

# CURSO INICIACIÓN JAVA I

Tutoriales de pildorasinformaticas

Descripción breve

Curso introductorio de Java por pildorasinformaticas.



Pere Manel Verdugo Zamora  
pereverdugo@gmail.com

## Presentación (Vídeo 1)

¿A quién va dirigido?

- Personas sin conocimientos de programación en general y Java en particular.
- Se utilizará una jerga sencilla y se explicarán los términos más farragosos y complejos.
- Insisto: Los primeros 20 vídeos no son aptos para personas con conocimientos de Java.

Entorno de desarrollo a utilizar en el curso

- Se utilizará Eclipse como entorno de desarrollo. ¿Por qué?
  - Porque es el que más le gusta al profesor.
  - Porque la curva de aprendizaje no es muy pronunciada.
  - Porque es uno de los entornos de desarrollo Java más utilizados hoy en día.
  - Porque es gratuito.
  - Porque hay numerosos foros, documentación online y tutoriales en línea sobre él.
  - El curso no trata sobre el aprendizaje de Eclipse. Se irá aprendiendo sobre la marcha a medida que avance el curso.

Otros entornos de desarrollo

- JDK
- NetBeans
- BlueJ
- JBuilder
- JCreator
- JDevenloper

Temario Parte 1

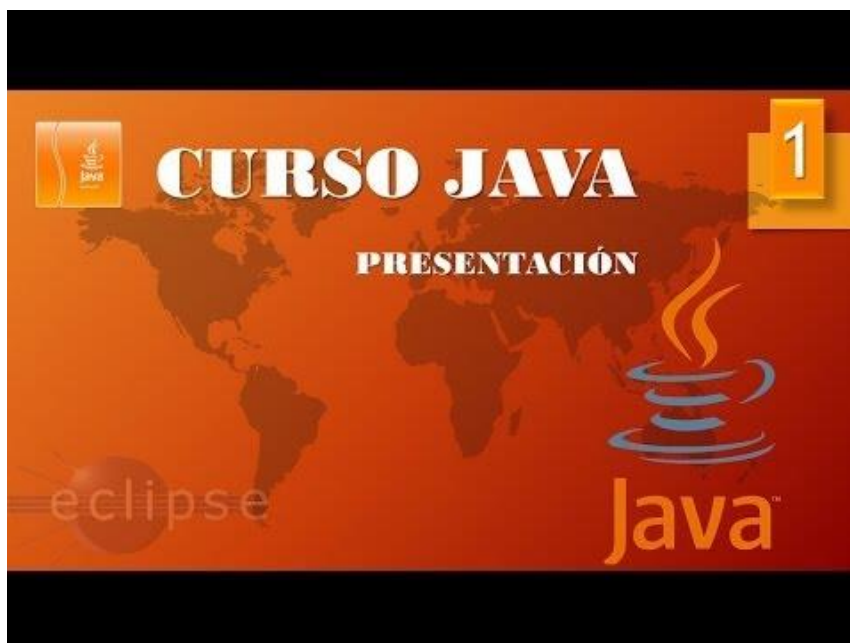
- Descarga e instalación de Eclipse
- Introducción a Jva
- Estructuras principales del lenguaje
- Objetos y clases
- Herencias
- Clases internas e interfaces.
- Programación de gráficos
- Eventos
- Componentes Swing
- Aplicaciones y Applets
- Tratamiento de errores (excepciones) y depuración
- Programación genérica
- Colecciones
- Programación multihilo (multithreading)

## Temario Parte 2

- Programación con archivos
- XML
- Programación para redes
- Programación para BBDD
- Programación cliente-servidor (objetos distribuidos)
- Swing avanzado
- AWT avanzado
- JavaBeans
- Seguridad
- Programación internacional
- Método nativos
- Anotaciones

¿Y todo esto cómo se distribuye?

- Más de 250 vídeos con una duración media de 20 minutos cada uno.
- A lo largo del curso se irán haciendo ejercicios prácticos, muy sencillos al comienzo, más complejos según vayamos avanzando.



## Instalación JRE y Eclipse (Vídeo 2)

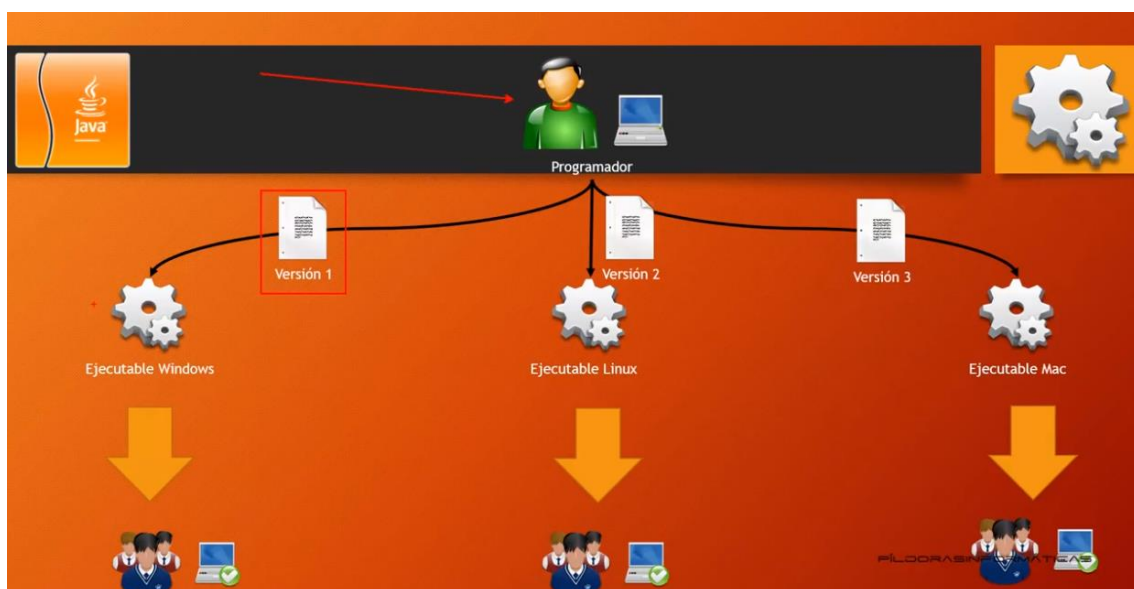
¿Qué es JRE?

- J.R.E = Java Runtime Enviroment (Entorno de ejecución Java), También denominado en sus inicios como J.V.M. = Java Virtual Machine (Máquina Virtual Java).
- ¿Por qué es necesaria su instalación?
  - Por culpa de la principal característica del lenguaje Java: ser MULTIPLATAFORMA.
  - La característica de ser multiplataforma implica que un programa escrito en Java debe ser **compilado** para posteriormente ser **interpretado** por la Máquina Virtual de Java o JRE.

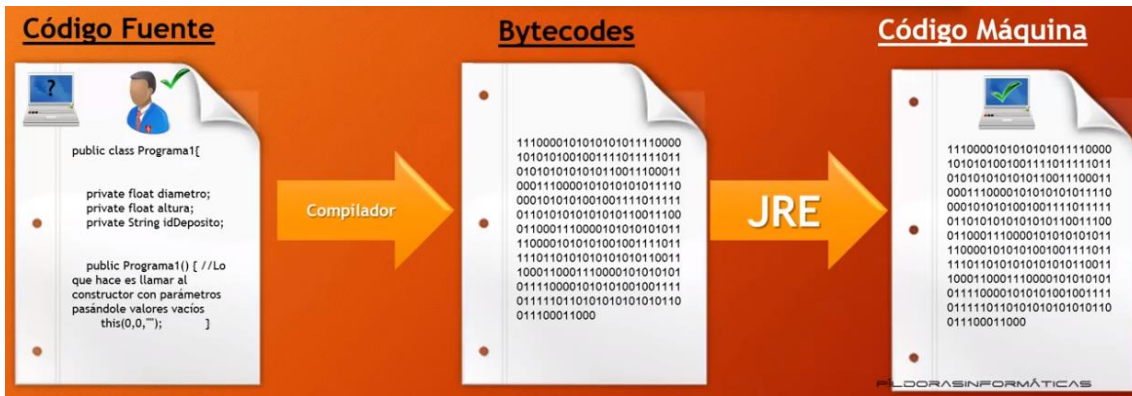
¿Qué es compilar?



Según el sistema operativo que utilizemos utilizaremos distintos compiladores.



Como Java es un lenguaje multiplataforma el proceso es el siguiente:



El fichero que generamos tiene una extensión java, cuando lo compilamos nos genera un fichero intermedio (fichero de Bytecodes), tendrá extensión .class, le aplicaremos la máquina virtual de Java, el objetivo es traducir dicho fichero a código máquina independientemente de la plataforma en la que estamos trabajando, por eso se necesita JRE.



Una vez tengamos el fichero con extensión .class se podrá ejecutar desde cualquier ordenador que tenga instalada la máquina virtual de Java (JRE).

**Write once,  
run everywhere**

Escríbelo una vez y ejecútalo allí donde quiera.

Vamos a descargar la máquina virtual de Java, para ello accederemos al siguiente enlace:

<https://www.java.com/es/>

# JAVA Y TÚ, DESCARGAR HOY

[Descarga gratuita de Java](#)

[» ¿Qué es Java?](#) [» ¿Tengo Java?](#) [» ¿Necesita ayuda?](#)

Acerca de Java (sitio en inglés)



[Seleccionar idioma](#) | [Acerca de Java](#) | [Soporte](#) | [Desarrolladores](#)  
[Privacidad](#) | [Preferencias sobre cookies](#) | [Condiciones de uso](#) | [Marcas registradas](#) | [Descargo de responsabilidad](#)

ORACLE

Seleccionaremos Descarga gratuita de Java.

Recursos de ayuda

- [» ¿Qué es Java?](#)
- [» Eliminar versiones anteriores de Java](#)
- [» Desactivar Java](#)
- [» Mensajes de error](#)
- [» Solucionar problemas de Java](#)
- [» Otra ayuda](#)

Usuarios de Windows de 64 bits

- [¿Utiliza exploradores de 32 y 64 bits?](#)
- [» Preguntas frecuentes sobre Java de 64 bits para Windows](#)

Instalación fuera de línea

- [¿Problemas al descargar? Intente con el instalador fuera de línea](#)

## Descargar Java para Windows

**Recomendado Version 8 Update 261 (Tamaño de archivo: 1.99 MB)**

Fecha de lanzamiento: 14 de julio de 2020

### ⚠ Actualización importante de la licencia de Oracle Java

**La licencia de Oracle Java ha cambiado para las versiones publicadas a partir del 16 de abril de 2019.**

El nuevo [acuerdo de licencia de Oracle Technology Network para Oracle Java SE](#) es sustancialmente diferente a las licencias de Oracle Java anteriores. La nueva licencia permite ciertos usos, como el uso personal y de desarrollo, sin coste alguno (aunque podría haber otros usos autorizados en licencias de Oracle Java anteriores que ya no estén disponibles). Revise las condiciones con atención antes de descargar y utilizar este producto. Puede consultar las preguntas frecuentes [aquí](#).

La licencia comercial y el soporte están disponibles con una [suscripción de Java SE](#) de bajo coste.

Oracle también ofrece la última versión de OpenJDK con la [licencia pública general](#) de código abierto en [jdk.java.net](#).

- ⚠** En Windows 10, el explorador Edge no es compatible con ningún plugin y, por lo tanto, no ejecutará Java. Cambie a un explorador diferente (Internet Explorer, por ejemplo) para ejecutar el plugin de Java. Seleccione la opción Más acciones situada en la parte superior derecha del explorador Edge y haga clic en Abrir con Internet Explorer. [Más información](#)

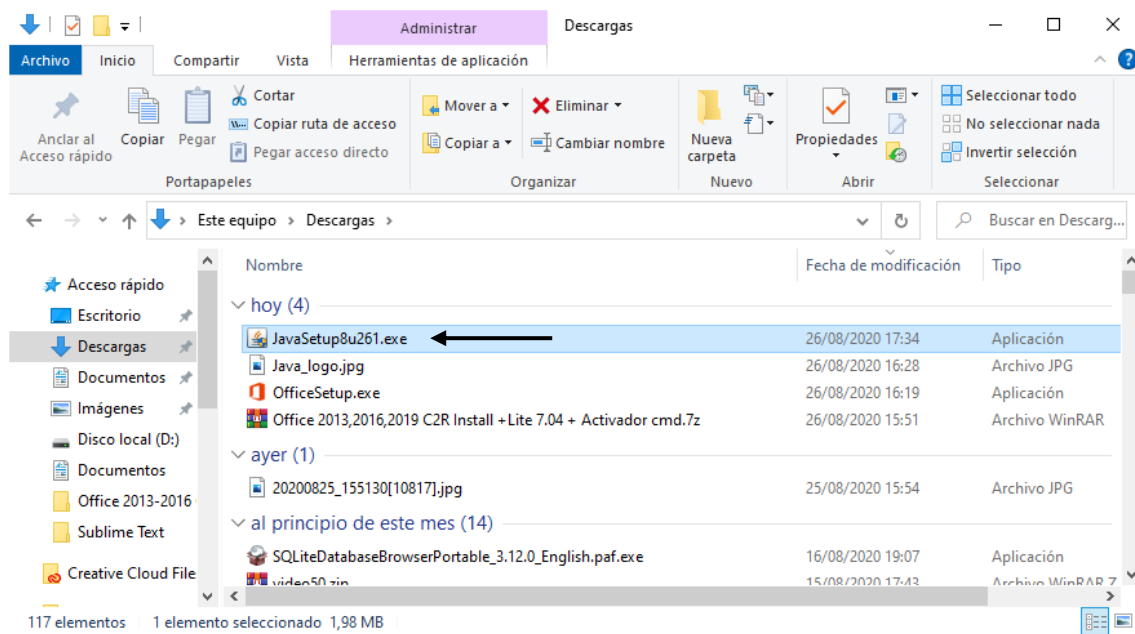
[Aceptar e iniciar descarga gratuita](#)

Al descargar Java, confirma que ha leído y acepta las condiciones del [acuerdo de licencia de Oracle Technology Network para Oracle Java SE](#)

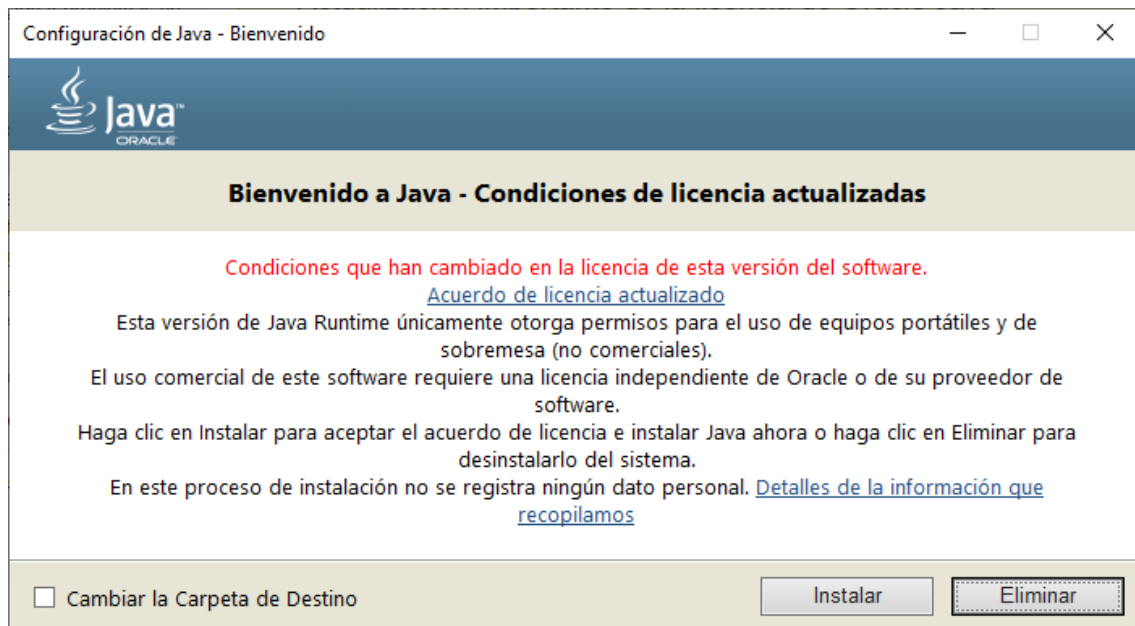
Seleccionaremos el botón Aceptar e iniciar descarga gratuita.



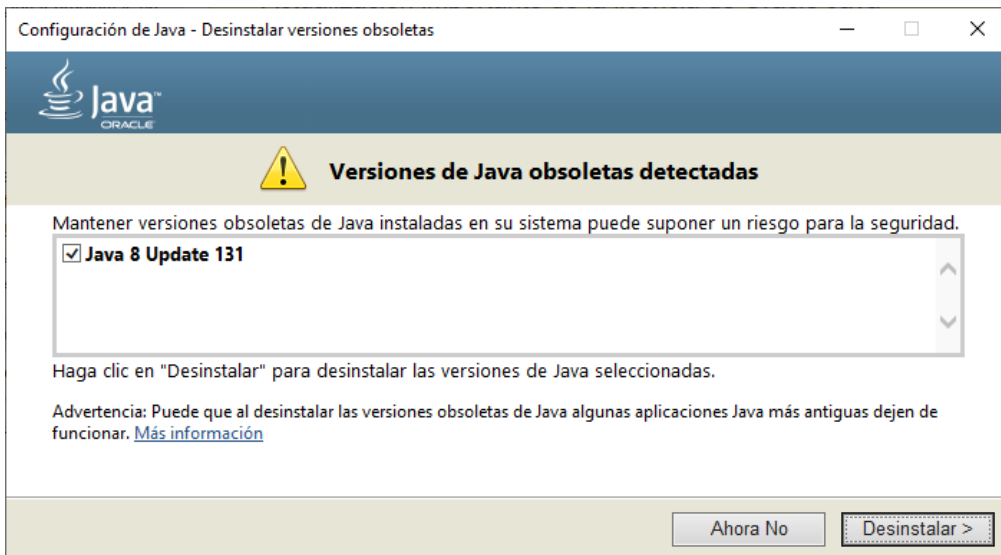
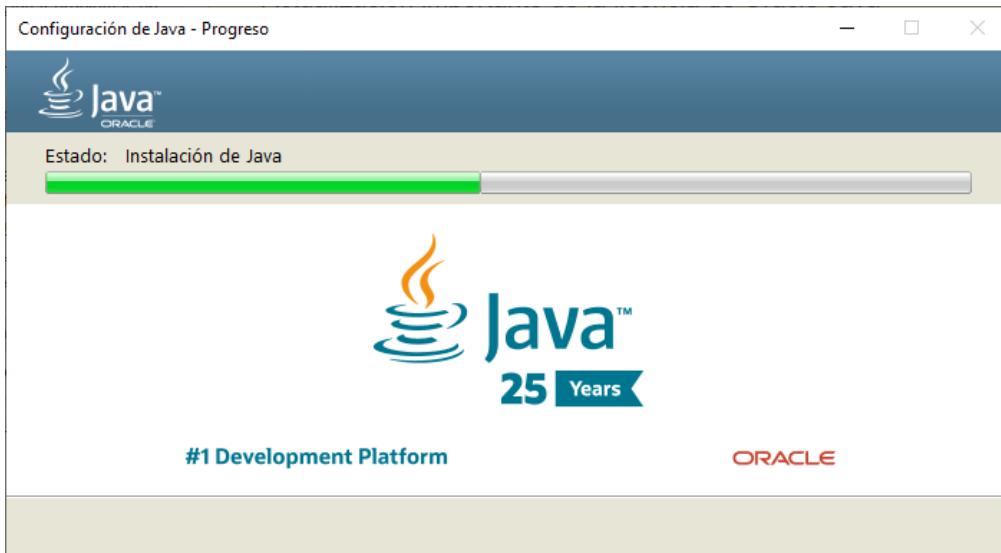
Iremos a descargas.



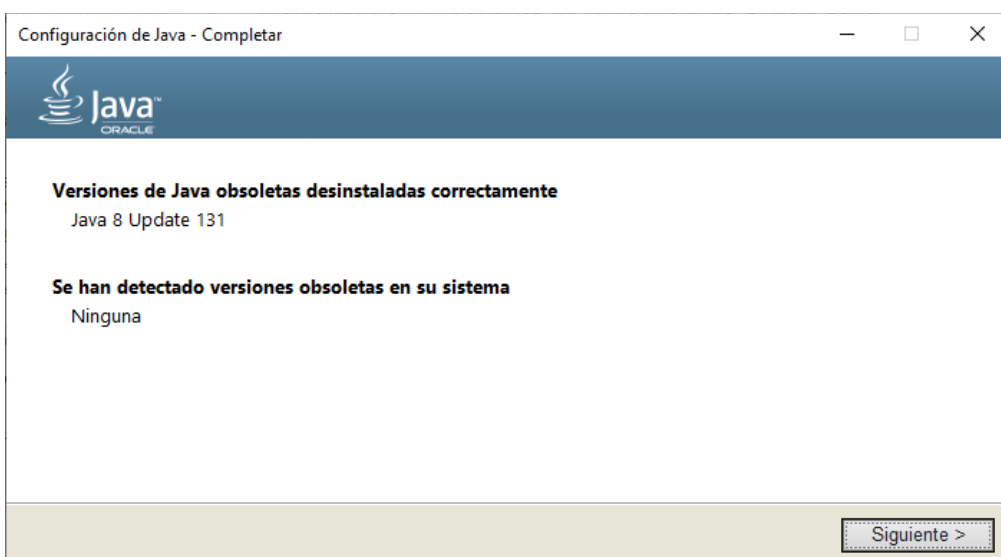
Ejecutaremos para su instalación.



Seleccionaremos el botón Instalar.

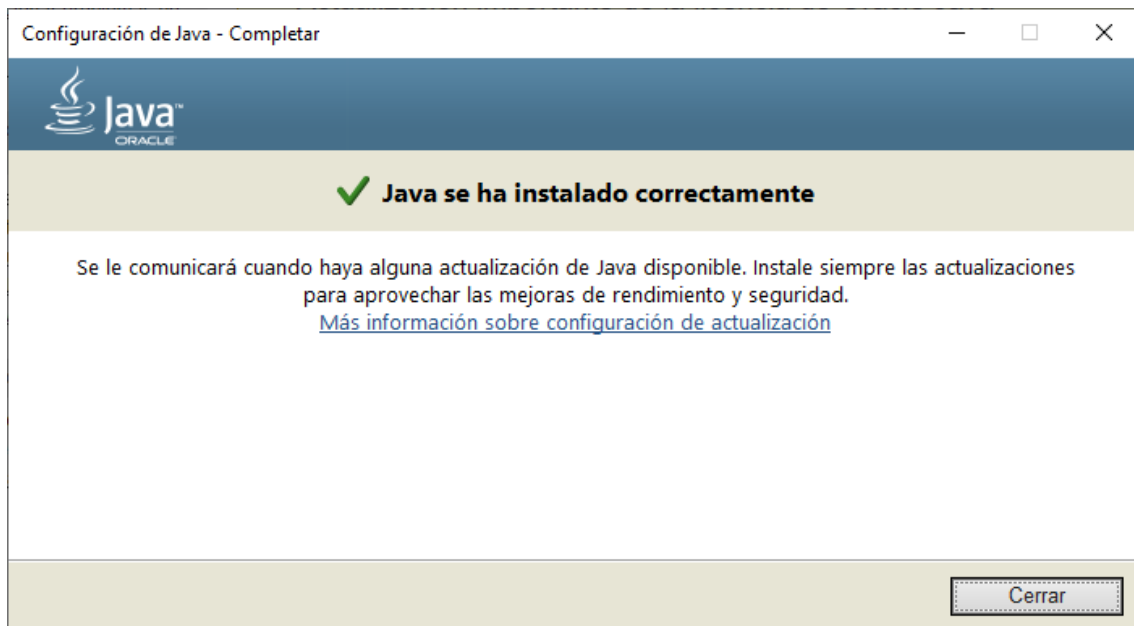


Ha detectado un versión anterior, seleccionaremos el botón Desintalar.



Seleccionaremos el botón Siguiente.





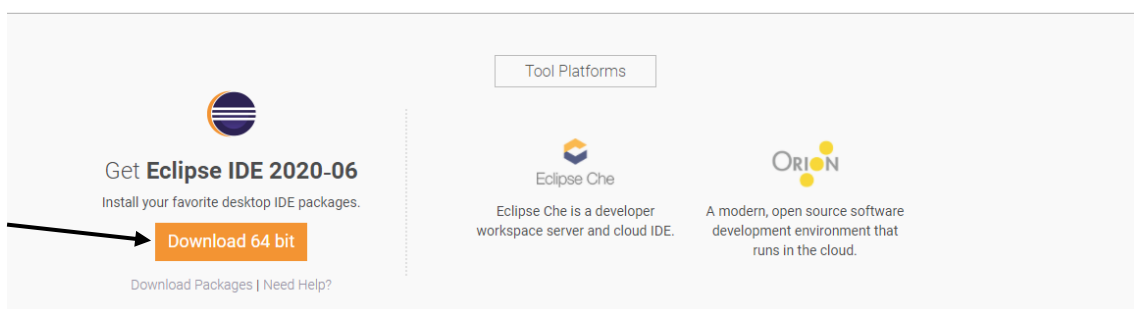
La instalación se ha realizado correctamente, seleccionaremos el botón Cerrar.

Ahora para la instalación de Eclipse accederemos al siguiente enlace:

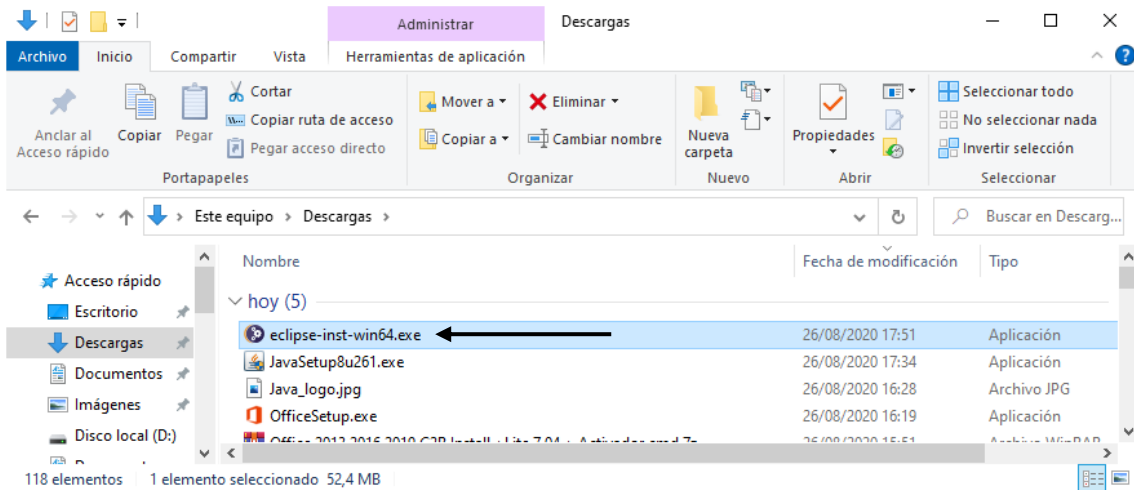
<https://www.eclipse.org/downloads/>



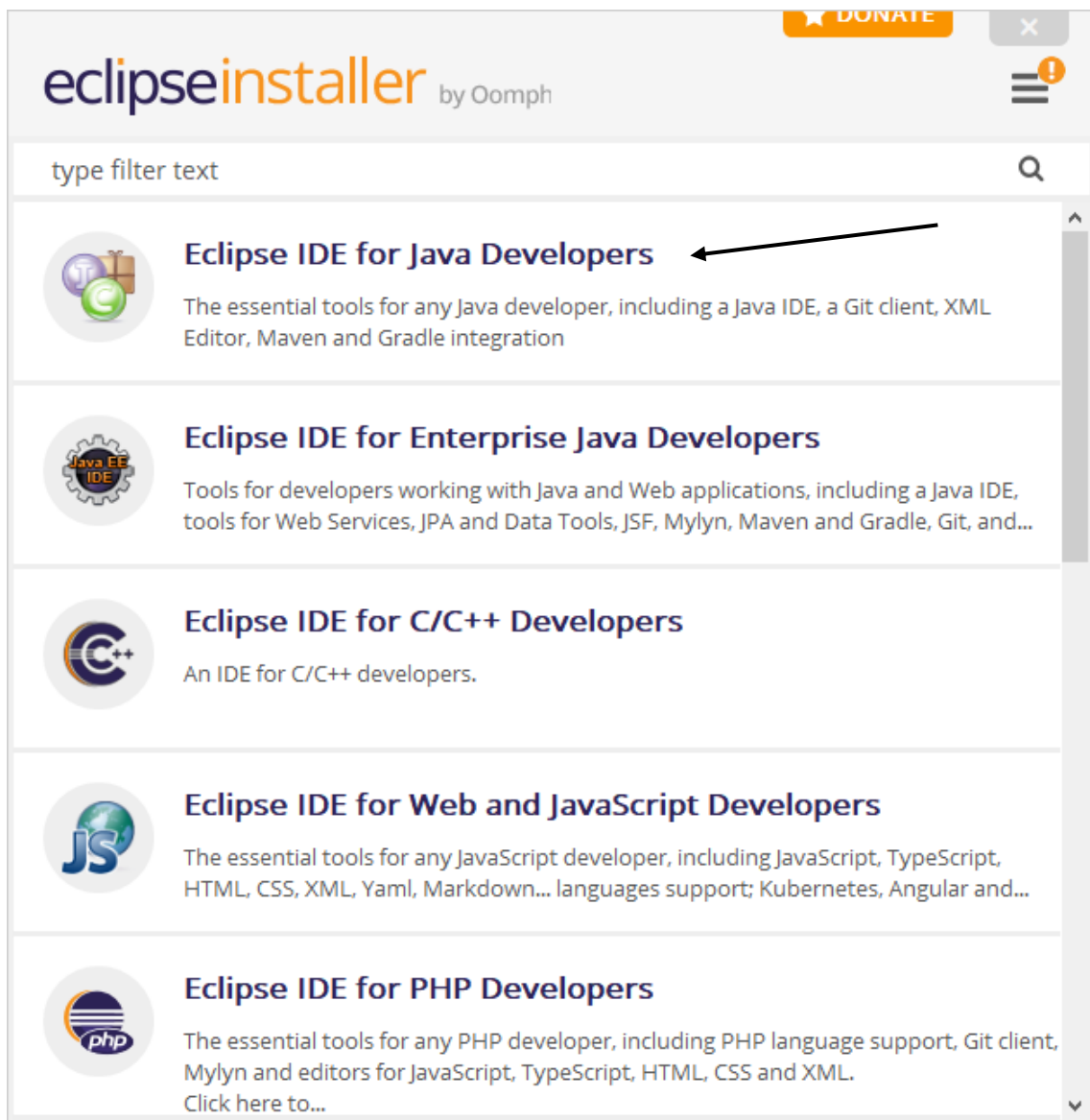
Download Eclipse Technology  
that is right for you



Seleccionaremos el botón Download 64 bit.

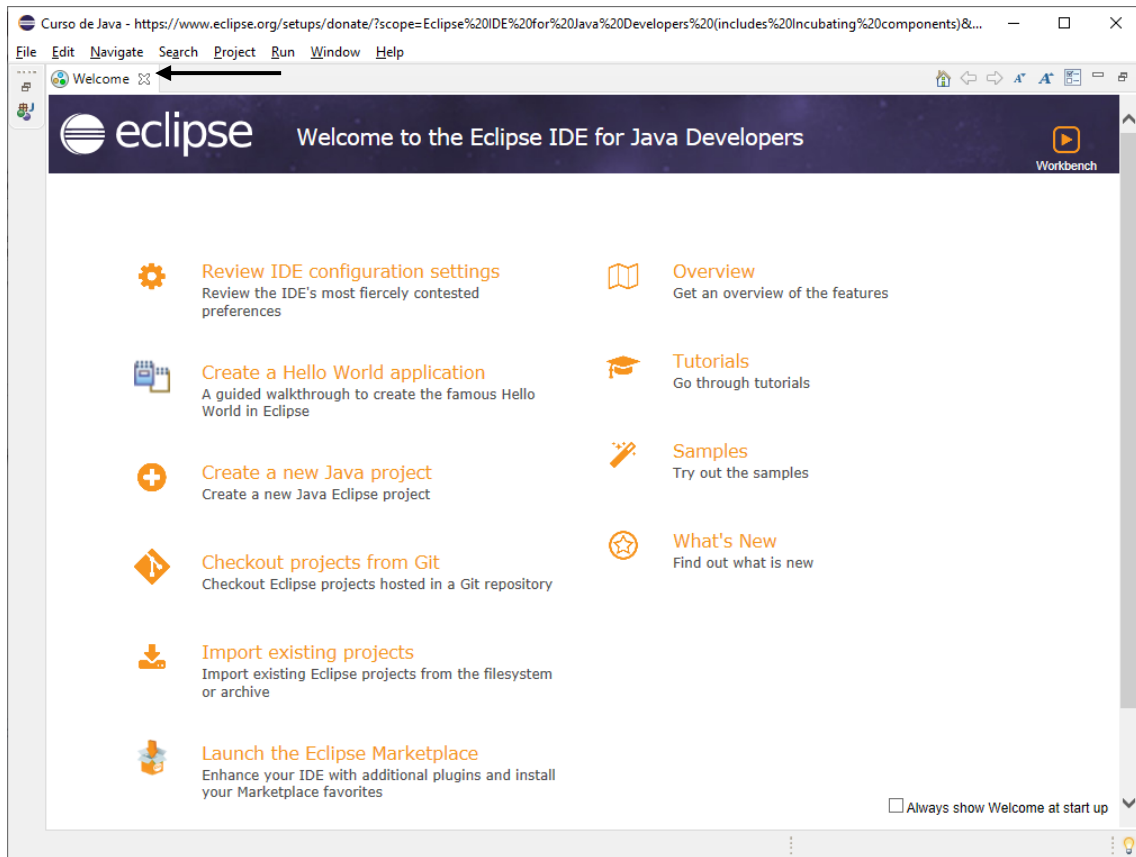


Ejecutamos para su instalación.

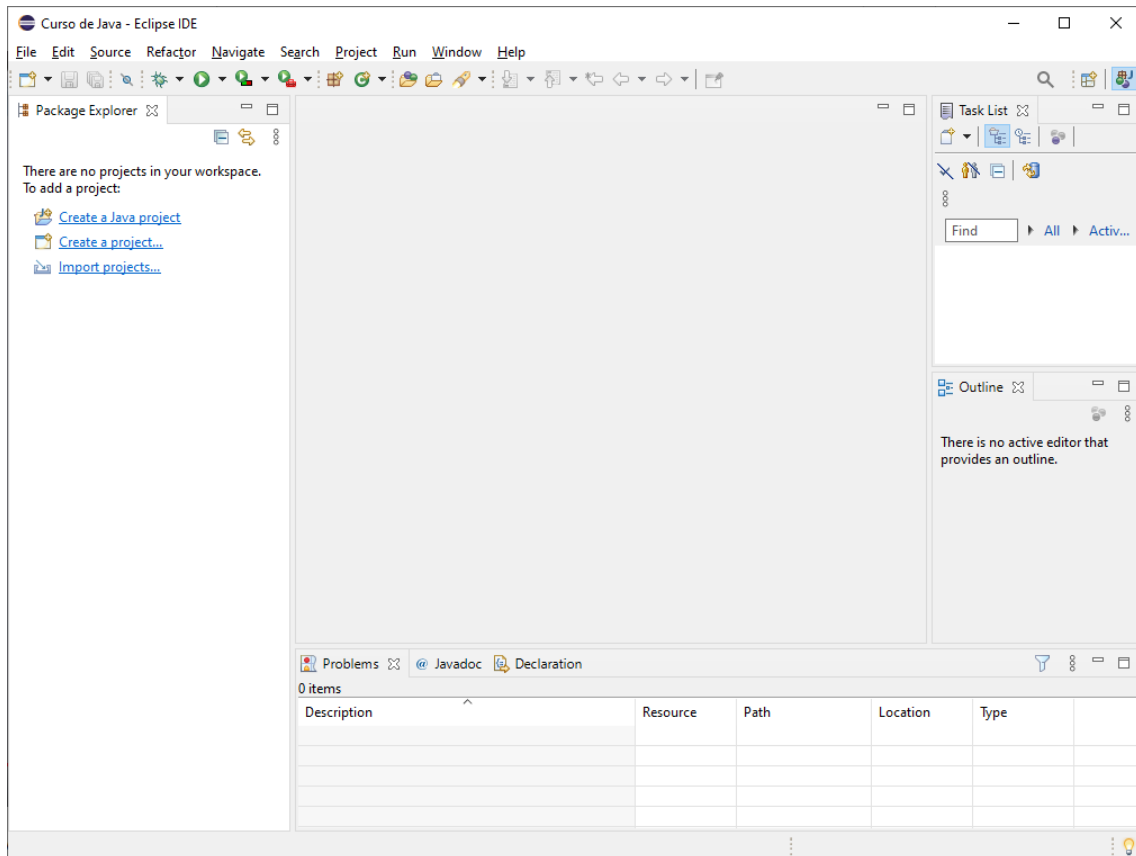




Vamos a crear una carpeta porque la primera vez que se ejecute Eclipse nos pedirá en que carpeta queremos guardar nuestros proyectos.



Cerramos la ventana de bienvenida dándole a la x.





## Introducción (Vídeo 3)

¿De dónde ha salido?

Características principales.

- Año 1991: un grupo de ingenieros de Sun Microsystems (liderados por James Gosling y Patrick Naughton) tenía que desarrollar un lenguaje de programación que se pudiera utilizar en pequeños electrodomésticos.



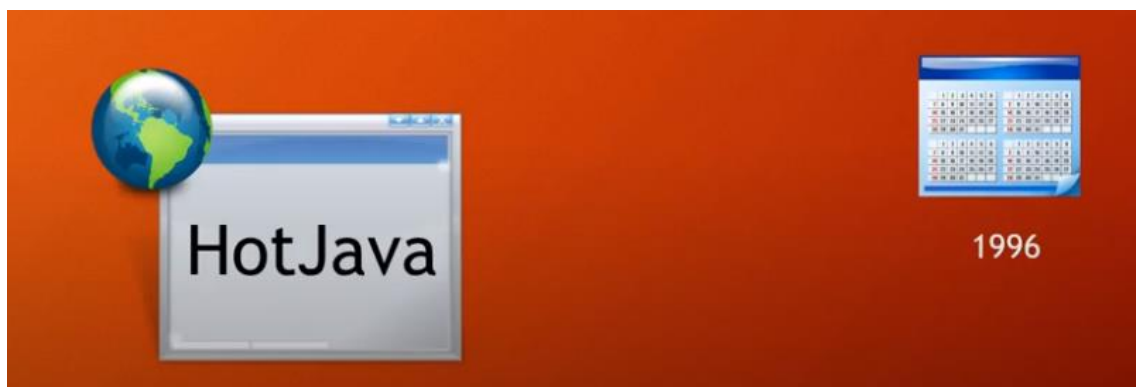
- Código pequeño
- Código compacto
- Neutro respecto arquitectura

En principio el nombre que se le puso fue OAK que significa roble ya que desde el despacho había una ventana en la que se divisaba un gran roble, pero este nombre ya estaba reservado y decidieron poner JAVA porque en aquel despacho solían tomar café y esto llama tomar Java.

Fracaso comercial

- De 1991 a 1994 intentaron vender la tecnología a diferentes empresas del ámbito tecnológico. No hubo éxito.
- El proyecto liderado por Gosling y Naughton queda en stand-by.
- Corría el año 1994 e Internet se hacía más grande... Pensaron que las características de Java se ajustaban como un guante a la naturaleza de Internet.

Alternativa



Hicieron un navegador llamado HotJava.

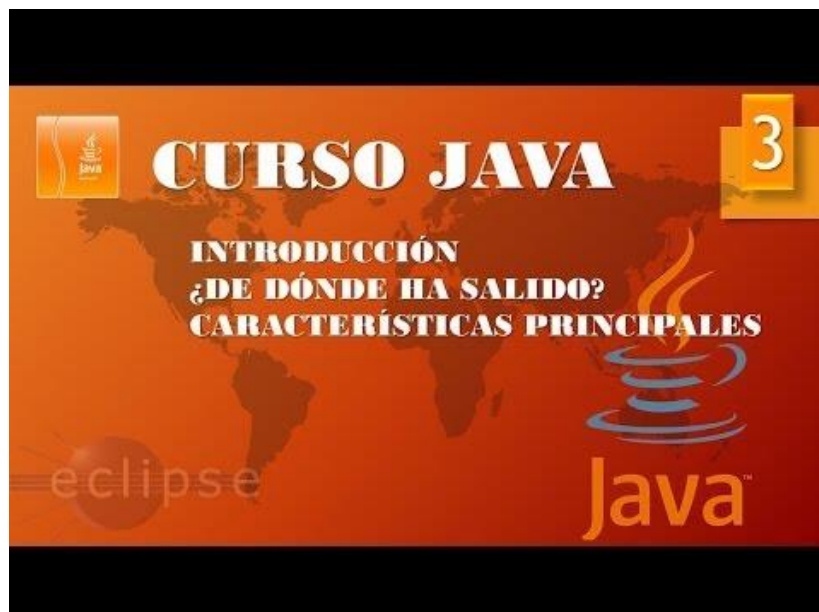
- Inventemos un navegador ligero, capaz de correr en cualquier plataforma y que pueda ejecutar código en su servidor (Applets).

## Características de Java

- Sencillo: en el sentido de que se intentó quitar las características más engorrosas de otros lenguajes de programación (aritmética de punteros, ficheros de encabezado, etc) y sobre todo que ocupa poco.
- Orientado a objetos.
- Distribuido: buen tratamiento de redes. Buena programación para Internet.
- Seguro: Como fue pensado para programar en red (Internet) se hizo seguro.
  - Leer o escribir ficheros sin permiso.
  - Desbordar la pila de ejecución.
  - Corrupción de memoria.
- Neutro.
- Adaptable: tipo de datos primitivos iguales en todas las plataformas.
- Interpretado.
- Alto rendimiento.

## Ideas erróneas

- Java no es una extensión de HTML.
- Java no tiene nada que ver con JavaScript.
- Todos los programas de Java se ejecutan en una página web.
- Java es inseguro.
- Java es 100% seguro.



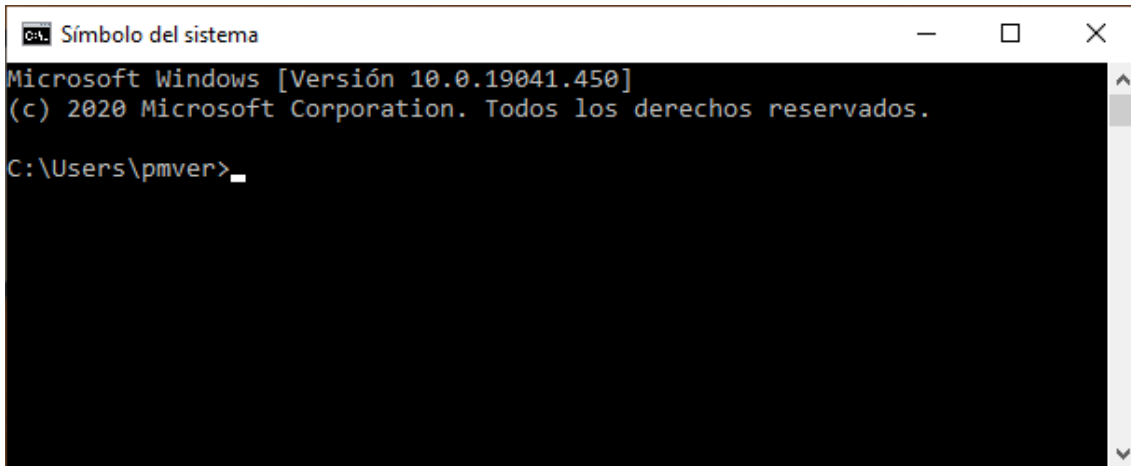


## Estructuras principales I (Vídeo 4)

### Tipos de programas Java

- Aplicaciones de consola.
- Aplicaciones de propósito general
- Applets

Las aplicaciones de consola se verán en una ventana de comandos.



```
ca. Símbolo del sistema
Microsoft Windows [Versión 10.0.19041.450]
(c) 2020 Microsoft Corporation. Todos los derechos reservados.
C:\Users\pmver>
```

También se utiliza la consola de Eclipse.

Aplicaciones de propósito general son aplicaciones realizadas con Java, el mismo Eclipse es un programa que está realizado con Java.

Applets: son programas realizados en Java que se ejecutan en un navegador como un plugin.



En algunas páginas web se tienen que cargar el plugin de Java para que una parte de la página web pueda funcionar.

Vamos a empezar por las aplicaciones de consola.

```
public class PrimerEjemplo {  
  
    public static void main(String args[]){  
  
        System.out.println("Hola alumnos!");  
    }  
}
```

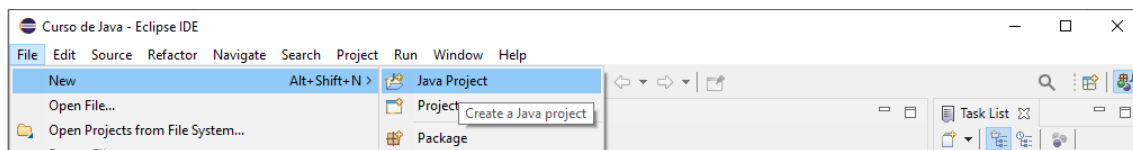
Case sensitive distingue entre mayúsculas y minúsculas.

Public: Modificador de acceso.

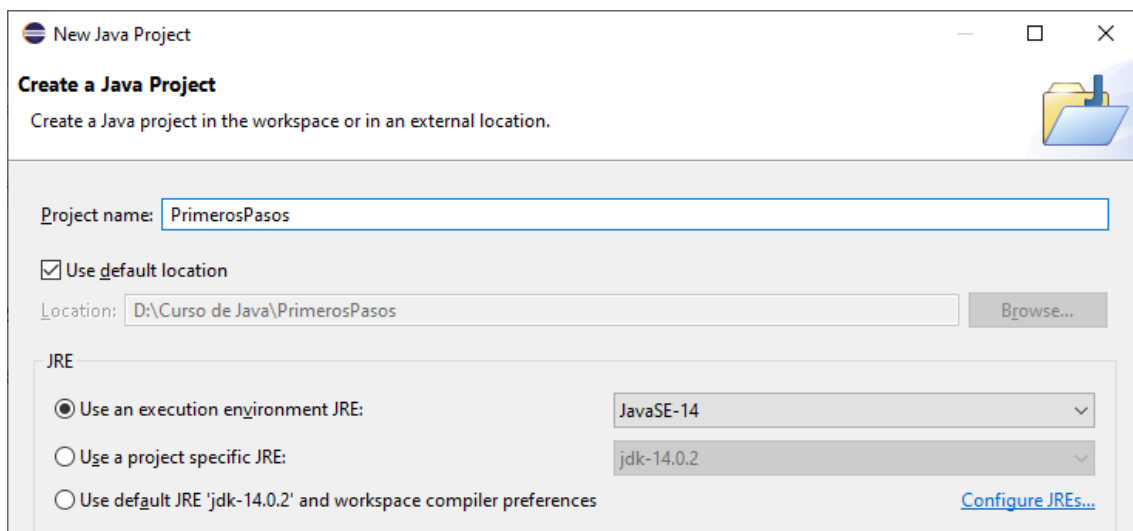
Class: clase (Todo programa tiene que estar dentro de una clase, está delimitadas por { } llaves.

Ahora vamos a ejecutar Eclipse.

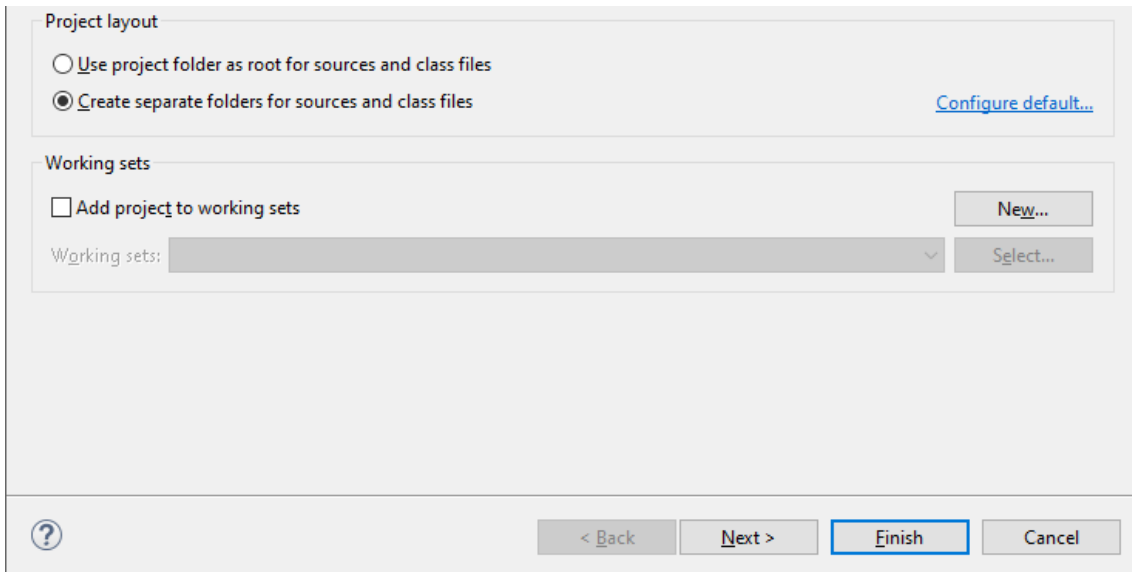
Vamos a crear un proyecto.



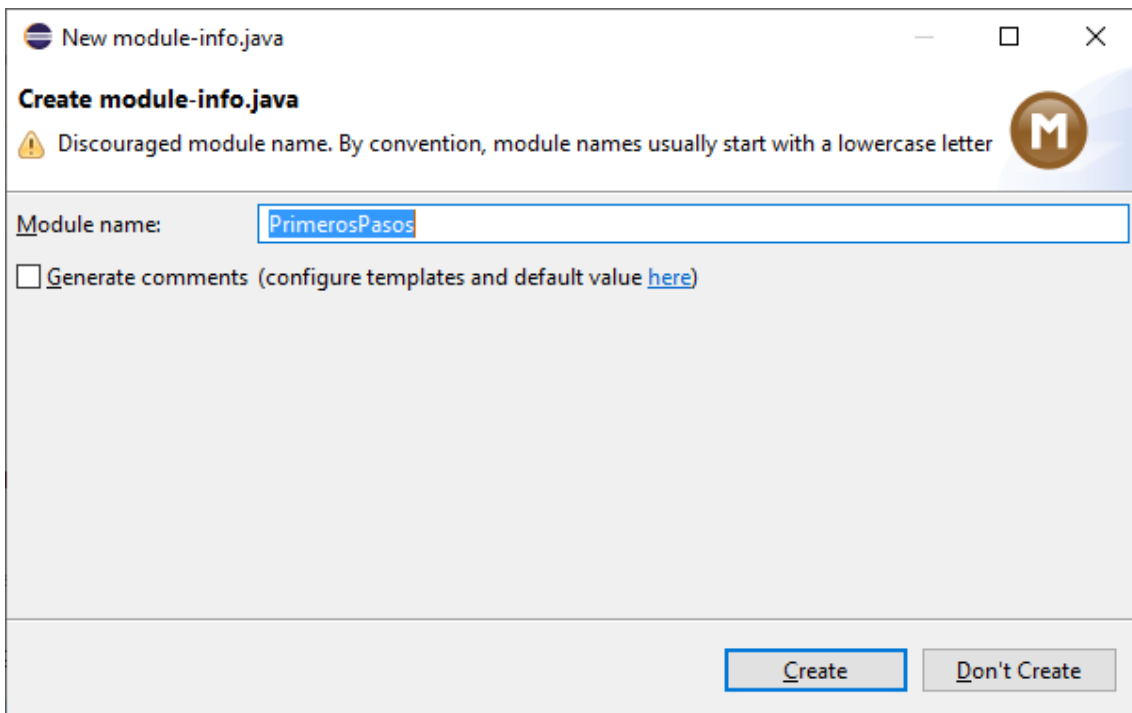
Del menú File seleccionamos New y de este Java Project.



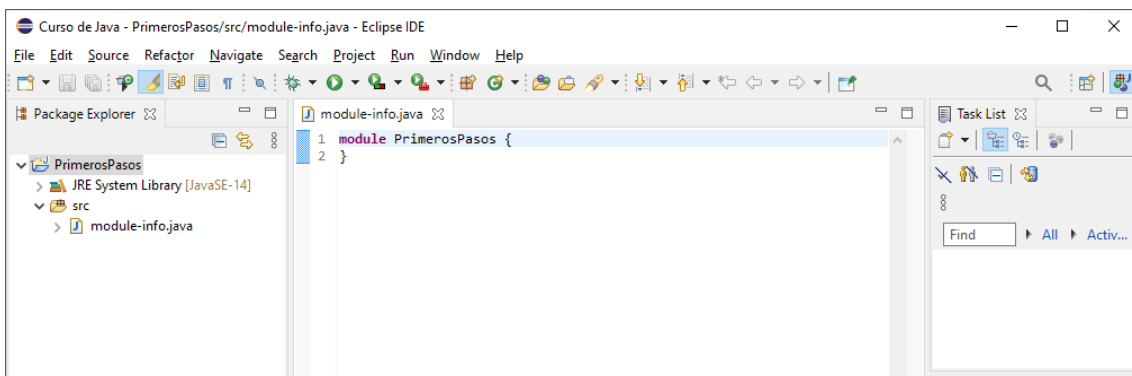
Como nombre del proyecto PrimerosPasos.



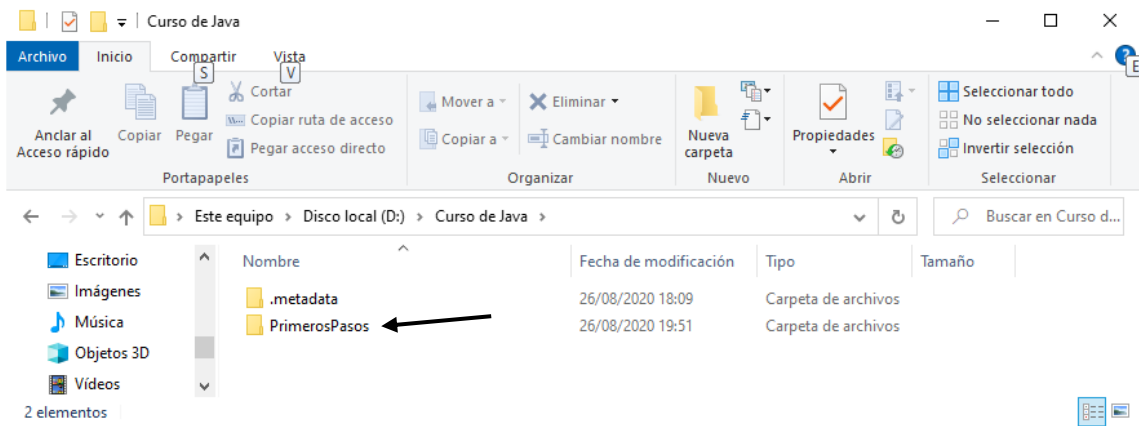
Seleccionaremos el botón Finish.



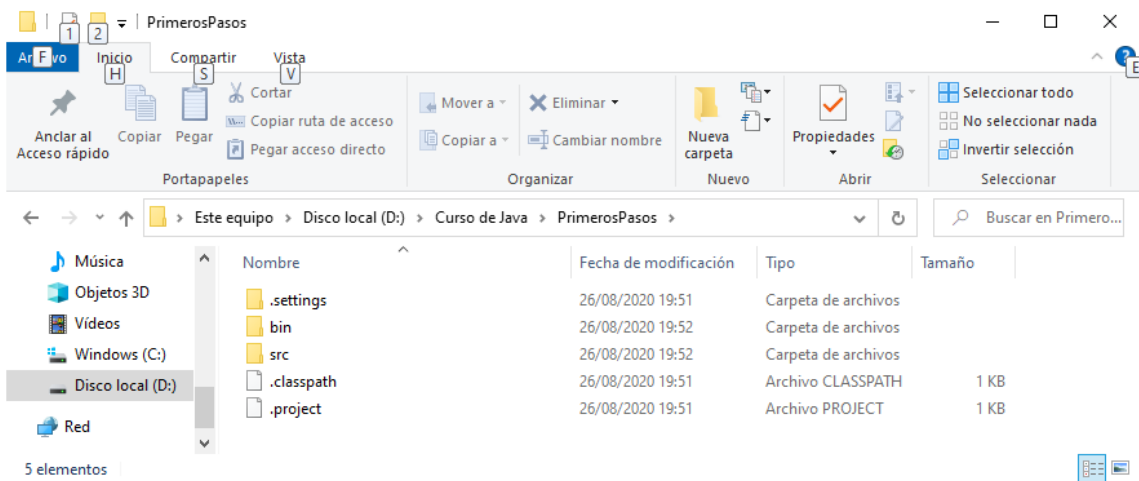
Seleccionaremos el botón Create.



Si vamos a la carpeta que creamos para los proyectos de Java.

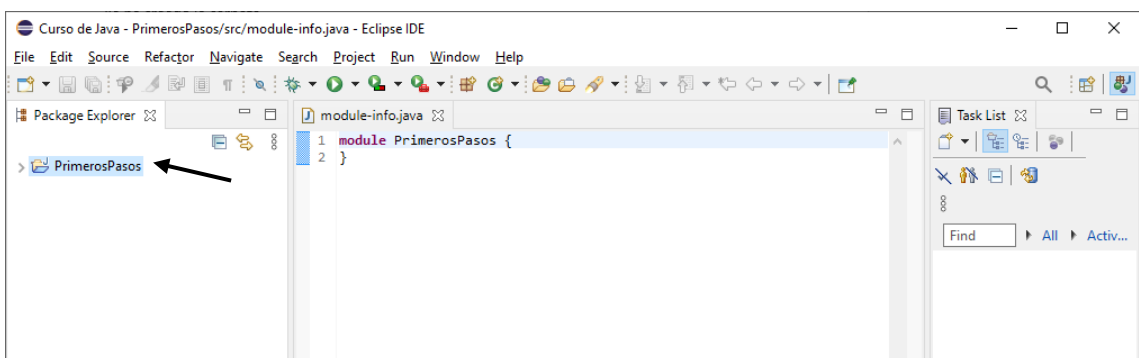


Ya ha creado la carpeta.

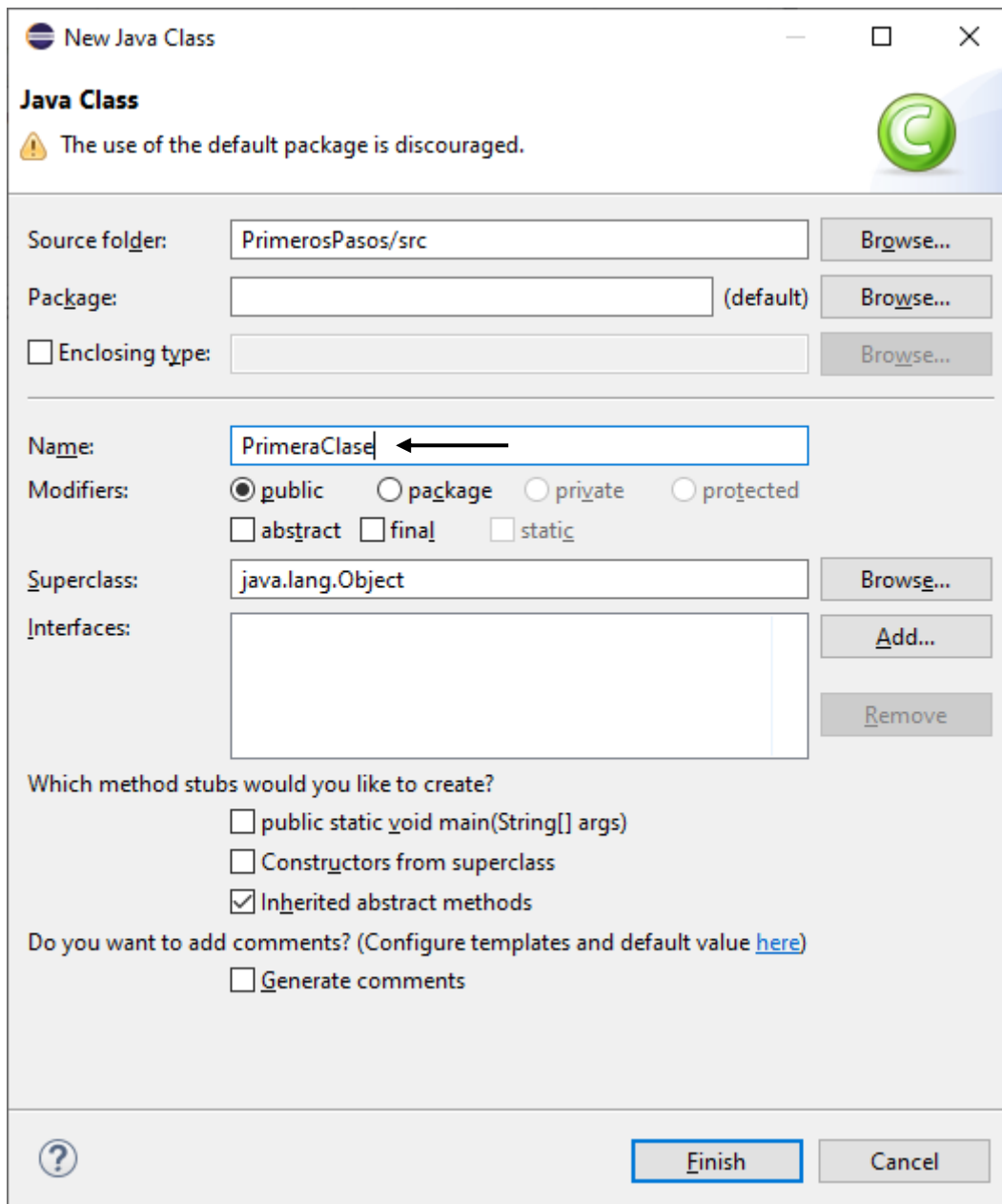


Con subcarpetas y archivos necesarios para el proyecto.

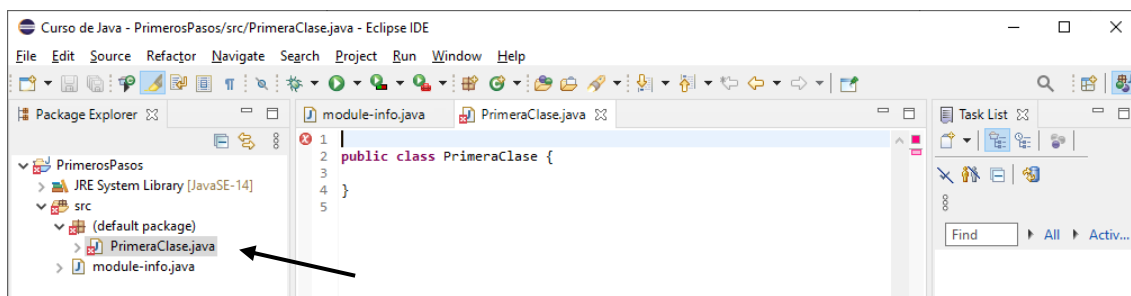
Ahora desde Eclipse



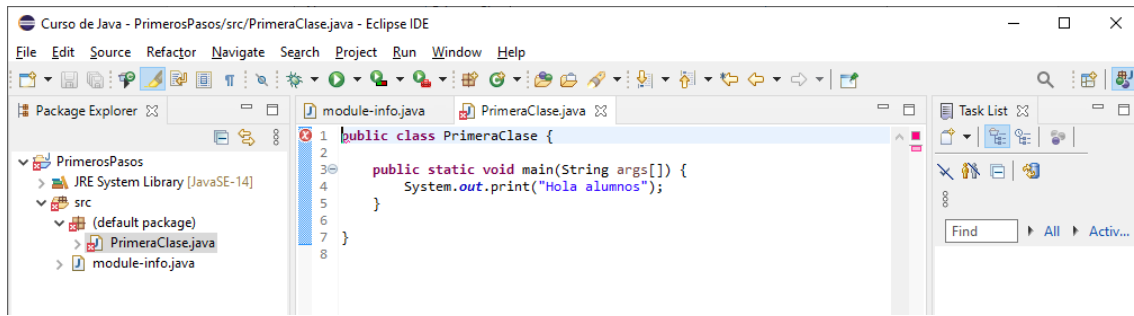
Hacemos un clic en el proyecto y ahora del menú File seleccionaremos New y de este Class.



Como nombre de la clase PrimeraClase, seguido del botón Finish.

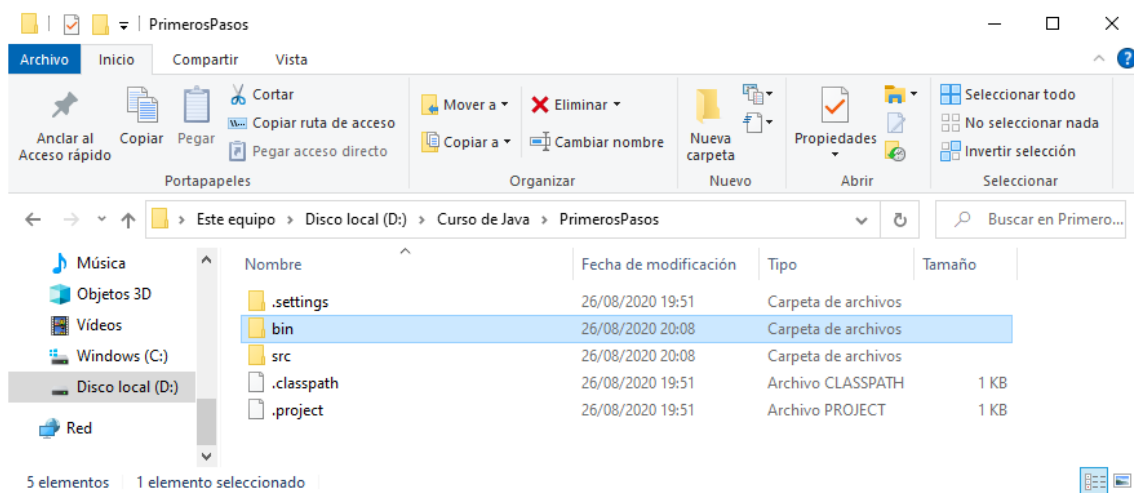


Ya se ha creado la clase.



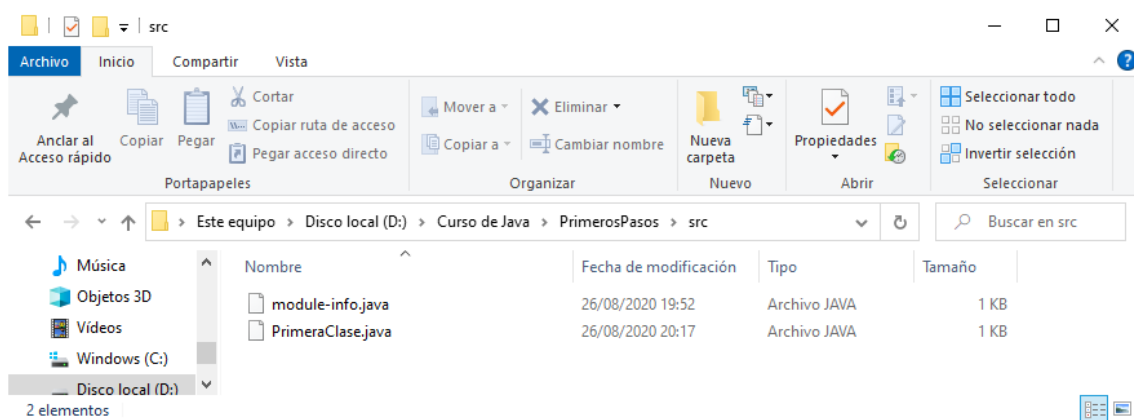
Ahora del menú File seleccionamos Save.

Si vamos a la carpeta de nuestros proyectos y entramos a la carpeta PrimerosPasos.



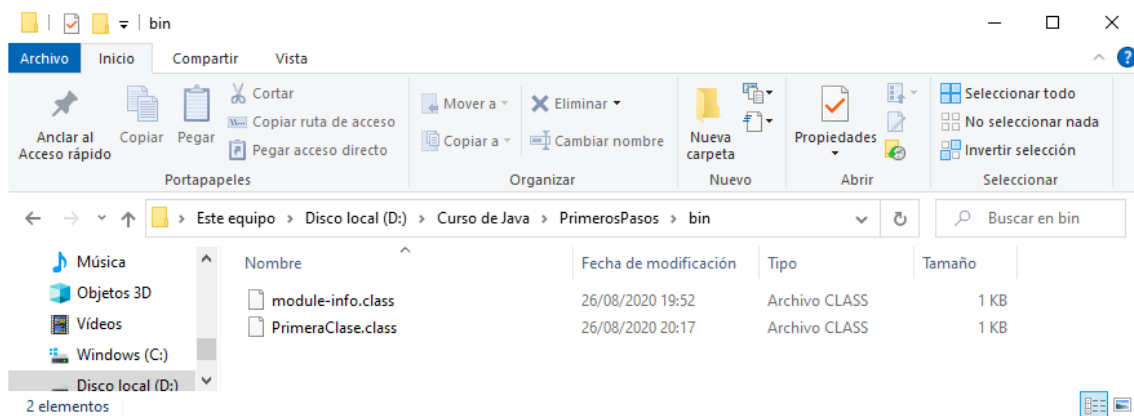
Vamos a ver el contenido de las carpetas src y bin.

En src



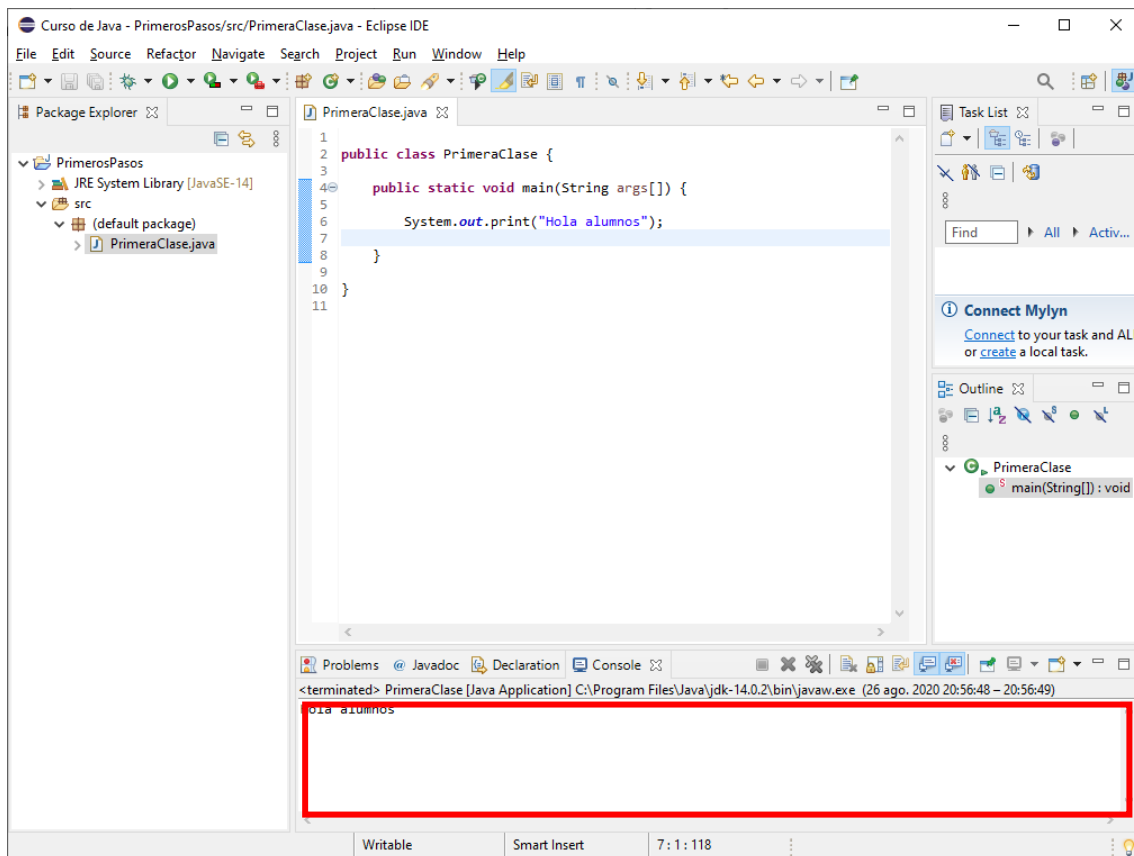
El proyecto con extensión java.

En bin



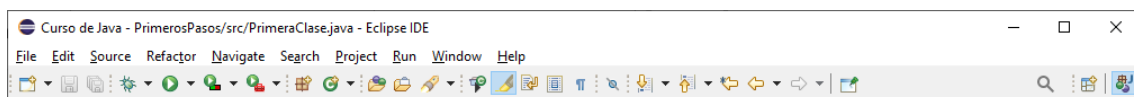
Ya nos ha creado el archivo compilado, recuerda que es multiplataforma.

Desde Eclipse vamos a habilitar la consola, para ello del menú Windows seleccionamos Show View y de este consola.



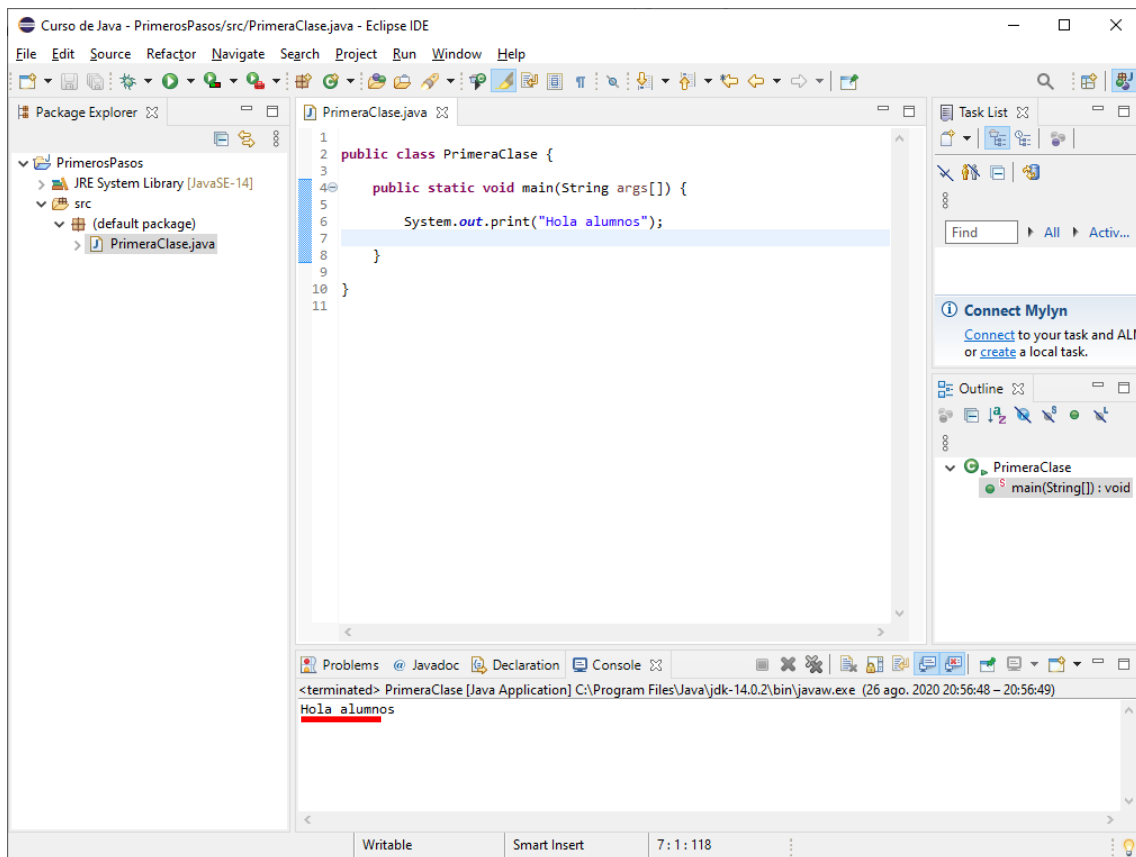
Es donde se verá el resultado de nuestros proyectos.

Ejecutamos el proyecto.





Este será el resultado:



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left shows a project named 'PrimerosPasos' with a source folder 'src' containing a file 'PrimeraClase.java'. The main editor window displays the following Java code:

```
1 public class PrimeraClase {  
2  
3  
4     public static void main(String args[]) {  
5  
6         System.out.print("Hola alumnos");  
7  
8     }  
9  
10 }  
11
```

The Console window at the bottom shows the output of the program: `<terminated> PrimeraClase [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (26 ago. 2020 20:56:48 - 20:56:49)` followed by `Hola alumnos`.



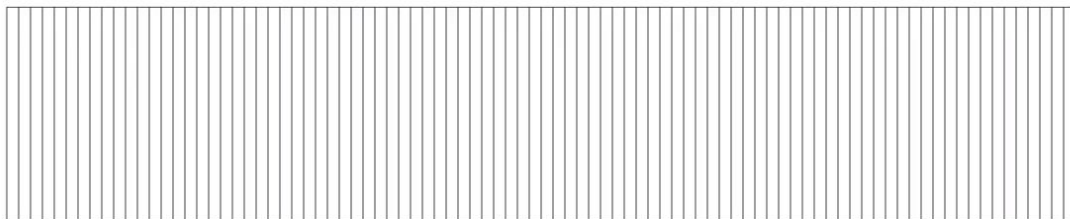
## Estructuras principales II (Vídeo 5)

### Tipos de datos en Java (Tipos primitivos)

- Enteros
  - Int: 4 bytes de espacio para almacenamiento. Desde -2.147.483.648 hasta 2.147.483.647.
  - Short: 2 bytes de espacio para almacenamiento. Desde -32.768 hasta 32.767.
  - Long: 8 bytes de espacio para almacenamiento. Una barbaridad. Sufijo L.
  - Byte: 1 byte de espacio para almacenamiento. Desde -128 hasta 127.
- Coma flotante (decimales)
  - Float: 4 bytes de espacio para almacenamiento. Aproximadamente 6 a 7 dígitos decimales significativas. Sufijo F.
  - Double: 8 bytes de espacio para almacenamiento. Aproximadamente 15 cifras decimales negativas.
- Char: Para representar caracteres.
- Boolean: True o False.

### Variables en Java

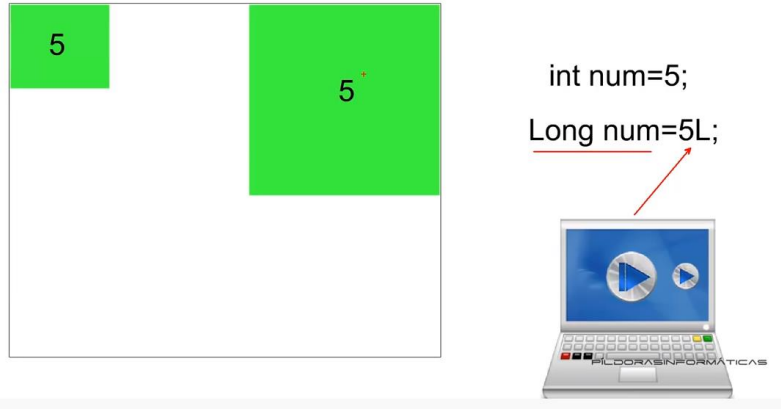
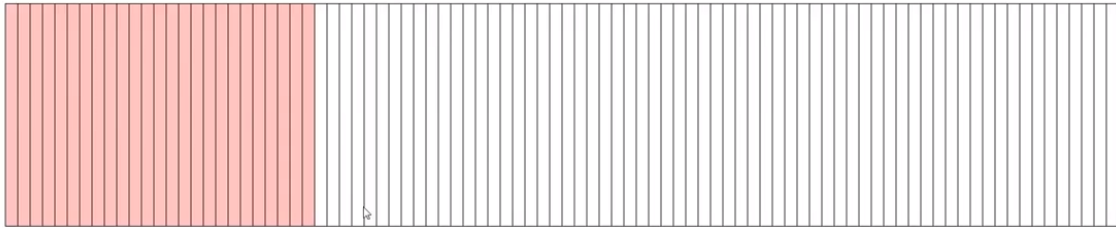
- ¿Qué es una variable? Espacio en la memoria del ordenador donde se almacenará un valor que podrá cambiar durante la ejecución del programa.
- ¿Por qué hay que utilizar variables? Porque a la hora de crear un programa surge la necesidad de guardar datos temporalmente que necesitarás utilizar en el futuro en ese mismo programa.
- ¿Cómo se crea una variable en Java? Especificando el tipo de dato que almacenará en su interior + el nombre de la variable. P. ej: `int salario;`
- ¿Qué es iniciar una variable? Es darle valor. `Nombre_variable=valor`. P. ej: `salario=2000;` Java no permite utilizar variables que no se hayan iniciado.



RAM 4GB



PÍLDORASINFORMÁTICAS



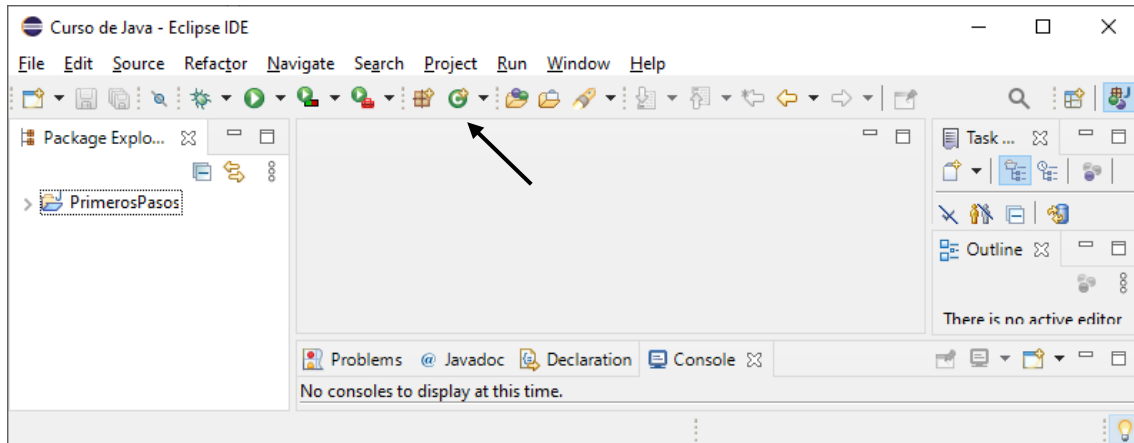
Tipos de variables que se utilizan con mayor frecuencia:

- Int, Double, Char y Boolean

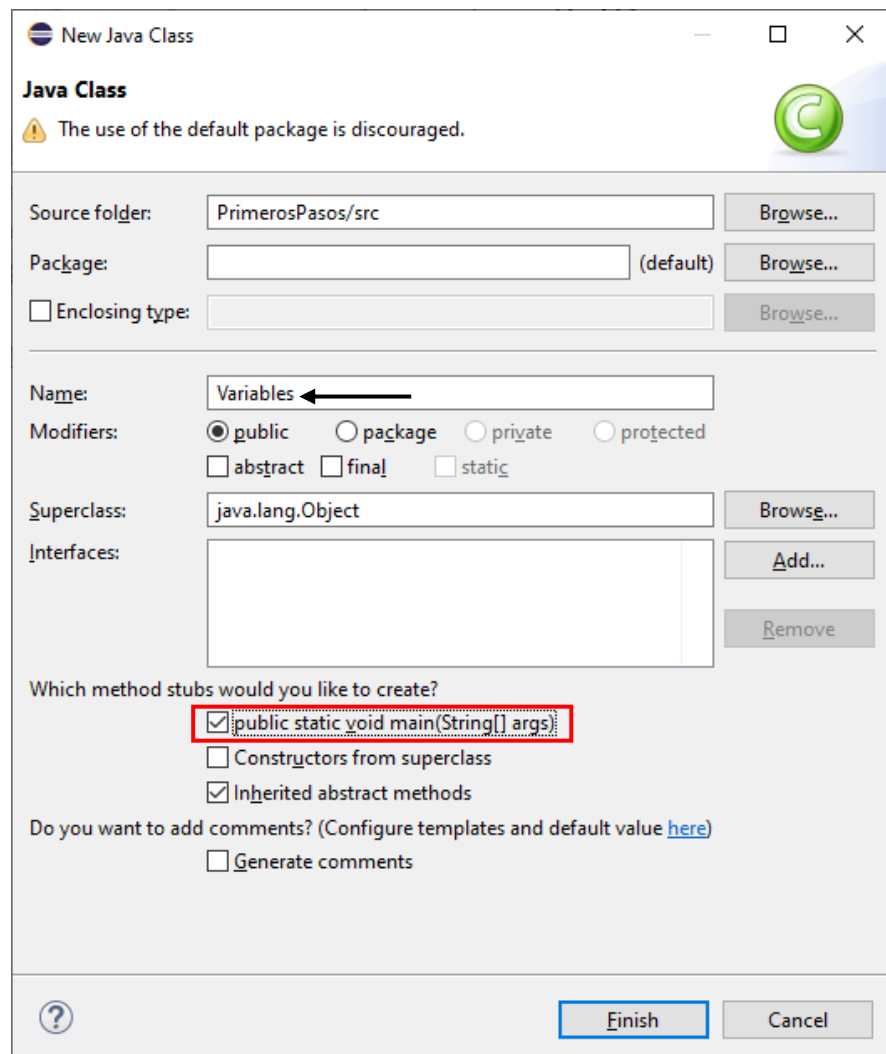


## Estructuras principales III. Declaración variables Eclipse (Video 6)

Vamos a crear una nueva clase.

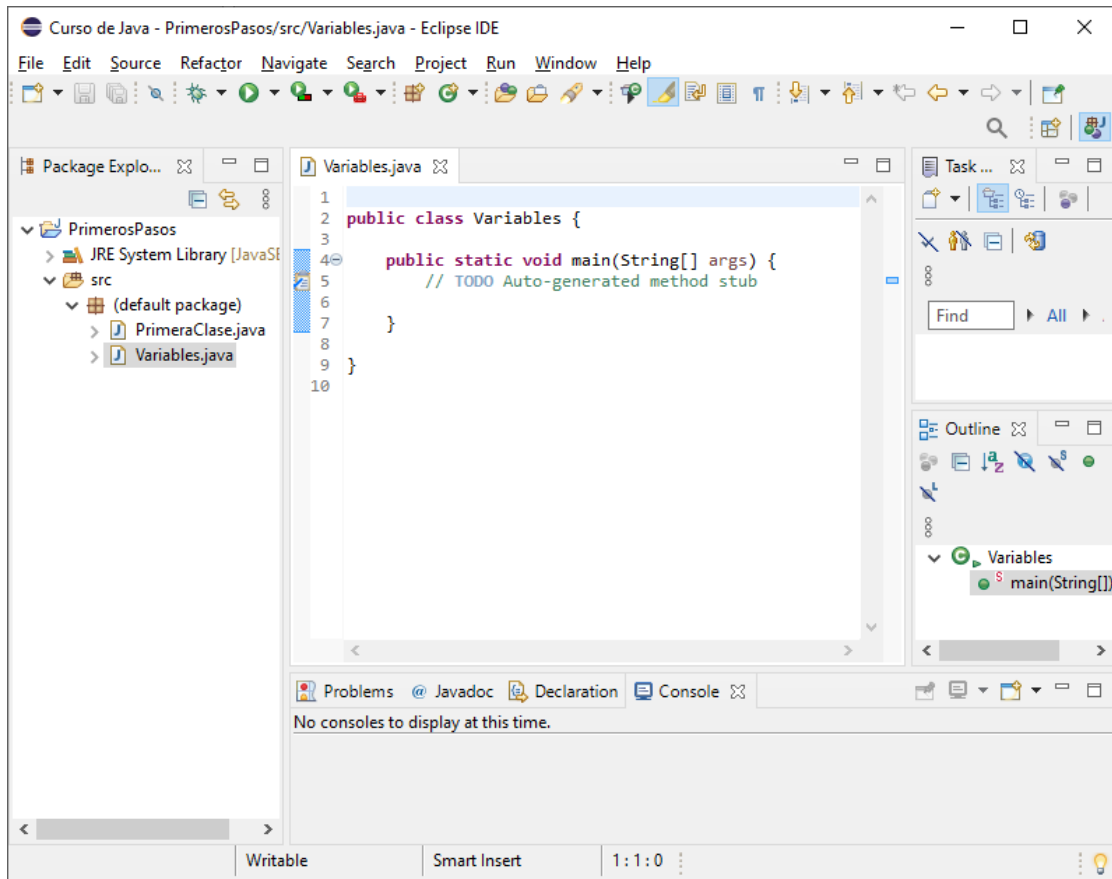


Seleccionaremos el correspondiente botón.

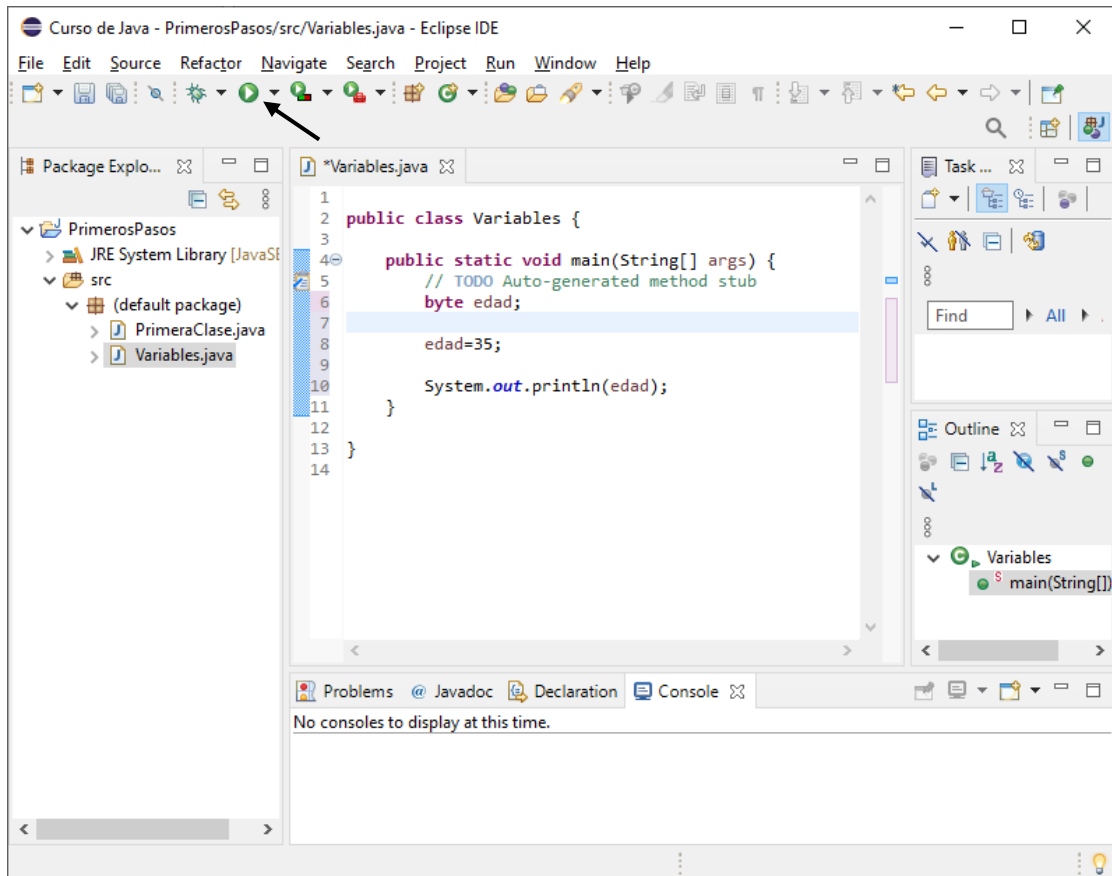


A esta clase la vamos a llamar Variables, activamos la casilla public static void main(String[] args).

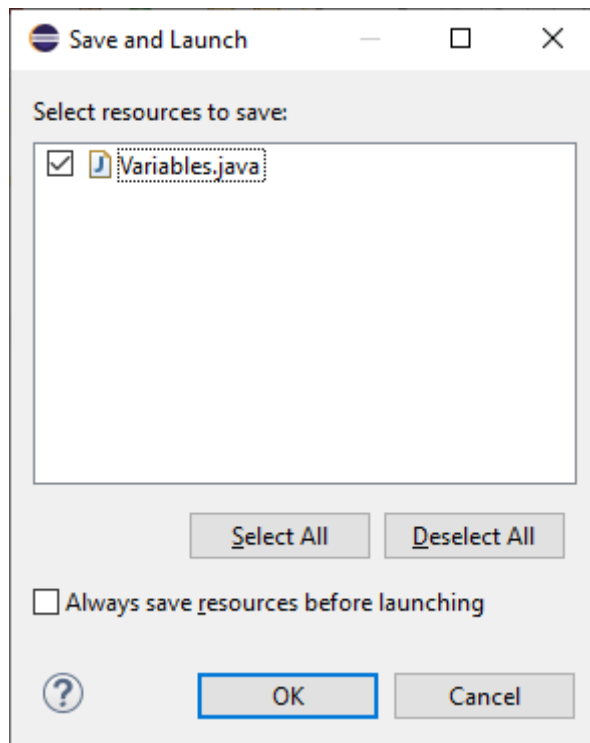
Seguido del botón Finish.



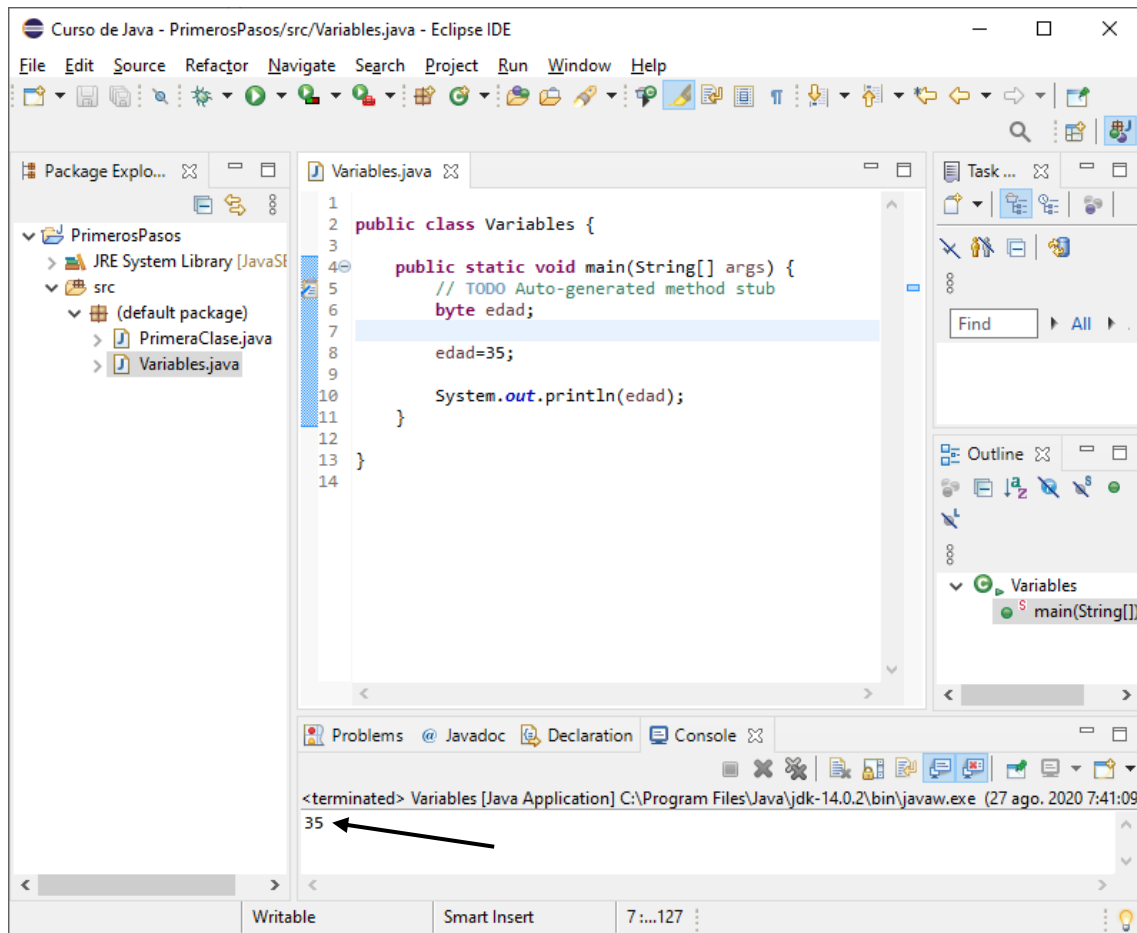
Vamos a declara e inicializar una variable.



Le vamos a dar al Play sin haber guardado la clase.



Nos preguntará si queremos guardar la clase a lo que contestaremos OK.



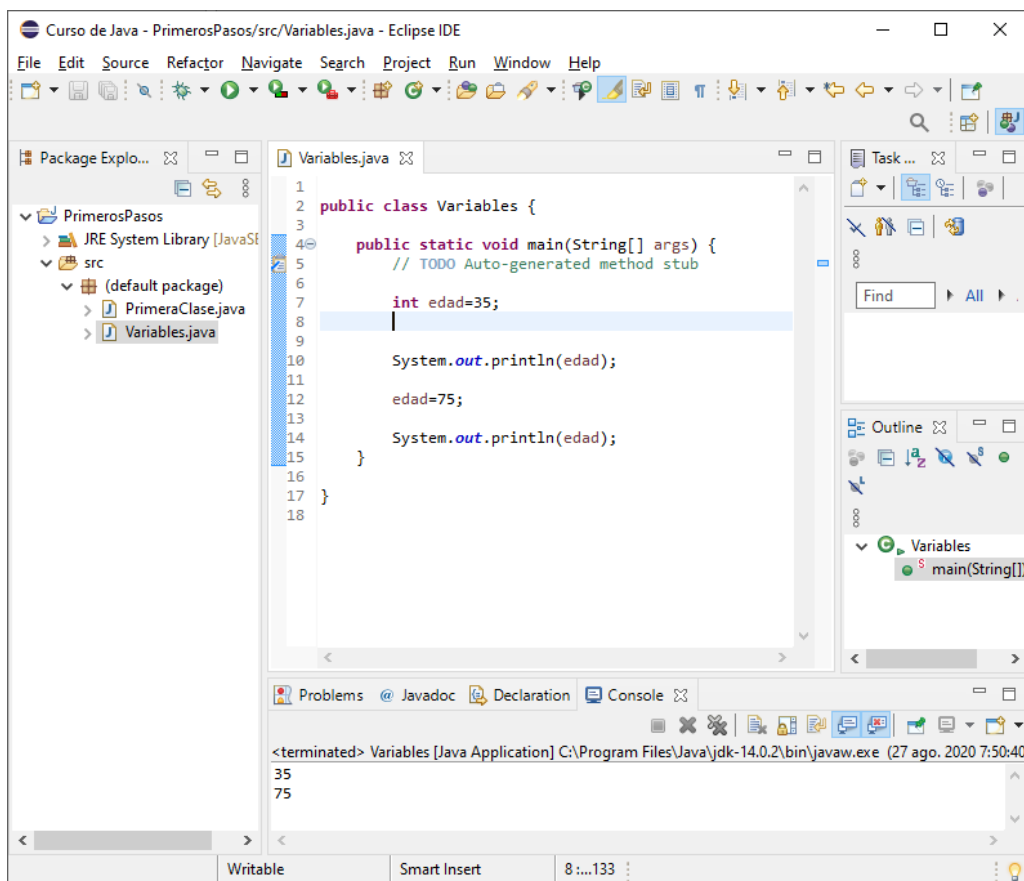
En la consola ya obtenemos el resultado.

```
1
2 public class Variables {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         byte edad=35;
8
9         System.out.println(edad);
10    }
11
12 }
13
```

En la misma línea declaramos e inicializamos la variable edad.

```
1
2 public class Variables {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         int edad=35;
8
9         edad=75;
10
11        System.out.println(edad);
12    }
13
14 }
```

Cambio el valor de la variable edad durante la ejecución del proyecto.





Java es un lenguaje orientado a objetos POO.

Los objetos tienen propiedades y métodos.

```
1
2 public class Variables {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         int edad=35; // Comentario
8
9         /* comentario de
10        * varias líneas */
11
12
13
14        System.out.println(edad);
15
16        edad=75;
17
18        System.out.println(edad);
19    }
20
21 }
```

Ejemplo de comentario de una y de varias líneas.



## Estructuras principales IV. Constantes y Operadores (VÍdeo 7)

- Declaración de constantes
- Operadores

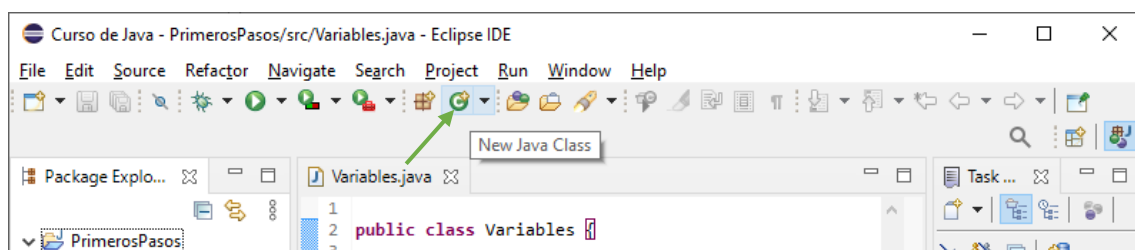
### Constantes en Java

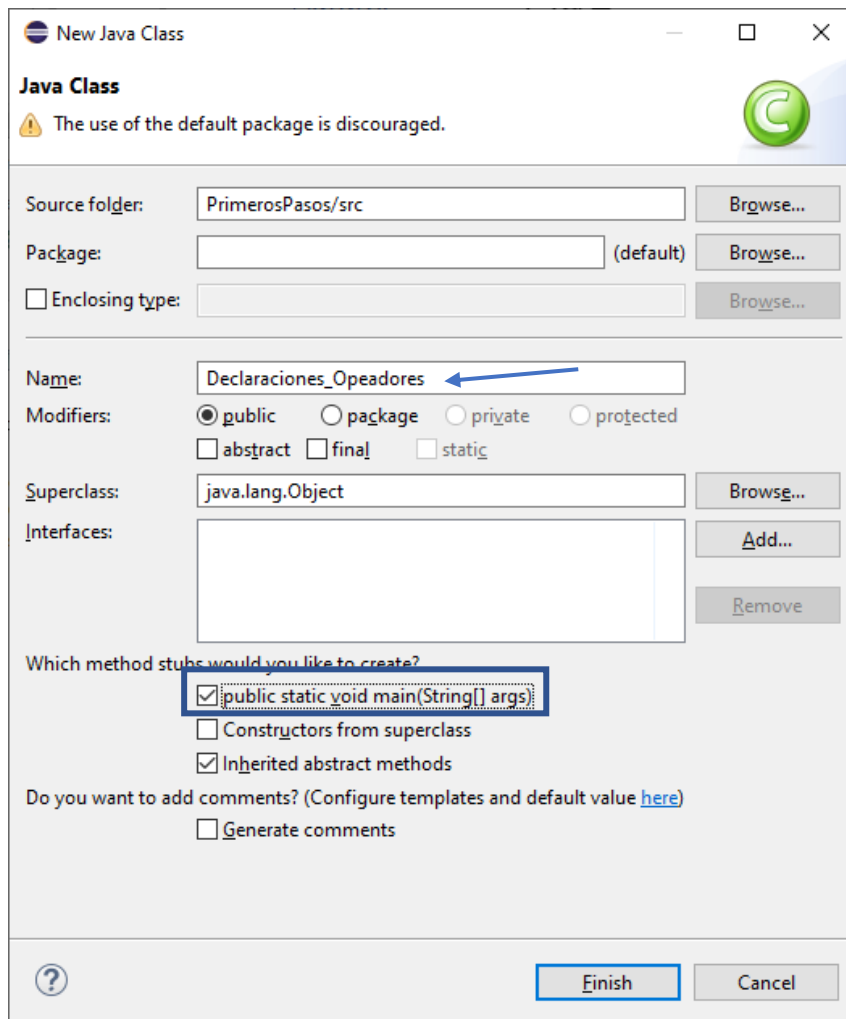
- ¿Qué es una constante? Espacio en la memoria del ordenador donde se almacenará un valor que no podrá cambiar durante la ejecución del programa.
- ¿Por qué hay que utilizar constantes? Porque a la hora de crear un programa a veces surge la necesidad de guardar datos temporalmente que necesitarás utilizar en el futuro en ese mismo programa. Dichos datos deberán ser fijos.
- ¿Cómo se crea una constante en Java? Utilizando la palabra clave final y a continuación especificando el tipo de dato que almacenará en su interior + el nombre de la consante = valor. P. ej: final double a\_pulgadas=2,54;
- Recuerda: El valor de una constante no podrá cambiar.

### Operadores en Java

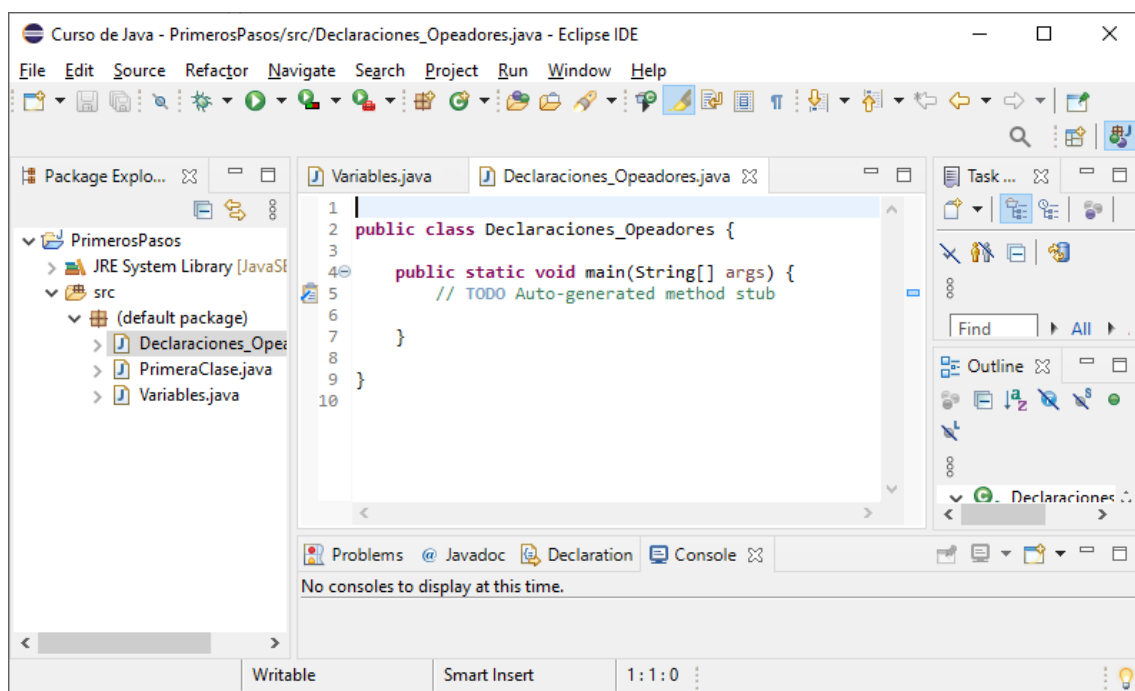
- Aritméticos
  - + : suma
  - - : resta
  - \* : multiplicación
  - / : división
- Lógicos, relacionales y booleanos
  - > : mayor que
  - < : menor que
  - <> : mayor o menor que
  - != : distinto que
  - == : igual que
  - && : y lógico
  - || : o lógico
- Incremento y decremento
  - ++ : Incremento
  - -- : Decremento
  - +=nº : incremento
  - -= nº : decremento
- Concatenación
  - + : une o concatena.

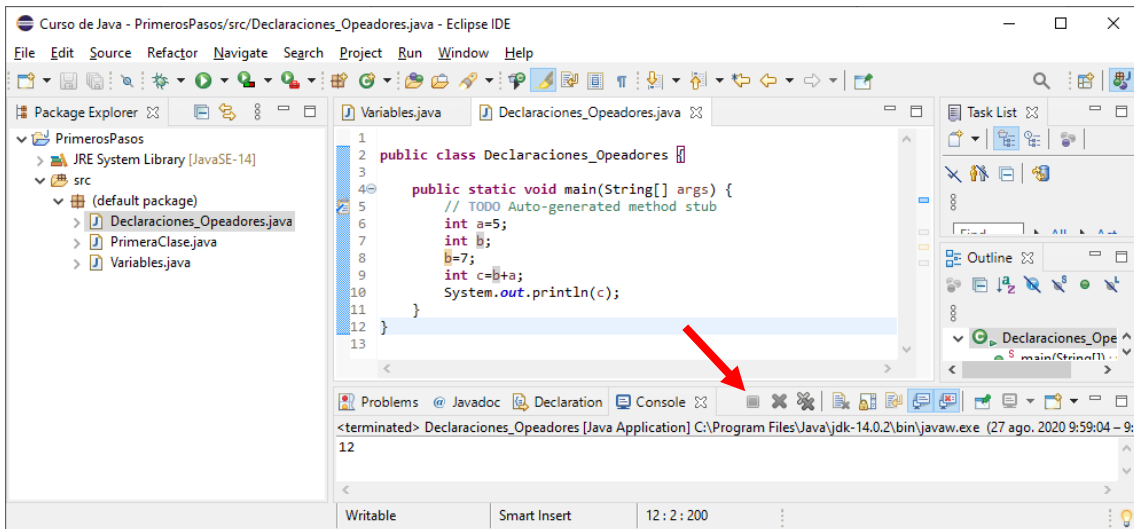
Ejecutamos Eclipse para crear una clase nueva.



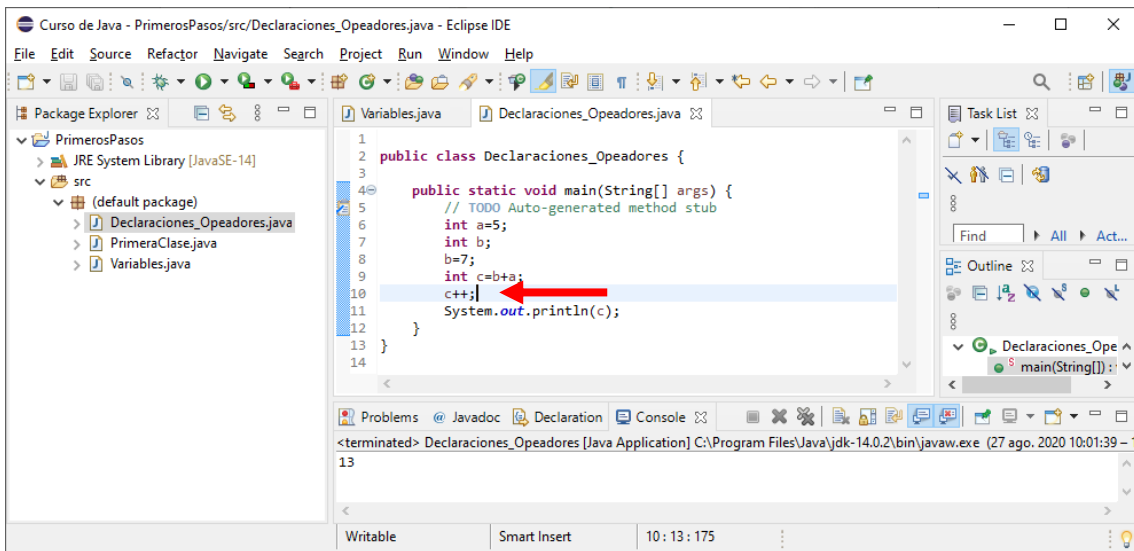


Seleccionaremos el botón Finish.

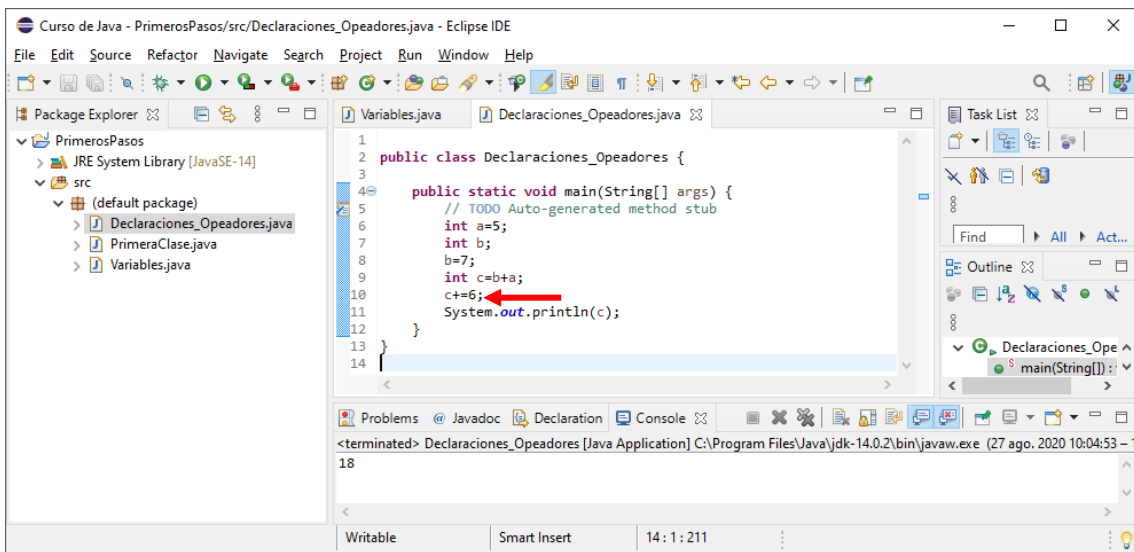




Si este punto está en rojo nos está diciendo que está compilando el programa.



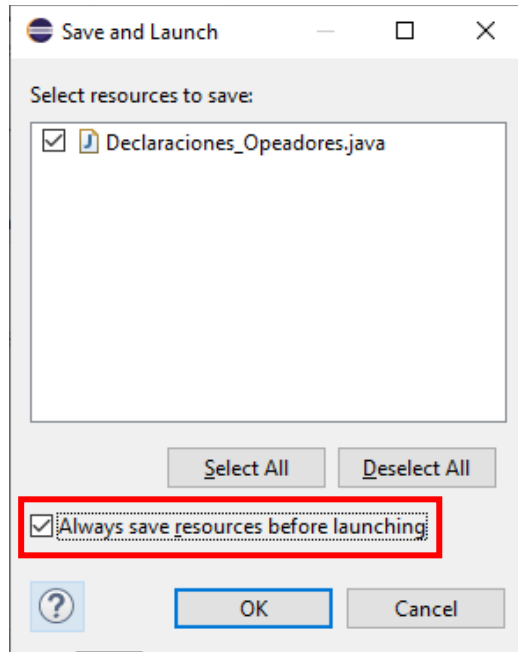
Incremento en 1.



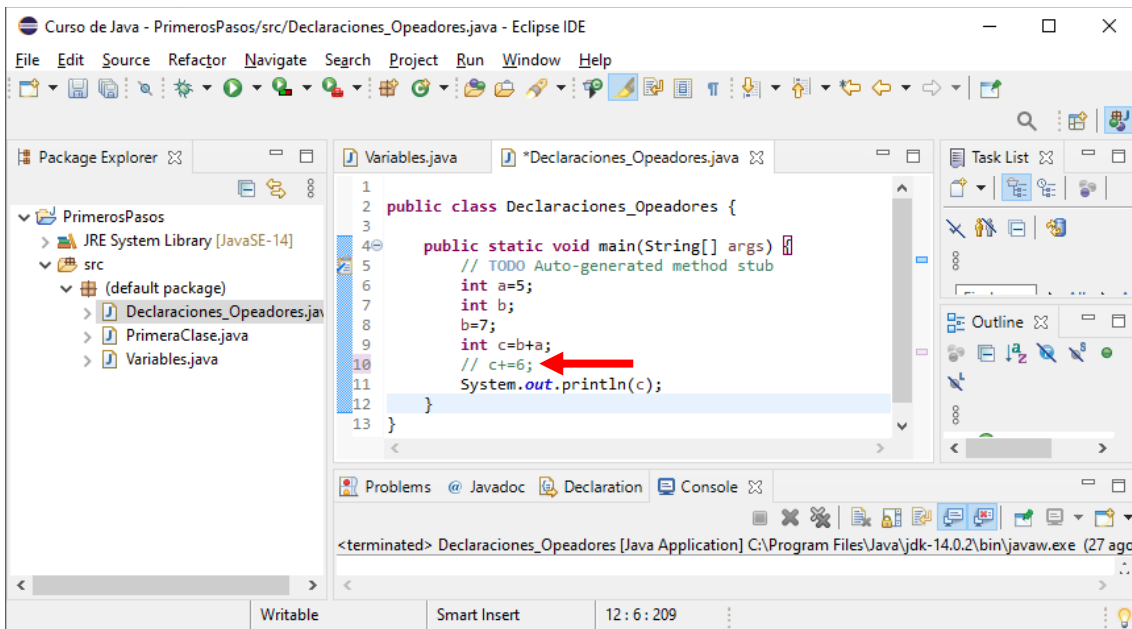
Le incrementamos en 6.



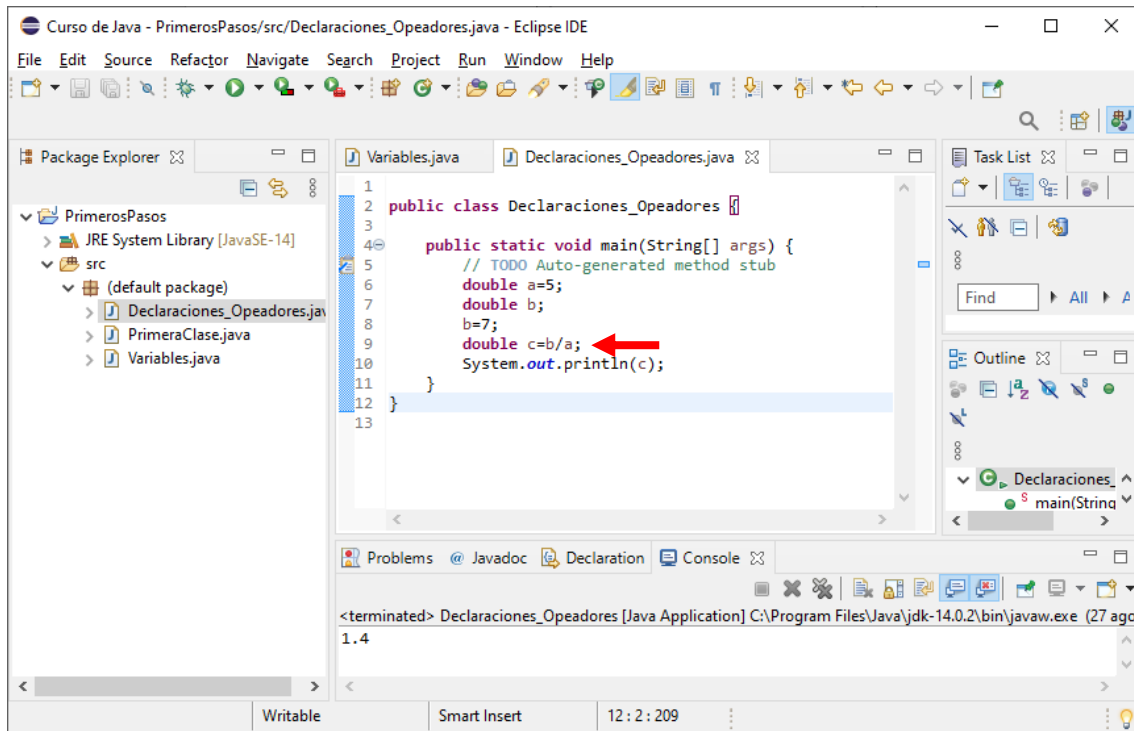
Cuando le damos al play y no hemos guardado el programa aparece una ventana para que confirmemos que nos guarde los cambios.



Si activamos “Always save resources before launching” ya no nos lo preguntará más.



Con // podemos comentar una línea de código para que esta no se ejecute, así no la tenemos que borrar y en un futuro quitando las // vuelve a estar en el código.



```
1  
2 public class Declaraciones_Opeadores {  
3  
4     public static void main(String[] args) {  
5         // TODO Auto-generated method stub  
6         double a=5;  
7         double b;  
8         b=7;  
9         double c=b/a; ←  
10        System.out.println(c);  
11    }  
12 }  
13
```

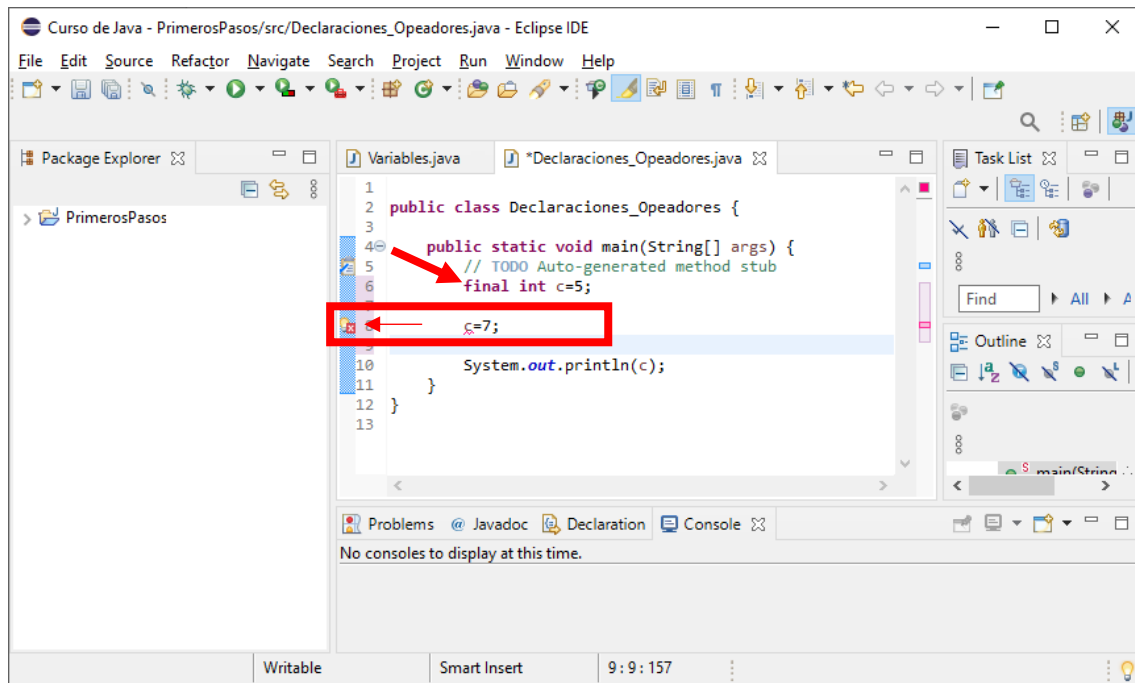
The screenshot shows the Eclipse IDE interface. The Package Explorer on the left shows a project named 'PrimerosPasos' with a source folder 'src' containing files 'Declaraciones\_Opeadores.java', 'PrimeraClase.java', and 'Variables.java'. The main editor window displays the code for 'Declaraciones\_Opeadores.java'. A red arrow points to the division operator '/' in the line 'double c=b/a;'. The Console window at the bottom shows the output '1.4'.

Como esta división va a tener decimales esta tiene que ser de tipo double, pero además todas las variables que intervienen en el cálculo también tienen que ser de tipo double.

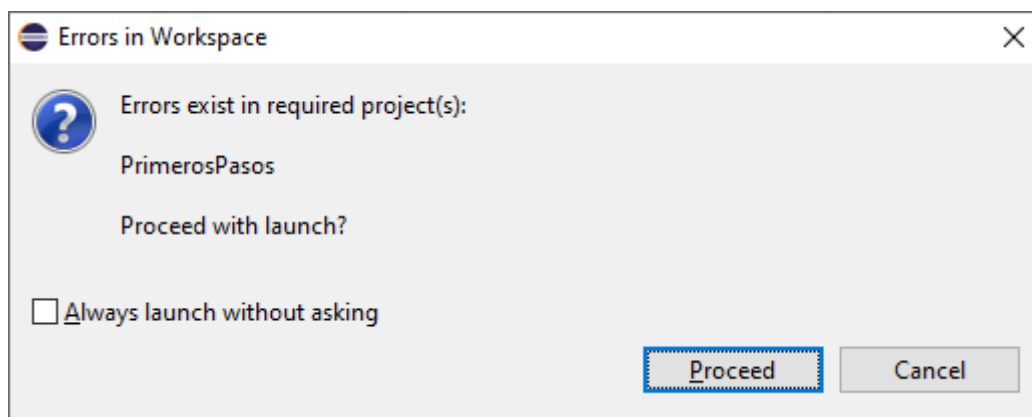


## Estructuras principales V. Constantes y Operadores II (VÍdeo 8)

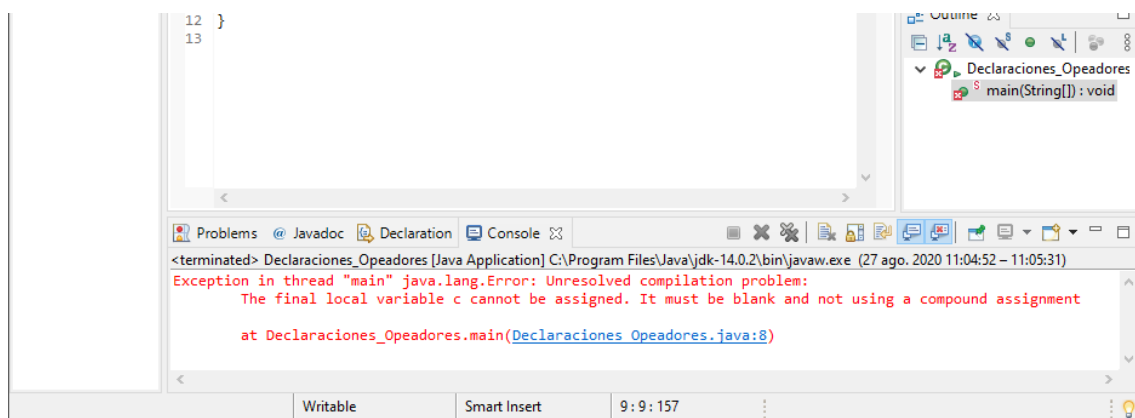
En el siguiente ejemplo vamos a definir una constante esta se declara poniendo al principio la clave final, si en el transcurso del programa intentamos cambiar su valor este nos dará error.



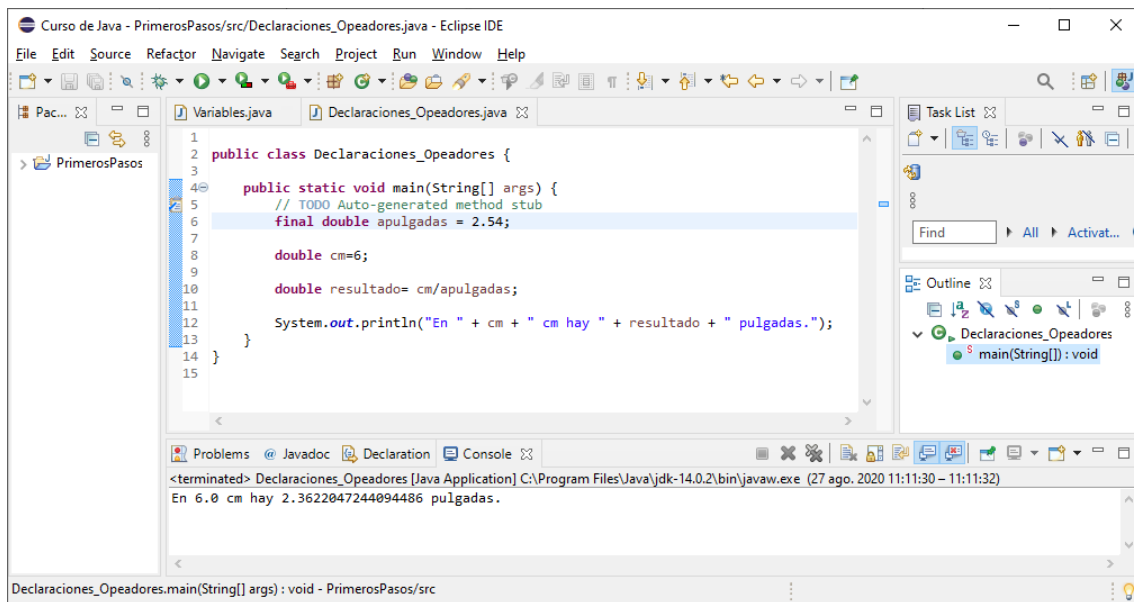
Si ejecutamos.



Nos avisa de que hay un error.



## Ejemplo de constante.



```
1 public class Declaraciones_Opeadores {
2
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         final double apulgadas = 2.54;
7
8         double cm=6;
9
10        double resultado= cm/apulgadas;
11
12        System.out.println("En " + cm + " cm hay " + resultado + " pulgadas.");
13    }
14 }
15
```

<terminated> Declaraciones\_Opeadores [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (27 ago. 2020 11:11:30 - 11:11:32)  
En 6.0 cm hay 2.3622047244094486 pulgadas.

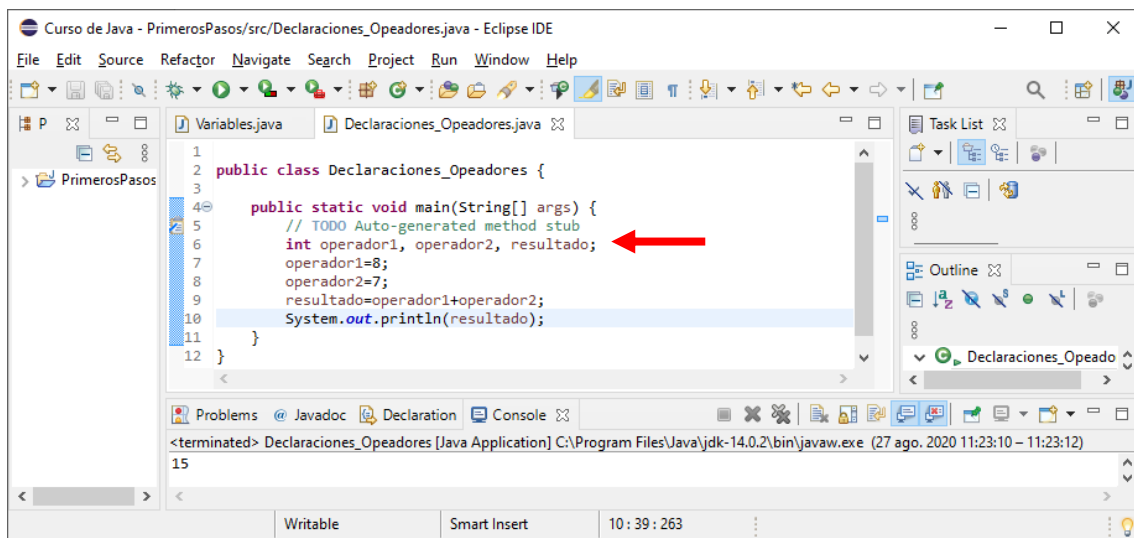
Definimos una constante con la conversión de pulgadas a cms. Que es 2,54 en una constante de tipo double llamada apulgadas.

Definimos una variable de tipo double con el nombre cm para almacenar el valor 6.

En la variable de tipo double llamada resultado calculamos la división del valor que tiene la variable cm con el valor de la constante apulgadas.

Por último enviamos a la consola la concatenación de texto y el valor de las variables.

Se pueden declarar varias variables en una misma línea.



```
1 public class Declaraciones_Opeadores {
2
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         int operador1, operador2, resultado;
7         operador1=8;
8         operador2=7;
9         resultado=operador1+operador2;
10        System.out.println(resultado);
11    }
12 }
13
14
15
```

<terminated> Declaraciones\_Opeadores [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (27 ago. 2020 11:23:10 - 11:23:12)  
15





The image shows the cover of a course titled 'CURSO JAVA'. The background is a dark orange-red gradient with a faint world map. In the top left corner, there is a small orange square with the Eclipse logo. The main title 'CURSO JAVA' is in large, bold, white capital letters. To the right of the title, there is a yellow square with the number '8' in white. Below the title, the text 'ESTRUCTURAS PRINCIPALES DEL LENGUAJE V.' is written in white, followed by 'Declaración de constantes' and 'Operadores' in a smaller white font. In the bottom left corner, the word 'eclipse' is written in a light, lowercase font. In the bottom right corner, the Java logo is displayed in its characteristic blue and orange colors.

## Estructuras principales IV Clase Math (Vídeo 9)

Otros cálculos numéricos.

Clase Math



Hay clase propias y clases predefinidas.

### Clase propia

```
public class Declaraciones_Operadores {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        int operador1,operador2, resultado;  
  
        operador1=8;  
  
        operador2=7;  
  
        resultado=operador1+operador2;  
        System.out.println(resultado);  
  
    }  
}
```

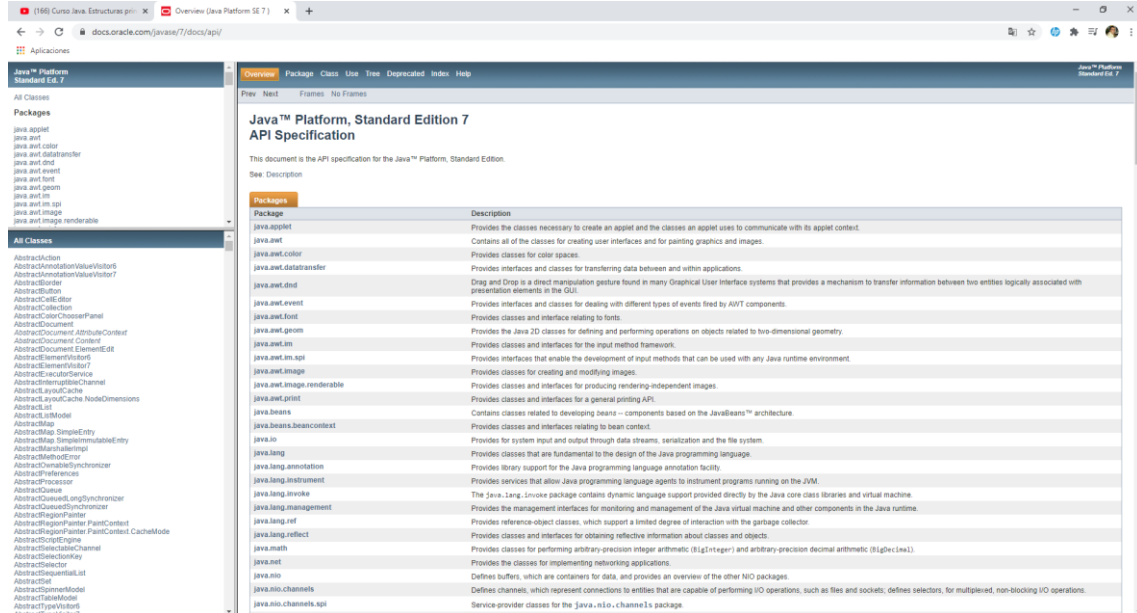
Este es un ejemplo de clase propia.

Ejemplos de predefinidas: String, Array, Math, Thread, etc.

API Java es una biblioteca donde vienen todas las clases de programación, muy útil para realizar consultas.

Accederemos al siguiente enlace:

<https://docs.oracle.com/javase/7/docs/api/>



Con el tiempo tendremos las clases, según las necesidades que surjan, para ello habrá que actualizar la versión de JRE.

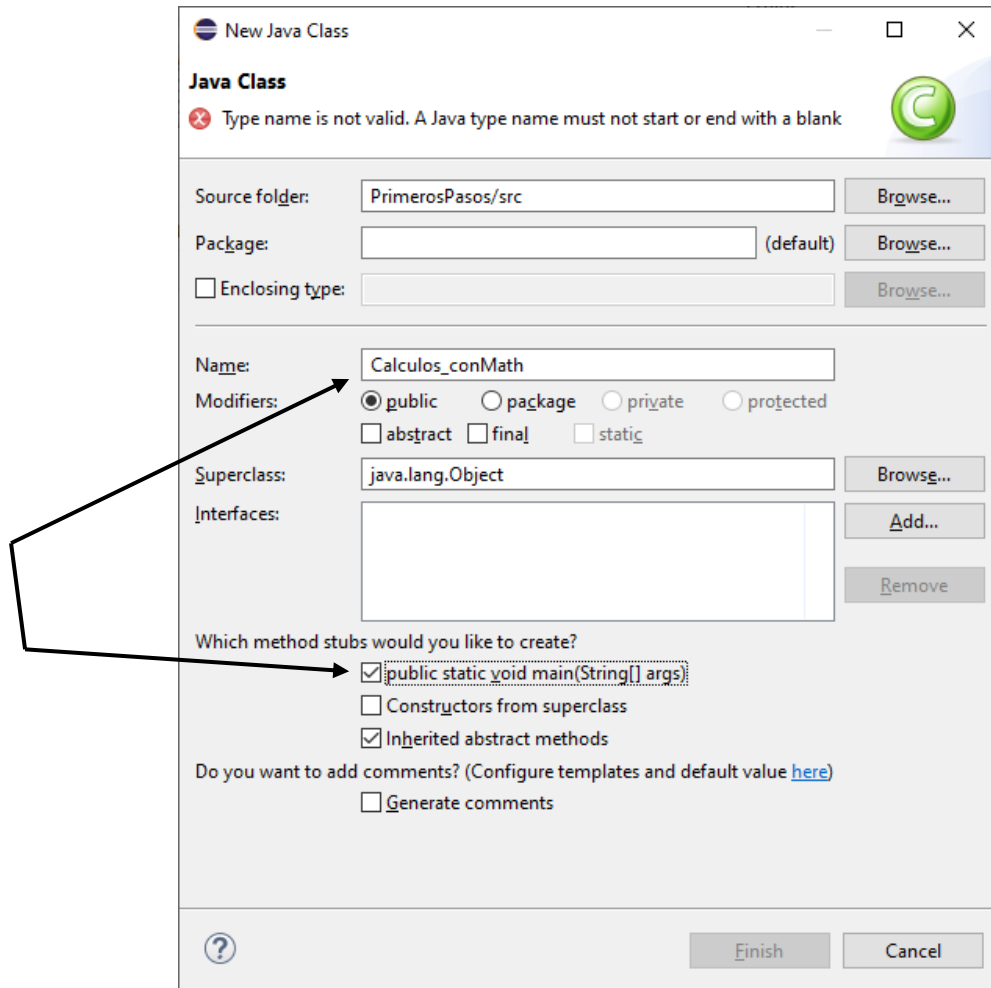
Se puede descargar para tenerla en tu disco duro local, pero si tienen internet siempre la tendrán actualizada.

## Clase Math

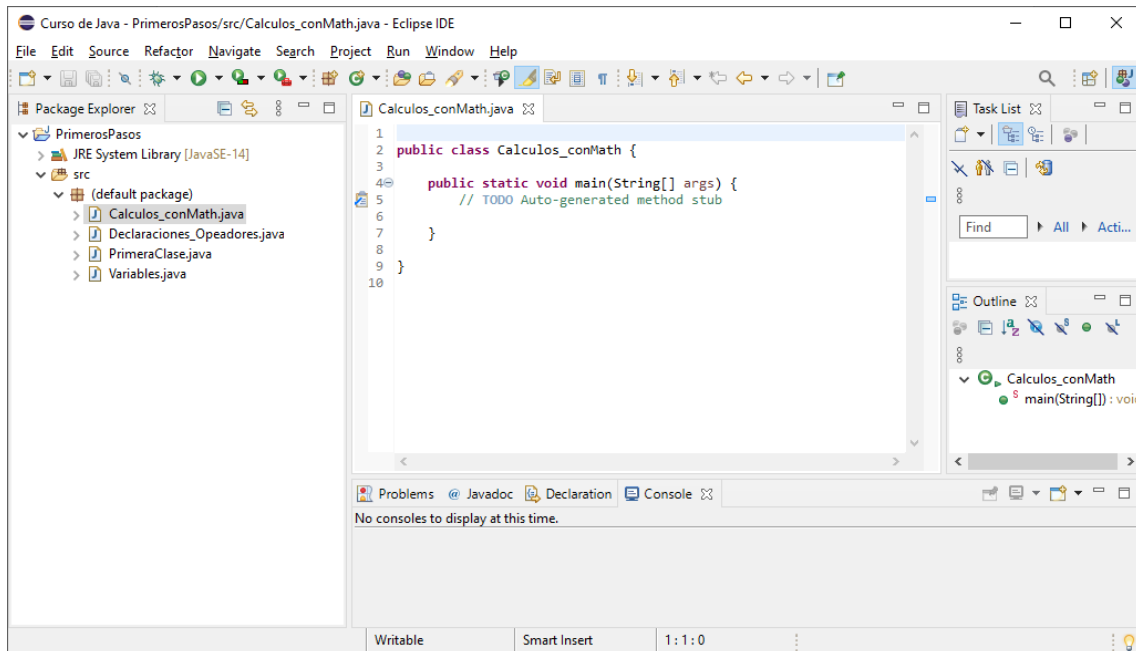
- La clase Math contiene un muestrario de métodos que nos permitirán realizar cálculos matemáticos.
  - `Math.sqrt(n)`: raíz cuadrada de un número.
  - `Math.pow(base, exponente)`: potencia de un número. Base y exponentes son doubles.
  - `Math.sin(ángulo)`. `Math.cos(ángulo)`. `Math.tan(ángulo)`. `Math.atan(/ángulo)`
  - `Math.round(decimal)`: redondeo de un número.
  - `Math.PI`: constante de clase con el número PI

Vamos a Eclipse.

Vamos crear una nueva clase llamada `Calculos_conMath`



Seleccionamos el botón Finish.

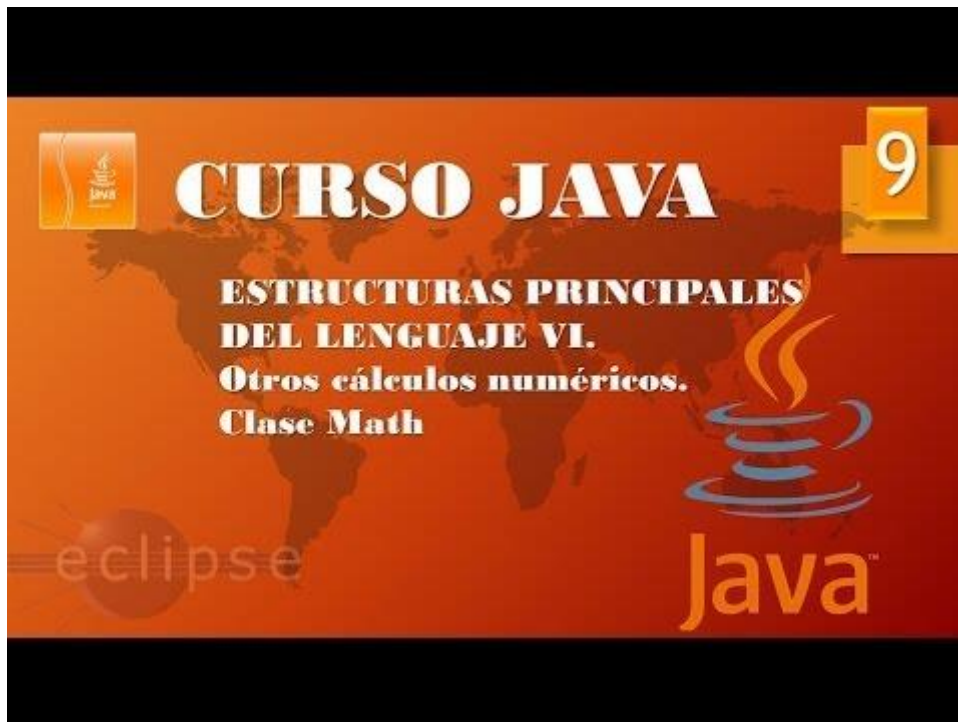


Ejemplo con la clase Math.

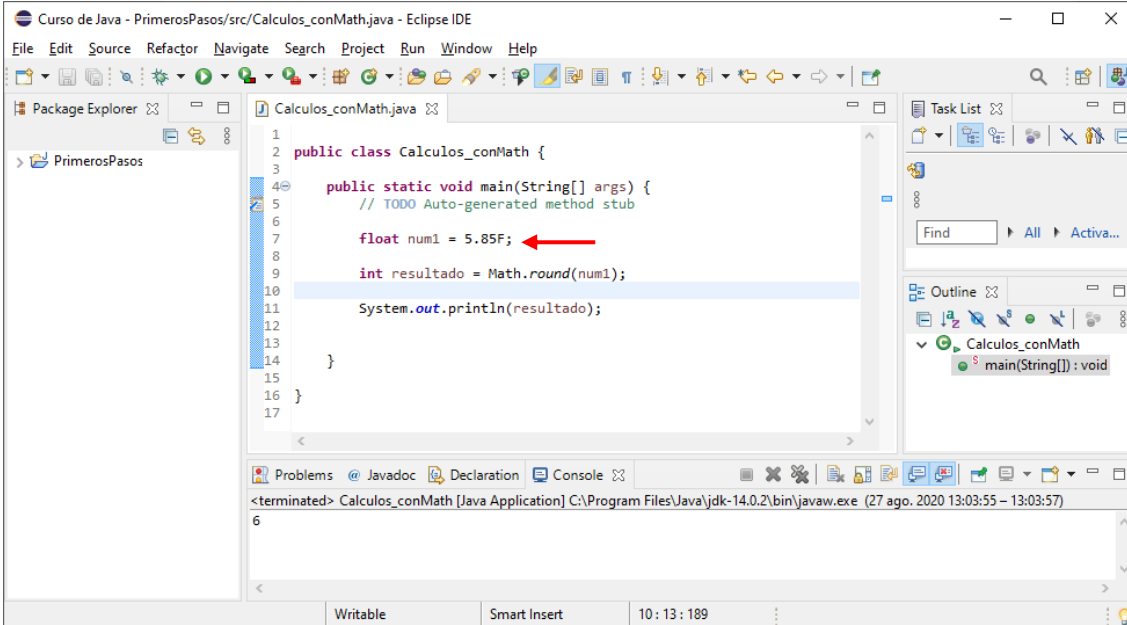
```
1  
2 public class Calculos_conMath {  
3  
4     public static void main(String[] args) {  
5         // TODO Auto-generated method stub  
6  
7         double raiz = Math.sqrt(9.0);  
8  
9         System.out.println(raiz);  
10  
11     }  
12  
13 }  
14  
15
```

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left shows a project named 'PrimerosPasos' with a source folder 'src' containing several Java files, including 'Calculos\_conMath.java'. The main editor displays the code for 'Calculos\_conMath.java'. The Console at the bottom shows the output '3.0' from the execution of the main method.

Tanto el parámetro que le pasamos como en que retorna es de tipo double.



## Estructuras principales VII. Clase Math II. (VÍdeo 10)



```
1 public class Calculos_conMath {
2
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         float num1 = 5.85F;
8
9         int resultado = Math.round(num1);
10
11         System.out.println(resultado);
12
13     }
14 }
15 }
16 }
17 }
```

Console output: 6

En este ejemplo vamos a utilizar `Math.round(float)`, para ello hemos de pasarle un parámetro de tipo `Float` y retorna un valor de tipo `int`.

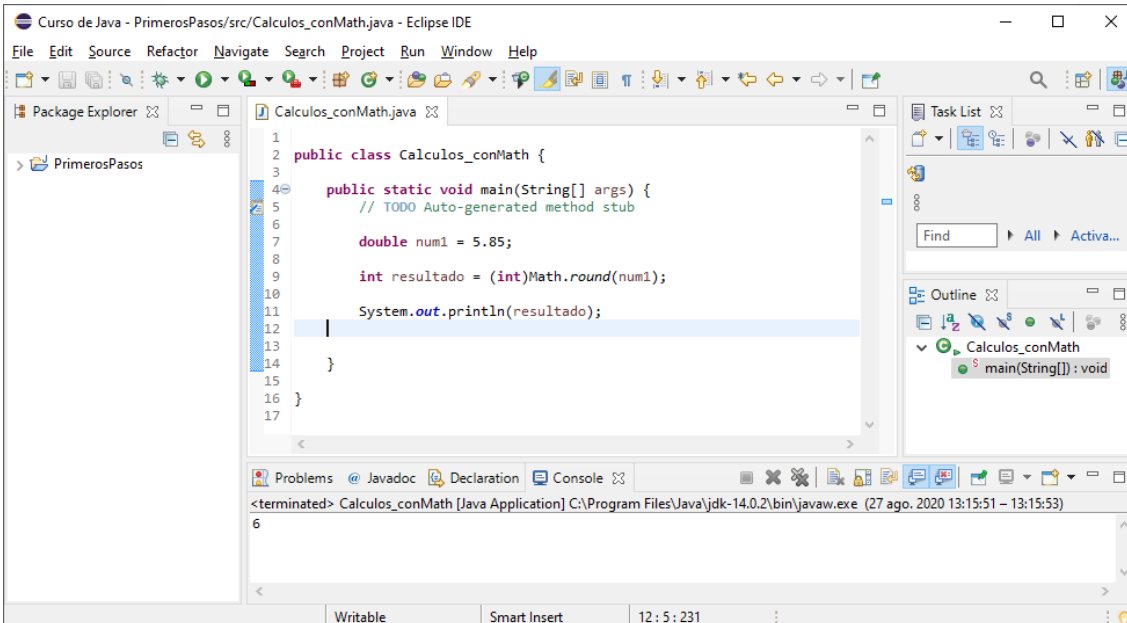
Para asignar una variable de tipo `float` hemos de pasar el parámetro `F`.

`float num1 = 5.85F;`

Refundiciones

`int raiz = (int)Math.round(num1);`

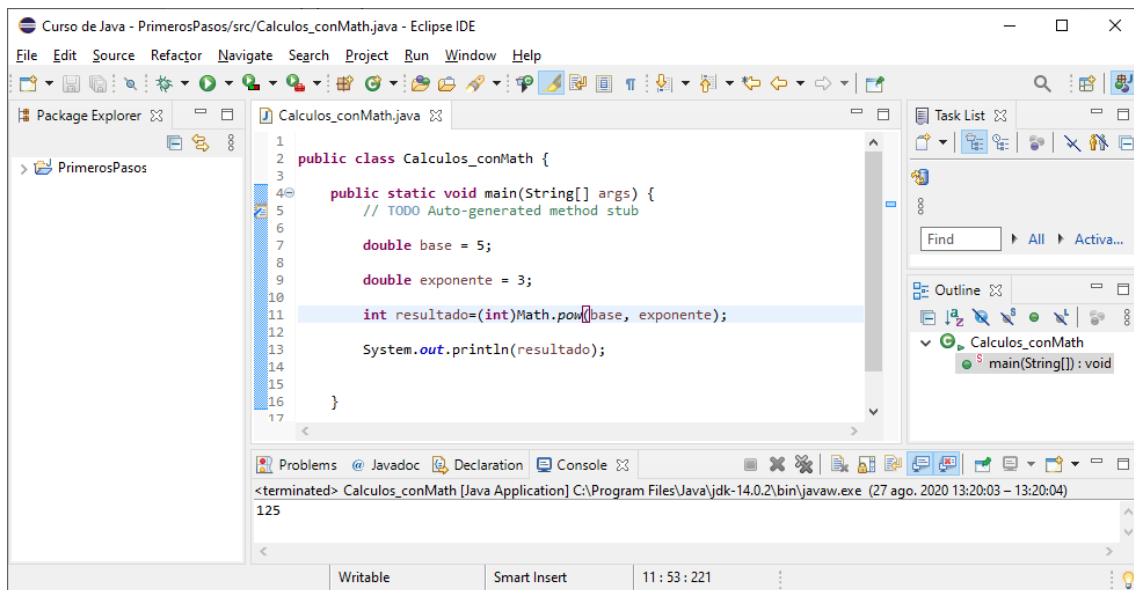
Cuando la clase `Math.round` retorna un valor de tipo `long` y lo queremos asignar a una variable de tipo `int`.



```
1 public class Calculos_conMath {
2
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         double num1 = 5.85;
8
9         int resultado = (int)Math.round(num1);
10
11         System.out.println(resultado);
12
13     }
14 }
15 }
16 }
17 }
```

Console output: 6

Nos ha retornado un 6.



Con `Math.pow(a,b)` se calcula del número `a` elevado a la `b`, retorna un valor `double`, con lo cual haremos como en el ejemplo anterior.

```
int resultado=(int)Math.pow(base, exponente);
```

Otro ejemplo de refundición.

Vamos a concatenar el mensaje de consola.

```
1
2 public class Calculos_conMath {
3
4 public static void main(String[] args) {
5     // TODO Auto-generated method stub
6
7     double base = 5;
8
9     double exponente = 3;
10
11     int resultado=(int)Math.pow(base, exponente);
12
13     System.out.println("El resultado " + " de " + base + " elevado a " + exponente + " es " + resultado);
14
15 }
16
17 }
18 }
```

Este será el resultado:

```
El resultado de 5.0 elevado a 3.0 es 125
```

En la biblioteca de API podremos consultar que tipo de valores hay que poner y que tipo de valor retorna.

The image shows a course cover with a dark orange background and a world map. At the top left is a small Java logo. The main title 'CURSO JAVA' is in large white letters. To its right, the number '10' is in a yellow box. Below the title, the text 'ESTRUCTURAS PRINCIPALES DEL LENGUAJE VII.' is followed by 'Clase Math II' and 'Refundiciones'. At the bottom left is the Eclipse logo, and at the bottom right is the Java logo.

**CURSO JAVA** 10

**ESTRUCTURAS PRINCIPALES  
DEL LENGUAJE VII.  
Clase Math II  
Refundiciones**

eclipse Java



## Manipulación de cadenas. Clase String I (Vídeo 11)

### Clase String

- String no es un tipo primitivo.
- ¿Cómo almacenar una cadena de caracteres?
  - String mi\_nombre="Juan"; donde mi\_nombre es un objeto (instancia, ejemplar) de la clase String
- Métodos (más usado) de la clase String para manipulación de cadena de textos:
  - Lenght(): devuelve la longitud de una cadena de caracteres.
  - CharAt(n): devuelve la posición de un carácter dentro de una cadena. (Las posiciones empiezan a contar de 0)
  - Substring(x,y): devuelve una subcadena dentro de la cadena, siendo x el carácter a partir del cuál se extrae e y e nº de caracteres que se quieren extraer.
  - Equals(cadena): devuelve true si dos cadenas que se comparan son iguales y false si no lo son. Distingue mayúsculas y minúsculas.
  - equalsIgnoreCase(cadena): igual que el anterior pero sin tener en cuenta mayúsculas y mminúsculas.

Vamos a crear una nueva clase llamada manipula\_cadena.

**New Java Class**

**Java Class**  
⚠ The use of the default package is discouraged.

Source folder:

Package:  (default)

Enclosing type:

Name:

Modifiers:  public  package  private  protected  
 abstract  final  static

Superclass:

Interfaces:

Which method stubs would you like to create?

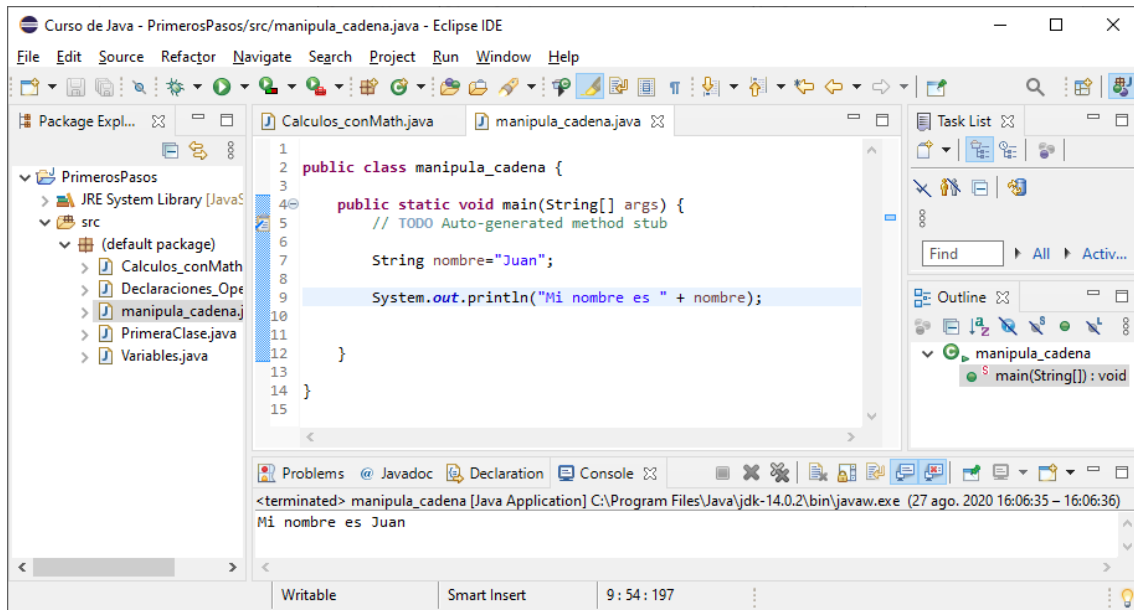
`public static void main(String[] args)`

Constructors from superclass

Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

Generate comments



En el objeto nombre almacenamos la cadena “Juan” que concatenamos para imprimirlo por la consola.

```
1
2 public class manipula_cadena {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         String nombre="Juan";
8
9         System.out.println("Mi nombre es " + nombre);
10
11         System.out.println("Mi nombre tine " + nombre.length()+ " letras.");|
12
13
14     }
15
16 }
17
```

Este será el resultado:

```
Mi nombre es Juan
Mi nombre tine 4 letras.
```

```

1
2 public class manipula_cadena {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         String nombre="Juan";
8
9         System.out.println("Mi nombre es " + nombre);
10
11        System.out.println("Mi nombre tine " + nombre.length()+ " letras.");
12
13        System.out.println("Mi nombre empieza por " + nombre.charAt(0));
14
15
16    }
17
18 }
19

```

Este será el resultado:

```

Mi nombre es Juan
Mi nombre tine 4 letras.
Mi nombre empieza por J

```

```

1
2 public class manipula_cadena {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         String nombre="Juan";
8
9         System.out.println("Mi nombre es " + nombre);
10
11        System.out.println("Mi nombre tine " + nombre.length()+ " letras.");
12
13        System.out.println("Mi nombre empieza por " + nombre.charAt(0));
14
15        System.out.println("Mi nombre termia por " + nombre.charAt(nombre.length()-1));
16
17
18    }
19
20 }
21

```

Este será el resultado:

```

Mi nombre es Juan
Mi nombre tine 4 letras.
Mi nombre empieza por J
Mi nombre termia por n

```



The image shows a course cover with a dark orange background and a world map. At the top left is a small logo with the text '100% JAVA'. The main title 'CURSO JAVA' is in large white letters. To the right, a yellow box contains the number '11'. Below the title, the text 'MANIPULACIÓN DE CADENAS' and 'Clase String' is displayed. At the bottom left is the 'eclipse' logo, and at the bottom right is the 'Java' logo with its characteristic blue and orange waves.

## Manipulación de cadenas. Clase String II (Vídeo 12)

The screenshot shows the 'New Java Class' dialog box. At the top, it says 'Java Class' and 'The use of the default package is discouraged.' Below this, there are fields for 'Source folder' (PrimerosPasos/src), 'Package' (default), and 'Enclosing type'. The 'Name' field contains 'manipula\_Cadena\_II'. Under 'Modifiers', 'public' is selected. The 'Superclass' is 'java.lang.Object'. Under 'Which method stubs would you like to create?', 'public static void main(String[] args)' and 'Inherited abstract methods' are checked. At the bottom, there are 'Finish' and 'Cancel' buttons.

Creamos un nuevo proyecto llamado `manipula_Cadena_II`.

Queremos extraer de una cadena parte de ella.

```
1
2 public class manipula_Cadena_II {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         String frase = "Hoy es un estupendo día para aprender a programar en Java";
8         String frase_resumen = frase.substring(29, 57);
9         System.out.println(frase_resumen);
10
11     }
12
13 }
14
```

Este será el resultado:

```
aprender a progamar en Java
```

Ahora en el siguiente ejemplo hacemos además una concatenación.

```
1
2 public class manipula_Cadena_II {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         String frase ="Hoy es un estupendo día para aprender a programar en Java";
8         String frase_resumen =frase.substring(0, 28) + " irnos a la playa";
9         System.out.println(frase_resumen);
10
11     }
12
13 }
14
```

Este será el resultado:

Hoy es un estupendo día para irnos a la playa

Para hacer un salto de línea solo hay que hacer intro.

```
1
2 public class manipula_Cadena_II {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         String frase ="Hoy es un estupendo día para aprender a programar en Java";
8         String frase_resumen =frase.substring(0, 28) + " irnos a la playa... " +
9         " y " + frase.substring(29, 57);
10        System.out.println(frase_resumen);
11
12    }
13
14 }
15
```

Este será el resultado:

Hoy es un estupendo día para irnos a la playa... y aprender a programar en Java

Vamos a crear otra clase llamada manipula\_cadena\_III.

```
1
2 public class manipula_cadena_III {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         String alumno1, alumno2;
8
9         alumno1="David";
10
11        alumno2="David";
12
13        System.out.println(alumno1.equals(alumno2));
14
15    }
16
17 }
18
```

En este ejemplo queremos comparar si las variables alumnos1 y alumno2 contienen la misma información, si es afirmativo retornará true y sino false.

Este será el resultado:

True

Si queremos que no distinga entre mayúsculas y minúsculas, este será el ejemplo:

```
1
2 public class manipula_cadena_III {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         String alumno1, alumno2;
8
9         alumno1="DAVID";
10
11        alumno2="david";
12
13        System.out.println(alumno1.equalsIgnoreCase(alumno2));
14
15    }
16
17 }
```

Este será el resultado:

true

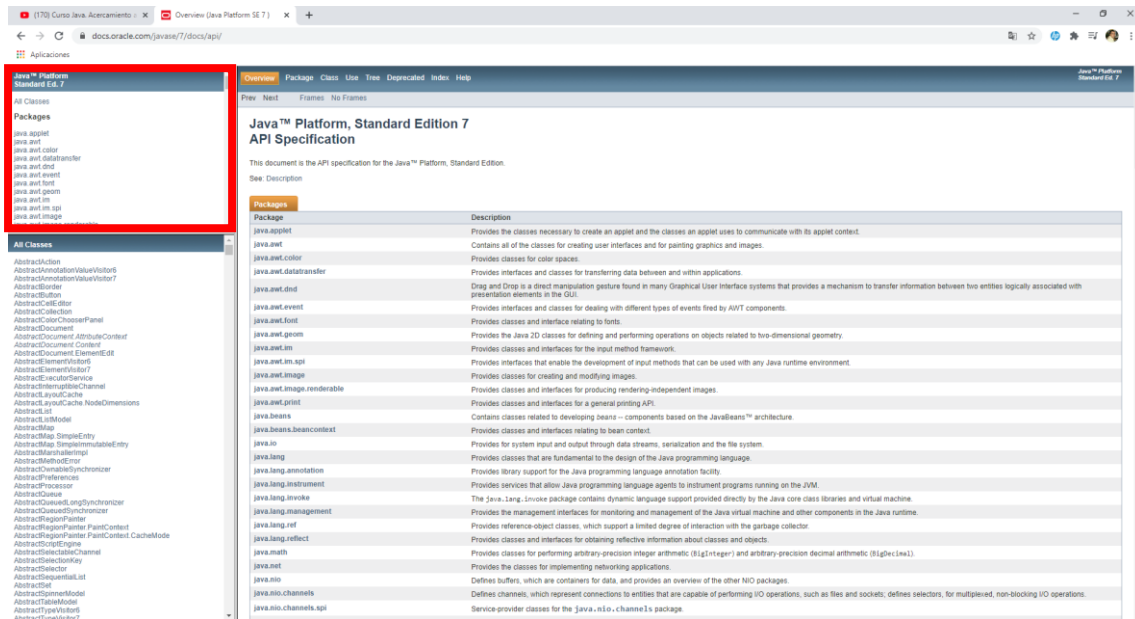


## Acercamiento a la API Paquetes (VÍdeo 13)

Las API de Java son las bibliotecas de clase que vienen de forma predefinida, para que las podamos utilizar en nuestros programas.

En Java tenemos las clases predefinidas y las propias.

Las clases de Java a las carpetas se les denomina paquetes.

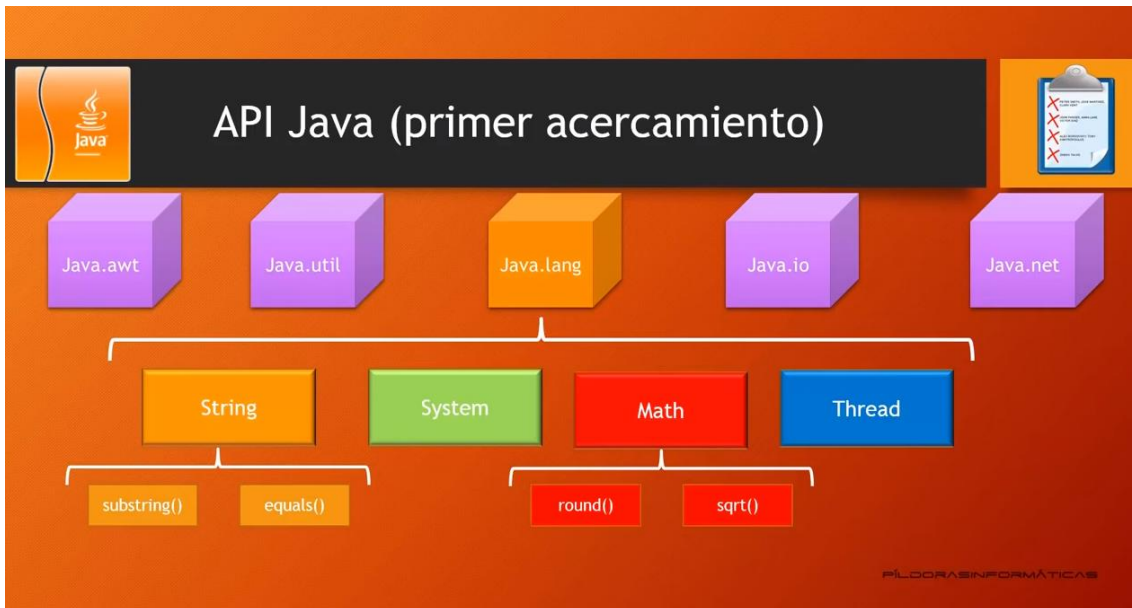


La parte superior izquierda se denomina panel de paquetes.



- ¿Por qué son necesarios los paquetes? ¿Por qué se inventaron?
  - Para organizar las clases
  - Para evitar conflictos de nombres
  - Para controlar la visibilidad de las clases

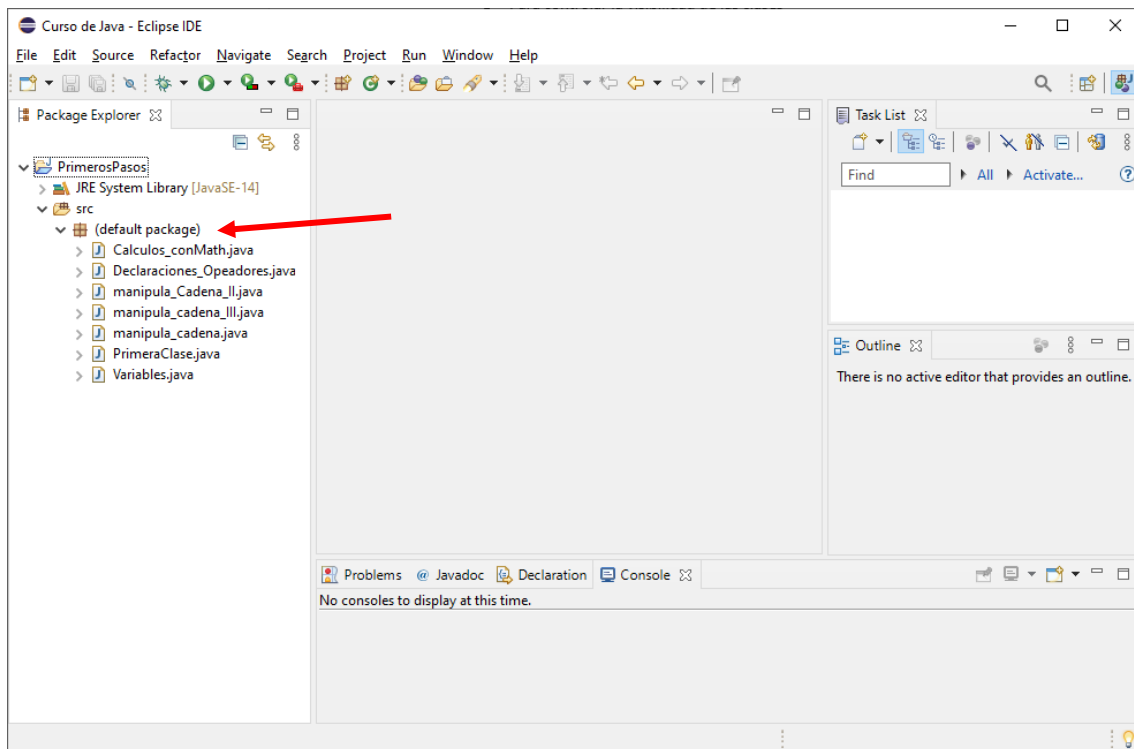




Java.lang: Es el paquete principal que a su vez por estos motivos hemos tenido acceso a las clases String, System, Math, Thread, así como a sus métodos sin ningún problema.

Si quieres utilizar otro paquete, por ejemplo Java.util que no se encuentre en Java.lang tendrás que indicárselo previamente.

Vamos a ejecutar Eclipse.



Lo que señalamos con una flecha en un paquete.

Cuando tengamos que utilizar una clase que no pertenece al paquete principal, habrá que importarla.

```

1 import java.util.*; ←
2 public class Prueba {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         Scanner miobjeto;
8
9     }
10
11 }
12

```

Al agregar un \* al final, le estamos diciendo que importe todas las clases del paquete java.util.

Si del paquete java.util solo utilizarás la clase Scanner

```

1 import java.util.Scanner; ←
2 public class Prueba {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         Scanner miobjeto;
8
9     }
10
11 }
12

```

Si utilizar el \* consumes más recursos porque le estás diciendo que te importe todas las clases en la memoria que si le dices que te importe solo una clase.

```

1 import java.util.*;
2 public class Prueba {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         Scanner miobjeto;
8
9         Locale miobjeto2;
10
11     }
12
13 }
14

```

Al utilizar el \* podemos utilizar todas las clases del paquete java.util.

```

1 import java.util.Scanner;
2 import java.util.Locale;
3
4 public class Prueba {
5
6     public static void main(String[] args) {
7         // TODO Auto-generated method stub
8
9         Scanner miobjeto;
10
11        Locale miobjeto2;
12
13    }
14
15 }
16

```

Importamos las clases Scanner y Locate del paquete java.util, es mucho más practico importar con el \* y ahorramos código.

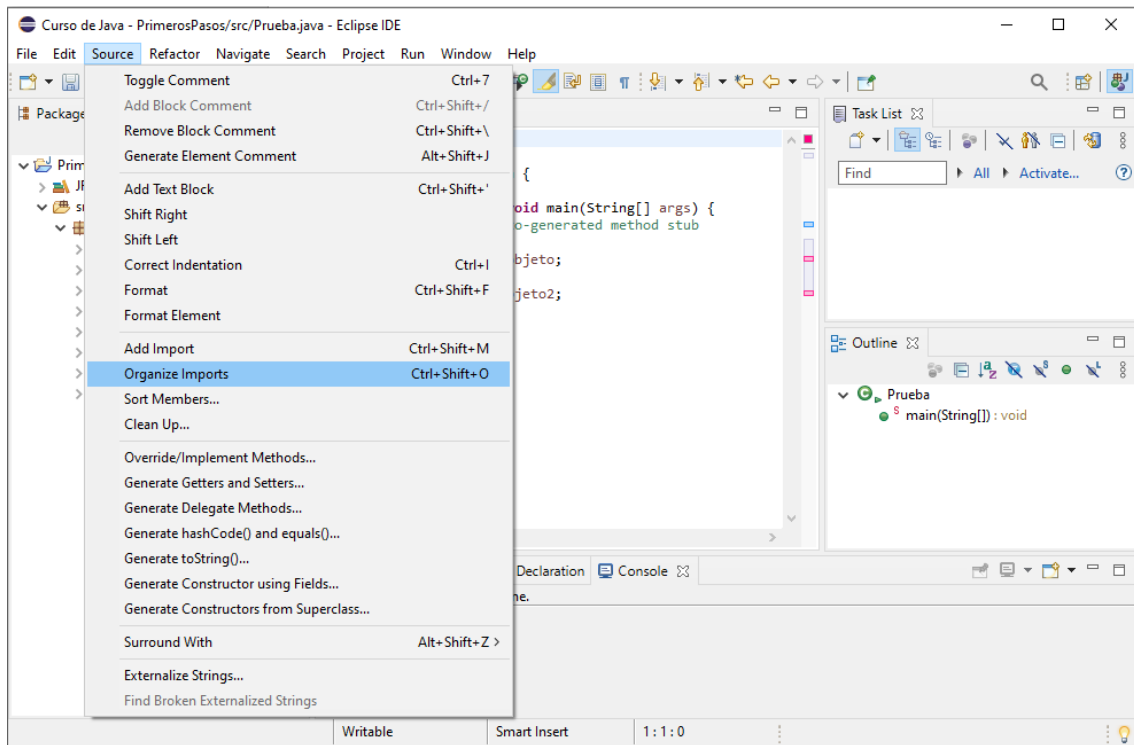
Supongamos que estamos trabajando con dos clases y no sabemos a que paquete pertenece.

```

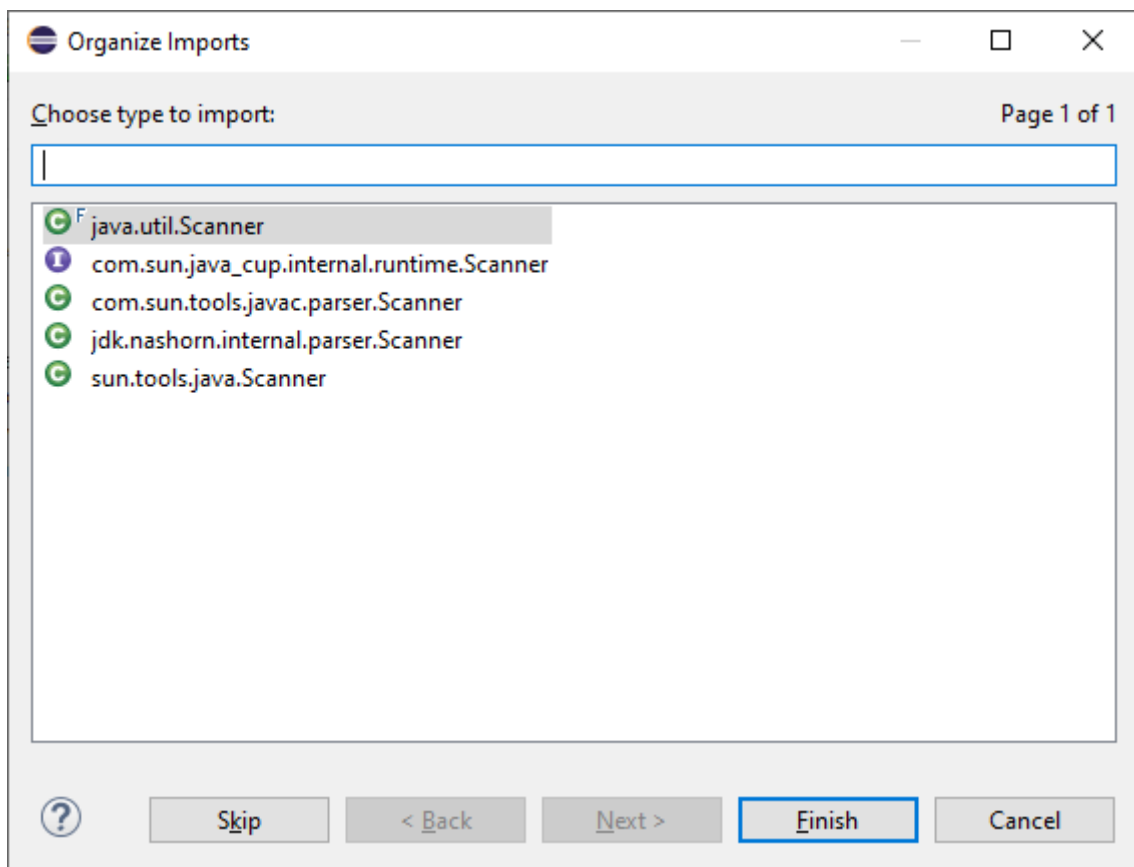
1
2
3 public class Prueba {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7
8         Scanner miobjeto;
9
10        Locale miobjeto2;
11
12    }
13
14 }
15

```

Observamos dos errores porque no las reconoce, pero además no sabemos a que paquete pertenece, para ello vamos a realizar los siguientes pasos:



Del menú Source seleccionaremos Organize Imports.



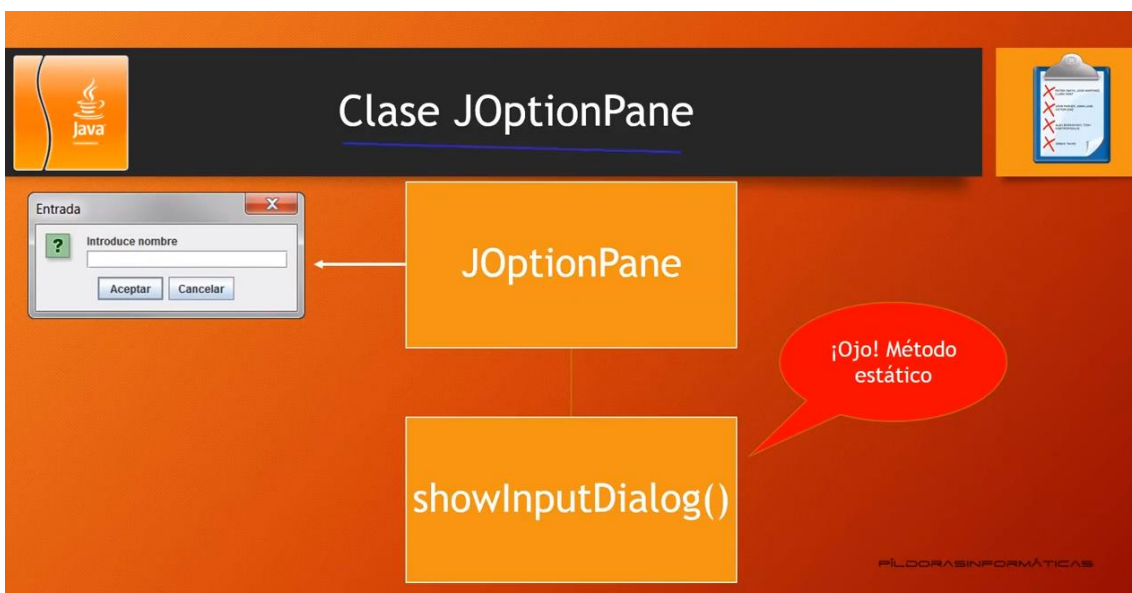
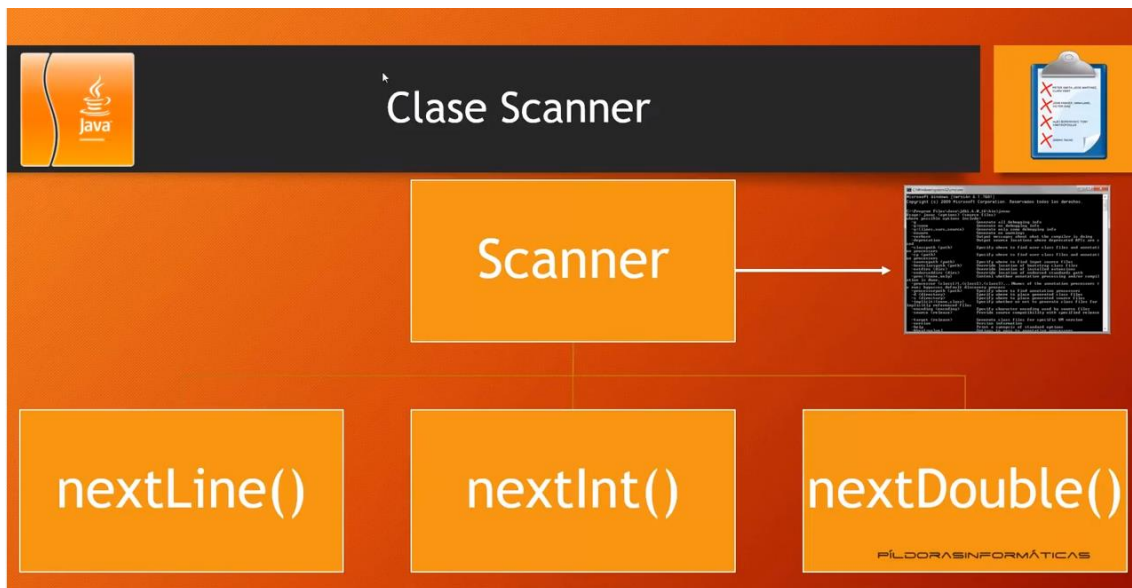
Seguido del botón Finish.

```
1 import java.util.Locale;
2 import java.util.Scanner;
3
4 public class Prueba {
5
6     public static void main(String[] args) {
7         // TODO Auto-generated method stub
8
9         Scanner miobjeto;
10
11        Locale miobjeto2;
12
13    }
14
15 }
16
```

Ya hemos importado las clases Locale y Scanner del paquete java.util.



## Entrada Salida datos I (Vídeo 14)

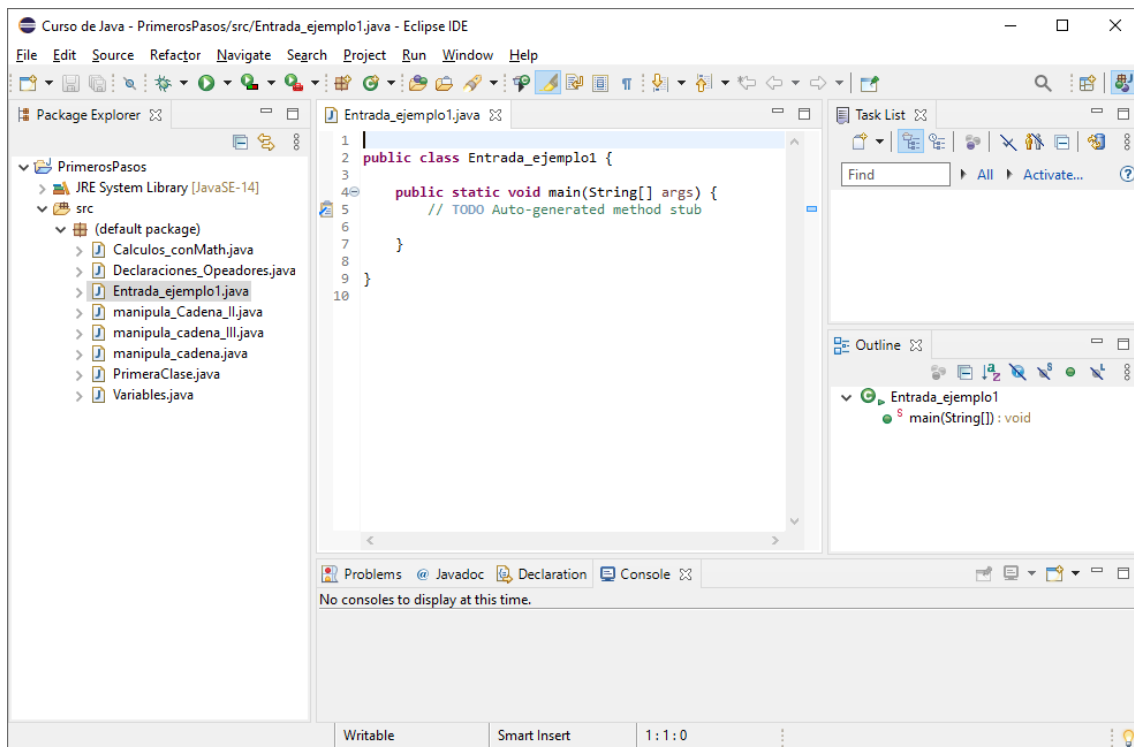
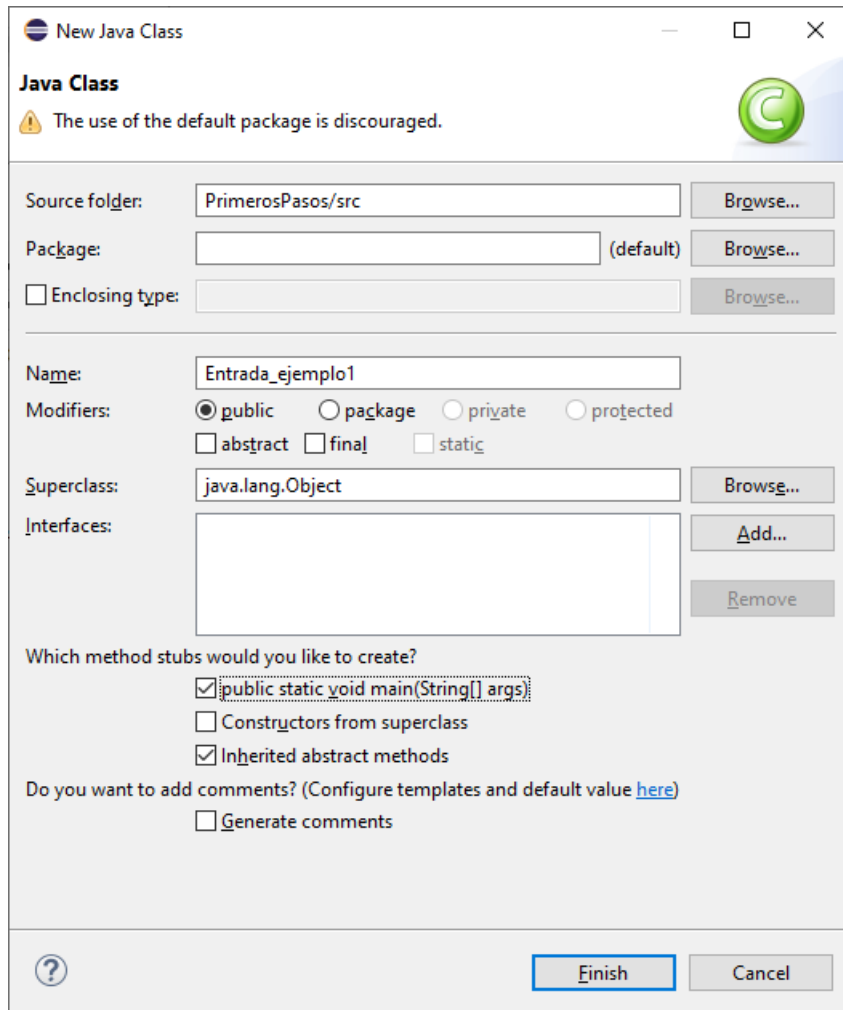


Un método es estático cuando delante del método hay que poner la clase separada por punto.

Los métodos que no son estáticos no vamos a utilizar el nombre de la clase, para ello hemos de crear un objeto de la clase Scanner.

Vamos a Eclipse.

Vamos a crear una nueva clase llamada Entrada\_ejemplo1.



Constructor: Tiene el mismo nombre que a la clase que pertenece.

Vamos a escribir el siguiente código:

```
1 import java.util.*;
2 public class Entrada_ejemplo1 {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         Scanner entrada=new Scanner(System.in);
7         System.out.println("Introduce tu nombre, por favor: ");
8         String nombre_usuario = entrada.nextLine();
9         System.out.println("Introduce edad, por favor: ");
10        int edad = entrada.nextInt();
11        System.out.println("Hola " + nombre_usuario + " el año que viene tendrás " +
12            (edad + 1) + " años.");
13    }
14
15 }
```

En la línea 1 importamos el paquete java.util, con el \* todas sus clases.

En la línea 6 definimos un objeto llamado entrada de tipo Scanner, que nos permite realizar entradas por la consola.

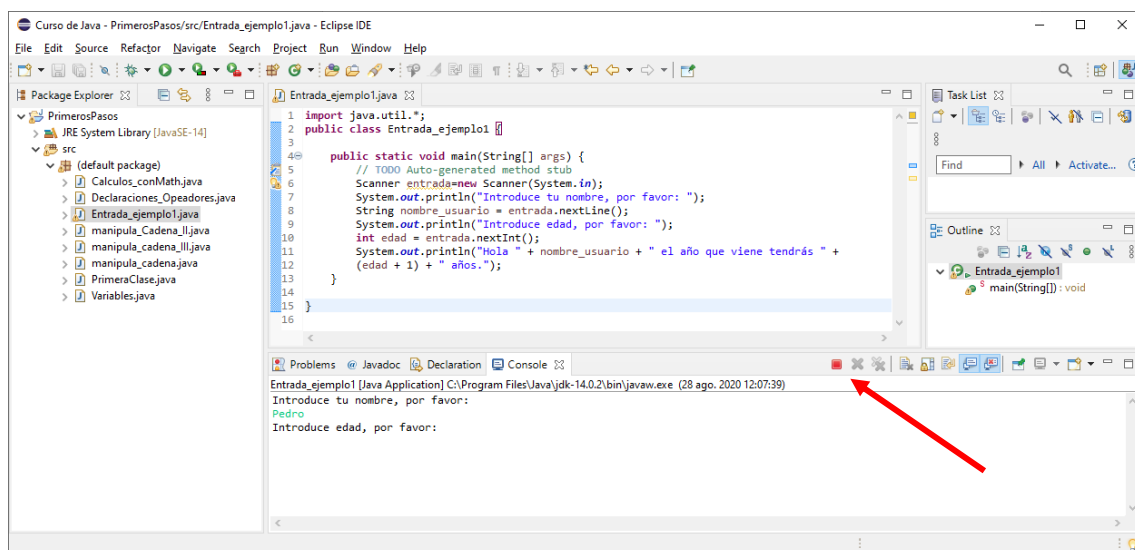
En la línea 8 definimos una variable de tipo String asignándole el objeto entrada que es de la clase Scanner para que la consola se detenga esperando que introduzcamos un datos de tipo string.

En la línea 10 definimos una variable de tipo int asignándole el objeto entrada para que la consola se detenga en espera que introduzcas un valor de tipo int.

Si ejecutamos este será el resultado:

```
Introduce tu nombre, por favor:
Pedro
Introduce edad, por favor:
60
Hola Pedro el año que viene tendrás 61 años.
```

Observarás que durante la ejecución del programa observamos el siguiente cuadrado en rojo.



Nos indica que el programa está en ejecución, cuando termine se volverá de color gris.



A slide with an orange-to-red gradient background. At the top left is a small Java logo. The main title 'CURSO JAVA' is in large white letters. To its right, the number '14' is in a yellow box. Below the title, the text 'ENTRADA Y SALIDA DE DATOS. FORMATO DE RESULTADOS' is centered. The background features a faint world map, the Eclipse logo, and the Java logo with a coffee cup icon.

**CURSO JAVA** 14

**ENTRADA Y SALIDA DE DATOS.  
FORMATO DE RESULTADOS**

eclipse Java

## Entrada Salida datos II. (Vídeo 15)

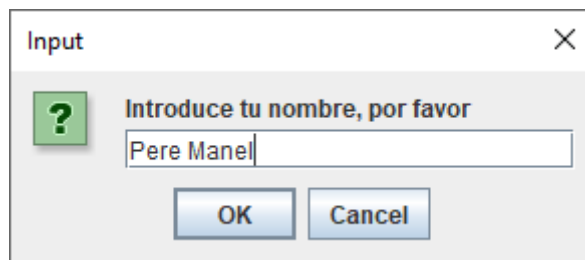
Vamos a ejecutar Eclipse y vamos a crear otra clase llama Entrada\_ejemplo2.

```
1 import javax.swing.*;
2 public class Entrada_ejemplo2 {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         String Nombre_usuario=JOptionPane.showInputDialog("Introduce tu nombre, por favor");
7         System.out.println("Tu nombre es " + Nombre_usuario + ".");
8     }
9
10 }
```

En la línea 1 importamos todas las clases del paquete javax.swing.

En la línea 6 declaramos una variable de tipo String llamada Nombre\_usuario que le asignaremos el dato introducido en la ventana de diálogo que se abrirá, que nos preguntará “Introduce tu nombre, por favor”.

En la línea 7 imprimiremos por consola tu nombre es + el nombre introducido en la ventana de diálogo.



Le damos al botón Ok, y este será el resultado:

```
|Tu nombre es Pere Manel.
```

Ahora además que nos pregunte por la edad.

```
1 import javax.swing.*;
2 public class Entrada_ejemplo2 {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         String Nombre_usuario=JOptionPane.showInputDialog("Introduce tu nombre, por favor");
7         String edad = JOptionPane.showInputDialog("Introduce tu edad, por favor");
8
9         System.out.println("Tu nombre es " + Nombre_usuario + " y tienes " +
10             edad + " años");
11     }
12
13 }
14 }
```

Creamos variable de tipo String que retorna un valor de este tipo.

```

1 import javax.swing.JOptionPane;
2 public class Entrada_ejemplo2 {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         String Nombre_usuario=JOptionPane.showInputDialog("Introduce tu nombre, por favor");
7         String edad = JOptionPane.showInputDialog("Introduce tu edad, por favor");
8         int edad1 = Integer.parseInt(edad);
9
10        System.out.println("Tu nombre es " + Nombre_usuario + " el año que viene tendrás | " +
11        (edad1 + 1) + " años");
12    }
13
14 }
15

```

En la línea 8 hacemos que la variable edad1 asuma el valor como un dato int, ya que en su momento en que introdujéramos una edad esta era almacenada de tipo String.

El método parseInt y observamos que es estático tendré que utilizar el nombre de la clase como argumento: Integer.parseInt(edad);

El resultado será:

Tu nombre es Pere Manel el año que viene tendrás 61 años

Vamos a utilizar el operador incremento.

```

1 import javax.swing.JOptionPane;
2 public class Entrada_ejemplo2 {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         String Nombre_usuario=JOptionPane.showInputDialog("Introduce tu nombre, por favor");
7         String edad = JOptionPane.showInputDialog("Introduce tu edad, por favor");
8         int edad1 = Integer.parseInt(edad);
9
10        edad1++; ←
11
12        System.out.println("Tu nombre es " + Nombre_usuario + " el año que viene tendrás " +
13        edad1 + " años");
14    }
15
16 }

```

Lo que hace es incrementar su valor en 1.

¿Cómo formatear los resultados?

```

1
2 public class Entrada_Numeros {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         double x=10000.0;
8
9         System.out.printf("%.2f", x/3);
10    }
11
12 }

```

En la línea 9 hacemos un printf de format y en comillas le decimos con el “%.2f”, que al número que viene a continuación le asignemos un formato de 2 decimales, este será el resultado:

3333,33

```

1 import javax.swing.*;
2 public class Entrada_Numeros {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         String num1=JOptionPane.showInputDialog("Introduce un número: ");
8
9         Double num2=Double.parseDouble(num1);
10
11         System.out.print("La raiz del número " + num1 + " es ");
12
13         System.out.printf("%.2f", Math.sqrt(num2));
14
15     }
16
17 }

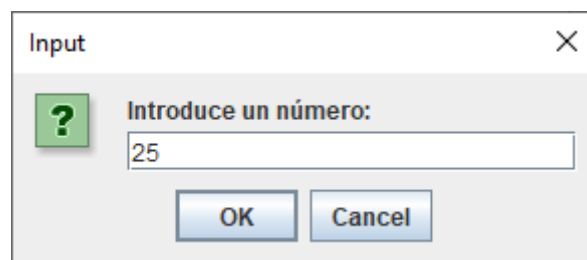
```

En el siguiente ejemplo en la línea 9 hacemos que un valor de tipo string pase a ser de tipo Double.

En la línea 11 hacemos un print este no hace salto de línea, con lo cual el siguiente print lo hará en la misma línea.

En la línea 13 hacemos un printf para darle formato de 2 decimales.

Si introducimos el valor 25 este será el resultado:



La raiz del número 25 es 5,00





## Condicionales I. (Vídeo 16)

Control de flujo.

Condicionales y bucles.



The slide features a dark blue header with the Java logo on the left and a clipboard icon on the right. The main content area is orange and contains a code snippet for a `main` method. A large white arrow points downwards from the code, indicating the flow of execution. The code is as follows:

```
• Public static void main (String args[]){  
  • Línea código 1  
  • Línea código 2  
  • Línea código 3  
  • Línea código 4  
  • Línea código 5  
• }
```

Para el salto usamos el condicional y para repetir unas líneas de código x veces utilizamos el bucle.

Condicionales

- Condicional if
  - `If(condición){`
    - Código a ejecutar si la condición es verdadera (true);
  - `}`
- Condicional switch
  - `Switch (valor a evaluar){`
    - Case valor1:
      - Código a ejecutar;
      - Break;
    - Case valor 2:
      - Código a ejecutar;
      - Break;

El condicional if puede llevar otra estructura que es else.

El condicional switch puede llevar otra estructura que es default.

Vamos a ejecutar Eclipse con una nueva clase llamada `Evalua_edad`.

```

1 import java.util.*;
2 public class Evalua_edad {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         Scanner entrada=new Scanner(System.in);
7         System.out.println("Introduce tu edad, por favor");
8         int edad=entrada.nextInt();
9
10        if(edad>=18) {
11            System.out.println("Eres mayor de edad");
12        }
13        else {
14            System.out.println("No eres mayor de edad");
15        }
16
17    }
18
19 }

```

En la línea 10 estamos comparando si la variable edad es mayor o igual a 18, si es así ejecutará el código que se encuentra entre llaves después del if, de lo contrario si no eres mayor o igual a 18 se ejecutará lo que viene después del else que se encuentra entre llaves.

Vamos a ver ahora la instrucción else if.

```

1 import java.util.*;
2 public class Evalua_edad {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         Scanner entrada=new Scanner(System.in);
7         System.out.println("Introduce tu edad, por favor");
8         int edad=entrada.nextInt();
9
10        if(edad<18) {
11            System.out.println("Eres un adolescente");
12        }
13        else if(edad<40){
14            System.out.println("Eres joven");
15        }
16        else if(edad<65){
17            System.out.println("Eres maduro");
18        }
19        else{
20            System.out.println("Cuidate");
21        }
22    }
23 }

```

En la línea 10 compara si edad es menor de 18 si es así ejecuta lo que se encuentra entre llaves, si no es así compara si edad es menor de 40 si es así ejecuta lo siguiente que encuentra entre llaves, si no es así compara si edad es menor de 65 ejecuta el código siguiente pero else (si no se cumple nada de lo anterior, ejecuta la siguiente línea.

Ejecuta el programa con diferentes edades.



The image shows the cover of a course titled "CURSO JAVA". The background is a dark orange-red gradient with a faint world map. In the top left corner, there is a small orange square icon with a white Java logo. The main title "CURSO JAVA" is written in large, bold, white capital letters. To the right of the title, the number "16" is displayed in white inside an orange square. Below the title, the text "FLUJO DE CONTROL. CONDICIONALES Y BUCLES" is written in smaller, bold, white capital letters. In the bottom left corner, the word "eclipse" is written in a light, lowercase font. In the bottom right corner, the Java logo is displayed in its characteristic blue and orange colors, with the word "Java" written in orange below it.



## Condicionales II (Vídeo 17)

Vamos a aprender como funciona la estructura Switch.

Abrimos Eclipse.

Vamos a crear un nuevo proyecto para que nos calcule el área de un cuadrado, rectángulo, triángulo y círculo.

Vamos a crear una clase nueva llamada Áreas.

```
1 import java.util.*;
2 import javax.swing.*;
3
4 public class Areas {
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 "
10             + "\n2: Rectángulo \n3: Triángulo \n4: Círculo");
11         int figura=entrada.nextInt();
12
13         switch(figura) {
14             case 1:
15                 int lado=Integer.parseInt(JOptionPane.showInputDialog("Introduce el lado:"));
16                 System.out.println("El área del cuadro es " + Math.pow(lado, 2));
17                 break;
18             case 2:
19                 int base=Integer.parseInt(JOptionPane.showInputDialog("Introduce la base del rectángulo:"));
20                 int altura=Integer.parseInt(JOptionPane.showInputDialog("Introduce la altura del rectángulo:"));
21                 System.out.println("El área del rectángulo es " + base * altura);
22                 break;
23             case 3:
24                 base=Integer.parseInt(JOptionPane.showInputDialog("Introduce la base del rectángulo:"));
25                 altura=Integer.parseInt(JOptionPane.showInputDialog("Introduce la altura del rectángulo:"));
26                 System.out.println("El área del triángulo es " + (base * altura)/2);
27                 break;
28             case 4:
29                 double radio=Double.parseDouble(JOptionPane.showInputDialog("Introduce el radio de un círculo:"));
30                 System.out.print("El area del círculo es ");
31                 System.out.printf("%.2f", (Math.PI * Math.pow(radio, 2)));
32                 break;
33             default:
34                 System.out.println("El número introducido es erróneo");|
```

Ahora lo ejecutas y pruebas todas las opciones de cálculo.





## Bucles I (VÍdeo 18)

- Bucles indeterminados
  - While
  - Do - while
- Bucles determinados
  - For
  - For - each

### Bucle while

- Sintaxis:
  - While (condición){
    - Línea1
    - Línea2
    - Línea3
    - Línea4
  - }

Vamos a realizar un proyecto en el que va a pedir una contraseña al usuario y mientras no la acierte la irá preguntado.

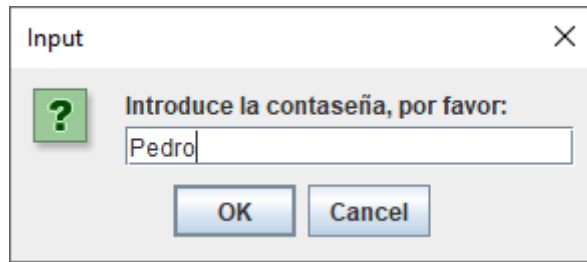
Vamos a crear una clase nueva llamada Acceso\_aplicacion.

```
1 import javax.swing.*;
2 public class Acceso_aplicacion {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         String clave="Juan";
7         String pass="";
8         while(clave.equals(pass)==false) {
9             pass=JOptionPane.showInputDialog("Introduce la contraseña, por favor:");
10            if(clave.equals(pass)==false) {
11                System.out.println("Contraseña incorrecta.");
12            }
13        }
14
15        System.out.println("Contraseña correcta, acceso permitido.");
16    }
17 }
18 }
```

En la línea 8 mientras clave sea distinto a pass el bucle que llega hasta la línea 13 se irá repitiendo.

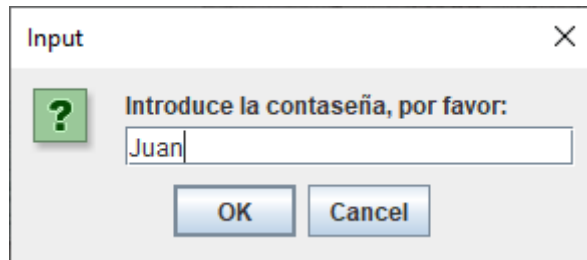
En la línea 9 hacemos que nos pregunte con una contraseña en una ventana emergente, si introducimos una contraseña distinta a Juan en la línea 11 con una condición nos dirá que la contraseña es incorrecta y no saldremos del bucle, cuando introduzcamos la contraseña correcta imprimirá "Contraseña correcta, acceso permitido" y saldremos del bucle.

Vamos a ejecutar el programa:



Pulsamos OK.

Nos dirá Clave incorrecta.



Pulsamos OK.

Nos dirá Contraseña correcta, acceso permitido.

Y terminará el programa.



## Bucles II. (Vídeo 19)

En este ejemplo vamos elaborar una especie de juego en que el programa generará un número aleatorio entre 0 y 100 y en la que tendremos que adivinar el número, mientras no lo acertemos nos lo irá preguntando.

Vamos a crear una nueva clase llamada Adivina\_numero.

En este programa vamos a introducir la consola tanto para introducir los números como para ver el resultado.

```
1 import java.util.*;
2 public class Adivina_numero {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         int aleatorio = (int)(Math.random()*100);
7         Scanner entrada=new Scanner(System.in);
8         int numero=0;
9         int intentos=0;
10        while(numero!=aleatorio) {
11            System.out.println("Introduce un número, por favor:");
12            intentos++;
13            numero=entrada.nextInt();
14            if(aleatorio<numero) {
15                System.out.println("Más bajo");
16            }
17            else if (aleatorio>numero) {
18                System.out.println("Más alto");
19            }
20        }
21        System.out.println("Correcto has acertado al intento " + intentos + ".");
22    }
23
24 }
```

En la línea 6 en la variable aleatorio almacenamos un valor entero aleatorio entre 0 y 100.

En la línea 7 creamos un objeto de tipo Scanner para decirle que los datos los introduciremos por la consola.

Declaramos las variables numero e intentos de tipo int y las inicializamos a 0.

Línea 10 Mientras las variables numero y aleatorio sean distintas ¡=

Línea 11 Pregunta por un número.

Línea 12 intentos++ es un contador que incrementará en 1 cada vez que introduzcamos un número.

Línea 13 el valor introducido por teclado se almacena en la variable número.

Línea 14 comparamos si el valor que hemos introducido es mayor al valor de la variable aleatorio, si es así dirá "Más bajo".

Línea 17 comparamos si el valor que hemos introducido es menor al valor de la variable aleatorio, si es así dirá "Más alto".

Este bucle se repetirá hasta que las variables numero y aleatorio sean iguales, como la condición mientras deja de cumplirse vamos directamente a la línea 21 donde nos dice "Correcto has

acertado al intento x, donde x es el valor de intentos variable que hemos utilizado como contador.



## Bucles III. (VÍdeo 20)

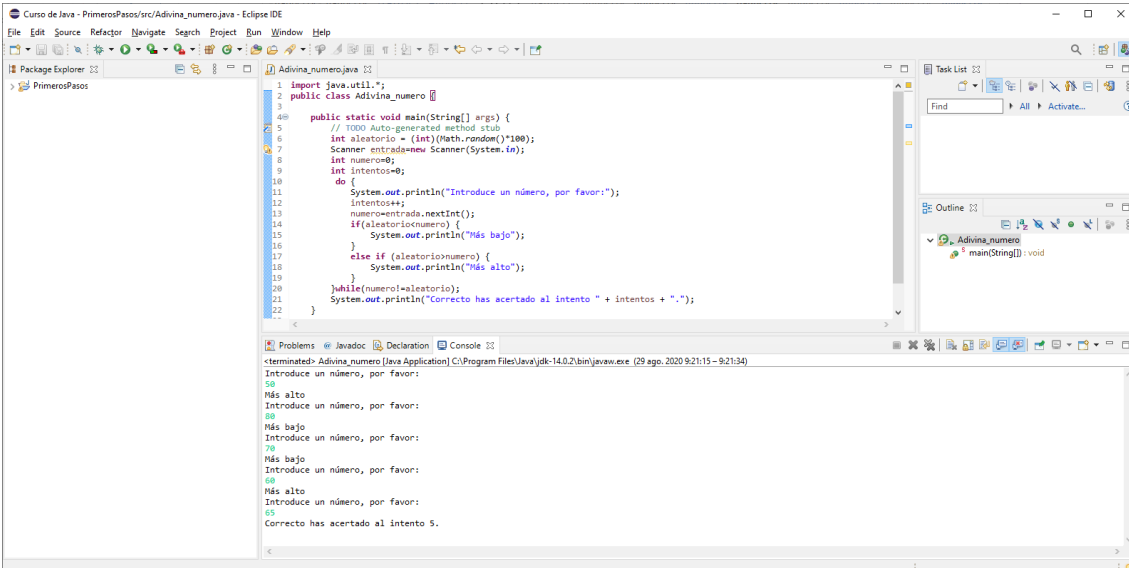
### Bucle Do – while

- Sintaxis:
  - Do {
    - Línea1
    - Línea2
    - Línea3
    - Línea4
  - }while(condición);

El bucle do-while se asegura que aun siendo la condición falsa, por lo menos se ejecuta una vez.

```
1 import java.util.*;
2 public class Adivina_numero {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         int aleatorio = (int)(Math.random()*100);
7         Scanner entrada=new Scanner(System.in);
8         int numero=0;
9         int intentos=0;
10        do {
11            System.out.println("Introduce un número, por favor:");
12            intentos++;
13            numero=entrada.nextInt();
14            if(aleatorio<numero) {
15                System.out.println("Más bajo");
16            }
17            else if (aleatorio>numero) {
18                System.out.println("Más alto");
19            }
20        }while(numero!=aleatorio);
21        System.out.println("Correcto has acertado al intento " + intentos + ".");
22    }
23
24 }
```

En este ejemplo garantizamos que al menos una vez entre en el bucle do-while ya que la comparación la realiza al final. Si lo ejecutamos este será el resultado:



```
Curso de Java - PrimerosPasos/src/Adivina_numero.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
> PrimerosPasos
Adivina_numero.java
1 import java.util.*;
2 public class Adivina_numero {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         int aleatorio = (int)(Math.random()*100);
7         Scanner entrada=new Scanner(System.in);
8         int numero=0;
9         int intentos=0;
10        do {
11            System.out.println("Introduce un número, por favor:");
12            intentos++;
13            numero=entrada.nextInt();
14            if(aleatorio<numero) {
15                System.out.println("Más bajo");
16            }
17            else if (aleatorio>numero) {
18                System.out.println("Más alto");
19            }
20        }while(numero!=aleatorio);
21        System.out.println("Correcto has acertado al intento " + intentos + ".");
22    }
23
24 }
Task List
Find
Outline
Adivina_numero
main(String[]): void
Problems
Declaration
Console
<terminated> Adivina_numero [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (29 ago. 2020 9:21:15 - 9:21:34)
Introduce un número, por favor:
50
Más alto
Introduce un número, por favor:
80
Más bajo
Introduce un número, por favor:
70
Más bajo
Introduce un número, por favor:
60
Más alto
Introduce un número, por favor:
65
Correcto has acertado al intento 5.
```

```

Introduce un número, por favor:
50
Más alto
Introduce un número, por favor:
80
Más bajo
Introduce un número, por favor:
70
Más bajo
Introduce un número, por favor:
60
Más alto
Introduce un número, por favor:
65
Correcto has acertado al intento 5.

```

Vamos a realizar un programa para que calcule el peso ideal de un hombre y de una mujer, para el peso del hombre se mide la altura en cm y se resta 110 y para una mujer se mide la altura en centímetros y se resta 120.

Vamos a crear una clase nueva con el nombre peso\_ideal.

```

1 import javax.swing.*;
2 public class peso_ideal {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         String genero="";
7         do {
8             genero=JOptionPane.showInputDialog("Introduce tu género (H/M):");
9         }while(genero.equalsIgnoreCase("H")==false && genero.equalsIgnoreCase("M")==false);
10
11         int altura=Integer.parseInt(JOptionPane.showInputDialog("Introduce altura en cm. "));
12
13         int pesoideal=0;
14
15         if(genero.equalsIgnoreCase("H")){
16             pesoideal=altura-110;
17         }
18         else if(genero.equalsIgnoreCase("M")){
19             pesoideal=altura-120;
20         }
21
22         System.out.println("Te peso ideal es " + pesoideal);
23     }
24 }

```

Línea 8 a la variable genero le asignamos un valor por teclado.

Entre las 7 y 9 controlamos que solo puedes contestar H o M indistinto mayúsculas o minúsculas, mientras no se así no saldremos de bucle.

La línea 11 asignamos a la variable altura de tipo int un valor por teclado.

La línea 13 a la variables pesoideal la definimos como int y un valor inicial de 0.

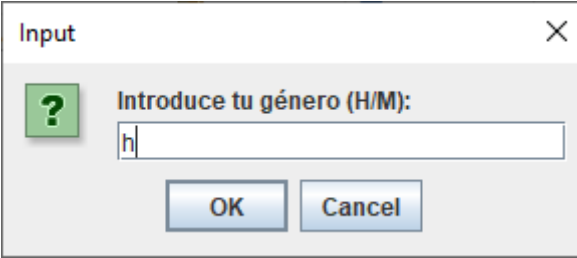
En la línea 15 comparamos que el valor de genero sea H o h para pasar el valor pesoideal=altura-110.

En la línea 18 comparamos que el valor de genero sea M o m para pasar el valor pesoideal=altura-120.

En la línea 22 te dice cual es tu peso ideal.



Vamos a ejecutarlo.



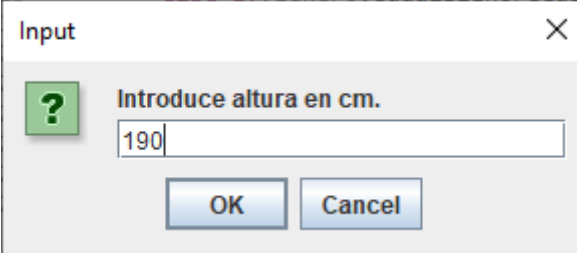
Input

Introduce tu género (H/M):

h

OK Cancel

Pulsamos Ok.



Input

Introduce altura en cm.

190

OK Cancel

Pulsamos Ok.

El resultado será:

Te peso ideal es 80



## Bucles IV (VÍdeo 21)

### El bucle For

- Sintaxis
  - For(inicio bucle, condición, contador bucle){
    - Línea1
    - Línea2
    - Línea3
    - Línea4
  - }

Vamos a ejecutar Eclipse.

Vamos a crear una clase nueva llamada Uso\_Bucle\_For.

```
1
2 public class Uso_Bucle_For {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         for(int i=0;i<10;i++) {
8             System.out.println("Juan");
9         }
10
11     }
12
13 }
```

En el bucle for declaramos una variable entera iniciando su valor a 0; la condición que ha de cumplir que *i* sea menor de 10; y el incremento sea de 1. *i++*

Vamos a ejecutar.

```
Juan
Juan
Juan
Juan
Juan
Juan
Juan
Juan
Juan
Juan
Juan
```

Para ver los valores que adquiere en cada vuelta vamos a realizar lo siguiente.

```

1
2 public class Uso_Bucle_For {
3
4 public static void main(String[] args) {
5     // TODO Auto-generated method stub
6
7     for(int i=0;i<10;i++) {
8         System.out.println("Juan " + i); ←
9     }
10
11 }
12
13 }

```

Cuando ejecutemos este será el resultado:

```

Juan 0
Juan 1
Juan 2
Juan 3
Juan 4
Juan 5
Juan 6
Juan 7
Juan 8
Juan 9

```

```

for(int i=10;i>0;i--) {
    System.out.println("Juan " + i);
}

```

Ahora cambiamos el valor inicial i igual 10 condición i mayor a 0 y como incremento i -- decrecerá en -1, este será el resultado:

```

Juan 10
Juan 9
Juan 8
Juan 7
Juan 6
Juan 5
Juan 4
Juan 3
Juan 2
Juan 1

```

```

for(int i=0;i<20;i+=2) {
    System.out.println("Juan " + i);
}

```

Inicializamos la variable i a 0, como condición i menor a veinte y como incremento le suma cada vez 2, este será el resultado:

Juan 0  
Juan 2  
Juan 4  
Juan 6  
Juan 8  
Juan 10  
Juan 12  
Juan 14  
Juan 16  
Juan 18

Vamos a realizar una nueva clase llamada `comprueba_mail` que evalúe si una dirección de correo electrónica es correcta, para ello solo vamos a controlar si lleva una "@" arroba.

```
1 import javax.swing.JOptionPane;
2
3 public class comprueba_mail {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         boolean arroba=false;
8         String mail=JOptionPane.showInputDialog("Introduce mail: ");
9         for(int i=0; i<mail.length();i++) {
10            if(mail.charAt(i)=='@') {
11                arroba=true;
12            }
13        }
14        if(arroba==true) {
15            System.out.println("El correo intrducido es correcto.");
16        }
17        else{
18            System.out.println("El correo introducido es incorrecto.");
19        }
20    }
21
22 }
```

Va con comillas simples

En la línea 7 definimos una variable de tipo boolean su valor puede ser true o false.

En la línea 8 asignamos el valor de la variable mail desde una ventana de diálogo.

En la línea 9 hacemos un ciclo for donde i empieza por cero has que sea menor a la longitud de la palabra que hemos introducido para ello es `mail.length()`, con un incremento de 1. De este modo en el interior de bucle comprueba carácter por carácter si encuentra la @, en caso afirmativo la variable arroba pasará a valer true, si en todo el bucle no ha encontrado la arroba la variable arroba seguirá valiendo false.

En la línea 14 cuando ya hemos salido del bucle comparamos si vale true para decirnos que "El correo introducido es correcto", de lo contrario nos dirá "El correo introducido es incorrecto".





## Bucles V (Vídeo 22)

Siguiendo con el ejemplo anterior queremos controlar que al introducir la dirección de correo electrónico este solo contenga una @ es decir que si tiene 0 o más de 1 que nos diga que el correo electrónico no es correcto.

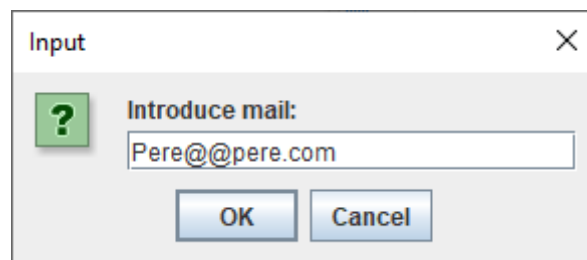
```
1 import javax.swing.JOptionPane;
2
3 public class comprueba_mail {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int arroba=0;
8         String mail=JOptionPane.showInputDialog("Introduce mail: ");
9         for(int i=0; i<mail.length();i++) {
10             if(mail.charAt(i)=='@') {
11                 arroba++;
12             }
13         }
14         if(arroba==1) {
15             System.out.println("El correo intrducido es correcto.");
16         }
17         else{
18             System.out.println("El correo introducido es incorrecto.");
19         }
20     }
21 }
22 }
```

En la línea 7 hemos cambiado el tipo de variable a int con un valor inicial de 0.

En la línea 11 hemos puesto un contador para saber cuántas @ tiene el correo electrónico que hemos introducido.

En la línea 14 controlamos que si solo ha encontrado 1 @ de nuestro correo electrónico como correcto de lo contrario en la línea 17 nos dirá que lo contrario nos diga que el correo electrónico no es correcto.

Ejecuta el proyecto poniendo más de una arroba.



Este será el resultado:

```
El correo introducido es incorrecto.
```

Ahora queremos controlar que nuestra dirección de correo por lo menos tenga un punto.

```

1 import javax.swing.JOptionPane;
2
3 public class comprueba_mail {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int arropa=0;
8         boolean punto = false;
9         String mail=JOptionPane.showInputDialog("Introduce mail: ");
10        for(int i=0; i<mail.length();i++) {
11            if(mail.charAt(i)=='@') {
12                arropa++;
13            }
14            if(mail.charAt(i)=='.') {
15                punto=true;
16            }
17        }
18        if(arropa==1 && punto==true) {
19            System.out.println("El correo intrducido es correcto.");
20        }
21        else{
22            System.out.println("El correo introducido es incorrecto.");
23        }
24    }

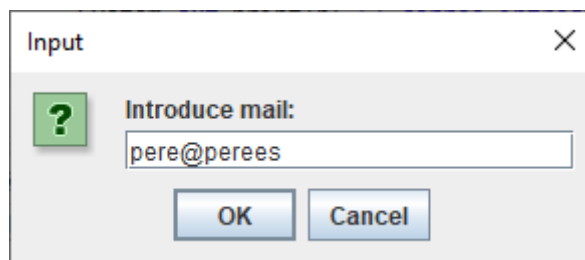
```

En la línea 8 hemos creado una variable llamada punto de tipo boolean con un valor inicial de false.

En la línea 14 controlamos que si encontramos un carácter que sea "." La variable punto pase a valer true.

En la línea 18 con un condicionante con el operador Y && queremos comprobar y ha encontrado una @ y por lo menos un . ya que una dirección de correo electrónico puede tener más de un punto.

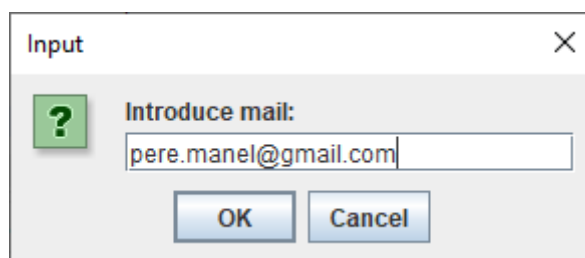
Prueba una dirección de correo electrónica que no tenga punto.



Este será el resultado:

El correo introducido es incorrecto.

Vamos a probar con una correo que tenga mas de un punto.



El resultado será:

El correo intrducido es correcto.

Vamos a realizar un nuevo proyecto para calcular el factorial de un número este se representa de la siguiente forma  $6!$  Y esto es igual a  $6 * 5 * 4 * 3 * 2 * 1$  es 720.

Vamos a crear una nueva clase llamado Factorial.

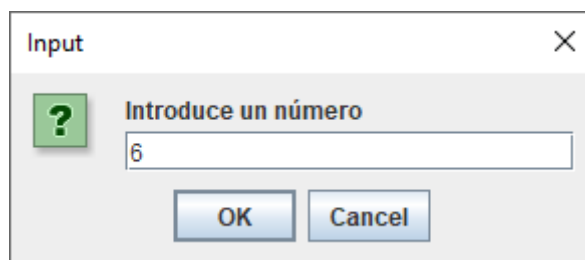
```
1 import javax.swing.*;
2 public class Factorial {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         int resultado=1;
8         int numero=Integer.parseInt(JOptionPane.showInputDialog("Introduce un número"));
9         for (int i=numero; i>0;i--) {
10            resultado=resultado*i;
11        }
12
13        System.out.println("El factorial de "+ numero + " es " + resultado);
14    }
15
16 }
```

En la línea 7 definimos una variable de tipo int llamada resultado con el valor inicial de 1, esto se hace porque en la línea 10 acumularemos el producto de resultado por el valor de i que tenemos en el bucle for, si fuera 0 multiplicaría por 0 siempre su resultado sería 0.

En la línea 10 resultado es multiplicado por el valor de i, luego por i-1 y así hasta llegar al final del bucle.

En la línea 13 fuera del ciclo for imprimimos el valor de la variable resultado que tiene el valor del factorial del número introducido desde la ventana de diálogo.

Vamos a ejecutarlo:



Este será el resultado:

El factorial de 6 es 720

Si al introducir valores estos se desbordan es porque la variable no puede almacenar dicho valor es decir en lugar de declararla int la podemos declarar long.

Long resultado=1L;

Recuerda poner la letra L después del valor de la variable.



The image shows a course cover with a dark orange to red gradient background. At the top left is a small Java logo icon. The main title 'CURSO JAVA' is in large white letters. To the right, the number '22' is in a yellow box. Below the title, the text 'FLUJO DE CONTROL. CONDICIONALES Y BUCLES V' is written in white. A world map is faintly visible in the background. At the bottom left is the Eclipse logo, and at the bottom right is the Java logo with its signature blue waves.

**CURSO JAVA** 22

**FLUJO DE CONTROL.  
CONDICIONALES Y BUCLES V**

eclipse Java

## Arrays I (VÍdeo 23)

### Matrices (Arrays, Arreglos)

- ¿Qué es? Estructura de datos que contiene una colección de valores del mismo tipo.
- ¿Para qué? Para almacenar valores que normalmente tienen alguna relación entre sí.
- Sintaxis:
  - Declaración: `int[] mi_matriz=new int[10];`

**Hipotética representación gráfica**

Variable de tipo entero

Matriz de tipo entero

PLDORASINFORMÁTICAS

**Declaración y tipo**

Matriz de tipo entero

Declaración: `int [] mi_matriz=new int[5];`

Tipo de la matriz      Nombre de la matriz      Nº de valores que almacena la matriz

PLDORASINFORMÁTICAS



# Dar valores iniciales



Pos 0    Pos 1    Pos 2    Pos 3    Pos 4



```
mi_matriz [0] = 15;
mi_matriz [1] = 25;
mi_matriz [2] = 8;
mi_matriz [3] = -7;
mi_matriz [4] = 92;
```

```
Int [] mi_matriz= new int [5];
```

FILEDRABINFORMÁTICAS

Para declarar la matriz e inicializarla en la misma línea:

```
Int [] mi_matriz = { 15, 25, 8, -7, 92 };
```

Vamos a crear una nueva clase llamada Uso\_Arrays.

```

1
2 public class Uso_Arrays {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         int [] mi_matriz=new int[5];
7         //También es correcto (int mi:matriz[]=new int[5]);
8         mi_matriz[0]=5;
9         mi_matriz[1]=38;
10        mi_matriz[2]=-15;
11        mi_matriz[3]=92;
12        mi_matriz[4]=71;
13
14        System.out.println(mi_matriz[3]);
15
16    }
17
18 }
19

```

En la línea 6 declaramos la matriz para almacenar 5 valores.

De la línea 8 hasta 12 vamos asignando los valores de la matriz.

En la línea 14 imprimimos por consola el valor de la matriz almacenado en la posición 4 que es el tres, ya que las matrices empiezan por 0.

Ejecutamos y observamos el resultado:

92

```

1
2 public class Uso_Arrays {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         int [] mi_matriz=new int[5];
7         //También es correcto (int mi:matriz[]=new int[5]);
8         mi_matriz[0]=5;
9         mi_matriz[1]=38;
10        mi_matriz[2]=-15;
11        mi_matriz[3]=92;
12        mi_matriz[4]=71;
13
14        System.out.println(mi_matriz[0]);
15        System.out.println(mi_matriz[1]);
16        System.out.println(mi_matriz[2]);
17        System.out.println(mi_matriz[3]);
18        System.out.println(mi_matriz[4]);
19    }
20
21 }

```

Si queremos ver todos los valores que tiene almacenada la matriz, este será el resultado:

```

5
38
-15
92
71

```

Para imprimir todos los valores de una matriz lo mejor es un bucle for.

```

1
2 public class Uso_Arrays {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         int [] mi_matriz=new int[5];
7         //También es correcto (int mi:matriz[]=new int[5]);
8         mi_matriz[0]=5;
9         mi_matriz[1]=38;
10        mi_matriz[2]=-15;
11        mi_matriz[3]=92;
12        mi_matriz[4]=71;
13
14        for(int i=0; i<5;i++) {
15            System.out.println("Valor del índice " + i + " = " + mi_matriz[i]);
16        }
17    }
18
19 }

```

La variable i dentro del bucle for asumirá los valores de 0 hasta 4, de este modo podemos consultar toda la matriz, este será el resultado:

```

Valor del índice 0 = 5
Valor del índice 1 = 38
Valor del índice 2 = -15
Valor del índice 3 = 92
Valor del índice 4 = 71

```

Ahora vamos a ver otra forma de declarar una matriz.

```

1
2 public class Uso_Arrays {
3
4 public static void main(String[] args) {
5     // TODO Auto-generated method stub
6     int[] mi_matriz= {5, 38, -15, 92, 71};
7
8     for(int i=0; i<5;i++) {
9         System.out.println("Valor del índice " + i + " = " + mi_matriz[i]);
10    }
11 }
12
13 }

```

En la línea 6 declaramos una matriz y además le asignamos los valores, esto se llama declaración simplificada o implícita.

Si ejecutamos , este será el resultado:

```

Valor del índice 0 = 5
Valor del índice 1 = 38
Valor del índice 2 = -15
Valor del índice 3 = 92
Valor del índice 4 = 71

```

```

1
2 public class Uso_Arrays {
3
4 public static void main(String[] args) {
5     // TODO Auto-generated method stub
6     int[] mi_matriz= {5, 38, -15, 92, 71, 95, 85, 65, 25, 14, 78};
7
8     for(int i=0; i<mi_matriz.length;i++) {
9         System.out.println("Valor del índice " + i + " = " + mi_matriz[i]);
10    }
11 }
12
13 }

```

Si queremos saber la longitud de una matriz, es decir cuantos valores almacena podemos utilizar la función length, tal como se muestra en el código, este será el resultado:

```

Valor del índice 0 = 5
Valor del índice 1 = 38
Valor del índice 2 = -15
Valor del índice 3 = 92
Valor del índice 4 = 71
Valor del índice 5 = 95
Valor del índice 6 = 85
Valor del índice 7 = 65
Valor del índice 8 = 25
Valor del índice 9 = 14
Valor del índice 10 = 78

```



## Arrays II (VÍdeo 24)

Bucle for each (por cada).

Vamos a crear una clase nueva llamada Uso\_Arrays\_II.

```
1
2 public class Uso_Arrays_II {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         String []países=new String[8];
7         países[0]="España";
8         países[1]="Méjico";
9         países[2]="Colombia";
10        países[3]="Perú";
11        países[4]="Chile";
12        países[5]="Argentina";
13        países[6]="Ecuador";
14        países[7]="Venezuela";
15
16        for(int i=0;i<8;i++) {
17            System.out.println("País "+ (i+1) + " " +países[i]);
18        }
19    }
20
21 }
```

Este será el resultado:

```
Pais 1 España
Pais 2 Mejico
Pais 3 Colombia
Pais 4 Perú
Pais 5 Chile
Pais 6 Argentina
Pais 7 Ecuador
Pais 8 Venezuela
```

Ahora vamos a modificar el bucle for con each.

```
1
2 public class Uso_Arrays_II {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         String []países=new String[8];
7         países[0]="España";
8         países[1]="Méjico";
9         países[2]="Colombia";
10        países[3]="Perú";
11        países[4]="Chile";
12        países[5]="Argentina";
13        países[6]="Ecuador";
14        países[7]="Venezuela";
15
16        for(String elemento:países) {
17            System.out.println(elemento);
18        }
19    }
20 }
```

El nuevo for sabe el número de elementos que tiene la matriz.

Para inicializar la matriz y declarar los valores en una sola línea sería de la siguiente forma:  
String [] países={"España", "Méjico", "Colombia", "Perú", "Chile", "Argentina", "Ecuador", "Venezuela"};

Ahora vamos a rellenar una matriz por mediación de una ventana de diálogo:

```

1 import javax.swing.*;
2 public class Uso_Arrays_II {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         String [] países=new String[8];
7         for (int i=0; i<8; i++){
8             países[i]=JOptionPane.showInputDialog("Introduce país (" + (i+1)+") :");
9         }
10
11         for(String elemento:países) {
12             System.out.println(elemento);
13         }
14     }

```

Ejecuta el programa y una vez introducidos los 8 países estos se verán por la consola.

Vamos a crear una matriz que se rellenará con valores aleatorios.

```

1
2 public class Uso_Arrays_II {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         int contador=0;
7         int[] matriz_aleatorios=new int[150];
8         for(int i=0; i<matriz_aleatorios.length;i++) {
9             matriz_aleatorios[i]=(int)Math.round(Math.random()*100);
10        }
11        for(int numeros:matriz_aleatorios){
12            System.out.print(numeros+" ");
13            contador++;
14            if(contador==15) {
15                System.out.println("");
16                System.out.println("-----");
17                contador=0;
18            }
19        }
20    }
21 }

```

Este será el resultado:

```

61 8 88 72 58 95 73 73 94 15 9 34 97 60 53
-----
97 4 19 5 2 16 49 100 48 50 12 54 90 97 51
-----
7 51 43 75 51 29 36 2 61 9 86 91 64 46 71
-----
3 33 27 63 4 93 11 55 40 31 65 52 0 46 17
-----
24 100 47 98 37 11 97 4 84 94 50 16 33 69 56
-----
62 10 27 0 96 59 52 75 69 28 74 86 21 19 13
-----
87 44 38 51 29 42 69 76 23 18 89 97 82 99 92
-----
35 73 61 3 99 28 41 2 0 92 25 30 34 62 27
-----
54 73 45 66 77 99 97 87 23 44 56 78 21 60 40
-----
75 80 43 12 49 28 39 20 44 91 93 12 4 27 25
-----

```





The image shows a course cover with a dark orange to red gradient background. At the top left is a small Java logo icon. The main title 'CURSO JAVA' is in large white serif font. To its right, the number '24' is in a yellow box. Below the title, the text 'MATRICES II. BUCLE FOR EACH' is centered in white. A world map is faintly visible in the background. At the bottom left is the Eclipse logo and text, and at the bottom right is the Java logo.

## Arrays III. Arrays bidimensionales I. (Vídeo 25)

### Matrices bidimensionales



0,0	1,0	2,0	3,0
0,1	1,1	2,1	3,1
0,2	1,2	2,2	3,2
0,3	1,3	2,3	3,3
0,4	1,4	2,4	3,4

Vamos a crear una clase nueva llamada Arrays\_bidimensionales.

```
1 public class Arrays_bidimensionales {  
2  
3     public static void main(String[] args) {  
4         // TODO Auto-generated method stub  
5         int[][] matrix=new int[4][5];  
6  
7         matrix[0][0]=15;  
8         matrix[0][1]=21;  
9         matrix[0][2]=18;  
10        matrix[0][3]=9;  
11        matrix[0][4]=15;  
12  
13        matrix[1][0]=10;  
14        matrix[1][1]=52;  
15        matrix[1][2]=17;  
16        matrix[1][3]=19;  
17        matrix[1][4]=7;  
18  
19        matrix[2][0]=19;  
20        matrix[2][1]=2;  
21        matrix[2][2]=19;  
22        matrix[2][3]=17;  
23        matrix[2][4]=7;  
24  
25        matrix[3][0]=92;  
26        matrix[3][1]=13;  
27        matrix[3][2]=13;  
28        matrix[3][3]=32;  
29        matrix[3][4]=41;  
30  
31        System.out.println(matrix[2][3]);  
32    }  
33 }
```

En la línea 5 definimos la matriz bidimensional y desde la línea 7 hasta la línea 29 le asignamos los valores.

En la línea 31 queremos ver en consola el valor que se guarda en la matriz en la posición 2-3 matrix[2][3].

---

```
public class Arrays_bidimensionales {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        int[][] matrix=new int[4][5];  
  
        matrix[0][0]=15;  
        matrix[0][1]=21;  
        matrix[0][2]=18;  
        matrix[0][3]=9;  
        matrix[0][4]=15;  
  
        matrix[1][0]=10;  
        matrix[1][1]=52;  
        matrix[1][2]=17;  
        matrix[1][3]=19;  
        matrix[1][4]=7;  
  
        matrix[2][0]=19;  
        matrix[2][1]=2;  
        matrix[2][2]=19;  
        matrix[2][3]=17;  
        matrix[2][4]=7;  
  
        matrix[3][0]=92;  
        matrix[3][1]=13;  
        matrix[3][2]=13;  
        matrix[3][3]=32;  
        matrix[3][4]=41;  
  
        for(int i=0;i<4;i++) {  
            for(int j=0;j<5;j++) {  
                System.out.print(matrix[i][j]+ " ");  
            }  
            System.out.println();  
        }  
    }  
}
```

---

Este sería el resultado:

```
15 21 18 9 15  
10 52 17 19 7  
19 2 19 17 7  
92 13 13 32 41
```



The image shows a course cover with a dark orange background and a world map. At the top left is a small logo with the text '12014'. The main title 'CURSO JAVA' is in large white letters. To the right, the number '25' is in a yellow box. Below the title, the text 'MATICES III. MATRICES BIDIMENSIONALES' is written in white. At the bottom left is the 'eclipse' logo, and at the bottom right is the 'Java' logo with its signature blue waves.

## Arrays IV. Arrays bidimensionales II. (Vídeo 26)

Otra forma de declarar una matriz bidimensional.

```
1 public class Arrays_bidimensionales {
2
3     public static void main(String[] args) {
4         // TODO Auto-generated method stub
5         int[][] matrix= {
6             {10,15,18,19,21},
7             {5,25,37,41,15},
8             {7,19,32,14,90},
9             {85,2,7,40,27}
10        };
11
12
13
14        for(int i=0;i<4;i++) {
15            for(int j=0;j<5;j++) {
16                System.out.print(matrix[i][j]+ " ");
17            }
18            System.out.println();
19        }
20    }
21 }
```

Si ejecutamos este será el resultado:

```
10 15 18 19 21
5 25 37 41 15
7 19 32 14 90
85 2 7 40 27
```

Si nosotros a la hora de leer una matriz por error intentamos leer valores que no tiene.

```
for(int i=0;i<4;i++) {
    for(int j=0;j<6;j++) {
        System.out.print(matrix[i][j]+ " ");
    }
    System.out.println();
}
```

Observaremos el siguiente error.

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5
out of bounds for length 5
    at Arrays_bidimensionales.main(Arrays_bidimensionales.java:16)
```

Vamos a leer la matriz bidimensional con el método for con each.

```

1 public class Arrays_bidimensionales {
2
3 public static void main(String[] args) {
4     // TODO Auto-generated method stub
5     int[][] matrix= {
6         {10,15,18,19,21},
7         {5,25,37,41,15},
8         {7,19,32,14,90},
9         {85,2,7,40,27}
10    };
11
12    for(int[]fila:matrix) {
13        System.out.println("");
14        for(int z: fila) {
15            System.out.print(z + " ");
16        }
17    }
18
19 }
20 }

```

Si ejecutamos este será el resultado:

```

10 15 18 19 21
5 25 37 41 15
7 19 32 14 90
85 2 7 40 27

```

10%	11%	12%	13%	14%	15%
10.000,00 €	10.000,00 €	10.000,00 €	10.000,00 €	10.000,00 €	10.000,00 €
11.000,00 €	11.100,00 €	11.200,00 €	11.300,00 €	11.400,00 €	11.500,00 €
12.100,00 €	12.321,00 €	12.544,00 €	12.769,00 €	12.996,00 €	13.225,00 €
13.310,00 €	13.676,31 €	14.049,28 €	14.428,97 €	14.815,44 €	15.208,75 €
14.641,00 €	15.180,70 €	15.735,19 €	16.304,74 €	16.889,60 €	17.490,06 €

Vamos a realizar un ejercicio para poder almacenar esta array en Java.

Vamos a crear una nueva case llamada Ejemplo\_Array\_2D.

```

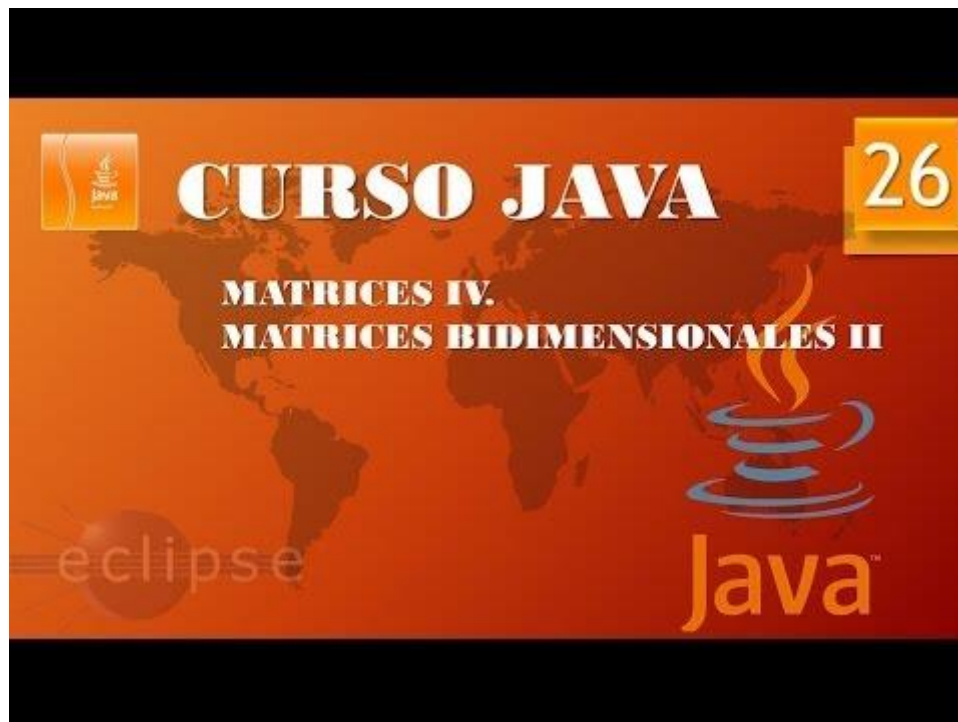
1
2 public class Ejemplo_Array_2D {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         double acumulado;
7         double interes=0.10;
8
9         double[][] saldo=new double[6][5];
10        for(int i=0; i<6;i++) {
11            saldo[i][0]=10000;
12            acumulado=10000;
13            for(int j=1;j<5;j++) {
14                acumulado=acumulado+(acumulado*interes);
15                saldo[i][j]=acumulado;
16            }
17
18            interes=interes+0.01;
19        }
20        for(int i=0;i<6;i++) {
21            System.out.println();
22            for(int j=0;j<5;j++) {
23                System.out.printf("%.2f",saldo[i][j]);
24                System.out.print(" €   ");
25            }
26        }
27    }
28
29 }

```

Este será el resultado:

10000,00 €	11000,00 €	12100,00 €	13310,00 €	14641,00 €
10000,00 €	11100,00 €	12321,00 €	13676,31 €	15180,70 €
10000,00 €	11200,00 €	12544,00 €	14049,28 €	15735,19 €
10000,00 €	11300,00 €	12769,00 €	14428,97 €	16304,74 €
10000,00 €	11400,00 €	12996,00 €	14815,44 €	16889,60 €
10000,00 €	11500,00 €	13225,00 €	15208,75 €	17490,06 €







## Contenido

Presentación (Vídeo 1) .....	1
Instalación JRE y Eclipse (Vídeo 2).....	3
Introducción (Vídeo 3).....	13
Estructuras principales I (Vídeo 4) .....	15
Estructuras principales II (Vídeo 5) .....	23
Estructuras principales III. Declaración variables Eclipse (Vídeo 6) .....	25
Estructuras principales IV. Constantes y Operadores (Vídeo 7) .....	30
Estructuras principales V. Constantes y Operadores II (Vídeo 8).....	35
Estructuras principales IV Clase Math (Vídeo 9) .....	38
Estructuras principales VII. Clase Math II. (Vídeo 10) .....	42
Manipulación de cadenas. Clase String I (Vídeo 11) .....	45
Manipulación de cadenas. Clase String II (Vídeo 12) .....	49
Acercamiento a la API Paquetes (Vídeo 13).....	52
Entrada Salida datos I (Vídeo 14) .....	58
Entrada Salida datos II. (Vídeo 15) .....	62
Condicionales I. (Vídeo 16).....	66
Condicionales II (Vídeo 17).....	69
Bucles I (Vídeo 18).....	71
Bucles II. (Vídeo 19).....	73
Bucles III. (Vídeo 20).....	75
Bucles IV (Vídeo 21).....	78
Bucles V (Vídeo 22).....	82
Arrays I (Vídeo 23).....	86
Arrays II (Vídeo 24).....	91
Arrays III. Arrays bidimensionales I. (Vídeo 25) .....	94
Arrays IV. Arrays bidimensionales II. (Vídeo 26) .....	97