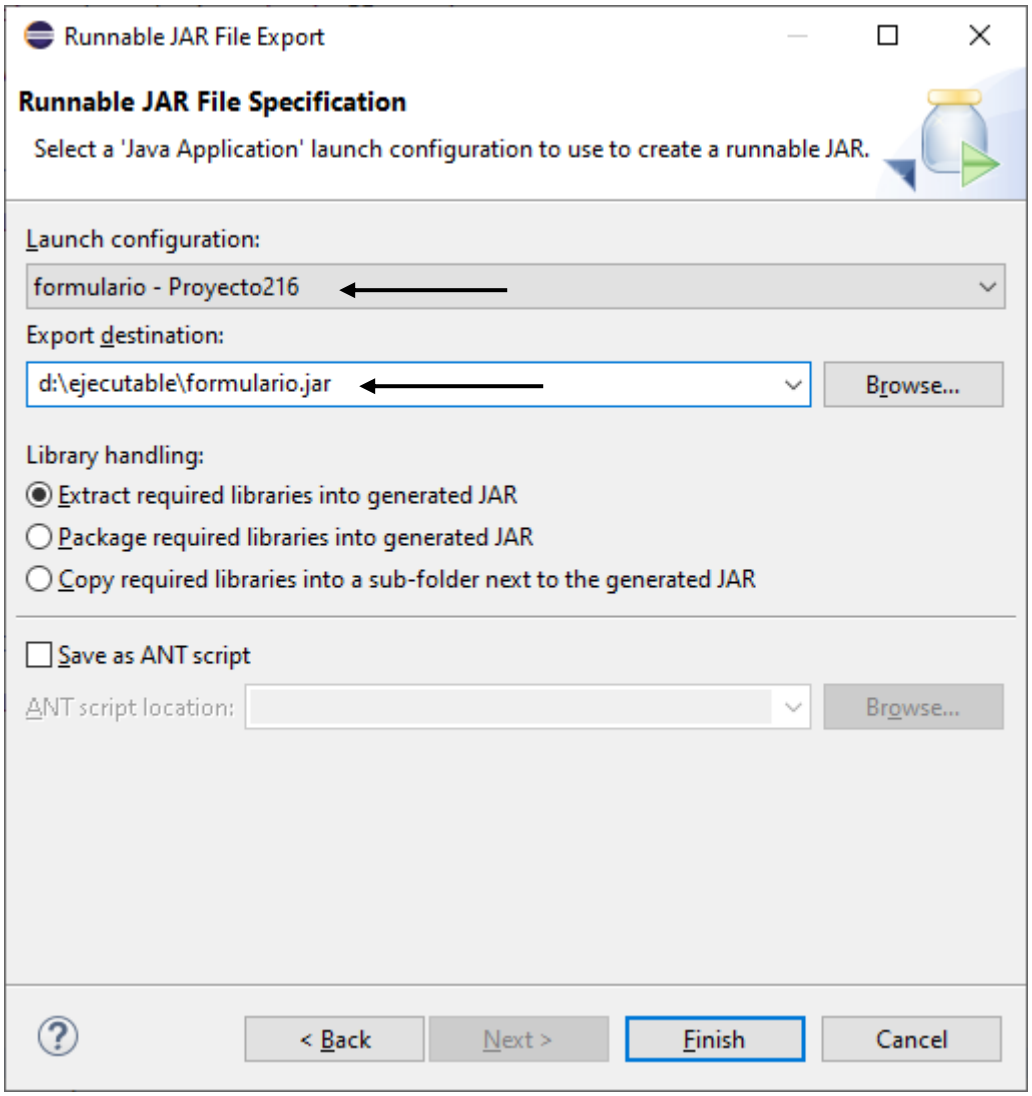
Al presionar el botón queremos que finalice el programa.



Ahora seleccionaremos el nombre del proyecto con el botón derecho del ratón y del menú seleccionaremos Export…

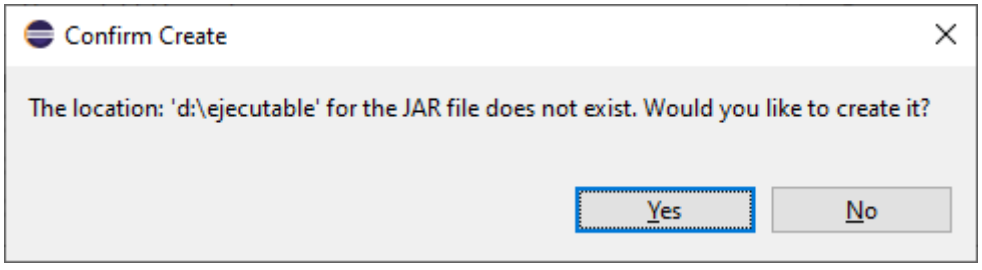


Seleccionaremos Runnable JAR file, seguido del botón Next.

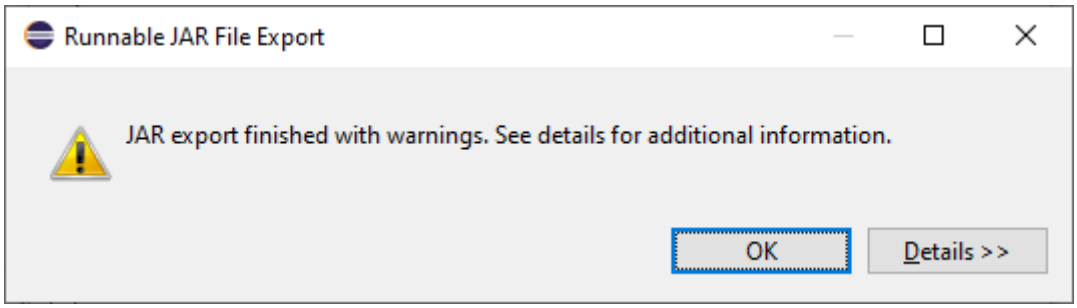


Seleccionamos el proyecto que queremos hacer ejecutable.

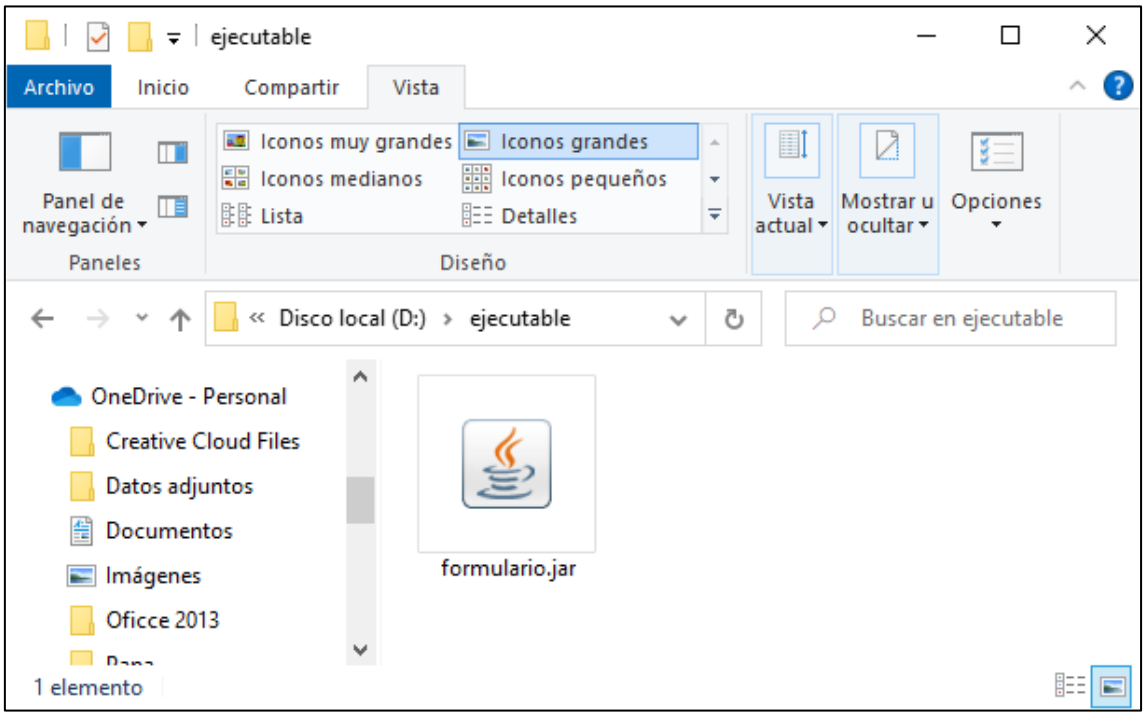
Le decimos en que carpeta lo queremos exportar, seguido del botón Finish.



Omitimos este pequeño error dándole al botón Yes.

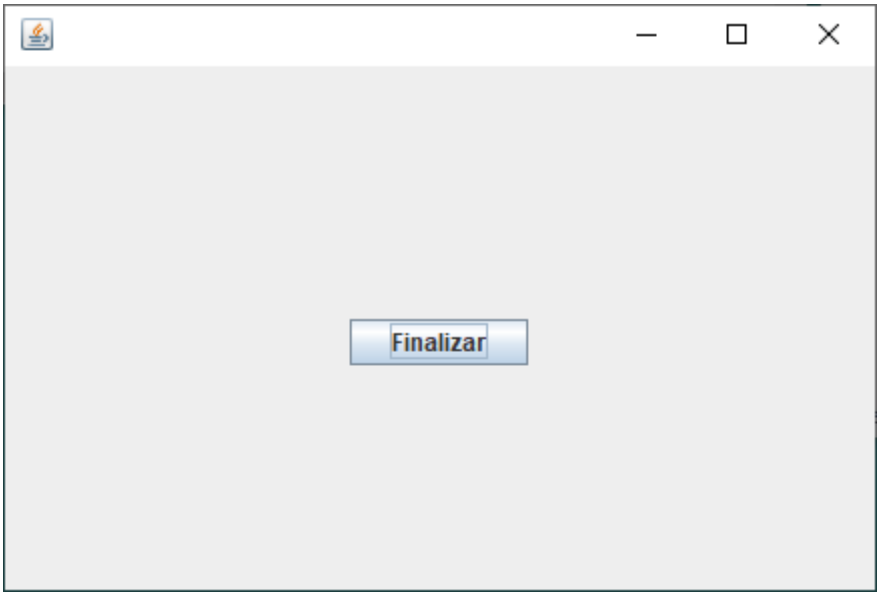


Le damos a OK.



Salimos de Eclipse y vamos a la carpeta donde tenemos el ejecutable.

Hacemos doble clic.



Capítulo 220.- Operador condicional: (condición)?valor1:valor2 – 1

Cuando hay que tomar decisiones dentro de un algoritmo teníamos hasta ahora dos estructuras condicionales:

- if/else
- switch

El único comando que se puede utilizar en todas las situaciones es el if/else, en algunas situaciones se puede remplazar por el switch y ahora veremos una tercera forma.

No se trata de una estructura condicional sino de un operador condicional, veremos que también se puede utilizar en algunos casos particulares y que tiene por objetivo escribir código más conciso.

La sintaxis del operador condicional:
[valor] = (condición) ? valor1 : valor2

La condición del operador es igual a lo que hacemos en el if clásico. Si el resultado de la condición se verifica verdadero luego el 'valor1' se almacena en [valor] y si la condición se verifica falsa el 'valor2' se almacena en [valor].

Por ejemplo si queremos almacenar en una variable si la persona es mayor de edad podemos hacerlo con la siguiente sintaxis:

```
int edad = 3;
String estado = edad >= 18 ? "Mayor" : "Menor";
System.out.println(estado); // Menor
```

La variable estado almacena la cadena "Menor" debido a que la condición del operador se verifica falsa.

Como veremos es más concisa la condición que planteado con el if/else:

```
int edad=3;
String estado;
if (edad >= 18)
        estado = "Mayor";
else
        estado = "Menor";
Ssytem.out.println(estado); // Menor
```

Problema

Generar 10 valores aleatorios comprendidos entre 1 y 50. Mostrar cada uno de los valores y un mensaje indicando si tiene 1 o 2 dígitos.

```
public class Ejemplo1 {

    public static void main(String[] args) {
        for(int x=0; x<10; x++) {
            int valor=1+(int)(Math.random()*50);
```

```
            System.out.println("El valor: "+valor
                    +(valor<10 ? " Tiene un dígito" : " Tiene dos digitos"));
        }
    }
}
```

Si ejecutamos este será el resultado:

```
El valor: 4 Tiene un dígito
El valor: 24 Tiene dos digitos
El valor: 27 Tiene dos digitos
El valor: 50 Tiene dos digitos
El valor: 13 Tiene dos digitos
El valor: 43 Tiene dos digitos
El valor: 26 Tiene dos digitos
El valor: 31 Tiene dos digitos
El valor: 2 Tiene un dígito
El valor: 11 Tiene dos digitos
```

Problema

Mostrar los números del 1 al 100 e indicar si se trata de un número par o impar.

```
public class Ejemplo2 {

    public static void main(String[] args) {
        for (int x=1; x<=100; x++) {
            System.out.print(x + (x%2==0 ? " es par - " : " es impar - "));
            if(x%10==0)
                System.out.println();
        }
    }
}
```

Si ejecutamos este será el resultado:

```
1 es impar - 2 es par - 3 es impar - 4 es par - 5 es impar - 6 es par - 7 es impar - 8 es par - 9 es impar - 10 es par -
11 es impar - 12 es par - 13 es impar - 14 es par - 15 es impar - 16 es par - 17 es impar - 18 es par - 19 es impar - 20 es par -
21 es impar - 22 es par - 23 es impar - 24 es par - 25 es impar - 26 es par - 27 es impar - 28 es par - 29 es impar - 30 es par -
31 es impar - 32 es par - 33 es impar - 34 es par - 35 es impar - 36 es par - 37 es impar - 38 es par - 39 es impar - 40 es par -
41 es impar - 42 es par - 43 es impar - 44 es par - 45 es impar - 46 es par - 47 es impar - 48 es par - 49 es impar - 50 es par -
51 es impar - 52 es par - 53 es impar - 54 es par - 55 es impar - 56 es par - 57 es impar - 58 es par - 59 es impar - 60 es par -
61 es impar - 62 es par - 63 es impar - 64 es par - 65 es impar - 66 es par - 67 es impar - 68 es par - 69 es impar - 70 es par -
71 es impar - 72 es par - 73 es impar - 74 es par - 75 es impar - 76 es par - 77 es impar - 78 es par - 79 es impar - 80 es par -
81 es impar - 82 es par - 83 es impar - 84 es par - 85 es impar - 86 es par - 87 es impar - 88 es par - 89 es impar - 90 es par -
91 es impar - 92 es par - 93 es impar - 94 es par - 95 es impar - 96 es par - 97 es impar - 98 es par - 99 es impar - 100 es par -
```

Capítulo 221.- Operador condicional: (condición)?valor1:valor2 – 2

Problema:

Codificar un programa que muestre en pantalla los números del 1 al 100, sustituyendo los múltiplos de 3 por la palabra "Fizz" y su vez, los múltiplos de 5 por "Buzz". Para que los números que al tiempo, son múltiplos de 3 y 5, mostrar el mensaje "FizzBuzz".

Este sencillo programa y otros similares se los ha utilizado para descartar candidatos que se presentan en un puesto de programador, si no lo puede resolver en unos pocos minutos significa que no tiene sentido entrevistarlo por temas profundos a la programación por el que se está buscando un candidato.

```
public class Ejemplo1 {

    public static void main(String[] args) {
        for(int x=1; x<=100; x++) {
            System.out.println((x%3!=0 && x%5!=0) ? x
                    : (x%3==0 && x%5==0 ? "FizzBuz"
                            : (x%3==0 ? "Fiz"
                                    : x%5==0 ? "Buz":"")));
        }
    }
}
```

Si ejecutamos este será el resultado:

```
1       11      Fiz     31      41      Fiz     61      71      Fiz     91
2       Fiz     22      32      Fiz     52      62      Fiz     82      92
Fiz     13      23      Fiz     43      53      Fiz     73      83      Fiz
4       14      Fiz     34      44      Fiz     64      74      Fiz     94
Buz     FizzBuz Buz     Buz     FizzBuz Buz     Buz     FizzBuz Buz     Buz
Fiz     16      26      Fiz     46      56      Fiz     76      86      Fiz
7       17      Fiz     37      47      Fiz     67      77      Fiz     97
8       Fiz     28      38      Fiz     58      68      Fiz     88      98
Fiz     19      29      Fiz     49      59      Fiz     79      89      Fiz
Buz     Buz     FizzBuz Buz     Buz     FizzBuz Buz     Buz     FizzBuz Buz
```

Capítulo 222.- Operador condicional: (condición)?valor1:valor2 – 3

Problema propuesto:

Se tiene los siguientes vectores paralelos con las notas y nombres de alumnos:

String[] alumnos = {"Juan", "Ana", "Luis", "Carla", "Pedro", "Laura", "Maria"};

int[] notas = { 3, 7, 10, 9, 2, 8, 5};

Generar otro vector paralelo con el String "Libre" si la nota es menor a 4 y la cadena "regular" si la nota en mayor o igual a cuatro.

Emplear el operador condicional para cargar el vector paralelo.

```
public class Ejemplo1 {

    public static void main(String[] args) {
        String[] alumnos = {"Juan", "Ana", "Luis", "Carla", "Pedro", "Laura", "Maria"};
        int[] notas = { 3, 7, 10, 9, 2, 8, 5};
        String [] valoracion = new String[notas.length];
        System.out.println("Alumnos y puntuacion");
        for(int x=0; x<7; x++) {
            System.out.println(alumnos[x] +"  "+notas[x]);
        }
        System.out.println();
        for (int x=0; x<7; x++) {
            valoracion[x]=(notas[x]>=4)? "regular":"libre";
        }
        System.out.println("Alumnos y valoracion");
        for(int x=0; x<7; x++) {
            System.out.println(alumnos[x] +"  "+valoracion[x]);
        }
    }
}
```

Si ejecutamos este será el resultado:

```
Alumnos y puntuacion
Juan  3
Ana  7
Luis  10
Carla  9
Pedro  2
Laura  8
Maria  5
```

```
Alumnos y valoracion
Juan  libre
Ana  regular
Luis  regular
Carla  regular
Pedro  libre
Laura  regular
Maria  regular
```

Capítulo 223.- Excepciones en Java – try/catch

Java dispone de un mecanismo de captura (catch) ciertos tipos de errores que solo pueden ser detectados en tiempo de ejecución del programa.

Los ejemplos más comunes que podemos nombrar de excepciones:

- Tratar de convertir un entero un String que no contiene valores numéricos.
- Tratar de dividir por cero.
- Abrir un archivo de texto inexistente o que se encuentra bloqueado por otra aplicación.
- Conectar con un servidor de base de datos que no se encuentra activo.
- Acceder a subíndices de vectores y matrices fuera de rango.

La captura de excepciones nos permite crear programas muchos más robustos y tolerante a fallas que ocurren en escasas situaciones, pero en caso que se presenten disponemos de un algoritmo alternativo para reaccionar a dicha situación evitando que el programa finalice la ejecución.

Veremos un ejemplo sin captura de excepciones y luego la sintaxis implementando su captura.

Problema:

Realizar la carga de un número por teclado e imprimir su cuadrado.

Primero lo vamos a programar sin excepciones y por ver el error que nos muestra.

```
import java.util.Scanner;
public class Cuadrado {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);
        int num;
        System.out.print("Ingrese un valor enetero:");
        num=teclado.nextInt();
        int cuadrado=num*num;
        System.out.println("El cuadrado de "+num+" es "+cuadrado);
    }
}
```

Vamos a ejecutar el programa y en lugar de introducir un número introducimos una palabra.

```
Ingrese un valor enetero:hola
Exception in thread "main" java.util.InputMismatchException
        at java.base/java.util.Scanner.throwFor(Scanner.java:947)
        at java.base/java.util.Scanner.next(Scanner.java:1602)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2267)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2221)
        at Cuadrado.main(Cuadrado.java:8)
```

El programa se interrumpe y nos da un mensaje de error.

```
import java.util.InputMismatchException;
import java.util.Scanner;
public class Cuadrado {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);
        int num;
```

```
        try {
            System.out.print("Ingrese un valor enetero:");
            num=teclado.nextInt();
            int cuadrado=num*num;
            System.out.println("El cuadrado de "+num+" es "+cuadrado);
        } catch(InputMismatchException ex) {
            System.out.println("Debe cargar un número entero de forma obligatoria");
        }
    }
}
```

Vamos a ejecutar de nuevo el programa y vamos a introducir una palabra en lugar de un entero.

Ahora este será el resultado:

```
Ingrese un valor enetero:hola
Debe cargar un número entero de forma obligatoria
```

Ahora hasta que no introduzcamos un dato numérico no saldrá del bucle do/while.

```
import java.util.InputMismatchException;
import java.util.Scanner;
public class Cuadrado {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);
        int num;
        boolean continua;
        do {
                try {
                    continua=false;
                    System.out.print("Ingrese un valor enetero:");
                    num=teclado.nextInt();
                    int cuadrado=num*num;
                    System.out.println("El cuadrado de "+num+" es "+cuadrado);
                } catch(InputMismatchException ex) {
                    System.out.println("Debe cargar un número entero de forma obligatoria");
                    teclado.next();
                    continua=true;
                }
        }while(continua);

    }
}
```

Ejecutamos y este será el resultado:

```
Ingrese un valor enetero:hola
Debe cargar un número entero de forma obligatoria
Ingrese un valor enetero:hola
Debe cargar un número entero de forma obligatoria
Ingrese un valor enetero:5
El cuadrado de 5 es 25
```

Capítulo 224.- Excepciones – múltiples catch para un try – 1

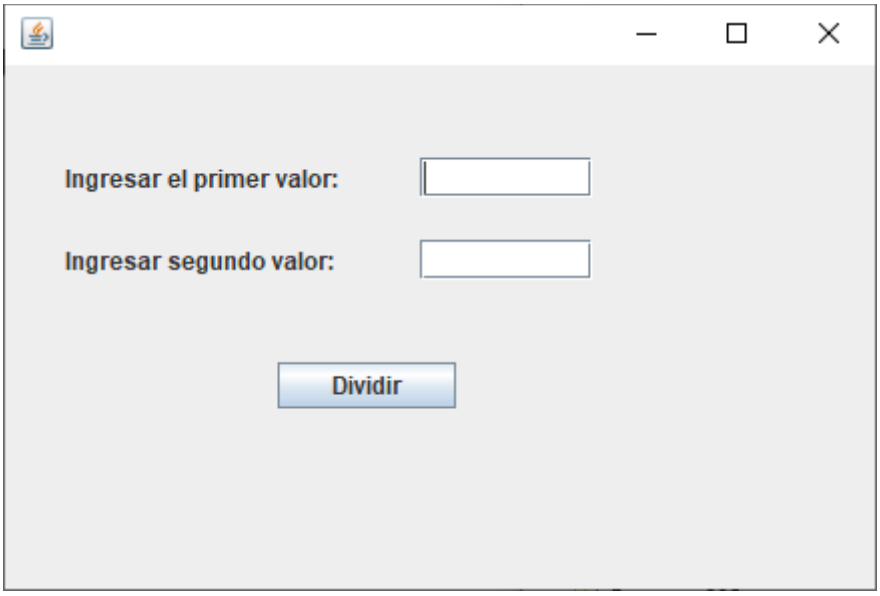
```
try {
  [instrucciones 1]
} catch([excepción 1]) {
  [instrucciones 2]
}
catch([excepción 2]) {
  [instrucciones 3]
}
catch([excepción n]) {
  [instrucciones n]
}
```

Problema:

Confeccionar una aplicación visual que permita ingresar en controles de tipo JTextField dos valores enteros. Al presionar un botón mostrar la división del primero respecto al segundo en el título del JFrame o un mensaje si no se ingresan datos o la división no se puede efectuar.

Trabajaremos con WindowBuilder.

Este tiene que ser el diseño:



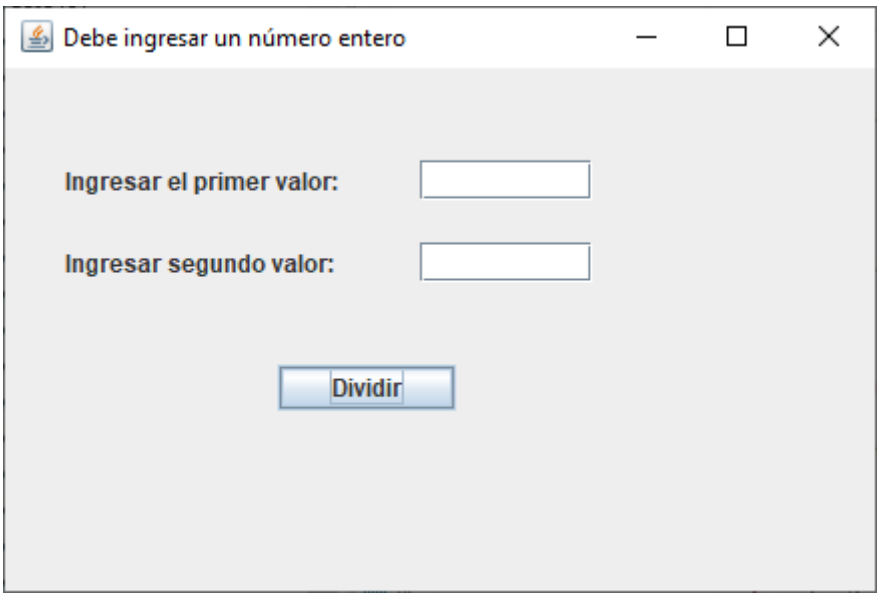
Hemos renombrado los JTextField por tf1 y tf2.

El código cuando presionamos el botón:

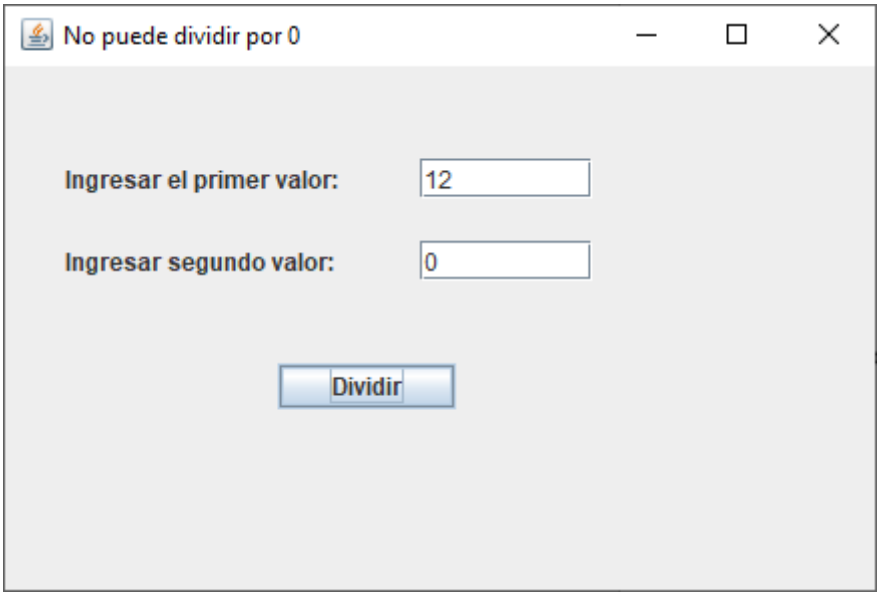
```
JButton btnNewButton = new JButton("Dividir");
btnNewButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String cad1=tf1.getText();
        String cad2=tf2.getText();
        try {
            int valor1=Integer.parseInt(cad1);
            int valor2=Integer.parseInt(cad2);
            int resultado=valor1/valor2;
            setTitle("La división es: "+resultado);
        }catch (NumberFormatException ex) {
            setTitle("Debe ingresar un número entero");
```

```
        }catch (ArithmeticException ex) {
            setTitle("No puede dividir por 0");
        }
    }
```

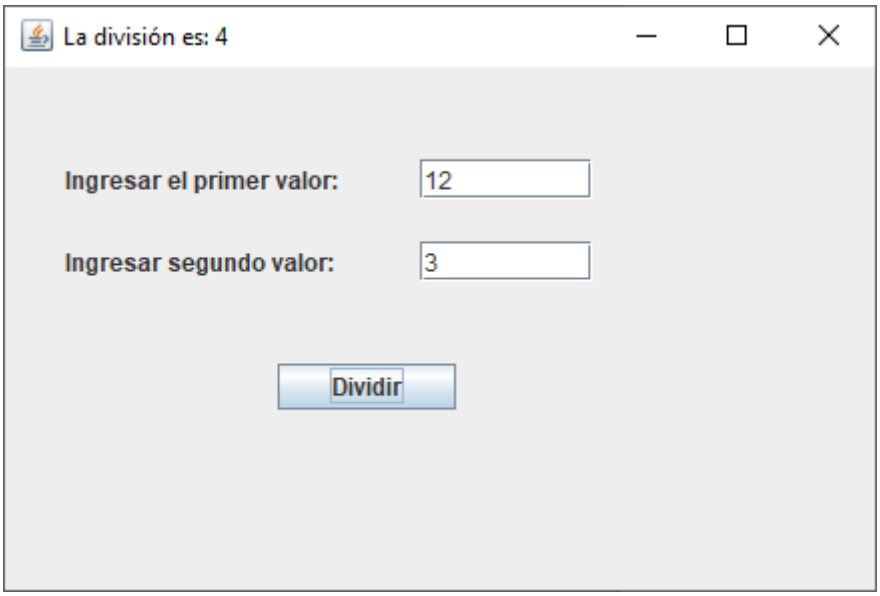
No introducimos valores y presionamos el botón:



En el segundo valor ponemos un 0.



Introducimos dos números enteros:


La división es: 4
Ingresar el primer valor:
12
Ingresar segundo valor:
3
Dividir

Capítulo 225.- Excepciones – múltiples catch para un try – 2

Problema:

Declarar un vector de 10 elementos enteros. Permitir que el usuario ingrese un subíndice del vector y nos muestre el contenido de dicho componente. Atrapar las excepciones de fuera de rango del vector y si ingresa un valor no entero.

```
import java.util.Scanner;
import java.util.InputMismatchException;
public class MultiplesCatch {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);
        int[] vec = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
        int indice;
        try {
            System.out.print("Ingrese un valor del 0 al 9:");
            indice=teclado.nextInt();
            System.out.println("En el vector se almacena en la posición "+indice+" el valor "+vec[indice]);
        }catch (InputMismatchException ex){
            System.out.println("Debe ingresar un valor entero");
        }catch(IndexOutOfBoundsException es) {
            System.out.println("El valor debe estar entre 0 y 9");
        }
    }
}
```

Vamos a ingresar un texto en lugar de un valor entero.

```
Ingrese un valor del 0 al 9:hola
Debe ingresar un valor entero
```

Ahora vamos a ingresar un valor superior a 9.

```
Ingrese un valor del 0 al 9:10
El valor debe estar entre 0 y 9
```

Que nos vaya preguntando hasta que introduzcamos un valor entero entre 0 y 9.

```
import java.util.Scanner;
import java.util.InputMismatchException;
public class MultiplesCatch {

    public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);
        int[] vec = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
        int indice;
        boolean error=false;
        do {
            try {
                System.out.print("Ingrese un valor del 0 al 9:");
                indice=teclado.nextInt();
                System.out.println("En el vector se almacena en la posición "+indice+" el valor "+vec[indice]);
                error=false;
            }catch (InputMismatchException ex){
                System.out.println("Debe ingresar un valor entero");
                error=true;
                teclado.next();
            }catch(IndexOutOfBoundsException es) {
                System.out.println("El valor debe estar entre 0 y 9");
                error=true;
                teclado.next();
        }
        } while (error);
    }
}
```

cuando ejecutemos este será el resultado:

```
Ingrese un valor del 0 al 9:Hola
Debe ingresar un valor entero
Ingrese un valor del 0 al 9:12
El valor debe estar entre 0 y 9
10
Ingrese un valor del 0 al 9:7
En el vector se almacena en la posición 7 el valor 80
```

Capítulo 226.- Excepciones – no verificadas y verificadas – 1

Existe dos tipos de excepciones en Java:

- No verificadas
- Verificadas

Las excepciones no verificadas son aquellas que dejan al programador tomar la decisión de la conveniencia de atraparla o no.

Todas las excepciones vistas hasta ahora son 'no veriricadas': InputMismatchException, ArtihmeticException, NumberfromatException y IndesOutOgBoundsException. Existe en el API de Java muchas otras excepciones de este tipo.

Si queremos una lista completa de excepciones no verificadas en Pava podemos visitar la documentación oficial. Todas estas clases heredan de la clase java.lang.RuntimeException.

Excepciones verificadas

Este nuevo tipo de excepción tiene la característica que deben ser capturadas en forma obligatoria por nuestro programa, si no la capturamos no se compila nuestra aplicación.

Las excepciones que se definen como verificadas generalmente son errores que no son directamente de nuestro programa sino a errores que surgen por ejemplo al intentar conectarnos a una base de datos, abrir un archivo inexistente, problemas de conexión a red, etc.

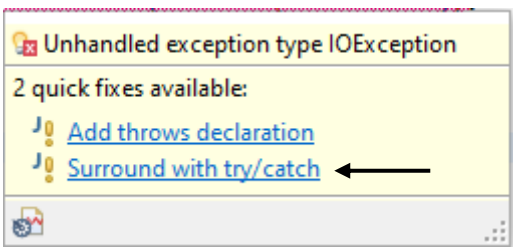
La captura y tratamiento de la excepción verificada es idéntica a las excepciones no verificadas, con la salvedad que no podemos omitirla.

Problema:

Crear un archivo de texto con dos líneas. Luego proceder a leer el contenido del archivo de texto y mostrarlos en pantalla.

```
1  import java.io.FileWriter;
2  import java.io.File;
3
4  public class CreacionLecturaArchivoTxt {
5
6      public static void main(String[] args) {
7          FileWriter fw=new FileWriter(new File("datos.txt"));
8
9      }
10 }
```

Nos genera el siguiente error:



Seleccionaremos Surround with try/catch.

```
1  import java.io.FileWriter;
2  import java.io.IOException;
3  import java.io.File;
4
5  public class CreacionLecturaArchivoTxt {
6
7      public static void main(String[] args) {
8          try {
9              FileWriter fw=new FileWriter(new File("datos.txt"));
10         } catch (IOException e) {
11             // TODO Auto-generated catch block
12             e.printStackTrace();
13         }
14
15     }
16 }
```

Ya no aparece el error.

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.File;
import java.io.BufferedWriter;

public class CreacionLecturaArchivoTxt {

    public static void main(String[] args) {
        try {
            FileWriter fw=new FileWriter(new File("datos.txt"));
            BufferedWriter bw=new BufferedWriter(fw);
            bw.write("Linea 1");
            bw.newLine();
            bw.write("Linea 2");
            bw.close();
            fw.close();
        } catch (IOException e) {
            System.out.println("Problemas en la creación del archivo.");
        }
    }
}
```

Cuando lo ejecutemos nos creará el archivo datos.txt con el contenido Linea1 y Linea 2.

Ahora vamos a agregar el código para poder leer el archivo.

```
import java.io.FileWriter;
import java.io.FileReader;
import java.io.IOException;
import java.io.File;
import java.io.BufferedReader;
import java.io.BufferedWriter;

public class CreacionLecturaArchivoTxt {

    public static void main(String[] args) {
        try {
            FileWriter fw=new FileWriter(new File("datos.txt"));
            BufferedWriter bw=new BufferedWriter(fw);
            bw.write("Linea 1");
            bw.newLine();
            bw.write("Linea 2");
```

```
            bw.close();
            fw.close();
        } catch (IOException e) {
            System.out.println("Problemas en la creación del archivo.");
        }

        try {
            FileReader fr=new FileReader(new File("datos.txt"));
            BufferedReader br=new BufferedReader(fr);
            String linea=br.readLine();
            while (linea!=null) {
                System.out.println(linea);
                linea=br.readLine();
            }

        } catch (IOException e) {
            System.out.println("Problemas con la lectura del archivo.");
            System.out.println(e.getMessage());
        }
    }
}
```

Si ejecutamos este será el resultado:

```
Linea 1
Linea 2
```

Vamos a cambiar el nombre del archivo en la lectura para que se produzca un error.

```
try {
    FileReader fr=new FileReader(new File("datos1.txt"));
    BufferedReader br=new BufferedReader(fr);
    String linea=br.readLine();
    while (linea!=null) {
        System.out.println(linea);
        linea=br.readLine();
    }

} catch (IOException e) {
    System.out.println("Problemas con la lectura del archivo.");
    System.out.println(e.getMessage());
}
```

Ejecutamos de nuevo:

```
Problemas con la lectura del archivo.
datos1.txt (El sistema no puede encontrar el archivo especificado)
```

Capítulo 227.- Excepciones no verificadas y verificadas – 2

Comando throws para elevar excepciones

Problema:

Leer un archivo de texto no existente en un método estático. Elevar la excepción.

```
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.BufferedReader;

public class LecturaTXT {

    public static void leer() {
        try {
            FileReader fr=new FileReader(new File("datos.txt"));
            BufferedReader br=new BufferedReader(fr);
            String linea=br.readLine();
            while (linea!= null) {
                System.out.println(linea);
                linea=br.readLine();
            }
            br.close();
            fr.close();

        } catch (IOException e) {

        }
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }
}
```

Si no controlamos las excepciones el programa nos dará un error.

```
1  import java.io.File;
2  import java.io.FileReader;
3  import java.io.IOException;
4  import java.io.BufferedReader;
5
6  public class LecturaTXT {
7
8      public static void leer() {
9
10             FileReader fr=new FileReader(new File("datos.txt"));
11             BufferedReader br=new BufferedReader(fr);
12             String linea=br.readLine();
13             while (linea!= null) {
14                 System.out.println(linea);
15                 linea=br.readLine();
16             }
```

```
17              br.close();
18              fr.close();
19
20      }
21
22      public static void main(String[] args) {
23          // TODO Auto-generated method stub
24
25      }
26  }
```

Pero hay la posibilidad de elevar las excepciones con el comando throws.

```
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.BufferedReader;

public class LecturaTXT {

    public static void leer() throws IOException {

            FileReader fr=new FileReader(new File("datos.txt"));
            BufferedReader br=new BufferedReader(fr);
            String linea=br.readLine();
            while (linea!= null) {
                System.out.println(linea);
                linea=br.readLine();
            }
            br.close();
            fr.close();

    }

    public static void main(String[] args) {
        try {
            leer();
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Ahora la excepción la tendremos que poner en el método main.

Vamos a ejecutar:

```
datos.txt (El sistema no puede encontrar el archivo especificado)
```

Podemos elevar la excepción:

```
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.BufferedReader;

public class LecturaTXT {

    public static void leer() throws IOException {
```

```
            FileReader fr=new FileReader(new File("datos.txt"));
            BufferedReader br=new BufferedReader(fr);
            String linea=br.readLine();
            while (linea!= null) {
                System.out.println(linea);
                linea=br.readLine();
            }
            br.close();
            fr.close();

    }

    public static void main(String[] args)  throws IOException {
        leer();
    }
}
```

Si ejecutamos ahora veremos los mensaje de excepción de la máquina virtual de Java.

```
Exception in thread "main" java.io.FileNotFoundException: datos.txt (El sistema no puede encontrar el archivo especificado)
        at java.base/java.io.FileInputStream.open0(Native Method)
        at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:158)
        at java.base/java.io.FileReader.<init>(FileReader.java:75)
        at LecturaTXT.leer(LecturaTXT.java:10)
        at LecturaTXT.main(LecturaTXT.java:23)
```

Capítulo 228.- Excepciones – bloque finally

El manejo de excepciones en Java puede incluir otro bloque llamado 'finally'.

```
try {
  [instrucciones 1]
} catch([excepción 1]) {
  [instrucciones 2]
} finally {
  [instrucciones 3]
}
```

El objetivo de este bloque es liberar recursos que se solicitan en el bloque try. El bloque finally se ejecuta siempre, inclusive si se genera la captura de una excepción.

La recursos más comunes que se deben liberar son las conexiones a base de datos, uso de archivos y conexiones a red. Un recurso que no se libera impide que otro programa lo pueda utilizar. Al disponer la liberación de recursos en el bloque 'finally' nos aseguramos que todo recurso se liberará, inclusive aunque se dispare una excepción.

Tener en cuenta que si no se dispara ninguna excepción en un bloque try luego de esto se ejecuta el bloque 'finally'.

El bloque finally es opcional y en su caso de estar presente se coloca después del último bloque catch.

Problema:

Crear un archivo de texto con dos líneas. Luego proceder a leer el contenido del archivo de texto y mostrarlo por pantalla. Asegurarse de liberar el archivo luego de trabajar con el mismo.

Este será el código:

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.File;
import java.io.FileReader;

public class CreacionLecturaTxt {

	public static void main(String[] args) {
		FileWriter fw=null;
		BufferedWriter bw=null;
		try {
			fw=new FileWriter(new File("datos.txt"));
			bw=new BufferedWriter(fw);
			bw.write("Linea 1");
			bw.newLine();
			bw.write("Linea 2");
		}catch(IOException ex) {
			System.out.println(ex.getMessage());
		}finally {
			try {
				if (bw!=null)
					bw.close();
				if (fw!=null)
```

```
                        fw.close();
                }catch (IOException ex) {
                        System.out.println(ex.getMessage());
                }
        }

        FileReader fr=null;
        BufferedReader br=null;
        try {
                fr=new FileReader(new File("datos.txt"));
                br=new BufferedReader(fr);
                String linea=br.readLine();
                while(linea!=null) {
                        System.out.println(linea);
                        linea=br.readLine();
                }
        }catch (IOException ex) {
                System.out.print(ex.getMessage());
        }finally {
                try {
                        if (br!=null)
                                br.close();
                        if (fr!=null)
                                fr.close();
                        System.out.println("Entro a cerrar el archivo.");

                }catch(IOException ex) {
                        System.out.println(ex.getMessage());
                }
        }
    }
}
```

Si ejecutamos este será el resultado:

```
Linea 1
Linea 2
Entro a cerrar el archivo.
```

Si en modo de apertura cambiamos el nombre del archivo.

```
FileReader fr=null;
BufferedReader br=null;
try {
    fr=new FileReader(new File("datos1.txt"));
    br=new BufferedReader(fr);
    String linea=br.readLine();
    while(linea!=null) {
        System.out.println(linea);
        linea=br.readLine();
    }
}catch (IOException ex) {
    System.out.println(ex.getMessage());
}finally {
```

```
        try {
            if (br!=null)
                br.close();
            if (fr!=null) {
                fr.close();
                System.out.println("Entro a cerrar el archivo.");
                }
        }catch(IOException ex) {
            System.out.println(ex.getMessage());
        }
    }
```

Vamos a ejecutar:

```
datos1.txt (El sistema no puede encontrar el archivo especificado)
```

Capítulo 229.- Excepciones – lanzar una excepción mediante comando throw – 1

Hemos visto hasta ahora cual es la sintaxis para capturar excepciones que lanzan métodos. Ahora veremos como podemos lanzar una excepción para que sea eventualmente capturada posteriormente.

Hay que definir con juicio que métodos de una clase deben lanzar excepciones cuando el dato es incorrecto debido que luego se hace más difícil consumir dicho método.

Aquellas partes críticas de nuestra clase pueden lanzar excepciones para que sean más robustas. Por ejemplo si tenemos una clase 'ClienteBanco' y queremos controlar que nunca pueda sacar más dinero del que tiene depositado, el método extraer puede lanzar una excepción evitando que quede en negativo el importe del cliente.

Problema:

Declarar una clase llamada 'PersonaAdulta' con los atributos nombre y edad. Lanzar una excepción de tipo Exception en caso que llegue en el constructor una edad menor de 18 años.

En este problema hemos tomado la decisión de validar y lanzar una excepción en caso que se intente la creación de un objeto de la clase PersonaAdulta con una edad inferior a 18. Esto es porque a juicio del programador considera que es muy importante que nunca haya un objeto de tipo PersonaAdulta que sea menor de edad.

```
public class PersonaAdulta {
    private String nombre;
    private int edad;

    public PersonaAdulta(String nombre, int edad) throws Exception {
        this.nombre=nombre;
        if (edad<18) {
            throw new Exception("La perona no es mayor de edad "+nombre+" porque tiene "+edad+" años");
        }
        this.edad=edad;
    }

    public void fijarEdad(int edad) throws Exception {
        if (edad<18) {
            throw new Exception("La perona no es mayor de edad "+nombre+" porque tiene "+edad+" años");
        }
        this.edad=edad;
    }

    public void imprimir() {
        System.out.println(nombre +" tiene "+edad+" años");
    }

    public static void main(String[] args) {
        try {
            PersonaAdulta persona1=new PersonaAdulta("Juan", 25);
            persona1.imprimir();
        }catch (Exception ex) {
            System.out.println(ex.getMessage());
        }
    }
}
```

Si ejecutamos este será el resultado:

```
Juan tiene 25 años
```

Vamos a añadir otra persona menor de edad:

```
public static void main(String[] args) {
    try {
        PersonaAdulta persona1=new PersonaAdulta("Juan", 25);
        persona1.imprimir();
        PersonaAdulta persona2=new PersonaAdulta("Luis", 12);
        persona2.imprimir();
    }catch (Exception ex) {
        System.out.println(ex.getMessage());
    }
}
```

Ejecutamos de nuevo.

```
Juan tiene 25 años
La perona no es mayor de edad Luis porque tiene 12 años
```

Ahora a Luis lo vamos a agregar con 22 años y a continuación la cambio la edad a 12.

```
public static void main(String[] args) {
    try {
        PersonaAdulta persona1=new PersonaAdulta("Juan", 25);
        persona1.imprimir();
        PersonaAdulta persona2=new PersonaAdulta("Luis", 22);
        persona2.fijarEdad(12);
        persona2.imprimir();
    }catch (Exception ex) {
        System.out.println(ex.getMessage());
    }
}
```

Si ejecutamos este será el resultado:

```
Juan tiene 25 años
La perona no es mayor de edad Luis porque tiene 12 años
```

Capítulo 230.- Excepciones – lanzar una excepción mediante comando throw – 2

Problema propuesto:

Implementar la clase Operaciones. Debe tener dos atributos enteros. Desarrollar los métodos para calcular su suma, resta, multiplicación y división, imprimir dichos resultados. Lanzar una excepción en el método que calcula la división si el segundo valor es cero.

```
public class Operaciones {
    private int num1;
    private int num2;

    public Operaciones(int num1, int num2) {
        this.num1=num1;
        this.num2=num2;
    }

    public void sumar() {
        int suma=num1+num2;
        System.out.println("La suma de "+num1+" más "+num2+ " es de "+suma);
    }

    public void restar() {
        int resta=num1-num2;
        System.out.println("La resta de "+num1+" menos "+num2+ " es de "+resta);
    }

    public void multiplicar() {
        int multiplica=num1*num2;
        System.out.println("La multiplicacion de "+num1+" por "+num2+ " es de "+multiplica);
    }

    public void dividir() throws Exception {
        if(num2==0)
            throw new Exception("Para la división en dividendo no puede ser 0");
        int divide=num1/num2;
        System.out.println("La divisdión de "+num1+" entre "+num2+ " es de "+divide);
    }

    public static void main(String[] args) {
        try {
            Operaciones operacion1 = new Operaciones(10,5);
            Operaciones operacion2 = new Operaciones(5, 0);
            operacion1.sumar();
            operacion1.restar();
            operacion1.multiplicar();
            operacion1.dividir();
            System.out.println();
            operacion2.sumar();
            operacion2.restar();
            operacion2.multiplicar();
            operacion2.dividir();
        }catch(Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Si ejecutamos este será el resultado:

```
La suma de 10 más 5 es de 15
La resta de 10 menos 5 es de 5
La multiplicacion de 10 por 5 es de 50
La divisdión de 10 entre 5 es de 2

La suma de 5 más 0 es de 5
La resta de 5 menos 0 es de 5
La multiplicacion de 5 por 0 es de 0
Para la división en dividendo no puede ser 0
```

Capítulo 231.- Excepciones – lanzar una excepción mediante comando throw – 3

Problema propuesto:

Declarar una clase 'Cliente' que represente un cliente de un banco. Definir los siguientes atributos y métodos:

Cliente
- atributos
  - nombre
  - importe
- métodos
  - Constructor
  - depositar
  - extraer
  - imprimir

Generar una excepción si se intenta extraer más dinero del que tiene el atributo 'importe'.

```
public class Cliente {
    private String nombre;
    private int importe;

    public Cliente(String nombre, int importe) {
        this.nombre=nombre;
        this.importe=importe;
    }

    public void depositar(int imp) {
        importe=importe+imp;
    }

    public void extraer(int imp) throws Exception{
        if(importe<imp)
            throw new Exception ("El cliente no puede extraer más de lo que tiene");
        importe=importe-imp;
        System.out.println("Operación realizada con exito");
    }

    public void imprimir() {
        System.out.println("El cliente "+nombre+" tiene un saldo en cuenta de "+importe);
    }

    public static void main(String[] args) {
        try {
            Cliente cliente1 = new Cliente("Juan", 1500);
            cliente1.imprimir();
            Cliente cliente2 = new Cliente("Luis", 2500);
            cliente2.imprimir();
            cliente1.depositar(500);
            cliente1.imprimir();
            cliente2.extraer(1500);
            cliente2.imprimir();
            cliente2.extraer(1200);

        }catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Si ejecutamos este será el resultado:

```
El cliente Juan tiene un saldo en cuenta de 1500
El cliente Luis tiene un saldo en cuenta de 2500
El cliente Juan tiene un saldo en cuenta de 2000
Operación realizada con exito
El cliente Luis tiene un saldo en cuenta de 1000
El cliente no puede extraer más de lo que tiene
```