# MANUAL JAVA PRINCIPIANTES

Java™

Jorge Moya Delgado

# Indice

# 1. Java desde cero

## 1.1 ¿Qué es Java?

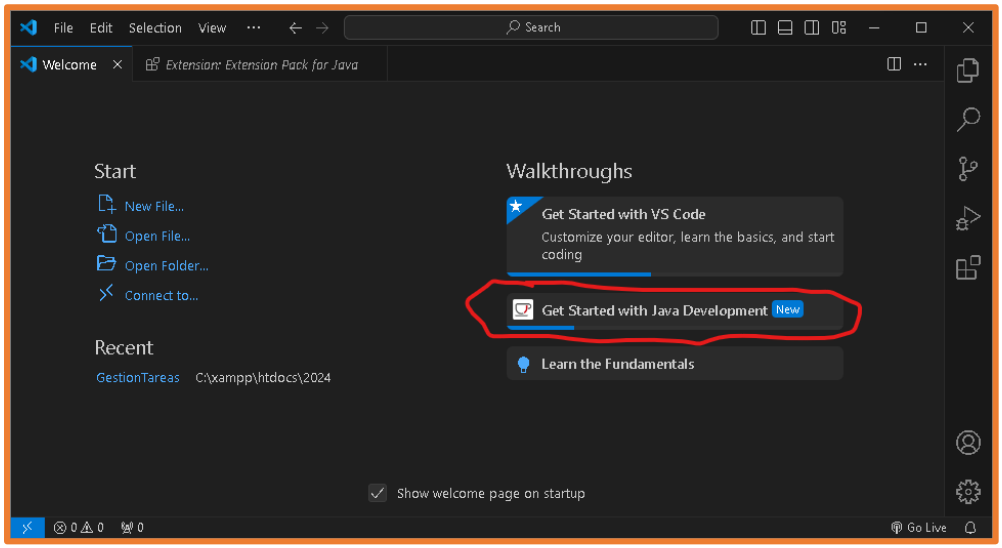
LENGUAJE DE OBJETOS
JAVA CON VSCODE

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

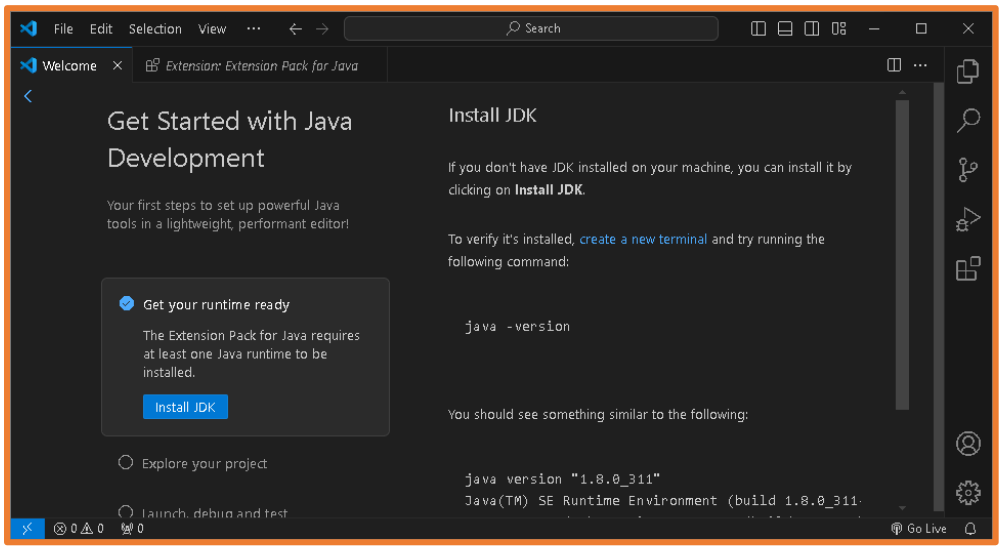
Extension Pack for Java:

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

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

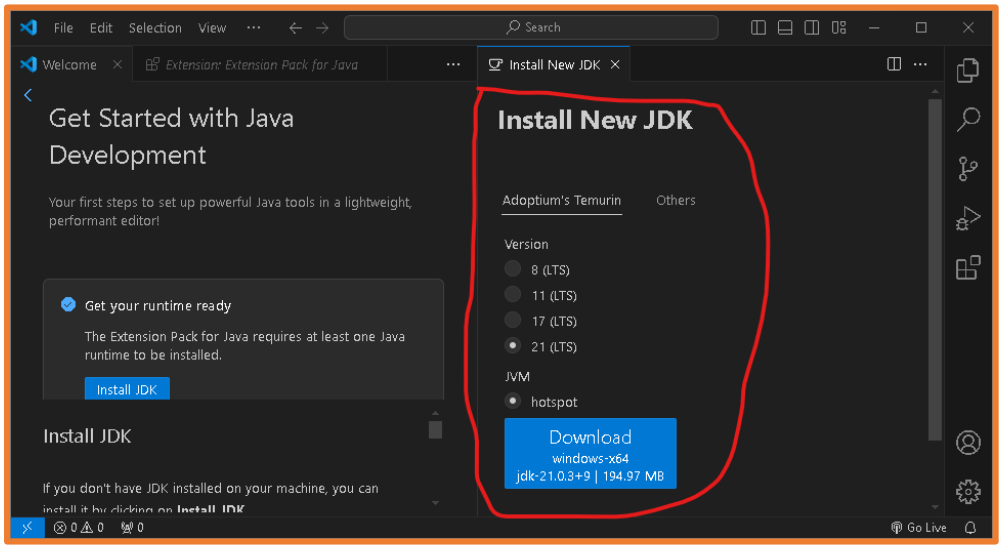



Seleccionamos la opción marcada. Y se nos abrirá la siguiente pestaña:



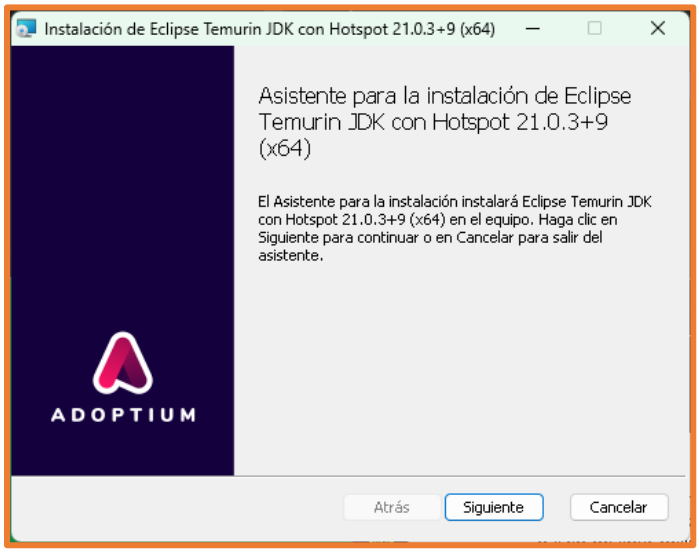

Seleccionamos ***Install JDK***.

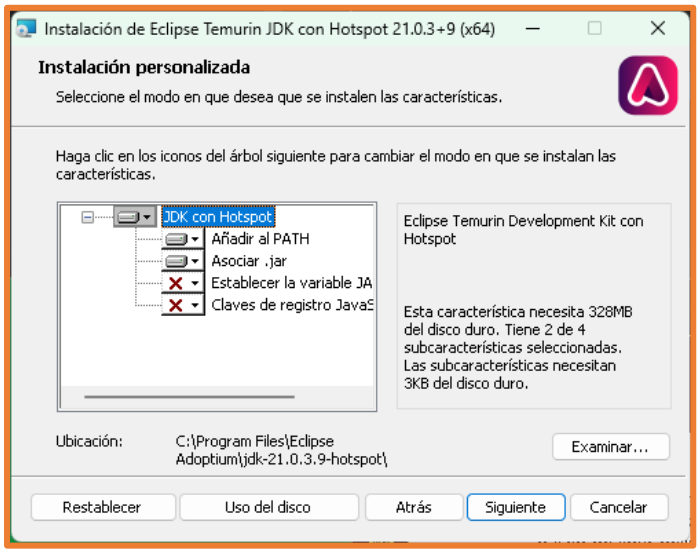
Se abrirá una 2º ventana, donde podremos elegir la versión y otras configuraciones a tu gusto.



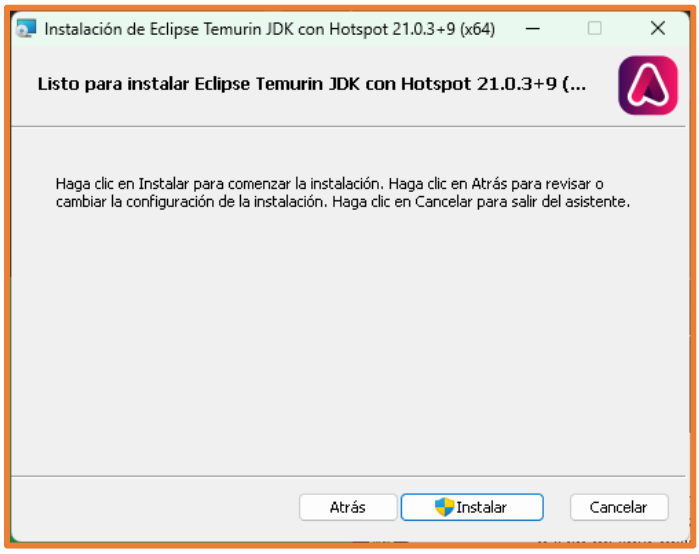

Descargamos el instalador


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

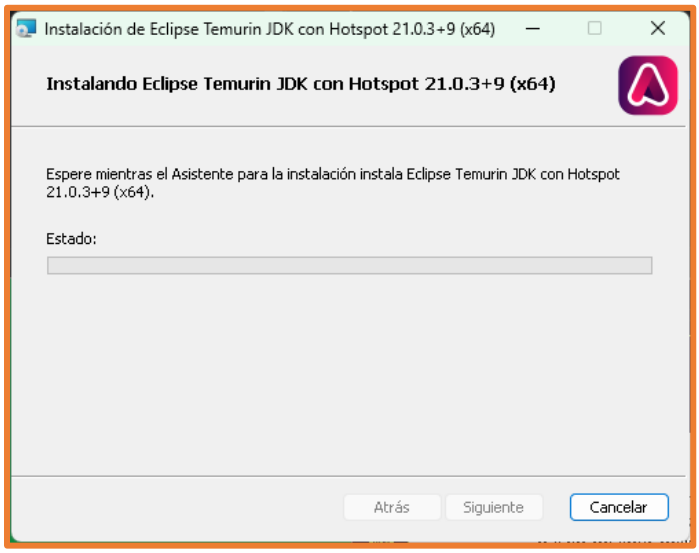


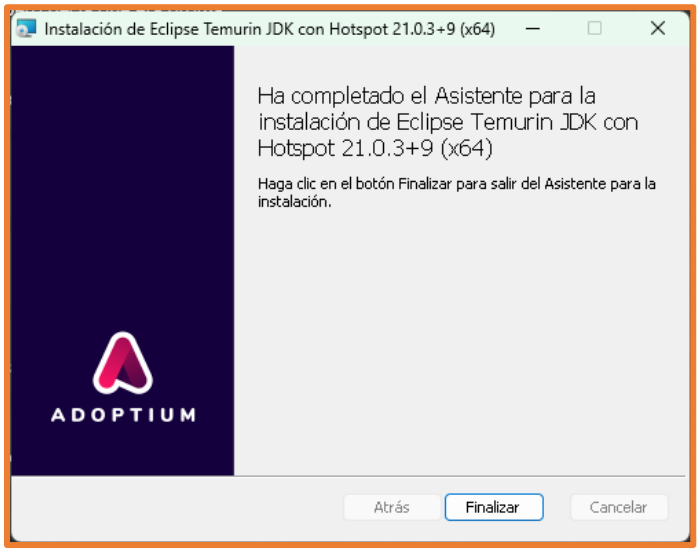



Ahora le damos a instalar:




Esperamos a que finalice la instalación.


Instalación de Eclipse Temurin JDK con Hotspot 21.0.3+9 (x64)
Instalando Eclipse Temurin JDK con Hotspot 21.0.3+9 (x64)
Espere mientras el Asistente para la instalación instala Eclipse Temurin JDK con Hotspot 21.0.3+9 (x64).
Estado:
Atrás
Siguiente
Cancelar


Instalación de Eclipse Temurin JDK con Hotspot 21.0.3+9 (x64)
ADOPTIUM
Ha completado el Asistente para la instalación de Eclipse Temurin JDK con Hotspot 21.0.3+9 (x64)
Haga clic en el botón Finalizar para salir del Asistente para la instalación.
Atrás
Finalizar
Cancelar

# 2. Primeras cosillas con java

## 2.1 Estructura de un programa en Java

```java
// 1. Una sentencia de paquete (package).
// package com.ejemplo.miapp;

// 2. Una o varias sentencias de importación (import).
// import java.util.Date;
```

```java
// 3. Declaraciones de clases privadas deseadas.
class ClasePrivada {
    private String atributo;

    public ClasePrivada(String atributo) {
        this.atributo = atributo;
    }

    public String getAtributo() {
        return atributo;
    }

    public void setAtributo(String atributo) {
        this.atributo = atributo;
    }
}
```

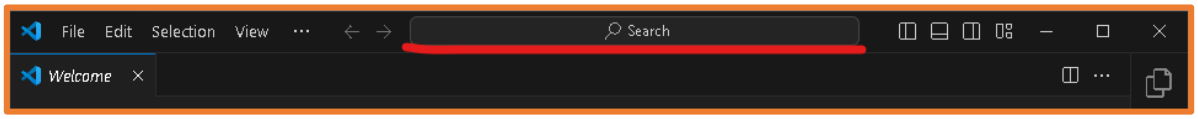
```java
// 4. Una declaración de clase pública.
public class Ejemplo {
    // Declaraciones de variables de la clase (estáticas).
    public static final String CONSTANTE = "Constante de la clase";

    // Declaraciones de variables de instancia.
    private String nombre;
    private int edad;

    // Definiciones de constructores.
    public Ejemplo(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    // Definiciones de métodos.
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }

    // Comentarios en cualquier parte del programa.
    public void mostrarInfo() {
        // Imprime la información del objeto
        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
        System.out.println("Constante: " + CONSTANTE);
    }

    // Método main para ejecutar el programa
    public static void main(String[] args) {
        Ejemplo persona = new Ejemplo("Juan", 25);
        persona.mostrarInfo();

        ClasePrivada cp = new ClasePrivada("Valor privado");
        System.out.println("Atributo de ClasePrivada: " + cp.getAtributo());
    }
}

```

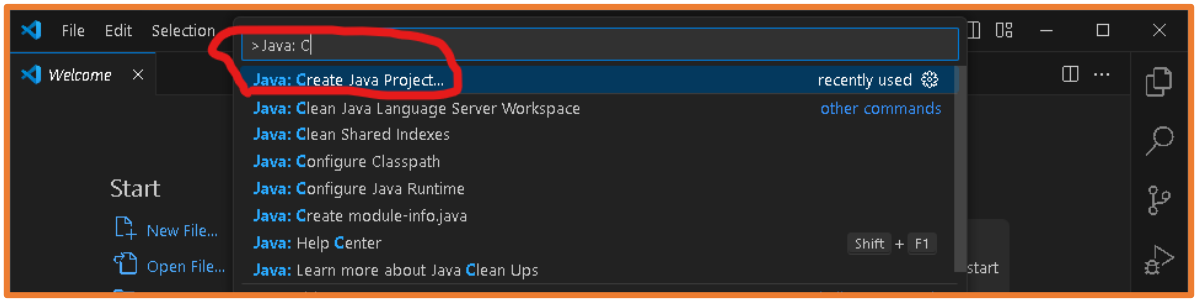
Resumiendo.

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
    a. Variables de clase
    b. Variables de instancia
    c. Definición de constructores
    d. Definiciones de métodos
    e. Comentarios por to' los laos'

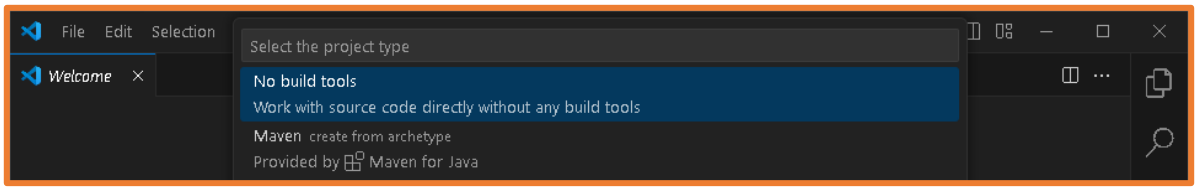
## 2.2 Crear código en VSCode

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



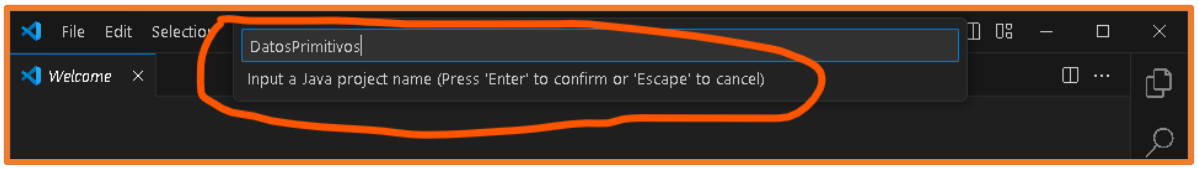




Seleccionamos No build tools:

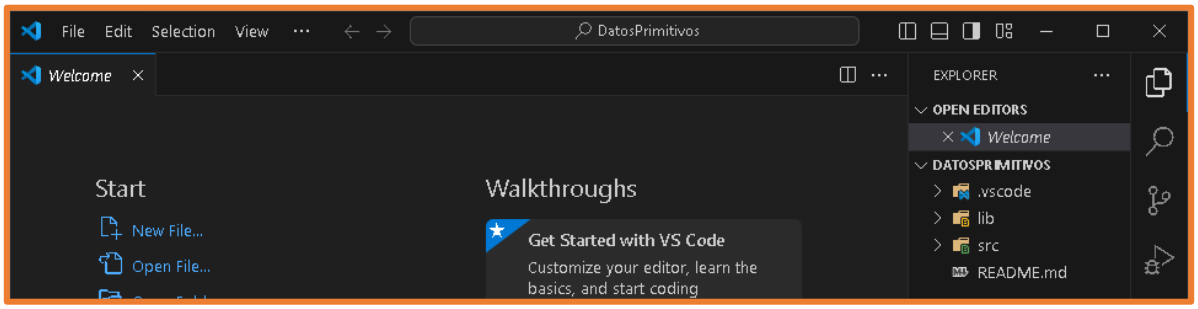



Elegimos nuestra carpeta de proyectos:


Le damos nombre a nuestro proyecto.

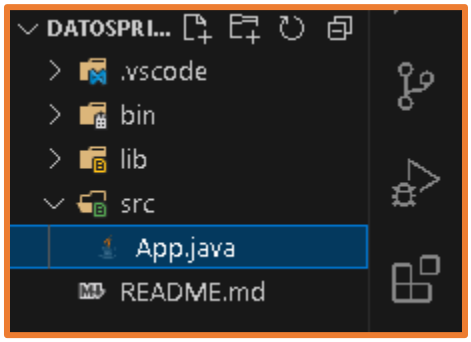



Se nos aparece el señor, bueno no, se aparece algo como esto:



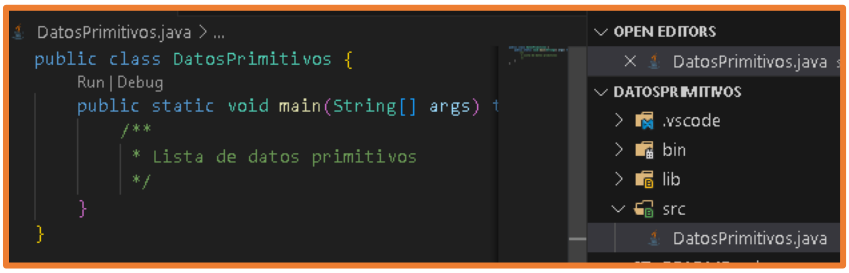

¿Dónde leñes se editan las movidas cojón? ¡Qué llevo leídas 10 páginas ya!

Empieza por aquí jefe crack titán.



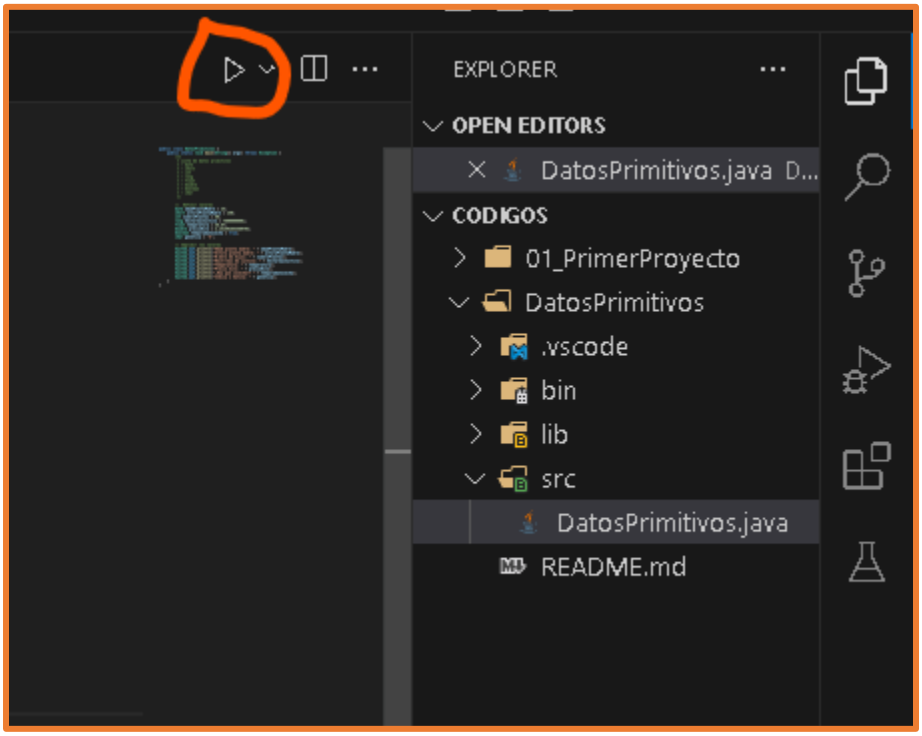


Si renombras el fichero App, por el nombre que te salga de los mismos, y lo cambias en el código también, podrás renombrar a tu gusto.




## 2.3 Ejecutar código

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



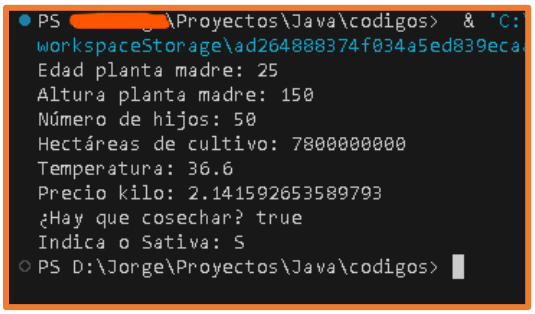
# 3. Variables

## 3.1 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).

```java
public static void main(String[] args) throws Exception {
        /**
         * Lista de datos primitivos
         * - byte
         * - short
         * - int
         * - long
         * - float
         * - double
         * - boolean
         * - char
         */

        //  Definir valores
        byte edadPlantaMadre = 25;
        short alturaPlantaMadre = 150;
        int numeroPlantas = 50;
        long hectareasCultivo = 7800000000L;
        float temperatura = 36.6f;
        double precioKilo = 2.141592653589793;
        boolean temporadaCosecha = true;
        char genetica = 'S';

        // Imprimir los valores
        System.out.println("Edad planta madre: " + edadPlantaMadre);
        System.out.println("Altura planta madre: " + alturaPlantaMadre);
        System.out.println("Número de hijos: " + numeroPlantas);
        System.out.println("Hectáreas de cultivo: " + hectareasCultivo);
        System.out.println("Temperatura: " + temperatura);
        System.out.println("Precio kilo: " + precioKilo);
        System.out.println("¿Hay que cosechar? " + temporadaCosecha);
        System.out.println("Indica o Sativa: " + genetica);
    }
```

Una vez ejecutemos nos aparecerá lo siguiente por consola:




## 3.2 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 $-2^{31}$ hasta $2^{31}-1$.

**Long:** Se trata de un tipo de dato entero de 64 bits con signo, que permite almacenar números entre $-2^{63}$ y $2^{63}-1$.

**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.

## 3.3 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 |

## 3.4 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

## 3.5 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)

## 3.6 Ejemplo completo

```java
public class AtleticoDeMadrid {

    // 1. Variable de clase (estática)
    // Esta variable pertenece a la clase y es compartida por todos los objetos de la clase.
    static String nombreDelEquipo = "Atlético de Madrid";

    // 2. Variable de instancia
    // Cada objeto tendrá un nombre de jugador único y la posición del jugador.
    String nombreDelJugador;
    String posicion;

    // 3. Constante
    // Esta constante indica el número máximo de jugadores permitidos en un equipo de fútbol.
    final int MAX_JUGADORES = 25;

    // Constructor para inicializar la variable de instancia
    public AtleticoDeMadrid(String nombre, String posicion) {
        this.nombreDelJugador = nombre; // Inicializando el nombre del jugador
        this.posicion = posicion; // Inicializando la posición del jugador
    }

    // Método que demuestra el uso de variables locales
    public void mostrarInformacionDelJugador() {
        // 4. Variable local
        // Esta variable solo es accesible dentro de este método.
        int numeroDeGoles = 10; // Supongamos que el jugador ha marcado 10 goles

        System.out.println("Nombre del jugador: " + nombreDelJugador); // Variable de instancia
        System.out.println("Posición: " + posicion); // Variable de instancia
        System.out.println("Goles marcados: " + numeroDeGoles); // Variable local
        System.out.println("Equipo: " + nombreDelEquipo); // Variable de clase
        System.out.println("Número máximo de jugadores permitidos en el equipo: " + MAX_JUGADORES); // Constante
    }

    public static void main(String[] args) {
        // Creación de un objeto de la clase AtleticoDeMadrid representando a un jugador
        AtleticoDeMadrid jugador1 = new AtleticoDeMadrid("Antoine Griezmann", "Delantero");

        // Llamada al método para mostrar la información del jugador
        jugador1.mostrarInformacionDelJugador();

        // Creación de otro objeto representando a un segundo jugador
        AtleticoDeMadrid jugador2 = new AtleticoDeMadrid("Jan Oblak", "Portero");
        jugador2.mostrarInformacionDelJugador();
    }
}
```

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

```
Nombre del jugador: Antoine Griezmann
Posición: Delantero
Goles marcados: 10
Equipo: Atlético de Madrid
Número máximo de jugadores permitidos en el equipo: 25
Nombre del jugador: Jan Oblak
Posición: Portero
Goles marcados: 10
Equipo: Atlético de Madrid
Número máximo de jugadores permitidos en el equipo: 25
```

# 4. 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.

## 4.1 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

```java
// 1. Función para Operadores Aritméticos
    public static void operadoresAritmeticos() {
        int a = 10;
        int b = 3;

        System.out.println("\n=== Operadores Aritméticos ===");
        System.out.println("Suma: " + (a + b)); // 13
        System.out.println("Resta: " + (a - b)); // 7
        System.out.println("Multiplicación: " + (a * b)); // 30
        System.out.println("División: " + (a / b)); // 3
        System.out.println("Resto: " + (a % b)); // 1
    }
```

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

## 4.2 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

```java
// 2. Función para Operadores Relacionales
    public static void operadoresRelacionales() {
        int a = 10;
        int b = 20;

        System.out.println("\n=== Operadores Relacionales ===");
        System.out.println("a es igual a b: " + (a == b)); // false
        System.out.println("a es diferente a b: " + (a != b)); // true
        System.out.println("a es mayor que b: " + (a > b)); // false
        System.out.println("a es menor o igual que b: " + (a <= b)); // true
    }
```

## 4.3 Operadores lógicos

Los operadores lógicos en Java se usan para combinar expresiones booleanas.

| Operador | Operación | Ejemplo |
|---|---|---|
| && | AND | (3 < 5) && (1 > 6) |
| \|\| | OR | (2 > 1) \|\| (2 > 13) |
| ! | NOT | !(3 > 2) |
| ? | Ternario | (expresion)?valor_true:valor_false; |

En el caso del operador lógico AND el resultado será **true** ***siempre y cuando las dos expresiones evaluadas sean*** **true**. **Si una de las expresiones es false el resultado de la expresión condicional AND será false.**

Para el operador lógico OR el resultado será **true** ***siempre que alguna de las dos expresiones sea true.***

El operador lógico NOT es algo duro de entender al principio, digamos que es una negación. En este caso el resultado del paréntesis (3 > 2) devolvería ***true*** pero como tenemos un **!** delante, la salida seria ***false***. Se le coge cariño, lo juro.

Vale a estas alturas de la película no nos vamos a preocupar mucho por el tema del ternario, ya llegaremos con las estructuras de control.

EJEMPLO

```java
// 3. Función para Operadores Lógicos
    public static void operadoresLogicos() {
        boolean x = true;
        boolean y = false;

        System.out.println("\n=== Operadores Lógicos ===");
        System.out.println("x && y: " + (x && y)); // false
        System.out.println("x || y: " + (x || y)); // true
        System.out.println("!x: " + (!x)); // false
        System.out.println("!(3 > 2): " + !(3 > 2)); // false
    }
```

## 4.4 Operadores de asignación

Realmente estos operadores combinan los operadores aritméticos con el operador de asignación ( **=** )

| Operador | Operación | Ejemplo |
|---|---|---|
| = | Asignación | A = 5 |
| += | Asignación de suma | A += 5 (a = a + 5) |
| -= | Asignación de resta | A -= 5 (a = a - 5) |
| *= | Asignación de multiplicación | A *= 5 |
| /= | Asignación de división | A /= 5 |
| %= | Asignación de resto | A %= 5 |

EJEMPLO

```java
// 4. Función para Operadores de Asignación
    public static void operadoresAsignacion() {
        int a = 10;
        a += 5; // a = a + 5
        int b = 10;
        b -= 5;

        System.out.println("\n=== Operadores de Asignación ===");
        System.out.println("Nuevo valor de a: " + a); // 15
        System.out.println("Nuevo valor de b: " + b); // 5
    }
```

## 4.5 Operadores unitarios

| Operador | Operación | Ejemplo |
|---|---|---|
| + | Valor positivo | +a |
| - | Valor negativo | -a |
| ++ | Incremento | A++ o ++a |
| -- | Decremento | a—o --a |

EJEMPLO

```java
// 5. Función para Operadores Unarios
    public static void operadoresUnarios() {
        int a = 5;

        System.out.println("\n=== Operadores Unarios ===");
        System.out.println("a: " + a); // 5
        System.out.println("a: " + -a); // -5
        System.out.println("a++: " + a++); // 5 (usa a y luego incrementa)
        System.out.println("++a: " + ++a); // 7 (incrementa antes de usar)
        System.out.println("a--: " + a--); // 7 (usa a y luego decrementa)
        System.out.println("--a: " + --a); // 5 (decrementa antes de usar)
    }
```