



DIPLOMADO VIRTUAL EN  
**PROGRAMACIÓN EN JAVA**  
*Guía didáctica 1: Fundamentos*



Formación Virtual

.....educación sin límites



## Competencia específica

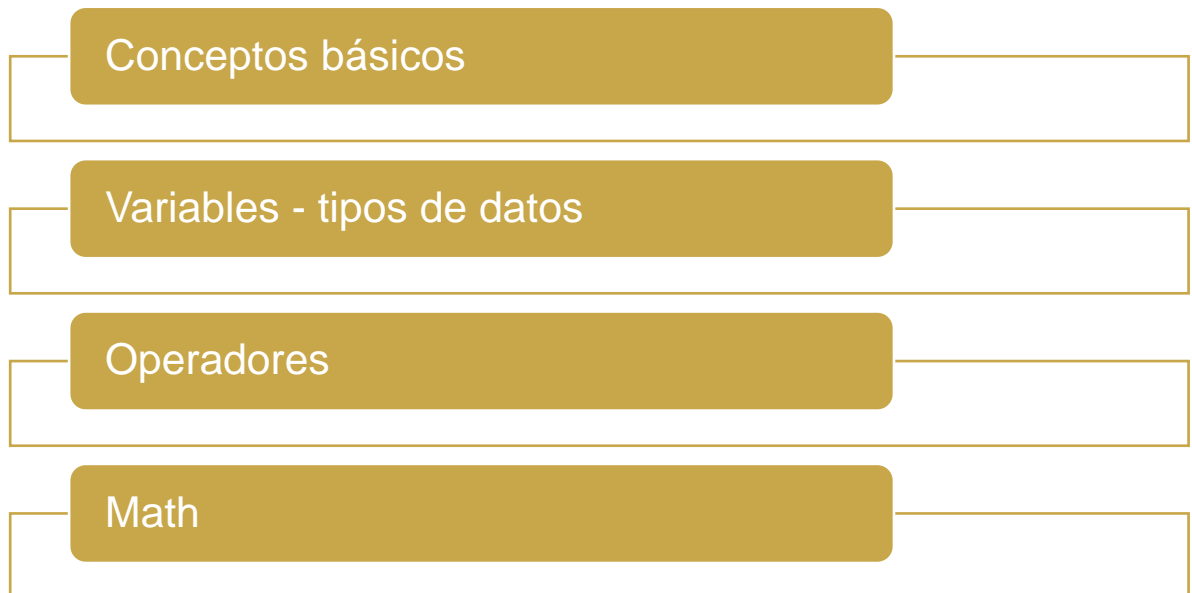
Se espera que con los temas abordados en la guía didáctica del módulo 1: Fundamentos, el estudiante logre la siguiente competencia específica:

- Conocer los conceptos básicos y aplicaciones del lenguaje Java para la programación en cuanto a variables, operadores, tipos de datos, clases de Java, entre otros.



## Contenidos temáticos

Los contenidos temáticos para desarrollar en la guía didáctica del módulo 1: Fundamentos son:



**Ilustración 1: caracterización de la guía didáctica.**

Fuente: autor.

## Tema 1: Conceptos básicos

### ¿Qué es programación?

La programación es «el proceso por medio del cual se diseña, codifica, limpia y protege el código fuente de programas computacionales». (Netec, 2023). A través de la programación se dictan los pasos que debe seguirse para la creación del código fuente de programas informáticos. De acuerdo con ellos, el código se escribe, se prueba y se perfecciona.

El objetivo de la programación es crear un *software* y posteriormente ejecutarlo de manera directa por el *hardware* de la computadora, o a través de otro programa.

La programación se guía por una serie de reglas y un conjunto pequeño de órdenes, instrucciones y expresiones que tienden a parecerse a una lengua natural acotada.

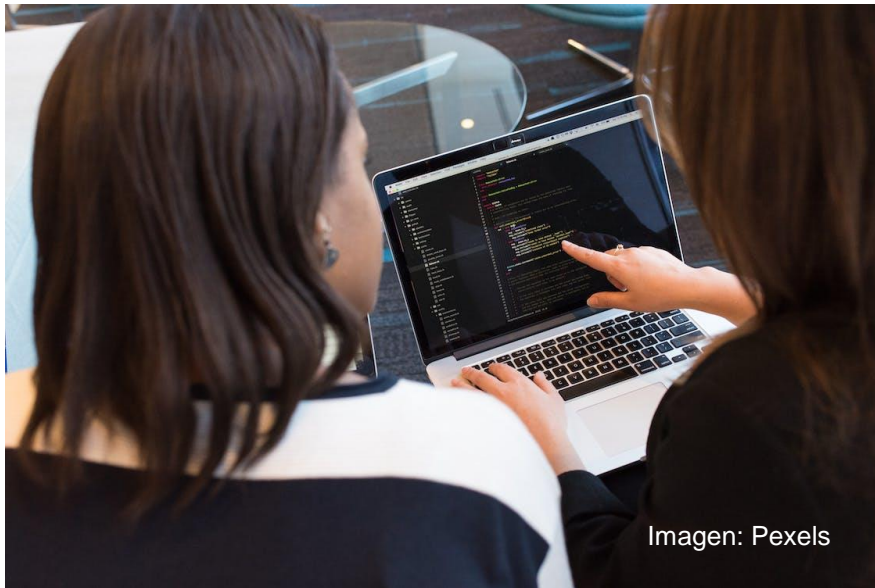


Imagen: Pexels

### ¿Qué es un lenguaje de programación?

1. El lenguaje de programación se define como todas aquellas reglas o normas, símbolos y palabras particulares empleadas para la creación de un programa.

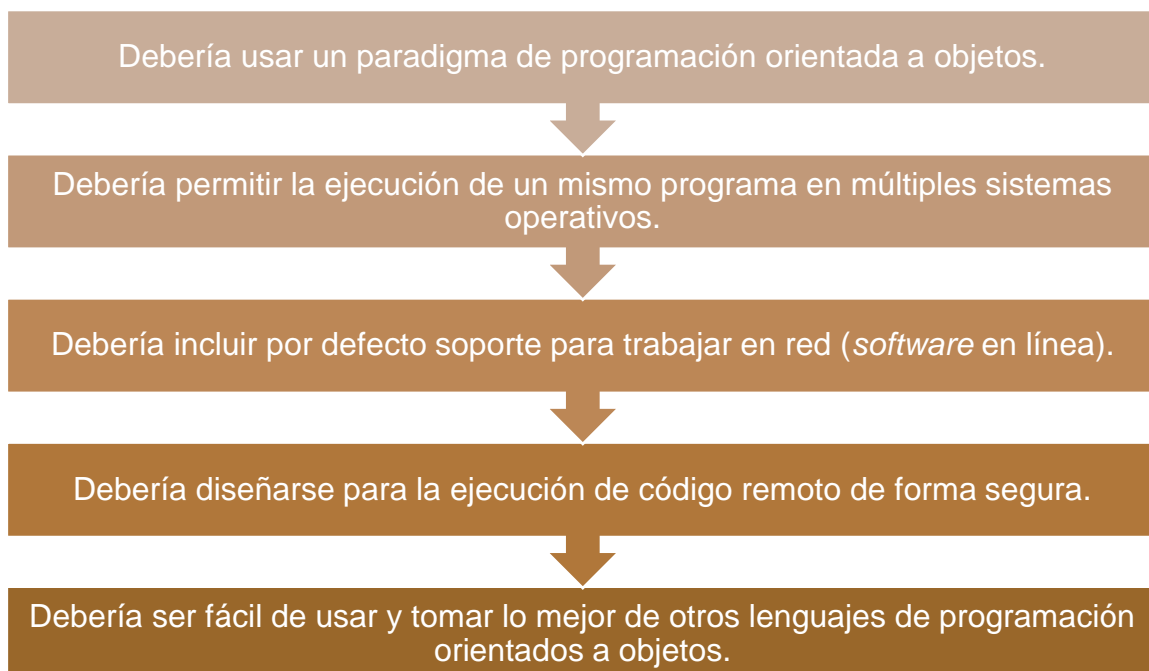
2. El lenguaje de programación es un lenguaje formal que especifica una serie de instrucciones para que una computadora produzca diversas clases de datos.

Los lenguajes de programación pueden usarse para crear programas que pongan en práctica algoritmos específicos que controlan el comportamiento físico y lógico de una computadora. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

### ¿Qué es Java?

Java es un lenguaje de programación creado por Sun Microsystems, en 1995, para el entorno de computación de mismo nombre. Actualmente el dueño de Java es Oracle (en 2009 Oracle compró Sun Microsystems) (Castro, 2019).

Java se creó para acogerse a una filosofía de cinco objetivos en todo el proceso de su creación e implementación:



### **Ilustración 2: objetivos de la creación e implementación de Java.**

Fuente: autor, a partir de Wikipedia (2023b).

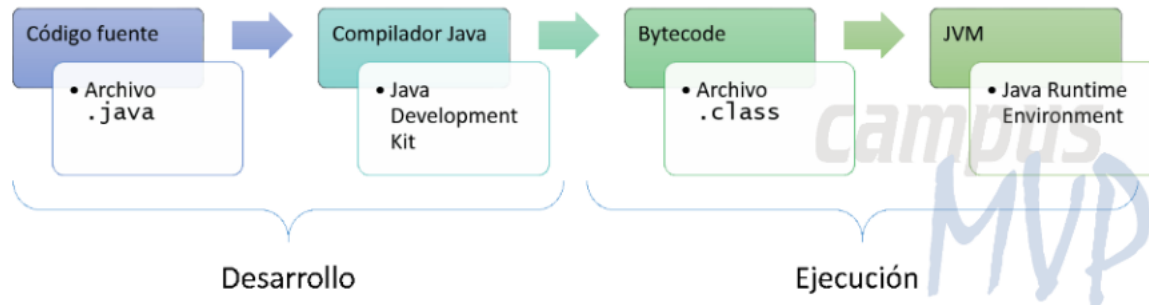


Como la mayoría de lenguajes de programación, Java se utiliza para crear aplicaciones y procesos que funcionen en multitud de dispositivos (Wikipedia, 2023c).

Para su operatividad y correcto funcionamiento, Java necesita una serie de herramientas que permiten esto, dentro del *stack* de herramienta se encuentran principalmente:

**JRE (Java runtime environment):** su objetivo es aportar el entorno necesario para ejecutar una aplicación Java.

**JDK (Java development kit):** es el paquete de herramientas precisas para llevar a cabo el desarrollo de dicha aplicación.



**Ilustración 3: desarrollo y ejecución de un programa en Java.**

Fuente: Campus MVP (2018).

### Instalación de Java y Eclipse

Para el desarrollo idóneo del diplomado, debemos contar con todas las herramientas necesarias y su material, por esto lo primero que se debe hacer es descargar el JDK de Java en el siguiente enlace:

<https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Factores para tener en cuenta:

- La versión del JDK.
- El sistema operativo.
- Tipo de sistema (x32 (x84) – x64).

Enlace de apoyo: <https://www.youtube.com/watch?v=j8ngvGNFXKA>

## Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- [Java Developer Newsletter](#): From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- [Java Developer Day](#) hands-on workshops (free) and other events
- [Java Magazine](#)

JDK 8u191 [checksum](#)

JDK 8u192 [checksum](#)

**Java SE Development Kit 8u191**

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

☐ Accept License Agreement
 ☒ Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.97 MB	<a href="#">jdk-8u191-linux-arm32-vfp-hflt.tar.gz</a>
Linux ARM 64 Hard Float ABI	69.92 MB	<a href="#">jdk-8u191-linux-arm64-vfp-hflt.tar.gz</a>
Linux x86	170.89 MB	<a href="#">jdk-8u191-linux-i586.rpm</a>
Linux x86	185.69 MB	<a href="#">jdk-8u191-linux-i586.tar.gz</a>
Linux x64	167.99 MB	<a href="#">jdk-8u191-linux-x64.rpm</a>
Linux x64	182.87 MB	<a href="#">jdk-8u191-linux-x64.tar.gz</a>
Mac OS X x64	245.92 MB	<a href="#">jdk-8u191-macosx-x64.dmg</a>
Solaris SPARC 64-bit (SVR4 package)	133.04 MB	<a href="#">jdk-8u191-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	94.28 MB	<a href="#">jdk-8u191-solaris-sparcv9.tar.gz</a>
Solaris x64 (SVR4 package)	134.04 MB	<a href="#">jdk-8u191-solaris-x64.tar.Z</a>
Solaris x64	92.13 MB	<a href="#">jdk-8u191-solaris-x64.tar.gz</a>
Windows x86	197.34 MB	<a href="#">jdk-8u191-windows-i586.exe</a>
Windows x64	207.22 MB	<a href="#">jdk-8u191-windows-x64.exe</a>

### Ilustración 4: descarga de Java.

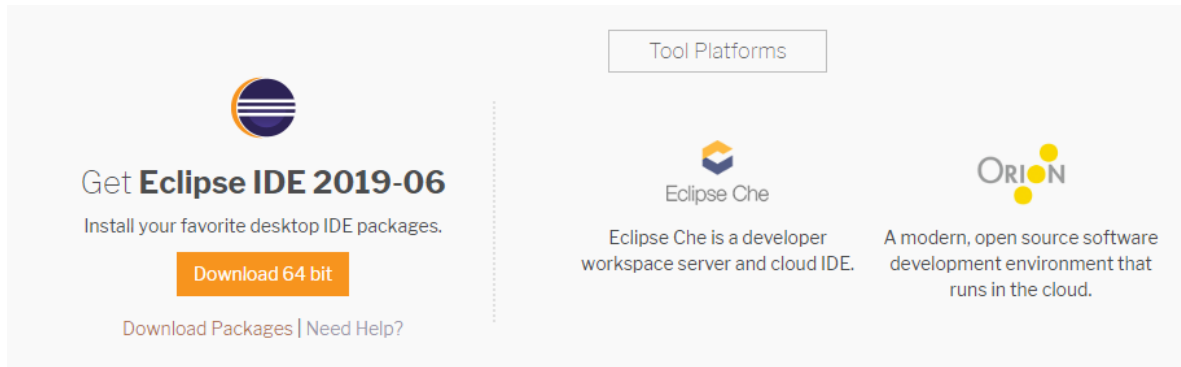
Fuente: autor.

Posterior a esta descarga, se deben seguir los pasos básicos de instalación y aceptar los términos y condiciones *accept license agreement*.

Una vez se ejecuten todos los pasos anteriores, Java y su JDK están correctamente instalados y funcionales. Se prosigue con la instalación del IDE de desarrollo Eclipse (NetBeans, compiladores *online* u otras herramientas también son válidas).

## ¿Qué es Eclipse?

Es una plataforma de *software* compuesta por un conjunto de herramientas de programación de código abierto multiplataforma para codificar. Para proceder con la descargar, ir al siguiente enlace: <https://www.eclipse.org/>



### Ilustración 5: Eclipse.

Fuente: Eclipse.

Es necesario tener en cuenta que Eclipse tiene varias distribuciones y versiones, por lo que se debe prestar atención en cuanto a:

- Versión del Eclipse.
- La distribución: *Eclipse IDE for Java Developers*.
- Tipo de sistema (x32) (x84) – x64.

En este video se ilustra en paso a paso de ambas instalaciones:

<https://www.youtube.com/watch?v=j8ngvGNFXKA>

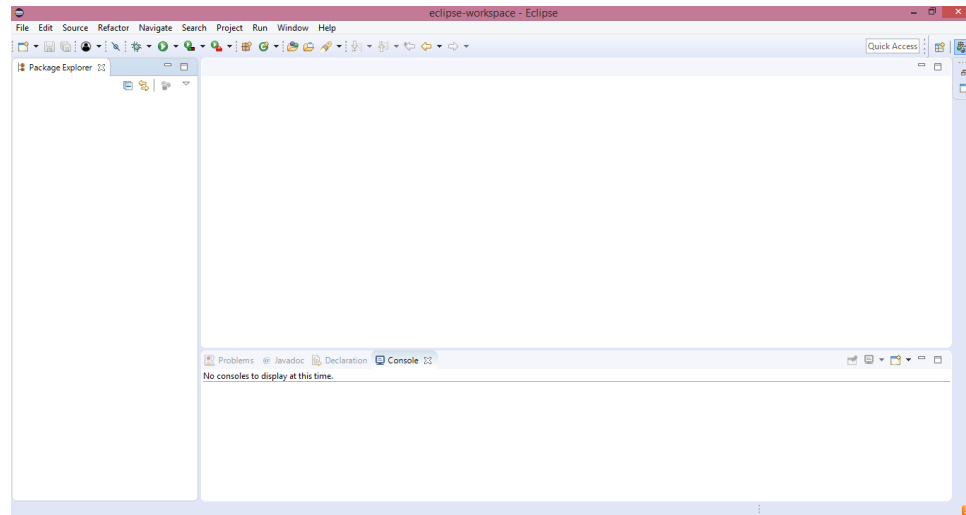
(Si se presentan dificultades con la instalación, utiliza el [compilador online](#) y escríbeme un correo).

### Primer programa: Hola Mundo

Ya que se conoce un poco qué es Java y se tienen instaladas las herramientas necesarias, se puede proceder a realizar un primer programa en el lenguaje de programación Java. En este caso, será el famoso «Hola Mundo». Donde adicionalmente se identificarán las primeras características que brinda el lenguaje de programación a la hora de ser empleado. El siguiente video ilustra el proceso: [Java desde cero con Eclipse \[Parte 1\] \(¡Hola Mundo!\) - YouTube](#)

Se deben realizar los siguientes pasos:

- Abrir el IDE de desarrollo, en este caso Eclipse, recordar que NetBeans y los editores de texto también aplican:

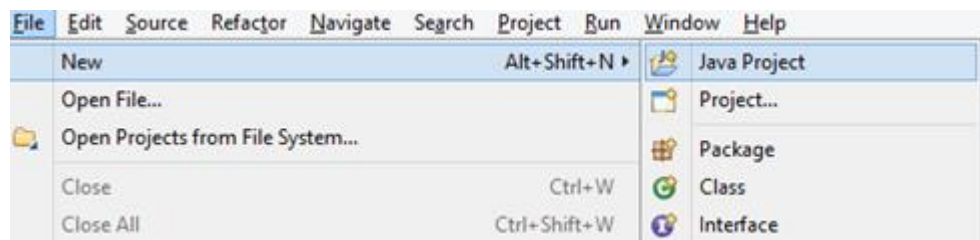


#### Ilustración 6: IDE Eclipse.

Fuente: Eclipse.

- Se debe crear ahora un nuevo proyecto en Java a través de las opciones proporcionadas por Eclipse en el menú:
  - *File:*
  - *New:*
  - *Java Project:*

(En caso de no encontrar la opción de Java Project, ir a **Other** y de ahí a **Java Project**).

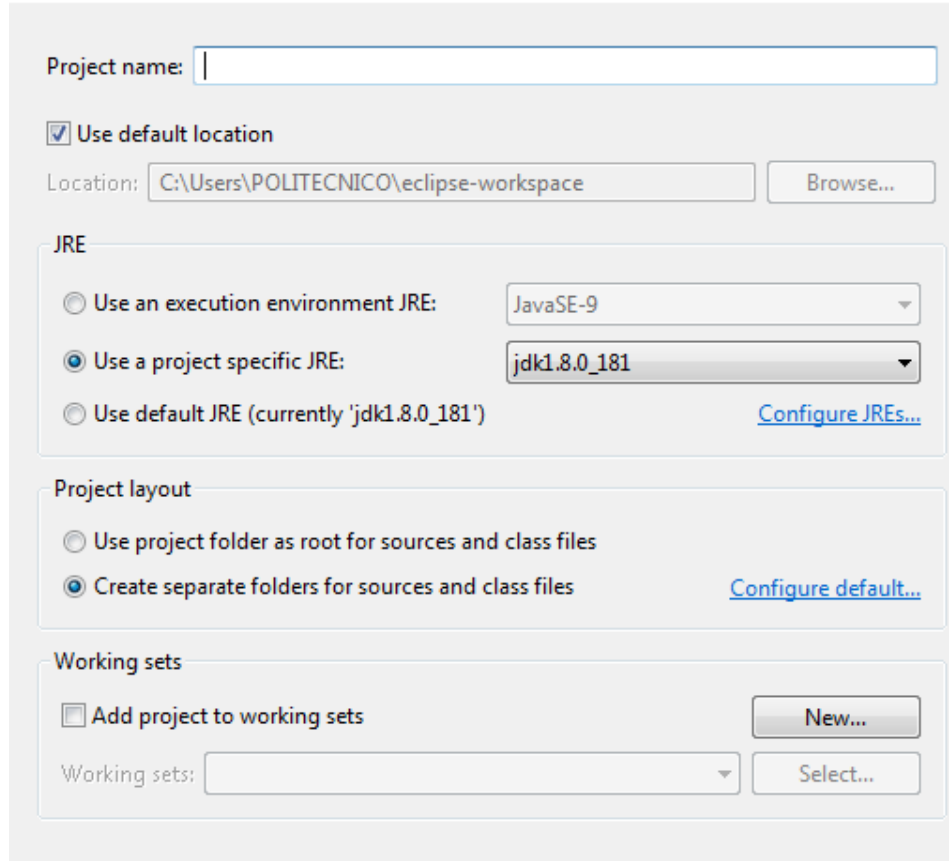


#### Ilustración 7: creación de proyectos en Eclipse.

Fuente: Eclipse.



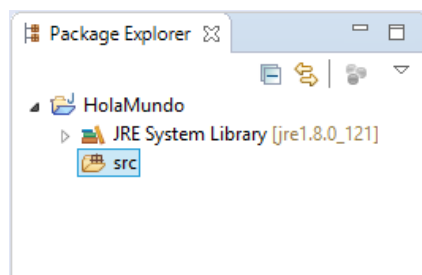
- Ahora se debe configurar el proyecto creado (nombre del proyecto, versión que se va a utilizar, marco del proyecto, entre otras opciones). Únicamente en este caso se establece el nombre del proyecto y la versión que se va a utilizar:



**Ilustración 8: configuración de proyectos en Eclipse.**

Fuente: Eclipse.

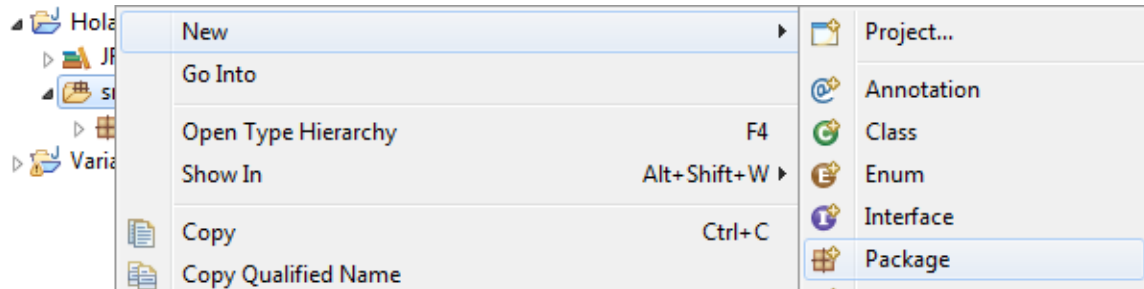
- El resultado será el siguiente:



**Ilustración 9: vista previa del proyecto en Eclipse.**

Fuente: Eclipse.

- Ahora, dentro de la carpeta creada por el IDE **src** (también se conoce como *source*), se debe crear un paquete que se encargará de contener todas las clases de Java:

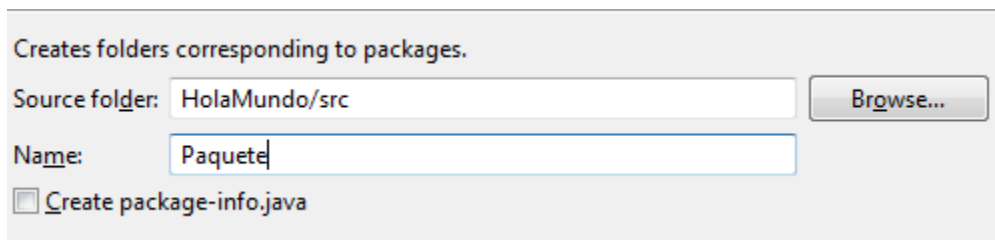


- *Src:*
- *New:*
- *Package:*

#### Ilustración 10: creación de paquetes en Eclipse.

Fuente: Eclipse.

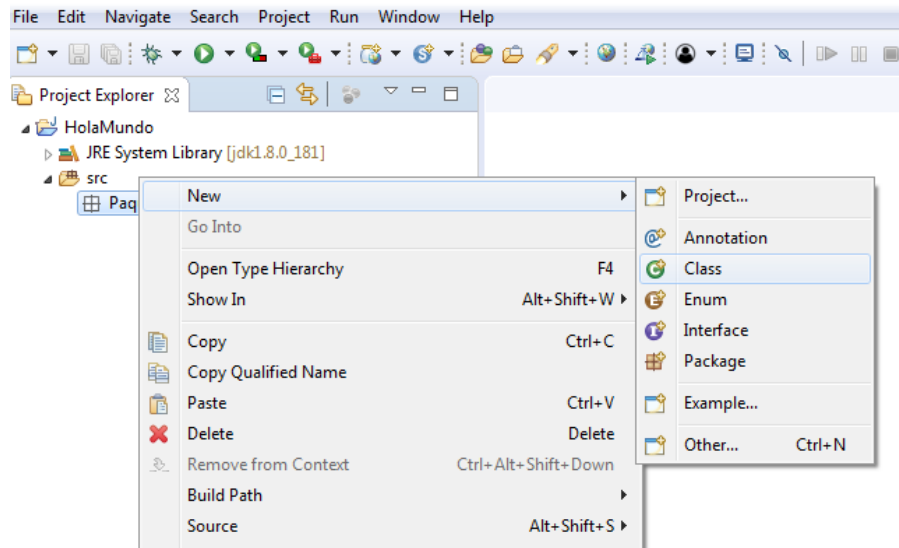
- Entre las configuraciones que se encuentran a la hora de crear un paquete, solo importa determinar el nombre del paquete.



#### Ilustración 11: configuración de paquetes en Eclipse.

Fuente: Eclipse.

- Lo único que falta ahora es crear la clase Java dentro del paquete para codificar el «Hola Mundo».

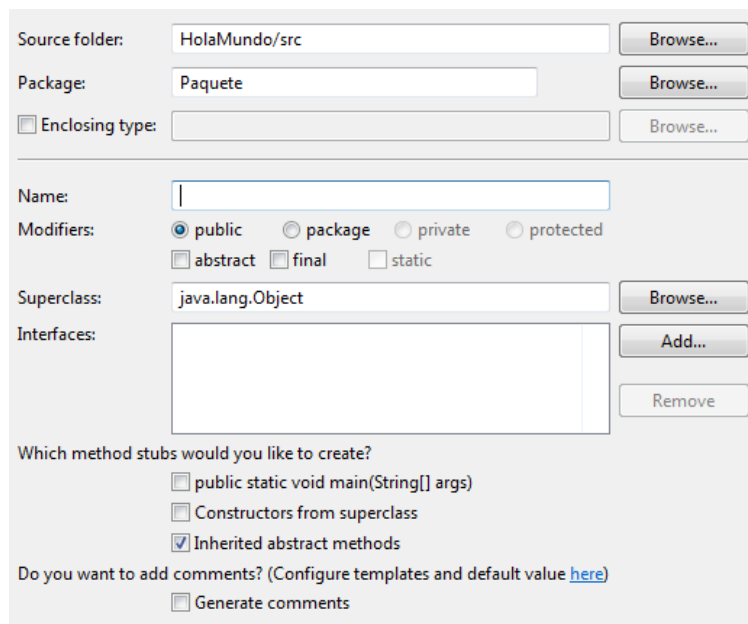


- Paquete:
- *New*:
- *Class*:

#### Ilustración 12: creación de clases en Eclipse.

Fuente: Eclipse.

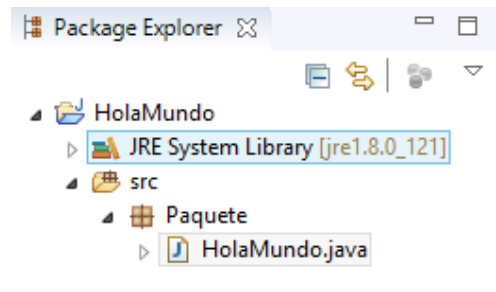
- Ahora se configuran las características de la clase «Hola Mundo» (nombre, paquete, tipo de clase, si va a ser principal o no, entre otros).



#### Ilustración 13: configuración de la clase HolaMundo.java.

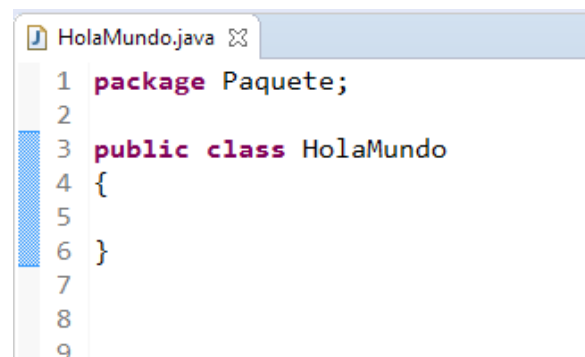
Fuente: Eclipse.

De esa forma se obtiene la siguiente estructura:



**Ilustración 14: estructura de la clase HolaMundo.java dentro del proyecto HolaMundo.**  
Fuente: Eclipse.

Se cuenta con un proyecto que se llama **HolaMundo**, dentro del cual hay un *source package* llamado **Paquete** que contiene una clase **HolaMundo.java**; dicha clase cuenta con la siguiente estructura:



```
1 package Paquete;
2
3 public class HolaMundo
4 {
5
6 }
7
8
9
```

**Ilustración 15: clase HolaMundo.java.**  
Fuente: Eclipse.

Esta es la estructura básica de una clase en Java, para el correcto funcionamiento y el desarrollo del programa «Hola Mundo», se deben codificar las primeras líneas para obtener la siguiente estructura:

```
HolaMundo.java
1 package Paquete;
2
3 public class HolaMundo
4 {
5     public static void main(String args[])
6     {
7         System.out.println("Hola Mundo");
8     }
9 }
```

#### Ilustración 16: estructura básica de una clase en Java.

Fuente: Eclipse.

```
package Paquete;
```

Indica el paquete en el que se encuentra la clase HolaMundo.java.

```
public class HolaMundo {
```

Indica el nombre de la clase y el nivel de acceso de esta.

1. El nombre de la clase debe ser el mismo del archivo. Java.
2. Siempre debe contener la palabra **class**.
3. Debe abrir y cerrar llaves “{}” (dentro de estas va todo el código).


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




Indica un método *public* (público) y *main* (principal) que va a permitir ejecutar el código.

```
System.out.println("Hola Mundo");
```

La instrucción «**System.out.println ()**» permite mostrar en consola (pantalla) el resultado que se indique dentro de los paréntesis **()**, de la siguiente forma, pero antes se debe ejecutar la clase.

El proceso de ejecutar se realiza desde una de las opciones del menú

superior de la clase: , en orden se encuentran las siguientes opciones:

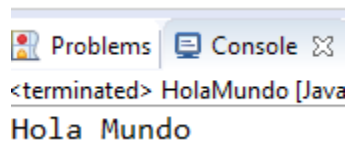
- *Run* (ejecutar) - 
- *Coverage* (cobertura) 
- *Run External Tool* (ejecutar herramienta externa) 



Las tres opciones cumplen la función de ejecutar, lo que cambia entre ellas es la forma de hacerlo.

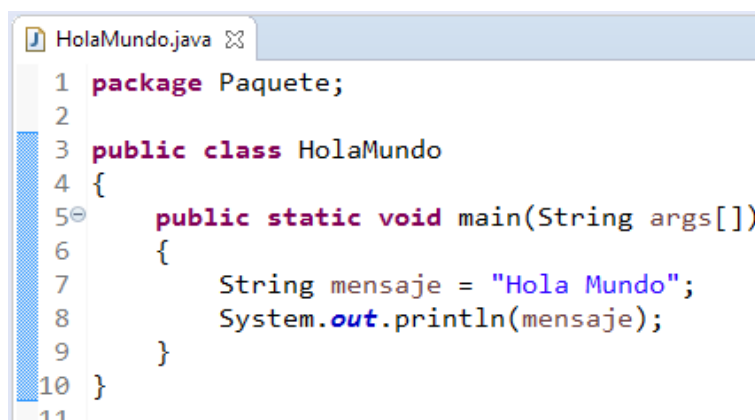
- **Run:** ejecutar y realizar los procesos descritos en la clase.
- **Coverage:** ejecutar, realizar pruebas y consultar cobertura de aplicaciones.
- **Run External Tool:** ejecutar la clase con una herramienta externa.

Finalmente, en la parte inferior de la pantalla, en el apartado de consola se obtendrá el resultado a raíz de la ejecución de la clase. En este caso, el mensaje de «Hola Mundo».



**Ilustración 17:** clase HolaMundo.java ejecutada.  
Fuente: Eclipse.

Otra opción que se puede usar para llevar a cabo el «Hola Mundo» es la siguiente, en la cual el mensaje en este caso «Hola Mundo» se encuentra contenido dentro de una variable y simplemente no se imprime el mensaje, sino la variable, y el resultado será exactamente el mismo.



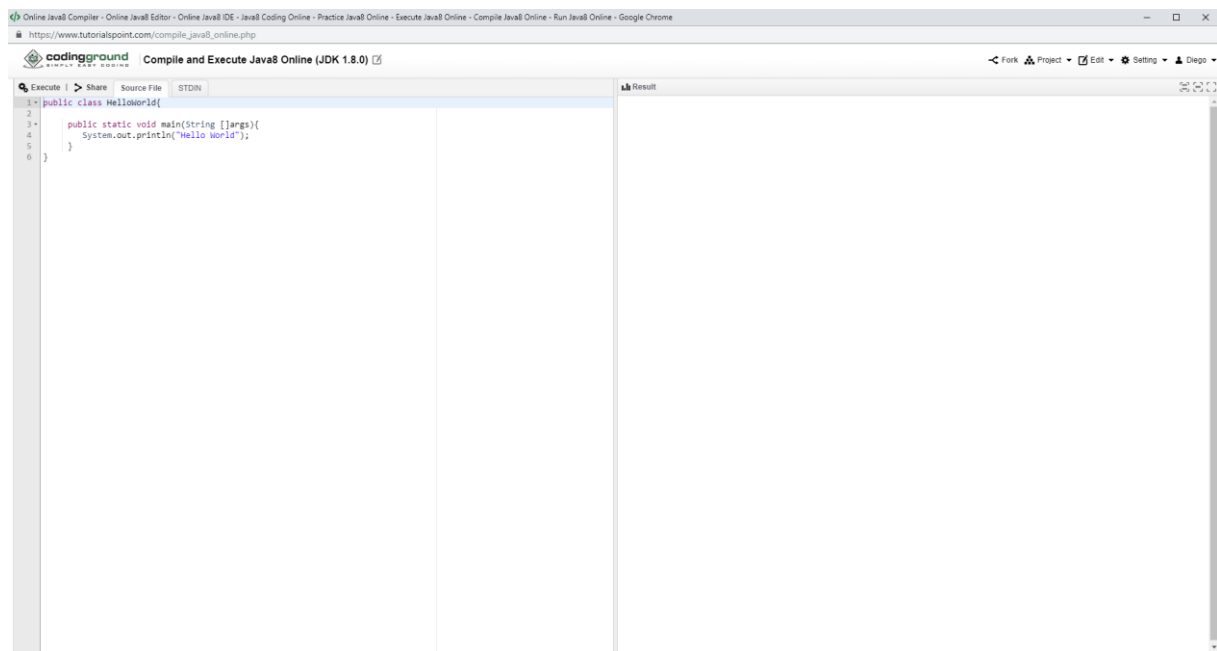
```
1 package Paquete;
2
3 public class HolaMundo
4 {
5     public static void main(String args[])
6     {
7         String mensaje = "Hola Mundo";
8         System.out.println(mensaje);
9     }
10 }
11
```

**Ilustración 18:** clase HolaMundo.java ejecutada manejando el mensaje por variable.  
Fuente: Eclipse.

## Compilador *online*

Existen varias alternativas con las que se puede programar en Java, entre las más utilizadas se encuentran principalmente: NetBeans y Eclipse; ambas, plataformas de escritorio. Aunque son muy prácticas y efectivas, resulta un poco complicado descargar, instalar y configurar dichas herramientas, por lo que existe una alternativa muy simple y fácil de utilizar en caso de tener dificultades con la instalación de Eclipse o NetBeans hasta cierto punto, dado que son cómodas para trabajar con una clase, pero cuando se pretende codificar más de una de estas, se convierten en mejor opción las dos principales plataformas mencionadas al inicio. Veamos:

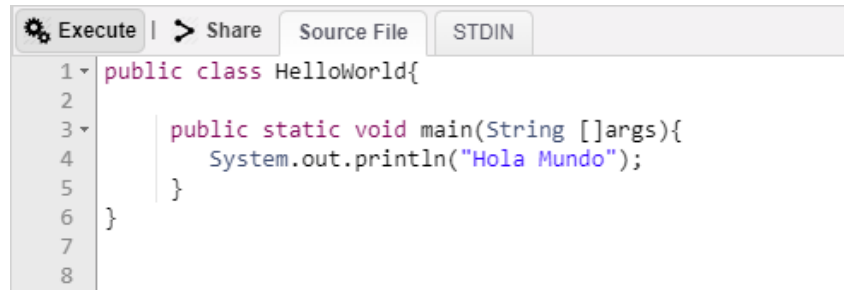
Ingresar al compilador directamente en el siguiente hipervínculo:  
[https://www.tutorialspoint.com/compile\\_java8\\_online.php](https://www.tutorialspoint.com/compile_java8_online.php)



**Ilustración 19: vista general del IDE.**

Fuente: Eclipse.

La plataforma ofrece una zona para codificar a mano izquierda que hace el papel de una clase, concepto que se verá a fondo más adelante:



```
1 public class HelloWorld{
2
3     public static void main(String []args){
4         System.out.println("Hola Mundo");
5     }
6 }
7
8
```

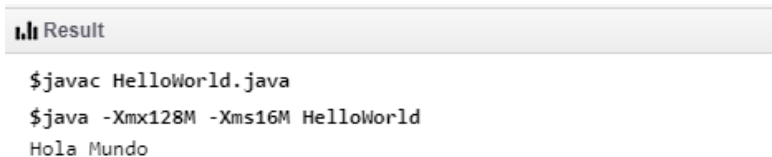
#### Ilustración 20: zona de codificación.

Fuente: Eclipse.

En la parte superior se ubican dos opciones muy importantes:

- **Execute:** permite ejecutar el código escrito en la zona de codificación.
- **Source File:** permite acceder a la zona de codificación.

A mano izquierda se encuentra la zona de ejecución, donde se arrojan los datos de impresión proporcionado por la zona de codificación, por ejemplo, el «Hola Mundo».



```
Result
$javac HelloWorld.java
$java -Xmx128M -Xms16M HelloWorld
Hola Mundo
```

#### Ilustración 21: zona de ejecución.

Fuente: Eclipse.

### Programación estructurada

La programación estructurada es un **paradigma de programación** (Wikipedia, 2023a), orientado a mejorar la claridad, la calidad y el tiempo de desarrollo de un programa de computadora recurriendo únicamente a subrutinas y tres estructuras básicas: secuencia, selección (*if* y *switch*) e iteración (*bucles for* y *while*).

En 1966 surgió una nueva forma de programar que no solamente permitía desarrollar programas fiables y eficientes, sino que además estos estaban escritos de manera que se facilitaba su comprensión en fases de mejora posteriores.

El teorema del programa estructurado, propuesto por Böhm y Jacopini,

demuestra que todo programa puede escribirse utilizando únicamente las tres instrucciones de control siguientes:

- Secuencia.
- Instrucción condicional.
- Iteración (bucle de instrucciones) con condición inicial (Wikipedia, 2023b).

Solo con estas tres estructuras se pueden escribir todos los programas y aplicaciones posibles. Si bien los lenguajes de programación tienen un mayor repertorio de estructuras de control, estas pueden ser construidas mediante las tres básicas citadas.

## Tema 2: Variables - tipos de datos

Todo programa de ordenador pretende ofrecer una funcionalidad determinada para la que, por regla general, necesitará almacenar y manipular información. Dicha información, que son los datos sobre los que se opera, deben almacenarse de manera temporal en la memoria del ordenador. Para almacenar y recuperar fácilmente información en la memoria de un ordenador, los lenguajes de programación ofrecen el concepto de variables, estos son nombres que «apuntan» a una determinada parte de la memoria y que el lenguaje utiliza para escribir y leer en esta de manera controlada.

El acceso a esta información puede mejorarse dependiendo del tipo de información almacenada. Por ejemplo, no es lo mismo tener la necesidad de manejar números que letras o que conjuntos de datos. Asimismo, dentro de estos no es igual tener que almacenar un número entero que uno decimal. Y pese a que al final todo son ceros y unos dentro de la memoria del ordenador, es la forma de interpretarlos lo que marca la diferencia, tanto al almacenarlos como al recuperarlos.

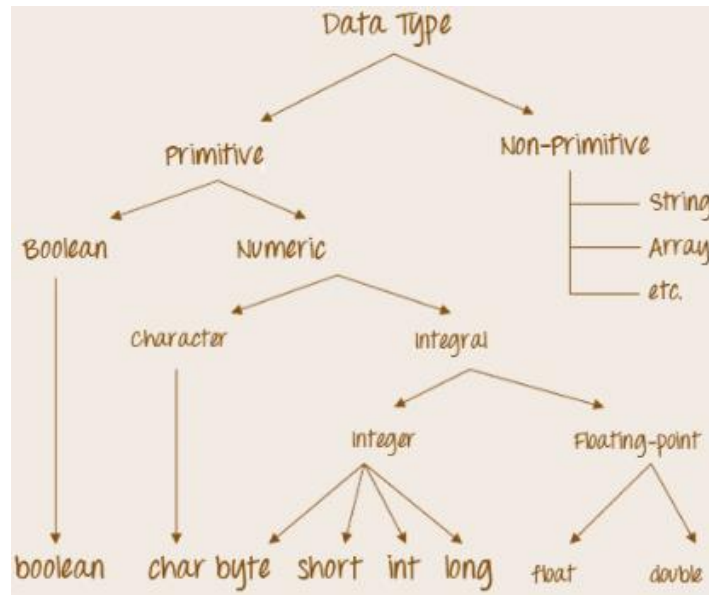
Esta es la razón por la cual los lenguajes de programación cuentan con el concepto de tipos de datos: se trata de distintas maneras de interpretar esos «ceros y unos» en función de ciertas configuraciones que establecen el espacio utilizado, así como la representación aplicada para codificar y decodificar esa información (Campus MVP, 2018).

### **Datos primitivos en Java**

Java cuenta con un pequeño conjunto de tipos de datos primitivos. Se podrían considerar fundamentales, pues la mayor parte de los demás tipos, los tipos estructurados o complejos, son composiciones a partir de estos más básicos. Estos tipos de datos primitivos sirven para gestionar los tipos de información más básicos: números de diversas clases o datos de tipo verdadero/falso (también conocidos como «valores booleanos» o simplemente «booleanos»).



De estos tipos primitivos, que son ocho en total, seis están destinados a facilitar el trabajo con números. Pueden reunirse en dos categorías: **tipos numéricos enteros** y **tipos numéricos en punto flotante**. Los primeros permiten operar exclusivamente con números enteros, sin parte decimal; en tanto, el segundo grupo abarca también números racionales o con parte decimal (Campus MVP, 2018).



**Ilustración 22: tipos de datos en Java.**

Fuente: Guru99 (2023).

## Números enteros

En Java existen cuatro tipos destinados a almacenar números enteros. La única diferencia entre ellos es el número de *bytes* usados para su almacenamiento y, por ende, el rango de valores que es posible representar con ellos. Todos ellos emplean una representación que permite el almacenamiento de números negativos y positivos. El nombre y características de estos tipos son los siguientes:

**Nota:** crear un proyecto de nombre **Variables**, que contenga un paquete de nombre **Clases** y por ultimo una clase llamada **Variables**. Ahí se desarrollará el contenido temático – El texto que se encuentre entre “//” de color verde son comentarios que ayudan a describir lo que se está realizando en el código.

```
//byte: Emplea un sólo byte (8 bits) de almacenamiento.  
//Esto permite almacenar valores entre [-128, 127].  
  
byte numeroByte = 9;  
  
//short: Emplea el doble almacenamiento de (byte).  
//Esto permite almacenar valores entre [-32.768, 32.767].  
  
short numeroShort = 32767;  
  
//int: Emplea un tamaño mayor, 4 bytes (32 bits).  
//Esto permite almacenar valores entre [-2147483648, 2147483647].  
  
int numeroInt = 41825;  
  
//long: Emplea el tamaño mayor de todos los enteros, 8 bytes (64 bits).  
//Esto permite almacenar valores entre [-9223372036854775808, 9223372036854775807].  
  
long numeroLong = 926465464697266565L;
```

### **Ilustración 23: datos y variables primitivas enteras.**

Fuente: Eclipse.

### **Números flotantes**

Los tipos numéricos en punto flotante permiten representar tanto números muy grandes como números muy pequeños, además de números decimales. Java dispone de 2 tipos concretos en esta categoría (Campus MVP, 2018).

```
//float: Emplea un tamaño de 32 bits (4 bytes).  
//Esto permite almacenar valores entre [-3.4028234E+8, 1.40239846E-45].  
  
float numeroFloat = 5976464F;  
  
//double: Emplea un tamaño de 64 bits (8 bytes).  
//Esto permite almacenar valores entre [-1.7976931348623157E+309, 4.94065645841246544E-324]  
  
double numeroDouble = 2654792142478F;
```

### **Ilustración 24: datos y variables flotantes.**

Fuente: Eclipse.

### **Booleanos y caracteres**

Aparte de esos seis tipos de datos vistos, destinados a trabajar con números en distintos rangos, Java establece otros dos tipos primitivos más:

```
//boolean: Se emplea con la finalidad de trabajar con valores verdaderos/falsos (booleanos).  
//Se traducen sus valores en true/falso.  
  
boolean variableBoolean = true;  
  
//char: Se emplea para almacenar caracteres individuales (letras, aunque puede contener números).  
//Utiliza 16 bits y se codifica sobre UTF-16 Unicode.  
  
char numeroChar = 1;  
char letraChar = 'D';
```

#### **Ilustración 25: datos y variables primitivas booleanas - char.**

Fuente: Eclipse.

### **Datos estructurados en Java**

Los tipos de datos primitivos se caracterizan por poder almacenar un único valor. Salvo este reducido conjunto de tipos de datos primitivos, que facilitan el trabajo con números, caracteres y valores booleanos, todos los demás tipos de Java son objetos, llamados también tipos de datos estructurados o «clases».

Los tipos de datos estructurados se denominan así porque en su mayor parte están destinados a contener múltiples valores de tipos más simples. También se les conoce como «tipos objeto» porque se usan para representar objetos (Campus MVP, 2018).

### **Cadenas de caracteres**

Pese a que las cadenas de caracteres no son un tipo simple en Java, sino una instancia de la clase *string*, el lenguaje otorga un tratamiento bastante especial a este tipo de dato, y esto hace que, a veces, nos parezca estar trabajando con un tipo primitivo. Aunque cuando se declare una cadena estamos creando un objeto, su declaración no se diferencia de la de una variable de tipo primitivo (Campus MVP, 2018):

```
//String: Se emplea creando una instancia de la clase String,  
//aunque parezca trabajar con datos primitivos.  
  
String variableString = "Hola a todos.";
```

#### **Ilustración 26: datos y variables estructuradas - string.**

Fuente: Eclipse.

**Las cadenas de caracteres se delimitan entre comillas dobles, en lugar de simples como los caracteres individuales.** No obstante, en la declaración no se indica explícitamente que se quiere crear un nuevo objeto de tipo *string*, esto es algo que infiere automáticamente el compilador.

Por tanto, las cadenas son objetos que disponen de métodos que permiten operar sobre la información almacenada en dicha cadena. De esta forma, se encuentran métodos para buscar una subcadena dentro de la cadena, sustituirla por otra, dividirla en varias cadenas atendiendo a un cierto separador, convertir a mayúsculas o minúsculas, etc. (Campus MVP, 2018).

### **Vectores – arrays**

Los vectores son una estructura de datos que permite almacenar un grupo de datos de un mismo tipo. Son conocidos popularmente como *arrays*. Asimismo, **es habitual llamar matrices a los vectores que trabajan con dos dimensiones.**

Los elementos de un vector o *array* se empiezan a numerar en el 0, y permiten gestionar desde una sola variable múltiples datos del mismo tipo.

Por ejemplo, si se debe almacenar una lista de diez números enteros, declararíamos un vector de tamaño 10 y de tipo entero, y no se declararían 10 variables separadas de tipo entero, una para cada número. (Campus MVP, 2018)

```
//Vector - Arreglo: Se emplea para almacenar
//un grupo de datos del mismo tipo.

//Forma 1:
int vectorNumeros1[] = new int[10];
//Forma 2:
int []vectorNumeros2 = new int[10];
//Forma 3:
int[] vectorNumeros3 = new int[10];
//Forma 4:
int vectorNumeros4[];
//Forma 5:
int vectorNumeros5[] = {};
//Forma 6:
int vectorNumeros6[] = {9,8,7,6,5,4,3,2,1,0};
//Forma 7:
int vectorNumeros7[] = new int[]{9,8,7,6,5,4,3,2,1,0};
```

**Ilustración 27: datos y variables estructuradas - vectores.**

Fuente: Eclipse.

```
//Matriz: Se emplea para almacenar un grupo de datos del mismo  
//de de forma bidimensional basados en [x],[y]
```

```
//Forma 1:  
int matrizNumeros1[][] = new int[4][5];  
//Forma 2:  
int [][]matrizNumeros2 = new int[4][5];  
//Forma 3:  
int[][] matrizNumeros3 = new int[4][5];  
//Forma 4:  
int matrizNumeros4[][];  
//Forma 5:  
int matrizNumeros5[][] = {};  
//Forma 6:  
int matrizNumeros6[][] = {{1,2},{3,9}};  
//Forma 7:  
int matrizNumeros7[][] = new int[][]{{6,2},{2,7}};
```

#### **Ilustración 28: datos y variables estructuradas - matrices.**

Fuente: Eclipse.

#### **Tipos estructurados definidos por el usuario**

Además de los tipos estructurados básicos (cadenas y vectores), en Java existen infinidad de clases en la plataforma y de terceros, para efectuar casi cualquier operación o tarea: leer y escribir archivos, enviar correos electrónicos, ejecutar otras aplicaciones o crear cadenas de texto más especializadas y miles de operaciones más. Todas esas clases también son tipos estructurados.

Y, por supuesto, se pueden crear clases propias para realizar todo tipo de tareas o almacenar información. Estos son tipos estructurados definidos por el usuario (Campus MVP, 2018).

```
//Variable de tipo persona (Persona es una clase);  
Persona P;  
//Variable de tipo animal (Animal es una clase);  
Animal A;  
//Variable de Java de la clase Math.  
Math M;  
//Variable de Java de la clase Scanner.  
Scanner S;  
//Variable de Java de la clase BufferedReader  
BufferedReader B;
```

#### **Ilustración 29: datos y variables estructuradas – definidas por el usuario.**

Fuente: Eclipse.



## Wrappers

Java cuenta con tipos de datos estructurados equivalentes a cada uno de los tipos primitivos. Otra de las finalidades de estos tipos «envoltorio» (*wrappers*) es facilitar el uso de esta clase de valores allí donde se espera un dato por referencia (un objeto) en lugar de un dato por valor.

De suerte que, por ejemplo, para representar un entero de 32 bits (*int*), Java determina una clase llamada *Integer* que representa y «envuelve» al mismo dato, y además le añade por encima ciertos métodos y propiedades útiles (Campus MVP, 2018).

Estos tipos equivalentes a los primitivos, pero en forma de objetos, son:

<input type="checkbox"/>		<input type="checkbox"/>	
<input type="checkbox"/>	<i>Byte</i>	<input type="checkbox"/>	<i>Integer</i>
<input type="checkbox"/>	<i>Long</i>	<input type="checkbox"/>	<i>Double</i>
<input type="checkbox"/>	<i>Float</i>	<input type="checkbox"/>	<i>Boolean</i>
<input type="checkbox"/>	<i>Short</i>	<input type="checkbox"/>	<i>Character</i>

### Ilustración 30: tipos en forma de objetos.

Fuente: autor, a partir de Campus MVP (2018).

```
//Estos tipos son equivalentes a los primitivos pero en forma
//de objetos son: Byte, Short, Integer, Long, Float, Double,
//Boolean y Character (8 igualmente).
```

```
//Representación de byte en Byte.
Byte numeroByte = 1;
//Representación de short en Short.
Short numeroShort = 2416;
//Representación de int en Integer.
Integer numeroInteger = 95256712;
//Representación de long en Long.
Long numeroLong = 5213714121L;
//Representación de float en Float.
Float numeroFloat = 6591342543251F;
//Representación de double en Double.
Double numeroDouble = 9.3;
//Representación de boolean en Boolean.
Boolean variableBoolean = true;
//Representación de char en Character.
Character variableCharacter = 'A';
Character numeroCharacter = 2;
```

### Ilustración 31: datos y variables estructuradas – *wrappers*.

Fuente: Eclipse.

Ya que se conocen un poco las estructuras de las variables dependiendo de su tipo y de su función, ahora es necesario precisar que hay características que se deben tener en cuenta al momento de utilizar las variables dentro del lenguaje:

- **No puede ser una palabra reservada del lenguaje o un literal booleano:**

```
String class "Hola";  
int true = 87;  
int double = 9;
```

- **No puede comenzar con un número:**

```
int 5numero;  
int 5 = 9;
```

- **No debe contener los símbolos que se utilicen como operadores:**

```
char genero-estudiante = 'M';  
double /altura = 1.80;  
String %nombre = "Alejandro";  
int edad*docente;
```

- **No debe contener espacios entre el nombre:**

```
String edad estudiante = 19;  
int numero celular;
```

- **Por convención, los nombres compuestos deben ir con el primer carácter inicial minúsculo, el resto de los caracteres iniciales en mayúsculas:**

```
String nombreDocente = "Andrés";  
boolean esEstudiante = true;
```

- **Se pueden crear dos variables del mismo tipo contiguamente:**

```
int a = 8, b = 9;  
String j, h;
```

- **Se pueden o no iniciar variables:**

```
String dia = "Martes";  
float peso;  
Character generoEstudiante = 'M';  
String fecha;
```

- Se puede asignar el valor de una variable a una nueva variable:

```
int numerosA[] = {3,9,2,4,8};  
int numerosB[] = numerosA;  
String nombrePerro = "Doggy";  
String nombreGato = nombrePerro;
```

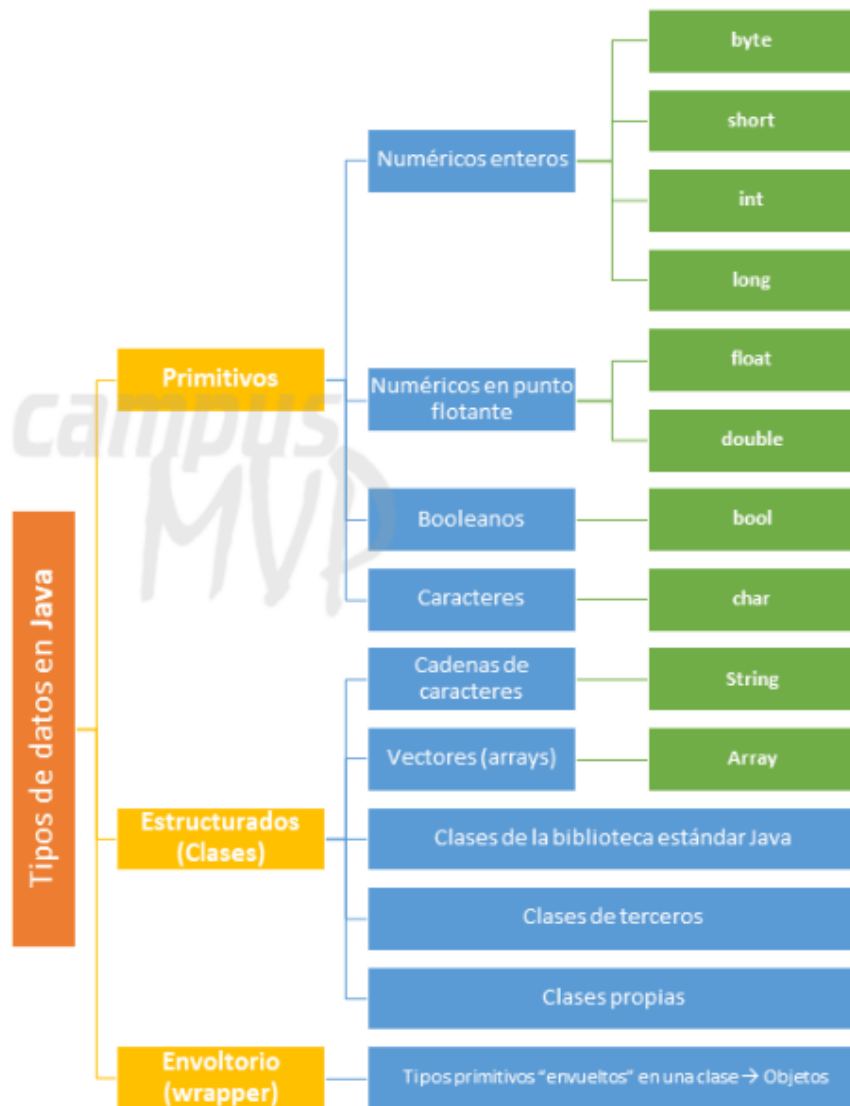
- Se puede castear el valor de las variables, es decir, pasar de *int* a *double*, de *string* a *int*, entre otras combinaciones:

```
String edadTexto = "19";  
int edadNumero = Integer.parseInt(edadTexto);  
String alturaDouble = "182";  
double alturaEntera = Double.parseDouble(alturaDouble);  
  
double numeroDouble = 9.2;  
int numeroEntero = (int) numeroDouble;  
  
int numeroEntero = 9;  
double numeroDouble = (double) numeroEntero;
```



Imagen: Pexels

A continuación, un resumen gráfico de todo lo referente a variables y tipos de datos para su mejor comprensión:



**Ilustración 32: variables y tipos de datos en Java.**

Fuente: Campus MVP (2018).

## Constantes

Una constante, desde el punto de vista de la programación, es un dato cuyo valor no cambia durante la ejecución del programa, en otras palabras, una vez que a una constante se le asigna un valor, este no podrá ser modificado y permanecerá así durante toda la ejecución del programa.

Las constantes son útiles para datos o atributos para los cuales el valor no tiene por qué cambiar. Con esto se puede evitar modificaciones en nuestro sistema que puedan causar problemas durante la ejecución del mismo. (Meza, 2018)

Java ha reservado la palabra clave «final» para definir constantes. En Java es muy simple definir constantes, solo basta con agregar el modificador «final» antes de la declaración del tipo (Meza, 2018).

```
final int documento = 921474159;  
final char letraA = 'A', letraB = 'B', letraC = 'C';  
final String acronimo = "CC";  
final float pi;  
pi = 3.1415F;
```

Algunas características se deben tener en cuenta con las constantes:

- Se pueden declarar N cantidad de constantes, siempre y cuando sean necesarias.
- **SIEMPRE** debe estar presente la palabra final.
- Se puede o no inicializar la constante en su creación.
- El valor que se asigne no puede ser modificado en ejecución.



## Tema 3: Operadores

### Operadores

Un operador hace operaciones sobre uno (**operador unario**), dos (**operador binario**) o tres (**operador ternario**) datos u operandos de tipo primitivo y devuelve un valor determinado también de un tipo primitivo. El tipo de valor devuelto tras la evaluación depende del operador y del tipo de los operandos.

Por ejemplo, los operadores aritméticos trabajan con operandos numéricos, realizan operaciones aritméticas básicas y devuelven el valor numérico correspondiente. Los operadores se pueden clasificar en distintos grupos según se muestra a continuación (Garro, 2014).

#### Operadores de asignación

El operador asignación (=) es un operador binario que asigna el valor del término de la derecha al operando de la izquierda. El operando de la izquierda suele ser el identificador de una variable. El término de la derecha es, en general, una expresión de un tipo de dato compatible; en particular, puede ser una constante u otra variable. Como caso particular, y a diferencia de los demás operadores, este operador no se evalúa devolviendo un determinado valor. (Garro, 2014)

```
int numero1 = 8;
```

Se crea una variable (**numero1**) entera (**int**) y se asigna el valor 8.

Operador	Símbolo	Ejemplo	Resultado
Asignación	=	numero1 = 8	numero1 vale 8

**Tabla 1: operadores de asignación.**

Fuente: autor, a partir de Garro (2014).

```
int numero1 = 8;  
int numero2 = numero1;
```

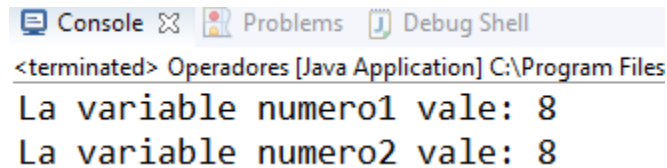
Se crea una variable (**numero2**) entera (**int**) y se asigna el valor de la variable (**numero1**).

Operador	Símbolo	Ejemplo	Resultado
Asignación	=	numero2 = numero1	Numero2 vale 8

**Tabla 2: operadores de asignación.**

Fuente: autor, a partir de Garro (2014).

```
System.out.println("La variable numero1 vale: "+numero1);
System.out.println("La variable numero2 vale: "+numero2);
```



Console Problems Debug Shell  
<terminated> Operadores [Java Application] C:\Program Files  
La variable numero1 vale: 8  
La variable numero2 vale: 8

Se imprime el valor de ambas variables concatenando estas con el «+».

## Operadores aritméticos

El lenguaje de programación Java tiene varios operadores aritméticos para los datos numéricos enteros y reales. En la siguiente tabla se resumen los diferentes operadores de esta categoría.

Operador	Descripción
-	Operador unario de cambio de signo
+	Suma
-	Resta
*	Producto
/	División
%	Módulo

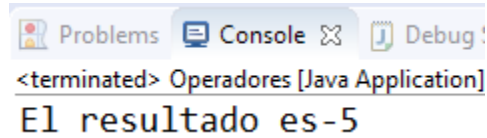
**Tabla 3: Operadores aritméticos.**

Fuente: autor, a partir de Garro (2014).

El resultado exacto depende de los tipos de operando involucrados. Es necesario considerar las siguientes particularidades, de acuerdo con Garro (2014):

- El resultado es de tipo *long* si, al menos, uno de los operandos es de tipo *long* y ninguno es real (*float* o *double*).

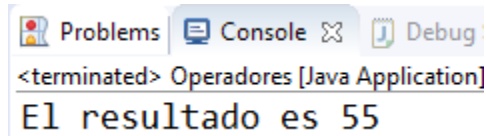
```
long numero1 = 8;
int numero2 = 13;
long resta = numero1-numero2;
System.out.println("El resultado es "+resta);
```



Problems Console Debug  
<terminated> Operadores [Java Application]  
El resultado es -5

- El resultado es de tipo `int` si ninguno de los operandos es de tipo `long` y tampoco es real (`float` o `double`).

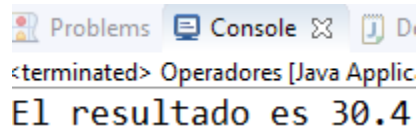
```
int numero1 = 3;
int numero2 = 52;
int suma = numero1+numero2;
System.out.println("El resultado es "+suma);
```



Problems Console Debug  
<terminated> Operadores [Java Application]  
El resultado es 55

- El resultado es de tipo `double` si, al menos, uno de los operandos es de tipo `double`.

```
double numero1 = 7.6;
int numero2 = 4;
double multiplicacion = numero1*numero2;
System.out.println("El resultado es "+multiplicacion);
```



Problems Console Debug  
<terminated> Operadores [Java Application]  
El resultado es 30.4

- El resultado es de tipo `float` si, al menos, uno de los operandos es de tipo `float` y ninguno es `double`.

```
float numero1 = 2.3F;
int numero2 = 5;
float division = numero1/numero2;
System.out.println("El resultado es "+division);
```

Problems Console Debug Shell  
<terminated> Operadores [Java Application] C:\Pro  
El resultado es 0.45999998

- El formato empleado para la representación de datos enteros es el complemento a dos. En la aritmética entera no se producen nunca desbordamientos (*overflow*) aunque el resultado sobrepase el intervalo de representación (int o long).
- La división entera o módulo se trunca hacia 0. La división o el resto de dividir por 0 es una operación válida que genera una excepción `ArithmeticException` que puede dar lugar a un error de ejecución y la consiguiente interrupción de la ejecución del programa.

```
double division = 15/0;  
System.out.println("El resultado es "+division);
```

Exception in thread "main" [java.lang.ArithmeticException: / by zero](#)  
at Operadores.Operadores.main([Operadores.java:12](#))

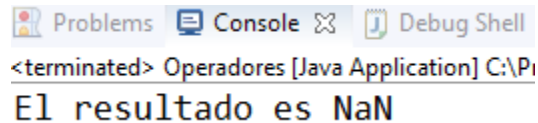
- La aritmética real (en coma flotante) puede desbordar al infinito (demasiado grande, *overflow*) o hacia 0 (demasiado pequeño, *underflow*).

```
int numero1 = 248154379456;  
double numero2 = 2e308;  
System.out.println("El numero1 es "+numero1);  
System.out.println("El numero2 es "+numero2);
```

Exception in thread "main" [java.lang.Error: Unresolved compilation problems:](#)  
The literal 248154379456 of type int is out of range  
The literal 2e308 of type double is out of range  
at Operadores.Operadores.main([Operadores.java:7](#))

- El resultado de una expresión inválida, por ejemplo, dividir infinito por infinito, no genera una excepción ni un error de ejecución: es un valor NaN (*not a number*).

```
double modulo = 2.0%0;  
System.out.println("El resultado es "+modulo);
```



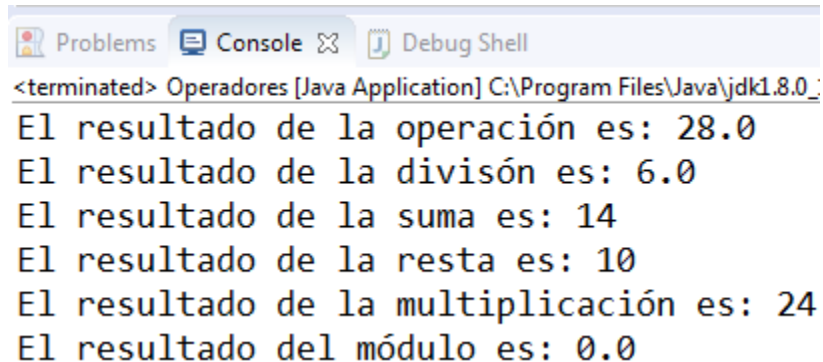
Problems Console Debug Shell  
<terminated> Operadores [Java Application] C:\Pi  
El resultado es NaN

## Ejercicios

1. Con base en dos números enteros, realiza las cinco operaciones básicas vistas hasta el momento y muestra el resultado.
2. Crea una operación utilizando números enteros y los símbolos aritméticos.

```
int numero1 = 12;
int numero2 = 2;
double division;
int suma, resta, multiplicacion;
double modulo;
double operacion;

operacion = 9/3*5-3+8*2;
System.out.println("El resultado de la operación es: "+operacion);
division = numero1/numero2;
System.out.println("El resultado de la división es: "+division);
suma = numero1+numero2;
System.out.println("El resultado de la suma es: "+suma);
resta = numero1-numero2;
System.out.println("El resultado de la resta es: "+resta);
multiplicacion = numero1*numero2;
System.out.println("El resultado de la multiplicación es: "+multiplicacion);
modulo = numero1%numero2;
System.out.println("El resultado del módulo es: "+modulo);
```



Problems Console Debug Shell  
<terminated> Operadores [Java Application] C:\Program Files\Java\jdk1.8.0\_  
El resultado de la operación es: 28.0  
El resultado de la división es: 6.0  
El resultado de la suma es: 14  
El resultado de la resta es: 10  
El resultado de la multiplicación es: 24  
El resultado del módulo es: 0.0

## Operadores aritméticos incrementales

«Los operadores aritméticos incrementales son operadores unarios (un único operando). El operando puede ser numérico o de tipo *char* y el resultado es del

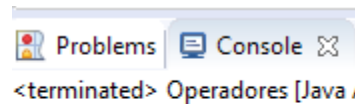
mismo tipo que el operando. Estos operadores pueden emplearse de dos formas dependiendo de su posición con respecto al operando». (Garro, 2014)

Operador	Descripción	Ejemplo	Resultado
++A	Incrementa el valor y luego se utiliza la variable	A = 5;	A = 6;
		B = ++A;	B = 6;
A++	Utiliza la variable y luego incrementa el valor	A = 5;	A = 6;
		B = A++;	B = 5;
--A	Decrementa el valor y luego se utiliza la variable	A = 5;	A = 4;
		B = --A;	B = 4;
A--	Utiliza la variable y luego decrementa el valor	A = 5;	A = 4;
		B = A--;	B = 5;

**Tabla 4: operadores aritméticos incrementales.**

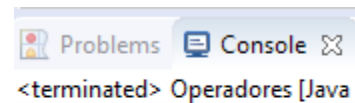
Fuente: autor, a partir de Garro (2014).

```
int a = 5;
int b = ++a;
```



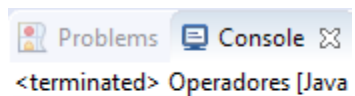
```
System.out.println(a); 6
System.out.println(b); 6
```

```
int a = 5;
int b = a++;
```



```
System.out.println(a); 6
System.out.println(b); 5
```


```
int a = 5;
int b = --a;
```



```
System.out.println(a); 4
System.out.println(b); 4
```

```
int a = 5;
int b = a--;

System.out.println(a); 4
System.out.println(b); 5
```

Problems Console 

<terminated> Operadores [Java]

## Operadores aritméticos combinados

«Combinan un operador aritmético con el operador asignación. Como en el caso de los operadores aritméticos, pueden tener operandos numéricos enteros o reales y el tipo específico de resultado numérico dependerá del tipo de estos. En la siguiente tabla se resumen los diferentes operadores de esta categoría». (Garro, 2014)


Operador	Descripción	Ejemplo	Resultado
+=	Suma combinada	a += b	a = a + b
-=	Resta combinada	a -= b	a = a - b
*=	Multiplicación combinada	a *= b	a = a * b
/=	División combinada	a /= b	a = a / b
%=	Módulo combinado	a %= b	a = a % b

**Tabla 5: operadores aritméticos combinados.**

Fuente: autor, a partir de Garro (2014).

```
int a = 5;
int b = 2;
b += a;


System.out.println(b); 7
```

Problems Console 

<terminated> Operadores [Java]

```
int a = 5;
int b = 2;
b -= a;


System.out.println(b); -3
```

Problems Console 

<terminated> Operadores [Java]




```
int a = 5;
int b = 2;
b *= a;
```

Problems Console   
<terminated> Operadores [Java


```
System.out.println(b); 10
```

```
int a = 5;
double b = 2.0;
b /= a;
```

Problems Console   
<terminated> Operadores [Java

```
System.out.println(b); 0.4
```

```
int a = 5;
int b = 2;
b %= a;
```

Problems Console   
<terminated> Operadores [Java

```
System.out.println(b); 2
```

## Operadores de relación

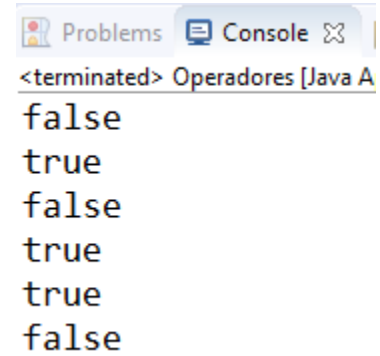
«Realizan comparaciones entre datos compatibles de tipos primitivos (numéricos, carácter y booleanos), teniendo siempre un resultado booleano. Los operandos booleanos solo pueden emplear los operadores de igualdad y desigualdad». (Garro, 2014)

Operador	Descripción	Ejemplo	Resultado
==	Igual que	5 == 4	False
!=	Diferente que	4 != 2	True
<	Menor que	5 < 2	False
>	Mayor que	5 > -5	True
<=	Menor o igual que	2 <= 3	True
>=	Mayor o igual que	3 >= 9	False

**Tabla 6: operadores de relación.**

Fuente: autor, a partir de Garro (2014).

```
boolean igualQue = 5 == 4;
System.out.println(igualQue);
boolean distintoQue = 4 != 2;
System.out.println(distintoQue);
boolean menorQue = 5 < 3;
System.out.println(menorQue);
boolean mayorQue = 5 > -5;
System.out.println(mayorQue);
boolean menorIgualQue = 2 <= 3;
System.out.println(menorIgualQue);
boolean mayorIgualQue = 3 >= 9;
System.out.println(mayorIgualQue);
```



Problems Console |  
<terminated> Operadores [Java A  
false  
true  
false  
true  
true  
false

## Operadores lógicos

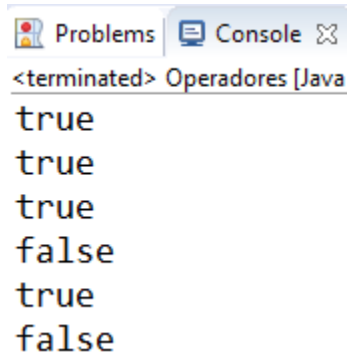
«Realizan operaciones sobre datos booleanos y tienen como resultado un valor booleano». Los diferentes operadores de esta categoría se resumen en la siguiente tabla:

Operador	Descripción	Ejemplo	Resultado
!	Negación (unario)	!False	<i>True</i>
		!(5 == 5)	<i>False</i>
	Suma lógica (binario)	True   False	<i>True</i>
		(5 == 5)   (5 < 4)	<i>True</i>
^	Suma lógica exclusiva	True ^ False	<i>True</i>
		(5 == 5) ^ (5 < 4)	<i>True</i>
&	Producto lógico (binario)	True & False	<i>False</i>
		(5 == 5) & (5 < 4)	<i>False</i>
	Suma lógica cortocircuito	True    False	<i>True</i>
		(5 == 5)    (5 < 4)	<i>True</i>
&&	Producto lógico cortocircuito	True && False	<i>False</i>
		(5 == 5) && (5 < 4)	<i>False</i>

**Tabla 7: operadores lógicos.**

Fuente: autor, a partir de Garro (2014).

```
boolean negacion = !false;
System.out.println(negacion);
boolean sumaLogica = true | false;
System.out.println(sumaLogica);
boolean sumaLogicaExclusiva = (5 == 5) ^ (5 < 4);
System.out.println(sumaLogicaExclusiva);
boolean productoLogico = (5 == 5) & (5 < 4);
System.out.println(productoLogico);
boolean sumaLogicaCortocircuito = true || false;
System.out.println(sumaLogicaCortocircuito);
boolean productoLogicoCortocircuito = (5 == 5) && (5 < 4);
System.out.println(productoLogicoCortocircuito);
```



Problems Console

<terminated> Operadores [Java]

```
true
true
true
false
true
false
```

## Operador condicional


Este operador ternario permite devolver valores en función de una expresión lógica.

Operador	Descripción	Ejemplo	Resultado
?:	Operador condicional	a = 4;	b = 9;
		b = a == 4 ? a + 5 : 6 - a;	
		b = a > 4 ? a * 7 : a + 8;	b = 12;

**Tabla 8: operador condicional.**

Fuente: autor, a partir de Garro (2014).

```
int a = 4;
int b = a == 4 ? a + 5 : 6 - a;
System.out.println(b);
b = a > 4 ? a * 7 : a + 8;
System.out.println(b);
```

Problems Console 

<terminated> Operadores [Java]

9

12

## Operador de concatenación de cadenas


«El operador concatenación + es un operador binario que devuelve una cadena resultado de concatenar las dos cadenas que actúan como operandos. Si solo uno de los operandos es de tipo cadena, el otro operando se convierte implícitamente en tipo cadena». (Garro, 2014)

Operador	Descripción	Ejemplo	Resultado
+	Operador de concatenación	"Hola" + "Juan"	"HolaJuan"

**Tabla 9: operador de concatenación de cadenas.**

Fuente: autor, a partir de Garro (2014).

```
String saludo = "Hola" + "Juan";
System.out.println(saludo);
```

Problems Console 

<terminated> Operadores [Java]

HolaJuan

## Operadores de separación

Algunos caracteres tienen un significado especial en el lenguaje Java. Los diferentes separadores que pueden encontrarse en el código fuente de un programa se resumen en la siguiente tabla (Garro, 2014):

Separador	Descripción
()	Permite modificar la prioridad de una expresión, contener expresiones para el control del flujo y realizar conversiones de tipo. Por otro lado, pueden contener la lista de parámetros o argumentos, tanto en la definición de un método como en la llamada al mismo.
{ }	Permite definir bloques de código y ámbitos y contener los valores iniciales de un <i>array</i> .

[]	Permite declarar bloques de <i>array</i> (vectores o matrices) y referenciar sus elementos.
;	Permite separar sentencias.
,	Permite separar identificadores consecutivos en la declaración de variables y en las listas de parámetros. También se emplea para encadenar sentencias dentro de un ciclo <i>for</i> .
.	Permite separar el nombre de un atributo o método de su instancia de referencia. También separa el identificador de un paquete de los subpaquetes y clases.

**Tabla 10: operadores de separación.**

Fuente: autor, a partir de Garro (2014).

```
int suma = (5+9)*2;
int arreglo[] = {2, 9};
int a, b, c;
double euler = Math.E;
```

### Prioridad entre operadores

Si dos operadores se encuentran en la misma expresión, el orden en el que se evalúan puede determinar el valor de la expresión. En la siguiente tabla se muestra el orden o prioridad en el que se ejecutan los operadores que se encuentren en la misma sentencia. Los operadores de la misma prioridad se evalúan de izquierda a derecha dentro de la expresión. (Garro, 2014)

Prioridad	Operador	Tipo	Operación
1	++	Aritmético	Incremento previo o posterior
	--	Aritmético	Decremento previo o posterior
	+, -	Aritmético	Suma, resta
	~	Integral	Cambio de bits
	!	Booleano	Negación
2	Tipo	Cualquiera	NA
3	*, /, %	Aritmético	Multiplicación, división y residuo

4	+, -	Aritmético	Suma, resta
	+	Cadena	Concatenación de cadenas
5	<<	Integral	Desplazamiento de bits a la izquierda
	>>	Integral	Desplazamiento de bits a la derecha con inclusión de signo
	>>>	Integral	Desplazamiento de bits a la derecha con inclusión del cero
6	<, <=	Aritmético	Menor que, menor o igual que
	>, >=	Aritmético	Mayor que, mayor o igual que
	instanceof	Objeto, tipo	Comparación de tipos
7	==	Primitivo	Igual
	!=	Primitivo	Desigual
	==	Objeto	Igual
	!=	Objeto	Desigual
8	&	Integral	Cambio de bits AND
	&	Booleano	Producto booleano
9	^	Integral	Cambio de bits XOR
	^	Booleano	Suma exclusiva booleana
10		Integral	Cambio de bits OR
		Booleano	Suma booleana
11	&&	Booleano	AND condicional
12		Booleano	OR condicional
13	? :	Booleano	Operador condicional ternario
14	=	Variable	Asignación
	*=, /=, %=		Asignación con operación
	+=, -=		
	<<=, >>=		
	>>>=		
	&=, ^=,  =		

**Tabla 11: prioridad entre operadores.**  
Fuente: autor, a partir de Garro (2014).

## Tema 4: *Math*

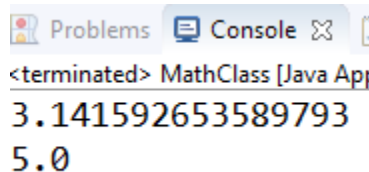
La clase *math* constituye la librería matemática de Java. Contiene funciones como las de todos los lenguajes, parece que se han metido en una clase solamente a propósito de agrupación, por eso se encapsulan en *math*, y lo mismo sucede con las demás clases que corresponden a objetos que tienen un tipo equivalente (*character*, *float*, etc.).

«La clase *math* contiene métodos para realizar operaciones numéricas básicas, como funciones exponenciales elementales, logaritmos, raíces cuadradas y trigonométricas». (Oracle, s.f.)

```
public class MathClass {

    public static void main(String[] args)
    {
        double valorPi = Math.PI;
        System.out.println(valorPi);

        double raiz = Math.sqrt(25);
        System.out.println(raiz);
    }
}
```



Problems Console [Java App]  
<terminated> MathClass [Java App]  
3.141592653589793  
5.0

Método	Descripción
Math.abs(x)	Devuelve el valor absoluto de un número
Math.acos(x)	Devuelve el arco coseno de un número
Math.acosh(x)	Devuelve el arco coseno hiperbólico de un número



Math.asin(x)	Devuelve el arco seno de un número
Math.atan(x)	Devuelve el arco tangente de un número
Math.atan2(y, x)	Devuelve el arco tangente del cociente de sus argumentos
Math.cbrt(x)	Devuelve la raíz cúbica de un número
Math.ceil(x)	Devuelve el entero más pequeño mayor o igual que un número
Math.cos(x)	Devuelve el coseno de un número
Math.cosh(x)	Devuelve el coseno hiperbólico de un número
Math.exp(x)	Devuelve $E^x$ , donde x es el argumento, y E es la constante de Euler (2.718...), la base de los logaritmos naturales
Math.expm1(x)	Devuelve $e^x - 1$
Math.floor(x)	Devuelve el mayor entero menor que o igual a un número
Math.hypot(x, y)	Devuelve la raíz cuadrada de la suma de los cuadrados de sus argumentos
Math.log(x)	Devuelve el logaritmo natural (log, también ln) de un número
Math.max(a, b)	Devuelve el mayor de cero o más números
Math.min(a, b)	Devuelve el más pequeño de cero o más números
Math.pow(x, y)	Las devoluciones de base a la potencia de exponente, que es base <i>exponent</i>
Math.random()	Devuelve un número pseudoaleatorio entre 0 y 1
Math.round(x)	Devuelve el valor de un número redondeado al número entero más cercano
Math.sin(x)	Devuelve el seno de un número
Math.sinh(x)	Devuelve el seno hiperbólico de un número

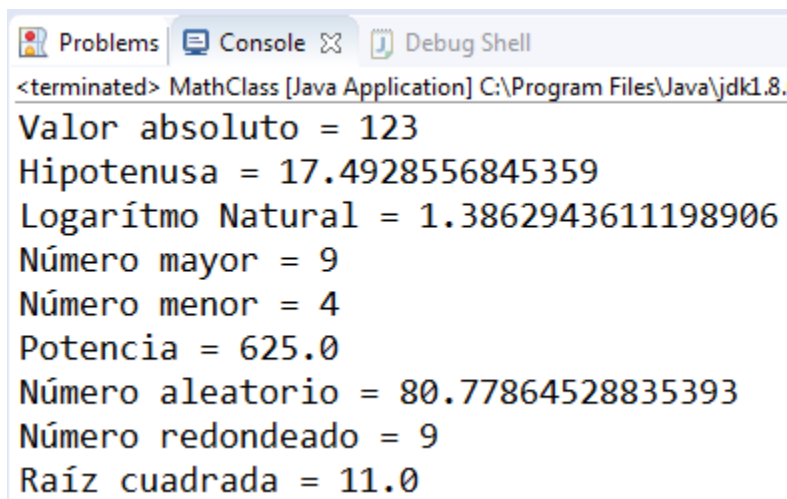
Math.sqrt(x)	Devuelve la raíz cuadrada positiva de un número
Math.tan(x)	Devuelve la tangente de un número
Math.tanh(x)	Devuelve la tangente hiperbólica de un número
Math.E	Devuelve el valor de E
Math.PI	Devuelve el valor de PI

**Nota:** estos son algunos de los métodos, la lista completa la podrás encontrar en: Oracle, (s.f.): <https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html>

**Tabla 12: métodos de la clase *math*.**

Fuente: autor, a partir de Garro (2014).

```
public static void main(String[] args)
{
    System.out.println("Valor absoluto = " + Math.abs(-123));
    System.out.println("Hipotenusa = " + Math.hypot(15, 9));
    System.out.println("Logaritmo Natural = " + Math.log(4));
    System.out.println("Número mayor = " + Math.max(4, 9));
    System.out.println("Número menor = " + Math.min(4, 9));
    System.out.println("Potencia = " + Math.pow(5, 4));
    System.out.println("Número aleatorio = " + Math.random()*100);
    System.out.println("Número redondeado = " + Math.round(Math.random()*10));
    System.out.println("Raíz cuadrada = " + Math.sqrt(121));
}
```



```
<terminated> MathClass [Java Application] C:\Program Files\Java\jdk1.8.
Valor absoluto = 123
Hipotenusa = 17.4928556845359
Logaritmo Natural = 1.3862943611198906
Número mayor = 9
Número menor = 4
Potencia = 625.0
Número aleatorio = 80.77864528835393
Número redondeado = 9
Raíz cuadrada = 11.0
```

### Palabras reservadas del lenguaje

Las palabras reservadas, como su nombre lo indica, son términos que el lenguaje de programación ya ha reservado para realizar ciertas tareas, por lo que no pueden ser usadas para otras. En Java actualmente existen 57 palabras reservadas, de las cuales 55 están en uso y 2 ya no. Por sus funciones especiales en el lenguaje, la mayoría de entornos de desarrollo integrados para Java usan el resaltado de sintaxis para mostrar las palabras clave en un color distinto, así se identifican fácilmente (Roldán, 2022).

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

#### **Ilustración 33: palabras reservadas.**

Fuente: Roldán (2022).

Algo importante para tener en cuenta es que las palabras reservadas no pueden ser utilizadas como nombres de variables, clases o métodos o cualquier otro identificador. A este listado se pueden agregar *true*, *false* y *null*. Aunque *const* y *goto* son palabras reservadas, estas no son utilizadas en la actualidad.



## Ejercicio práctico

Ahora que has revisado algunas definiciones de Java y conoces un poco su alcance, te invitamos a construir tu propia definición de **Java**, a partir de los conceptos facilitados en esta guía y que son fortalecidos con los recursos disponibles para el aprendizaje. El uso de mapas conceptuales te puede ayudar a conocer la relación entre los elementos que conforman un concepto.

Ahora que realizaste tu primer programa en Java sobre el Hola Mundo, conociste las primeras características del lenguaje y viste cómo se realiza el ejercicio en otros 35 lenguajes, te preguntamos: ¿por qué «Hola Mundo»?

¿Deseas profundizar en la temática de tipos de datos y variables? Entonces te sugerimos realizar los siguientes ejercicios que pondrán a prueba los conocimientos adquiridos: módulo 1 – ejercicios con variables.

¿Deseas profundizar en la temática de operadores? Entonces te sugerimos desarrollar los siguientes ejercicios que pondrán a prueba los conocimientos adquiridos: módulo 1 – ejercicios con operadores.

**¡Inténtalo!**



## Referencias bibliográficas

- Alegsa, A. (2023, 9 de julio). *Definición de dato (informática)*. Alegsa.  
<https://www.alegsa.com.ar/Dic/dato.php>
- Campus MVP. (2018, 17 de julio). Variables y tipos de datos en Java: tipos simples, clases y tipos envoltorio o wrapper [imagen]. Campus MVP.  
<https://tinyurl.com/yc5xdrt7>
- Castro, C. (2019, 1 de noviembre). *¿Qué es un Applet en Java?* About Español.  
<https://www.aboutespanol.com/que-es-un-applet-en-java-157840>
- Denisable Programación. (2016, 27 de octubre). *Java desde cero con Eclipse [parte 1] (¡Hola Mundo!)* [Video]. YouTube. <https://tinyurl.com/23t48rvy>
- Eclipse Foundation. (2023). *Eclipse*. Eclipse Foundation. <https://www.eclipse.org/>
- Garro, A. (2014, 15 de abril). Capítulo 4. Operadores. En *Java*. Arkaitz Garro.  
<https://www.arkaitzgarro.com/java/capitulo-4.html>
- Guru99. (2023, 2 de noviembre). Variables Java y tipos de datos [imagen].  
Guru99. <https://guru99.es/java-variables/>
- Leducatec. (2019, 6 de enero). Descarga e instalación de Eclipse [video].  
YouTube. <https://www.youtube.com/watch?v=j8ngvGNFXKA>
- Oracle. (s.f.). Class Math. En *Java Docs*. Oracle Docs.  
<https://tinyurl.com/785d8aym>
- Meza, J. (2018). Declaración y uso de constantes en Java. Uso del final en Java, datos final. Programar Ya. <https://tinyurl.com/yvvayfdw>
- Netec. (2023). *¿Qué es programación?* Netec. <https://tinyurl.com/mreeyrd8>
- Roldán, Á. (2022). *Palabras reservadas en Java*. Ciber Aula.  
<https://tinyurl.com/2yuxnsvw>
- Significados. (2023, 6 de octubre). Qué es la información. En *Significados*.  
<https://www.significados.com/informacion/>
- Sistemas. (s.f.). *Definición de variable*. Sistemas. <https://tinyurl.com/3bpanfty>
- Wikilibros. (2019, 31 de mayo). Programación en Java/Variables. Wikilibros.  
[https://es.wikibooks.org/wiki/Programaci%C3%B3n\\_en\\_Java/Variables](https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_Java/Variables)


Wikipedia. (2023a, 29 de octubre). Paradigma de programación. En *Wikipedia*.

[https://es.wikipedia.org/wiki/Paradigma\\_de\\_programaci%C3%B3n](https://es.wikipedia.org/wiki/Paradigma_de_programaci%C3%B3n)

Wikipedia. (2023b, 14 de noviembre). Teorema del programa estructurado. En *Wikipedia*.

[https://es.wikipedia.org/wiki/Teorema\\_del\\_programa\\_estructurado](https://es.wikipedia.org/wiki/Teorema_del_programa_estructurado)

Wikipedia. (2023c, 16 de noviembre). Java (lenguaje de programación). En *Wikipedia*. <https://tinyurl.com/y5knnyt7>



Esta guía fue elaborada para ser utilizada con fines didácticos como material de consulta de los participantes en el Diplomado Virtual en PROGRAMACIÓN EN JAVA del Politécnico de Colombia, y solo podrá ser reproducida con esos fines. Por lo tanto, se agradece a los usuarios referirla en los escritos donde se utilice la información que aquí se presenta.

## **GUÍA DIDÁCTICA 1**

M2-DV59-GU01

MÓDULO 1: FUNDAMENTOS

© DERECHOS RESERVADOS - POLITÉCNICO DE COLOMBIA, 2023  
Medellín, Colombia

Proceso: Gestión Académica Virtual  
Realización del texto: Diego Palacio, docente  
Revisión del texto: Comité de Revisión  
Diseño: Comunicaciones

Editado por el Politécnico de Colombia