



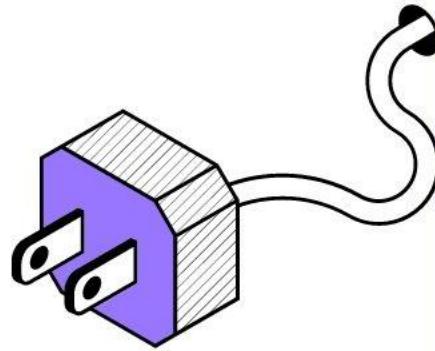
# Java 7 SE Fundamentals





# ¿Qué es Java?

Java es un lenguaje de programación poderoso y versátil para desarrollar software que se ejecuta en dispositivos móviles, computadoras de escritorio y servidores.





ORACLE



Java

### ¿Por quien fue desarrollado?

Por un equipo dirigido por James Gosling en Sun MicroSystems quien después fue adquirida por Oracle en 2010.

### ¿Cuando?

Diseñado en 1991, y llamado originalmente **Oak**, fue pensado para su uso en chips integrados en aparatos electrodomésticos.

### ¿Y después?

En 1995 es renombrado como **Java** y rediseñado para el desarrollo de aplicaciones web.



# Conceptos importantes

Relación de Java con otros lenguajes



## Lenguaje de alto nivel

Independientes de la plataforma, significa poder escribir un programa y ejecutarlo en diferentes tipos de máquinas.



## Interprete

Lee una *sentencia* del *código fuente* y la traduce a *código maquina* para después ejecutarla.



## Compilador

Traduce todo el código fuente a un archivo de código maquina y luego este archivo se ejecuta.



# Java y otros lenguajes de alto nivel

C	C++	Python	Java	Visual Basic	C#
1972	1983	1990	1991	1991	1999
Desarrollado en los Laboratorios Bell. C combina el poder de un lenguaje ensamblador con la facilidad de uso y portabilidad de un lenguaje de alto nivel.	C++ es un lenguaje orientado a objetos, basado en C.	Un lenguaje de secuencias de comandos simple de propósito general bueno para escribir programas cortos	Desarrollado por Sun Microsystems, ahora parte de Oracle. Es ampliamente utilizado para desarrollar aplicaciones de Internet independientes de la plataforma.	Visual Basic fue desarrollado por Microsoft y permite a los programadores desarrollar rápidamente interfaces gráficas de usuario.	Pronunciado "C sharp". Es un híbrido de Java y C++ y fue desarrollado por Microsoft.



# Conceptos Java importantes

## Particularidades de java



### ByteCode

Guardas el código fuente en un archivo .java.  
El código fuente es compilado a bytecode y guardado en un archivo .class.



### Java Virtual Machine

El archivo .class (bytecode) puede ser ejecutado en cualquier computadora con la Java Virtual Machine (JVM).



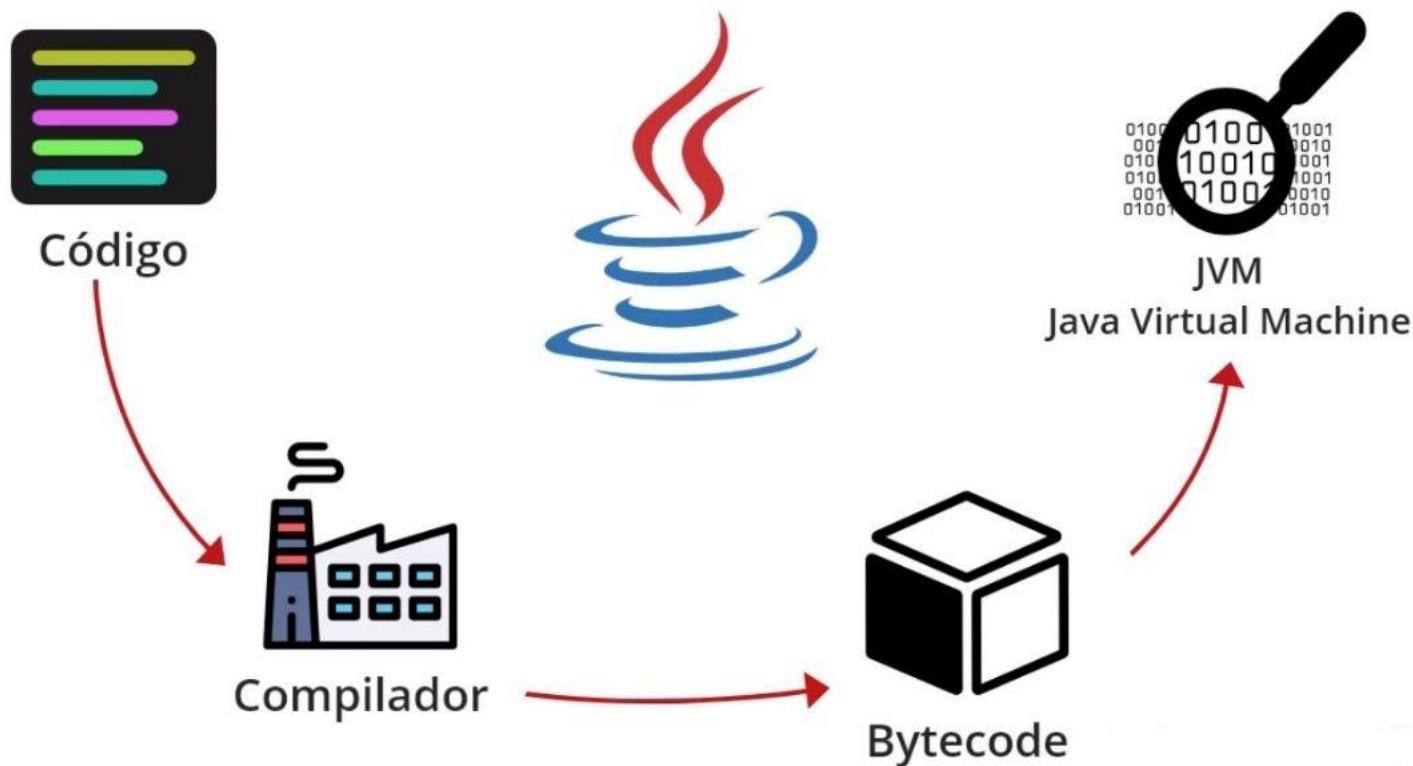
### Java Development Kit

Software para los desarrolladores que incluye herramientas como: compilador, depurador, desensamblador y generador de documentación



# Conceptos Java importantes

Particularidades de java





# Lenguajes JVM

**Scala**

Scala es un lenguaje puramente orientado a objetos y también un lenguaje funcional.

**Groovy**

Groovy es un lenguaje de programación orientado a objetos implementado sobre la plataforma Java.

Tiene características similares a Python, Ruby, Perl y Smalltalk

**Kotlin**

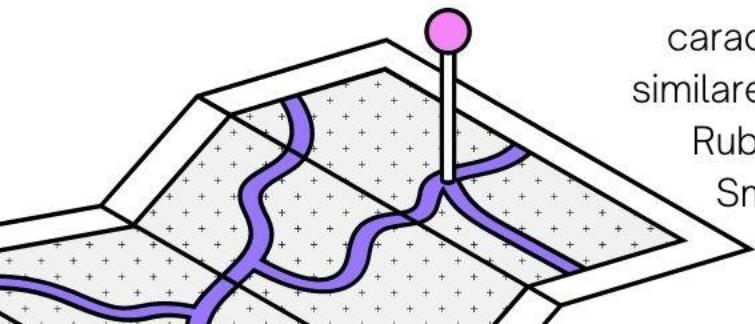
Kotlin es un lenguaje de programación de tipado estático que corre sobre la máquina virtual de Java y que también puede ser compilado a código fuente de JavaScript.

**Python**

Un lenguaje de secuencias de comandos simple de propósito general bueno para escribir programas cortos

**Ruby**

Un lenguaje de programación dinámico y de código abierto enfocado en la simplicidad y productividad. Su elegante sintaxis se siente natural al leerla y fácil al escribirla.





# Tecnologías Java

Existen distintas tecnologías Java según su enfoque



## Java Standard Edition (Java SE)

Desarrollo de aplicaciones de escritorio, sobre plataformas como Windows, Linux o MacOS



## Java Enterprise Edition (Java EE)

Desarrollo pensado para grandes corporaciones y aplicaciones de gran dimensión enfocadas al lado del servidor.

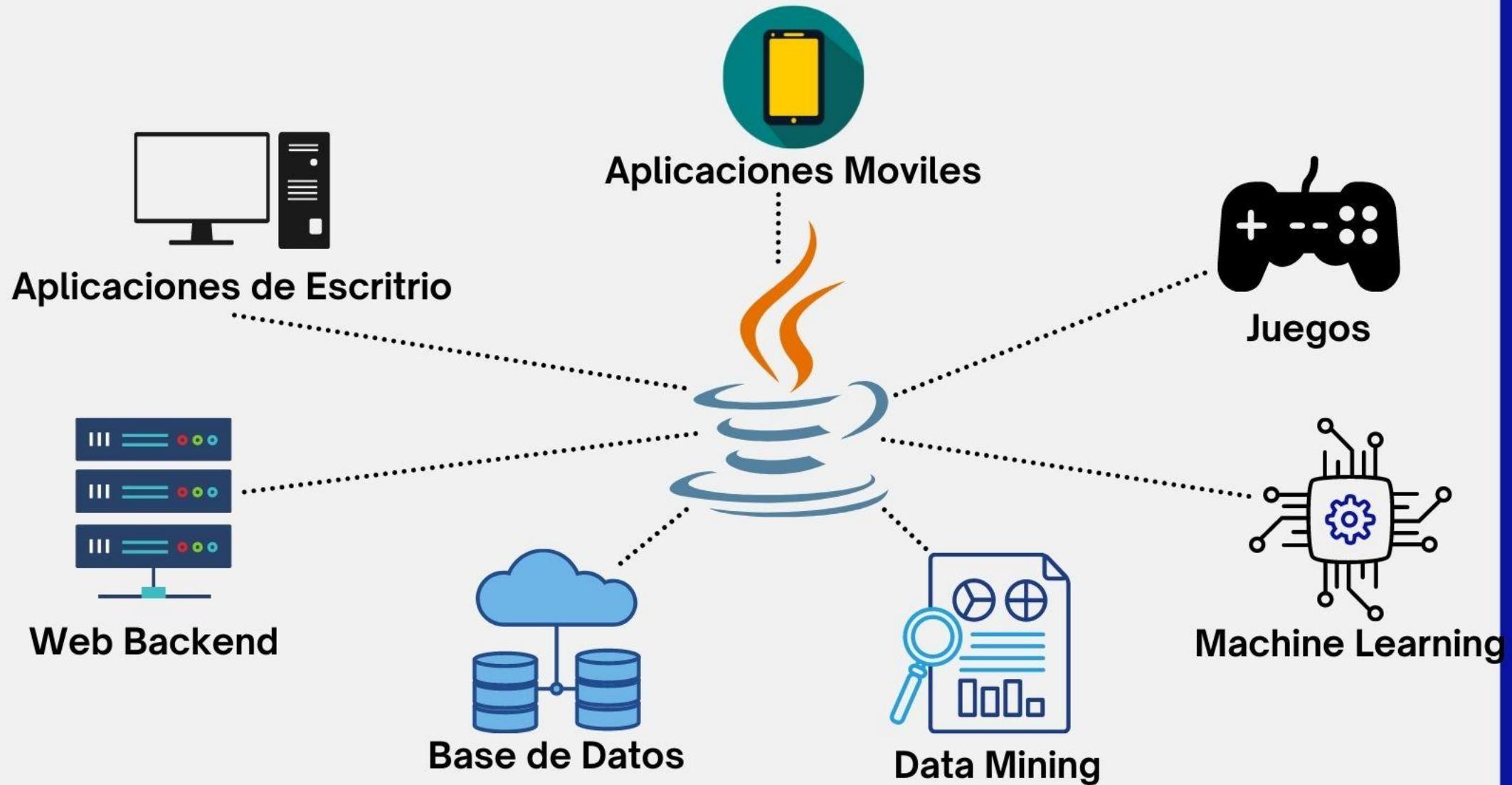


## Java Micro Edition (Java ME)

Desarrollo de aplicaciones para dispositivos móviles como teléfonos celulares, microcontroladores, decodificadores, televisores etc.



# ¿Dónde se usa Java?





# Características clave y ventajas de la tecnología Java

Simple, Orientado a Objetos

Distribuido, Interpretado

Robusto, Seguro

Arquitectura Neutra, Portable

Alto rendimiento, Multiproceso

Dinámico, Varias plataformas de hardware y SO



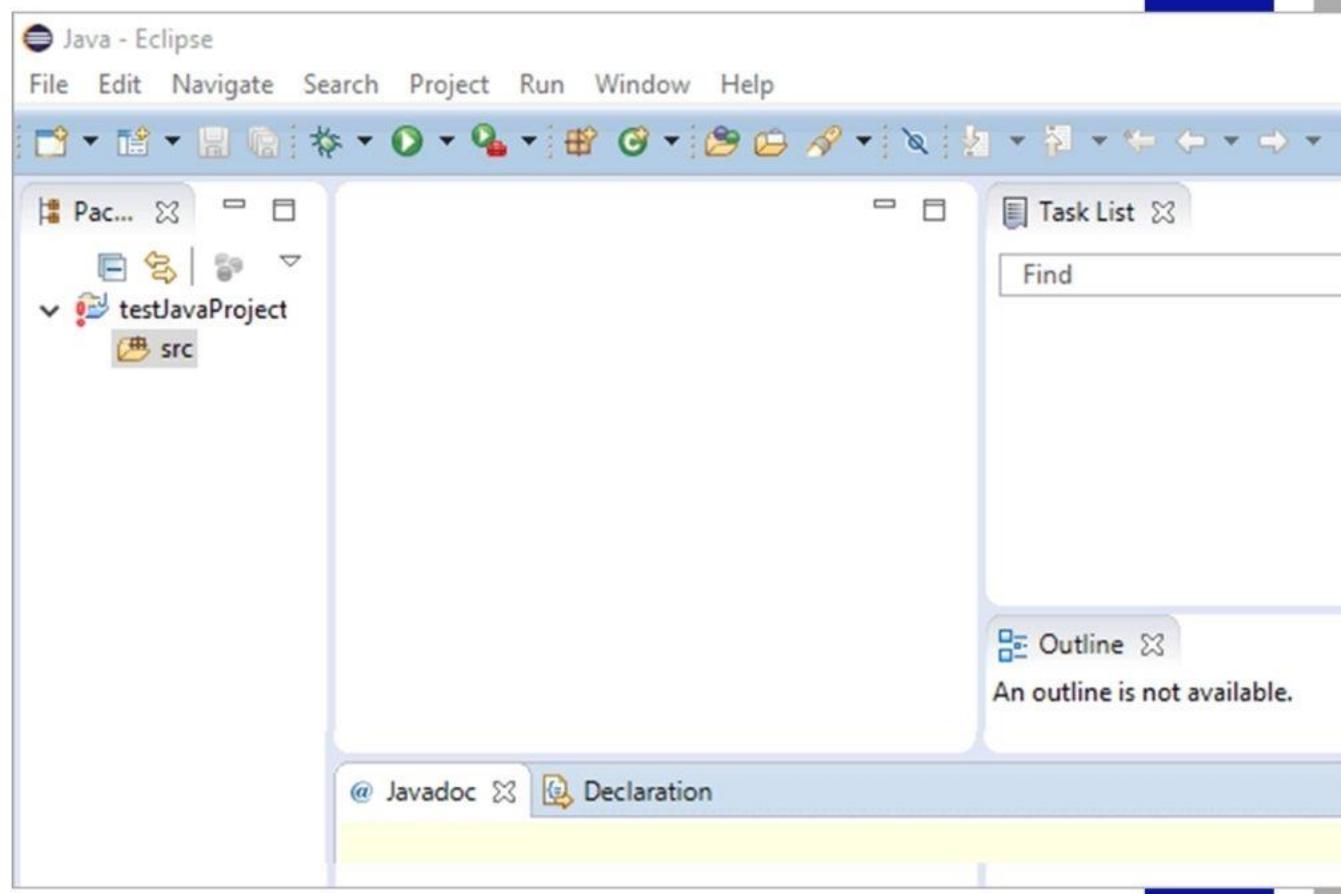
**James Gosling**



# ¿Que necesito para empezar a programar?

Un Entorno de Desarrollo Integrado (IDE)

Software que nos permite el desarrollo de programas de una manera rápida y eficaz presionando un botón para compilar y ejecutar .



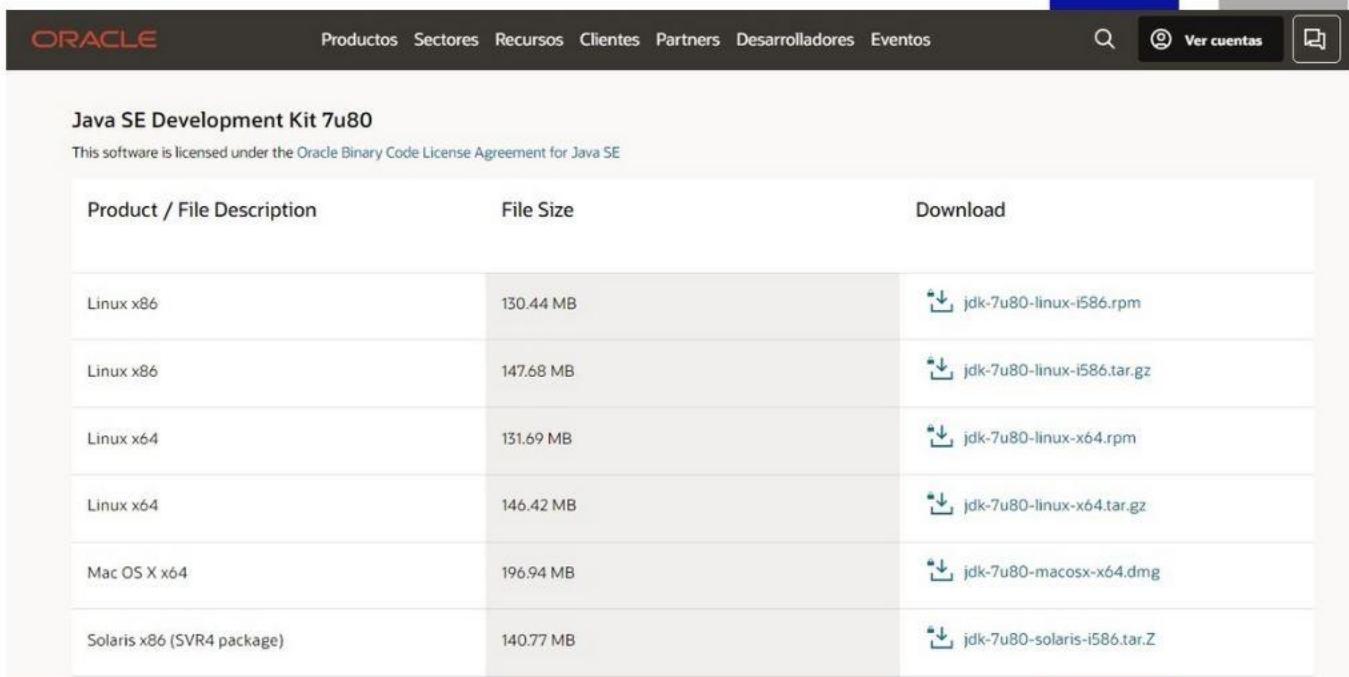
IDE Eclipse



# ¿Que necesito para empezar a programar?

## Java Development Kit (JDK)

Software para los desarrolladores que incluye herramientas como: compilador, depurador, desensamblador y generador de documentación



The screenshot shows the Oracle Java SE Development Kit 7u80 download page. The header includes the Oracle logo and navigation links for Products, Sectores, Recursos, Clientes, Partners, Desarrolladores, and Eventos. A search bar and user account links are also present. The main content displays a table with download links for various Java versions across different platforms.

Product / File Description	File Size	Download
Linux x86	130.44 MB	<a href="#">jdk-7u80-linux-i586.rpm</a>
Linux x86	147.68 MB	<a href="#">jdk-7u80-linux-i586.tar.gz</a>
Linux x64	131.69 MB	<a href="#">jdk-7u80-linux-x64.rpm</a>
Linux x64	146.42 MB	<a href="#">jdk-7u80-linux-x64.tar.gz</a>
Mac OS X x64	196.94 MB	<a href="#">jdk-7u80-macosx-x64.dmg</a>
Solaris x86 (SVR4 package)	140.77 MB	<a href="#">jdk-7u80-solaris-i586.tar.Z</a>



# verificar si todo esta instalado

## Java Development Kit (JDK )

Ejecutaremos los siguientes comandos en la consola:

**java -version**

**javac -version**

A screenshot of a Windows command prompt window titled 'cmd.exe'. The window shows the following text:

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19044.1526]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\pavel>javac -version
javac 1.8.0_241

C:\Users\pavel>java -version
java version "1.8.0_321"
Java(TM) SE Runtime Environment (build 1.8.0_321-b07)
Java HotSpot(TM) 64-Bit Server VM (build 25.321-b07, mixed mode)

C:\Users\pavel>
```

The window has a dark theme and is set against a white background with blue vertical bars on the right side.

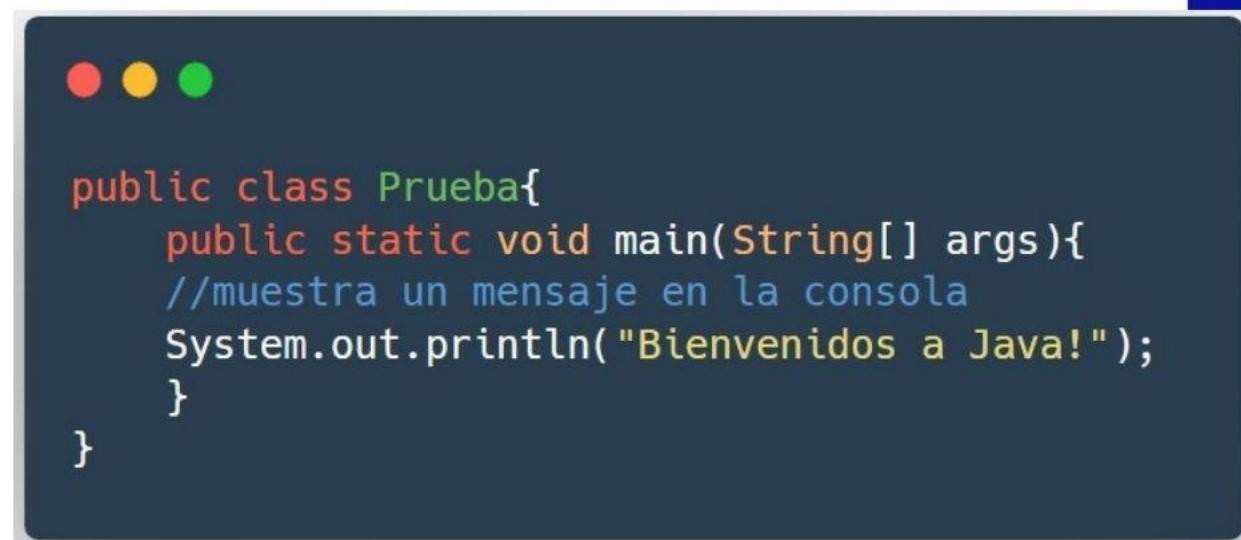


# verificar si todo esta instalado

Una prueba rápida.

Ejecutaremos los siguientes comandos en la consola:

**javac Prueba.java**  
**java Prueba**



A screenshot of a Java code editor window. At the top, there are three colored dots (red, yellow, green). The code itself is a simple Java class named 'Prueba' with a main method that prints a welcome message to the console.

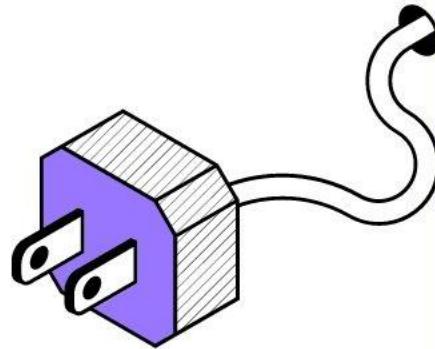
```
public class Prueba{  
    public static void main(String[] args){  
        //muestra un mensaje en la consola  
        System.out.println("Bienvenidos a Java!");  
    }  
}
```



# ¿Qué es un Objeto?

Definición dentro de la POO

- Un objeto representa una **entidad** en el mundo real que se puede identificar claramente.
- Un objeto tiene una **identidad**, un **estado** y un **comportamiento** únicos.





## Estado de un Objeto

El estado de un objeto (también conocido como sus propiedades o atributos) está representado por campos de datos con sus valores actuales.

## Comportamiento de un Objeto

El comportamiento de un objeto (también conocido como sus acciones) se define mediante métodos.

## De donde proviene un Objeto

Los objetos del mismo tipo se definen utilizando una clase común. Una clase es una plantilla o modelo que define cuáles serán los campos de datos y métodos de un objeto.

## Relación entre Clase y Objeto

Un objeto es la instancia de una clase. Puede crear muchas instancias de cierta clase. La creación de una instancia se conoce como **instanciación**.



# Conceptos importantes

Las siguientes cincuenta palabras clave están reservadas para uso del lenguaje Java:

<code>abstract</code>	<code>double</code>	<code>int</code>	<code>super</code>
<code>assert</code>	<code>else</code>	<code>interface</code>	<code>switch</code>
<code>boolean</code>	<code>enum</code>	<code>long</code>	<code>synchronized</code>
<code>break</code>	<code>extends</code>	<code>native</code>	<code>this</code>
<code>byte</code>	<code>final</code>	<code>new</code>	<code>throw</code>
<code>case</code>	<code>finally</code>	<code>package</code>	<code>throws</code>
<code>catch</code>	<code>float</code>	<code>private</code>	<code>transient</code>
<code>char</code>	<code>for</code>	<code>protected</code>	<code>try</code>
<code>class</code>	<code>goto</code>	<code>public</code>	<code>void</code>
<code>const</code>	<code>if</code>	<code>return</code>	<code>volatile</code>
<code>continue</code>	<code>implements</code>	<code>short</code>	<code>while</code>
<code>default</code>	<code>import</code>	<code>static</code>	
<code>do</code>	<code>instanceof</code>	<code>strictfp*</code>	



# Conceptos importantes

Las siguientes cincuenta palabras clave están reservadas para uso del lenguaje Java:

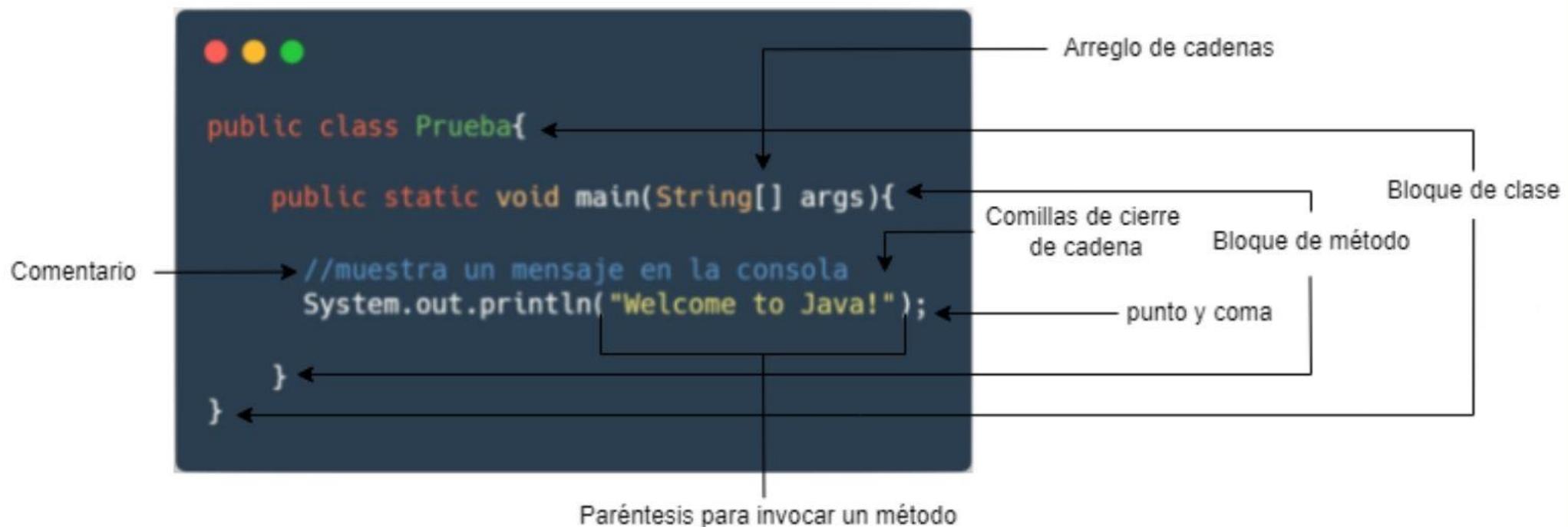
---

Carácter	Nombre	Descripción
{}	Llaves de apertura y cierre	Denota un bloque para encerrar sentencias.
()	Paréntesis de apertura y cierre	Se utiliza con métodos.
[]	Corchetes de apertura y cierre	Denota una matriz.
//	Doble slash	Preceda una línea de comentario.
""	Comillas de apertura y cierre	Encierre una cadena (es decir, una secuencia de caracteres).
;	Punto y coma	Marcar el final de una declaración.



# Elementos de un programa

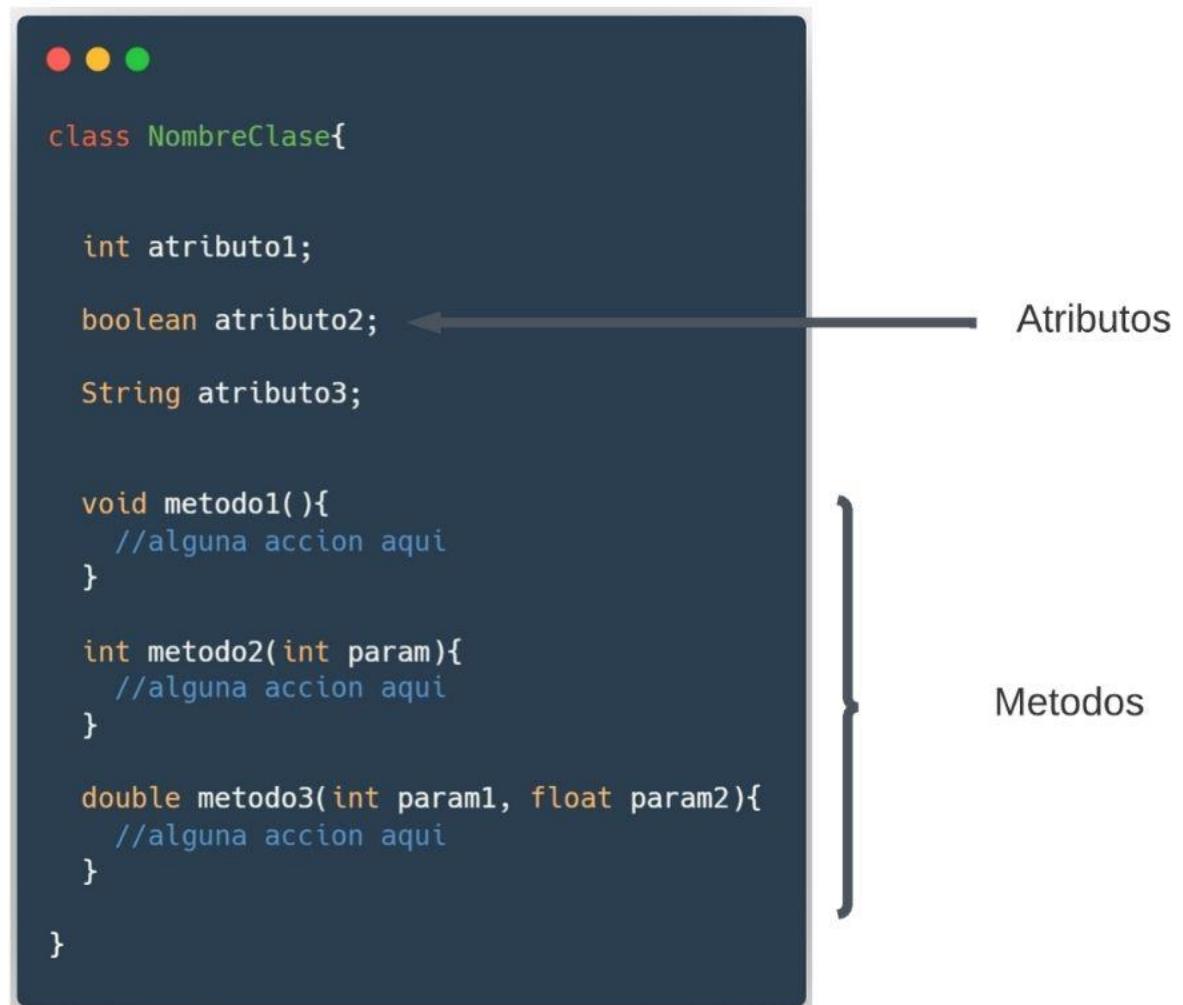
muestra un mensaje en consola





# Componentes de una clase

todas las clases tienen esta estructura básica

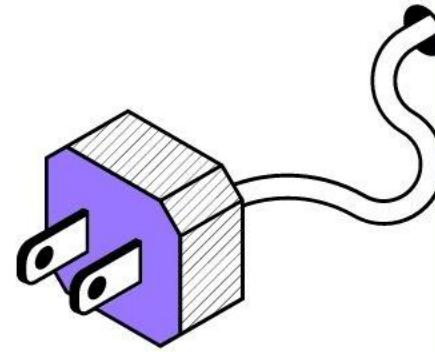




# ¿Qué es una variable?

## Definición de variable

- Las variables se utilizan para representar valores que pueden cambiar durante la ejecución de un programa.
- Las variables tienen un nombre, que es un identificador para hacer referencia a ellas en nuestros programas.



$x$   
 $\diagup$   
 $y$

$f(x)$



## Reglas para definir el nombre de una variable en Java

- Un identificador es una secuencia de caracteres que consta de letras, dígitos, guiones bajos (\_) y signos de dólar (\$).
- Un identificador debe comenzar con una letra, un guion bajo (\_) o un signo de dólar (\$). No puede comenzar con un dígito.
- Un identificador no puede ser una palabra reservada.
- La sintaxis para declarar una variable es **tipo\_de\_dato nombre\_variable;**

## Notas y tips para definir nombres de variables.

- Dado que Java distingue entre mayúsculas y minúsculas, **area**, **Area** y **AREA** son identificadores diferentes.
- Los identificadores descriptivos hacen que los programas sean fáciles de leer.



# Declaración variables

Ejemplo donde se declaran tres variables.

```
● ● ●

public class Prueba{

    public static void main(String[] args){

        //Ejemplo DECLARACION de variables
        int contador;
        double radio;
        double tasaInteres;

    }
}
```



## Asignar valor a una variable

- Una declaración de asignación designa un valor para una variable. Una declaración de asignación se puede utilizar como una expresión en Java.
- Después de declarar una variable, puede asignarle un valor. En Java, el signo igual (=) se utiliza como operador de asignación.
- Un identificador no puede ser una palabra reservada.
- La sintaxis para asignar valor a una variable es *nombre\_variable = expresión o valor;*

## Notas y tips para expresiones de asignación.

- Una expresión representa un cálculo que involucra valores, variables y operadores que, tomándolos juntos, se evalúa como un valor.



# Asignar valor a una variable

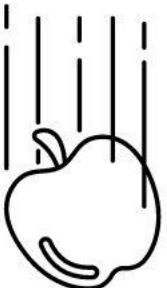
Ejemplo donde se declaran tres variables y se les asigna un valor.

```
public class Prueba{  
  
    public static void main(String[] args){  
  
        //Ejemplo ASIGNACION a variables  
        int contador = 1;  
        double radio = 1.0;  
        int x = 5 * (3 / 2);  
        double area;  
        contador = x + 1;  
        area = radius * radius * 3.14159;  
  
    }  
}
```



$\pi$

$e$



## Constantes

- Una constante es un identificador que representa un valor permanente.
- El valor de una variable puede cambiar durante la ejecución de un programa, pero una constante, representa datos permanentes que nunca cambian.
- La sintaxis para asignar valor a una variable es **final datatype NOMBRE\_CONSTANTE = valor;**

## Notas y tips para expresiones de asignación.

- Una constante debe declararse e inicializarse en la misma instrucción.



# Declaración Constante

Ejemplo donde se declaran cuatro constantes y sus valores.

```
● ● ●

public class Prueba{

    public static void main(String[] args){

        //Ejemplo declaracion de constantes
        final double PI = 3.14159;
        final double E = 2.7182;
        final int VALOR_MINIMO = 1;
        final int VALOR_MAXIMO = 20;

    }
}
```



# Tipo de datos primitivos

Diferencia entre tipos primitivos y objetos.



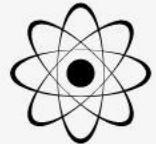
## No poseen atributos

Los tipos de datos no poseen atributo alguno, estos solo guardan un valor en alguna dirección en memoria.



## No poseen métodos

Los tipos de datos no poseen métodos, de hecho estos solo son utilizados por los métodos en partes del programa.



## Son indivisibles, atómicos

Con baja latencia hay menos retrasos; da una reacción genuina en tiempo real.



# Tipo de datos numéricos

Entre los tipos numéricos tenemos: **enteros** y de **punto flotante**

**byte**

$-2^7$  to  $2^7 - 1$  (-128 to 127)

8 bits, con signo

**int**

$-2^{31}$  to  $2^{31} - 1$  (-2147483648 to 2147483647)

32 bits, con signo

**float**

32 bits, IEEE 754

Negative range: -3.4028235E+38 to -1.4E-45  
Positive range: 1.4E-45 to 3.4028235E+38

**short**

16 bits, con signo

$-2^{15}$  to  $2^{15} - 1$  (-32768 to 32767)

**long**

64 bits, con signo

$-2^{63}$  to  $2^{63} - 1$

(i.e., -9223372036854775808 to 9223372036854775807)

**double**

64 bits, IEEE 754

Negative range: -1.7976931348623157E+308 to -4.9E-324

Positive range: 4.9E-324 to 1.7976931348623157E+308





# Tipo de datos no numéricos

Entre los no numéricos, tenemos los de carácter y los booleanos.

## boolean

Un tipo de datos booleano declara una variable con el valor **true** (verdadero) o **false** (falso)



A

## char

Un tipo de datos de carácter representa un único carácter.

## "String"

Una cadena es una secuencia de caracteres.  
No es primitivo, pero ...





# Operadores aritméticos

Los operadores aritméticos se utilizan para modificar variables.

Simbolo	Nombre	Ejemplo	Resultado
+	Adición	34 + 1	35
-	Sustracción	34.0 - 0.1	33.9
*	Multiplicación	300 * 30	9000
/	División	1.0 / 2.0	0.5
%	residuo o modulo	20 % 3	2



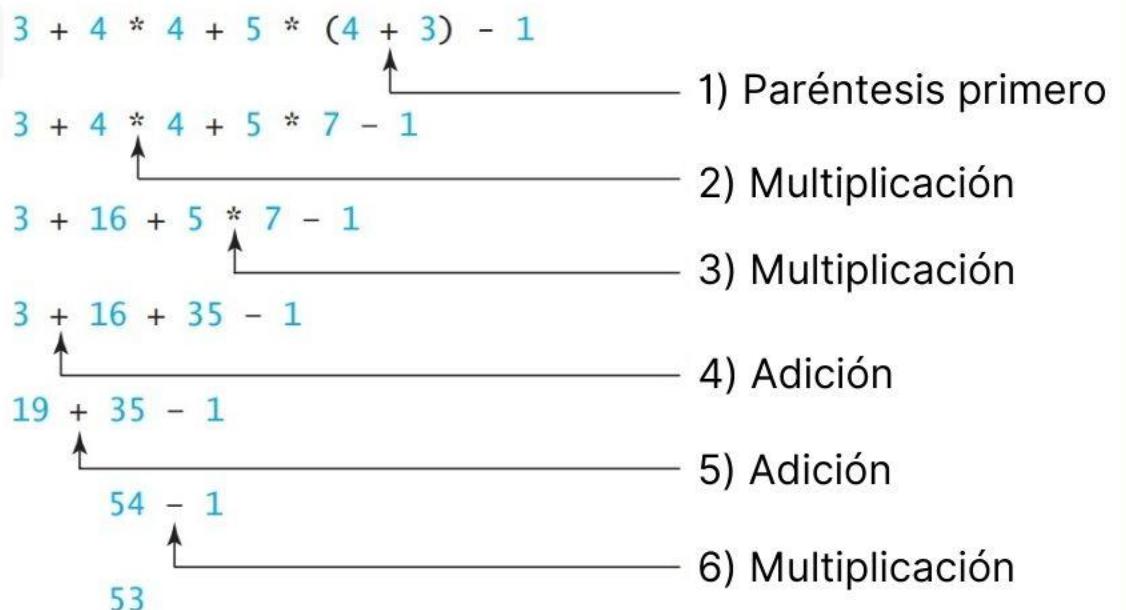


# Jerarquía de los operadores aritméticos

Los operadores aritméticos se utilizan para modificar variables.

## Constantes

- La jerarquía funciona tal y como es en la aritmética.
- Primero los paréntesis, pudiendo estos estar anidados.
- Después multiplicación, división y resto, modulo o residuo. Si una expresión contiene varios operadores de multiplicación, división y modulo , se aplican de izquierda a derecha.
- Por ultimo adición y sustracción. Si una expresión contiene varios operadores de suma y resta, se aplican de izquierda a derecha.



# Ejercicios practicos



- Dado un numero entero de segundos, calcular a cuantos minutos equivale y cuantos segundos restantes quedan. Ejemplo :  
500 segundos equivalen a 8 minutos y 20 segundos.
- Dado un numero de grados Fahrenheit, convertirlos a Celsius.
- Muestre la hora actual con realizando cálculos y expresiones aritméticas. \*\*



# Operadores de asignación aumentados.

Los operadores +, -, \*, / y % se pueden combinar con el operador de asignación para formar operadores aumentados.

Operador	Nombre	Ejemplo	Equivalente
<code>+=</code>	Asignación de adición	<code>i += 8</code>	<code>i = i + 8</code>
<code>-=</code>	Asignación de resta	<code>i -= 8</code>	<code>i = i - 8</code>
<code>*=</code>	Asignación de multiplicación	<code>i *= 8</code>	<code>i = i * 8</code>
<code>/=</code>	Asignación de división	<code>i /= 8</code>	<code>i = i / 8</code>
<code>%=</code>	Asignación de modulo	<code>i %= 8</code>	<code>i = i % 8</code>



# Operadores de incremento y decrecimiento.

Los operadores de incremento (++) y decremento (--) son para incrementar y decrementar una variable en 1.

Operador	Nombre	Descripción	Ejemplo (i = 1)
<code>++var</code>	preincremento	Incrementa var en 1 y usa el nuevo valor de var en la declaración	<code>int j = ++i;</code> <code>// j es 2, i es 2</code>
<code>var++</code>	postincremento	Incrementa var en 1, pero use el valor original de var en la declaración	<code>int j = i++;</code> <code>// j es 1, i es 2</code>
<code>--var</code>	predecremento	Disminuye var en 1 y usa el nuevo valor de var en la declaración	<code>int j = - -i;</code> <code>// j es 0, i es 0</code>
<code>var- -</code>	postdecremento	Disminuye var en 1 y usa el valor original de var en la declaración	<code>int j = i- -;</code> <code>// j es 1, i es 0</code>



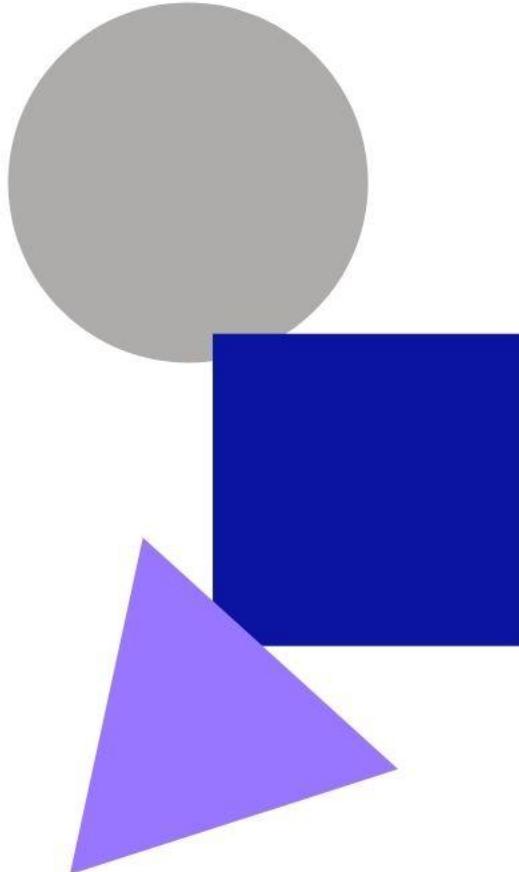
# Variables de campo

Declaración e inicialización de variables de campo

```
public class Circulo{  
  
    //declaracion e inicializacion  
    //de una constante  
    final double PI = 3.1416;  
  
    //variables de campo  
  
    //declaracion e inicializacion  
    double radio = 1;  
  
    //declaracion  
    double area;  
    double perimetro;  
  
    void computeArea(){  
        area = radio * radio * PI;  
    }  
  
    void computePerimetro(){  
        perimetro = 2 * radio * PI;  
    }  
  
    void setRadio(double newRadio ){  
        radio = newRadio;  
    }  
}
```

Todo lo que este dentro de los parentesis que definen la clase pero fuera de los parentesis de algun metodo , son variables de campo, es decir atributos de la clase.

Las variables definidas dentro de los parentesis de los metodos, no son variables de campo.  
*(aqui no hay ninguna declaracion)*



## Estado de un Objeto

El estado de un objeto (también conocido como sus propiedades o atributos) está representado por campos de datos con sus valores actuales.

## Comportamiento de un Objeto

El comportamiento de un objeto (también conocido como sus acciones) se define mediante métodos.

## De donde proviene un Objeto

Los objetos del mismo tipo se definen utilizando una clase común. Una clase es una plantilla o modelo que define cuáles serán los campos de datos y métodos de un objeto.

## Relación entre Clase y Objeto

Un objeto es la instancia de una clase. Puede crear muchas instancias de cierta clase. La creación de una instancia se conoce como **instanciación**.



## Ejemplo de creación de objetos

- Las clases son definiciones de objetos y los objetos se crean a partir de clases.

```
public class Pruebas {  
  
    /** Metodo principal */  
    public static void main(String[] args) {  
  
        // Creamos un circulo con radio = 1;  
  
        Circulo circulo1 = new Circulo();  
        System.out.println("El area del circulo 1 de radio "  
        circle1.radio + " es " + circulo1.getArea());  
  
        // Creamos un circulo con radio = 1, y despues es cambiado;  
  
        Circulo circulo2 = new Circulo();  
        circulo2.setRadio(25);  
        System.out.println("El area del circulo 2 de radio "  
        circle2.radio + " es " + circulo2.getArea());  
  
        // Modificamos el radio de circulo1  
        //calculamos de nuevo el area  
        circulo1.radio = 8;  
        System.out.println("El area del circulo 1 de radio "  
        circle1.radio + " es " + circulo1.getArea());  
  
    }  
}
```

# Ejercicios practicos



Cree objetos de las clases Rectángulo, Televisor y la que usted diseño (creadas en la sección 2), además use sus atributos y métodos.



## Descripción general de la documentación de Java y cómo usarla

Existen muchas clases predefinidas en java, que son la base de funcionamiento. Resulta útil tener una documentación en la cual sea posible consultar todo la información relacionada con una clase, para hacer un correcto uso de ella en nuestros programas.

The screenshot shows the Java™ Platform, Standard Edition 7 API Specification documentation. The top navigation bar includes links for Overview, Package, Class, Use, Tree, Deprecated, Index, and Help. The left sidebar has sections for Java™ Platform Standard Ed. 7, All Classes, and Packages. The Packages section lists several packages: java.applet, java.awt, java.awt.color, java.awt.datatransfer, java.awt.dnd, java.awt.event, java.awt.font, java.awt.geom, java.awt.im, and java.awt.im.sni. The main content area displays the title "Java™ Platform, Standard Edition 7 API Specification" and a brief description stating it is the API specification for the Java™ Platform, Standard Edition. Below this is a "See: Description" link. A table titled "Packages" provides detailed descriptions for each package listed in the sidebar. The table has two columns: "Package" and "Description".

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing beans -- components based on the JavaBeans™ architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.



## La clase String, StringBuilder y StringBuffer

- Las clases **String**, **StringBuilder** y **StringBuffer** se utilizan para procesar cadenas.
- Una cadena es una secuencia de caracteres. En Java una cadena se trata como un objeto.

### La clase String



- Un objeto String es inmutable: su contenido no se puede cambiar una vez que se crea la cadena.
- Un objeto String es inmutable; su contenido no se puede cambiar. ¿El siguiente código cambia el contenido de la cadena?
- `String s = "Java";  
s = "HTML";`

Spoiler NO!

# Ejercicios practicos



- Cree objetos de la clase String.
- Verificar si dos cadenas son iguales.
- Dadas dos cadenas, y suponiendo que las queremos colocar en orden , verificar cual se coloca primero.
- Contar cuantas letras tiene una cadena.



## La clase String, StringBuilder

- Las clases **String, StringBuilder** se utilizan para procesar cadenas.
- Una cadena es una secuencia de caracteres. En Java una cadena se trata como un objeto.

## La clase StringBuilder



- La clase StringBuilder es similar a la clase String excepto que la clase String es inmutable.
- En general, la clase StringBuilder se puede usar de la misma manera que la clase String, solo que esta clase es más flexible.
- Puedes agregar más contenido a un objeto de la clase StringBuilder, mientras que eso no es posible con la clase String.

# Ejercicios practicos



- Cree objetos de las clases String y StringBuilder.
- Contrastar las diferencias que tienen.
- Generar una cadena invertida.



## Almacenamiento de los objetos en la memoria.

- Todos los datos para variables de tipo primitivo, son guardados en un lugar de la memoria de nuestra computadora, llamada Stack.
- Para los tipos que usan referencias (Es decir para los objetos), lo que se guarda también en el Stack, es la referencia (dirección de memoria) donde se ubica dicho objeto.
- Cuando tenemos algo como:  
Circulo **circulo1** = new Circulo();  
Circulo **circulo2** = **circulo1**;  
**circulo2** solo tiene una copia de la referencia de **circulo1**, ambos hacen referencia al mismo objeto.



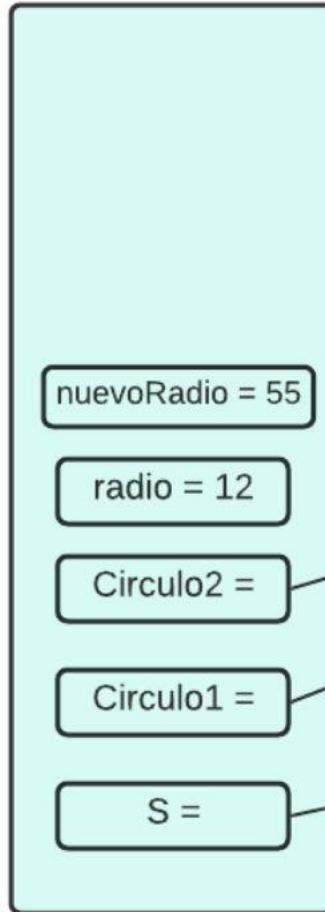


# Almacenamiento de los objetos en la memoria.

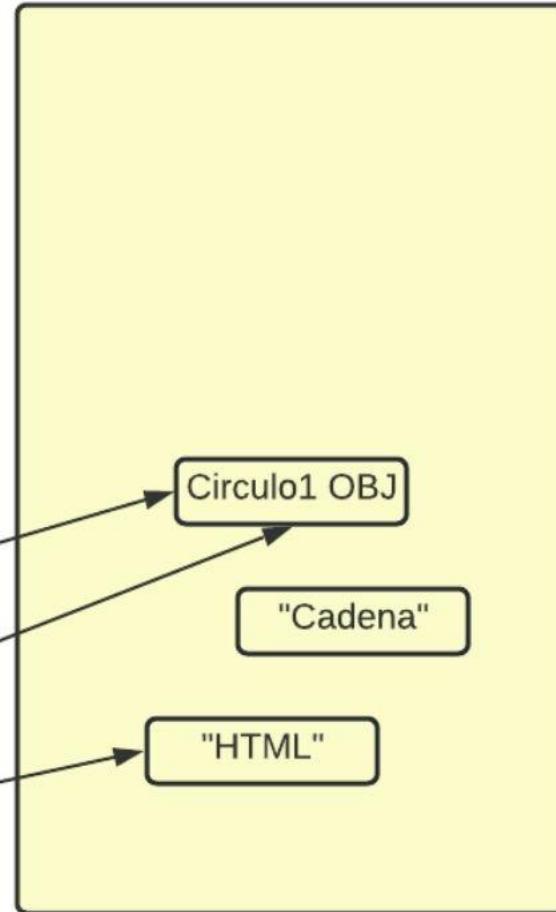
Programa

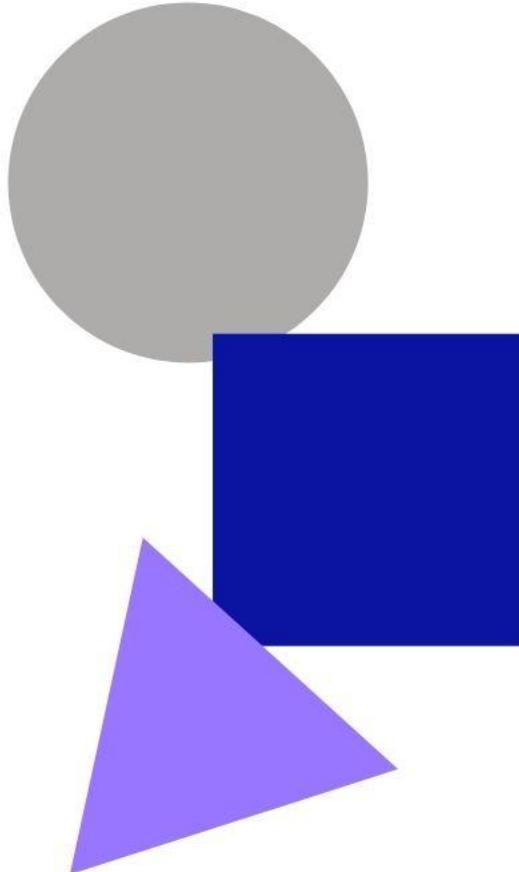
```
● ● ●  
public class Pruebas {  
  
    /** Metodo principal */  
    public static void main(String[] args) {  
  
        String s = "Cadena";  
        S = "HTML";  
  
        Circulo circulo1 = new Circulo();  
        Circulo circulo2 = circulo1;  
  
        int radio = 12;  
        int nuevoRadio = radio;  
        int nuevoRadio  = 55;  
    }  
}
```

Stack



Heap





## Estado de un Objeto

El estado de un objeto (también conocido como sus propiedades o atributos) está representado por campos de datos con sus valores actuales.

## Comportamiento de un Objeto

El comportamiento de un objeto (también conocido como sus acciones) se define mediante métodos.

## De donde proviene un Objeto

Los objetos del mismo tipo se definen utilizando una clase común. Una clase es una plantilla o modelo que define cuáles serán los campos de datos y métodos de un objeto.

## Relación entre Clase y Objeto

Un objeto es la instancia de una clase. Puede crear muchas instancias de cierta clase. La creación de una instancia se conoce como **instanciación**.



## Ejemplo de creación de objetos

- Las clases son definiciones de objetos y los objetos se crean a partir de clases.

```
public class Pruebas {  
  
    /** Metodo principal */  
    public static void main(String[] args) {  
  
        // Creamos un circulo con radio = 1;  
  
        Circulo circulo1 = new Circulo();  
        System.out.println("El area del circulo 1 de radio "  
        circle1.radio + " es " + circulo1.getArea());  
  
        // Creamos un circulo con radio = 1, y despues es cambiado;  
  
        Circulo circulo2 = new Circulo();  
        circulo2.setRadio(25);  
        System.out.println("El area del circulo 2 de radio "  
        circle2.radio + " es " + circulo2.getArea());  
  
        // Modificamos el radio de circulo1  
        //calculamos de nuevo el area  
        circulo1.radio = 8;  
        System.out.println("El area del circulo 1 de radio "  
        circle1.radio + " es " + circulo1.getArea());  
  
    }  
}
```

# Ejercicios practicos



Cree objetos de las clases Rectángulo, Televisor y la que usted diseño (creadas en la sección 2), además use sus atributos y métodos.



## Descripción general de la documentación de Java y cómo usarla

Existen muchas clases predefinidas en java, que son la base de funcionamiento. Resulta útil tener una documentación en la cual sea posible consultar todo la información relacionada con una clase, para hacer un correcto uso de ella en nuestros programas.

The screenshot shows the Java™ Platform, Standard Edition 7 API Specification documentation. The top navigation bar includes links for Overview, Package, Class, Use, Tree, Deprecated, Index, and Help. The left sidebar has sections for Java™ Platform Standard Ed. 7, All Classes, and Packages. The Packages section lists several Java packages: java.applet, java.awt, java.awt.color, java.awt.datatransfer, java.awt.dnd, java.awt.event, java.awt.font, java.awt.geom, java.awt.im, and java.awt.im.sni. The main content area displays the title "Java™ Platform, Standard Edition 7 API Specification" and a brief description stating it is the API specification for the Java™ Platform, Standard Edition. Below this is a "See: Description" link. A table titled "Packages" provides detailed descriptions for each package listed in the sidebar. The table has two columns: "Package" and "Description".

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing beans -- components based on the JavaBeans™ architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.



## La clase String, StringBuilder y StringBuffer

- Las clases **String**, **StringBuilder** y **StringBuffer** se utilizan para procesar cadenas.
- Una cadena es una secuencia de caracteres. En Java una cadena se trata como un objeto.

### La clase String



- Un objeto String es inmutable: su contenido no se puede cambiar una vez que se crea la cadena.
- Un objeto String es inmutable; su contenido no se puede cambiar. ¿El siguiente código cambia el contenido de la cadena?
- `String s = "Java";  
s = "HTML";`

Spoiler NO!

# Ejercicios practicos



- Cree objetos de la clase String.
- Verificar si dos cadenas son iguales.
- Dadas dos cadenas, y suponiendo que las queremos colocar en orden , verificar cual se coloca primero.
- Contar cuantas letras tiene una cadena.



## La clase String, StringBuilder

- Las clases **String, StringBuilder** se utilizan para procesar cadenas.
- Una cadena es una secuencia de caracteres. En Java una cadena se trata como un objeto.

## La clase StringBuilder



- La clase StringBuilder es similar a la clase String excepto que la clase String es inmutable.
- En general, la clase StringBuilder se puede usar de la misma manera que la clase String, solo que esta clase es más flexible.
- Puedes agregar más contenido a un objeto de la clase StringBuilder, mientras que eso no es posible con la clase String.

# Ejercicios practicos



- Cree objetos de las clases String y StringBuilder.
- Contrastar las diferencias que tienen.
- Generar una cadena invertida.



## Almacenamiento de los objetos en la memoria.

- Todos los datos para variables de tipo primitivo, son guardados en un lugar de la memoria de nuestra computadora, llamada Stack.
- Para los tipos que usan referencias (Es decir para los objetos), lo que se guarda también en el Stack, es la referencia (dirección de memoria) donde se ubica dicho objeto.
- Cuando tenemos algo como:  
Circulo **circulo1** = new Circulo();  
Circulo **circulo2** = **circulo1**;  
**circulo2** solo tiene una copia de la referencia de **circulo1**, ambos hacen referencia al mismo objeto.



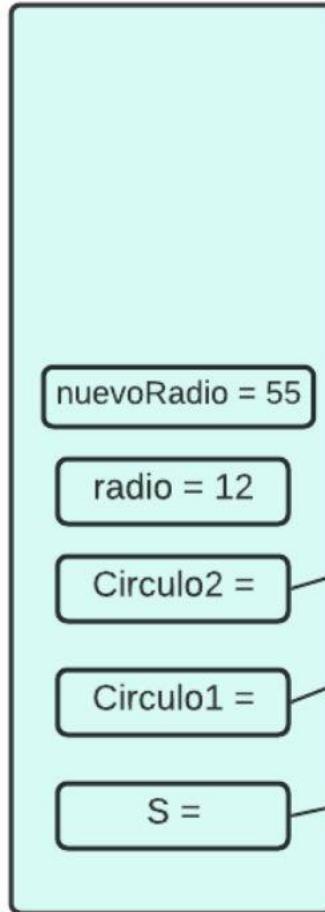


# Almacenamiento de los objetos en la memoria.

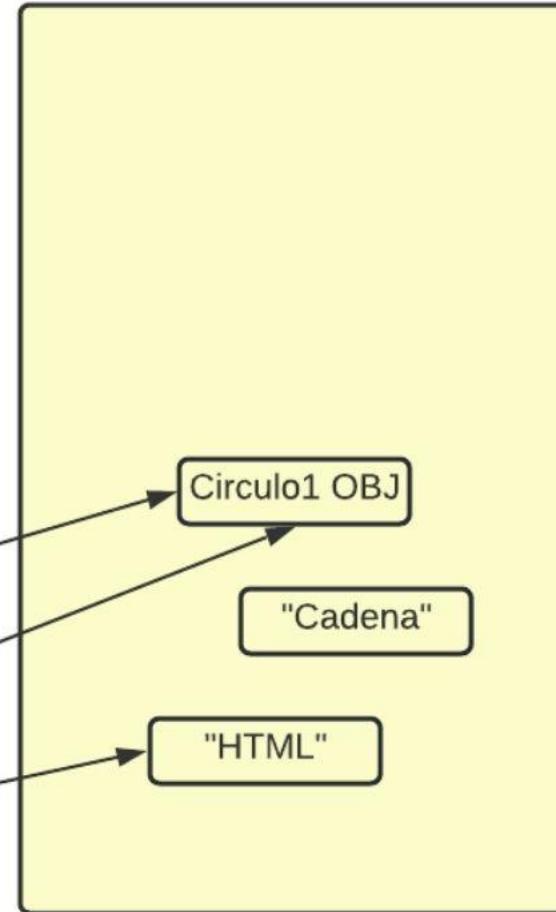
Programa

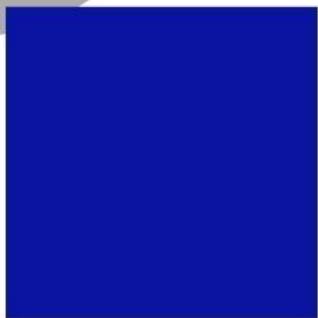
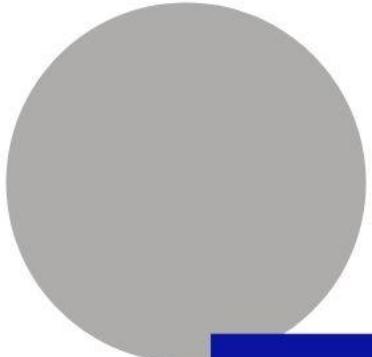
```
● ● ●  
public class Pruebas {  
  
    /** Metodo principal */  
    public static void main(String[] args) {  
  
        String s = "Cadena";  
        S = "HTML";  
  
        Circulo circulo1 = new Circulo();  
        Circulo circulo2 = circulo1;  
  
        int radio = 12;  
        int nuevoRadio = radio;  
        int nuevoRadio  = 55;  
    }  
}
```

Stack



Heap





## Que es un arreglo

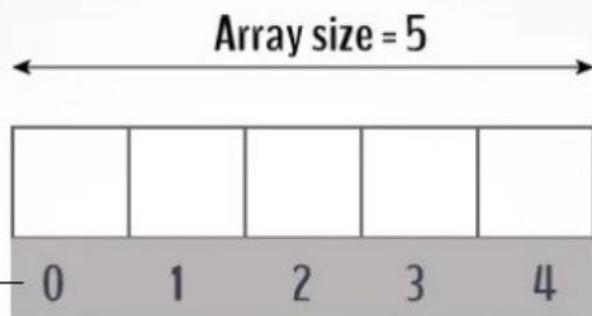
Es una colección de datos de un mismo tipo. Una sola variable de tipo arreglo o array puede hacer referencia a una gran colección de datos.

## Conceptos de arreglos

- Una vez que se crea un arreglo, su tamaño es fijo.
- Una variable de referencia de matriz se utiliza para acceder a los elementos de una matriz mediante un índice.

## Como se declara un arreglo en Java

**tipoElemento variableReferencia[ ] = new tipoElemento [tamaño]**





## Creación de un arreglo

```
double[] miArray = new double[10];
```

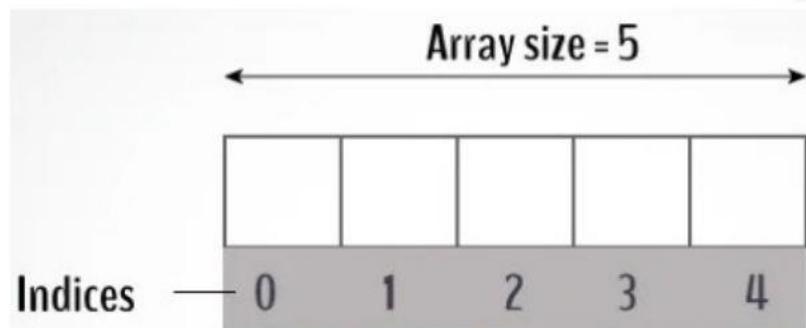
Se crea un arreglo, que guardara datos de tipo double, con una capacidad de 10. El arreglo tiene valores por defecto.

## Creación de un arreglo inicializado

```
double[] miArray= {1.9, 2.9, 3.4, 3.5};
```

Se crea un arreglo, que guarda los valores que están entre las llaves. El arreglo contiene valores específicos desde su creación.

## Asignar valores a un arreglo con valores por defecto.



```
miArreglo[0] = 4.0;  
miArreglo[1] = 34.33;  
miArreglo[2] = 34.0;  
miArreglo[3] = 45.45;  
miArreglo[4] = 99.993;  
miArreglo[5] = 11123;
```



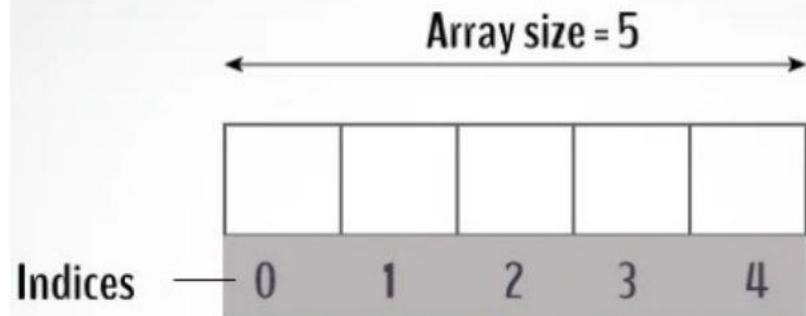
## Creación de un arreglo bidimensional.

```
int matriz[ ][ ] = new int[3][3];
```

## Creación de un arreglo bidimensional inicializado

```
int[ ][ ] miArreglo= {{1, 2, 3},  
                      {4, 5, 6},  
                      {7, 8, 9},  
                      {10, 11, 12}};
```

## Dar valores a elementos de un arreglo bidimensional.



```
miArreglo[0][0] = 1;  
miArreglo[0][1] = 47;  
miArreglo[0][2] = 88;  
miArreglo[1][0] = 19;  
miArreglo[1][1] = 17;  
miArreglo[1][2] = 189;
```



## Que son los paquetes de java

- Los paquetes son el mecanismo que usa Java para facilitar la modularidad del código.
- Un paquete puede contener una o más definiciones de interfaces y clases, distribuyéndose habitualmente como un archivo.
- Para utilizar los elementos de un paquete es necesario importar este en el módulo de código en curso, usando para ello la sentencia



## Como utilizo los paquetes

- Para utilizar los elementos de un paquete es necesario importar este en el módulo de código en curso, usando para ello la sentencia **import**.



## uso de import

- Importando una sola clase:

```
import java.util.Date;
```

- Importando un paquete completo de clases:

```
import java.util.*;
```

- Para utilizar los elementos de un paquete es necesario importar este en el módulo de código en curso, usando para ello la sentencia



## Utilidades con import

- puedes incluir múltiples sentencias **import**:

```
import java.util.Date;
```

```
import java.util.Calendar;
```

- Por default siempre se importa `java.lang.*`

- No necesitas importar las clases que estén en el mismo paquete.



## ArrayList

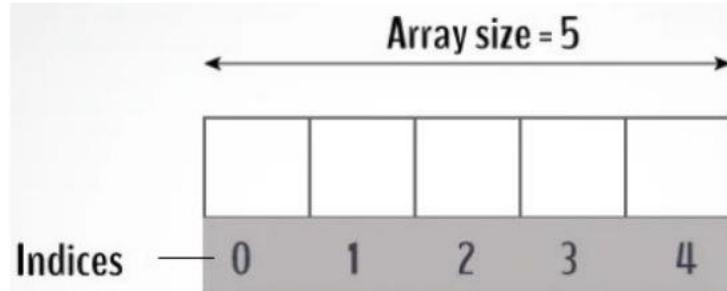
Una ArrayList se implementa usando un arreglo. En un arreglo convencional el tamaño de este era fijo una vez definido, y este no podía cambiar, en un ArrayList esto no es así, ya que el arreglo crece de manera dinámica, es decir su tamaño puede variar.

## Creación de un ArrayList

```
import java.util.*;  
  
ArrayList<String> miArrayList = new ArrayList<>();
```

## Agregar valores a un ArrayList

```
miArrayList.add("Arturo");  
miArrayList.add("Maria");  
miArrayList.add("Pedro");  
miArrayList.add("Fernanda");
```





## Arreglo de argumentos (`String[ ] args`)

Es un arreglo que guarda objetos de tipo cadena (String), este es pasado justo cuando ejecutamos nuestro programa en la linea de comandos.

### Pasando valores al arreglo (`String[ ] args`)

Cuando se ejecuta un programa, es posible pasar valores para que estos sean almacenados en el arreglo args.

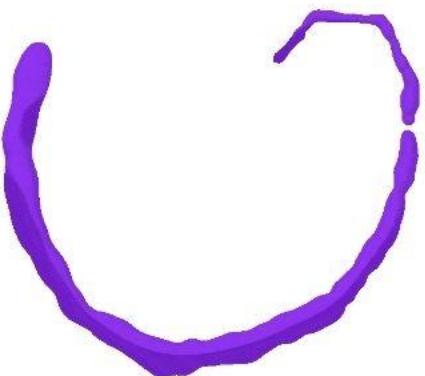
```
java nombrePrograma [argumento1, argumento2 ... ]
```

### Ejemplo

Suponiendo que tenemos un programa que recibe nombres de personas mediante la linea de comandos, para después imprimir un saludo para estas, la ejecucion seria la siguiente.

```
java SaludoPersonas Arturo Sofia Jorge Pedro Camila
```





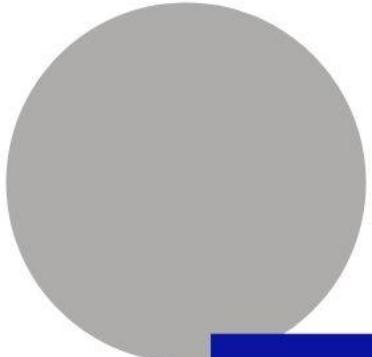
## Procesado de un arreglo de manera eficaz.

- Existen maneras de realizar tareas que son repetitivas.  
Para el procesado de un arreglo, tenemos al ciclo **for**.
- Con el ciclo **for** es posible recorrer los elementos de un arreglo, para acceder a sus datos o modificarlos de una manera eficaz.

## Ejemplo de uso del ciclo for con un arreglo.

Suponiendo que tenemos definido un arreglo de tamaño 5 y que la referencia se llama miArreglo, es posible recorrerlo con la siguiente instrucción:

```
for ( int i = 0; i < miArreglo.length; i++ ){  
    System.out.println(miArreglo[i]);  
}
```



## Que es un arreglo

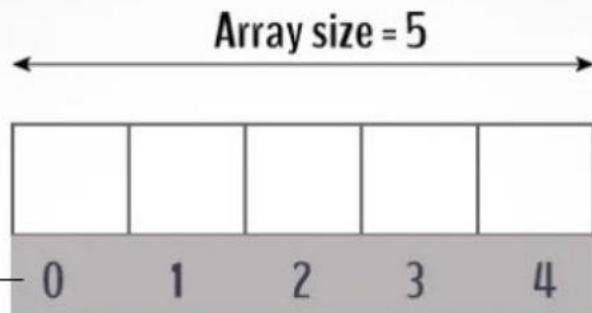
Es una colección de datos de un mismo tipo. Una sola variable de tipo arreglo o array puede hacer referencia a una gran colección de datos.

## Conceptos de arreglos

- Una vez que se crea un arreglo, su tamaño es fijo.
- Una variable de referencia de matriz se utiliza para acceder a los elementos de una matriz mediante un índice.

## Como se declara un arreglo en Java

**tipoElemento variableReferencia[ ] = new tipoElemento [tamaño]**





## Creación de un arreglo

```
double[] miArray = new double[10];
```

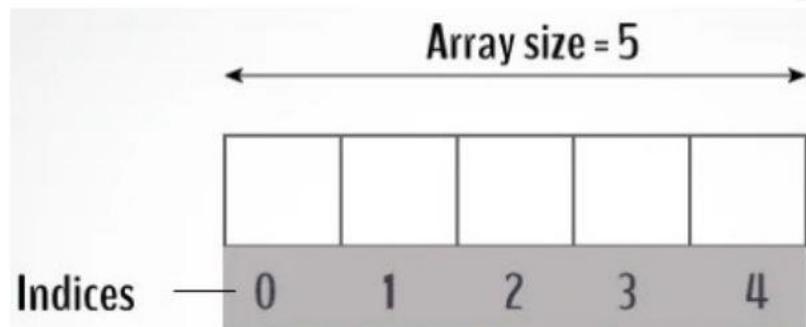
Se crea un arreglo, que guardara datos de tipo double, con una capacidad de 10. El arreglo tiene valores por defecto.

## Creación de un arreglo inicializado

```
double[] miArray= {1.9, 2.9, 3.4, 3.5};
```

Se crea un arreglo, que guarda los valores que están entre las llaves. El arreglo contiene valores específicos desde su creación.

## Asignar valores a un arreglo con valores por defecto.



```
miArreglo[0] = 4.0;  
miArreglo[1] = 34.33;  
miArreglo[2] = 34.0;  
miArreglo[3] = 45.45;  
miArreglo[4] = 99.993;  
miArreglo[5] = 11123;
```



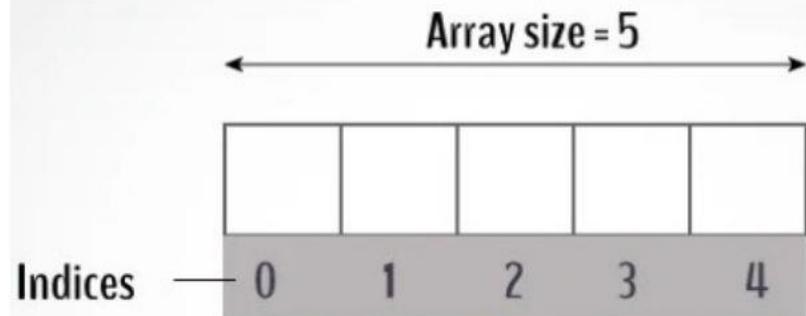
## Creación de un arreglo bidimensional.

```
int matriz[ ][ ] = new int[3][3];
```

## Creación de un arreglo bidimensional inicializado

```
int[ ][ ] miArreglo= {{1, 2, 3},  
                      {4, 5, 6},  
                      {7, 8, 9},  
                      {10, 11, 12}};
```

## Dar valores a elementos de un arreglo bidimensional.



```
miArreglo[0][0] = 1;  
miArreglo[0][1] = 47;  
miArreglo[0][2] = 88;  
miArreglo[1][0] = 19;  
miArreglo[1][1] = 17;  
miArreglo[1][2] = 189;
```



## Que son los paquetes de java

- Los paquetes son el mecanismo que usa Java para facilitar la modularidad del código.
- Un paquete puede contener una o más definiciones de interfaces y clases, distribuyéndose habitualmente como un archivo.
- Para utilizar los elementos de un paquete es necesario importar este en el módulo de código en curso, usando para ello la sentencia



## Como utilizo los paquetes

- Para utilizar los elementos de un paquete es necesario importar este en el módulo de código en curso, usando para ello la sentencia **import**.



## uso de import

- Importando una sola clase:

```
import java.util.Date;
```

- Importando un paquete completo de clases:

```
import java.util.*;
```

- Para utilizar los elementos de un paquete es necesario importar este en el módulo de código en curso, usando para ello la sentencia



## Utilidades con import

- puedes incluir múltiples sentencias **import**:

```
import java.util.Date;
```

```
import java.util.Calendar;
```

- Por default siempre se importa `java.lang.*`

- No necesitas importar las clases que estén en el mismo paquete.



## ArrayList

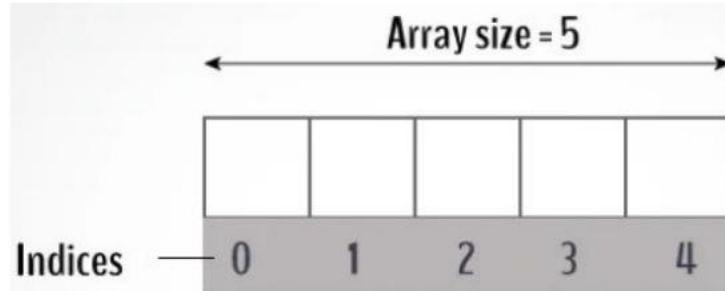
Una ArrayList se implementa usando un arreglo. En un arreglo convencional el tamaño de este era fijo una vez definido, y este no podía cambiar, en un ArrayList esto no es así, ya que el arreglo crece de manera dinámica, es decir su tamaño puede variar.

## Creación de un ArrayList

```
import java.util.*;  
  
ArrayList<String> miArrayList = new ArrayList<>();
```

## Agregar valores a un ArrayList

```
miArrayList.add("Arturo");  
miArrayList.add("Maria");  
miArrayList.add("Pedro");  
miArrayList.add("Fernanda");
```





## Arreglo de argumentos (`String[ ] args`)

Es un arreglo que guarda objetos de tipo cadena (String), este es pasado justo cuando ejecutamos nuestro programa en la linea de comandos.

### Pasando valores al arreglo (`String[ ] args`)

Cuando se ejecuta un programa, es posible pasar valores para que estos sean almacenados en el arreglo args.

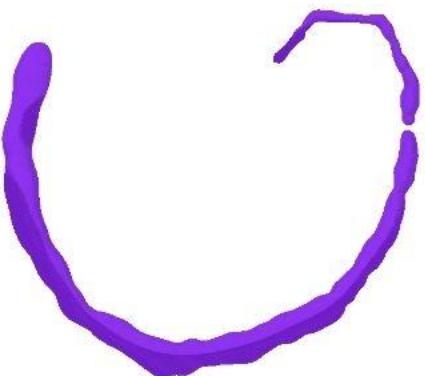
```
java nombrePrograma [argumento1, argumento2 ... ]
```

### Ejemplo

Suponiendo que tenemos un programa que recibe nombres de personas mediante la linea de comandos, para después imprimir un saludo para estas, la ejecucion seria la siguiente.

```
java SaludoPersonas Arturo Sofia Jorge Pedro Camila
```





## Procesado de un arreglo de manera eficaz.

- Existen maneras de realizar tareas que son repetitivas.  
Para el procesado de un arreglo, tenemos al ciclo **for**.
- Con el ciclo **for** es posible recorrer los elementos de un arreglo, para acceder a sus datos o modificarlos de una manera eficaz.

## Ejemplo de uso del ciclo for con un arreglo.

Suponiendo que tenemos definido un arreglo de tamaño 5 y que la referencia se llama miArreglo, es posible recorrerlo con la siguiente instrucción:

```
for ( int i = 0; i < miArreglo.length; i++ ){  
    System.out.println(miArreglo[i]);  
}
```



## Que son los modificadores de java

- Se usan en clases, constructores, métodos, datos y bloques de nivel de clase), pero el modificador **final** también se puede usar en variables locales en un método.
- Un modificador que se puede aplicar a una clase se denomina modificador de clase.
- Un modificador que se puede aplicar a un método se denomina modificador de método.
- Un modificador que se puede aplicar a un campo de datos se denomina modificador de datos.



## Modificadores existentes

- **public:** Una clase, constructor, método o campo de datos es visible para todos los programas de cualquier paquete.
- **private:** Un constructor, método o campo de datos solo es visible en esta clase.
- **protected:** Un constructor, método o campo de datos es visible en este paquete y en las subclases de esta clase en cualquier paquete.
- **static:** Define un método, un campo de datos o un bloque de inicialización estático.
- **final:** Una clase no puede heredar. Un método no se puede modificar en una subclase. Un campo de datos es una constante.



# Métodos

## Conceptos clave



### En que consiste un método

La definición de un método consiste en el nombre de este, sus parámetros, su tipo de valor de retorno y el cuerpo.

### Definición en Java

La definición de un método consta de un encabezado y de un cuerpo.

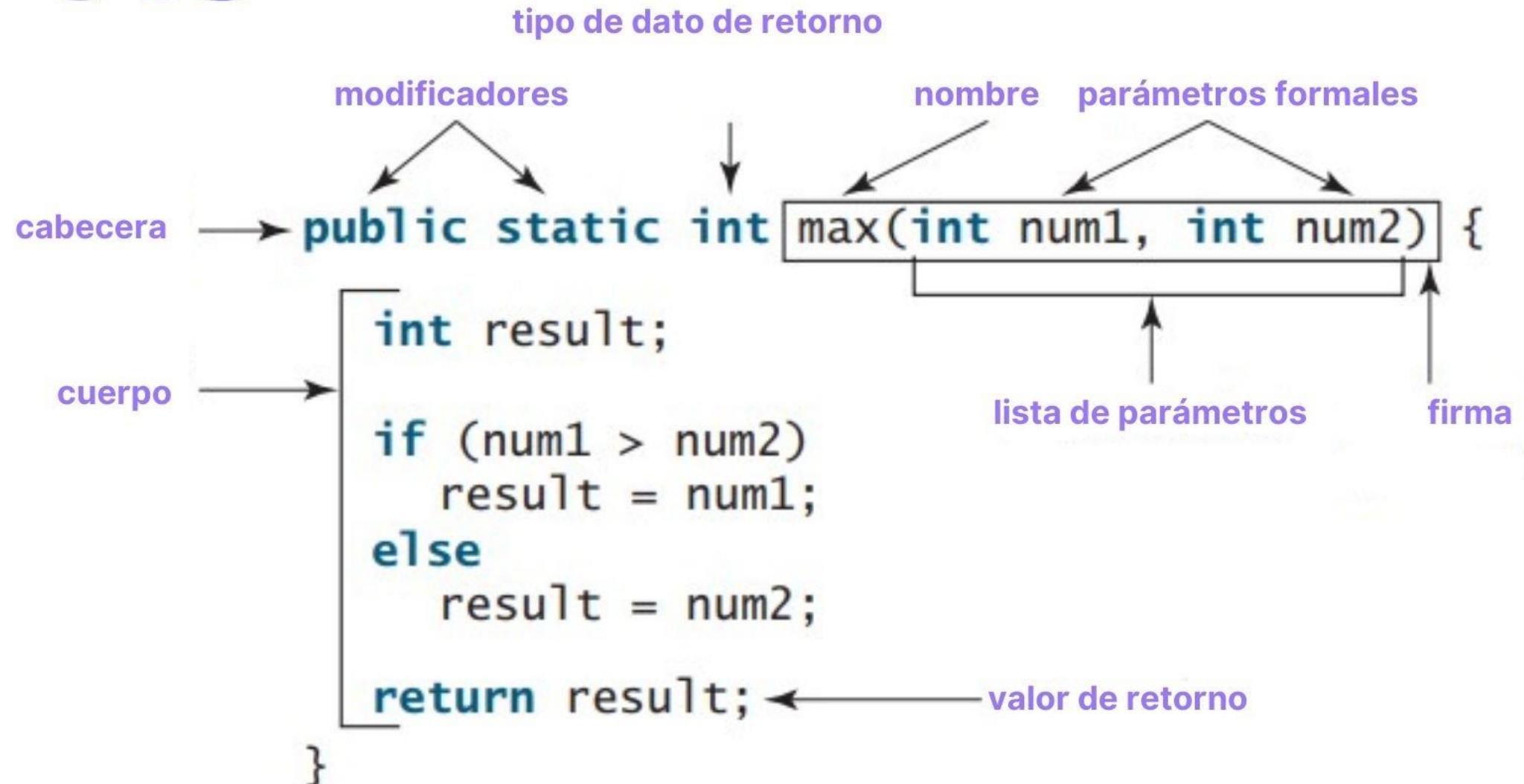
### Que define el encabezado

Especifica los modificadores, el tipo de valor de retorno, el nombre del método y sus parámetros.

Si un método no devuelve ningún valor se denomina método **void**.



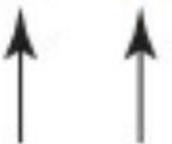
# Definición de un Método





# Invocación de un Método

```
int z = max(x, y);
```



parámetros actuales (argumentos)

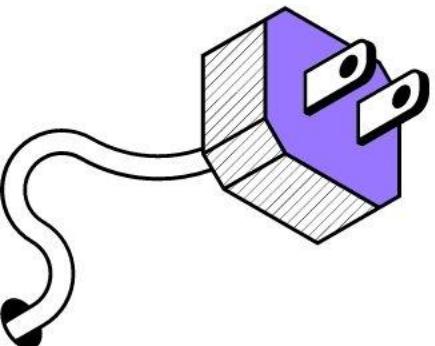


## variables y métodos estáticos

- Una variable estática es compartida por todos los objetos de la clase. Un método estático no puede acceder a los miembros de instancia de la clase (sus atributos).
- Las variables estáticas almacenan valores para las variables en una ubicación de memoria común..
- Para utilizar los elementos de un paquete es necesario importar este en el módulo de código en curso, usando para ello la sentencia

## Como defino una variable o método estático

- Para el caso de una variable, basta con colocar la palabra reservada **static** en su declaración.
- Para el caso de un método, basta con colocar la palabra reservada **static** en su cabecera.





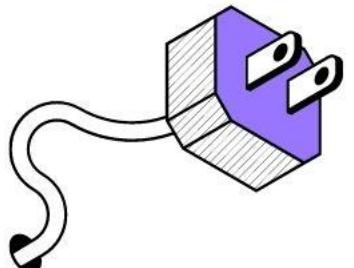
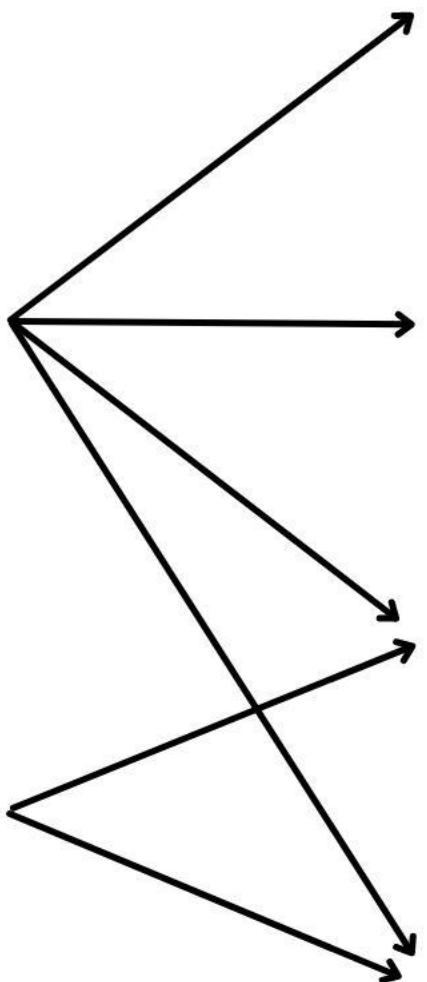
Un método de instancia

Método de instancia

Un método estático

Método estático

Un campo de datos estático





# Sobrecarga de Métodos

Conceptos clave



## En que consiste la sobrecarga de un método

La sobrecarga de métodos le permite definir los métodos con el mismo nombre siempre que sus firmas sean diferentes.

```
/*Retorna el maximo de dos valores enteros*/
public static int max(int num1, int num2) {
    if (num1 > num2)
        return num1;
    else
        return num2;
}

/*Retorna el maximo de dos valores double*/
public static double max(double num1, double num2) {
    if (num1 > num2)
        return num1;
    else
        return num2;
}

/*Retorna el maximo de tres valores double*/
public static double max(double num1, double num2, double num3) {
    return max(max(num1, num2), num3);
}
```



# La encapsulación

## Como funciona



- Hacer que los campos de datos sean privados protege los datos y hace que la clase sea fácil de mantener.
- En primer lugar, los datos pueden ser **manipulados**.
- En segundo lugar, la clase se vuelve difícil de mantener y vulnerable a errores.
- Para evitar modificaciones directas de los campos de datos, debe declarar los campos de datos como privados, utilizando el modificador privado. Esto se conoce como encapsulación de campos de datos.



# Los constructores

## Como funciona



- Se invoca un constructor para crear un objeto usando el operador new.
- Los constructores son un tipo especial de método.
- Un constructor debe tener el mismo nombre que la propia clase.
- Los constructores no tienen un tipo de retorno, ni siquiera **void**.
- Los constructores se invocan mediante el operador **new** cuando se crea un objeto. Los constructores juegan el papel de inicializar a los objetos.



```
public class CircleWithPrivateDataFields {  
    private double radius = 1;  
  
    private static int numberofObjects = 0;  
  
    public CircleWithPrivateDataFields() {  
        numberofObjects++;  
    }  
  
    public CircleWithPrivateDataFields(double newRadius) {  
        radius = newRadius;  
        numberofObjects++;  
    }  
  
    public double getRadius() {  
        return radius;  
    }  
  
    public void setRadius(double newRadius) {  
        radius = (newRadius >= 0) ? newRadius : 0;  
    }  
  
    public static int getNumberofObjects() {  
        return numberofObjects;  
    }  
  
    public double getArea() {  
        return radius * radius * Math.PI;  
    }  
}
```



# La encapsulación

## Como funciona



- Hacer que los campos de datos sean privados protege los datos y hace que la clase sea fácil de mantener.
- En primer lugar, los datos pueden ser **manipulados**.
- En segundo lugar, la clase se vuelve difícil de mantener y vulnerable a errores.
- Para evitar modificaciones directas de los campos de datos, debe declarar los campos de datos como privados, utilizando el modificador privado. Esto se conoce como encapsulación de campos de datos.



# Los constructores

## Como funciona



- Se invoca un constructor para crear un objeto usando el operador new.
- Los constructores son un tipo especial de método.
- Un constructor debe tener el mismo nombre que la propia clase.
- Los constructores no tienen un tipo de retorno, ni siquiera **void**.
- Los constructores se invocan mediante el operador **new** cuando se crea un objeto. Los constructores juegan el papel de inicializar a los objetos.



```
public class CircleWithPrivateDataFields {  
    private double radius = 1;  
  
    private static int numberofObjects = 0;  
  
    public CircleWithPrivateDataFields() {  
        numberofObjects++;  
    }  
  
    public CircleWithPrivateDataFields(double newRadius) {  
        radius = newRadius;  
        numberofObjects++;  
    }  
  
    public double getRadius() {  
        return radius;  
    }  
  
    public void setRadius(double newRadius) {  
        radius = (newRadius >= 0) ? newRadius : 0;  
    }  
  
    public static int getNumberofObjects() {  
        return numberofObjects;  
    }  
  
    public double getArea() {  
        return radius * radius * Math.PI;  
    }  
}
```



# Grandes aplicaciones hechas enteramente con Java

Twitter



Una de las redes sociales más famosas y utilizadas del mundo es Twitter. Esta red social originalmente fue construida en Ruby; sin embargo, con el paso del tiempo surgieron ciertas deficiencias ante la creciente demanda de usuarios.



# Grandes aplicaciones hechas enteramente con Java

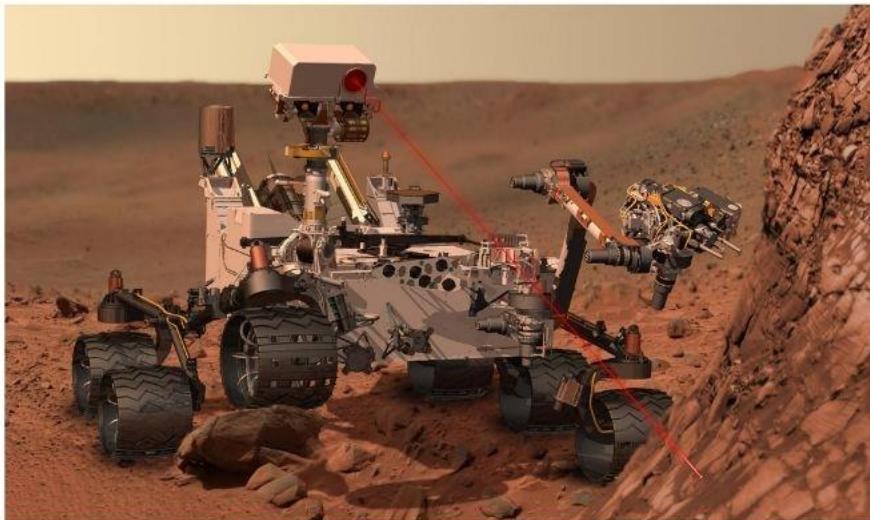
# NETFLIX

Su plataforma originalmente estuvo escrita en Java, aunque con el paso de los años mediante constante investigación y desarrollo, se han ido implementando otros lenguajes y APIs para fortalecer su presencia en el mercado.



# Grandes aplicaciones hechas enteramente con Java

Rover de Marte

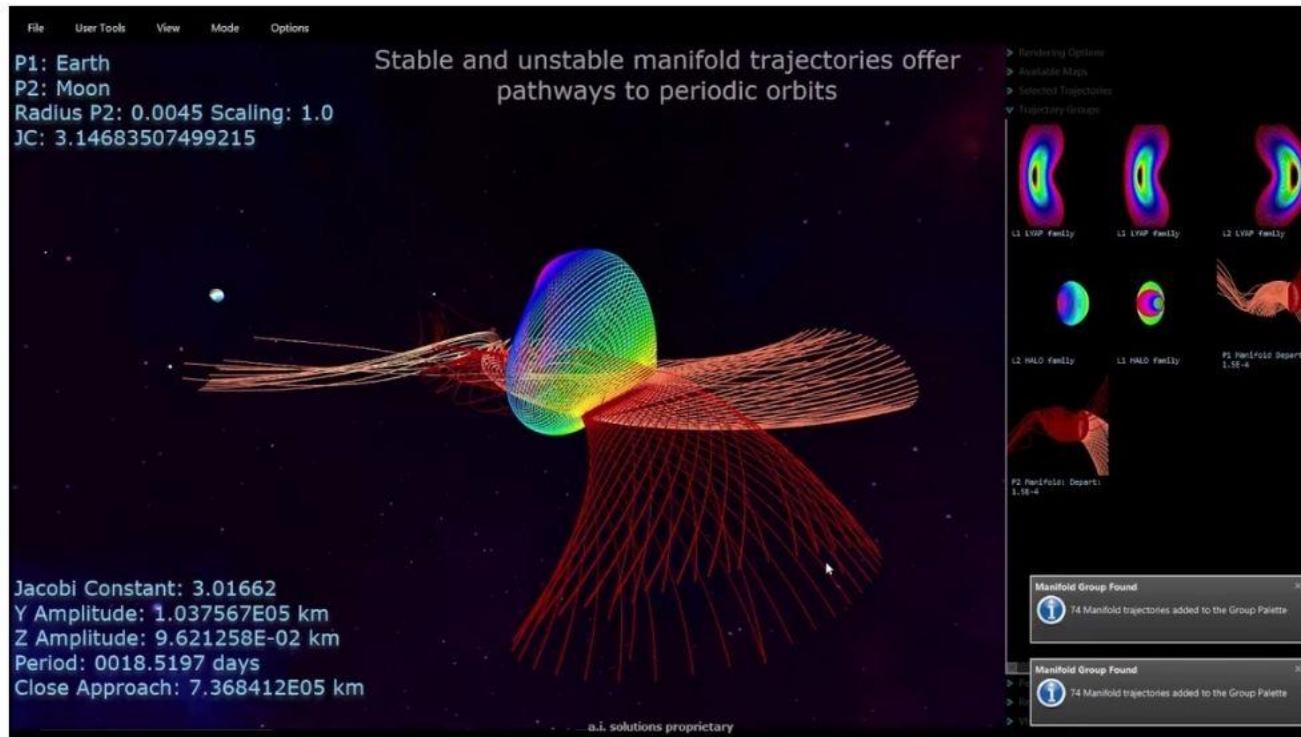


una aplicación utilizada para controlar el Mars Rover el vehículo que se desplaza por la superficie de Marte. Los científicos de la NASA utilizaron durante tres meses el , basado en Java, para controlar el vehículo.



# Grandes aplicaciones hechas enteramente con Java

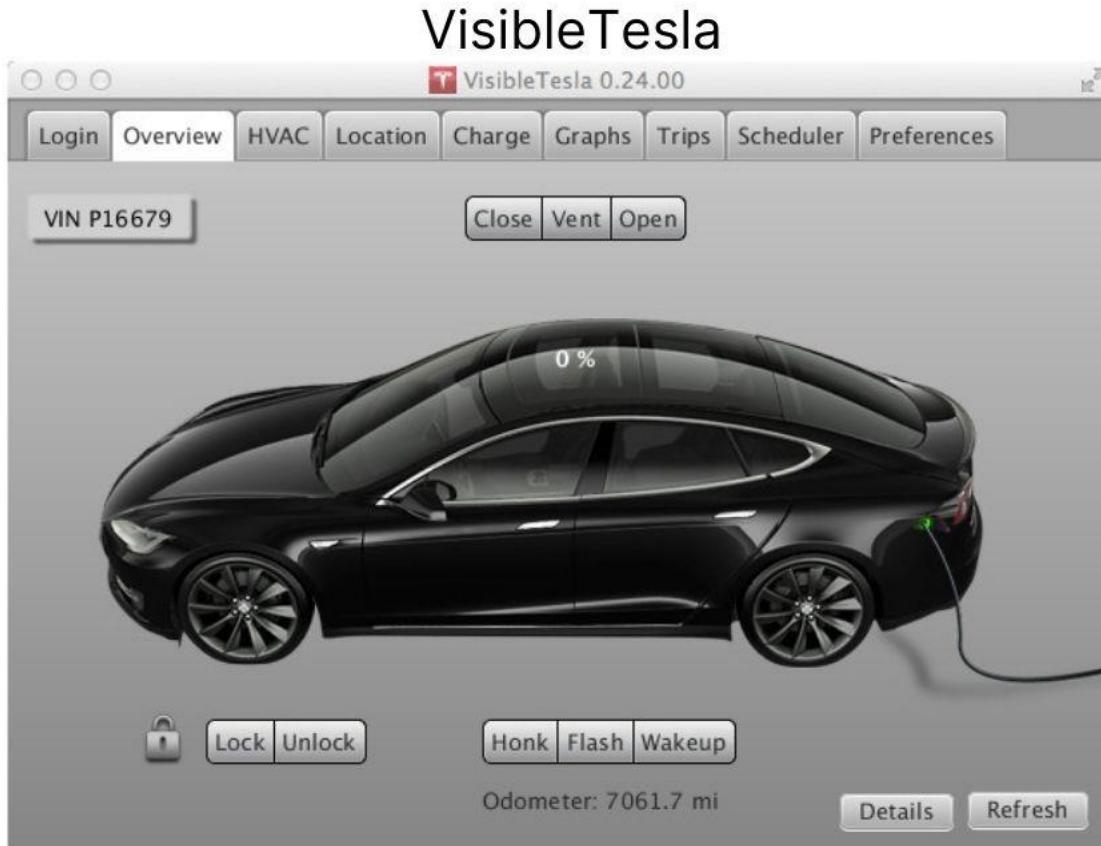
JavaFX Deep Space Trajectory Explorer



Es una herramienta programada en Java que permite generar vistas y modelos multidimensionales para cualquier sistema o asteroide y filtrar millones de puntos en una búsqueda visual densa.



# Grandes aplicaciones hechas enteramente con Java

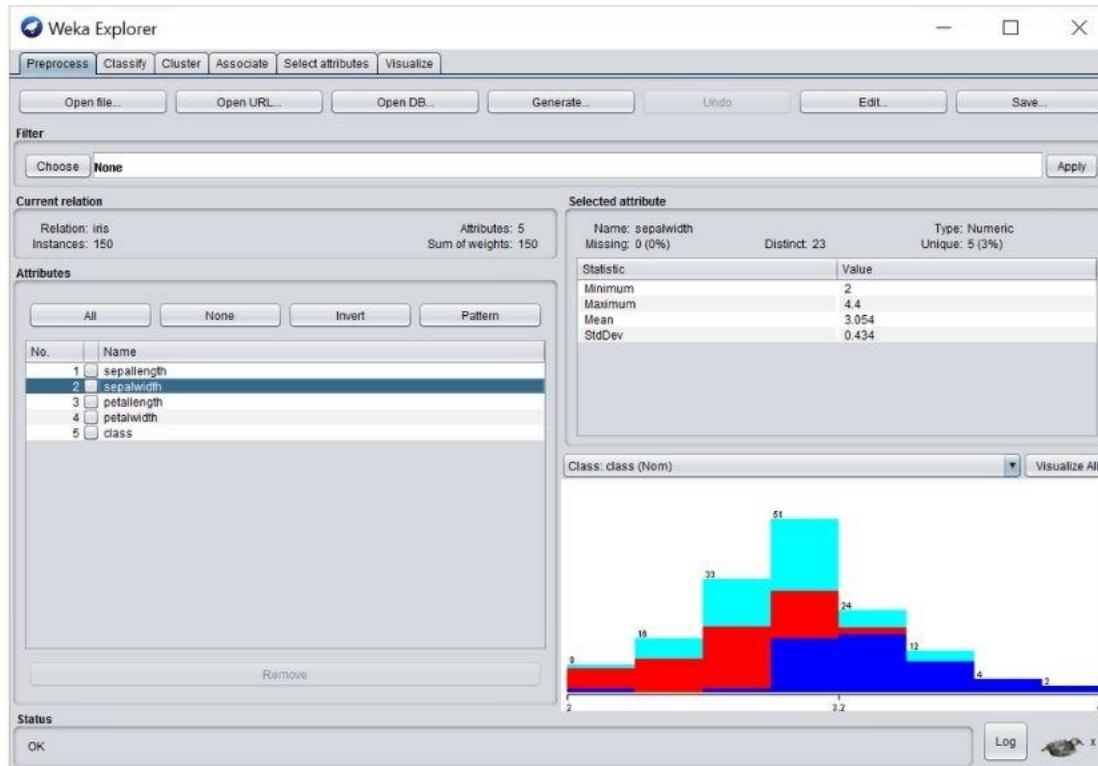


Es una aplicación creada en 2013 para monitorizar y controlar el vehículo Tesla Model S. El código fuente de la aplicación, basada en Java, se encuentra disponible en GitHub.



# Grandes aplicaciones hechas enteramente con Java

weka



Weka es una plataforma de software para el aprendizaje automático y la minería de datos escrito en Java y desarrollado en la Universidad de Waikato.



# Grandes aplicaciones hechas enteramente con Java

Android

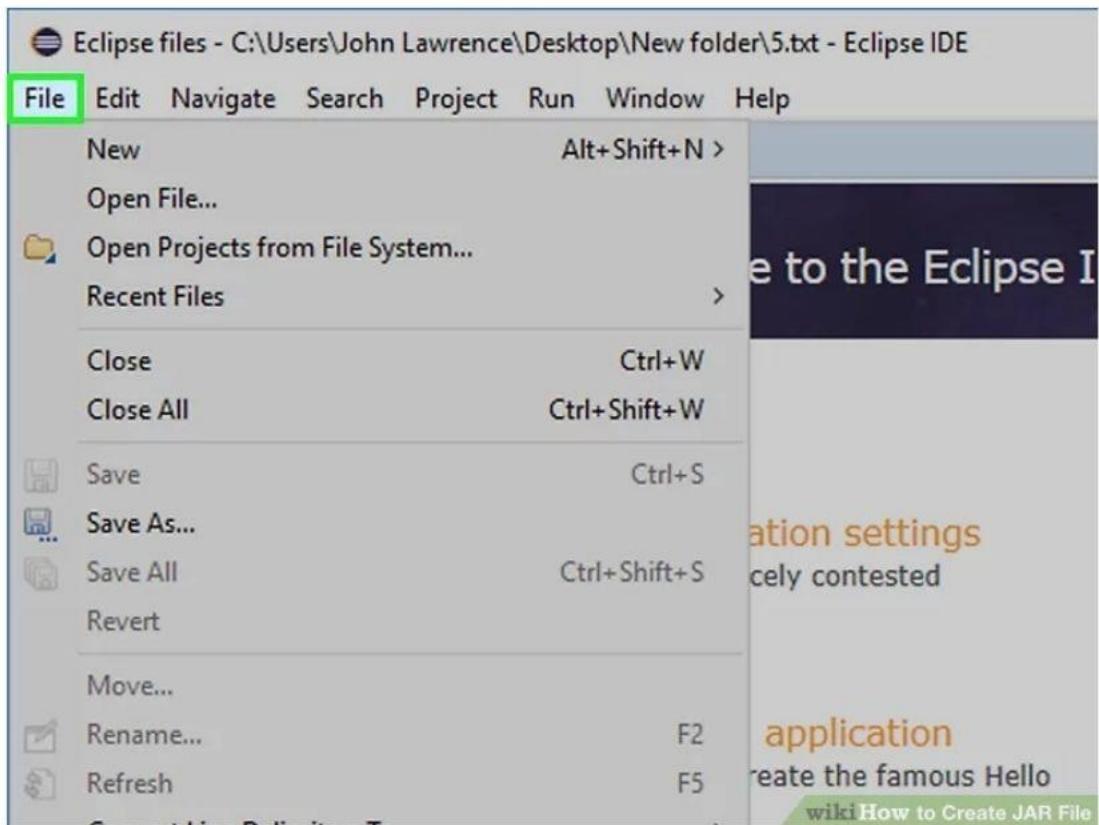


El idioma oficial para el desarrollo de Android es Java. Gran parte de Android está escrito en Java y sus API están diseñadas para ser llamadas principalmente desde Java.



# Creacion de archivos JAR

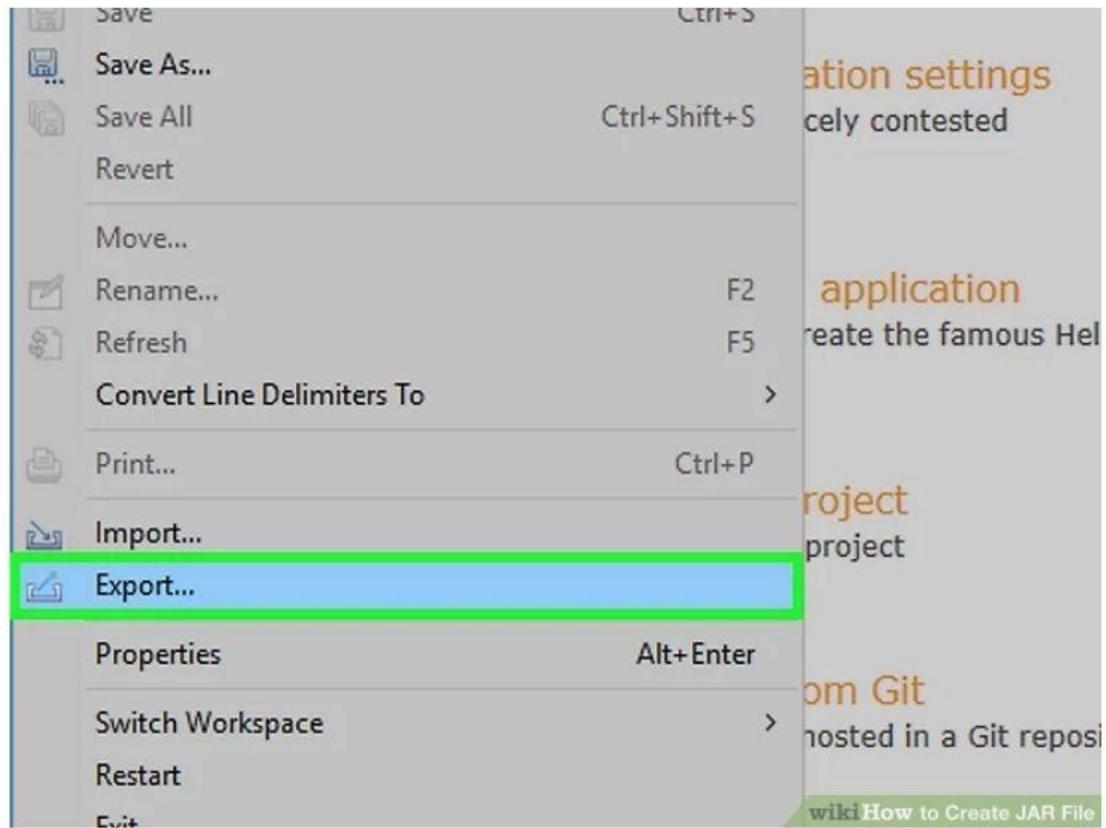
1





# Creacion de archivos JAR

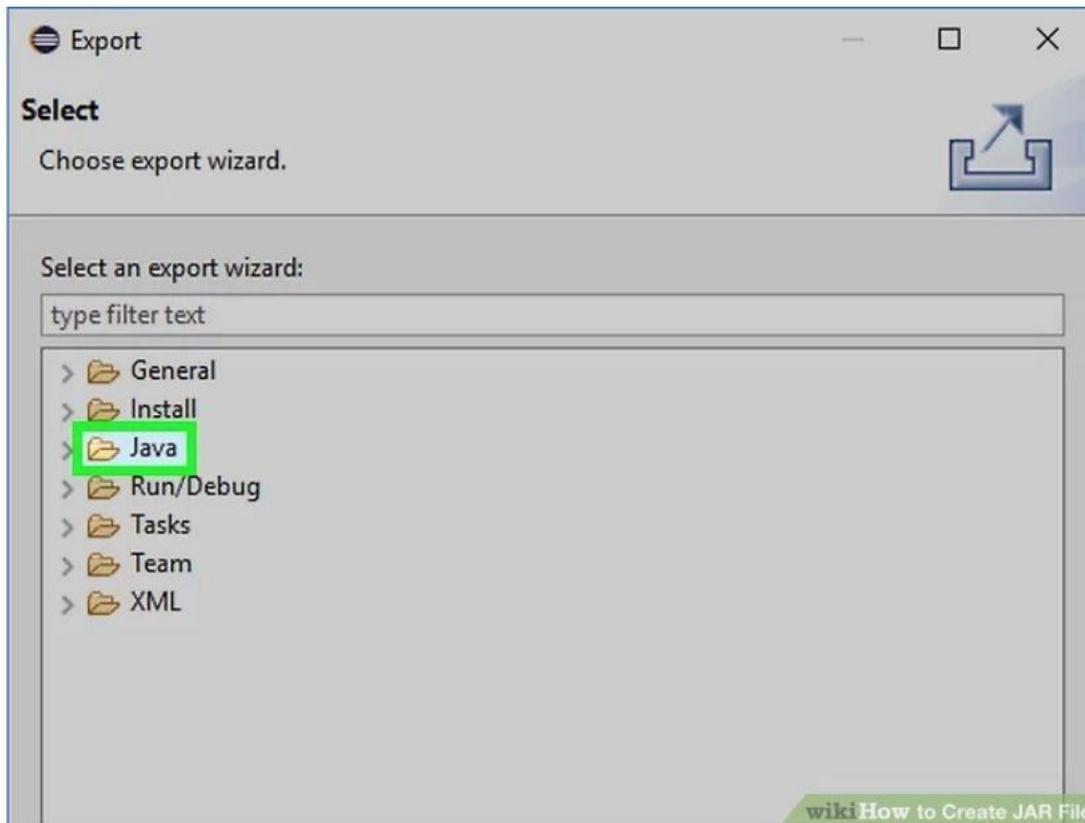
2





# Creacion de archivos JAR

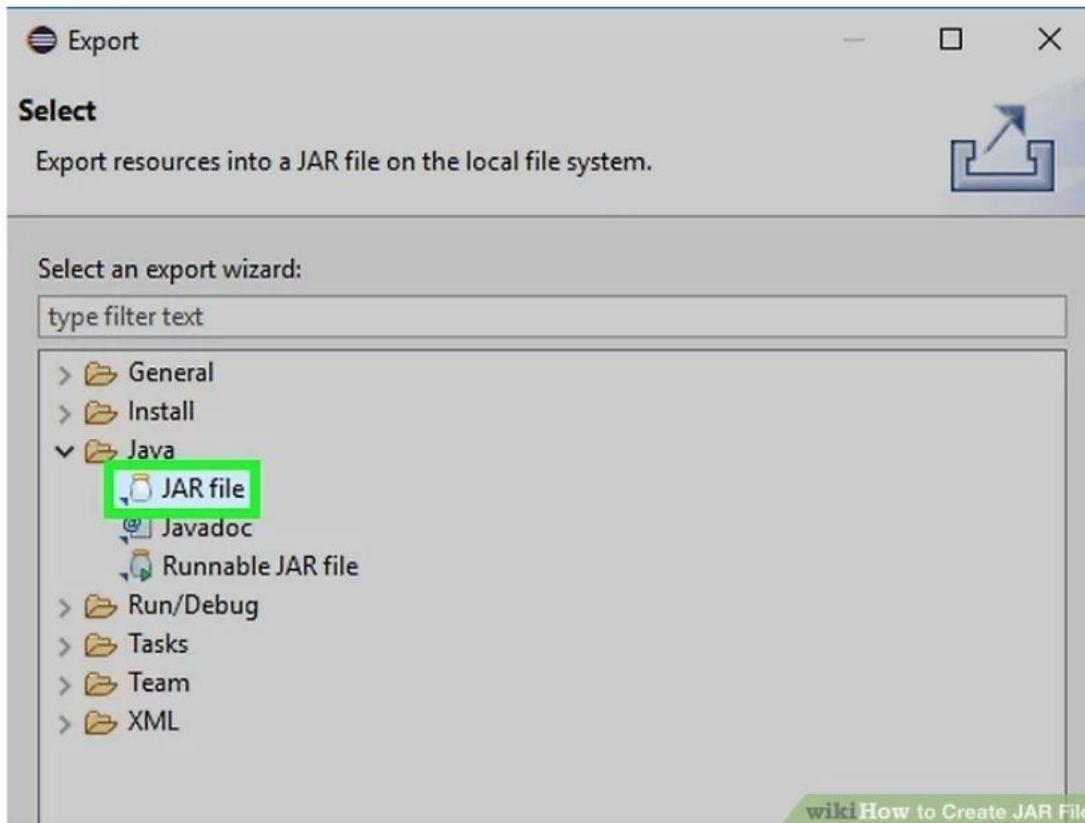
3





# Creacion de archivos JAR

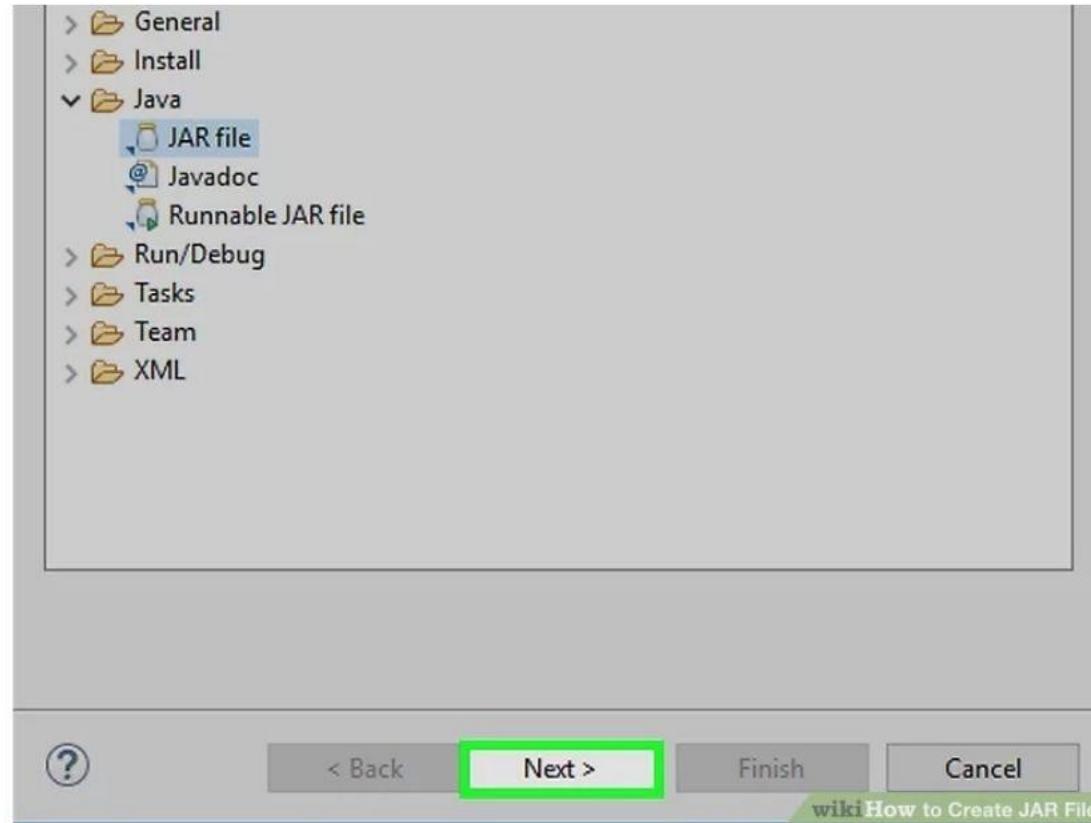
4





# Creacion de archivos JAR

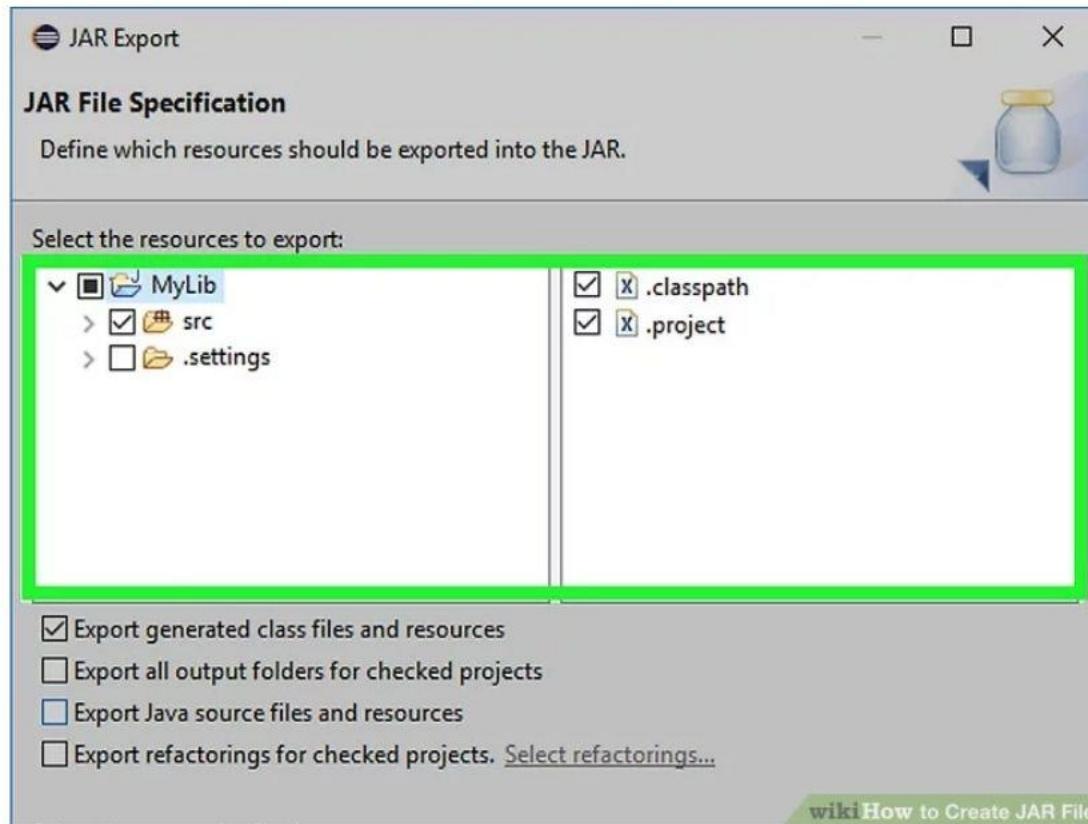
5





# Creacion de archivos JAR

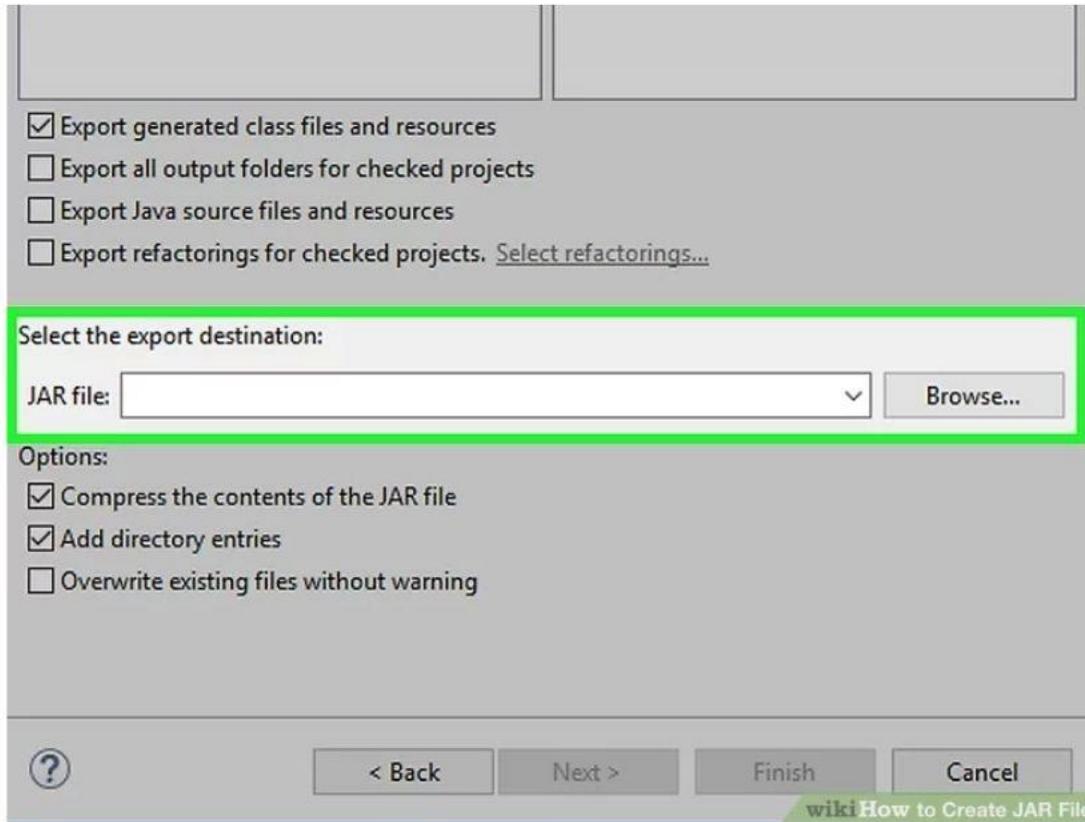
6





# Creacion de archivos JAR

7



wiki [How to Create JAR File](#)



# Creacion de archivos JAR

8

