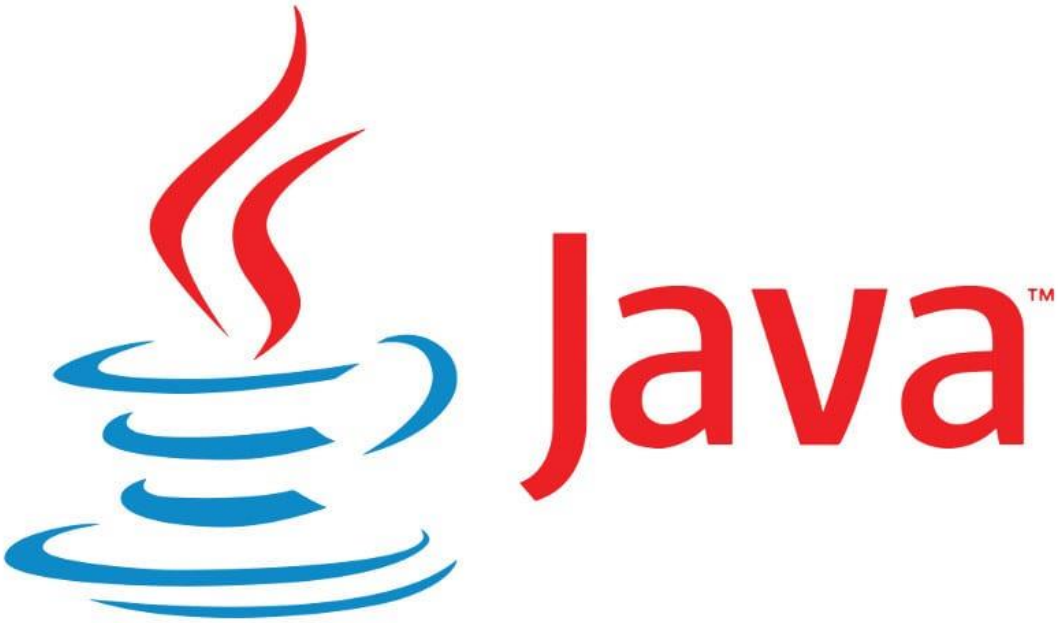
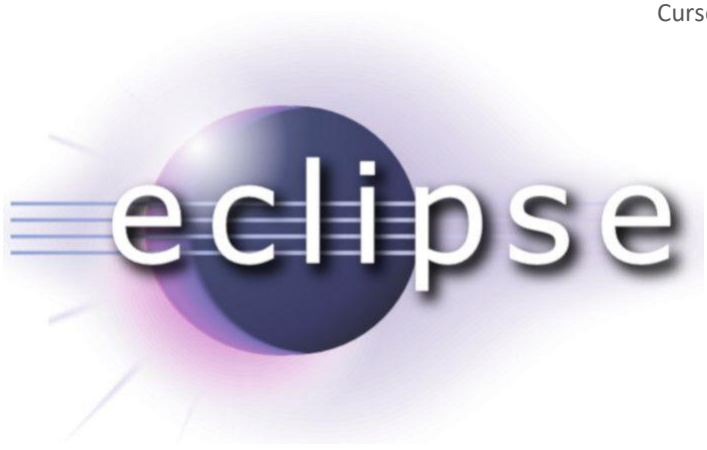


CURSO INICIACIÓN JAVA VI

Tutoriales de pildorasinformaticas

Descripción breve

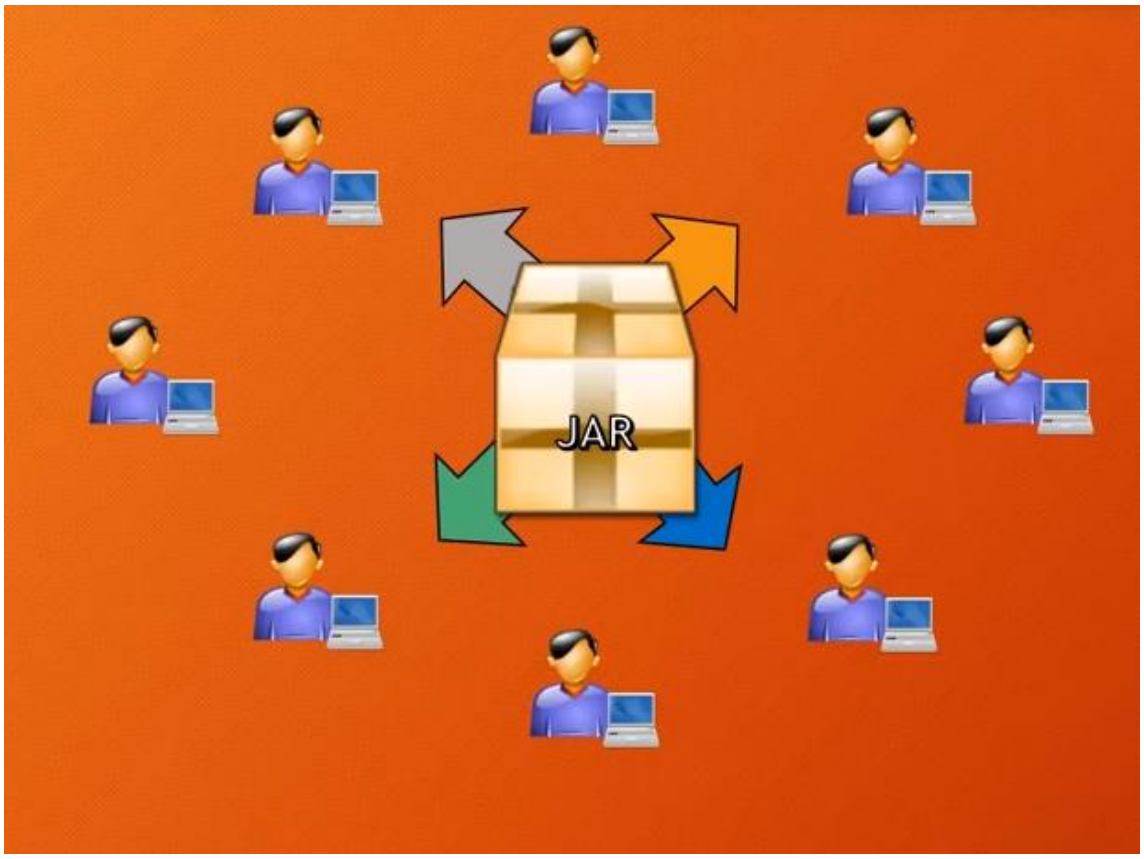
Curso introductorio de Java por pildorasinformaticas.



Pere Manel Verdugo Zamora
pereverdugo@gmail.com

Despliegue Aplicaciones. Java Web Start. (Vídeo 141)

Distribución de aplicaciones Java



- ¿Qué ocurre si hay que distribuir la aplicación a miles de usuarios?
- ¿Qué ocurre si se actualiza la aplicación después de haberla distribuido?
- ¿Qué ocurre si mis aplicaciones no se adapta a un Applet?

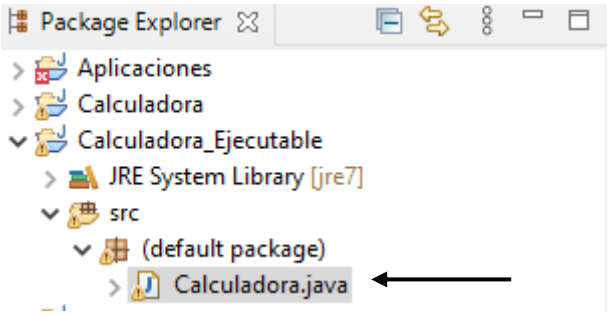
Java Web Start

Permite distribuir una aplicación Java en un entorno de red como Internet..

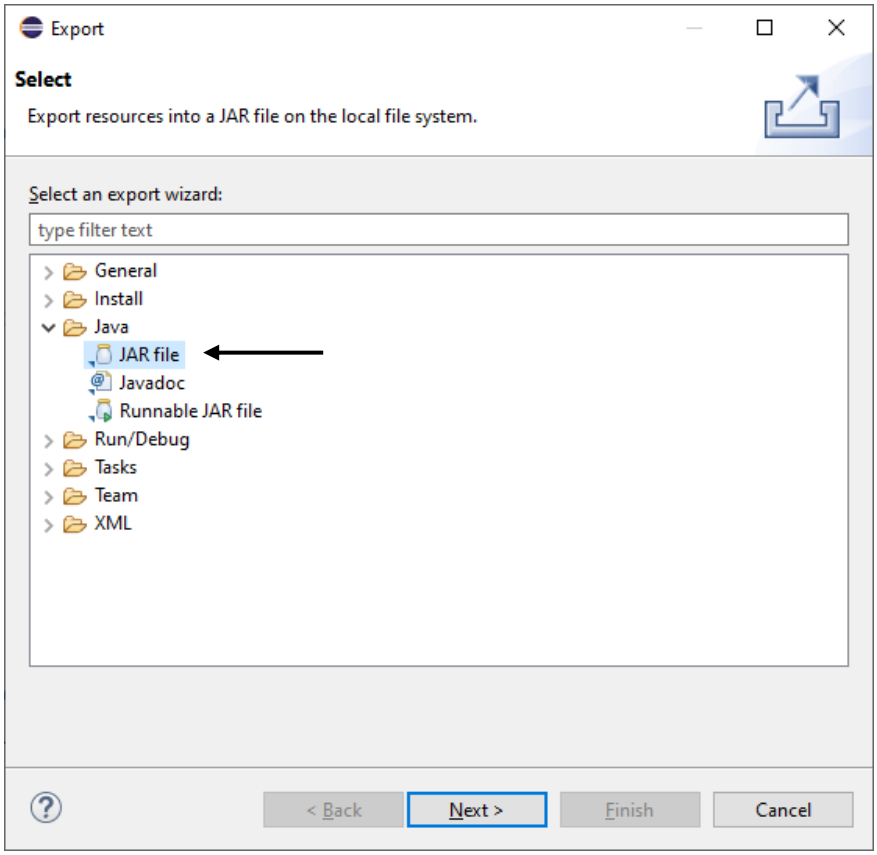
Subir una aplicación empaquetada jar a un servidor remoto.
Y junto con la aplicación java construir un fichero XML que veremos a continuación.
Este fichero XML debe de tener la extensión JNLP.
Cuando el usuario acceda a esta página desde el navegador se proceda a la descarga de dicha aplicación.

Las siglas JNLP vienen de (Java Network Launching Protocol).

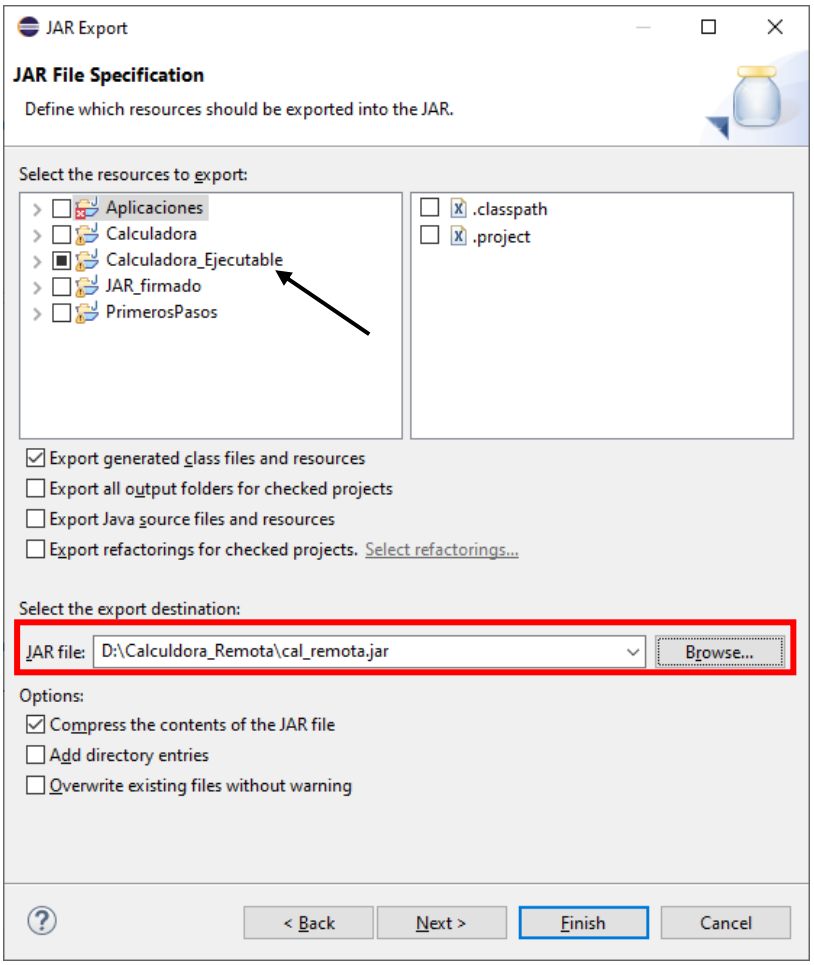
Vamos a Eclipse y abrimos la calculadora que teníamos en un proyecto llamado Calculadora_Ejecutable.



Del menú File seleccionamos Export…

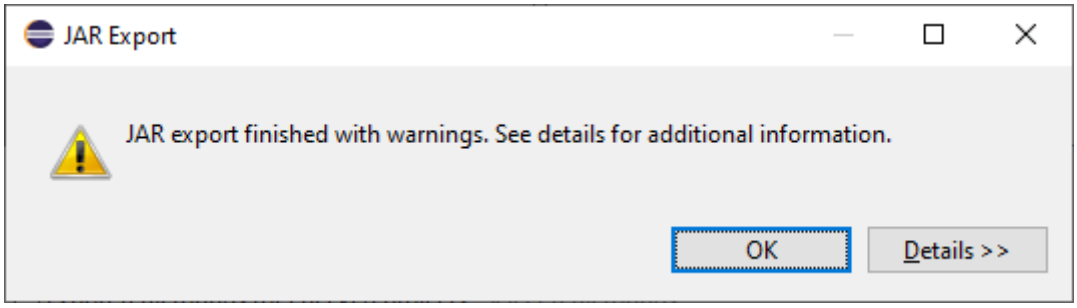


Seleccionamos JAR file, siguiendo del botón Next.



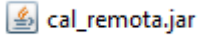
Seleccionamos el proyecto a empaquetar y el destino del mismo.

Le damos al botón Finish.

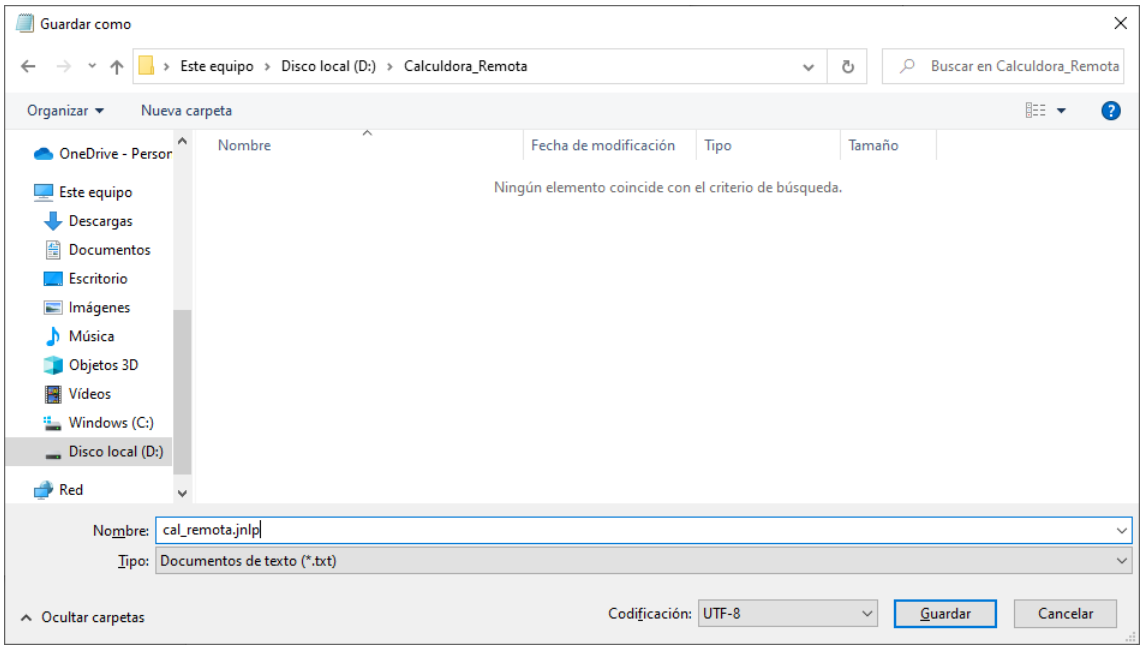


Pulsamos OK.

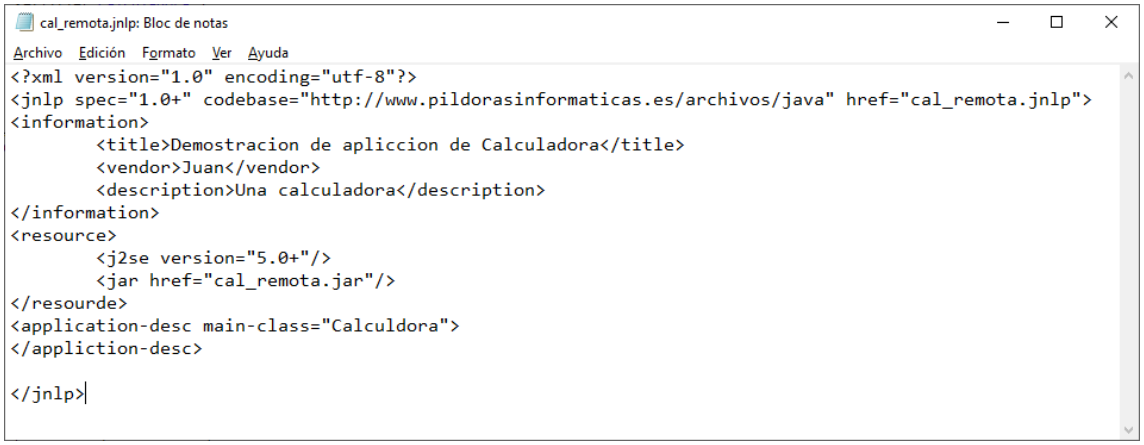
Ya tenemos nuestro archivo.



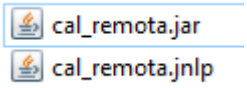
Ahora con el bloc de notas vamos a crear un archivo con extensión jnlp.



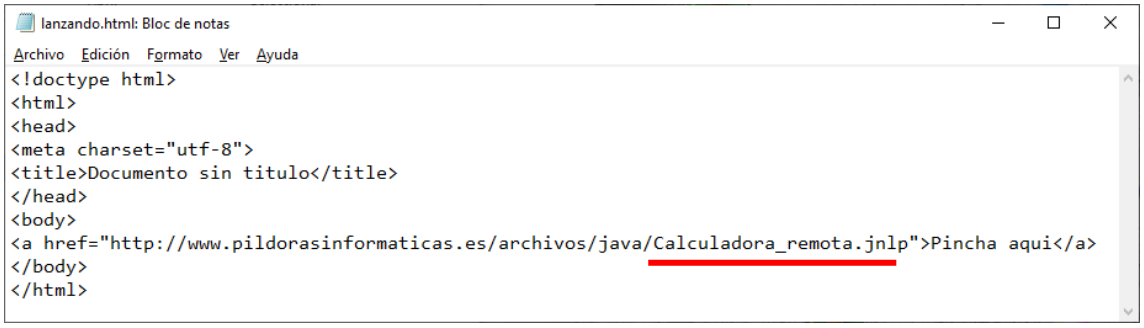
Lo guardaremos donde tenemos el proyecto empaquetado.



Ahora tenemos dos archivos.



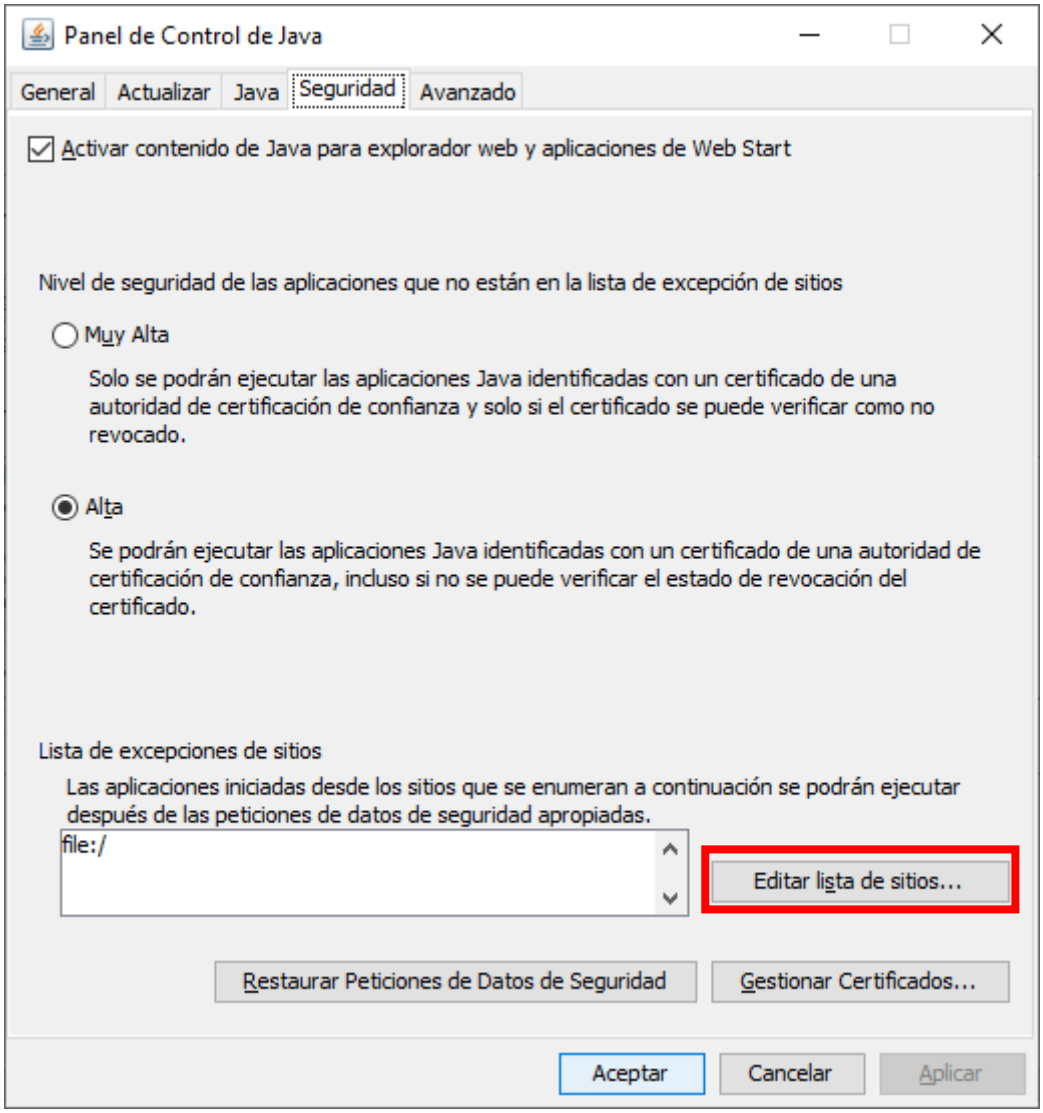
Ahora vamos a realizar un archivo html con el bloc de notas.

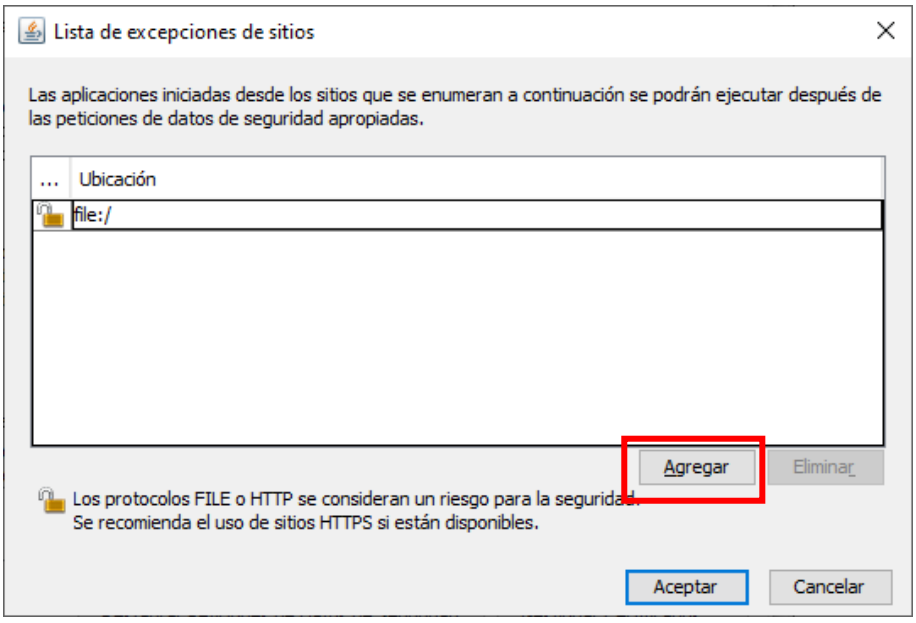


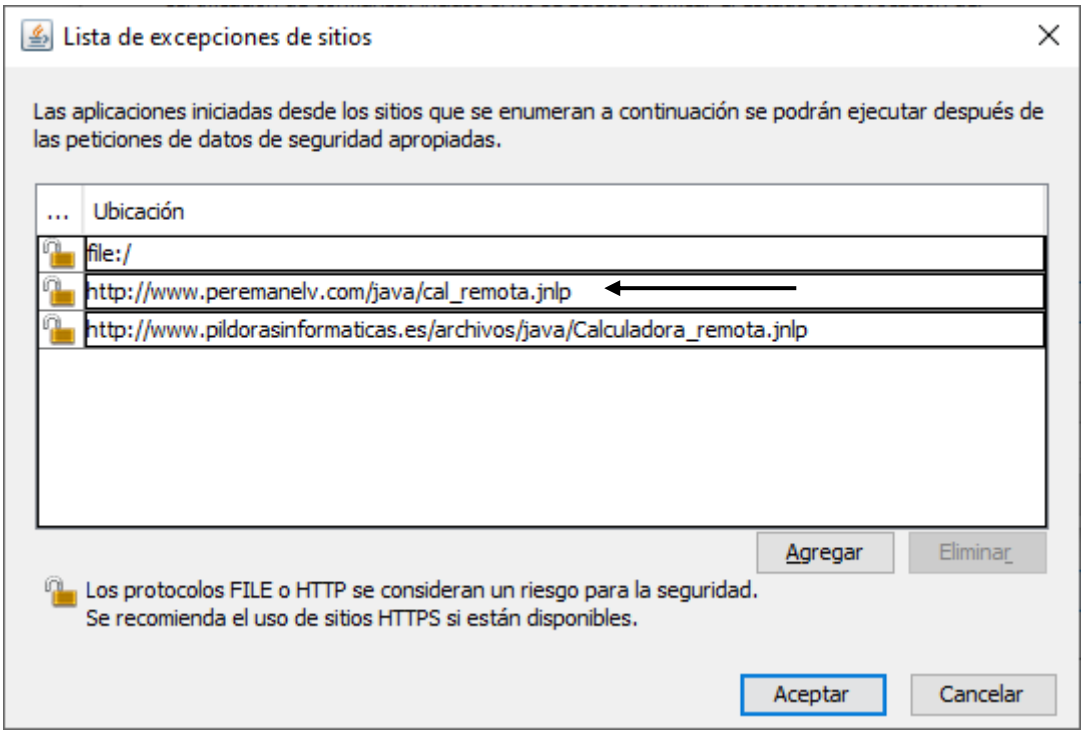
El nombre Calculadora_remota.jnlp es el ejemplo del profesor, en nuestro caso si pudiéramos subir los archivos sería cal_remota.jnlp.

En el panel de control de java tenemos que agregar este sitio como sitio de confianza.

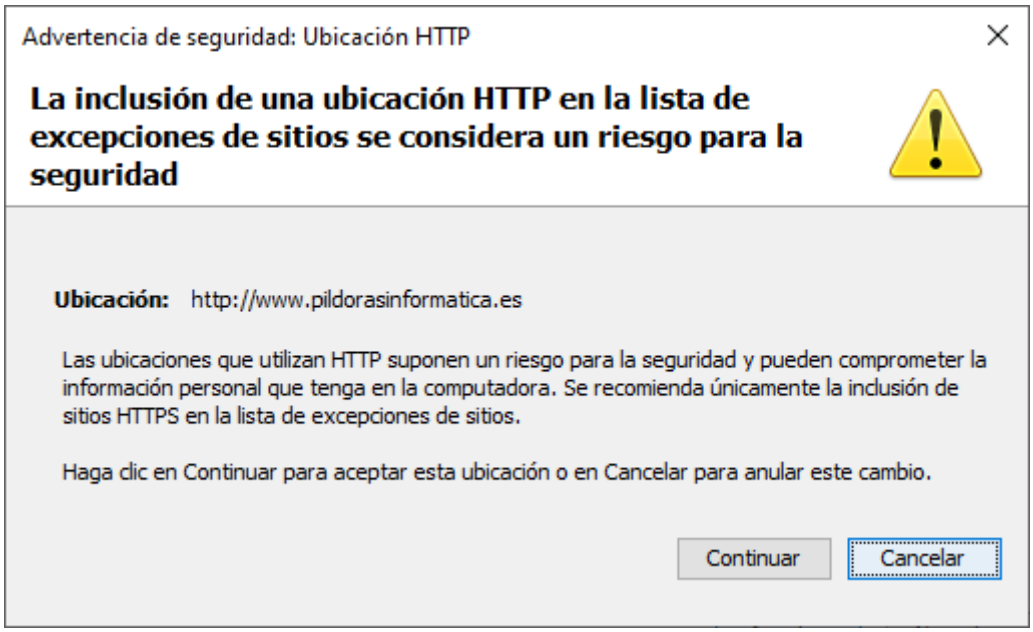
En la pestaña seguridad.







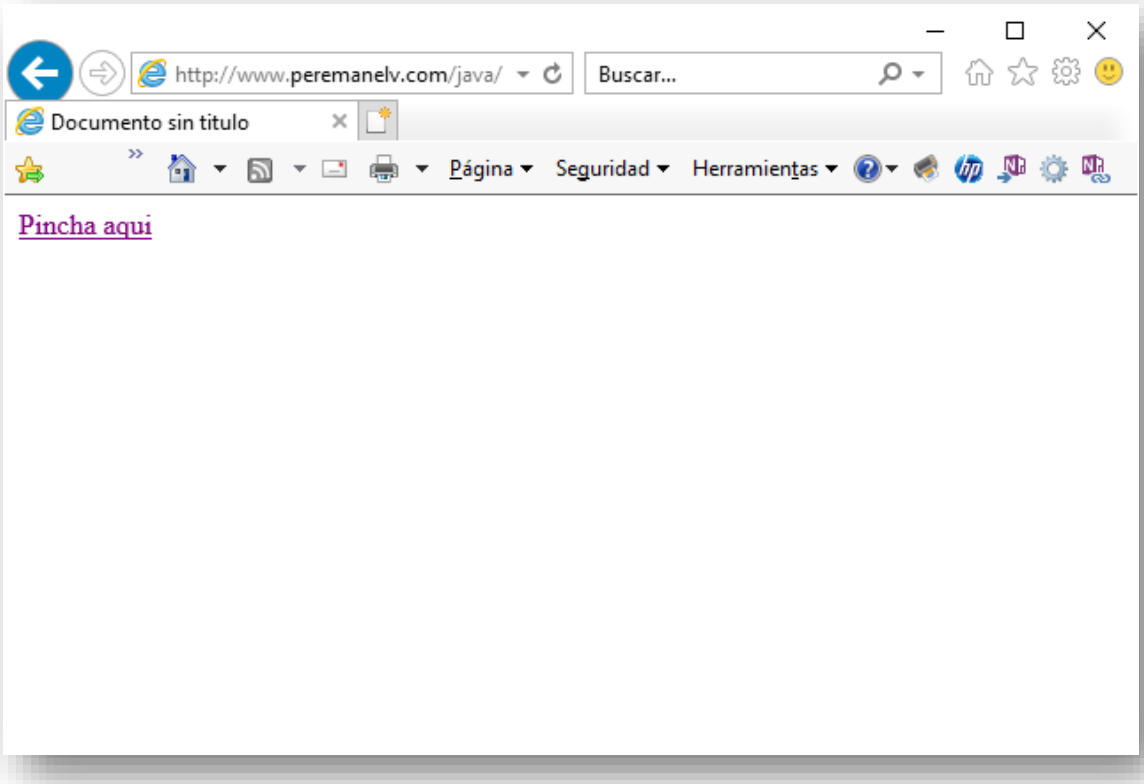
Seleccionamos Agregar.



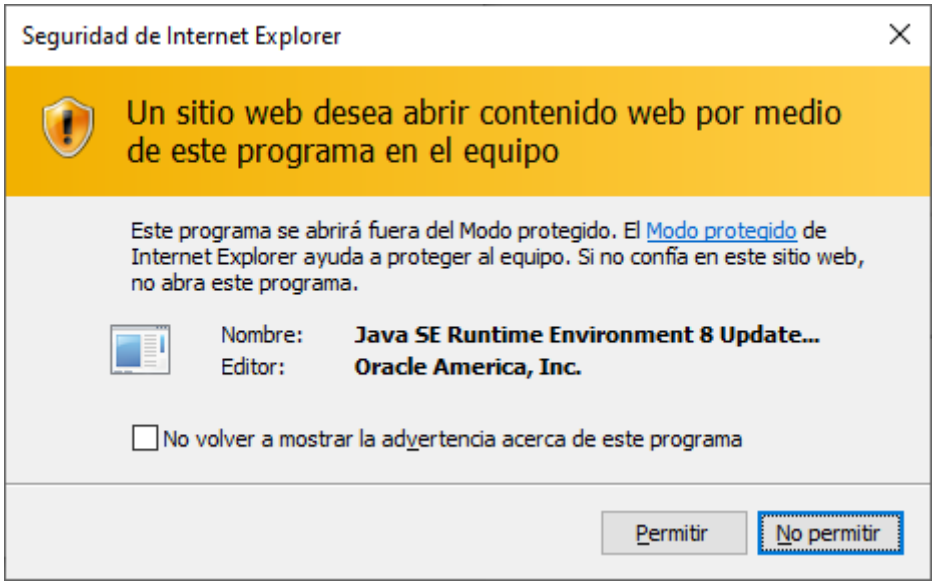
Nos avisa de que no lleva un protocolo no seguro no es un https, le damos a continuar, seguido de Aceptar.

A continuación accedo al navegador con la siguiente url.

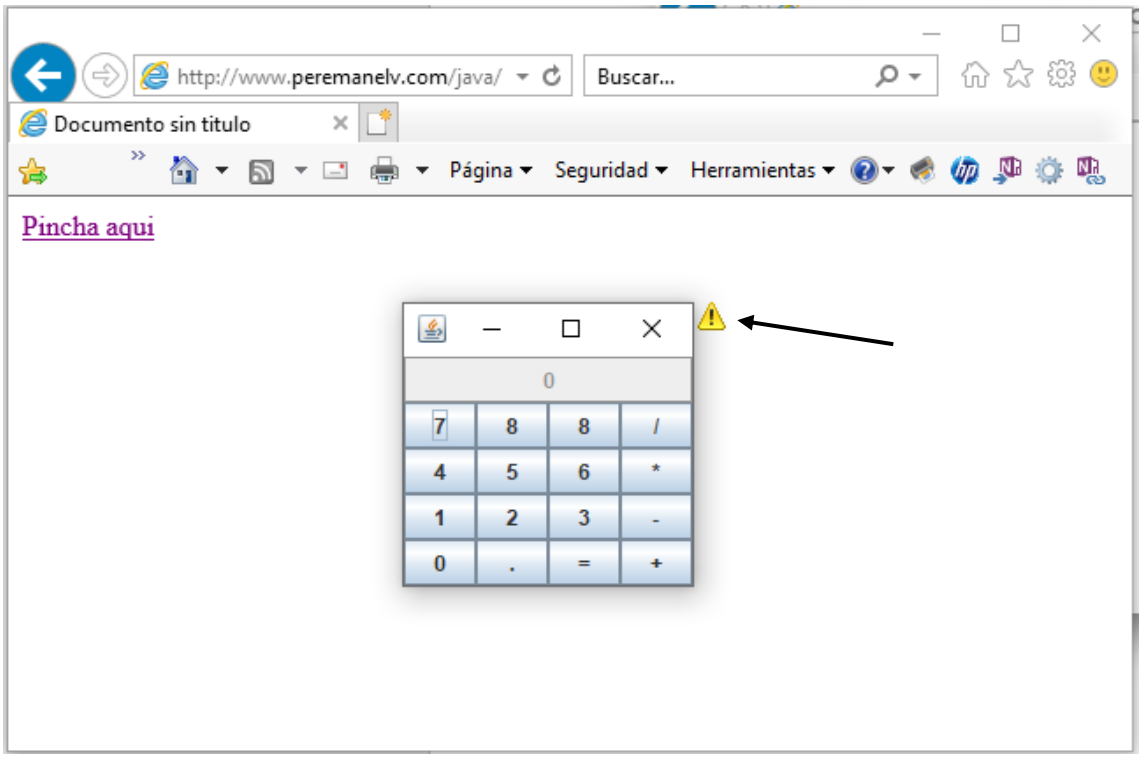
http://www.peremanelv.com/java/lanzando.html



Hacemos clic en el enlace.



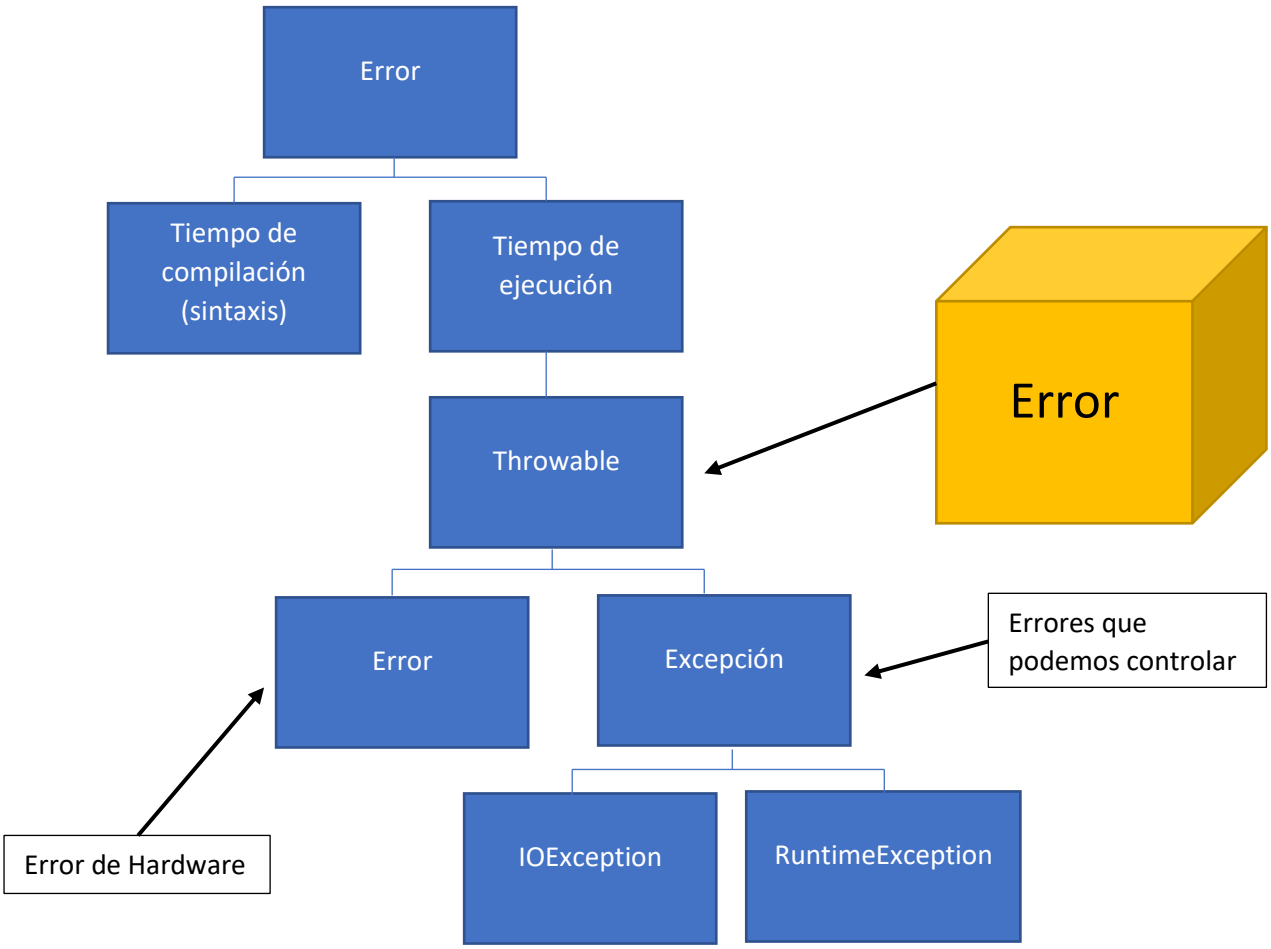
Seleccionamos Permitir.



Nos indica que la aplicación se está ejecutando remotamente.

Excepciones I. (Vídeo 142)

Jerarquía de errores



Errores IOException no son culpa del programador de java.

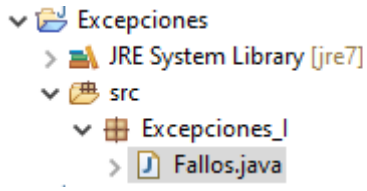
RuntimeException son errores por parte del programador.

Para entender mejor lo explicado vamos a Eclipse.

Vamos a crear un proyecto nuevo llamado Excepciones.

Seleccionamos Excepciones y creamos un paquete llamado Excepciones_I.

Seleccionamos Excepciones_I y creamos una clase llamada Fallos.



Escribimos el siguiente código:

```
1  package Excepciones_I;
2  import javax.swing.*;
3
4  public class Fallos {
5
6      public static void main(String[] args) {
7          // TODO Auto-generated method stub
8          int[] mi_matriz=new int[5];
9
10         mi_matriz[0]=5;
11         mi_matriz[1]=38;
12         mi_matriz[2]=-15;
13         mi_matriz[3]=92;
14         mi_matriz[4]=71;
15
16         for(int i=0;i<5;i++) {
17             System.out.println("Posición " + i + " = " + mi_matriz[i]);
18         }
19
20         //Petición de datos personales
21         String nombre=JOptionPane.showInputDialog("Introduce tu nombre");
22         int edad=Integer.parseInt(JOptionPane.showInputDialog("Introduce tu edad"));
23         System.out.println("Hola " + nombre + " tienes " + edad + " años." +
24         "El programa terminó su ejecución");
25
26     }
27
28 }
```

Los errores de sintaxis, me dejo un punto y coma por ejemplo no nos va a dejar compilar el proyecto y hasta que no lo reparemos el programa no funcionará.

```
mi_matriz[0]=5;
mi_matriz[1]=38;
mi_matriz[2]=-15;
mi_matriz[3]=92;
mi_matriz[4]=71;
mi_matriz[5]=81;
```

Supón que agregamos un valor en la array y la array está definida para 5 valores, si ejecutamos que pasa.

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
        at Excepciones_I.Fallos.main(Fallos.java:15)
```

Que sale un mensaje de Excepción con el tipo de error y no se ejecuta ninguna línea del programa.

Ahora iremos a buscar el proyecto PruebaImagenes donde buscamos una imagen.

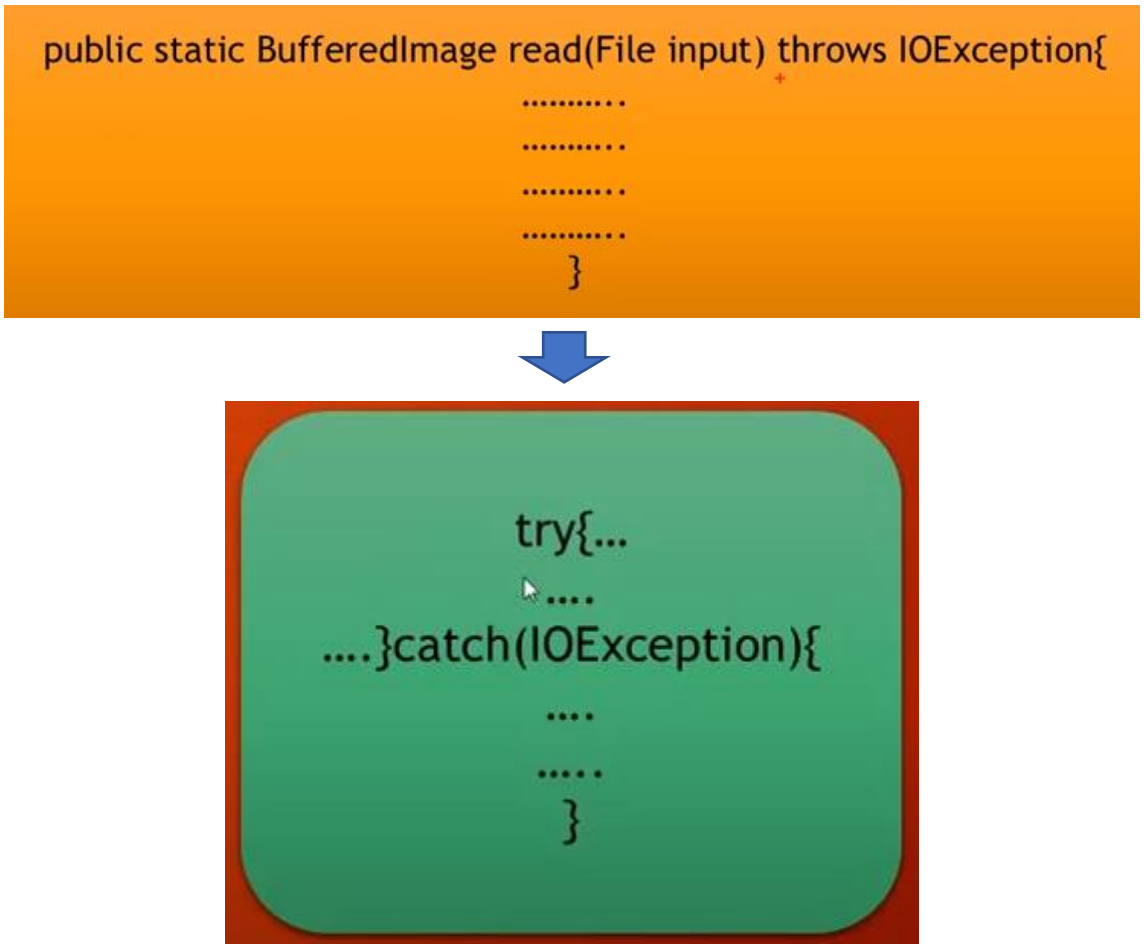
```java
package graficos;
import java.awt.*;
import javax.swing.*;
import javax.imageio.*;
import java.io.*;

public class PruebaImagenes {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MarcoImagen mimarco=new MarcoImagen();
        mimarco.setVisible(true);
        mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
class MarcoImagen extends JFrame{
    public MarcoImagen() {
        setTitle("Marco con Imagen");
        setBounds(700,300,304,200);
        LaminaConImagen milamina=new LaminaConImagen();
        add(milamina);
    }
}
class LaminaConImagen extends JPanel{
    public LaminaConImagen() {
        try {
            imagen=ImageIO.read(new File("src/graficos/bola.gif"));
        }catch(IOException e) {
            System.out.println("Imagen no encontrada.");
        }
    }
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        int anchuraImagen=imagen.getWidth(this);
        int alturaImagen=imagen.getHeight(this);
        g.drawImage(imagen, 0,0,null);
        for(int i=0;i<300; i++) {
            for(int j=0;j<200; j++) {
                if(i+j>0) {
                g.copyArea(0,0,anchuraImagen,alturaImagen,anchuraImagen*i,alturaImagen*j);
                }
            }
        }
    }
    private Image imagen;
}
```

Try captura la siguiente línea y se genera un error de tipo IOException que imprima el mensaje “Imagen no encontrada”, y el programa no se cae.


CURSO JAVA
142
APPLETS Y APLICACIONES
Manejo de errores. Excepciones I
eclipse
Java

Excepciones II. Throws try carch. (Vídeo 143)

Cláusula throws. Lanzamiento de excepciones.



En este ejemplo estamos intentando leer una imagen desde nuestro ordenador, en este caso la instrucción ImageIO.read nos obliga a controlas las excepciones el error que puede generar el no encontrar la imagen en la ubicación especificada.

El método throws IOException lanza una excepción que nos obliga a tratarla para capturarla y manipularla.

Vamos a recordar el siguiente código:

```
package graficos;
import java.awt.*;
import javax.swing.*;
import javax.imageio.*;
import java.io.*;

public class PruebaImagenes {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		MarcoImagen mimarco=new MarcoImagen();
		mimarco.setVisible(true);
		mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}
```

```
}
class MarcoImagen extends JFrame{
	public MarcoImagen() {
		setTitle("Marco con Imagen");
		setBounds(700,300,304,200);
		LaminaConImagen milamina=new LaminaConImagen();
		add(milamina);
	}
}

class LaminaConImagen extends JPanel{
	public LaminaConImagen() {
		try { // Intentar.
			imagen=ImageIO.read(new File("src/graficos/bola.gif"));
		}catch(IOException e) {
			System.out.println("La imagen no se encuentra");
		}
	}
	public void paintComponent(Graphics g) {
		super.paintComponent(g);
		int anchuraImagen=imagen.getWidth(this);
		int alturaImagen=imagen.getHeight(this);
		g.drawImage(imagen, 0,0,null);
		for(int i=0;i<300; i++) {
			for(int j=0;j<200; j++) {
				if(i+j>0) {

	g.copyArea(0,0,anchuraImagen,alturaImagen,anchuraImagen*i,alturaImagen
*j);
				}
			}
		}
	}
	private Image imagen;
}
```

```
24 class LaminaConImagen extends JPanel{
25     public LaminaConImagen() {
26             imagen=ImageIO.read(new File("src/graficos/bola.gif"));
27
28     }
```

Si eliminamos el try y el catch esto nos genera un error.

```
24 class LaminaConImagen extends JPanel{
25     public LaminaConImagen() {
26
27         try { // Intentar.
28             imagen=ImageIO.read(new File("src/graficos/bola.gif"));
29         }catch(IOException e) {
30             System.out.println("La imagen no se encuentra");
31     }
32 }
```

Podrás observar que el error ha desaparecido, le estamos diciendo try (intenta) capturar la imagen especificada, si esto genera un error porque la imagen no está en dicha ubicación pues

catch (caputra) IOException (la excepción) y en este caso me imprimes por consola un mensaje, de este modo el programa no se interrumpe y sigue con su ejecución.

Esto se llama una excepción comprobada.

Sigue la ejecución pero está mostrando errores en consola, es debido a que el programa necesita almacenar unos valores de la imagen capturada como esta no ha sido capturada observamos los siguientes mensajes en consola.

```
        at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:715)
        at java.base/java.security.AccessController.doPrivileged(AccessController.java:391)
        at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:85)
        at java.desktop/java.awt.EventQueue.dispatchEvent(EventQueue.java:740)
        at java.desktop/java.awt.EventDispatchThread.pumpOneEventForFilters(EventDispatchThread.java:203)
        at java.desktop/java.awt.EventDispatchThread.pumpEventsForFilter(EventDispatchThread.java:124)
        at java.desktop/java.awt.EventDispatchThread.pumpEventsForHierarchy(EventDispatchThread.java:113)
        at java.desktop/java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:109)
        at java.desktop/java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:101)
        at java.desktop/java.awt.EventDispatchThread.run(EventDispatchThread.java:90)
Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException
```

En las excepciones no comprobadas no estamos obligados a utilizar try/catch.

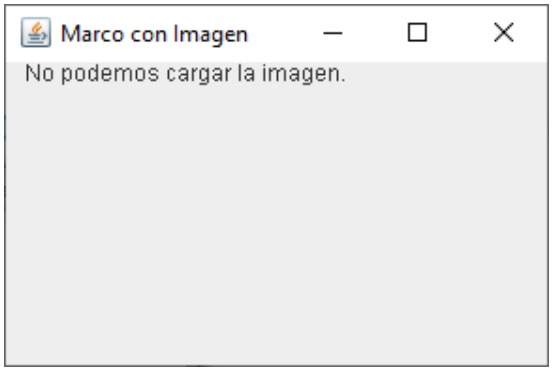
```
public void paintComponent(Graphics g) {
            super.paintComponent(g);
            if(imagen==null) {
                  g.drawString("No podemos cargar la imagen.", 10, 10);
            }else {
            int anchuraImagen=imagen.getWidth(this);
            int alturaImagen=imagen.getHeight(this);
            g.drawImage(imagen, 0,0,null);
            for(int i=0;i<300; i++) {
                  for(int j=0;j<200; j++) {
                        if(i+j>0) {

      g.copyArea(0,0,anchuraImagen,alturaImagen,anchuraImagen*i,alturaImagen
*j);
                        }
                  }
            }
      }
}
```

En el método paintComponent si controlamos que la imagen es igual a null que nos muestre un mensaje de lo contrario si ha encontrado la imagen que siga con la ejecución normal.

En este caso vamos a ejecutar la aplicación.

```
La imagen no se encuentra
```

Observamos el siguiente mensaje en consola.



Nos muestra el marco vacío, pero no vemos mensajes de error en la consola.

Si ahora escribimos la ruta y el archivo correctamente y ejecutamos.

Muestra el marco con su imagen.

Este programa daba dos tipos de errores una de excepción comprobado y otro error de excepción no comprobado.

```
24 class LaminaConImagen extends JPanel{
25     public LaminaConImagen() {
26
27         try { // Intentar.
28             imagen=ImageIO.read(new File("src/graficos/bola.gif"));
29         }catch(IOException e) {
30             System.out.println("La imagen no se encuentra");
31     }
32 }
33     public void paintComponent(Graphics g) {
34         super.paintComponent(g);
35         if(imagen==null) {
36             g.drawString("No podemos cargar la imagen.", 10, 10);
37         }else {
38         int anchuraImagen=imagen.getWidth(this);
39         int alturaImagen=imagen.getHeight(this);
40         g.drawImage(imagen, 0,0,null);
41         for(int i=0;i<300; i++) {
42             for(int j=0;j<200; j++) {
43                 if(i+j>0) {
44                 g.copyArea(0,0,anchuraImagen,alturaImagen,anchuraImagen*i,alturaImagen*j);
45                 }
46             }
47         }
48     }
49 }
50     private Image imagen;
51 }
```

Comprobado

No comprobado


CURSO JAVA
143
EXCEPCIONES II
Excepciones comprobadas y no comprobadas.
Lanzamiento de excepciones (throws/try/catch)
eclipse
Java

Excepciones III. Throws try catch. (Vídeo 144)

Excepciones ▸ src ▸ Excepciones_I ▸ Entrada_datos ▸ main(String[]) : void

```java
package Excepciones_I;

import java.util.*;

public class Entrada_datos {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("¿Qué deseas hacer?");
        System.out.println("1.- Introducir datos");
        System.out.println("2.- Salir del programa ");

        Scanner entrada=new Scanner (System.in);
        int decision=entrada.nextInt();
        if(decision==1) {
            pedirDatos();
        }else {
            System.out.println("Adios");
            System.exit(0);
        }
        entrada.close();
    }
    static void pedirDatos() {
        Scanner entrada=new Scanner(System.in);
        System.out.println("Introduce tu nombre, por favor");
        String nombre_usuario=entrada.nextLine();
        System.out.println("Introduce edad, por favor");
        int edad=entrada.nextInt();
        System.out.println("Hola " + nombre_usuario +
                " El año que viene tendrás " + (edad+1) + " años");
        entrada.close();
        System.out.println("Hemos terminado");
    }
}
```

Con este programa lo que pretendemos es que salga por consola un menú con dos opciones, si contestamos por la opción 2 nos dice Adios y termina el programa.

```
¿Qué deseas hacer?
1.- Introducir datos
2.- Salir del programa
2
Adios
```

En el caso que elijamos la opción 1.

```
¿Qué deseas hacer?
1.- Introducir datos
2.- Salir del programa
1
Introduce tu nombre, por favor
Pere Manel
Introduce edad, por favor
60
Hola Pere Manel El año que viene tendrás 61 años
Hemos terminado
```

Este será el resultado.

Supongamos que a la hora de ejecutar el programa elegimos la opción 1 y a la hora de introducir la edad la ponemos en texto, esto nos genera un error.

```
¿Qué deseas hacer?
1.- Introducir datos
2.- Salir del programa
1
Introduce tu nombre, por favor
Pere Manel
Introduce edad, por favor
sesenta
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Unknown Source)
        at java.util.Scanner.next(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at Excepciones_I.Entrada_datos.pedirDatos(Entrada_datos.java:28)
        at Excepciones_I.Entrada_datos.main(Entrada_datos.java:16)
```

Es un error no comprobado, no estoy obligado a utilizar try-catch, esto no quiere decir que no pueda utilizar.

```
1  package Excepciones_I;
2
3  import java.util.*;
4
5  public class Entrada_datos {
6
7      public static void main(String[] args) {
8          // TODO Auto-generated method stub
9          System.out.println("¿Qué deseas hacer?");
10         System.out.println("1.- Introducir datos");
11         System.out.println("2.- Salir del programa ");
12
13         Scanner entrada=new Scanner (System.in);
14         int decision=entrada.nextInt();
15         if(decision==1) {
16             pedirDatos();
17         }else {
18             System.out.println("Adios");
19             System.exit(0);
20         }
21         entrada.close();
22     }
23     static void pedirDatos() throws InputMismatchException {
24         try {
25         Scanner entrada=new Scanner(System.in);
26         System.out.println("Introduce tu nombre, por favor");
27         String nombre_usuario=entrada.nextLine();
28         System.out.println("Introduce edad, por favor");
29         int edad=entrada.nextInt();
30         System.out.println("Hola " + nombre_usuario +
31                 " El año que viene tendrás " + (edad+1) + " años");
32         entrada.close();
33         }catch(InputMismatchException e) {
34             System.out.println("La edad debes introducirla en números");
35         }
```

```
36          System.out.println("Hemos terminado");
37      }
38  }
```

Cuando introducimos un texto que lo tiene que almacenar en una variable de tipo int y en su lugar introducimos un texto esto genera un error de tipo InputMismatchException.

En la línea 23 throws, lanzamos una excepción de tipo InputMismatchException.

En la línea 24 le decimos con try que si se genera dicho error se vaya a la línea 34 después de catch (captura) de este modo el programa no se corta y además no genera ningún tipo de error.

Vamos a ejecutar la aplicación y introduciremos la edad en letras.

```
¿Qué deseas hacer?
1.- Introducir datos
2.- Salir del programa
1
Introduce tu nombre, por favor
Pere Manel
Introduce edad, por favor
sesenta
La edad debes introducirla en números
Hemos terminado
```

En la línea 33 también será correcto:" }catch(Exception e){" es más genérico.

El programa funcionará correctamente.

En el ejemplo siguiente:

```
package graficos;
import java.awt.*;
import javax.swing.*;
import javax.imageio.*;
import java.io.*;

public class PruebaImagenes {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		MarcoImagen mimarco=new MarcoImagen();
		mimarco.setVisible(true);
		mimarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}
}
class MarcoImagen extends JFrame{
	public MarcoImagen() {
		setTitle("Marco con Imagen");
		setBounds(700,300,304,200);
		LaminaConImagen milamina=new LaminaConImagen();
		add(milamina);
	}
}

class LaminaConImagen extends JPanel{
	public LaminaConImagen() {

		try { // Intentar.
```

```
                imagen=ImageIO.read(new File("src/grafico/bola.gif"));
            }catch(IOException e) {
                System.out.println("La imagen no se encuentra");
        }
}
        public void paintComponent(Graphics g) throws NullPointerException {
            super.paintComponent(g);

            try {
            int anchuraImagen=imagen.getWidth(this);
            int alturaImagen=imagen.getHeight(this);
            g.drawImage(imagen, 0,0,null);
            for(int i=0;i<300; i++) {
                for(int j=0;j<200; j++) {
                    if(i+j>0) {

        g.copyArea(0,0,anchuraImagen,alturaImagen,anchuraImagen*i,alturaImagen
*j);
                    }
                }
            }

            }catch(NullPointerException e) {
                g.drawString("No se ha podido cargar la imagen", 10, 10);
            }
        }
        private Image imagen;
}
```

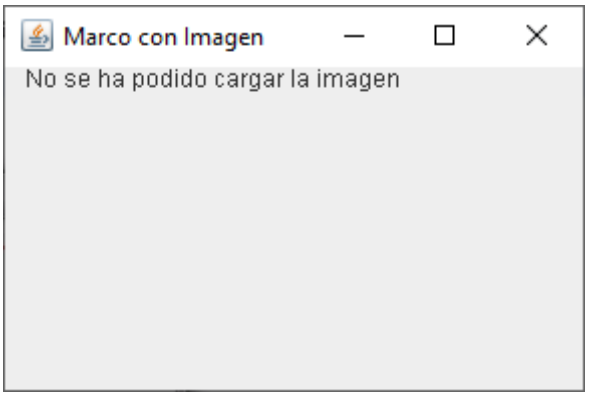
Tipo de error

La dirección donde se encuentra la imagen es errónea.

En este ejemplo introducimos una dirección errónea de donde se encuentra nuestro archivo.

Genera un error NullPoinerRxception que lo ponemos a la escucha con throws.

Si se genera el error le decimos que todas las instrucciones entre el try y el catch las ignore, evitando así mensajes de error en la consola y a continuación emprime en consola “La imagen no se encuentra” y en la venta que se abre el mensaje “No se ha podido cargar la imagen”.




CURSO JAVA
144
EXCEPCIONES III
Excepciones comprobadas y no comprobadas.
Lanzamiento de excepciones (throws/try/catch)
eclipse
Java

Excepciones IV. Throws try catch. (Vídeo 145)

```java
package Excepciones_I;

import java.util.*;

public class Entrada_datos {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("¿Qué deseas hacer?");
        System.out.println("1.- Introducir datos");
        System.out.println("2.- Salir del programa ");

        Scanner entrada=new Scanner (System.in);
        int decision=entrada.nextInt();
        if(decision==1) {
            pedirDatos();
        }else {
            System.out.println("Adios");
            System.exit(0);
        }
        entrada.close();
    }
    static void pedirDatos() throws InputMismatchException {
        try {
        Scanner entrada=new Scanner(System.in);
        System.out.println("Introduce tu nombre, por favor");
        String nombre_usuario=entrada.nextLine();
        System.out.println("Introduce edad, por favor");
        int edad=entrada.nextInt();
        System.out.println("Hola " + nombre_usuario +
                " El año que viene tendrás " + (edad+1) + " años");
        entrada.close();
        }catch(Exception e) {
            System.out.println("La edad debes introducirla en números");
        }
        System.out.println("Hemos terminado");
    }
}

```

Vamos a seguir trabajando en este proyecto.

Realizamos algunas modificaciones que nos parecen más correctas, el código funciona igual.

```java
package Excepciones_I;

import java.util.*;

public class Entrada_datos {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("¿Qué deseas hacer?");
        System.out.println("1.- Introducir datos");
        System.out.println("2.- Salir del programa ");

        Scanner entrada=new Scanner (System.in);
        int decision=entrada.nextInt();
```

```
15          if(decision==1) {
16              try {
17              pedirDatos();
18              }catch(InputMismatchException e) {
19                  System.out.println("La edad debes introducirla en números");
20              }
21          }else {
22              System.out.println("Adios");
23              System.exit(0);
24          }
25          entrada.close();
26      }
27      static void pedirDatos() throws InputMismatchException {
28          Scanner entrada=new Scanner(System.in);
29          System.out.println("Introduce tu nombre, por favor");
30          String nombre_usuario=entrada.nextLine();
31          System.out.println("Introduce edad, por favor");
32          int edad=entrada.nextInt();
33          System.out.println("Hola " + nombre_usuario +
34                  " El año que viene tendrás " + (edad+1) + " años");
35          entrada.close();
36          System.out.println("Hemos terminado");
37      }
38  }
```

Cuando el programa pasa por la línea 17 llamados al método pedirDatos().

Lanzamos una alerta por el correspondiente error, que consiste en introducir un texto alfanumérico cuando tienen que ser numérico.

Si en la línea 32 detecta un error porque espera un valor numérico y se lo damos de tipo String no termina el método y se va a la línea después del catch.

Este será el resultado:

```
¿Qué deseas hacer?
1.- Introducir datos
2.- Salir del programa
1
Introduce tu nombre, por favor
Pere
Introduce edad, por favor
sesenta
La edad debes introducirla en números
```

```
1   package Excepciones_I;
2
3   import java.io.*;
4   import java.util.*;
5
6   public class Entrada_datos {
7
8       public static void main(String[] args) {
9           // TODO Auto-generated method stub
10          System.out.println("¿Qué deseas hacer?");
11          System.out.println("1.- Introducir datos");
12          System.out.println("2.- Salir del programa ");
13
```

```
14          Scanner entrada=new Scanner (System.in);
15          int decision=entrada.nextInt();
16          if(decision==1) {
17              //try {
18              pedirDatos();
19              //}catch(InputMismatchException e) {
20                  //System.out.println("La edad debes introducirla en números");
21              //}
22          }else {
23              System.out.println("Adios");
24              System.exit(0);
25          }
26          entrada.close();
27      }
28      static void pedirDatos() throws IOException {
29          Scanner entrada=new Scanner(System.in);
30          System.out.println("Introduce tu nombre, por favor");
31          String nombre_usuario=entrada.nextLine();
32          System.out.println("Introduce edad, por favor");
33          int edad=entrada.nextInt();
34          System.out.println("Hola " + nombre_usuario +
35                  " El año que viene tendrás " + (edad+1) + " años");
36          entrada.close();
37          System.out.println("Hemos terminado");
38      }
39  }
```

Si la excepción es de tipo comprobadas, observaremos como en la línea 18 donde ejecutamos el método nos señala un error, esto es porque está esperando try/catch.

Es decir lanza una excepción que está sin capturar.

```
1   package Excepciones_I;
2
3   import java.io.*;
4   import java.util.*;
5
6   public class Entrada_datos {
7
8       public static void main(String[] args) {
9           // TODO Auto-generated method stub
10          System.out.println("¿Qué deseas hacer?");
11          System.out.println("1.- Introducir datos");
12          System.out.println("2.- Salir del programa ");
13
14          Scanner entrada=new Scanner (System.in);
15          int decision=entrada.nextInt();
16          if(decision==1) {
17              try {
18              pedirDatos();
19              }catch(IOException e) {
20                  System.out.println("La edad debes introducirla en números");
21              }
22          }else {
23              System.out.println("Adios");
24              System.exit(0);
25          }
26          entrada.close();
27      }
```

```
28      static void pedirDatos() throws IOException {
29          Scanner entrada=new Scanner(System.in);
30          System.out.println("Introduce tu nombre, por favor");
31          String nombre_usuario=entrada.nextLine();
32          System.out.println("Introduce edad, por favor");
33          int edad=entrada.nextInt();
34          System.out.println("Hola " + nombre_usuario +
35                  " El año que viene tendrás " + (edad+1) + " años");
36          entrada.close();
37          System.out.println("Hemos terminado");
38      }
39  }
```

Tanto en la línea 19 como la línea 20 tiene que ser la misma excepción, la línea 28 la lanza y la línea 19 la captura.

Este tipo de excepción obliga forzosamente a utilizar try / catch, como podréis observar ahora en el código no hay ningún tipo de error, cosa que en el ejemplo anterior si lo marcaba.

Pero veremos que el programa no funciona correctamente, lo vamos a ejecutar y cometeremos el error de introducir texto en lugar de un número en años.

```
¿Qué deseas hacer?
1.- Introducir datos
2.- Salir del programa
1
Introduce tu nombre, por favor
Pere
Introduce edad, por favor
sesenta
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Unknown Source)
        at java.util.Scanner.next(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at Excepciones_I.Entrada_datos.pedirDatos(Entrada_datos.java:33)
        at Excepciones_I.Entrada_datos.main(Entrada_datos.java:18)
```

```
1  package Excepciones_I;
2
3  import java.io.*;
4  import java.util.*;
5
6  public class Entrada_datos {
7
8      public static void main(String[] args) {
9          // TODO Auto-generated method stub
10         System.out.println("¿Qué deseas hacer?");
11         System.out.println("1.- Introducir datos");
12         System.out.println("2.- Salir del programa ");
13
14         Scanner entrada=new Scanner (System.in);
15         int decision=entrada.nextInt();
16         if(decision==1) {
17             try {
18             pedirDatos();
19             }catch(Exception e) {
20                 System.out.println("La edad debes introducirla en números");
21             }
```

```
22          }else {
23              System.out.println("Adios");
24              System.exit(0);
25          }
26          entrada.close();
27      }
28      static void pedirDatos() throws IOException {
29          Scanner entrada=new Scanner(System.in);
30          System.out.println("Introduce tu nombre, por favor");
31          String nombre_usuario=entrada.nextLine();
32          System.out.println("Introduce edad, por favor");
33          int edad=entrada.nextInt();
34          System.out.println("Hola " + nombre_usuario +
35                  " El año que viene tendrás " + (edad+1) + " años");
36          entrada.close();
37          System.out.println("Hemos terminado");
38      }
39  }
```

En la línea 19 cambias la excepción IOException por Exception ya que la primera hereda de la segunda.

Ahora vamos a ejecutar de nuevo.

```
¿Qué deseas hacer?
1.- Introducir datos
2.- Salir del programa
1
Introduce tu nombre, por favor
Pere
Introduce edad, por favor
Sesenta
La edad debes introducirla en números
```

Ya no observamos los mensajes de error y la ejecución del programa no se interrumpe.


CURSO JAVA
145
EXCEPCIONES IV
Excepciones comprobadas y no comprobadas.
Lanzamiento de excepciones (throws/try/catch)
eclipse
Java

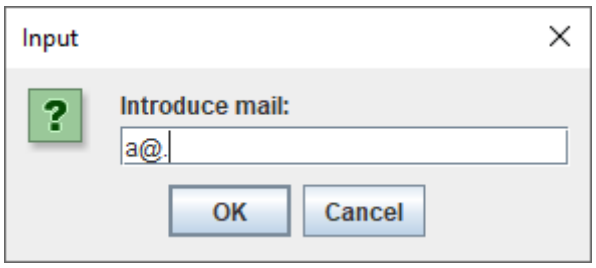
Excepciones V. Cláusula throw. (Vídeo 146)

Vamos a realizar la siguiente clase llamada Comprueba_mail.java en el paquete de Excepciones.

```
1  package Excepciones_I;
2  import javax.swing.*;
3  public class Comprueba_mail {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          String el_mail=JOptionPane.showInputDialog("Introduce mail");
8          examina_mail(el_mail);
9      }
10
11     static void examina_mail(String mail) {
12         int arroba=0;
13         boolean punto=false;
14         for(int i=0;i<mail.length();i++) {
15             if(mail.charAt(i)=='@') {
16                 arroba++;
17             }
18             if(mail.charAt(i)=='.') {
19                 punto=true;
20             }
21         }
22         if(arroba==1 && punto==true) {
23             System.out.println("Es correcto");
24         }
25         else {
26             System.out.println("No es correcto");
27         }
28     }
29 }
```

Este programa comprueba que nuestro correo electrónico tiene una @ y como mínimo un punto.

Si ejecutamos la aplicación



Ponemos este nombre de correo electrónico el programa nos da la siguiente información por consola.

```
Es correcto
```

Queremos lanzar una excepción si el nombre de correo tiene 3 o menos de 3 caracteres.

```java
package Excepciones_I;
import javax.swing.*;
public class Comprueba_mail {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String el_mail=JOptionPane.showInputDialog("Introduce mail");
        examina_mail(el_mail);
    }

    static void examina_mail(String mail) {
        int arroba=0;
        boolean punto=false;
        if(mail.length()<=3) {
            ArrayIndexOutOfBoundsException mi_excepcion=new ArrayIndexOutOfBoundsException();
            throw mi_excepcion;
        }
        else {
            for(int i=0;i<mail.length();i++) {
                if(mail.charAt(i)=='@') {
                    arroba++;
                }
                if(mail.charAt(i)=='.') {
                    punto=true;
                }
            }
            if(arroba==1 && punto==true) {
                System.out.println("Es correcto");
            }
            else {
                System.out.println("No es correcto");
            }
        }
    }
}
```

Ampliando lo seleccionado.

```java
ArrayIndexOutOfBoundsException mi_excepcion=new
ArrayIndexOutOfBoundsException();
                    throw mi_excepcion;
```

En el caso que introduzcamos un mail menos o igual a 3 caracteres aun que cumplan las condiciones de una @ y al menos un punto, lanzaremos un error del tipo especificado.

Vamos a ejecutar la aplicación.

Esto observaremos por consola:

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException
        at Excepciones_I.Comprueba_mail.examina_mail(Comprueba_mail.java:15)
        at Excepciones_I.Comprueba_mail.main(Comprueba_mail.java:8)
```

Y la ejecución del programa de detendrá.

Lo que no tiene demasiado sentido porque no se corresponde la excepción lanzada con el tipo de error que ha ocurrido, aprenderemos a solucionarlo más adelante.

Vamos a simplificarlo.

```
//ArrayIndexOutOfBoundsException mi_excepcion=new ArrayIndexOutOfBoundsException();
//throw mi_excepcion;

throw new ArrayIndexOutOfBoundsException();
```

Las dos líneas pasadas a comentario se simplifica con la única línea que viene a continuación.

Ahora a este código le faltaría capturar esta excepción para el caso que ocurra decirle lo que tiene que hacer.

```
11    static void examina_mail(String mail) throws ArrayIndexOutOfBoundsException{
12        int arroba=0;
13        boolean punto=false;
14        if(mail.length()<=3) {
15            //ArrayIndexOutOfBoundsException mi_excepcion=new ArrayIndexOutOfBoundsException();
16            //throw mi_excepcion;
17
18            throw new ArrayIndexOutOfBoundsException();
```

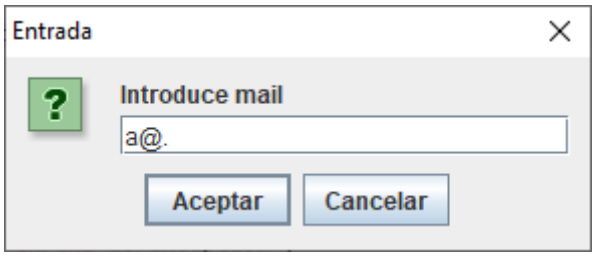
Si al método le agregamos lo que está marcado en rojo estamos diciendo que dicho método lanza una excepción y así lo tengan en cuenta.

Vamos a modificar el código con una excepción comprobada.

```
package Excepciones_I;
import java.io.EOFException;

import javax.swing.*;
public class Comprueba_mail {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		String el_mail=JOptionPane.showInputDialog("Introduce mail");
		try {
		examina_mail(el_mail);
		}catch(EOFException e) {
			System.out.println("La dirección de email no es
correcta.");
		}
	}

	static void examina_mail(String mail) throws EOFException{
		int arroba=0;
		boolean punto=false;
		if(mail.length()<=3) {

			throw new EOFException();
		}
		else {
			for(int i=0;i<mail.length();i++) {
				if(mail.charAt(i)=='@') {
					arroba++;
				}
				if(mail.charAt(i)=='.') {
					punto=true;
				}
			}
			if(arroba==1 && punto==true) {
				System.out.println("Es correcto");
```

```
                        }
                        else {
                                System.out.println("No es correcto");
                        }
                }
        }
}
```

Si ejecutamos y contestamos



Ahora este será el resultado en consola:

```
La dirección de email no es correcta.
```

Excepciones VI. Creación de excepciones propias. (Vídeo 147)

```java
package Excepciones_I;
import java.io.EOFException;
import javax.swing.*;
public class Comprueba_mail {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String el_mail=JOptionPane.showInputDialog("Introduce mail");
        try {
        examina_mail(el_mail);
        }catch(EOFException e) {
            System.out.println("La dirección de email no es correcta.");
        }
    }
    static void examina_mail(String mail) throws EOFException{
        int arroba=0;
        boolean punto=false;
        if(mail.length()<=3) {

            throw new EOFException();
        }
        else {
            for(int i=0;i<mail.length();i++) {
                if(mail.charAt(i)=='@') {
                    arroba++;
                }
                if(mail.charAt(i)=='.') {
                    punto=true;
                }
            }
            if(arroba==1 && punto==true) {
                System.out.println("Es correcto");
            }
            else {
                System.out.println("No es correcto");
            }
        }
    }
}
```

Seguimos con el mismo proyecto del capítulo anterior.

Vamos a crear nuestras propias excepciones, que se adapten a nuestras necesidades.

```java
package Excepciones_I;
import java.io.EOFException;
import javax.swing.*;
public class Comprueba_mail {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String el_mail=JOptionPane.showInputDialog("Introduce mail");
        examina_mail(el_mail);
    }
    static void examina_mail(String mail) throws Longitud_mail_erronea{
        int arroba=0;
        boolean punto=false;
        if(mail.length()<=3) {
```

Lanzamos el método de excepción.

Comparamos que el mail sea igual o menos a 3 caracteres.

```
                        throw new Longitud_mail_erronea("el mail es demasiado
corto");
                }
                else {
                        for(int i=0;i<mail.length();i++) {
                                if(mail.charAt(i)=='@') {
                                        arroba++;
                                }
                                if(mail.charAt(i)=='.') {
                                        punto=true;
                                }
                        }
                        if(arroba==1 && punto==true) {
                                System.out.println("Es correcto");
                        }
                        else {
                                System.out.println("No es correcto");
                        }
                }
        }
}
```

Si es así que salga el siguiente error cuando introduzcamos un mail de 3 caracteres o menos.

```
class Longitud_mail_erronea extends RuntimeException{
        public Longitud_mail_erronea() {}
        public Longitud_mail_erronea(String msj_error) {
                super(msj_error);
        }
}
```

Creamos una clase de excepción que hereda de RuntimeException, esta no necesita de try / catch.

Ejecutamos el programa.

El mensaje de error que introdujimos en throw.

Le damos a aceptar.

Observaremos el siguiente mensaje en consola:

```
Exception in thread "main" Excepciones_I.Longitud_mail_erronea: el mail es demasiado corto
        at Excepciones_I.Comprueba_mail.examina_mail(Comprueba_mail.java:16)
        at Excepciones_I.Comprueba_mail.main(Comprueba_mail.java:9)
```

Otra forma de plantear el código anterior.

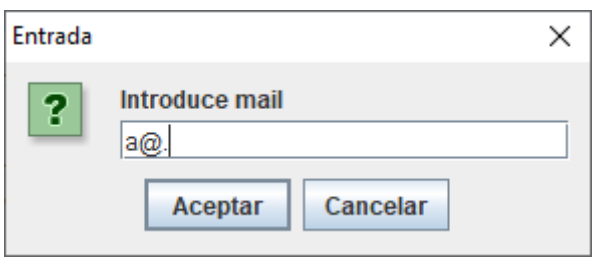
```
package Excepciones_I;
import java.io.EOFException;
import javax.swing.*;
public class Comprueba_mail {

        public static void main(String[] args) {
```

```
		// TODO Auto-generated method stub
		String el_mail=JOptionPane.showInputDialog("Introduce mail");

		try {
		examina_mail(el_mail);
		}catch(Exception e) {
			System.out.println("La dirección de email no es
correcta.");
		}
	}
	static void examina_mail(String mail) throws Longitud_mail_erronea{
		int arroba=0;
		boolean punto=false;
		if(mail.length()<=3) {

			throw new Longitud_mail_erronea();
		}
		else {
			for(int i=0;i<mail.length();i++) {
				if(mail.charAt(i)=='@') {
					arroba++;
				}
				if(mail.charAt(i)=='.') {
					punto=true;
				}
			}
			if(arroba==1 && punto==true) {
				System.out.println("Es correcto");
			}
			else {
				System.out.println("No es correcto");
			}
		}
	}
}

class Longitud_mail_erronea extends Exception{
	public Longitud_mail_erronea() {}
	public Longitud_mail_erronea(String msj_error) {
		super(msj_error);
	}
}
```

En este ejemplo el método de excepción hereda de Exception, esto nos obliga a utilizar try / catch para que el programa no de error.

En el código lo verás marcado con flechas azules.

Vamos a ejecutar.



Este será el mensaje que aparecerá en consola:

```
La dirección de email no es correcta.
```

La clase Longitud_mail_erronea tiene dos constructores, uno sin parámetros y otro con un parámetro de tipo String, puedes utilizar el que más te convenga.

```
1  package Excepciones_I;
2  import java.io.EOFException;
3  import javax.swing.*;
4  public class Comprueba_mail {
5
6      public static void main(String[] args) {
7          // TODO Auto-generated method stub
8          String el_mail=JOptionPane.showInputDialog("Introduce mail");
9
10         try {
11         examina_mail(el_mail);
12         }catch(Exception e) {
13             System.out.println("La dirección de email no es correcta.");
14             e.printStackTrace();
15         }
16     }
17     static void examina_mail(String mail) throws Longitud_mail_erronea{
18         int arroba=0;
19         boolean punto=false;
20         if(mail.length()<=3) {
21
22             throw new Longitud_mail_erronea();
23         }
24         else {
25             for(int i=0;i<mail.length();i++) {
26                 if(mail.charAt(i)=='@') {
27                     arroba++;
28                 }
29                 if(mail.charAt(i)=='.') {
30                     punto=true;
31                 }
32             }
33             if(arroba==1 && punto==true) {
34                 System.out.println("Es correcto");
35             }
36             else {
37                 System.out.println("No es correcto");
38             }
39         }
40     }
41 }
42
43 class Longitud_mail_erronea extends Exception{
44     public Longitud_mail_erronea() {}
45     public Longitud_mail_erronea(String msj_error) {
46         super(msj_error);
47     }
48 }
```

Si agregamos después de imprimir en consola la instrucción e.printStackTrace(); conseguimos que además de imprimir el mensaje en consola “La dirección de email no es correcta.” Nos dará información del error.

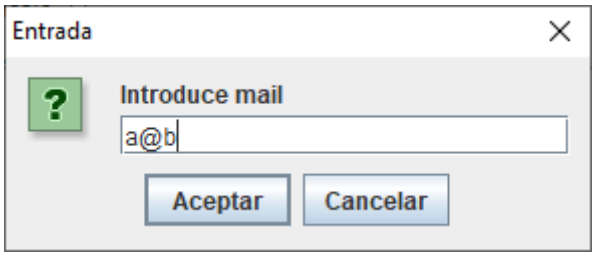
Vamos a ejecutar y contestaremos con 3 caracteres, para provocar el error.

Esta será la información en consola:

```
La dirección de email no es correcta.
Excepciones_I.Longitud_mail_erronea
        at Excepciones_I.Comprueba_mail.examina_mail(Comprueba_mail.java:22)
        at Excepciones_I.Comprueba_mail.main(Comprueba_mail.java:11)
```

```
17      static void examina_mail(String mail) throws Longitud_mail_erronea{
18          int arroba=0;
19          boolean punto=false;
20          if(mail.length()<=3) {
21
22              throw new Longitud_mail_erronea("El mail no puede tener menos de tres caracteres");
23          }
24          else {
25              for(int i=0;i<mail.length();i++) {
26                  if(mail.charAt(i)=='@') {
27                      arroba++;
28                  }
29                  if(mail.charAt(i)=='.') {
30                      punto=true;
31                  }
32              }
33              if(arroba==1 && punto==true) {
34                  System.out.println("Es correcto");
35              }
36              else {
37                  System.out.println("No es correcto");
38              }
39          }
40      }
41  }
```

Si en el throw agregamos un comentario como marca la flecha, esto a otro programador le puede ayudar, ahora vamos a ejecutar el programa contestando con 3 caracteres para provocar el error.



Este será el resultado en consola:

Aq

```
La dirección de email no es correcta.
Excepciones_I.Longitud_mail_erronea: El mail no puede tener menos de tres caracteres
        at Excepciones_I.Comprueba_mail.examina_mail(Comprueba_mail.java:22)
        at Excepciones_I.Comprueba_mail.main(Comprueba_mail.java:11)
```

Aquí nos muestra el mensaje que introdujimos en el throw.

Excepciones VII. Captura de varias excepciones. (Vídeo 148)

```java
package Excepciones_I;
import javax.swing.*;
public class Varias_excepciones {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    division();

}
    static void division() {
        int num1=Integer.parseInt(JOptionPane.showInputDialog("Introduce el dividendo"));
        int num2=Integer.parseInt(JOptionPane.showInputDialog("Introduce el divisor"));
        System.out.println("El resultado es: " + num1/num2);
    }
}
```

Partiendo del siguiente ejemplo.

Si como dividendo ponemos un 9 y como divisor 0 veremos qué pasa una excepción.

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Excepciones_I.Varias_excepciones.division(Varias_excepciones.java:14)
        at Excepciones_I.Varias_excepciones.main(Varias_excepciones.java:8)
```

Si en lugar de números ponemos textos:

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "pepe"
        at java.lang.NumberFormatException.forInputString(Unknown Source)
        at java.lang.Integer.parseInt(Unknown Source)
        at java.lang.Integer.parseInt(Unknown Source)
        at Excepciones_I.Varias_excepciones.division(Varias_excepciones.java:12)
        at Excepciones_I.Varias_excepciones.main(Varias_excepciones.java:8)
```

Son excepciones no controladas.

```java
package Excepciones_I;
import javax.swing.*;
public class Varias_excepciones {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
    try {
    division();
    }catch(Exception e) {
        System.out.println("Ha ocurrido un error");
    }
}
    static void division() {
        int num1=Integer.parseInt(JOptionPane.showInputDialog("Introduce el dividendo"));
        int num2=Integer.parseInt(JOptionPane.showInputDialog("Introduce el divisor"));
        System.out.println("El resultado es: " + num1/num2);
    }
}
```

Si ahora ejecutamos la aplicación dividimos por 0 o introducimos texto este será el mensaje por consola.

```
Ha ocurrido un error
```

```java
package Excepciones_I;
import javax.swing.*;
public class Varias_excepciones {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
    try {
    division();
    }catch(ArithmeticException e) {
        System.out.println("Estás dividiendo por 0");
    }catch(NumberFormatException e) {
        System.out.println("No has introducido un número entero");
    }
}
    static void division() {
        int num1=Integer.parseInt(JOptionPane.showInputDialog("Introduce el dividendo"));
        int num2=Integer.parseInt(JOptionPane.showInputDialog("Introduce el divisor"));
        System.out.println("El resultado es: " + num1/num2);
    }
}
```

Ahora tenemos varias excepciones, la primera si intentamos dividir por 0 y la segunda si ponemos texto en lugar de un entero.

Dividiendo por 0 este será el resultado:

```
Estás dividiendo por 0
```

Poner un texto en lugar de un número entero:

```
No has introducido un número entero
```

Para informarnos de los tipos de errores podemos utilizar los métodos:

getMessage()

getClass()

getName()

```java
package Excepciones_I;
import javax.swing.*;
public class Varias_excepciones {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
    try {
    division();
    }catch(ArithmeticException e) {
        System.out.println("Estás dividiendo por 0");
    }catch(NumberFormatException e) {
        System.out.println("No has introducido un número entero");
        System.out.println(e.getMessage());
        System.out.println("Se ha generado un error de este tipo: " + e.getClass().getName());
    }
}
    static void division() {
        int num1=Integer.parseInt(JOptionPane.showInputDialog("Introduce el dividendo"));
        int num2=Integer.parseInt(JOptionPane.showInputDialog("Introduce el divisor"));
        System.out.println("El resultado es: " + num1/num2);
    }
}
```

Si ejecutamos e introducimos un texto por error.

```
No has introducido un número entero
For input string: "saas"
Se ha generado un error de este tipo: java.lang.NumberFormatException
```


CURSO JAVA
148
EXCEPCIONES VII
Captura de varias excepciones
eclipse
Java

Excepciones VIII. Cláusula finally. (Vídeo 149)

```java
package Excepciones_I;
import java.util.*;
import javax.swing.JOptionPane;
public class Areas_Peso {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner entrada=new Scanner(System.in);
        System.out.println("elige una opción: \n1: Cuadrado \n2: Rectángulo \n3: "
                + "Triángulo \n4: Circulo");
        figura=entrada.nextInt();
        switch (figura) {
        case 1:
            int lado=Integer.parseInt(JOptionPane.showInputDialog("Introduce el lado"));
            System.out.println(Math.pow(lado, 2));
            break;
        case 2:
            int base=Integer.parseInt(JOptionPane.showInputDialog("Introduce la base"));
            int altura=Integer.parseInt(JOptionPane.showInputDialog("Introduce la altura"));
            System.out.println("El area del rectángulo es " + base * altura);
            break;
        case 3:
            base=Integer.parseInt(JOptionPane.showInputDialog("Introduce la base"));
            altura=Integer.parseInt(JOptionPane.showInputDialog("Introduce la altura"));
            System.out.println("El area del triángulo es " + (base * altura)/2);
            break;
        case 4:
            int radio=Integer.parseInt(JOptionPane.showInputDialog("Introduce el radio"));
            System.out.print("El área del círulo es ");
            System.out.println(Math.PI*(Math.pow(radio, 2)));
            break;
        default:
            System.out.println("La opción no es correcta");
        }
        int altura=Integer.parseInt(JOptionPane.showInputDialog("Dime tu altura en cm."));
        System.out.println("Si eres hombre tu peso ideal es: " + (altura-110)+ " kg.");
        System.out.println("Si eres mujer tu peso ideal es: " + (altura-120)+ " kg.");
        }
    static int figura;
}
```

Vamos a partir de este proyecto.

```java
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner entrada=new Scanner(System.in);
        System.out.println("elige una opción: \n1: Cuadrado \n2: Rectángulo \n3: "
                + "Triángulo \n4: Circulo");
        figura=entrada.nextInt();
```

En esta línea nos aparece una advertencia diciendo que estamos abriendo una línea con la consola, para poder introducir datos desde ella y que no la cerramos, esto hace que la aplicación coma recursos innecesarios.

```java
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner entrada=new Scanner(System.in);
        System.out.println("elige una opción: \n1: Cuadrado \n2: Rectángulo \n3: "
                + "Triángulo \n4: Circulo");
        figura=entrada.nextInt();
        entrada.close();
        switch (figura) {
```

Cuando el valor de entrada se lo hemos pasado a la variable figura ya podemos cerrar entrada, observarás que en la línea 8 ha desaparecido el error.

Que ocurre si cuando ejecutamos la aplicación en lugar de poner una opción del 1 al 4 ponemos un texto.

```
elige una opción:
1: Cuadrado
2: Rectángulo
3: Triángulo
4: Circulo
uno
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Unknown Source)
        at java.util.Scanner.next(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at Excepciones_I.Areas_Peso.main(Areas_Peso.java:11)
```

El programa se cae.

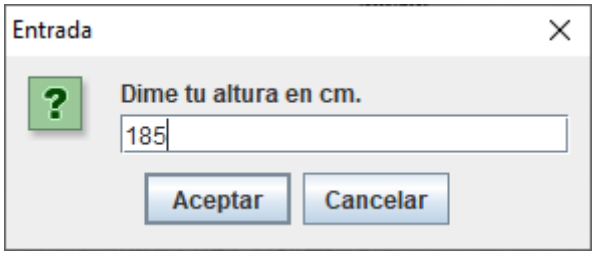
El problema es que la segunda parte del programa donde te pide la altura se pudiera ejecutar.

```
1  package Excepciones_I;
2  import java.util.*;
3  import javax.swing.JOptionPane;
4  public class Areas_Peso {
5
6      public static void main(String[] args) {
7          // TODO Auto-generated method stub
8          Scanner entrada=new Scanner(System.in);
9          System.out.println("elige una opción: \n1: Cuadrado \n2: Rectángulo \n3: "
10                 + "Triángulo \n4: Circulo");
11         try {
12         figura=entrada.nextInt();
13
14         }catch(Exception e) {
15             System.out.println("Ha ocurrido un error");
16         }finally{
17             entrada.close();
18         }
19         switch (figura) {
20         case 1:
21             int lado=Integer.parseInt(JOptionPane.showInputDialog("Introduce el lado"));
22             System.out.println(Math.pow(lado, 2));
23             break;
24         case 2:
25             int base=Integer.parseInt(JOptionPane.showInputDialog("Introduce la base"));
26             int altura=Integer.parseInt(JOptionPane.showInputDialog("Introduce la altura"));
27             System.out.println("El area del rectángulo es " + base * altura);
28             break;
29         case 3:
30             base=Integer.parseInt(JOptionPane.showInputDialog("Introduce la base"));
31             altura=Integer.parseInt(JOptionPane.showInputDialog("Introduce la altura"));
32             System.out.println("El area del triángulo es " + (base * altura)/2);
33             break;
34         case 4:
35             int radio=Integer.parseInt(JOptionPane.showInputDialog("Introduce el radio"));
36             System.out.print("El área del círulo es ");
37             System.out.println(Math.PI*(Math.pow(radio, 2)));
38             break;
39         default:
40             System.out.println("La opción no es correcta");
41         }
42         int altura=Integer.parseInt(JOptionPane.showInputDialog("Dime tu altura en cm."));
43         System.out.println("Si eres hombre tu peso ideal es: " + (altura-110)+ " kg.");
44         System.out.println("Si eres mujer tu peso ideal es: " + (altura-120)+ " kg.");
45         }
46     static int figura;
47 }
```

La opción finally se ejecutará siempre tanto si se ejecuta la excepción o no, de esta forma nos estamos asegurando que simpre se cerrará entrada.close().

Ahora si ejecutamos e introducimos un texto en lugar de una valor entero del 1 al 4 este será el resultado:

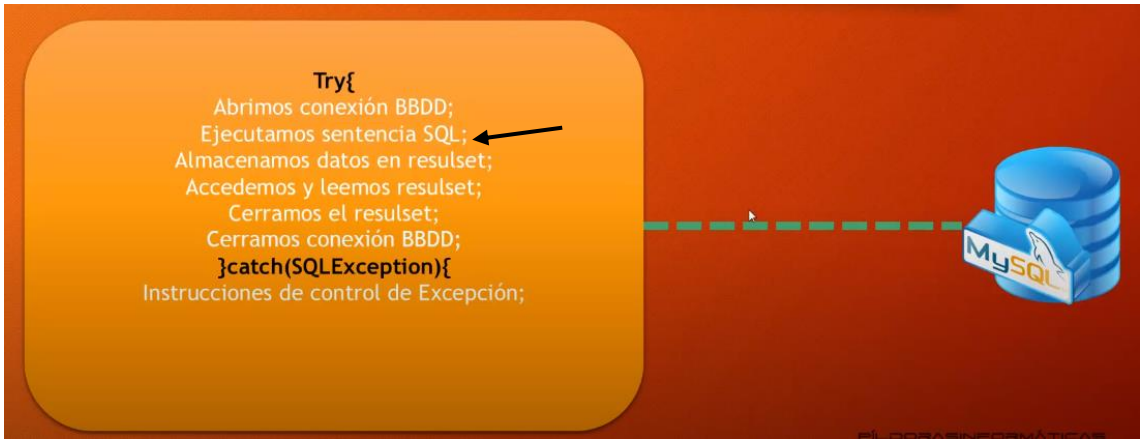
```
elige una opción:
1: Cuadrado
2: Rectángulo
3: Triángulo
4: Circulo
klkl
Ha ocurrido un error
La opción no es correcta
```



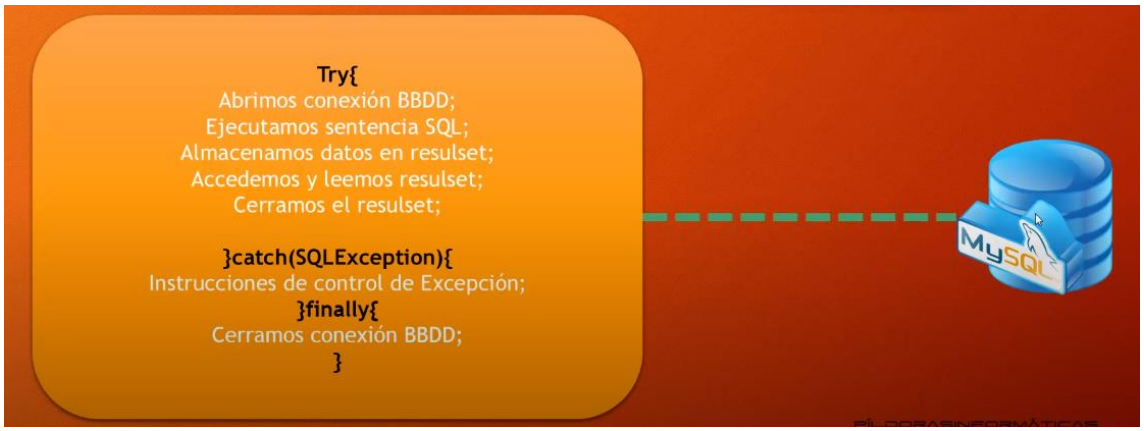

```
elige una opción:
1: Cuadrado
2: Rectángulo
3: Triángulo
4: Circulo
klkl
Ha ocurrido un error
La opción no es correcta
Si eres hombre tu peso ideal es: 75 kg.
Si eres mujer tu peso ideal es: 65 kg.
```

Ha preguntado por la altura y nos ha dado por consola el peso ideal, es decir el programa a continuado.

Cláusula finally. Ejemplo utilidad. Conexión BBDD.



Para evitar que la base de datos quede abierta porque el error ha sido después de abrir la conexión, ya no cierra la base de datos, para que esta no quede abierta lo correcto sería.

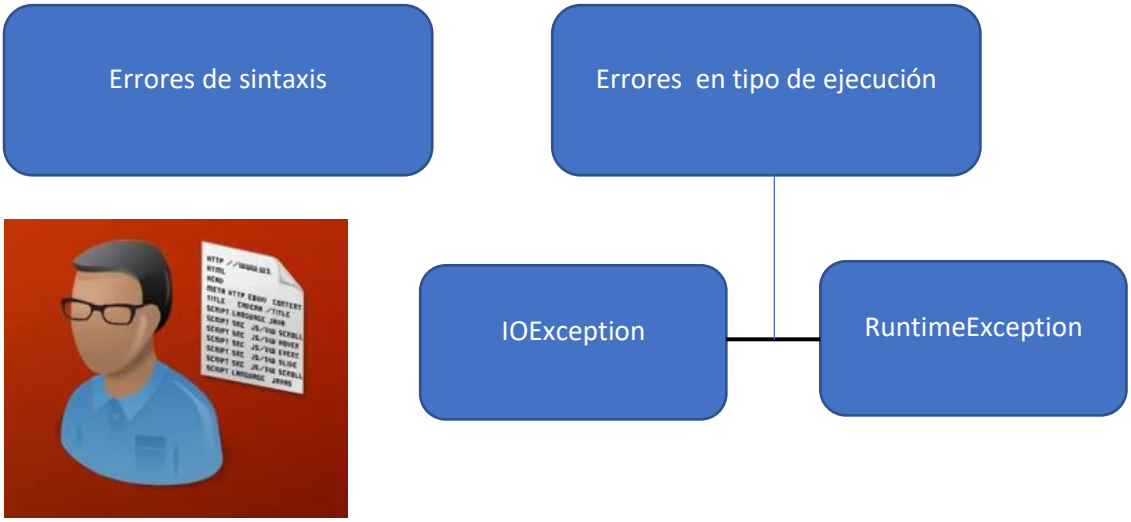


Que pase por finallly de este modo nos aseguramos de que siempre se cerrará.

Depurando con Eclipse. Debugging I. (Vídeo 150)

Depurar código.

Otros tipos de “errores”

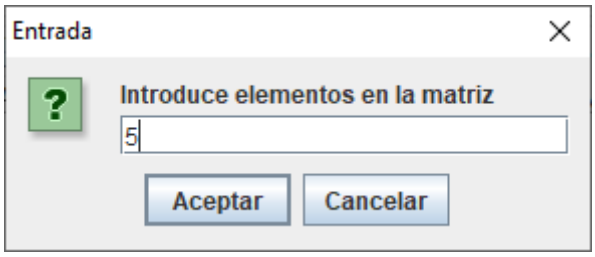


¡Depuración!

Lo vamos a ver con un ejemplo muy sencillo.

```java
package Excepciones_I;
import javax.swing.JOptionPane;
public class Aleatorios {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int elementos=Integer.parseInt(JOptionPane.showInputDialog
                ("Introduce elementos en la matriz"));
        int num_aleat[]=new int[elementos];
        for(int i=0;i<num_aleat.length;i++) {
            num_aleat[i]=(int)Math.random()*100;
        }
        for(int elem: num_aleat) {
            System.out.println(elem);
        }
    }

}
```

Si ejecutamos.



Este será el resultado en consola:

```
0
0
0
0
0
```

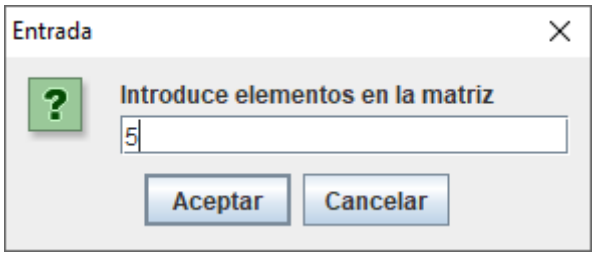
El programa no realiza lo que nosotros esperábamos ya que reíamos imprimir una serie de números aleatorios.

Para poder ver donde está el error en esta aplicación vamos a realizar un punto de interrupción.

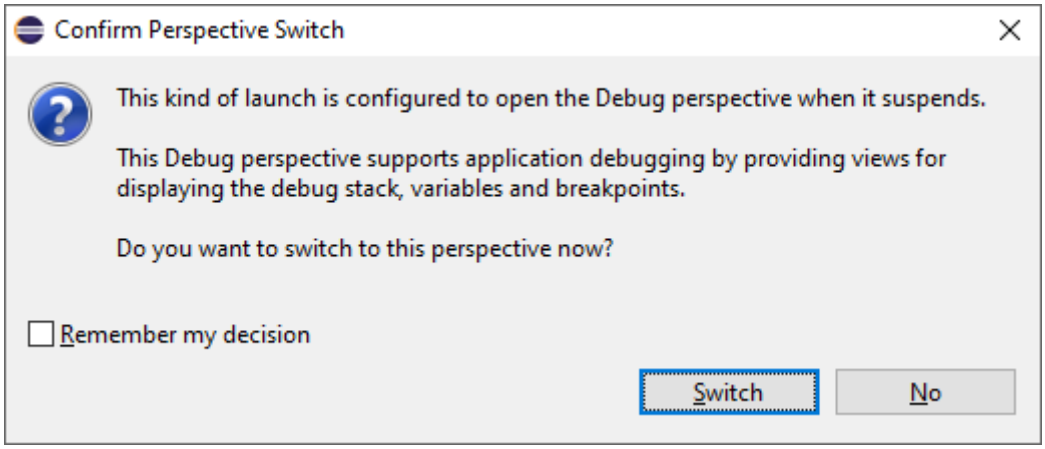
```java
package Excepciones_I;
import javax.swing.JOptionPane;
public class Aleatorios {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int elementos=Integer.parseInt(JOptionPane.showInputDialog
                ("Introduce elementos en la matriz"));
        int num_aleat[]=new int[elementos];
        for(int i=0;i<num_aleat.length;i++) {
            num_aleat[i]=(int)Math.random()*100;
        }
        for(int elem: num_aleat) {
            System.out.println(elem);
        }
    }
}
```

Hacemos doble clic en la parte izquierda donde está la franja azul y aparecerá un punto azul.

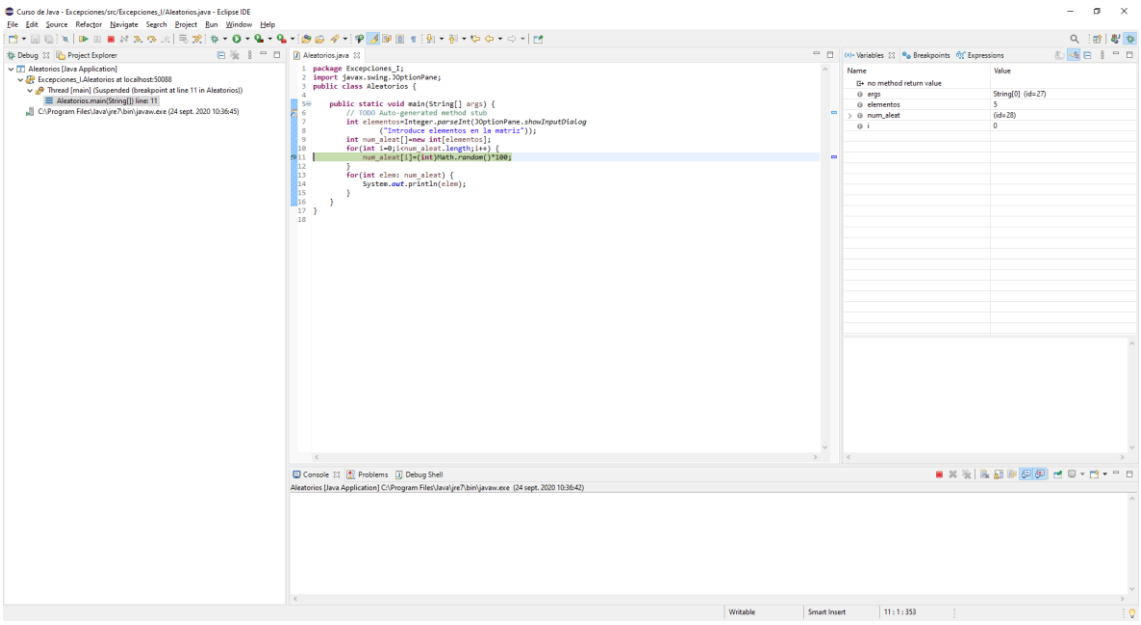
Seleccionamos el botón debug aleatorios.



Le damos a aceptar.

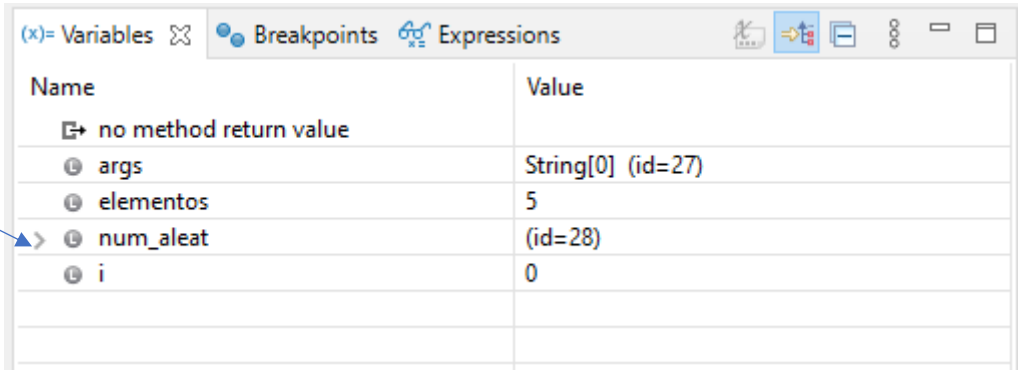


Nos pregunta si queremos entrar a vista de debug a lo que contestaremos con Switch.



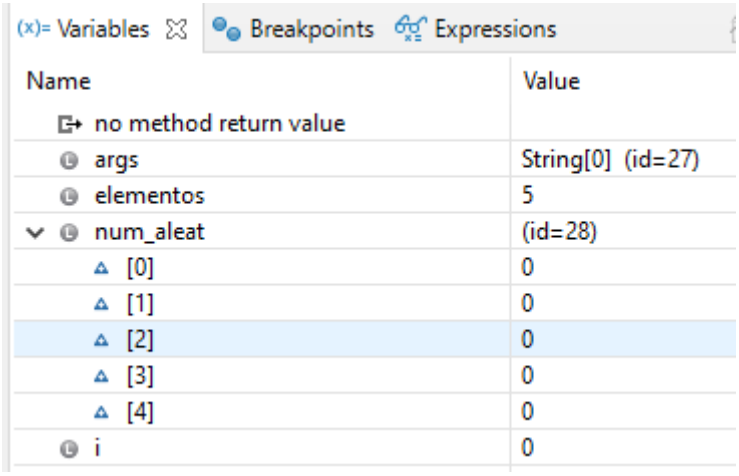
```java
package Excepciones_I;
import javax.swing.JOptionPane;
public class Aleatorios {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int elementos=Integer.parseInt(JOptionPane.showInputDialog
                ("Introduce elementos en la matriz"));
        int num_aleat[]=new int[elementos];
        for(int i=0;i<num_aleat.length;i++) {
            num_aleat[i]=(int)Math.random()*100;
        }
        for(int elem: num_aleat) {
            System.out.println(elem);
        }
    }
}

```

Donde ha parado la ejecución del programa.



Vamos a recalcar el panel de variables.

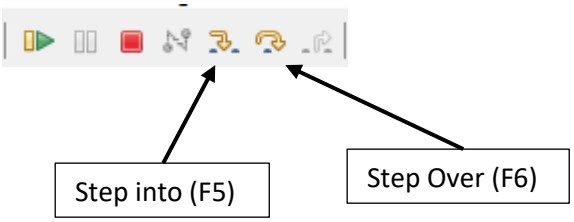
Seleccionamos en el triángulo.



Nos dice cuántos elementos tiene la array y el valor actual de estos elementos.

Y además la variable i del bucle for.

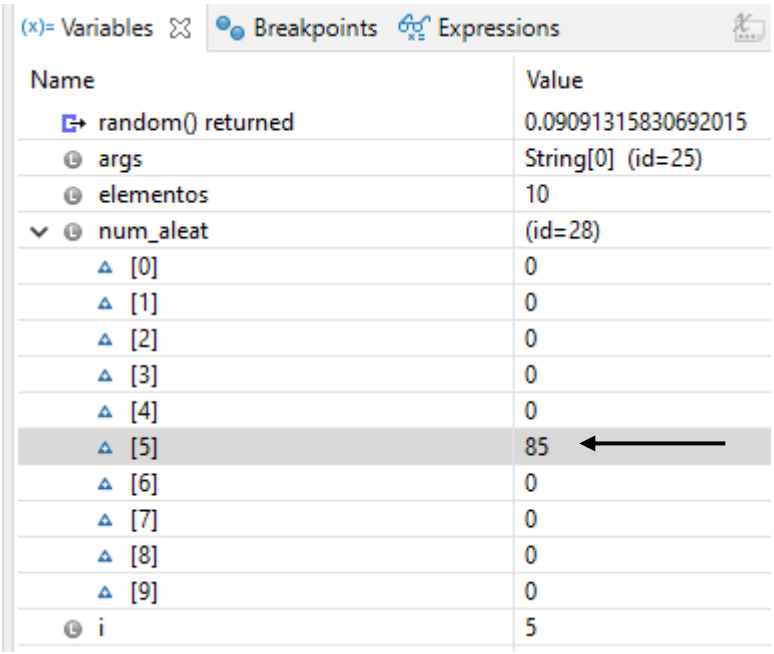
Hay dos botones a tener en cuenta:



Step Over: siver para saltar a la siguiente línea.

Observamos que la variable i cambia de valor pero los valores de la array continúan a 0.

Hemos observado que entramos en el bucle for pero no se están guardando los números aleatorios.



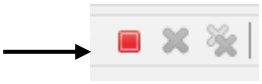
Para comprobar si lee los valores vamos a introducir un valor manualmente.

Seguimos dando al botón step over para que siga el proceso y cuando salta al segundo ciclo for que tiene que imprimir los valores en consola observamos que el valor introducido lo ha leído correctamente.

```
0
0
0
0
0
85
```

Ahora sabemos que el error se encuentra en el primer bucle for y no en el segundo.

Si queremos finalizar antes de que termine el programa.



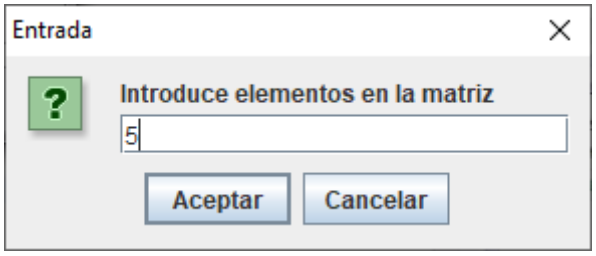
Le damos a stop.

```java
package Excepciones_I;
import javax.swing.JOptionPane;
public class Aleatorios {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int elementos=Integer.parseInt(JOptionPane.showInputDialog
                ("Introduce elementos en la matriz"));
        int num_aleat[]=new int[elementos];
        for(int i=0;i<num_aleat.length;i++) {
            num_aleat[i]=(int)(Math.random()*100);
        }
        for(int elem: num_aleat) {
            System.out.println(elem);
        }
    }
}
```

El error era la falta de paréntesis, ya que al calcular un número al azar realiza el casting es decir lo convierte a entero con lo cual el valor es 0 que multiplicado por 0 su valor será siempre 0.

Al poner paréntesis damos prioridad a la multiplicación antes de realizar el casting.

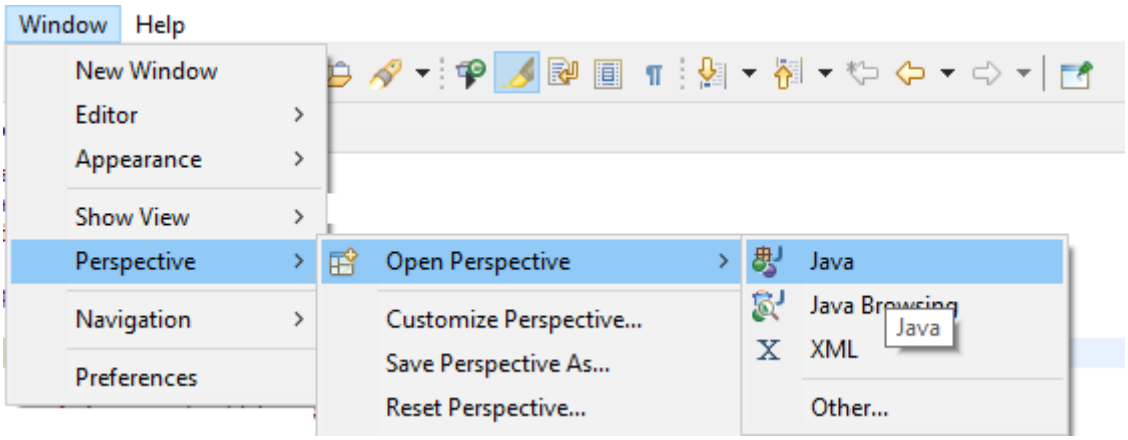
Ejecutamos de nuevo el programa:



Este será el resultado en consola:

```
11
82
23
37
64
```

Para dejar el Eclipse como esta antes del menú Windows, seleccionamos Perspective, Open Perspectieve y de este Hava.



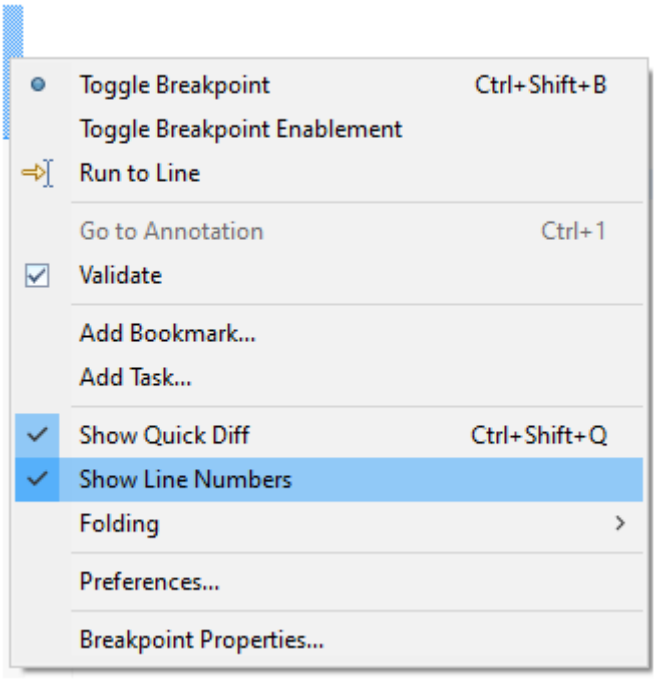

CURSO JAVA
150
DEPURANDO CON ECLIPSE
(Debugging)
eclipse
Java

Depurando con Eclipse. Debugging II. (Vídeo 151)

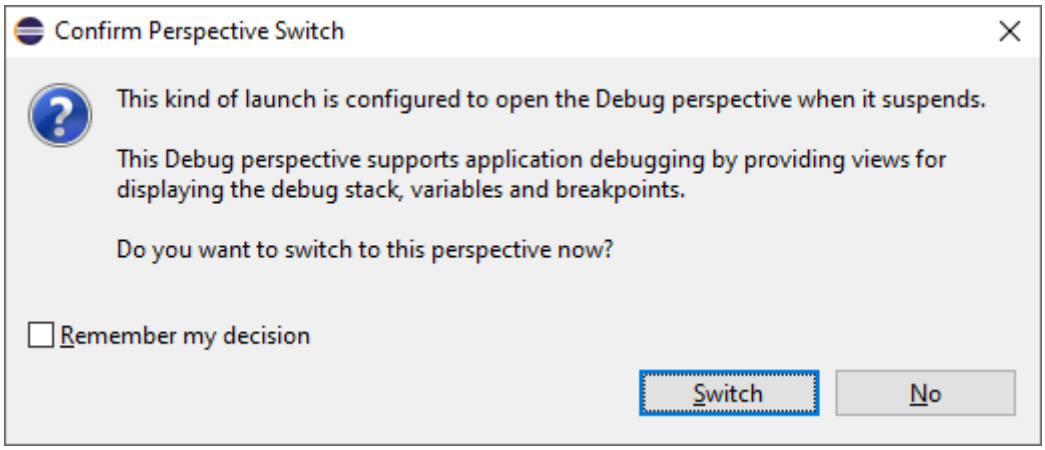
```
1  package Excepciones_I;
2  import javax.swing.JOptionPane;
3  public class Aleatorios {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int elementos=Integer.parseInt(JOptionPane.showInputDialog
8                  ("Introduce elementos en la matriz"));
9          int num_aleat[]=new int[elementos];
10         for(int i=0;i<num_aleat.length;i++) {
11             num_aleat[i]=(int)(Math.random()*100);
12         }
13         for(int elem: num_aleat) {
14             System.out.println(elem);
15         }
16     }
17 }
```

Cuando el programa es muy largo es interesante tener más de un punto de interrupción.

Para que las líneas estén activadas si estas alguna vez las perdemos es seleccionar con el botón derecho sobre la franja azul.



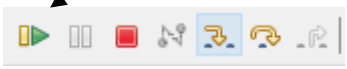
Ahora que tenemos marcado dos puntos de interrupción vamos a ejecutar el debug aleatorios, recuerda que el botón tiene un pequeño escarabajo.

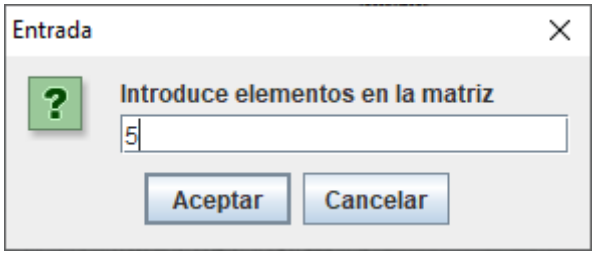


Seleccionamos Switch.

```
1  package Excepciones_I;
2  import javax.swing.JOptionPane;
3  public class Aleatorios {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int elementos=Integer.parseInt(JOptionPane.showInputDialog
8                  ("Introduce elementos en la matriz"));
9          int num_aleat[]=new int[elementos];
10         for(int i=0;i<num_aleat.length;i++) {
11             num_aleat[i]=(int)(Math.random()*100);
12         }
13         for(int elem: num_aleat) {
14             System.out.println(elem);
15         }
16     }
17 }
18
```

Si queremos saltar al siguiente punto de interrupción ejecutando las líneas intermedias seleccionaremos el botón Resume o F8.





Pregunta por los elementos de la matriz, seguido del botón Aceptar.

```java
package Excepciones_I;
import javax.swing.JOptionPane;
public class Aleatorios {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int elementos=Integer.parseInt(JOptionPane.showInputDialog
                ("Introduce elementos en la matriz"));
        int num_aleat[]=new int[elementos];
        for(int i=0;i<num_aleat.length;i++) {
            num_aleat[i]=(int)(Math.random()*100);
        }
        for(int elem: num_aleat) {
            System.out.println(elem);
        }
    }
}
```

Ya se ha ido al otro punto de interrupción.

Podemos hacer un seguimiento del valor de las variables.

Name	Value
no method return value	
args	String[0] (id=16)
elementos	5
num_aleat	(id=28)
[0]	0
[1]	0
[2]	0
[3]	0
[4]	0
i	0

Con F6 vas saltando línea a línea.

(x)= Varia... | Outline | Break... | Expr...

Name	Value
no method return value	
args	String[0] (id=16)
elementos	5
num_aleat	(id=28)
[0]	30
[1]	11
[2]	72
[3]	71
[4]	85

Hasta comprobar que ya tiene todos los valores en el array.

Si le damos a este botón le estamos diciendo que te saltes todos los puntos de interrupción.

```
1  package Excepciones_I;
2  import javax.swing.JOptionPane;
3  public class Aleatorios {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int elementos=Integer.parseInt(JOptionPane.showInputDialog
8                  ("Introduce elementos en la matriz"));
9          int num_aleat[]=new int[elementos];
10         for(int i=0;i<num_aleat.length;i++) {
11             num_aleat[i]=(int)(Math.random()*100);
12         }
13         for(int elem: num_aleat) {
14             System.out.println(elem);
15         }
16     }
17 }
```

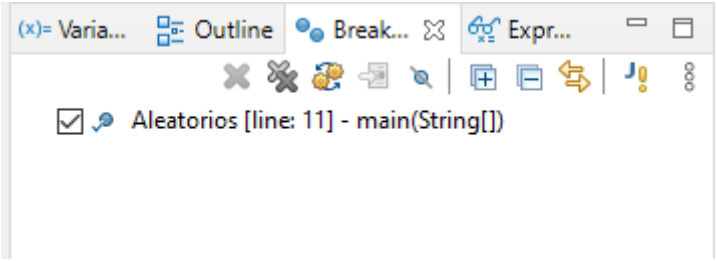
Podrás observar que hemos anulado los puntos de interrupción si pulsamos F8 se ejecutará el programa hasta el final.

Si le damos de nuevo al botón estos puntos de interrupción volverán a estar activos.

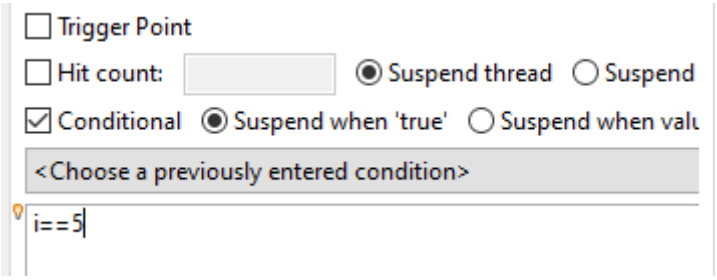
```
1  package Excepciones_I;
2  import javax.swing.JOptionPane;
3  public class Aleatorios {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int elementos=Integer.parseInt(JOptionPane.showInputDialog
8                  ("Introduce elementos en la matriz"));
9          int num_aleat[]=new int[elementos];
10         for(int i=0;i<num_aleat.length;i++) {
11             num_aleat[i]=(int)(Math.random()*100);
12         }
13         for(int elem: num_aleat) {
14             System.out.println(elem);
15         }
16     }
17 }
```

Vamos a quitar el punto de interrupción que teníamos en la línea 7 haciendo doble clic sobre el punto.

En la pestaña breakpoints nos muestra todos los puntos de interrupción de todos los programas incluso de programas que no estén abiertos.



Seleccionamos la interrupción y en la parte inferior



Seleccionamos la casilla Condicional y como condición ponemos que i==5, esto significa que si nosotros contestamos por el número de valores que tendrá la array por un 3 como nunca se cumplirá esta condición el programa llegará a su final, pero si ponemos como valor un 8 verás que al llegar al valor 5 el programa se detendrá.

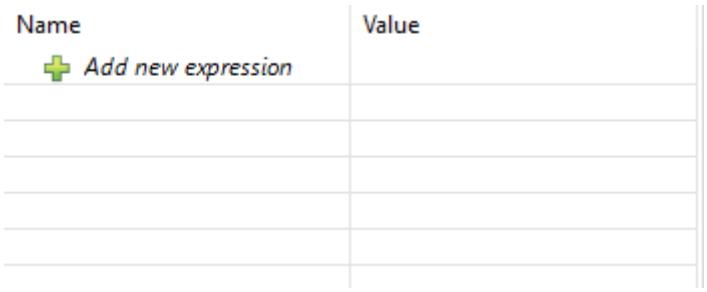
```
1  package Excepciones_I;
2  import javax.swing.JOptionPane;
3  public class Aleatorios {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int elementos=Integer.parseInt(JOptionPane.showInputDialog
8                  ("Introduce elementos en la matriz"));
9          int num_aleat[]=new int[elementos];
10         for(int i=0;i<num_aleat.length;i++) {
11             num_aleat[i]=(int)(Math.random()*100);
12         }
13         for(int elem: num_aleat) {
14             System.out.println(elem);
15         }
16     }
17 }
```

El programa se interrumpe en dicha línea.

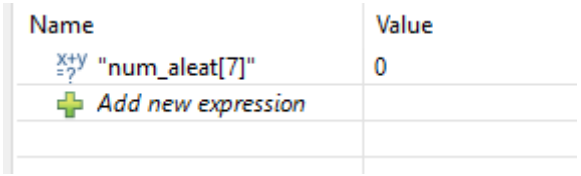
Name	Value
no method return value	
args	String[0] (id=25)
elementos	10
num_aleat	(id=27)
[0]	37
[1]	22
[2]	80
[3]	24
[4]	14
[5]	0
[6]	0
[7]	0
[8]	0
[9]	0
i	5

Vemos como 5 valores de los 10 ya tienen valores y la variable i vale 5.

Otro panel interesante es el panel Expresiones, si no lo ves nos vamos al menú Windows, Show view y de este activamos Expresions.

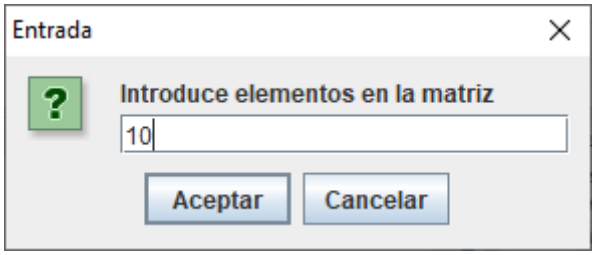


Quitamos la condición del punto de interrupción anterior.



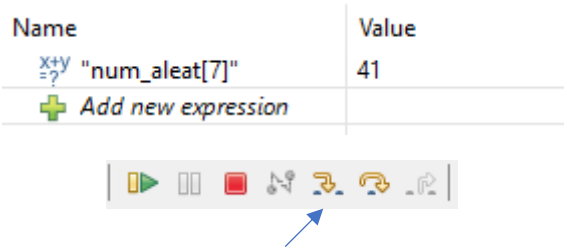
Cuando este valor lo tenga que nos lo muestre.

Ejecutamos nuestro programa con el debug.



Le decimos que tiene 10 elementos.

A continuación damos a la tecla F6 para que salte de línea en línea, cuando el elemento de la matriz 7 tenga un valor este nos lo dirá.



Para que sirve el botón step Into (F5), para este ejemplo vamos a realizar primero la siguiente interface llamada Calcula:

```
1  package Excepciones_I;
2
3  public interface Calcula {
4      public int calculo(int num1, int num2);
5
6  }
```

Vamos a crear una clase llamada Suma:

```java
package Excepciones_I;

public class Suma  implements Calcula{
    public int calculo(int num1, int num2) {
        return num1+num2;
    }

}
```

Vamos a crear una clase llamada Resta:

```java
package Excepciones_I;

public class Resta implements Calcula {
    public int calculo(int num1, int num2) {
        return num1-num2;
    }

}
```

Vamos a crear una clase llamada Multiplicacion:

```java
package Excepciones_I;

public class Multiplicacion implements Calcula {
    public int calculo(int num1, int num2) {
        return num1+num2;
    }
}
```

Observarás un error en lugar de multiplicar suma esta realizado a propósito.

Vamos a crear la clase Divide:

```java
package Excepciones_I;

public class Divide implements Calcula {
    public int calculo(int num1, int num2) {
        return num1/num2;
    }
}
```

Vamos a crear la clase principal llamada Operaciones.

```java
package Excepciones_I;

public class Operaciones {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Suma operacion1=new Suma();
        Resta operacion2=new Resta();
        Multiplicacion operacion3=new Multiplicacion();
        Divide operacion4=new Divide();
        System.out.println(operacion1.calculo(7,8));
        System.out.println(operacion2.calculo(7,8));
        System.out.println(operacion3.calculo(7,8));
        System.out.println(operacion4.calculo(7,8));
    }
}
```

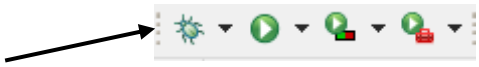
Si ejecutamos observaremos el siguiente resultado:

```
15
-1
15
0
```

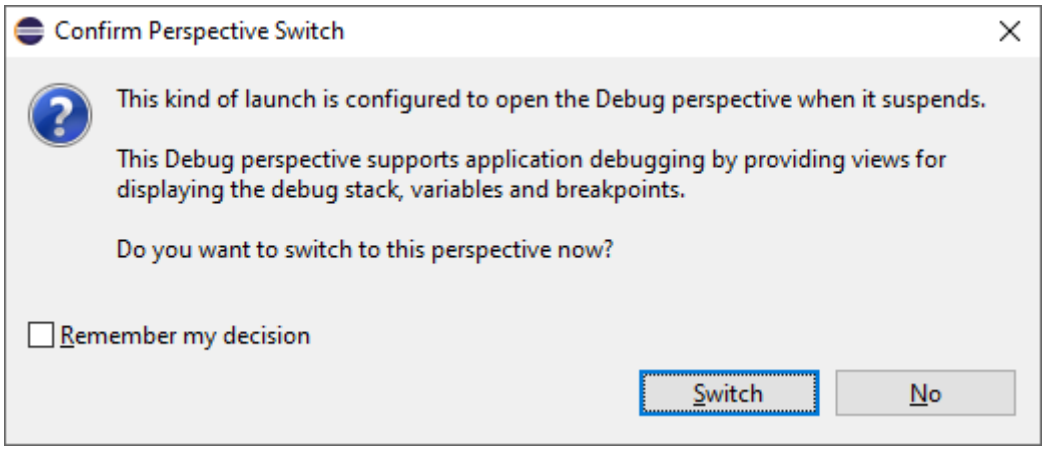
```
1  package Excepciones_I;
2
3  public class Operaciones {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          Suma operacion1=new Suma();
8          Resta operacion2=new Resta();
9          Multiplicacion operacion3=new Multiplicacion();
10         Divide operacion4=new Divide();
11         System.out.println(operacion1.calculo(7,8));
12         System.out.println(operacion2.calculo(7,8));
13         System.out.println(operacion3.calculo(7,8));
14         System.out.println(operacion4.calculo(7,8));
15     }
16 }
```

Introducimos un punto de interrupción.

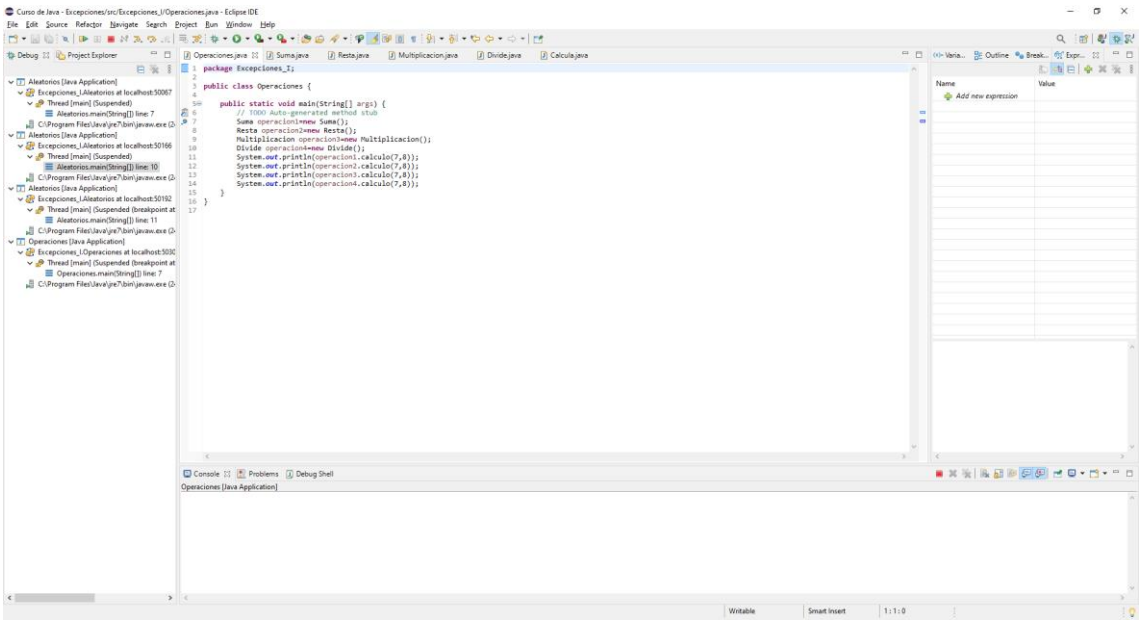
Estamos viendo que la suma y la multiplicación dan el mismo resultado, si pasamos los valores 7 y 8 la multiplicación sería 56.



Ejecutamos el debuger.



Seleccionamos Switch.



Vamos pulsando F6 y cuando nos encontramos en la línea que hace el error.

```
1  package Excepciones_I;
2
3  public class Operaciones {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          Suma operacion1=new Suma();
8          Resta operacion2=new Resta();
9          Multiplicacion operacion3=new Multiplicacion();
10         Divide operacion4=new Divide();
11         System.out.println(operacion1.calculo(7,8));
12         System.out.println(operacion2.calculo(7,8));
13         System.out.println(operacion3.calculo(7,8));
14         System.out.println(operacion4.calculo(7,8));
15     }
16 }
```

Pulsamos el botón step Into o F5.

```
1  package Excepciones_I;
2
3  public class Multiplicacion implements Calcula {
4      public int calculo(int num1, int num2) {
5          return num1+num2;
6      }
7  }
```

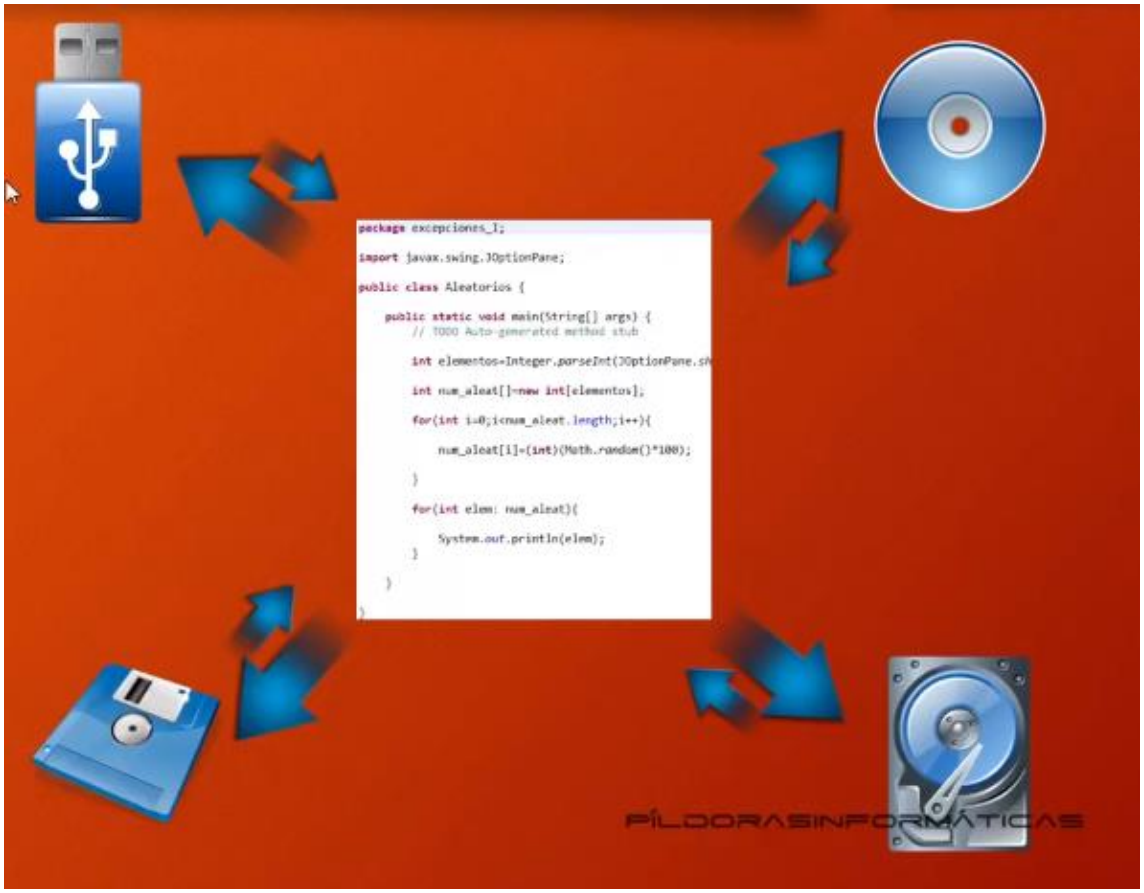
Nos muestra la clase que contiene el error vemos que suma en lugar de multiplica.

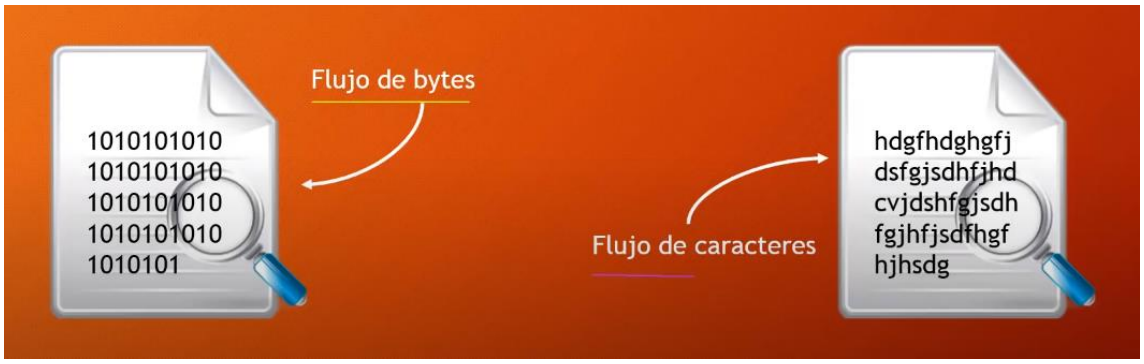
Y reparamos el error.

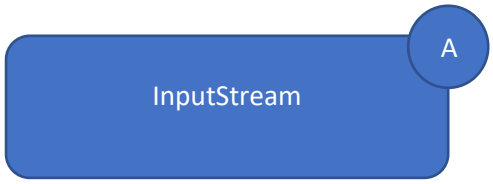
```java
package Excepciones_I;

public class Multiplicacion implements Calcula {
    public int calculo(int num1, int num2) {
        return num1*num2;
    }
}
```

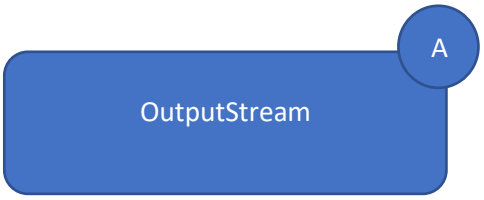
Streams I. Accediendo a ficheros. Lectura. (Vídeo 152)

¿Por qué el uso de ficheros?

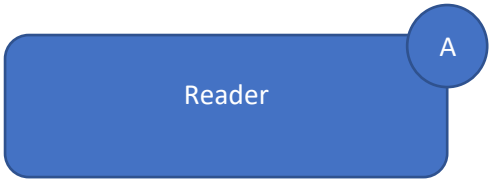


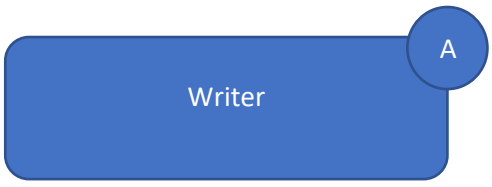




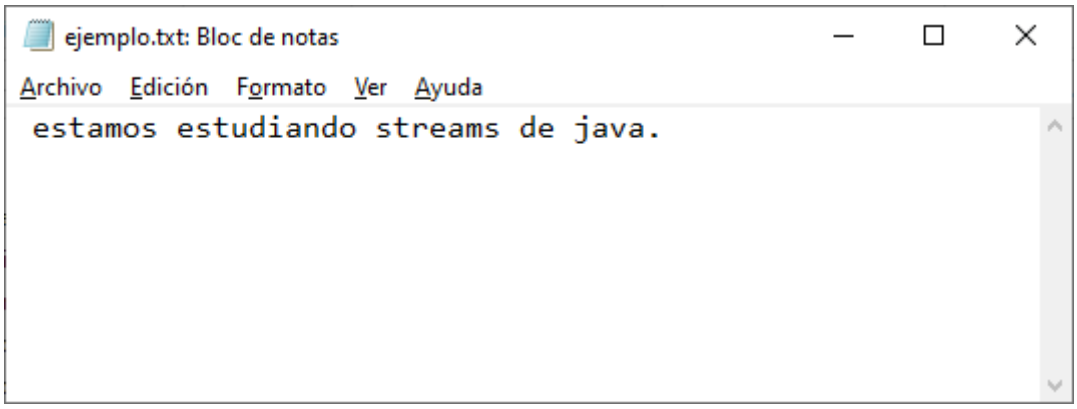


Son clases abstractas.



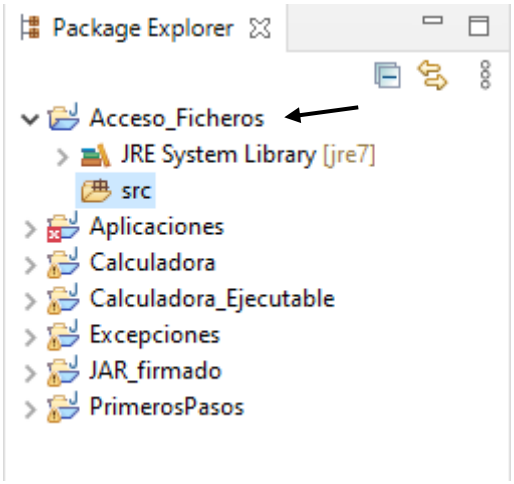


Vamos a realizar un archivo con el bloc de notas llamado ejemplo.txt.

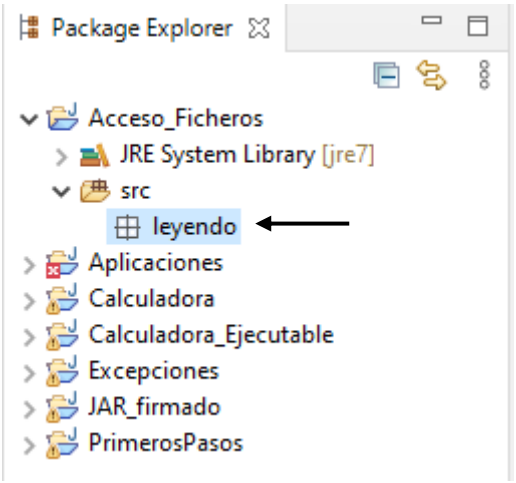


Lo he guardado en la carpeta del curso de java.

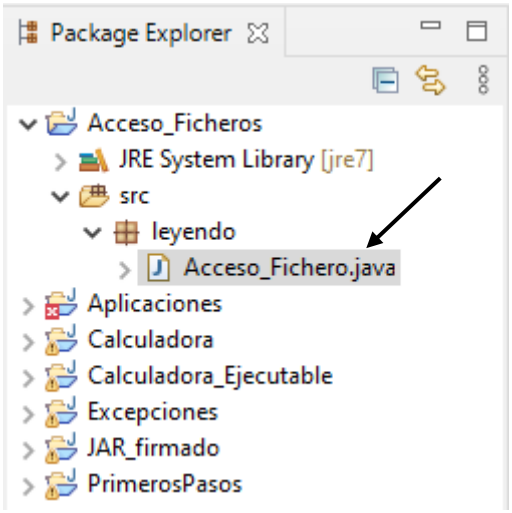
Dentro de Eclipse vamos a crear un proyecto nuevo llamado Acceso_Ficheros.



Vamos a crear un nuevo paquete leyendo.



Vamos a crear una clase principal llamada Acceso_Fichero.




```java
package leyendo;

import java.io.*;

public class Acceso_Fichero {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Leer_Fichero accediendo = new Leer_Fichero();
        accediendo.lee();
    }
}
class Leer_Fichero{
    public void lee() {
        try {
            FileReader entrada=new FileReader("D:/Curso de Java/ejemplo.txt");
            int c=entrada.read();
            while(c!=-1) {
                c=entrada.read();
                        System.out.print(c + " ");
            }
        } catch (IOException e) {
            // TODO Auto-generated catch block
            System.out.println("No se ha encotrado el archivo");
        }

    }
}
```

Una vez escrito el siguiente código lo ejecutamos.

```
115 116 97 109 111 115 119 32 101 115 116 117 100 105 97 110 100 111 32 115 116 114 101 97 109 115 32 100 101 32 106 97 118 97 -1
```

Vemos que el último carácter es -1 esto nos indica que ha llegado al final.

Ahora hemos de convertir estos caracteres a su correspondiente carácter.

```java
package leyendo;

import java.io.*;

public class Acceso_Fichero {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Leer_Fichero accediendo = new Leer_Fichero();
        accediendo.lee();
    }
}
class Leer_Fichero{
    public void lee() {
        try {
            FileReader entrada=new FileReader("D:/Curso de Java/ejemplo.txt");
            int c=entrada.read();
            while(c!=-1) {
                c=entrada.read();
                char letra=(char)c;
                System.out.print(letra);
            }
        } catch (IOException e) {
            // TODO Auto-generated catch block
            System.out.println("No se ha encotrado el archivo");
        }

    }
}
```

Hacemos un casting para reconvertirlo en su carácter que luego se imprime en consola.

Si ejecutamos este será el resultado:

```
estamos estudiando streams de java.?
```

El ¿ corresponde al -1.

Si modificamos la ubicación del archivo para que no lo encuentre.

```
No se ha encotrado el archivo
```


CURSO JAVA
152
SECUENCIAS (STREAMS)
Manejo de archivos. Acceso a Ficheros
eclipse
Java

Streams II. Accediendo a ficheros Escritura. (Vídeo 153)

```
1  package leyendo;
2
3  import java.io.*;
4
5  public class Acceso_Fichero {
6
7      public static void main(String[] args) {
8          // TODO Auto-generated method stub
9          Leer_Fichero accediendo = new Leer_Fichero();
10         accediendo.lee();
11     }
12 }
13 class Leer_Fichero{
14     public void lee() {
15         try {
16             FileReader entrada=new FileReader("D:/Curso de Java/ejemplo.txt");
17             int c=entrada.read();
18             while(c!=-1) {
19                 c=entrada.read();
20                 char letra=(char)c;
21                 System.out.print(letra);
22             }
23             entrada.close();
24         } catch (IOException e) {
25             // TODO Auto-generated catch block
26             System.out.println("No se ha encotrado el archivo");
27         }
28
29     }
30 }
```

En el capítulo anterior nos olvidamos cerrar el fichero que abrimos para leerlo.

En el ejemplo anterior en la línea 17 la variable c ya está leyendo el primer carácter y el la línea 19 lee el segundo carácter que a continuación imprime y durante el bucle while imprime el resto de caracteres, para que se imprima desde el primer carácter realizaremos las siguientes modificaciones.

```
1  package leyendo;
2
3  import java.io.*;
4
5  public class Acceso_Fichero {
6
7      public static void main(String[] args) {
8          // TODO Auto-generated method stub
9          Leer_Fichero accediendo = new Leer_Fichero();
10         accediendo.lee();
11     }
12 }
13 class Leer_Fichero{
14     public void lee() {
15         try {
16             FileReader entrada=new FileReader("D:/Curso de Java/ejemplo.txt");
```

```
17              int c;
18               do {
19                  c=entrada.read();
20                  char letra=(char)c;
21                  System.out.print(letra);
22              }while(c!=-1);
23              entrada.close();
24          } catch (IOException e) {
25              // TODO Auto-generated catch block
26              System.out.println("No se ha encotrado el archivo");
27          }
28
29      }
30  }
```

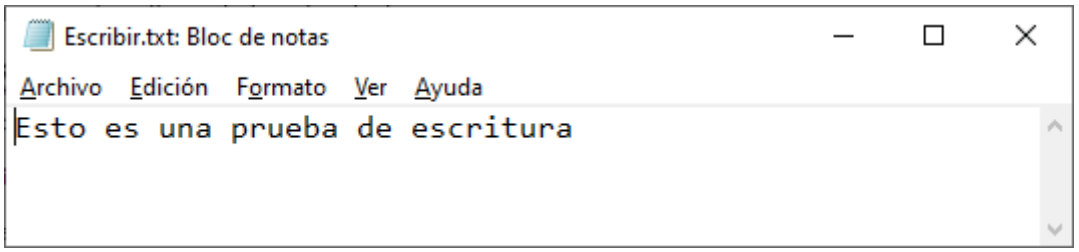
Definimos la variable pero no la inicializamos

Al utilizar un do/while no necesitamos tener inicializada la variable c.

Vamos a realizar una nueva clase llamada Escribir_Fichero.

```
1  package leyendo;
2  import java.io.*;
3  public class Escribir_Fichero {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          Escribiendo accede_es=new Escribiendo();
8          accede_es.escribir();
9      }
10 }
11
12 class Escribiendo{
13     public void escribir() {
14         String frase="Esto es una prueba de escritura";
15         try {
16             FileWriter escritura=new FileWriter("D:/Curso de Java/Escribir.txt");
17             for (int i=0;i<frase.length();i++) {
18                 escritura.write(frase.charAt(i));
19             }
20             escritura.close();
21         } catch (IOException e) {
22             // TODO Auto-generated catch block
23             e.printStackTrace();
24         }
25     }
26 }
```

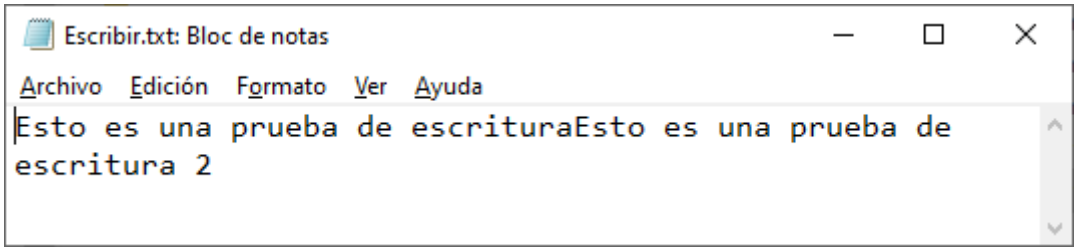
Al ejecutar observamos que se ha creado un fichero llamado Escribir.txt, si lo abrimos este será el contenido.



Si queremos escribir en un fichero ya existente hemos de agregar el true.

```java
package leyendo;
import java.io.*;
public class Escribir_Fichero {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Escribiendo accede_es=new Escribiendo();
        accede_es.escribir();
    }
}

class Escribiendo{
    public void escribir() {
        String frase="Esto es una prueba de escritura 2";
        try {
            FileWriter escritura=new FileWriter("D:/Curso de Java/Escribir.txt", true);
            for (int i=0;i<frase.length();i++) {
                escritura.write(frase.charAt(i));
            }
            escritura.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Ejecutamos de nuevo.

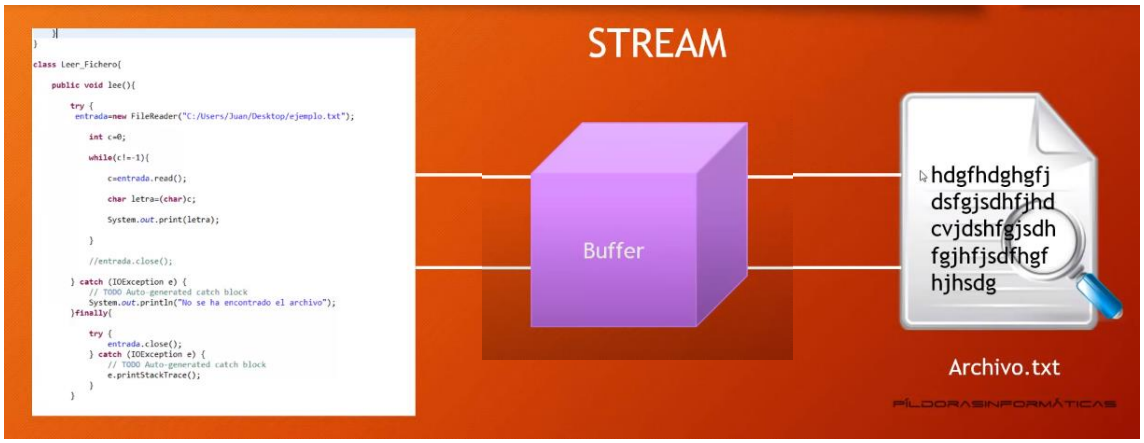


Hemos agregado texto.

Si quitamos el true lo que hará será sobrescribir el fichero existente.

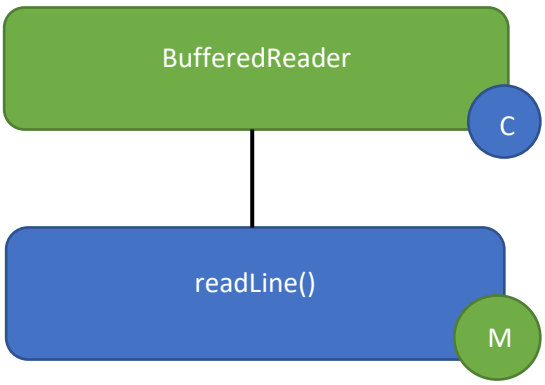

CURSO JAVA
153
SECUENCIAS (STREAMS)
Manejo de archivos. Acceso a Ficheros
Escribiendo ficheros
eclipse
Java

Streams III. Usando buffers. (Vídeo 154)



Buffer es la memoria interna la información se vuelca completamente al buffer y es el programa java el que accede para ir descarándolo poco a poco.

El buffer lo podemos tanto para leer información de un archivo como para escribir.

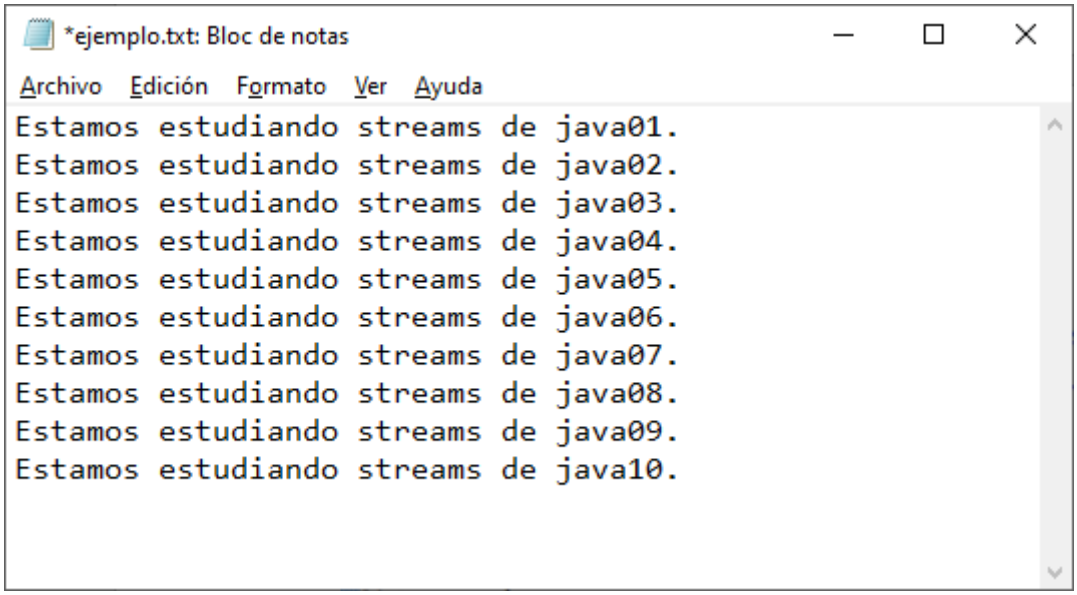


Para este ejemplo vamos a trabajar sobre el fichero Acceso_Fichero.

```
1  package leyendo;
2
3  import java.io.*;
4
5  public class Acceso_Fichero {
6
7      public static void main(String[] args) {
8          // TODO Auto-generated method stub
9          Leer_Fichero accediendo = new Leer_Fichero();
10         accediendo.lee();
11     }
12 }
```

```java
class Leer_Fichero{
    public void lee() {
        try {
            FileReader entrada=new FileReader("D:/Curso de Java/ejemplo.txt");
            int c=0;
            while(c!=-1) {
                c=entrada.read();
                char letra=(char)c;
                System.out.print(letra);
            }
            entrada.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            System.out.println("No se ha encotrado el archivo");
        }

    }
}
```

Antes abrimos el archivo ejemplo.txt lo modificamos como se muestra en la siguiente figura y lo guardamos.



Hemos realizado las siguientes modificaciones:

```java
package leyendo;

import java.io.*;

public class Acceso_Fichero {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Leer_Fichero accediendo = new Leer_Fichero();
        accediendo.lee();
    }
}
```

```
13 class Leer_Fichero{
14     public void lee() {
15         try {
16             FileReader entrada=new FileReader("D:/Curso de Java/ejemplo.txt");
17             BufferedReader mibuffer=new BufferedReader(entrada);
18             String linea="";
19             while(linea!=null) {
20                 linea=mibuffer.readLine();
21                 System.out.println(linea);
22             }
23             entrada.close();
24         } catch (IOException e) {
25             // TODO Auto-generated catch block
26             System.out.println("No se ha encotrado el archivo");
27         }
28     }
29 }
```

Si ejecutamos este será el resultado:

```
Estamos estudiando streams de java01.
Estamos estudiando streams de java02.
Estamos estudiando streams de java03.
Estamos estudiando streams de java04.
Estamos estudiando streams de java05.
Estamos estudiando streams de java06.
Estamos estudiando streams de java07.
Estamos estudiando streams de java08.
Estamos estudiando streams de java09.
Estamos estudiando streams de java10.
null
```

Si queremos que la palabra null no se muestre realizaríamos lo siguiente:

```
13 class Leer_Fichero{
14     public void lee() {
15         try {
16             FileReader entrada=new FileReader("D:/Curso de Java/ejemplo.txt");
17             BufferedReader mibuffer=new BufferedReader(entrada);
18             String linea="";
19             while(linea!=null) {
20                 linea=mibuffer.readLine();
21                     if(linea!=null) {
22                 System.out.println(linea);
23                 }
24             }
25             entrada.close();
26         } catch (IOException e) {
27             // TODO Auto-generated catch block
28             System.out.println("No se ha encotrado el archivo");
29         }
30     }
31 }
```

En la clase Leer_Fichero agregaríamos in if.

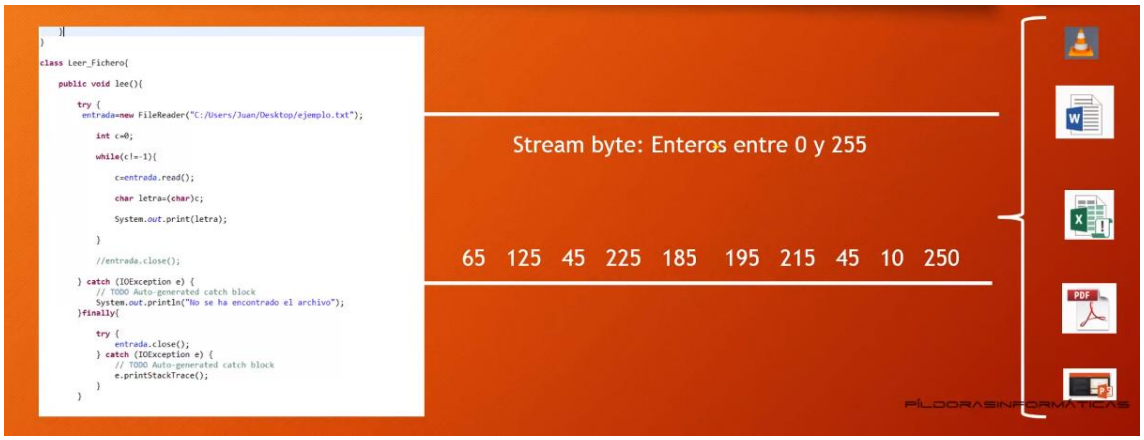
Este será el resultado:

…

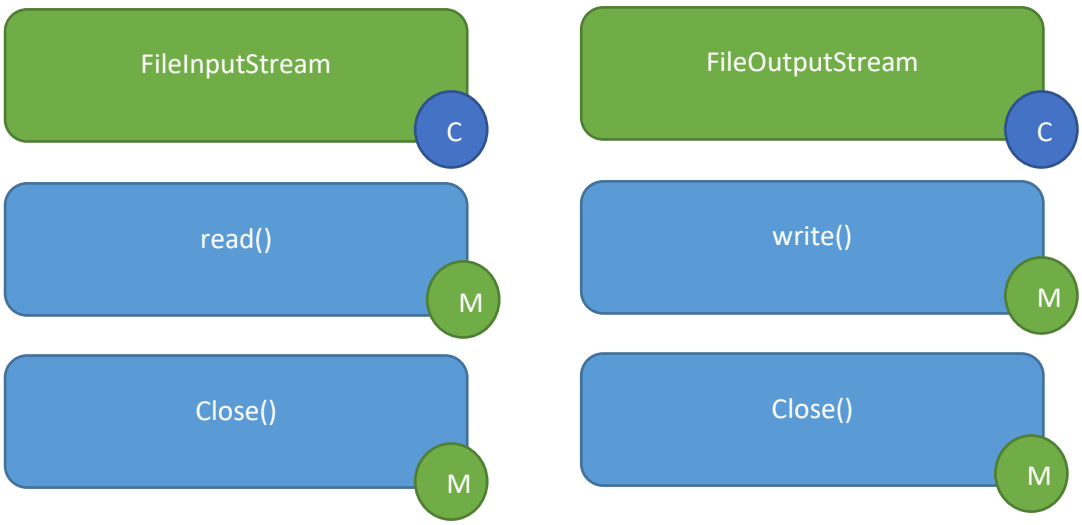
```
Estamos estudiando streams de java08.
Estamos estudiando streams de java09.
Estamos estudiando streams de java10.
```


CURSO JAVA
154
SECUENCIAS (STREAMS)
Manejo de archivos. Acceso a Ficheros
Buffers
eclipse
Java

Streams IV. Leyendo archivos. Streams Byte I. (Vídeo 155)



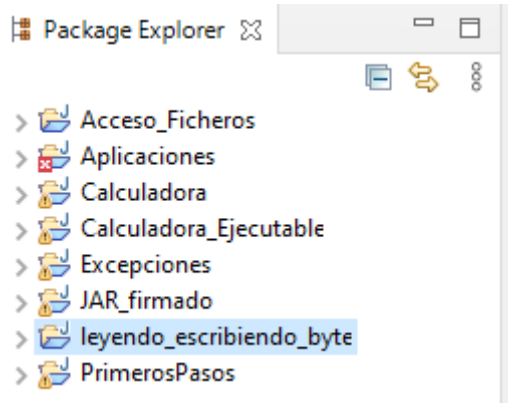
Streams byte



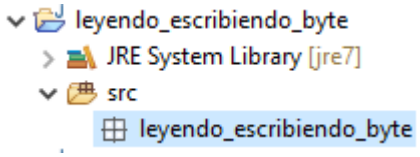
En este capítulo vamos a leer una imagen que está guardada en nuestra carpeta del trabajo.

Vamos a utilizar esta imagen de ejemplo.

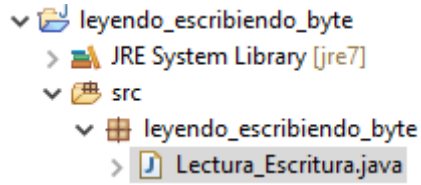
Vamos a crear un nuevo proyecto llamado leyendo_escribiendo_byte.



Creamos un paquete con el mismo nombre.



Vamos a crear una clase llamada Lectura_Escritura con el método main.




```java
package leyendo_escribiendo_byte;
import java.io.*;
public class Lectura_Escritura {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try {
            FileInputStream archivo_lectura=new
                    FileInputStream("D:/Curso de Java/castillo.jpg");
            boolean final_ar=false;
            while(!final_ar) {
                int byte_entrada=archivo_lectura.read();
                if(byte_entrada==-1)
                    final_ar=true;
                System.out.println(byte_entrada);
            }
            archivo_lectura.close();
        }catch(IOException e){}
    }

}
```

Se muestra una gran serie de números y terminando por -1.

```
0
123
62
191
242
127
255
217
-1
```

Queremos saber cuantos bytes tiene la imagen.

```java
package leyendo_escribiendo_byte;
import java.io.*;
public class Lectura_Escritura {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int contador=0;
        try {
            FileInputStream archivo_lectura=new
                    FileInputStream("D:/Curso de Java/castillo.jpg");
            boolean final_ar=false;
            while(!final_ar) {
                int byte_entrada=archivo_lectura.read();
                if(byte_entrada==-1)
                    final_ar=true;
                System.out.println(byte_entrada);
                contador++;
            }
            System.out.println("El número de bytes " + contador);
            archivo_lectura.close();
        }catch(IOException e){}
    }

}
```

Declaramos e inicializamos un contador

A cada vuelta cuenta.

Imprime las vueltas.

Este será el resultado_

```
242
127
255
217
-1
El número de bytes 20096
```

Ahora tenemos el número de bytes, cogerlos y crear un nuevo archivo con ellos, si tenemos éxito deberías crear una imagen exactamente igual como la que estamos leyendo.

```java
package leyendo_escribiendo_byte;
import java.io.*;
public class Lectura_Escritura {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int contador=0;
        int datos_entrada[]=new int[20096];
```

Gracias al contador sabemos el número de bytes que tiene la imagen.

```
 9          try {
10              FileInputStream archivo_lectura=new
11                      FileInputStream("D:/Curso de Java/castillo.jpg");
12              boolean final_ar=false;
13              while(!final_ar) {
14                  int byte_entrada=archivo_lectura.read();
15                  if(byte_entrada!=-1)
16                      datos_entrada[contador]=byte_entrada;
17                  else
18                      final_ar=true;
19                  System.out.println(datos_entrada[contador]);
20
21                  contador++;
22              }
23              System.out.println("El número de bytes " + contador);
24              archivo_lectura.close();
25          }catch(IOException e){}
26      }
27
28  }
```

Valores que pasaos a la array

Que después podemos imprimirl

El resultado será:

```
191
242
127
255
217
0
El número de bytes 20096
```

El valor -1 final de fichero ya no se imprime, ya que este valor no es información del fichero.

Ahora te planteo que el resultado se muestre de la siguiente forma:

```
104  204  140  140  174  89  24  140  130  160  113  246  189  59  140  14  199  254  222  179  158  248  35  56  239
130  193  143  253  35  247  244  220  61  222  127  42  47  107  249  253  207  119  220  79  107  220  229  254  12
127  215  245  234  255  0  181  241  190  79  11  63  43  135  249  62  247  185  247  103  151  217  203  60  179
199  235  213  191  141  254  102  19  143  181  203  219  227  136  249  227  31  210  207  30  60  127  222  235  91
159  145  239  123  178  124  79  115  158  61  175  115  238  247  61  223  241  123  158  159  199  233  213  174  60
57  123  137  242  56  242  247  61  204  253  120  255  0  135  211  31  183  168  125  252  255  0  249  175  235
140  123  124  223  221  231  203  183  31  215  166  246  243  142  255  0  187  25  108  240  250  227  63  199  169
190  15  249  223  22  95  242  51  238  252  207  112  251  95  35  31  251  92  184  125  49  158  143  204  227
236  114  251  56  242  246  249  224  112  246  191  103  185  159  94  217  206  122  255  0  31  197  247  7  198
247  51  238  112  193  231  241  249  255  0  83  219  206  115  158  221  63  181  242  185  123  131  216  225  143
115  220  239  237  231  151  211  245  207  127  225  212  223  43  28  62  223  141  238  115  255  0  59  43  207
217  229  247  99  151  243  227  191  240  233  62  39  249  159  33  248  126  188  176  61  121  125  216  206  113
158  248  206  122  139  228  251  190  199  24  253  190  62  223  47  119  63  249  88  254  167  183  255  0  135
237  233  185  127  55  124  227  56  231  223  57  255  0  123  62  191  242  127  255  217  0
```

El resultado lo encontrarás e la página siguiente:

```java
package leyendo_escribiendo_byte;
import java.io.*;
public class Lectura_Escritura {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int contador=0;
        int datos_entrada[]=new int[20096];
        try {
            FileInputStream archivo_lectura=new
                    FileInputStream("D:/Curso de Java/castillo.jpg");
            boolean final_ar=false;
            while(!final_ar) {
                int byte_entrada=archivo_lectura.read();
                if(byte_entrada!=-1)
                    datos_entrada[contador]=byte_entrada;
                else
                    final_ar=true;
                System.out.println(datos_entrada[contador]);

                contador++;
            }
            System.out.println("El número de bytes " + contador);
            int salto_linea=0;
            for(int i=0; i< 20096; i++) {
                System.out.print(datos_entrada[i]+ "  ");
                salto_linea++;
                if (salto_linea==25) {
                    System.out.println();
                    salto_linea=0;
                }
            }
            archivo_lectura.close();
        }catch(IOException e){}
    }
}
```


CURSO JAVA
155
SECUENCIAS (STREAMS)
Manejo de archivos. Streams Byte
eclipse
Java

Stream V. Escribiendo archivos Stream Byte II. (Vídeo 156)

El objetivo de este capítulo consiste en utilizar la array del ejemplo anterior que almacenaba todos los bytes y poder realizar una copia de la imagen castillo.jpg.

```
1  package leyendo_escribiendo_byte;
2  import java.io.*;
3  public class Lectura_Escritura {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int contador=0;
8          int datos_entrada[]=new int[20096];
9          try {
10             FileInputStream archivo_lectura=new
11                     FileInputStream("D:/Curso de Java/castillo.jpg");
12             boolean final_ar=false;
13             while(!final_ar) {
14                 int byte_entrada=archivo_lectura.read();
15                 if(byte_entrada!=-1)
16                     datos_entrada[contador]=byte_entrada;
17                 else
18                     final_ar=true;
19                 System.out.println(datos_entrada[contador]);
20
21                 contador++;
22             }
23             System.out.println("El número de bytes " + contador);
24
25             archivo_lectura.close();
26         }catch(IOException e){
27             System.out.println("Error al acceder a la imagen.");
28         }
29
30         crea_fichero(datos_entrada);
31     }
32     static void crea_fichero(int datos_nuevo_fichero[]) {
33         try {
34
35             FileOutputStream fichero_nuevo = new FileOutputStream
36                     ("D:/Curso de Java/castillo_copia.jpg");
37             for(int i=0;i<datos_nuevo_fichero.length;i++) {
38                 fichero_nuevo.write(datos_nuevo_fichero[i]);
39             }
40             fichero_nuevo.close();
41         }catch(IOException e) {
42             System.out.println("Error al crear el archivo " + "castillo_copia.jpg");
43         }
44
45     }
46 }
```

Después de ejecutar iremos a la carpeta donde se tenía que hacer la copia.

castillo.jpg

castillo_copia.jpg

Creamos el método crea_fichero con un parámetro de tipo array.

Desde la clase principal llamamos a método pasando el parámetro de la array.


CURSO JAVA
156
SECUENCIAS (STREAMS)
Manejo de archivos. Streams Byte II
eclipse
Java

Serialización. (Vídeo 157)

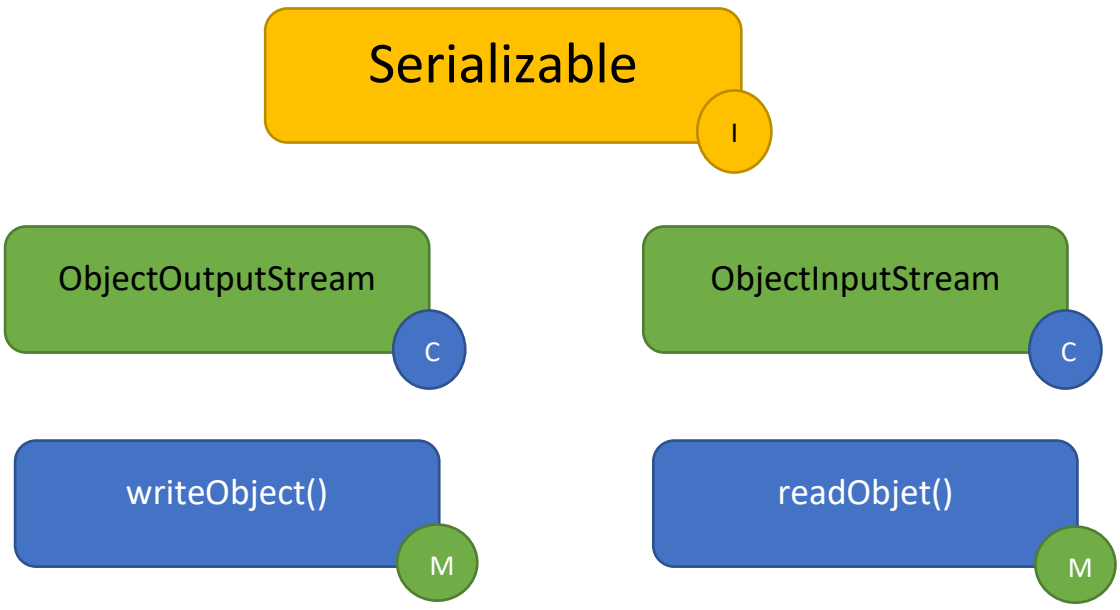
¿Qué es la serialización?

Consiste en convertir un objeto que podemos haber creado nosotros dentro de un programa de java en una sucesión de bytes.

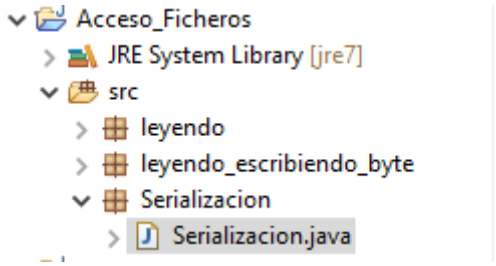
¿Con qué objetivo? Poder almacenar este objeto en un medio de almacenamiento como un disco duro, pendrive, etc. y en el futuro restaurar o recomponer este objeto en el estado que se encontraba cuando lo serializamos que significa convertir en bytes.

Y lo más importante este objeto que hemos convertido en una serie de bytes distribuirlo a través de la red a ordenadores remotos y en estos ordenadores remotos sean restablecidos en el estado que se encontraban cuando se serializó.

API necesaria para serializar.



Vamos a crear un nuevo paquete en Acceso_Ficheros llamado serialización y dentro de el una clase principal llamada Serializando.



Vamos a escribir el siguiente código:

```
package Serializacion;
import java.util.*;
import java.io.*;
public class Serializacion {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Administrador jefe=new Administrador("Juan", 80000, 2005,12,15);
		jefe.setIncentivo(5000);

		Empleado[] persona=new Empleado[3];
		persona[0]=jefe;
		persona[1]=new Empleado("Ana", 25000, 2008, 10, 1);
		persona[2]=new Empleado("Lus", 18000, 2012, 9, 15);
	}
}

class Empleado{
	public Empleado(String n, double s, int agno, int mes, int dia) {
		nombre=n;
		sueldo=s;
		GregorianCalendar calendario=new GregorianCalendar(agno, mes,
dia);
		fechaContrato=calendario.getTime();
	}
	public String getNombre() {
		return nombre;
	}

	public double getSueldo() {
		return sueldo;
	}

	public Date getFechaContrato() {
		return fechaContrato;
	}

	public void subirSueldo(double porcentaje) {
		double aumento=sueldo*porcentaje/100;
		sueldo+=aumento;
	}

	public String toString() {
		return "Nombre=" + nombre + " sueldo=" + sueldo + "fecha de
contrato=" + fechaContrato;
	}
	private String nombre;
	private double sueldo;
```

```
        private Date fechaContrato;
}

class Administrador extends Empleado{
        public Administrador(String n, double s, int agno, int mes, int dia) {
                super(n,s,agno, mes, dia);
                incentivo=0;

        }

        public double getSueldo() {
                double sueldoBase=super.getSueldo();
                return sueldoBase + incentivo;
        }

        public String toString() {
                return super.toString() + "Incentivo=" + incentivo;
        }

        public void setIncentivo(double b) {
                incentivo=b;
        }
        private double incentivo;
}
```

```
class Empleado implements Serializable{
    public Empleado(String n, double s, int agno, int mes, int dia) {
        nombre=n;
        sueldo=s;
        GregorianCalendar calendario=new GregorianCalendar(agno, mes, dia);
        fechaContrato=calendario.getTime();

    }
    public String getNombre() {
        return nombre;
    }

    public double getSueldo() {
        return sueldo;
    }

    public Date getFechaContrato() {
        return fechaContrato;
    }

    public void subirSueldo(double porcentaje) {
        double aumento=sueldo*porcentaje/100;
        sueldo+=aumento;
    }

    public String toString() {
        return "Nombre=" + nombre + " sueldo=" + sueldo + "fecha de contrato=" + fechaContrato;
    }
    private String nombre;
    private double sueldo;
    private Date fechaContrato;
}
```

Implementamos la clase empleados añadiendo “implements Serializable”, para que se puedan convertir en bytes.

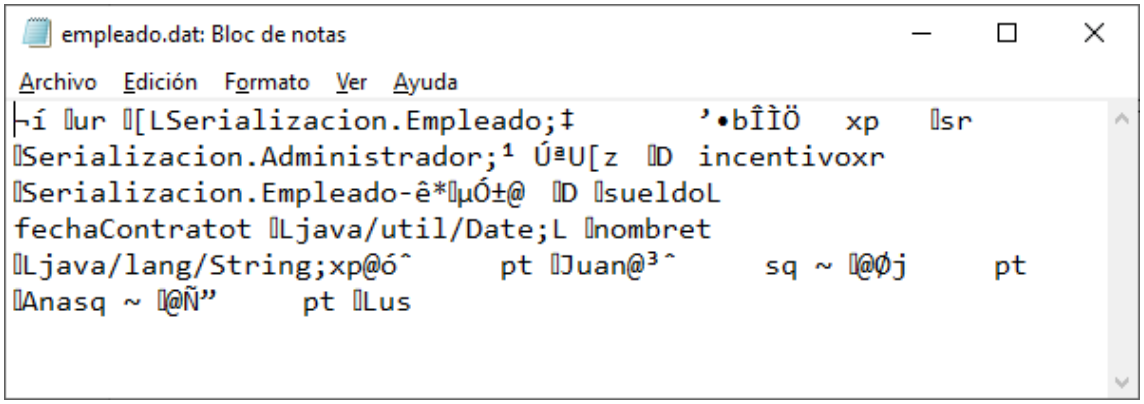
```
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Administrador jefe=new Administrador("Juan", 80000, 2005,12,15);
        jefe.setIncentivo(5000);

        Empleado[] persona=new Empleado[3];
        persona[0]=jefe;
        persona[1]=new Empleado("Ana", 25000, 2008, 10, 1);
        persona[2]=new Empleado("Lus", 18000, 2012, 9, 15);

        try {
            ObjectOutputStream escribiendo_fichero =
                    new ObjectOutputStream(new FileOutputStream("D:/Curso de Java/empleado.dat"));
            escribiendo_fichero.writeObject(persona);
            escribiendo_fichero.close();
        }catch(Exception e) {

        }
    }
}
```

En la clase principal agregamos el siguiente código:

Vamos a ejecutar el código.

empleado.dat

A creado el correspondiente archivo, si lo abrimos veremos su contenido:



Vemos que el archivo es ilegible, ahora vamos a ver como rescatar este objeto y poderlo imprimir en consola.

```
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Administrador jefe=new Administrador("Juan", 80000, 2005,12,15);
        jefe.setIncentivo(5000);

        Empleado[] persona=new Empleado[3];
        persona[0]=jefe;
        persona[1]=new Empleado("Ana", 25000, 2008, 10, 1);
        persona[2]=new Empleado("Lus", 18000, 2012, 9, 15);

        try {
            ObjectOutputStream escribiendo_fichero =
                    new ObjectOutputStream(new FileOutputStream("D:/Curso de Java/empleado.dat"));
            escribiendo_fichero.writeObject(persona);
            escribiendo_fichero.close();

            ObjectInputStream recuperando_fichero=
                    new ObjectInputStream(new FileInputStream("D:/Curso de Java/empleado.dat"));

            Empleado[] personal_recuperado=(Empleado[])recuperando_fichero.readObject();
            recuperando_fichero.close();
            for(int i=0; i<personal_recuperado.length;i++) {
                System.out.println(personal_recuperado[i]);
            }
        }catch(Exception e) {

        }
    }
}
```

En la función principal agregamos el siguiente código que leerá el fichero empleado.dat y nos lo imprimirá en consola.

```
Nombre=Juan sueldo=80000.0fecha de contrato=Sun Jan 15 00:00:00 CET 2006Incentivo=5000.0
Nombre=Ana sueldo=25000.0fecha de contrato=Sat Nov 01 00:00:00 CET 2008
Nombre=Lus sueldo=18000.0fecha de contrato=Mon Oct 15 00:00:00 CEST 2012
```

El bucle for podría ser de la siguiente forma:

For(Empleado e: peronal_recuperdo){

System.out.println(e);

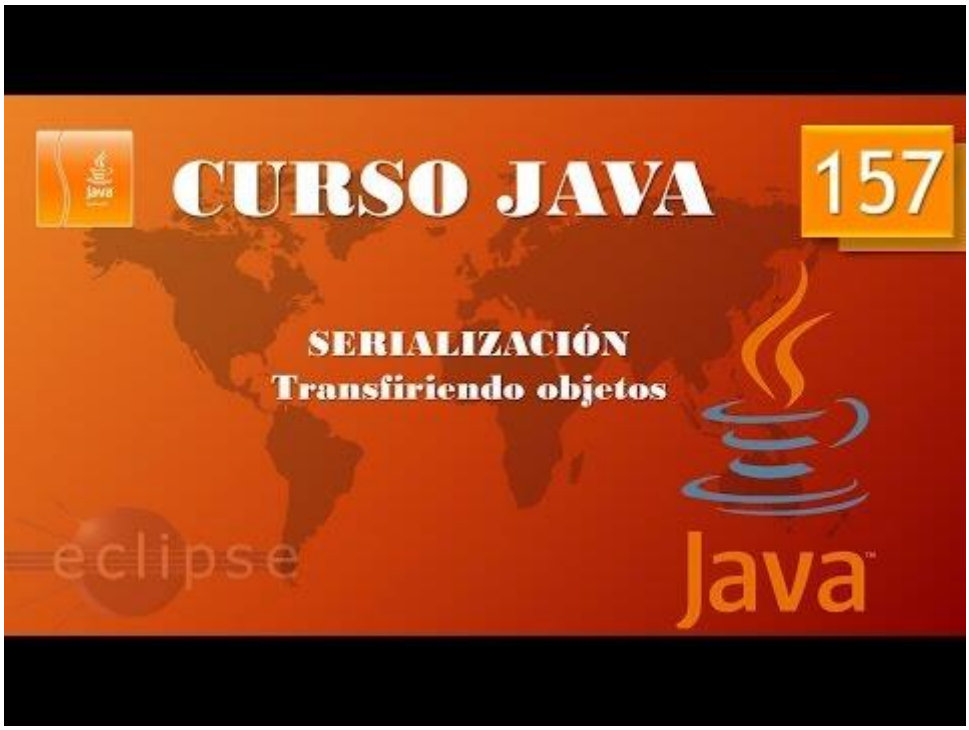
}

El código final será:

```
package Serializacion;
import java.util.*;
import java.io.*;
public class Serializacion {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Administrador jefe=new Administrador("Juan", 80000, 2005,12,15);
		jefe.setIncentivo(5000);

		Empleado[] persona=new Empleado[3];
		persona[0]=jefe;
		persona[1]=new Empleado("Ana", 25000, 2008, 10, 1);
		persona[2]=new Empleado("Lus", 18000, 2012, 9, 15);

		try {
			ObjectOutputStream escribiendo_fichero =
                                new ObjectOutputStream(new
FileOutputStream("D:/Curso de Java/empleado.dat"));
                        escribiendo_fichero.writeObject(persona);
```

```
                    escribiendo_fichero.close();

                    ObjectInputStream recuperando_fichero=
                                        new ObjectInputStream(new
FileInputStream("D:/Curso de Java/empleado.dat"));

                    Empleado[]
personal_recuperado=(Empleado[])recuperando_fichero.readObject();
                    recuperando_fichero.close();
                    for(Empleado e: personal_recuperado) {
                          System.out.println(e);
                    }
             }catch(Exception e) {

             }
       }
}
//-----------------------------------------------------------
class Empleado implements Serializable{
       public Empleado(String n, double s, int agno, int mes, int dia) {
             nombre=n;
             sueldo=s;
             GregorianCalendar calendario=new GregorianCalendar(agno, mes,
dia);
             fechaContrato=calendario.getTime();
       }
       public String getNombre() {
             return nombre;
       }

       public double getSueldo() {
             return sueldo;
       }

       public Date getFechaContrato() {
             return fechaContrato;
       }

       public void subirSueldo(double porcentaje) {
             double aumento=sueldo*porcentaje/100;
             sueldo+=aumento;
       }

       public String toString() {
             return "Nombre=" + nombre + " sueldo=" + sueldo + "fecha de
contrato=" + fechaContrato;
       }
       private String nombre;
       private double sueldo;
       private Date fechaContrato;
}
//-----------------------------------------------------------------
class Administrador extends Empleado{
       public Administrador(String n, double s, int agno, int mes, int dia) {
             super(n,s,agno, mes, dia);
             incentivo=0;

       }
```

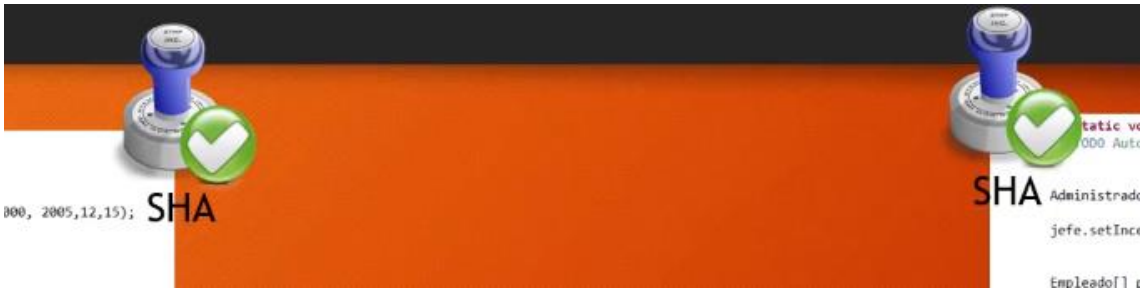
```
	public double getSueldo() {
		double sueldoBase=super.getSueldo();
		return sueldoBase + incentivo;
	}

	public String toString() {
		return super.toString() + "Incentivo=" + incentivo;
	}

	public void setIncentivo(double b) {
		incentivo=b;
	}
	private double incentivo;
}
```



Serialización II. SerialIVersionUID. (Vídeo 158)

Conflicto versiones serialización.

Deben de tener la misma versión del programa java.



Cuando nosotros creamos un programa java, aunque nosotros no lo vemos el programa java tiene une huella que podemos asemejarlo a un número identificativo único para este programa java que acabamos de crear.

Ese número se denomina SHA.

Esta huella la realiza el compilador de java automáticamente.

Es una sucesión de 20 bytes que se denomina serialVersionUID.

Si el emisor quiere serializar un objeto para enviárselo al receptor ambos tienen que tener una copia idéntica del programa, es decir una copia con la misma serialVersionUID.

Si es así se serializa el objeto y se envía al receptor.

Si actualizamos el programa y este no lo distribuimos con todos los receptores que van a recibir un objeto serializado nos vamos a encontrar con un problema, ¿Por qué? Cuando un programa cambia o bien modificando un método o bien quitando o agregando campos dentro de esta clase o cualquier otro cambio también cambia la huella, el serialVersionUID.

Por lo cual la versión del programa el emisor y del receptor no será la misma.

¿Qué ocurre cuando el serialVersionUID es diferente? El emisor serializará el objeto, intentará enviarlo, pero el receptor no lo podrá leer.

Java hace una comprobación muy estricta a la hora de serializar objetos. Que el objeto serializado tiene la misma versión que el programa que va a tratar de leer el objeto.

Vamos a realizar una práctica, utilizando el proyecto del capítulo anterior llamado Serializacion.

Antes de empezar iremos a la carpeta donde se creó el archivo empleado.dat y lo eliminamos.

Ahora lo ejecutamos de nuevo y este creará de nuevo el archivo.

```
6      public static void main(String[] args) {
7          // TODO Auto-generated method stub
8          Administrador jefe=new Administrador("Juan", 80000, 2005,12,15);
9          jefe.setIncentivo(5000);
10
11         Empleado[] persona=new Empleado[3];
12         persona[0]=jefe;
13         persona[1]=new Empleado("Ana", 25000, 2008, 10, 1);
14         persona[2]=new Empleado("Lus", 18000, 2012, 9, 15);
15
16         try {
17             //ObjectOutputStream escribiendo_fichero =
18             //      new ObjectOutputStream(new FileOutputStream("D:/Curso de Java/empleado.dat"));
19             //escribiendo_fichero.writeObject(persona);
20             //escribiendo_fichero.close();
21
22             ObjectInputStream recuperando_fichero=
23                     new ObjectInputStream(new FileInputStream("D:/Curso de Java/empleado.dat"));
24
25             Empleado[] personal_recuperado=(Empleado[])recuperando_fichero.readObject();
26             recuperando_fichero.close();
27             for(Empleado e: personal_recuperado) {
28                 System.out.println(e);
29             }
30         }catch(Exception e) {
31
32         }
33     }
34 }
```

Ahora comentamos la parte de creación de archivo y solo dejamos la parte de recuperación del archivo.

Dejamos las líneas encargadas de leer el objeto e imprimirlo en pantalla.

Borramos la consola.

Ejecutamos el programa.

```
Nombre=Juan sueldo=80000.0fecha de contrato=Sun Jan 15 00:00:00 CET 2006Incentivo=5000.0
Nombre=Ana sueldo=25000.0fecha de contrato=Sat Nov 01 00:00:00 CET 2008
Nombre=Lus sueldo=18000.0fecha de contrato=Mon Oct 15 00:00:00 CEST 2012
```

Podemos ver el contenido en la consola

A continuación quitamos las marcas de comentarios.

Reemplazamos el campo sueldo por sueldos en todo el programa.

Borramos de nuevo el archivo empleado.dat.

Ejecutamos la aplicación de nuevo, el programa funciona con normalidad y crea el archivo de nuevo.

Volvemos a comentar las cuatro líneas, según muestra imagen anterior.

Y reemplazamos el campo sueldos como sueldo (versión anterior).

Borramos el contenido de la consola.

Le damos al play.

Y vemos que en consola no se imprime absolutamente nada.

Que está pasando, pues que el receptor tienen una versión antigua del programa y no es capaz de leer el objeto.

Y eso que el objeto está donde tiene que estar.

empleado.dat

Si hacemos una actualización del programa y no se envía a todos los receptores, esos receptores no podrán leer o recibir los objetos serializados.

¿Cuál es la solución?

La solución consiste en declarar nuestro propio Id, lo podemos genera de forma manual.

```
private static final long serialVersionUID = 1L;
```

Si ponemos manualmente esta versión y el programa se actualiza como la huella no es generada automáticamente, no cambia.

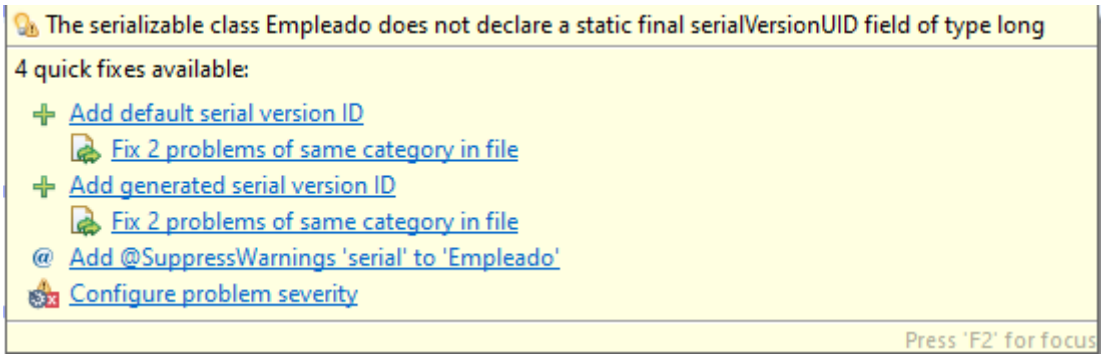
A pesar de realizar cambios podremos serializar objetos e enviándolo por la red y que el receptor sea capaz de leerlo.

```
36 class Empleado implements Serializable{
37     public Empleado(String n, double s, int agno, int mes, int dia) {
38         nombre=n;
39         sueldo=s;
40         GregorianCalendar calendario=new GregorianCalendar(agno, mes, dia);
41         fechaContrato=calendario.getTime();
42     }
```

Esta clase no nuestra una advertencia y nos está diciendo que la clase serializable Empleado no declara la contante extática serialVersiónUID de tipo long.

No es algo que impida el funcionamiento del programa pero si que es una advertencia que nos lanza Eclipse.

Si nos colocamos en el nombre Empleado que tiene in subrayado amarillo observaremos e siguiente mensaje:



Podemos que Eclipse genere la versión automáticamente.

Encontramos dos tipos

Add default serial version ID: Generar la versión de forma manual.

Add generated serial version ID: Generar, declarar constante de clase con el valor que el compilador java da por defecto.

Vamos a seleccionar la opción “Add default serial versión ID”.

```
36  class Empleado implements Serializable{
37      /**
38       *
39       */
40      private static final long serialVersionUID = 1L;
41      public Empleado(String n, double s, int agno, int mes, int dia) {
42          nombre=n;
43          sueldo=s;
44          GregorianCalendar calendario=new GregorianCalendar(agno, mes, dia);
45          fechaContrato=calendario.getTime();
```

Podrás observar que el mensaje que aparecía en la class Empleado ha desaparecido.

Lo vamos a repetir en la clase Administrador.

```
72  class Administrador extends Empleado{
73      /**
74       *
75       */
76      private static final long serialVersionUID = 1L;
77      public Administrador(String n, double s, int agno, int mes, int dia) {
78          super(n,s,agno, mes, dia);
79          incentivo=0;
```

Si seleccionamos la segunda opción “Add generated serial versión ID”.

```
36  class Empleado implements Serializable{
37      /**
38       *
39       */
40      private static final long serialVersionUID = -5914868870684626624L;
41      public Empleado(String n, double s, int agno, int mes, int dia) {
42          nombre=n;
43          sueldo=s;
44          GregorianCalendar calendario=new GregorianCalendar(agno, mes, dia);
45          fechaContrato=calendario.getTime();
```

Lo borramos y volvemos a dejar con el de default.

```
41      private static final long serialVersionUID = 1L;
42      public Empleado(String n, double s, int agno, int mes, int dia) {
43          nombre=n;
44          sueldo=s;
45          GregorianCalendar calendario=new GregorianCalendar(agno, mes, dia);
46          fechaContrato=calendario.getTime();
```

```
73  class Administrador extends Empleado{
74
75      /**
76       *
77       */
78      private static final long serialVersionUID = 1L;
79      public Administrador(String n, double s, int agno, int mes, int dia) {
80          super(n,s,agno, mes, dia);
81          incentivo=0;
```

En las dos clases.

Esto implica que incluso haciendo cambios podremos serializar.

Repetimos el proceso.

Eliminamos el archivo empleado.dat.

```
16          try {
17              ObjectOutputStream escribiendo_fichero =
18                      new ObjectOutputStream(new FileOutputStream("D:/Curso de Java/empleado.dat"));
19              escribiendo_fichero.writeObject(persona);
20              escribiendo_fichero.close();
21
22              ObjectInputStream recuperando_fichero=
23                      new ObjectInputStream(new FileInputStream("D:/Curso de Java/empleado.dat"));
24
25              Empleado[] personal_recuperado=(Empleado[])recuperando_fichero.readObject();
26              recuperando_fichero.close();
27              for(Empleado e: personal_recuperado) {
28                  System.out.println(e);
29              }
30          }catch(Exception e) {
31
32          }
```

Quitamos los comentarios.

Ahora somos el emisor y ejecutamos.

Si vamos a la carpeta veremos que el archivo se ha creado.

Ahora somos el receptor.

Limpiamos la consola.

Cambiamos el campo sueldo por sueldos en todos los lugares de la aplicación.

Ahora pulsamos el play.

Este será el resultado:

```
Nombre=Juan sueldo=0.0fecha de contrato=Sun Jan 15 00:00:00 CET 2006Incentivo=5000.0
Nombre=Ana sueldo=0.0fecha de contrato=Sat Nov 01 00:00:00 CET 2008
Nombre=Lus sueldo=0.0fecha de contrato=Mon Oct 15 00:00:00 CEST 2012
```

El programa funciona correctamente porque la versión Id al ser manual no cambia, aunque realicemos modificaciones en el programa.

La huella no ha cambiado.

Si el receptor cambia la versión el objeto no podrá ser leído.

```
36 class Empleado implements Serializable{
37
38     /**
39      *
40      */
41     private static final long serialVersionUID = 2L;
42     public Empleado(String n, double s, int agno, int mes, int dia) {
43         nombre=n;
44         sueldos=s;
45         GregorianCalendar calendario=new GregorianCalendar(agno, mes, dia);
```

```
73 class Administrador extends Empleado{
74
75     /**
76      *
77      */
78     private static final long serialVersionUID = 2L;
79     public Administrador(String n, double s, int agno, int mes, int dia) {
80         super(n,s,agno, mes, dia);
81         incentivo=0;
```

Limpiamos la consola y ejecutamos de nuevo.
No puede leer el objeto.
Por último eliminados las versiones en las dos clases.
En la clase Empledo hacemos un “Add generated serial version ID”.

```
36 class Empleado implements Serializable{
37
38     /**
39      *
40      */
41     private static final long serialVersionUID = 6564532313797965464L;
42     public Empleado(String n, double s, int agno, int mes, int dia) {
43         nombre=n;
44         sueldos=s;
45         GregorianCalendar calendario=new GregorianCalendar(agno, mes, dia);
46         fechaContrato=calendario.getTime();
```

Borramos serialVersionUID.
Cambiamos la variable sueldos por sueldo en todo el programa.
Volvemos a generar el número serialVersionUID.

```
36 class Empleado implements Serializable{
37
38     /**
39      *
40      */
41     private static final long serialVersionUID = -5914868870684626624L;
42     public Empleado(String n, double s, int agno, int mes, int dia) {
43         nombre=n;
44         sueldo=s;
45         GregorianCalendar calendario=new GregorianCalendar(agno, mes, dia);
46         fechaContrato=calendario.getTime();
```

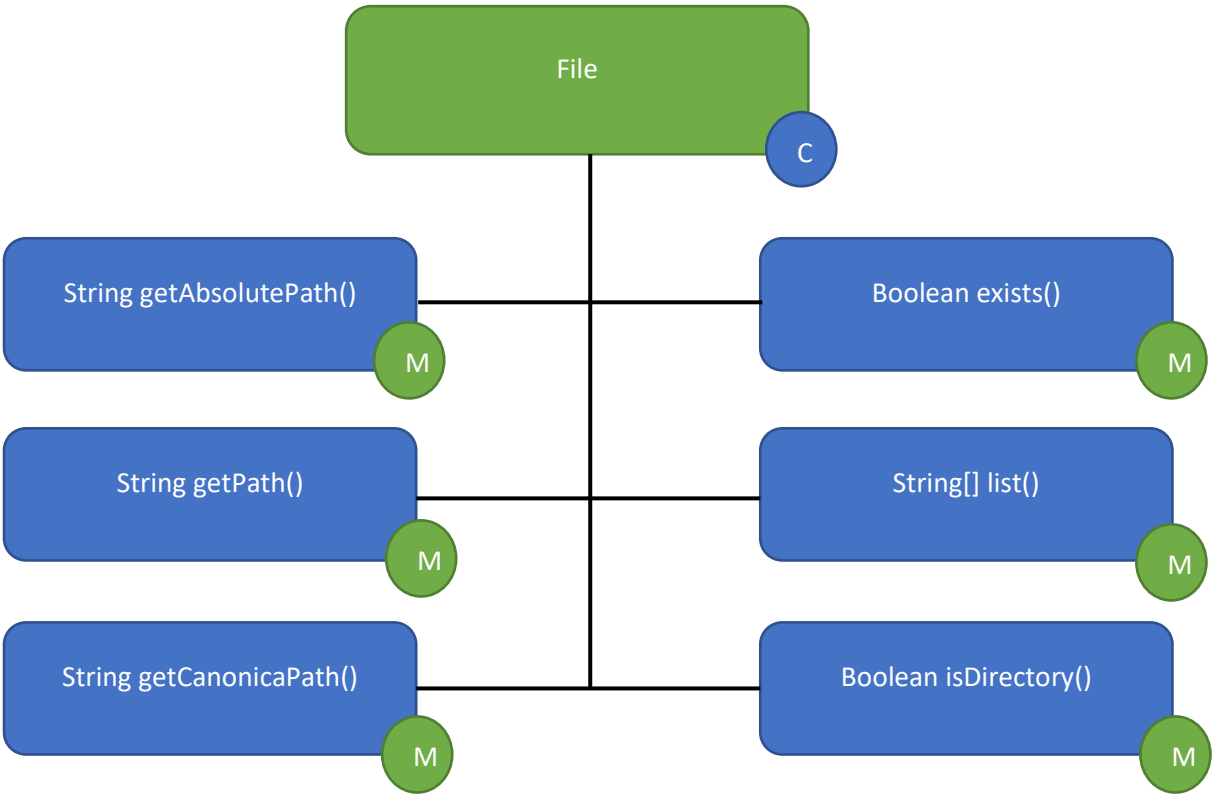
Vemos que a generado un número distinto.

Si borramos de nuevo el serialVersionUID y cambiamos de nuevo la variable sueldo por sueldos, podríamos observar que el numero sería de nuevo el que apareció anteriormente.

Manipulación archivos y directorios. Clase File I. (Vídeo 159)

Clase File

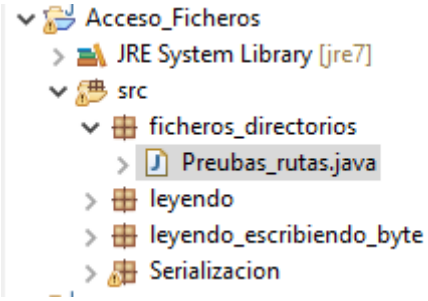
<



En el proyecto llamado acceso_Ficheros.

Vamos a crear un paquete llamado ficheros_directorios.

Y la clase principal Pruebas_rutas.




```java
package ficheros_directorios;
import java.io.*;
public class Preubas_rutas {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        File archivo=new File("D:/curso de Java/archivo.txt ");
        System.out.println(archivo.getAbsolutePath());
    }
}
```

Definimos un archivo en una ruta determinada, aunque el archivo no existe en dicha ruta.

Ejecutamos.

```
D:\curso de Java\archivo.txt
```

getAbsolutePath() muestra igualmente la ruta que le hemo definido, así como el archivo.

```
1  package ficheros_directorios;
2  import java.io.*;
3  public class Preubas_rutas {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          File archivo=new File("D:/curso de Java/archivo.txt ");
8          System.out.println(archivo.getAbsolutePath());
9          System.out.println(archivo.exists());
10     }
11 }
```

Exists() nos dice si el archivo existe o no, devuelve un valor booleano true o false.

Este será el resultado:

```
D:\curso de Java\archivo.txt
false
```

```
1  package ficheros_directorios;
2  import java.io.*;
3  public class Preubas_rutas {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          File archivo=new File("bin");
8          System.out.println(archivo.getAbsolutePath());
9          System.out.println(archivo.exists());
10     }
11 }
```

Queremos saber si en el directorio actual hay una carpeta llamada bin, nos muestre su ruta absoluta y si existe o no.

Ejecutamos:

```
D:\Curso de Java\Acceso_Ficheros\bin
true
```

¿Cómo podemos decirle a un programa que liste con todos los archivos que se encuentren en un directorio?

Vamos a crear otra clase principal llamada Acceso_Ficheros.

```
1  package ficheros_directorios;
2  import java.io.*;
3  public class Acceso_Ficheros {
4
```

```
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          File ruta=new File("D:/Curso de Java");
8          System.out.println(ruta.getAbsolutePath());
9          String[]datos=ruta.list();
10         for(int i=0; i<datos.length;i++) {
11             System.out.println(datos[i]);
12         }
13     }
14 }
```

En la línea 7 le decimos la carpeta que queremos ir.

En la línea 8 imprime la ruta absoluta.

En la línea 9 pasamos el contenido de archivos y directorios a la array datos.

En la línea 10 a la 12 hacemos un recorrido en bucle para imprimir los archivos y directorios de dicha carpeta.

Este será el resultado:

```
D:\Curso de Java
.metadata
Acceso_Ficheros
Aplicaciones
Calculadora
Calculadora_Ejecutable
castillo.jpg
castillo_copia.jpg
ejemplo.txt
empleado.dat
Escribir.txt
Excepciones
JAR_firmado
languageServers-log
leyendo_escribiendo_byte
PrimerosPasos
```

¿Cómo podemos ver el contenido de una de las carpetas, sin modificar la ruta.

```
1  package ficheros_directorios;
2  import java.io.*;
3  public class Acceso_Ficheros {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          File ruta=new File("D:/Curso de Java");
8          System.out.println(ruta.getAbsolutePath());
9          String[]datos=ruta.list();
10         for(int i=0; i<datos.length;i++) {
11             System.out.println(datos[i]);
12             File f=new File(ruta.getAbsolutePath(),datos[i]);
13             if(f.isDirectory()) {
14                 String[] archivos_subcarpetas= f.list();
15                 for(int j=0; j<archivos_subcarpetas.length; j++) {
16                     System.out.println(archivos_subcarpetas[j]);
```

```
17                  }
18              }
19          }
20      }
21  }
```

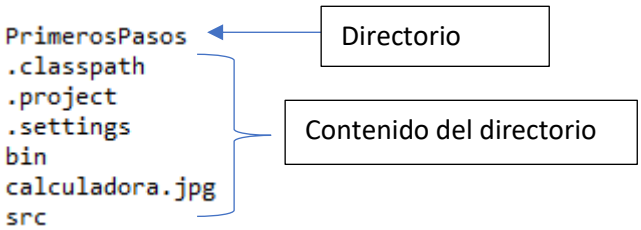
En la línea 12 definimos una variable llamada f de tipo File que dentro del bucle va leyendo los archivos y directorios que encuentra.

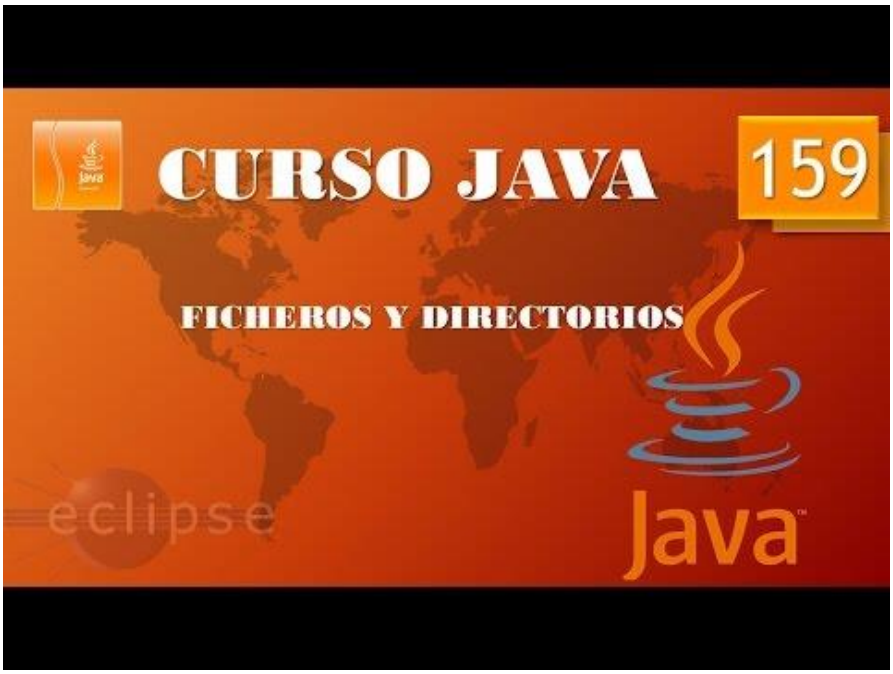
En la línea 13 con un condicional comprobamos si es un directorio, si es así declaramos un array de tipo string llamado archivos_subcarpetasy le pasamos su ruta.

En la línea 15 un nuevo bucle for que recorre la longitud de la array archivos_subcarpetas.

Desde la línea 16 va imprimiendo el contenido del directorio.

Este será el resultado:





Manipulación archivos y directorios. Clase File II. (Vídeo 160)

Creación, escritura y eliminación.

Cuando trabajamos con Windows la ruta de entre varios directorios se separa con la barra:

```
File ruta=new File("D:/Curso de Java");
```

Pero esta "/" en otros sistemas operativos pueden ser "\" y como sabemos que este lenguaje es multiplataforma que puede funcionar con varios sistemas operativos y que no tengamos problemas a la hora de especificar una ruta.

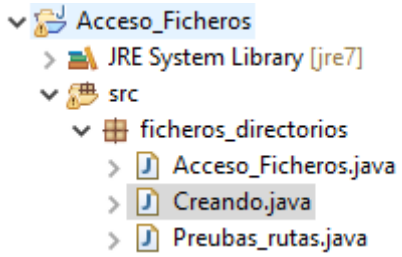
Para realizarlo en la ruta especificada anteriormente sería de la siguiente forma:

```
File ruta=new File("D:" + File.separator +"Curso de Java");
```

Lo que hemos realizado es eliminar la barra y concatenar con File.separator.

Ejecutamos el programa y este funciona correctamente.

Para seguir con este capítulo vamos a crear la clase "Creando", dentro del paquete "ficheros_directorios".

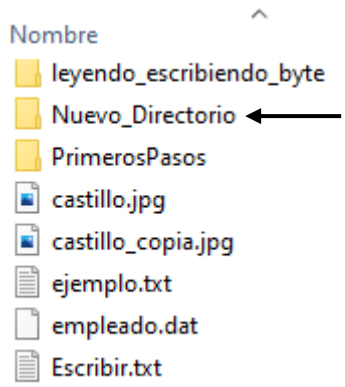



```
1  package ficheros_directorios;
2  import java.io.*;
3  public class Creando {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          File ruta=new File("D:" + File.separator + "Curso de Java" +
8                  File.separator + "Nuevo_Directorio");
9          ruta.mkdir();
10
11     }
12 }
```

La línea 7 y 8 es una única línea que para que no se alargue hemos realizado un salto de línea, verás que el final de línea termina con ;

Definimos la ruta y al final el directorio en este caso se llama "Nuevo_Directorio".

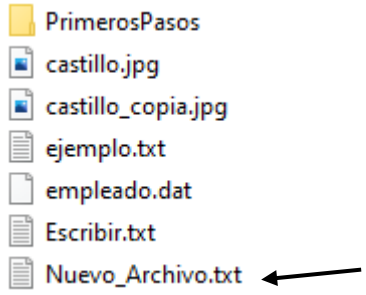
Cuando ejecutemos, observaremos que en la carpeta "Curso de Java" a creado una carpeta o directorio nuevo con este nombre.



Ahora como crear un nuevo archivo.

```
1  package ficheros_directorios;
2  import java.io.*;
3  public class Creando {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          File ruta=new File("D:" + File.separator + "Curso de Java" +
8                  File.separator + "Nuevo_Archivo.txt");
9          try {
10             ruta.createNewFile();
11         } catch (IOException e) {
12             // TODO Auto-generated catch block
13             e.printStackTrace();
14         }
15
16     }
17 }
```

Ahora ejecutamos y miramos en el directorio “Curso de Java”.



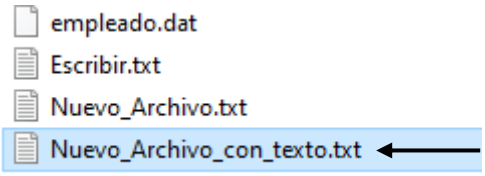
A creado el archivo.

Ahora como crear un archivo y escribir en él.

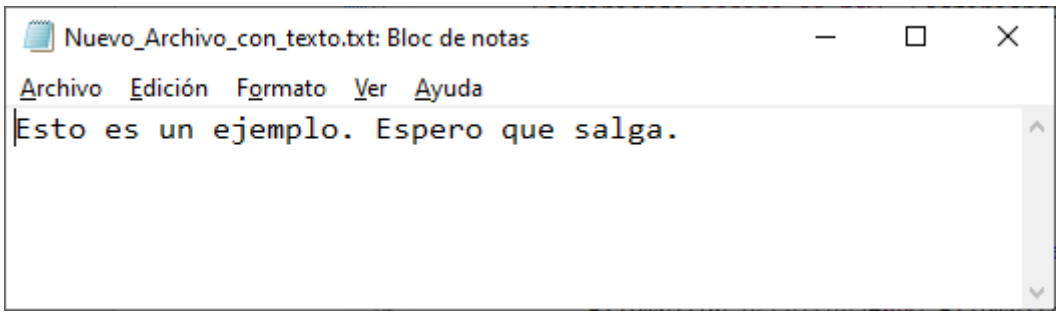
```
1  package ficheros_directorios;
2  import java.io.*;
3  public class Creando {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          File ruta=new File("D:" + File.separator + "Curso de Java" +
8                  File.separator + "Nuevo_Archivo_con_texto.txt");
9          String archivo_destino=ruta.getAbsolutePath();
```

```
10          try {
11              ruta.createNewFile();
12          } catch (IOException e) {
13              // TODO Auto-generated catch block
14              e.printStackTrace();
15          }
16          Escribiendo accede_es=new  Escribiendo();
17
18          accede_es.escribir(archivo_destino);
19      }
20  }
21
22  class Escribiendo{
23      public void escribir(String ruta_archivo) {
24          String frase="Esto es un ejemplo. Espero que salga.";
25          try {
26              FileWriter escritura=new FileWriter(ruta_archivo);
27              for(int i=0; i<frase.length();i++){
28                  escritura.write(frase.charAt(i));
29              }
30              escritura.close();
31          }catch(IOException e) {}
32
33      }
34  }
```

Si ejecutamos vemos que en el directorio destino a creado un archivo.



Si lo abrimos:



Ahora por último vamos a ver como borrar un archivo.

Vamos a crear una nueva clase llamada Eliminar.

```
1  package ficheros_directorios;
2
3  import java.io.File;
4
5  public class Eliminar {
6
```

```
7⊖    public static void main(String[] args) {
8         // TODO Auto-generated method stub
9         File ruta=new File("D:" + File.separator + "Curso de Java" +
10                File.separator + "Escribir.txt");
11        ruta.delete();
12    }
13
14 }
```

Ejecutamos la aplicación y el archivo Escribir.txt ha sido eliminado.

Contenido