

Indice

[1. Java desde cero 1](#_Toc177667937)

[1.1 ¿Qué es Java? 1](#_Toc177667938)

[Lenguaje de objetos 1](#_Toc177667939)

[Java con vscode 1](#_Toc177667940)

[2. Primeras cosillas con java 5](#_Toc177667941)

[2.1 Estructura de un programa en Java 5](#_Toc177667942)

[2.2 Crear código en VSCode 7](#_Toc177667943)

[2.3 Ejecutar código 9](#_Toc177667944)

[3. Variables 10](#_Toc177667945)

[3.1 Datos primitivos 10](#_Toc177667946)

[3.2 Tipos de datos primitivos 11](#_Toc177667947)

[3.3 Valores por defecto 12](#_Toc177667948)

[3.4 Tipos de variables 12](#_Toc177667949)

[3.5 Constantes 12](#_Toc177667950)

[3.6 Ejemplo completo 13](#_Toc177667951)

[Explicación 14](#_Toc177667952)

[sALIDA DEL CODIGO 14](#_Toc177667953)

[4. Operadores 15](#_Toc177667954)

[4.1 Operadores aritméticos 15](#_Toc177667955)

1. Java desde cero
   1. ¿Qué es Java?

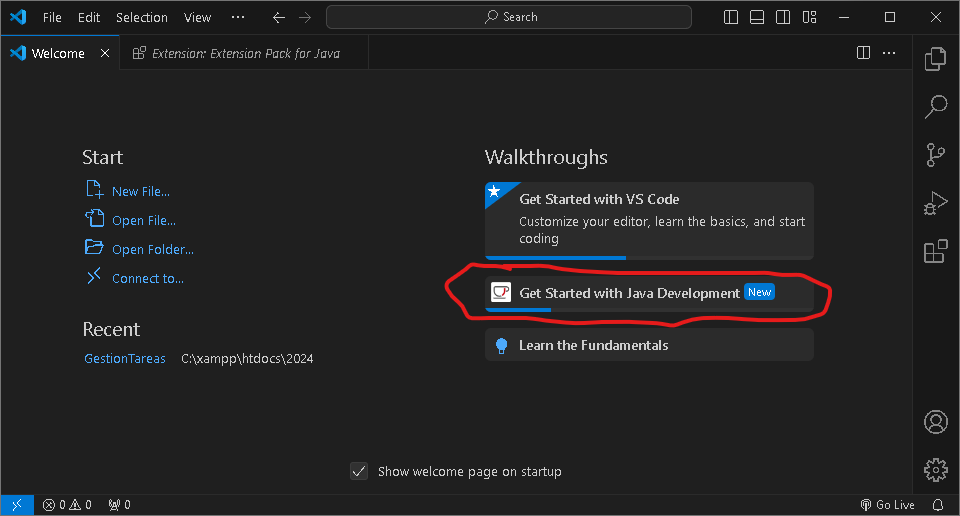
Lenguaje de objetos

Java con vscode

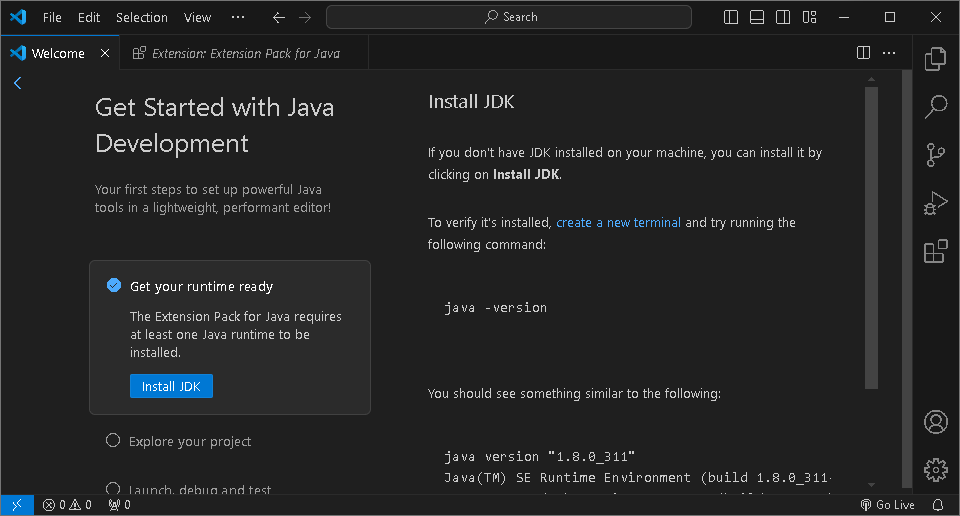
Una vez tenemos instalado el VSCode en nuestro equipo debemos instalar la siguiente extensión:

Extension Pack for Java:

<https://marketplace.visualstudio.com/items?itemName=vscjava.vscode-java-pack>

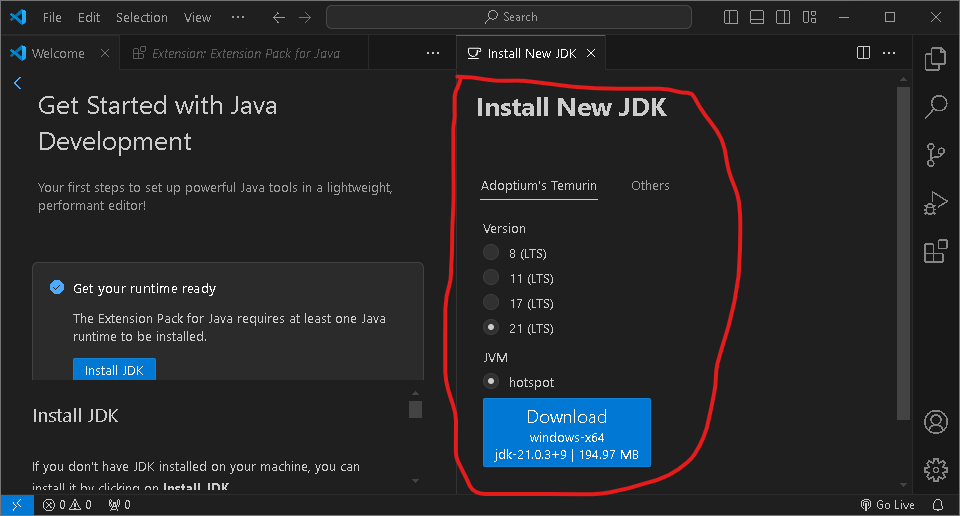
Una vez descargada e instalada. Cerramos VSCode y lo volvemos a abrir. 

Seleccionamos la opción marcada. Y se nos abrirá la siguiente pestaña:



Seleccionamos ***Install JDK***.

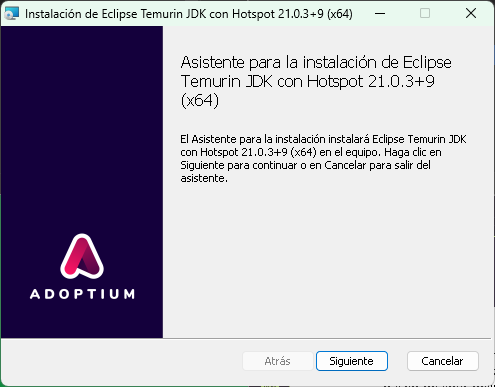
Se abrirá una 2º ventana, donde podremos elegir la versión y otras configuraciones a tu gusto.

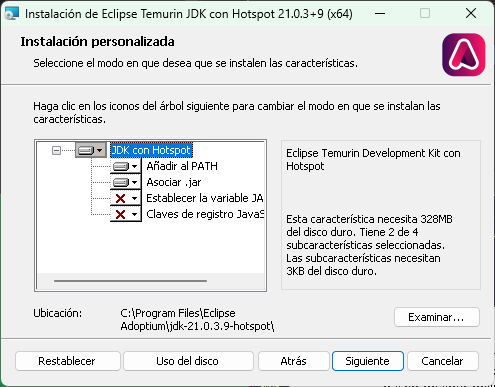


Descargamos el instalador

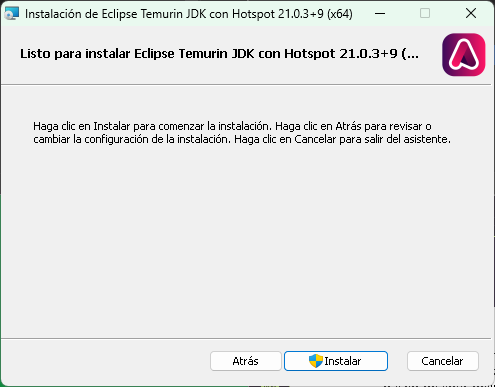


Una vez descargado lo abrimos para continuar con la instalación: (sucesión de capturas “irrelevantes”) damos a siguiente siempre

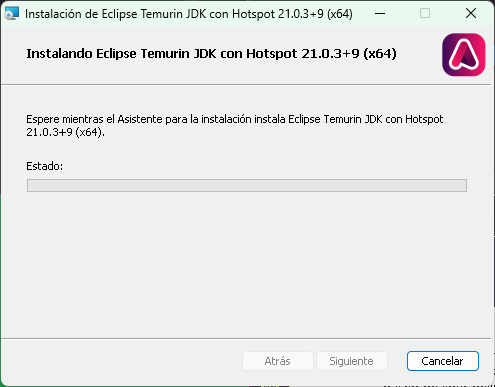


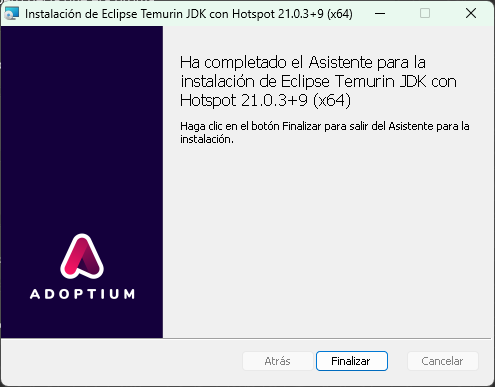


Ahora le damos a instalar:

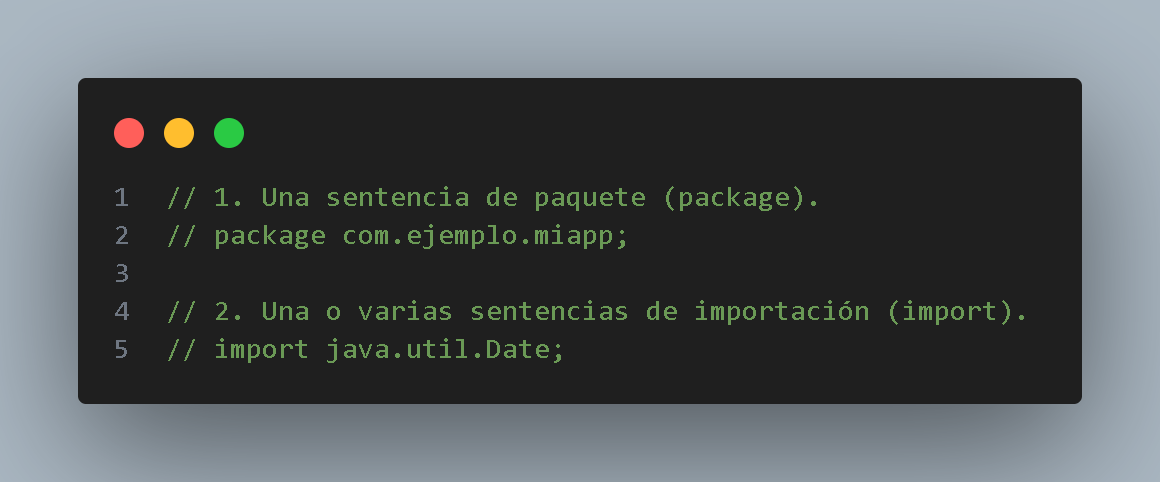


Esperamos a que finalice la instalación.





1. Primeras cosillas con java
   1. Estructura de un programa en Java



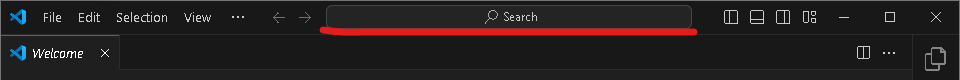


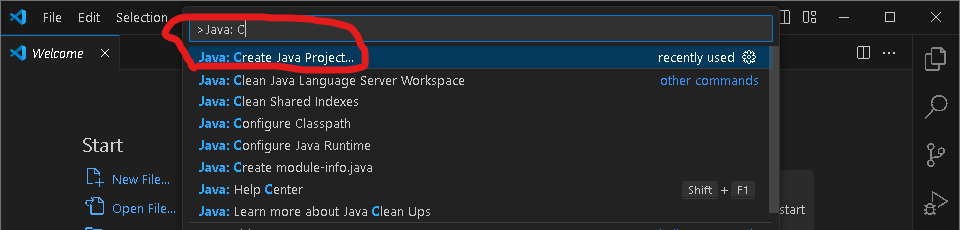


Resumiendo.

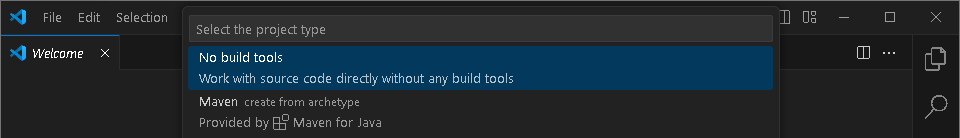
1. Sentencia de paquete (package)
2. Una o varias sentencias de importación (import)
3. Declaración de las clases privadas deseadas
4. Declaración de clase pública
   1. Variables de clase
   2. Variables de instancia
   3. Definición de constructores
   4. Definiciones de métodos
   5. Comentarios por to’ los laos’
   6. Crear código en VSCode

Para crear un proyecto en Java debemos dirigirnos a la barra superior y poner lo siguiente:

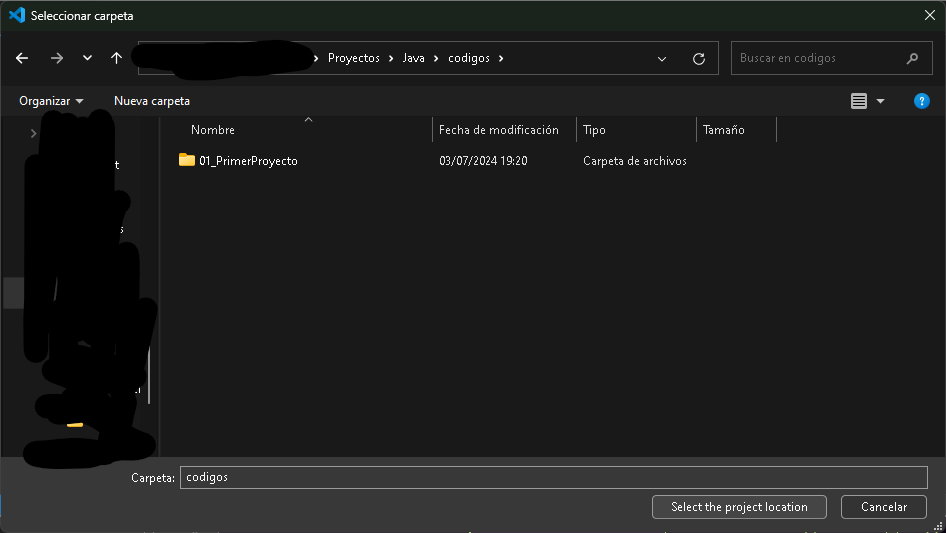




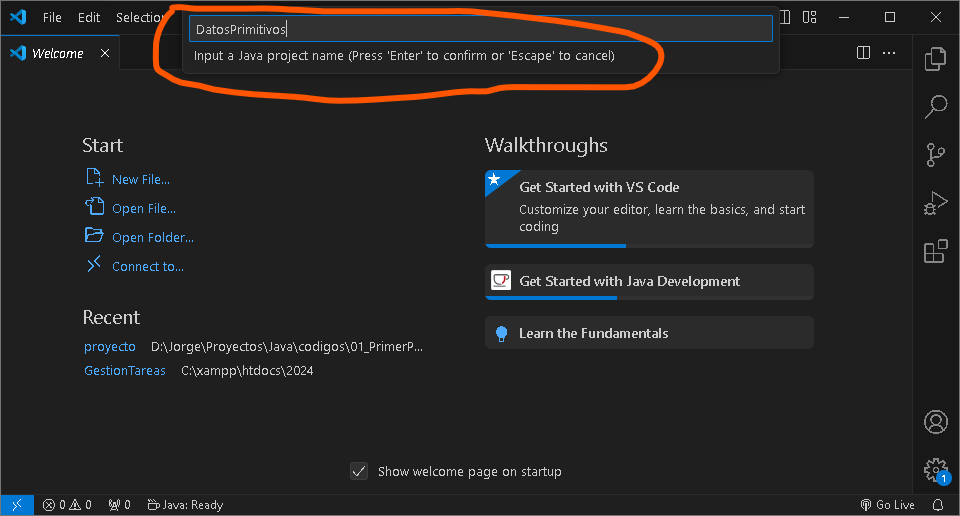
Seleccionamos No build tools:



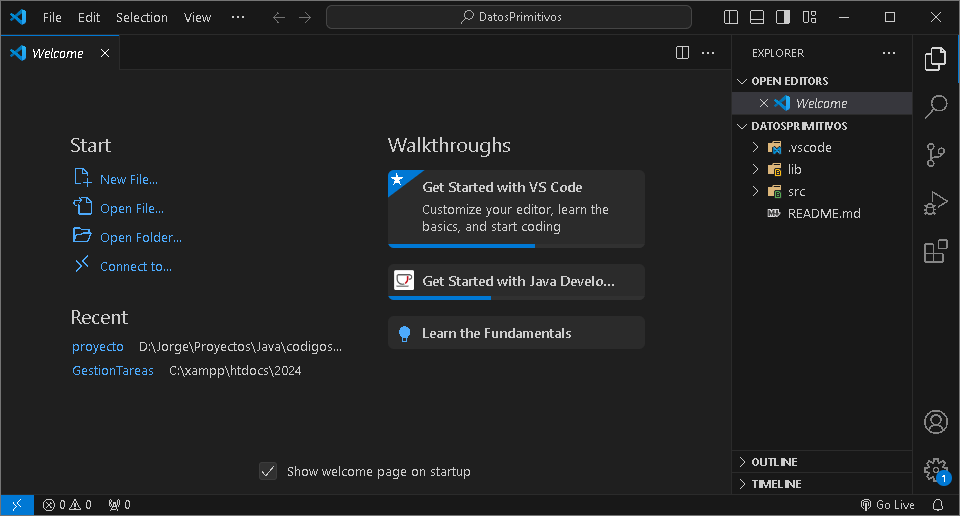
Elegimos nuestra carpeta de proyectos:



Le damos nombre a nuestro proyecto.

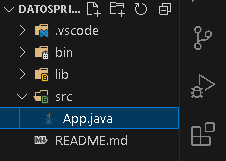


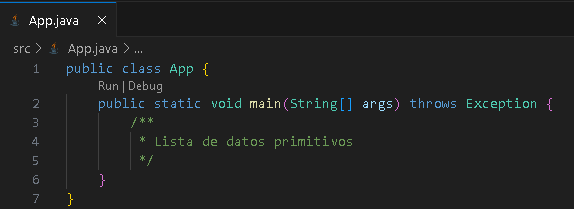
Se nos aparece el señor, bueno no, se aparece algo como esto:



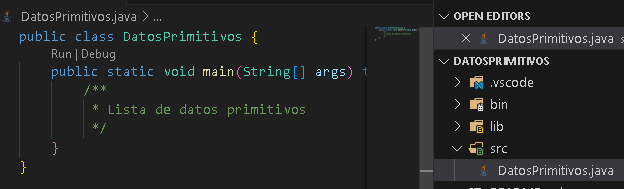
¿Dónde leñes se editan las movidas cojón? ¡Qué llevo leídas 10 páginas ya!

Empieza por aquí jefe crack titán.



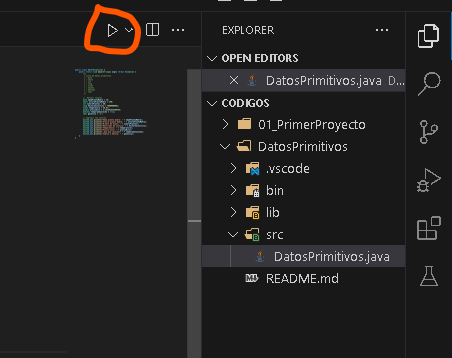


Si renombras el fichero App, por el nombre que te salga de los mismos, y lo cambias en el código también, podrás renombrar a tu gusto.



* 1. Ejecutar código

Para ejecutar nuestro proyecto debemos hacer clic en el icono del “play” de arriba a la izquierda:

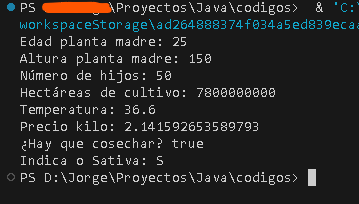


1. Variables
   1. Datos primitivos

Hay que saber que Java, al ser de tipado estático (moviolas que ya veremos) nos obliga a indicarle el tipo de dato de la variable antes de definir ésta (si, con tilde porque soy un rebelde).



Una vez ejecutemos nos aparecerá lo siguiente por consola:



* 1. Tipos de datos primitivos

**Byte:** Es un tipo de dato entero de 8 bits con signo, capaz de almacenar valores numéricos desde -128 hasta 127, incluidos ambos extremos.

**Short:** Es un tipo de dato entero de 16 bits con signo que puede almacenar valores numéricos que oscilan entre -32,768 y 32,767.

**Int:** Es un tipo de dato entero de 32 bits con signo, utilizado para almacenar números. Su rango va desde -2³¹ hasta 2³¹-1.

**Long:** Se trata de un tipo de dato entero de 64 bits con signo, que permite almacenar números entre -2⁶³ y 2⁶³-1.

**Float:** Es un tipo de dato de 32 bits para almacenar números en coma flotante con precisión simple.

**Double:** Este tipo de dato de 64 bits se utiliza para almacenar números en coma flotante con doble precisión.

**Boolean:** Sirve para definir datos booleanos, que solo pueden tener dos valores: true o false. Utiliza 1 bit de espacio.

Char: Es un tipo de dato que representa un carácter Unicode de 16 bits.

* 1. Valores por defecto

|  |  |
| --- | --- |
| Byte | 0 |
| Short | 0 |
| Int | 0 |
| Long | 0L |
| float | 0.0f |
| Double | 0.0d |
| Char | ‘u0000’ |
| String (o cualquier objeto) | Null |
| boolean | False |

* 1. Tipos de variables
* **Variables locales:** Se declaran dentro de un método y solo son accesibles dentro de este método
* **Variables de instancia:** Se declaran dentro de una clase, pero fuera de los métodos. Cada objeto de la clase tendrá su propia copia
* **Variables de clase (estáticas):** Se declaran con la palabra *static.* Pertenecen a la clase y no a instancias (objetos) de la clase
  1. Constantes

Una constante es una manera de llamar a una variable (Si algún profesor lee eso me cruje) que no cambia de valor NUNCA durante la ejecución del programa.

Se declaran con la palabra *final* y el nombre de la constante (Convencionalmente se usan letras mayúsculas separadas por barra baja)

* 1. Ejemplo completo



Explicación

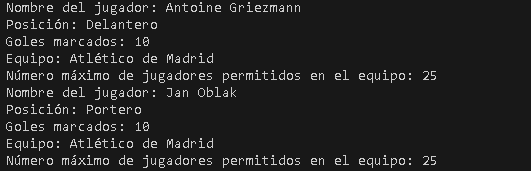
**Variables de clase (estáticas):** *nombreDelEquipo* es una variable estática. Todos los objetos creados de la clase ***AtleticoDeMadrid*** compartirán este valor, ya que todos los jugadores forman parte del mismo equipo.

**Variables de instancia:** *nombreDelJugador* y *posicion* son variables de instancia. Cada jugador tendrá su propio nombre y posición, por lo que cada objeto (que representa a un jugador) tendrá diferentes valores para estas variables.

**Variables locales:** La variable *numeroDeGoles* es una variable local. Está definida dentro del método ***mostrarInformacionDelJugador*** y representa la cantidad de goles que un jugador ha marcado. Solo existe dentro del método y no es accesible fuera de él.

**Constantes:** La constante ***MAX\_JUGADORES*** indica el número máximo de jugadores permitidos en el equipo de fútbol, que es 25. Este valor no puede cambiar y es igual para todos los objetos de la clase.

Salida del código



1. Operadores

Los operadores son generalmente las personas que realizan una operación. Pero como esto es Java y no ciencias de la salud pues hay que dar otra explicación.

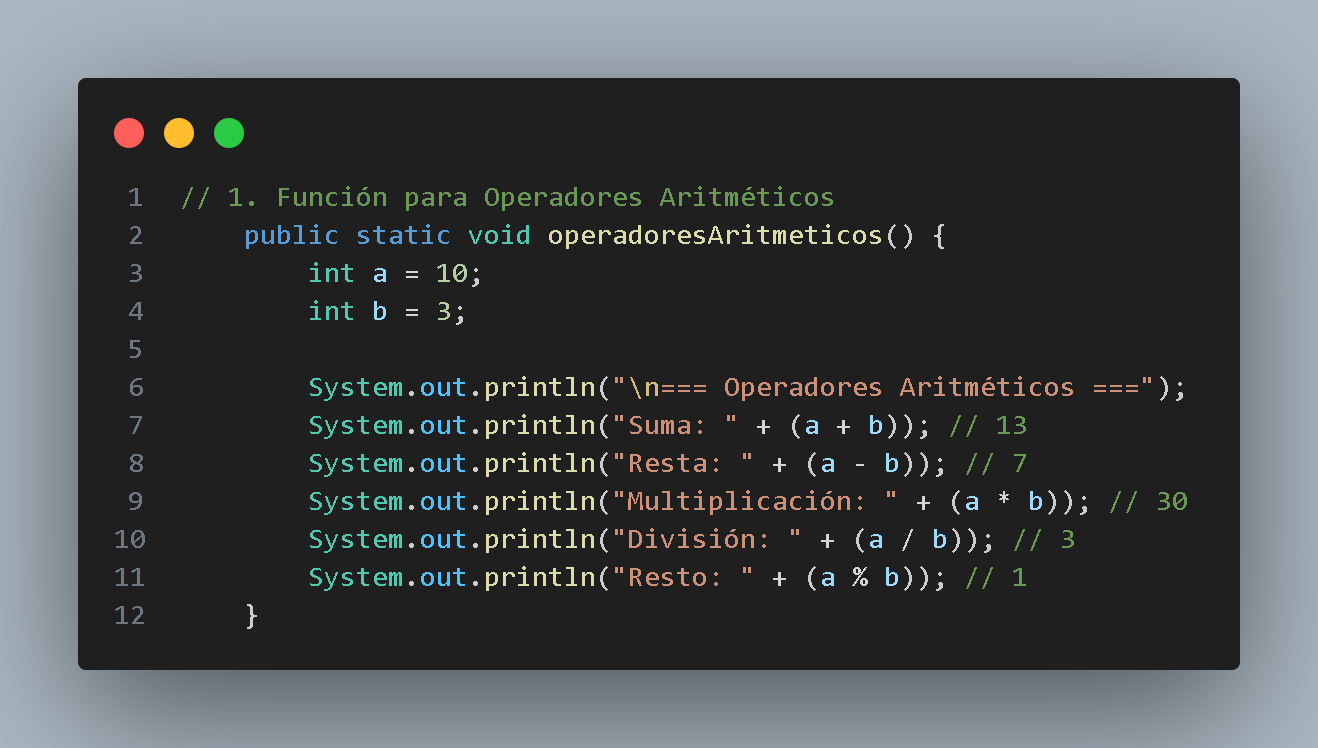
Los operadores en Java son símbolos que nos ayudan a realizar operaciones sobre 1,2 o las variables que sean, también pueden afectar a los valores.

* 1. Operadores aritméticos

Estos operadores (simbolos) nos permiten realizar operaciones matemáticas como la suma, resta, multiplicación, división y resto.

|  |  |  |
| --- | --- | --- |
| Operador | Operación | Ejemplo |
| + | Suma | A + b |
| - | Resta | A – b |
| \* | Multiplicación | A \* b |
| / | División | A / b |
| % | Resto de la división | A % b |

Ejemplo



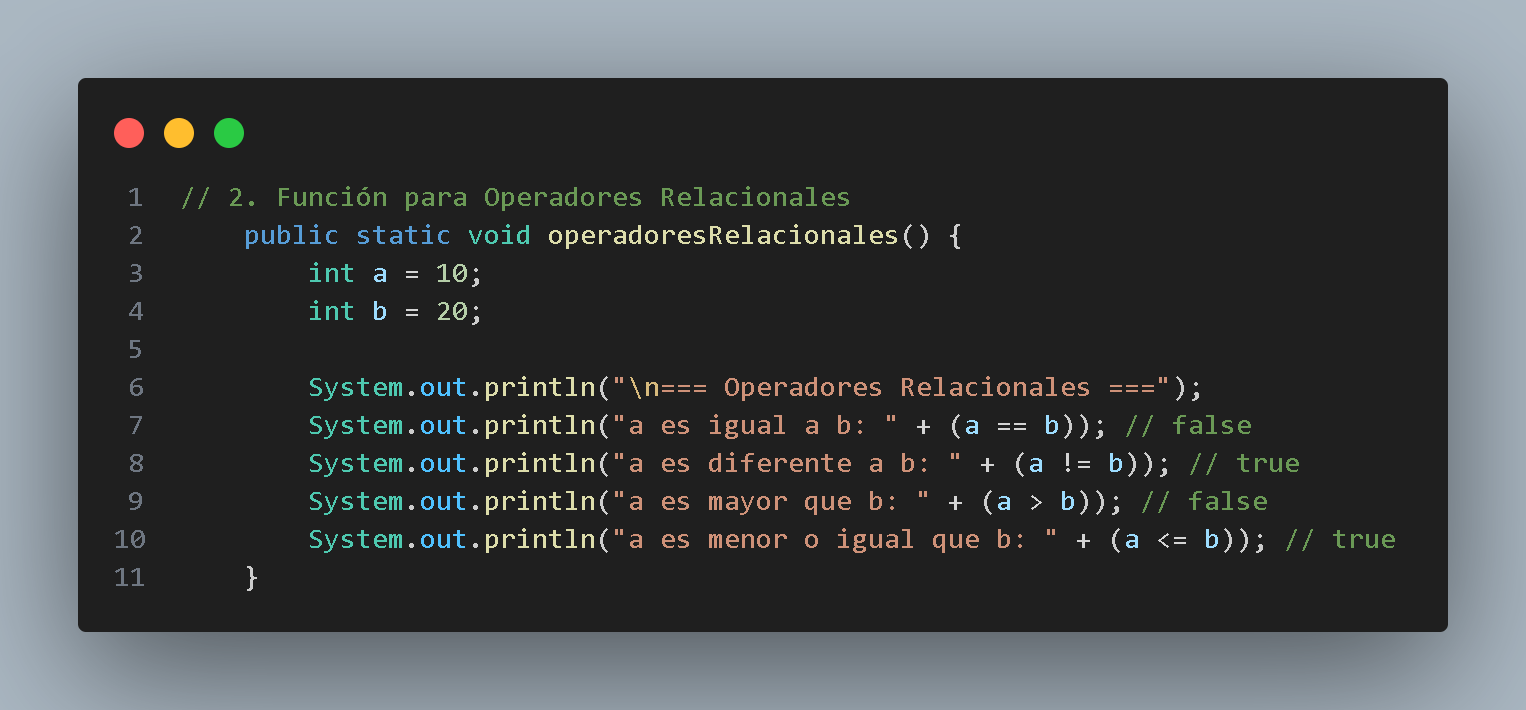
Importante recordar que sirven para variables (ejemplo de arriba) como para valores (por ejemplo: 3 + 2 ) te la hinco.

* 1. Operadores relacionales

Los operadores de igualdad en Java nos sirven para comparar valores y/o variables entre ellas. Imaginate que quieres saber quien tiene menos ganas de vivir en tu grupo de amigos, con estos operadores podrías saberlo.

|  |  |  |
| --- | --- | --- |
| Operador | Operación | Ejemplo |
| == | Igual a | A == b |
| != | NO es igual a | A != b |
| > | Mayor que | A > b |
| < | Menor que | A < b |
| >= | Mayor o Igual que | A >= b |
| <= | Menor o igual que | A <= b |

Ejemplo



* 1. Operadores lógicos

Los operadores lógicos en Java se usan para combinar expresiones booleanas