INTRODUCCIÓN A LA PROGRAMACIÓN EN JAVA

Programación Orientada a Objetos POO

- La programación orientada a objetos (POO) es un paradigma de programación que organiza el código en torno a objetos, los cuales son entidades que combinan datos y funciones relacionadas en una única estructura.

- La POO surge con el objetivo de mejorar la forma en que se desarrollan y mantienen los programas. Antes de su adopción, los programas se escribían de manera procedimental, dividiendo el código en rutinas o funciones que procesaban los datos de forma secuencial. Sin embargo, este enfoque resultaba limitado en la gestión de la complejidad de los programas más grandes y dificultaba la reutilización del código.

- La POO proporciona una forma más estructurada y modular de escribir programas. Al organizar la lógica en objetos autónomos y encapsulados, se fomenta la reutilización de código y se facilita el mantenimiento.

El Lenguaje Java

Java es un lenguaje de programación orientado a objetos que produce software para múltiples plataformas. Cuando un programador escribe una aplicación Java, el código compilado (conocido como código de bytes) se ejecuta en la mayoría de los sistemas operativos (SO), incluidos Windows, Linux y Mac OS. Java deriva gran parte de su sintaxis de los lenguajes de programación C y C++.

Fuente: https://www.tuexpertoit.com/tag/sun/

- Desarrollado originalmente por James Gosling de Sun Microsystems.
- Creado en 1995.
- Sintaxis derivada de C y C++.
- Plataforma universal para ser ejecutada en cualquier plataforma.

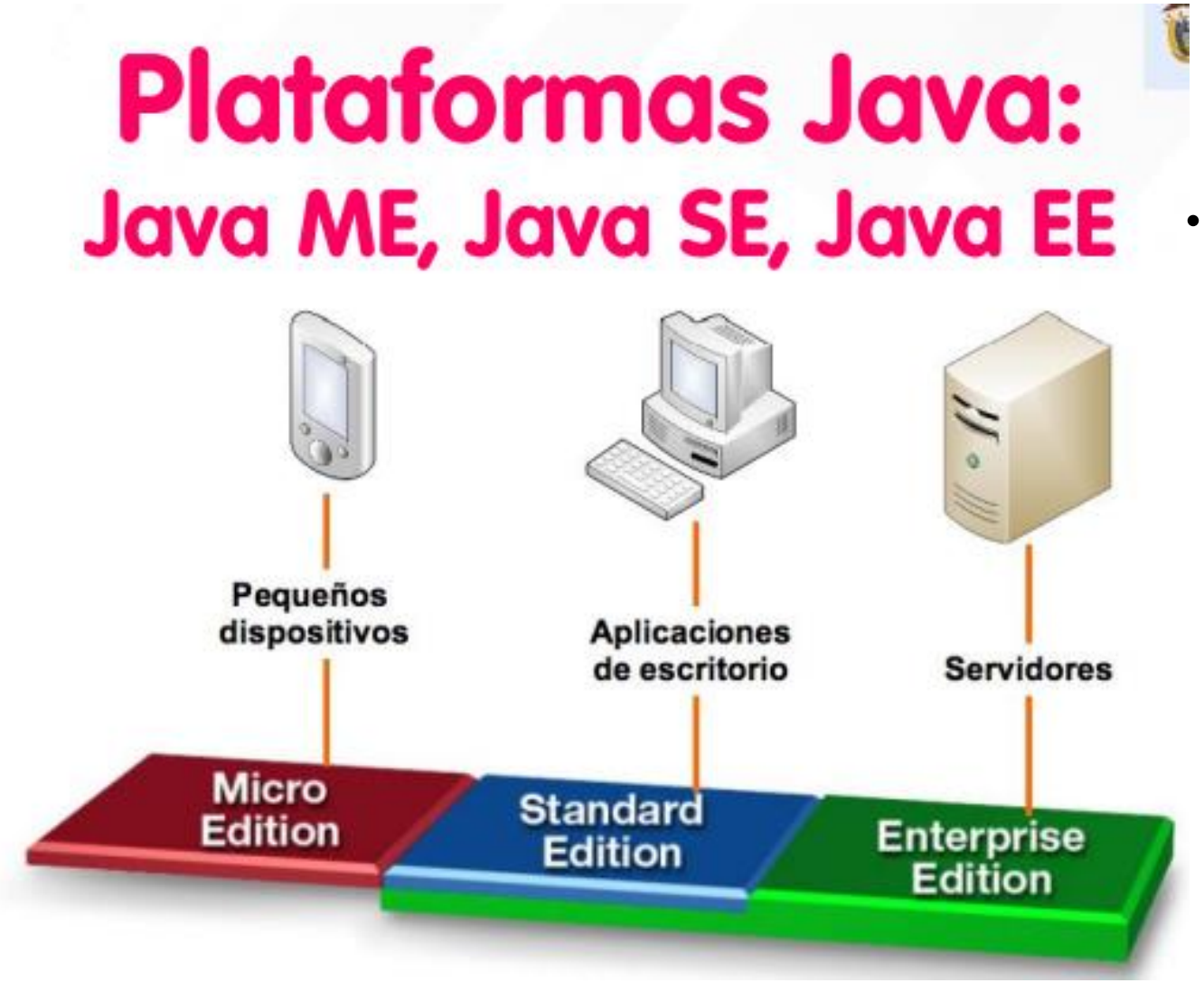
Fuente: https://picodotdev.github.io/

Versiones de Java

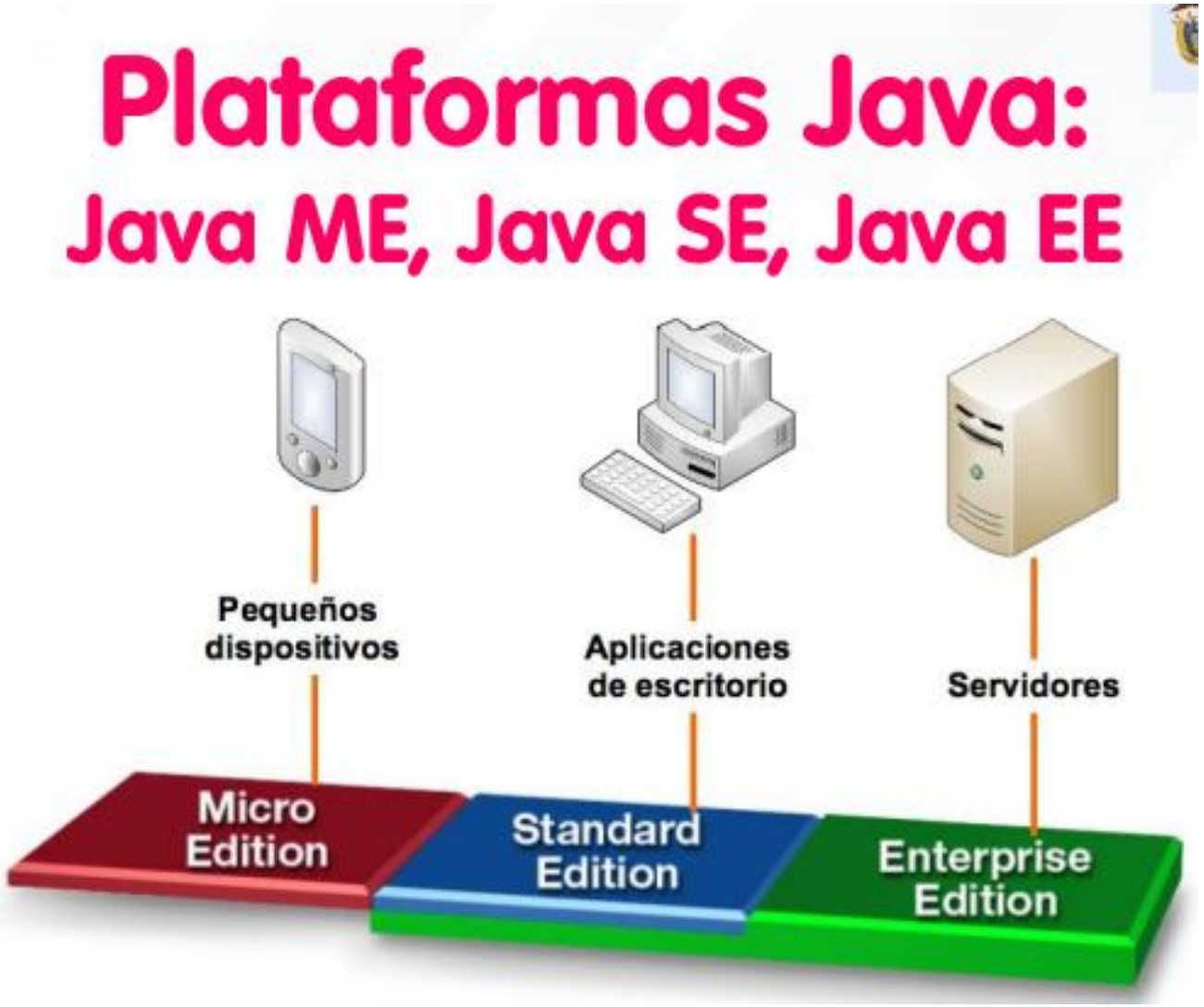
- Las versiones **LTS** (Long-Term Support) Soporte a largo plazo, son versiones especiales que reciben un período de soporte extendido. Estas versiones están diseñadas para ofrecer estabilidad y soporte a largo plazo, especialmente para aplicaciones empresariales y críticas..

Versión Java	Fecha de desarrollo	Soporte hasta
Java 1.0	Enero de 1996	Sin soporte
Java 1.1	Febrero de 1997	Sin soporte
Java 1.2	Diciembre de 1998	Sin soporte
Java 1.3	Mayo de 2000	Sin soporte
Java 1.4	Febrero de 2002	Sin soporte
Java 5 (J2SE 5.0)	Septiembre de 2004	Octubre de 2009
Java 6 (Java SE 6)	Diciembre de 2006	Diciembre de 2018
Java 7 (Java SE 7)	Julio de 2011	Julio de 2022
Java 8 (Java SE 8)	Marzo de 2014	Diciembre de 2030
Java 9 (Java SE 9)	Septiembre de 2017	Marzo de 2018
Java 10 (Java SE 10)	Marzo de 2018	Septiembre de 2018
Java 11 (Java SE 11)	Septiembre de 2018	Septiembre de 2023
Java 12 (Java SE 12)	Marzo de 2019	Septiembre de 2019
Java 13 (Java SE 13)	Septiembre de 2019	Marzo de 2020
Java 14 (Java SE 14)	Marzo de 2020	Septiembre de 2020
Java 15 (Java SE 15)	Septiembre de 2020	Marzo de 2021
Java 16 (Java SE 16)	Marzo de 2021	Septiembre de 2021
Java 17 (Java SE 17)	Septiembre de 2021	Septiembre de 2026

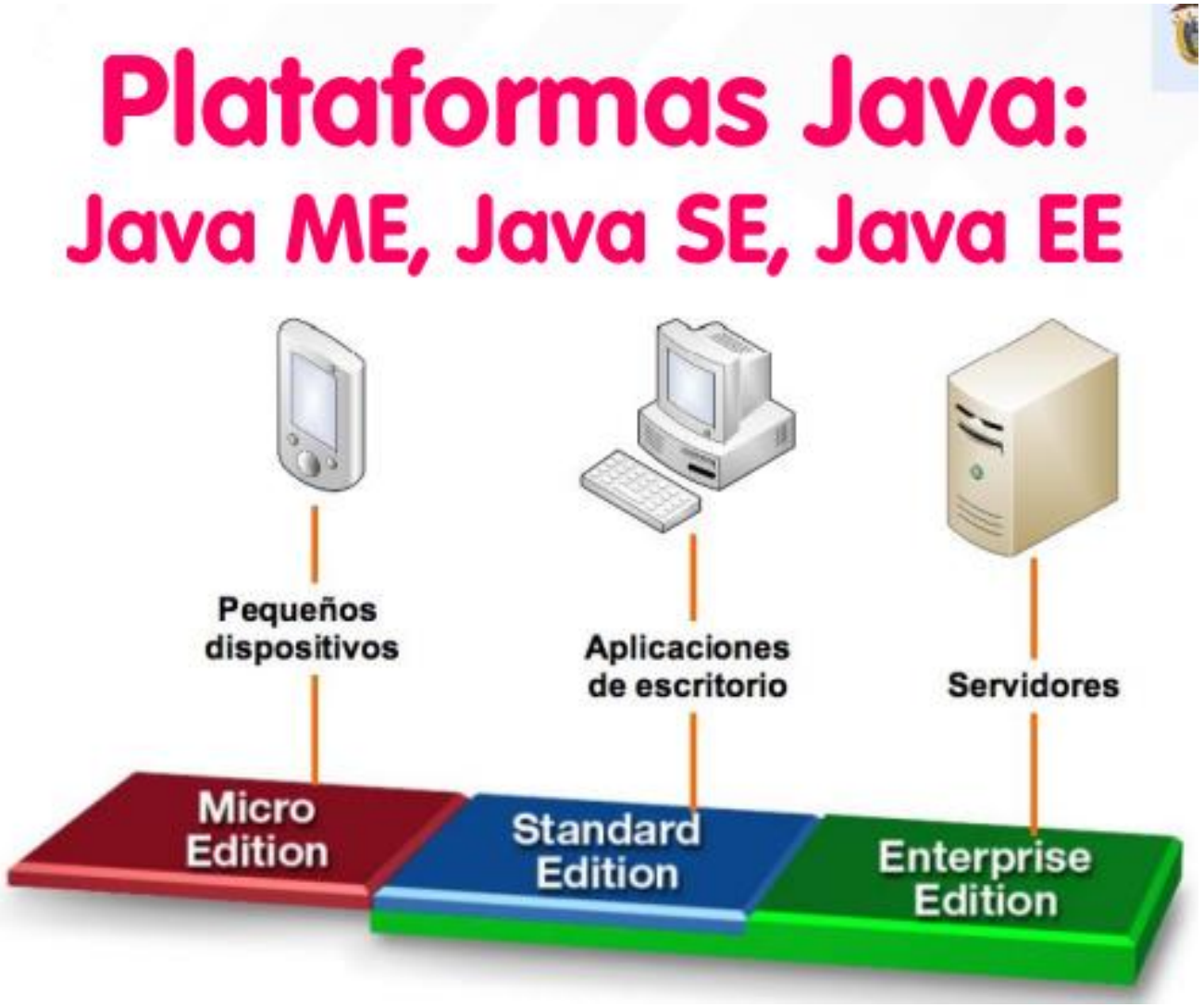
https://en.wikipedia.org/wiki/Java_version_history



- **Java ME** se utiliza principalmente en dispositivos móviles y embebidos, como teléfonos móviles, PDA (Asistentes Personales Digitales) y otros dispositivos con recursos limitados. Esta edición proporciona una plataforma de desarrollo simplificada y optimizada para dispositivos con capacidad de procesamiento y memoria limitados.



- **Java SE** es la edición estándar de Java y es la más comúnmente utilizada. Proporciona un conjunto completo de bibliotecas y herramientas para desarrollar aplicaciones de escritorio, aplicaciones de consola y aplicaciones basadas en servidor. Java SE es conocido por su portabilidad, lo que significa que una vez que se escribe el código en Java, puede ejecutarse en diferentes plataformas sin necesidad de reescribirlo.

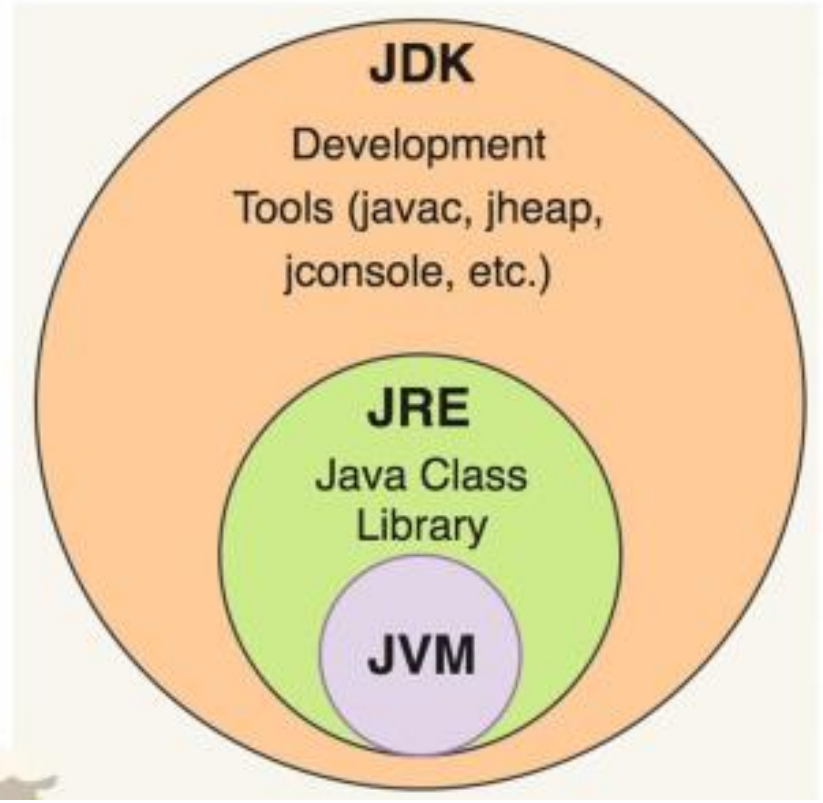


- **Java EE** es la edición empresarial de Java y está diseñada para aplicaciones de gran escala y empresariales. Proporciona un conjunto de extensiones y API (Interfaz de Programación de Aplicaciones) para el desarrollo de aplicaciones web, servicios web, aplicaciones distribuidas y otras soluciones empresariales. Java EE es utilizado en entornos corporativos para crear sistemas complejos y escalables.

JDK (Java Development Kit) de Java

Es un conjunto de herramientas y utilidades que se utiliza para desarrollar, compilar y depurar aplicaciones en el lenguaje de programación Java. JDK proporciona todo lo necesario para desarrollar software Java, incluyendo un compilador, bibliotecas, documentación y otras herramientas relacionadas.

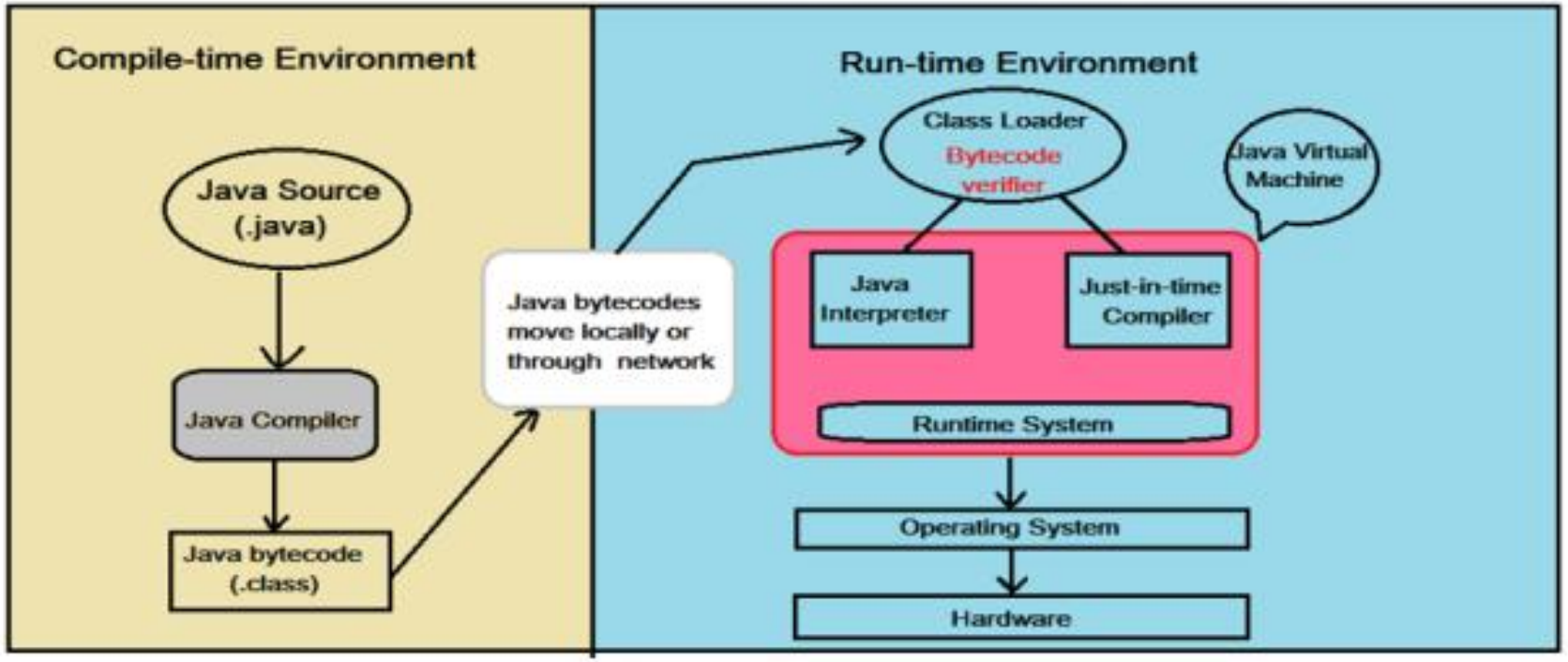
- **Java Virtual Machine (JVM)**
  - Es una abstracción de una máquina de cómputo, encargada de ejecutar los programas Java.
- **Java Runtime Environment (JRE)**
  - Es un paquete de software que contiene los artefactos requeridos para ejecutar un programa Java.
- **Java Development Kit (JDK)**
  - Es un superconjunto del JRE y contiene las herramientas para los programadores Java.



Java Compilado o Interpretado

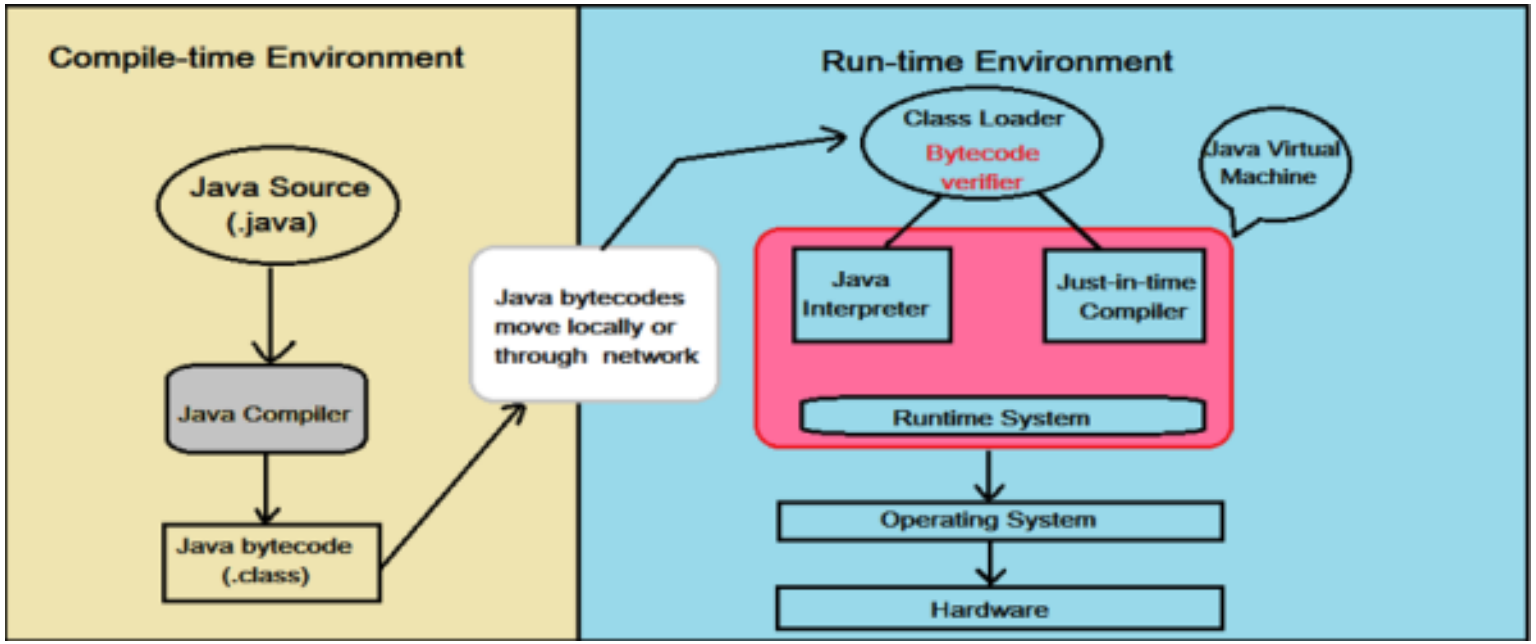
Java es un lenguaje de programación que utiliza una combinación de compilación y ejecución interpretada.

- Compilación: El código fuente de Java se compila utilizando el compilador de Java (javac) del JDK. El compilador traduce el código fuente escrito en Java a bytecode, que es un lenguaje de bajo nivel y plataforma independiente. El bytecode se guarda en archivos con extensión .class.

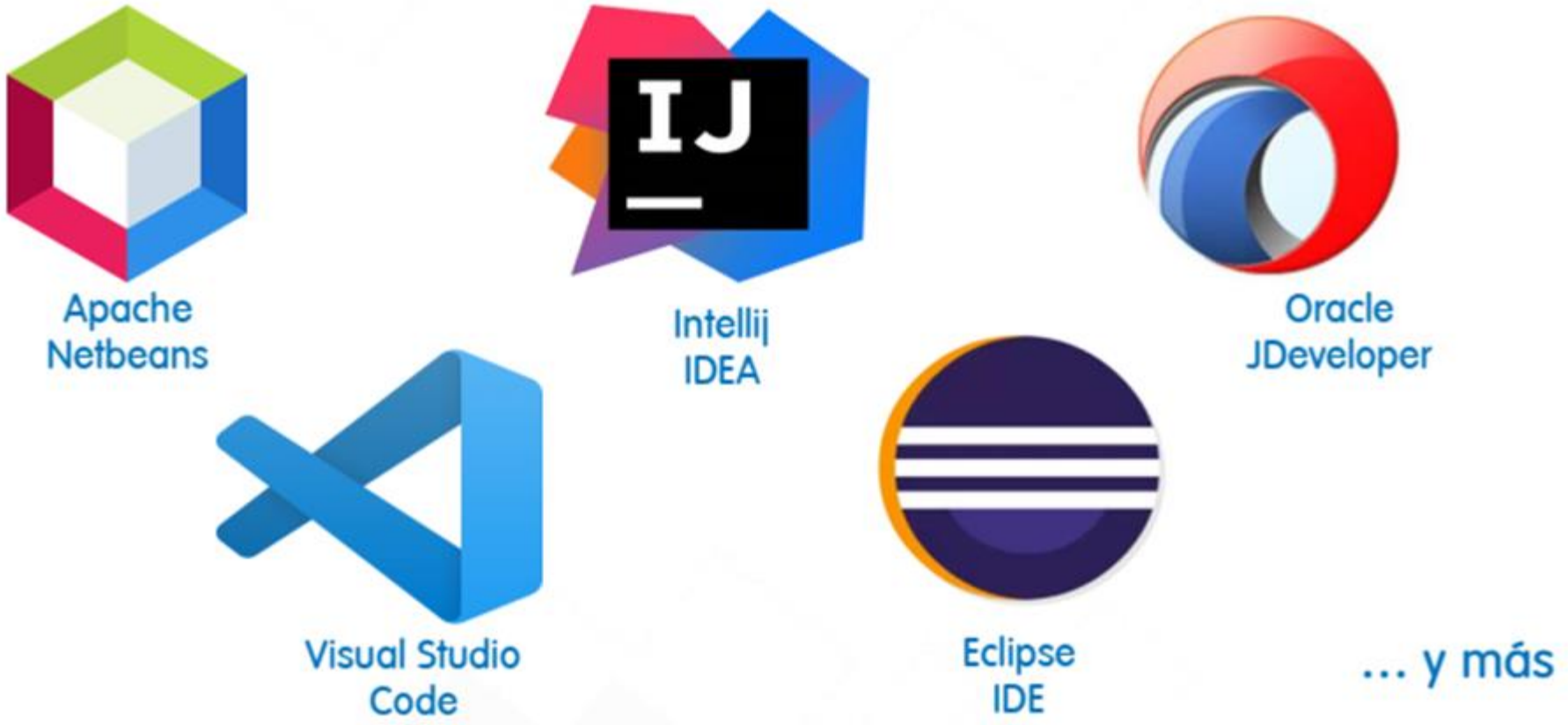


Java Compilado o Interpretado

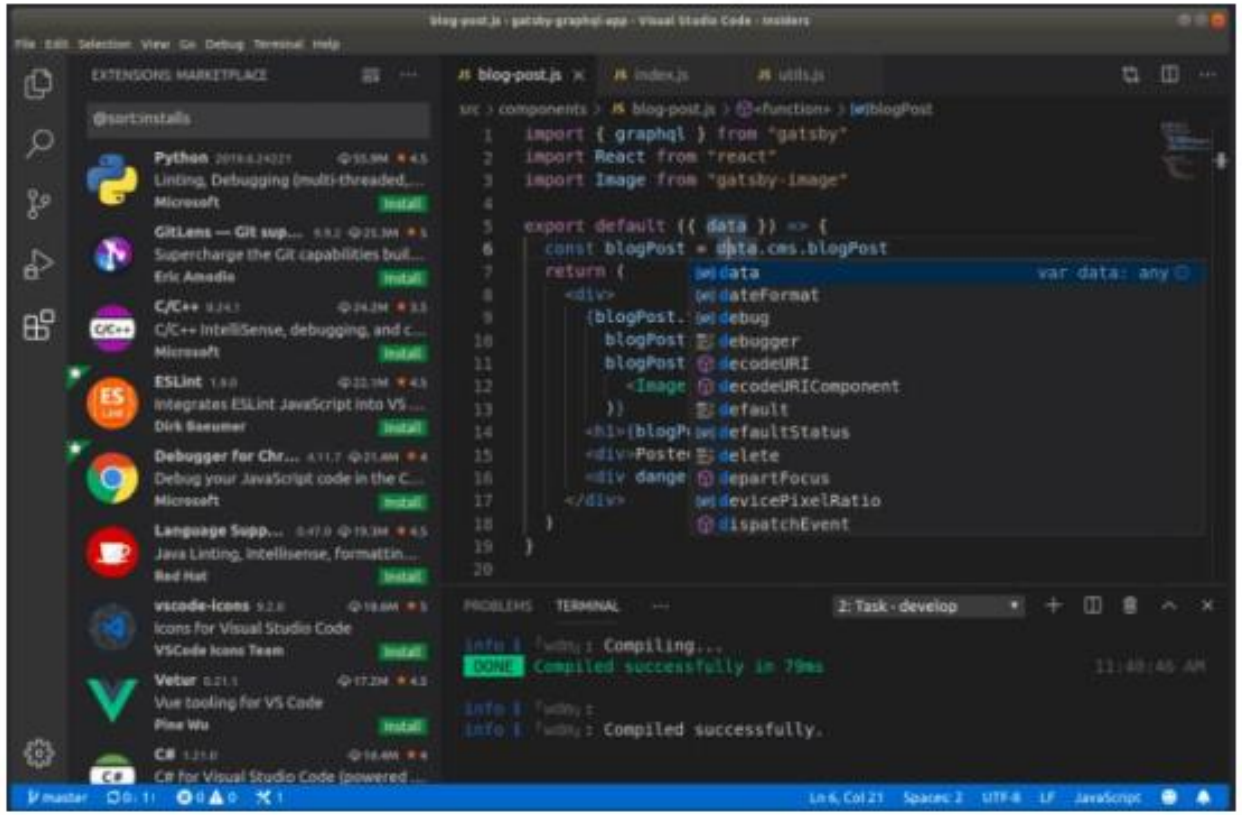
- Interpretación: El bytecode generado se ejecuta en una máquina virtual Java (JVM). La JVM es un entorno de tiempo de ejecución que interpreta y ejecuta el bytecode de forma dinámica. La interpretación ocurre en tiempo de ejecución, línea por línea, a medida que el programa se está ejecutando. Sin embargo, es importante destacar que la JVM también realiza optimizaciones y técnicas de compilación en tiempo de ejecución. Por ejemplo, puede utilizar la compilación JIT (Just-In-Time) para traducir el bytecode a código máquina nativo de la plataforma en tiempo real, lo que mejora el rendimiento y la velocidad de ejecución.



Entornos de Desarrollo Integrado – IDE para Java



Entornos de Desarrollo Integrado – IDE para Java



https://code.visualstudio.com/

Instalación de JDK de Java y Configuración de Visual Studio Code

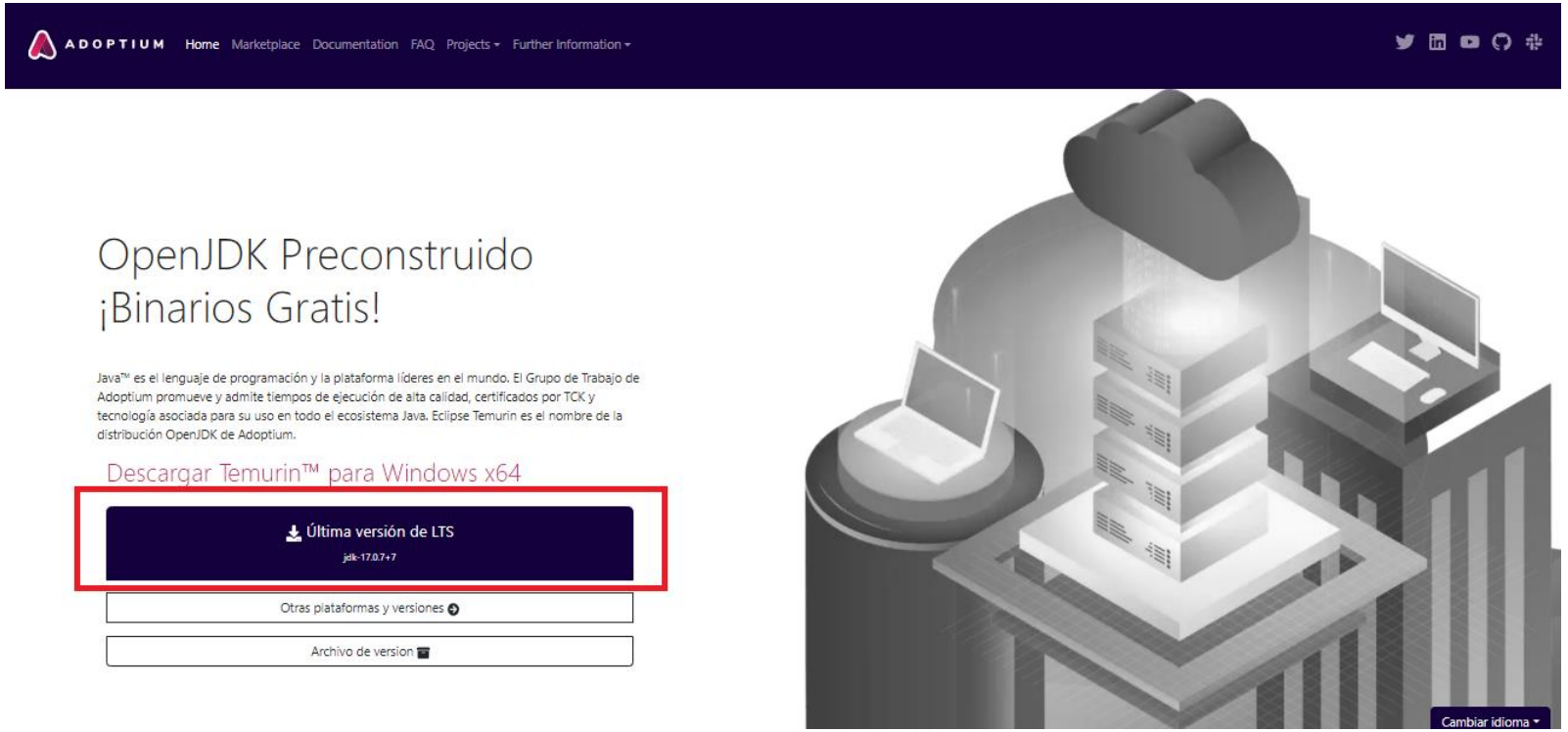
JDK 17 LTS

Pasos Para Instalación y Configuración de Java en Visual Studio Code

1. Instalación JDK de Java
2. Instalar Extensiones de Java en Visual Studio Code
3. Configurar Variable Java Home en Visual Studio Code,
4. Crear el primer Proyecto

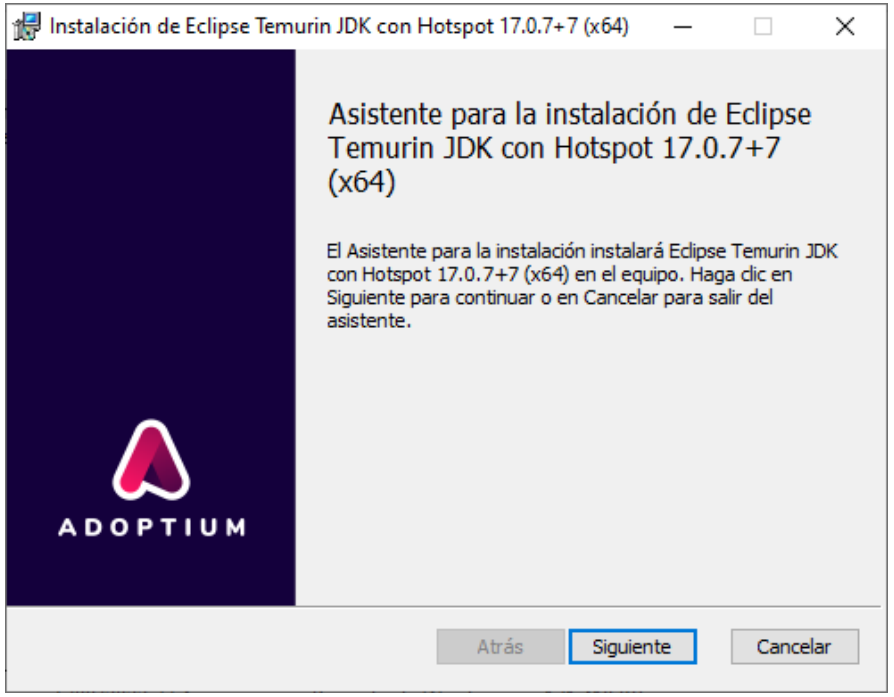
1. Instalación JDK de Java

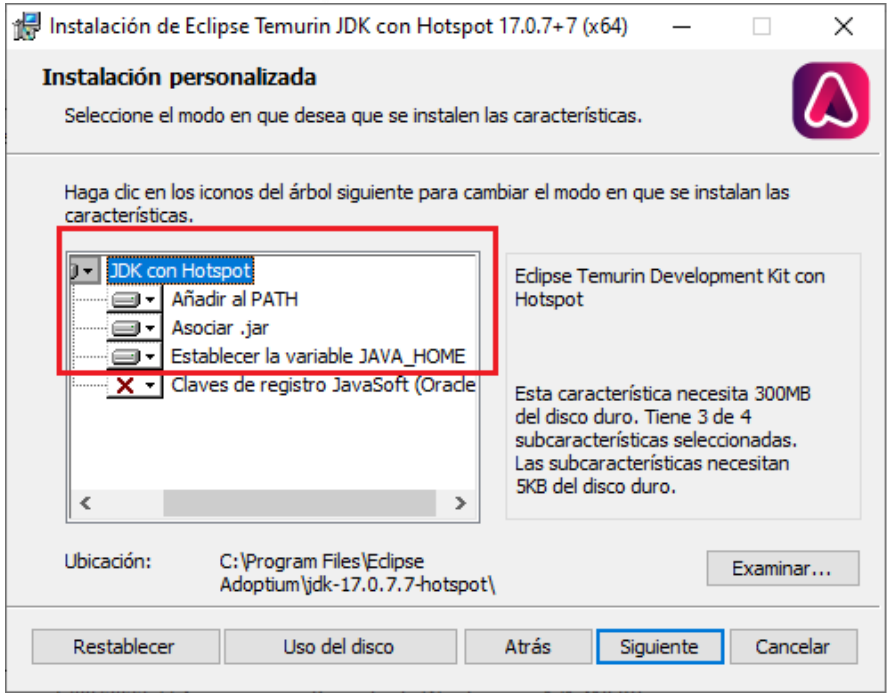
Ir a la página: https://adoptium.net/es/. Descargar y guardar la ultima versión del jdk



Instalación JDK de Java

Proceder a la instalación

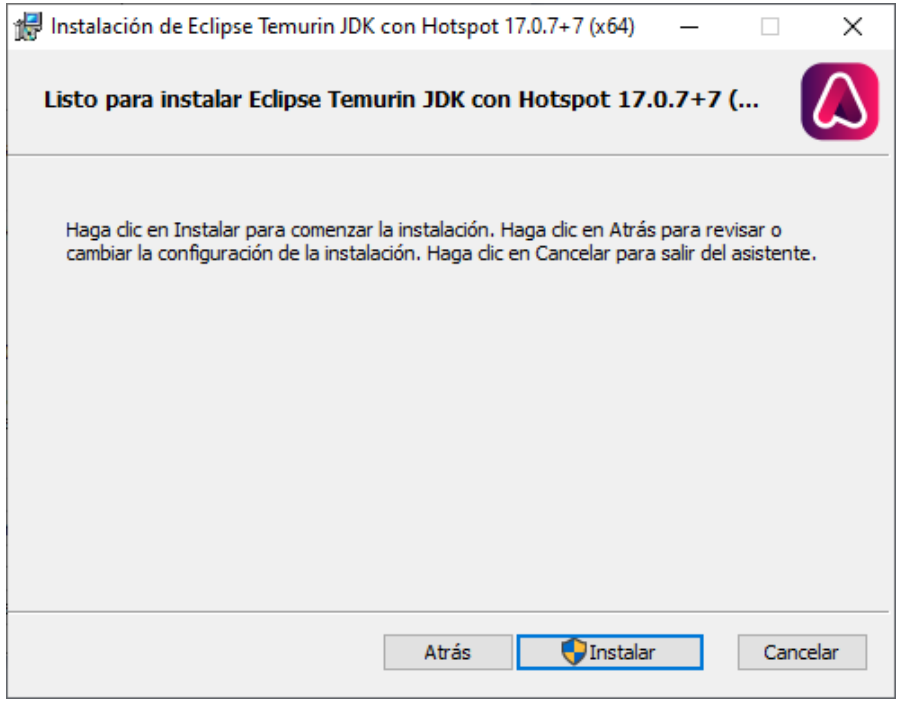


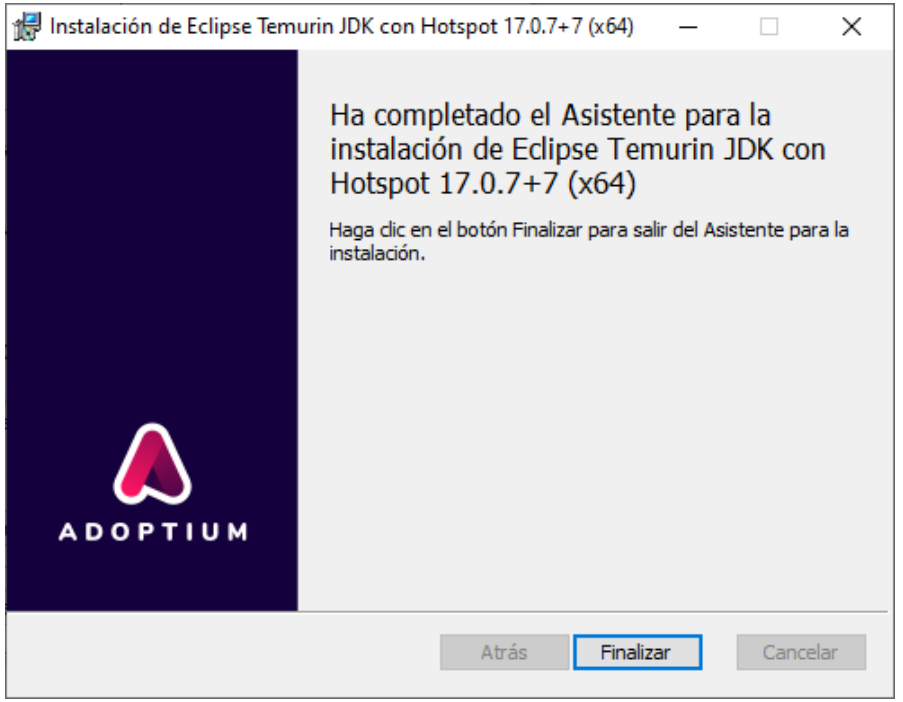


Verificar que estas tres opciones estén seleccionadas, como se muestra en la imagen.

Instalación JDK de Java

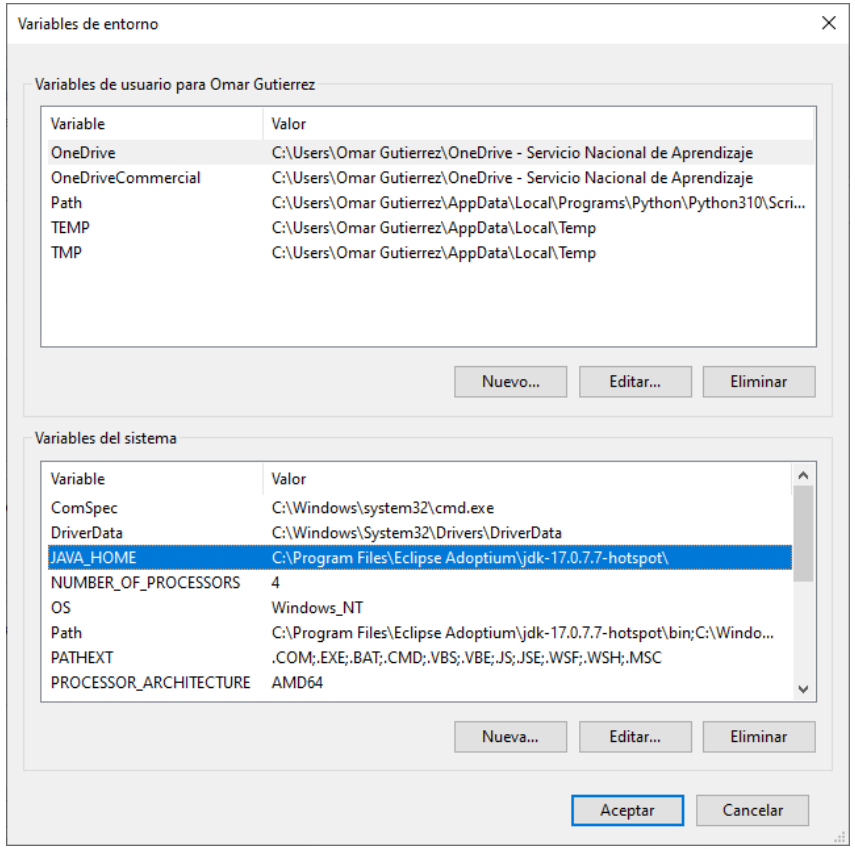
Proceder a la instalación

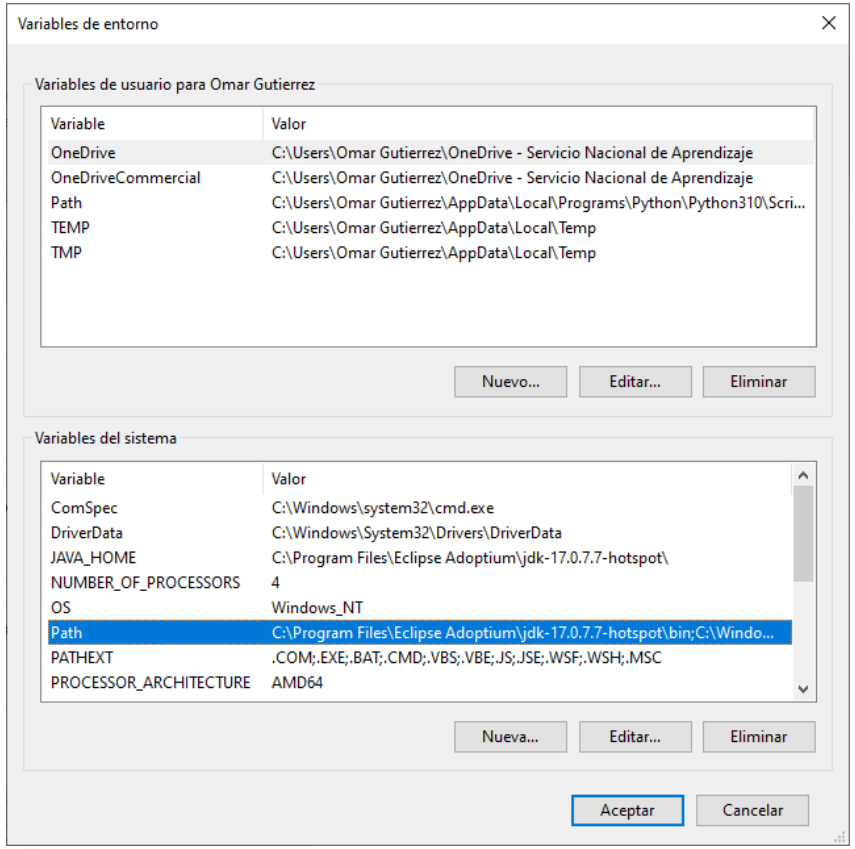




Instalación JDK de Java

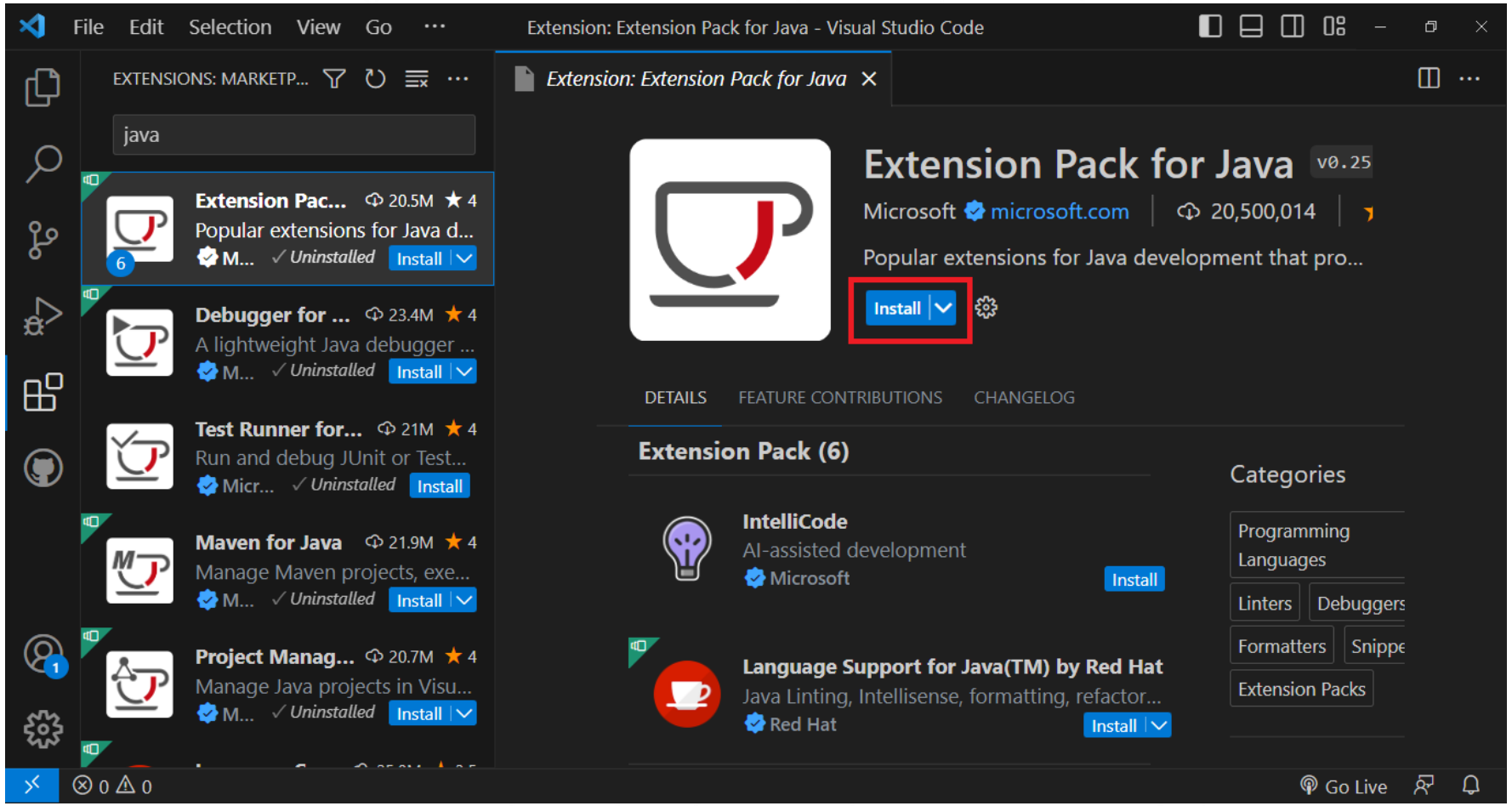
Verificar variables de Entorno JAVA_HOME y Path





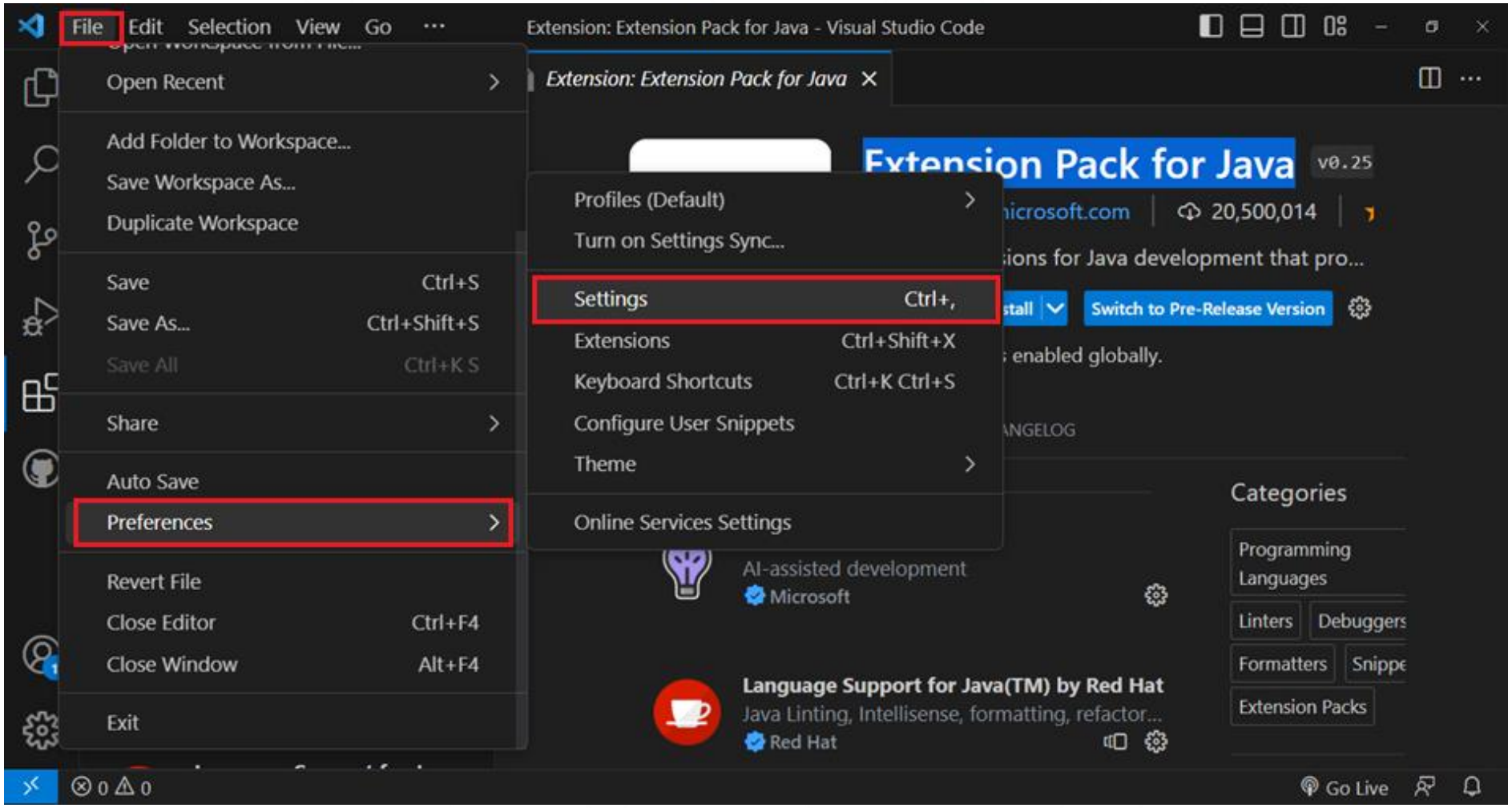
2. Instalar Extensiones de Java en Visual Studio Code

Abrir Visual Studio Code e instalar la extensión: **Extension Pack for Java** de *Microsoft*



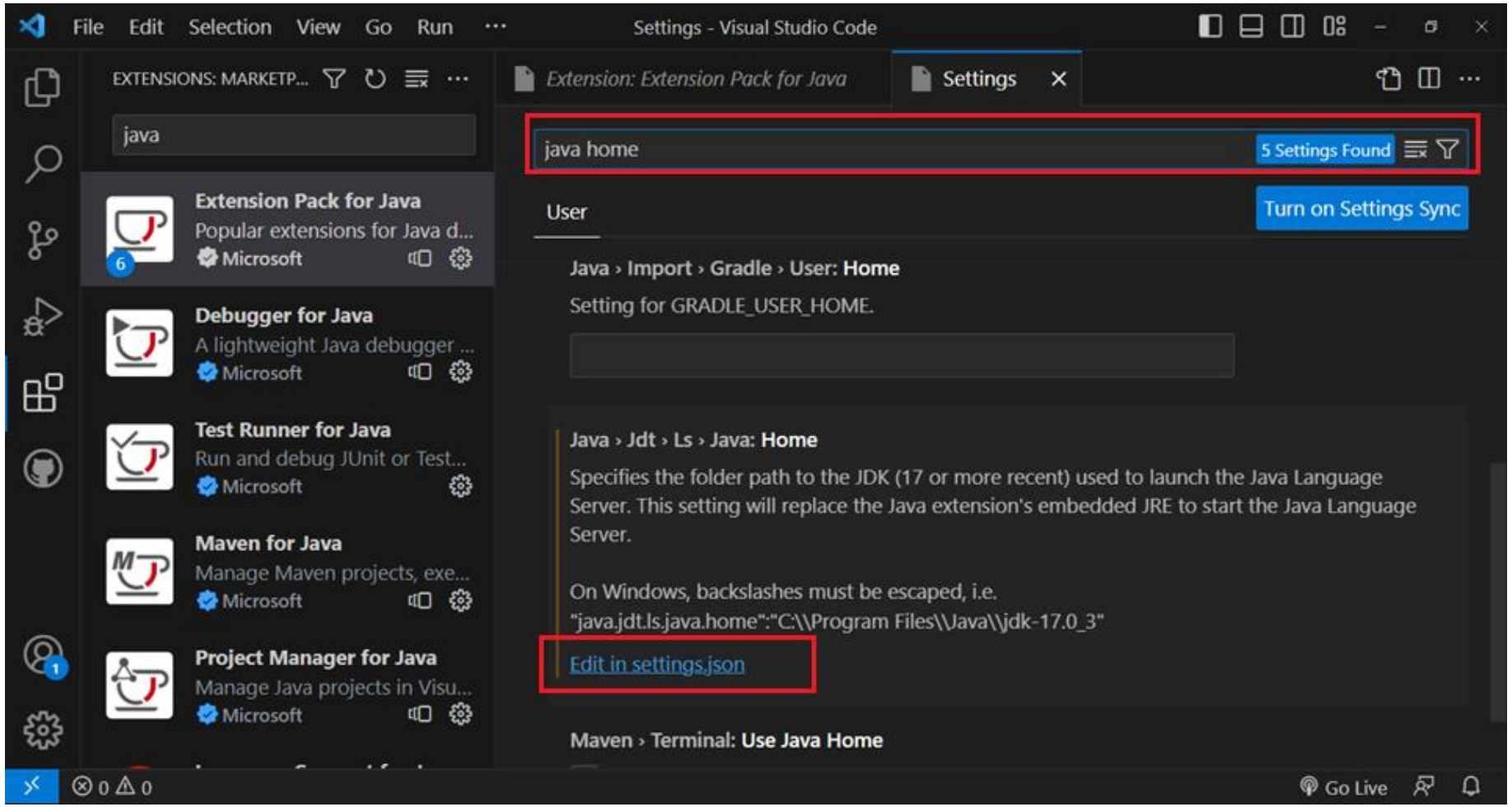
2. Configurar Variable Java Home en Visual Studio Code

En Visual Studio Code ir a la ruta File -> Preferences -> Settings



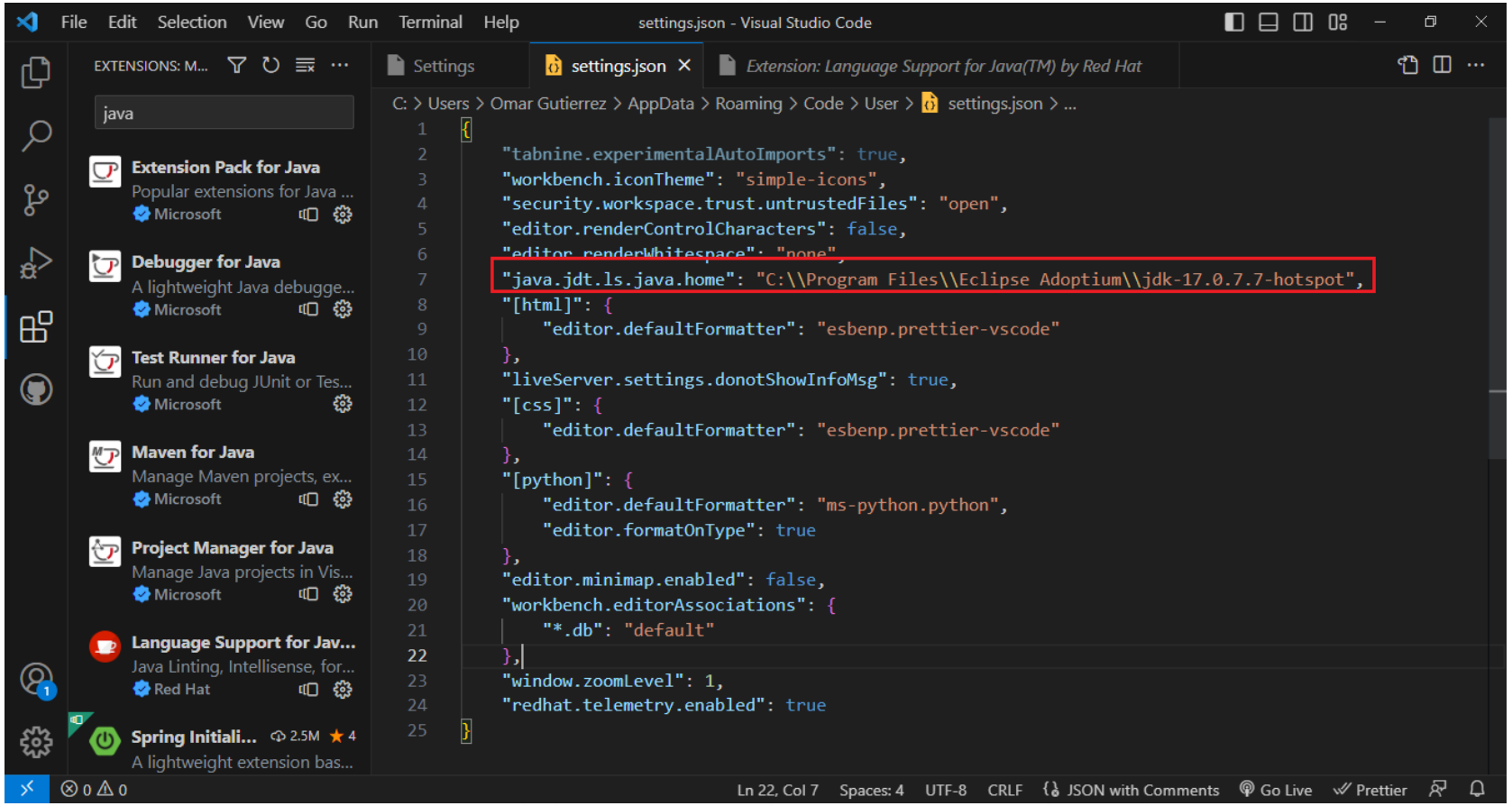
2. Configurar Variable Java Home en **Visual Studio Code**

1. Escribir Java Home y bajar hasta la opción resaltada en azul Edit in settings.json y hacer clic en esta opción



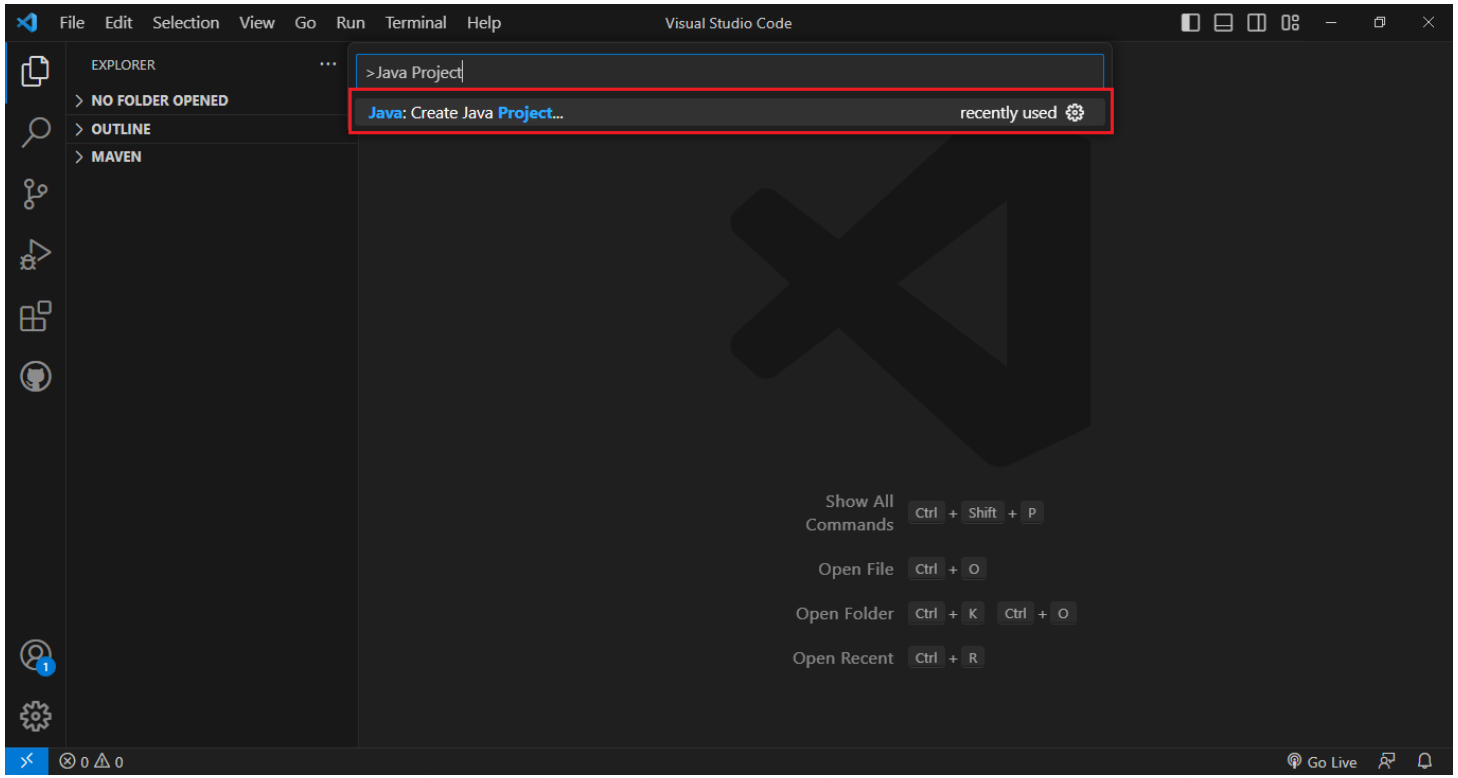
3. Configurar Variable Java Home en Visual Studio Code

1. Escribir en "java.jdt.ls.java.home": "C:\\Program Files\\Eclipse Adoptium\\jdk-17.0.7.7-hotspot" y guardar los cambios y cerrar las pestañas de los archivos abiertas.



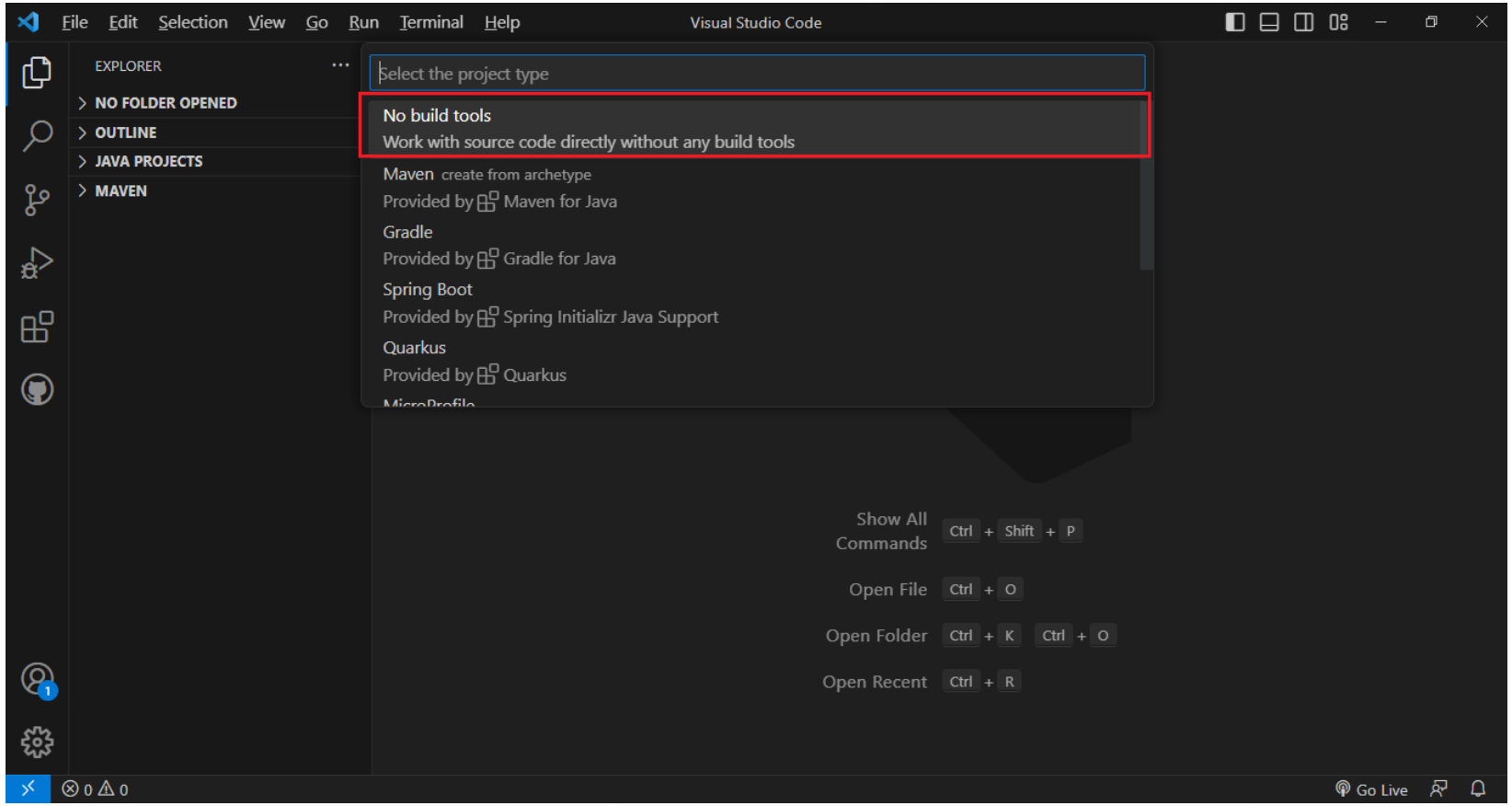
4. Crear el primer Proyecto

1. En el menú View -> Command Palette... o con la combinación de teclas Ctrl+Shift+P en la ventana que se nos abre escribimos Java Project y seleccionamos la opción mostrada



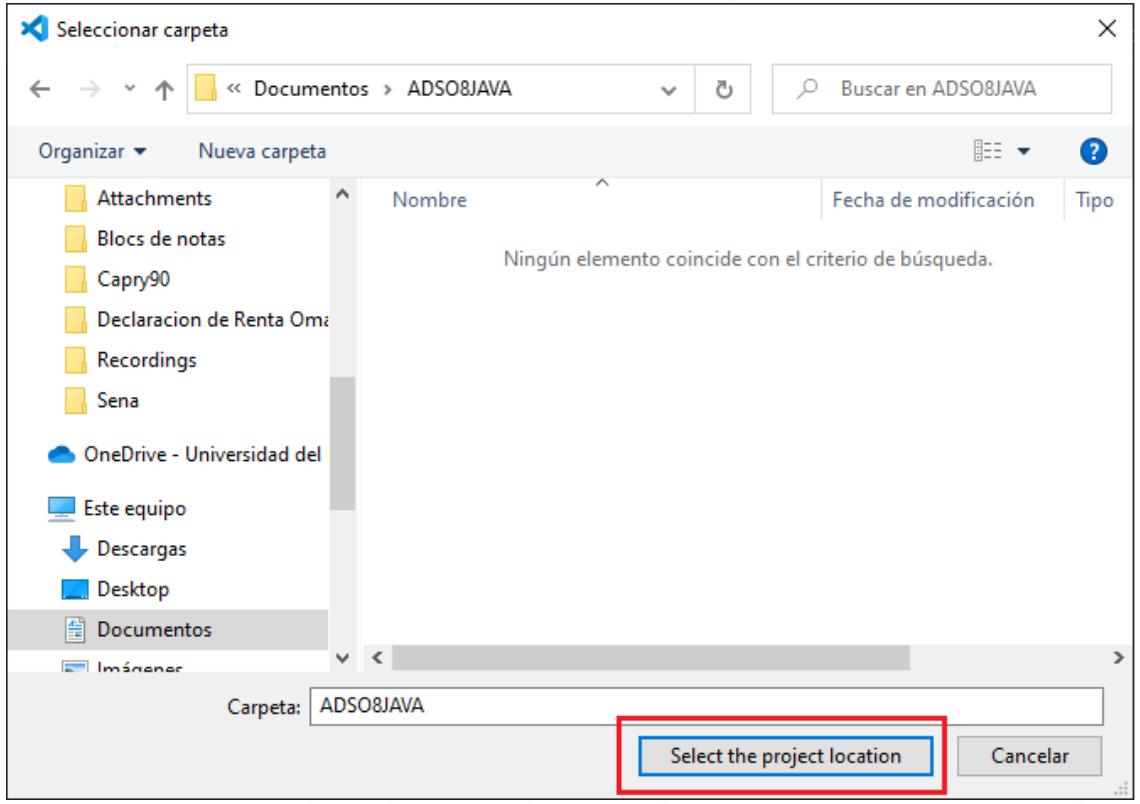
4. Crear el primer Proyecto

1. Seleccionamos la opción No builds Tools



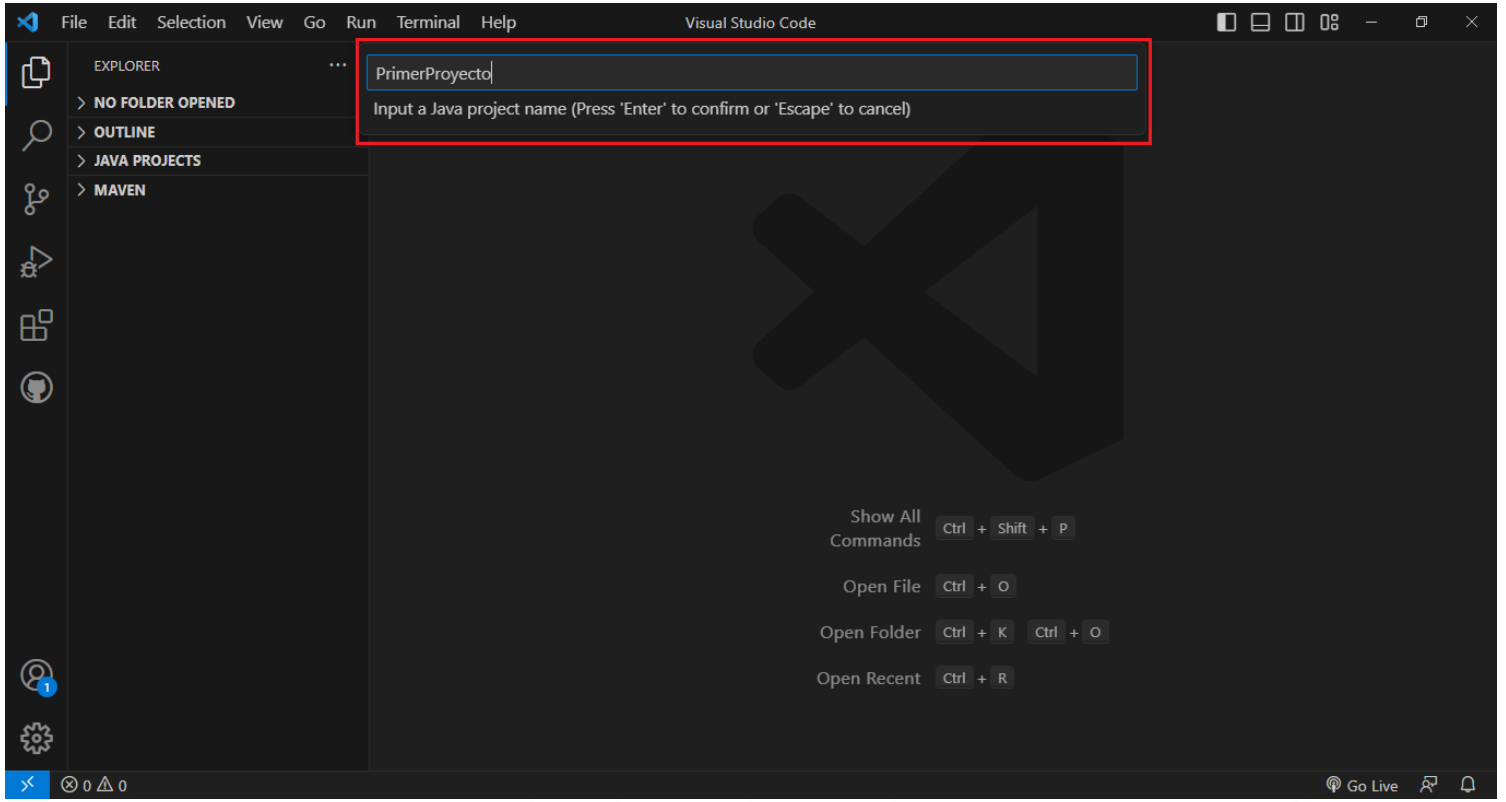
4. Crear el primer Proyecto

1. Seleccionamos la ubicación donde guardaremos nuestro proyecto



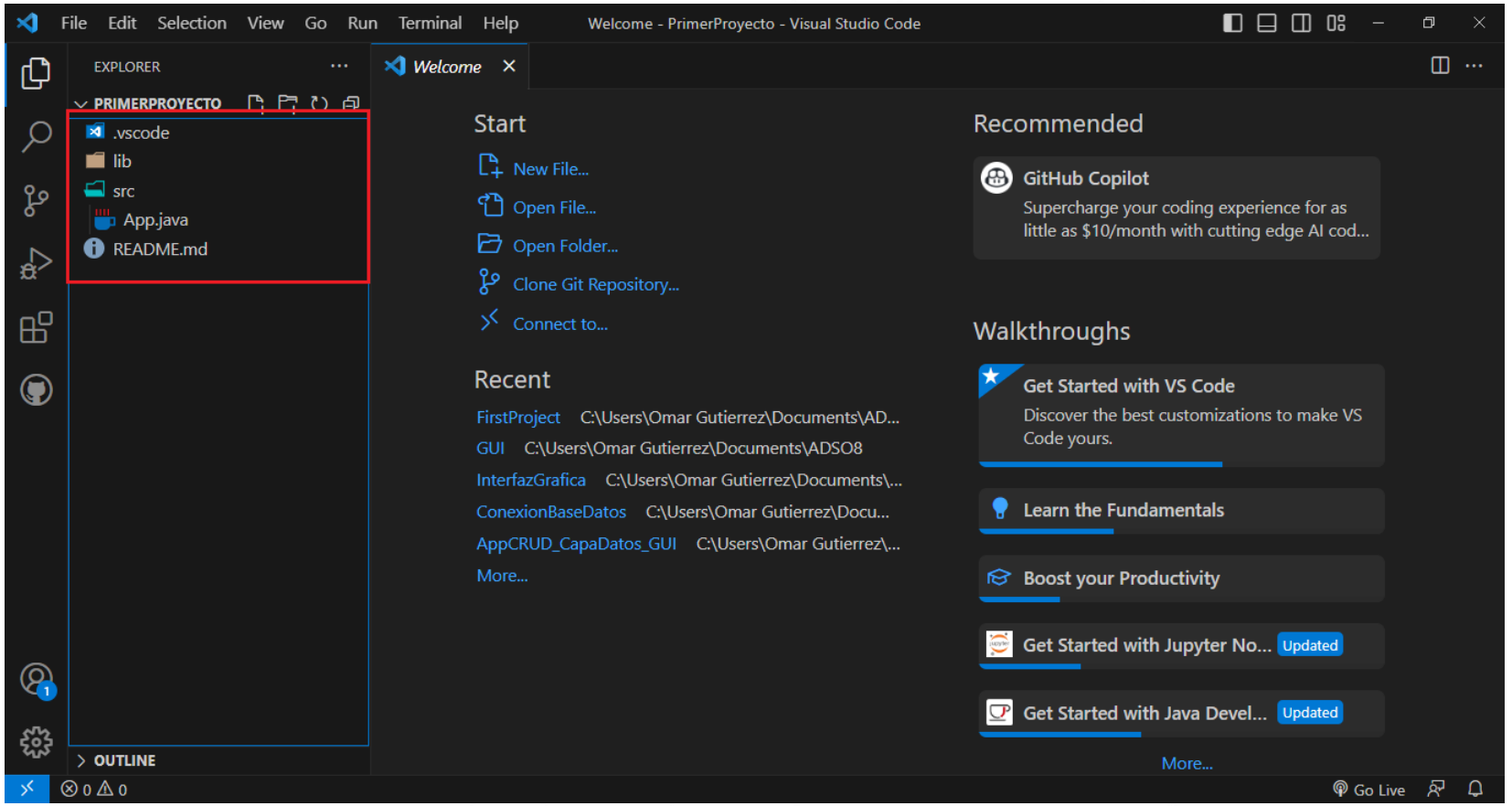
4. Crear el primer Proyecto

1. Ingresamos un Nombre a nuestro Proyecto en este caso PrimerProyecto y damos Enter para que se cree nuestro proyecto.



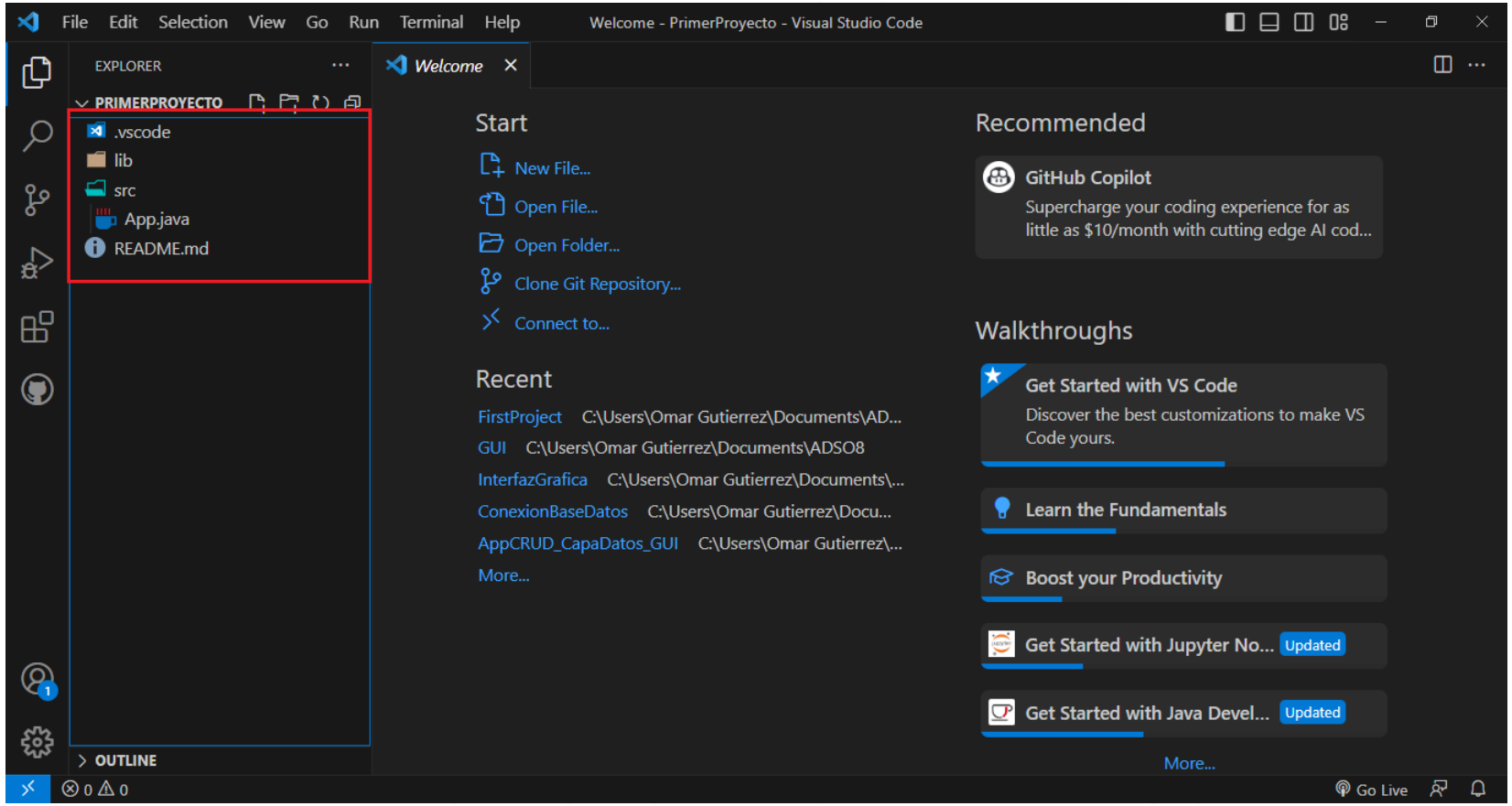
4. Crear el primer Proyecto

1. Se debe crear una estructura de proyecto como la mostrada y en la carpeta src debe estar nuestra aplicación de ejemplo.



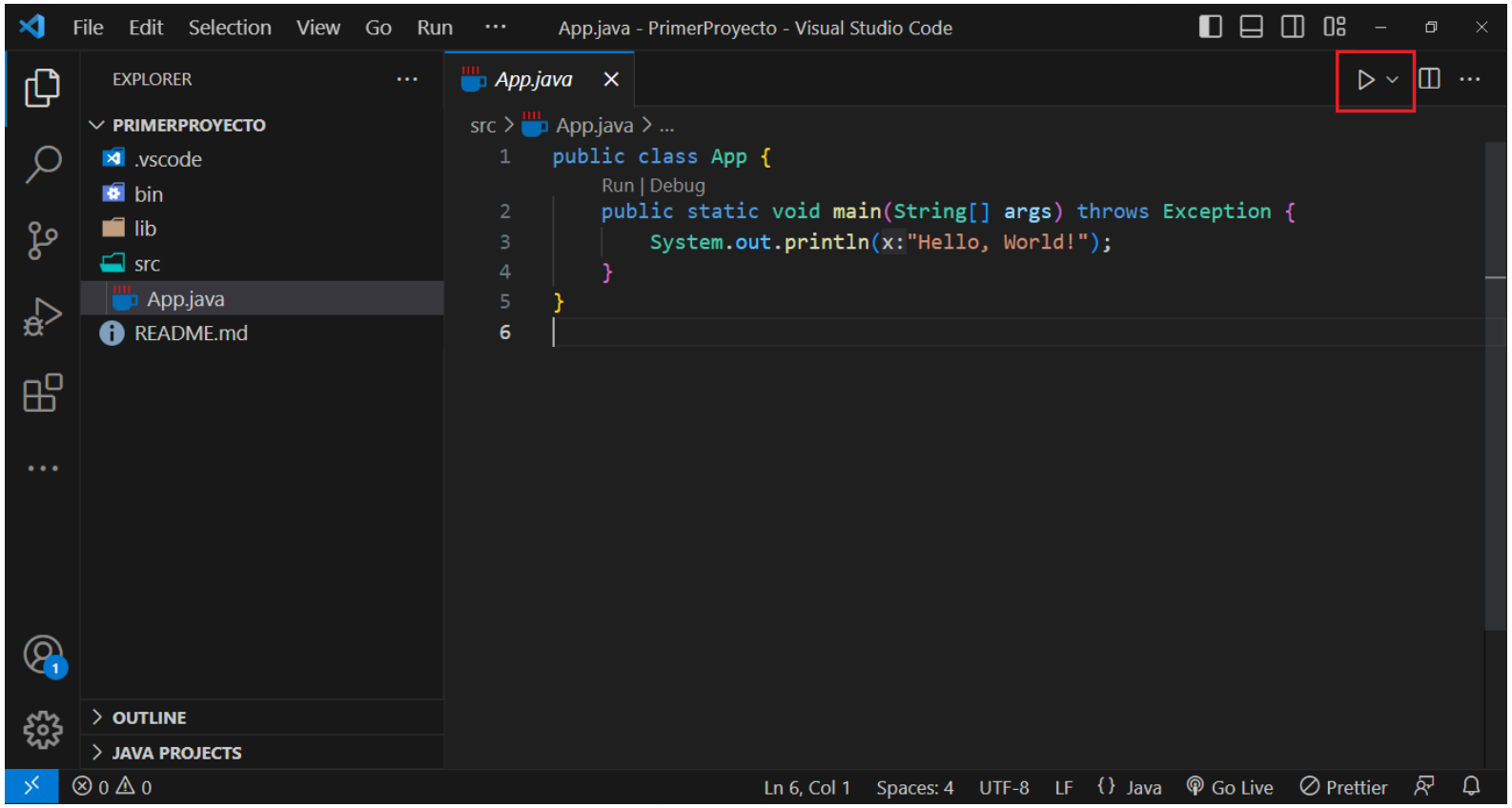
4. Crear el primer Proyecto

1. Procedemos a abrir el archivo App.java.



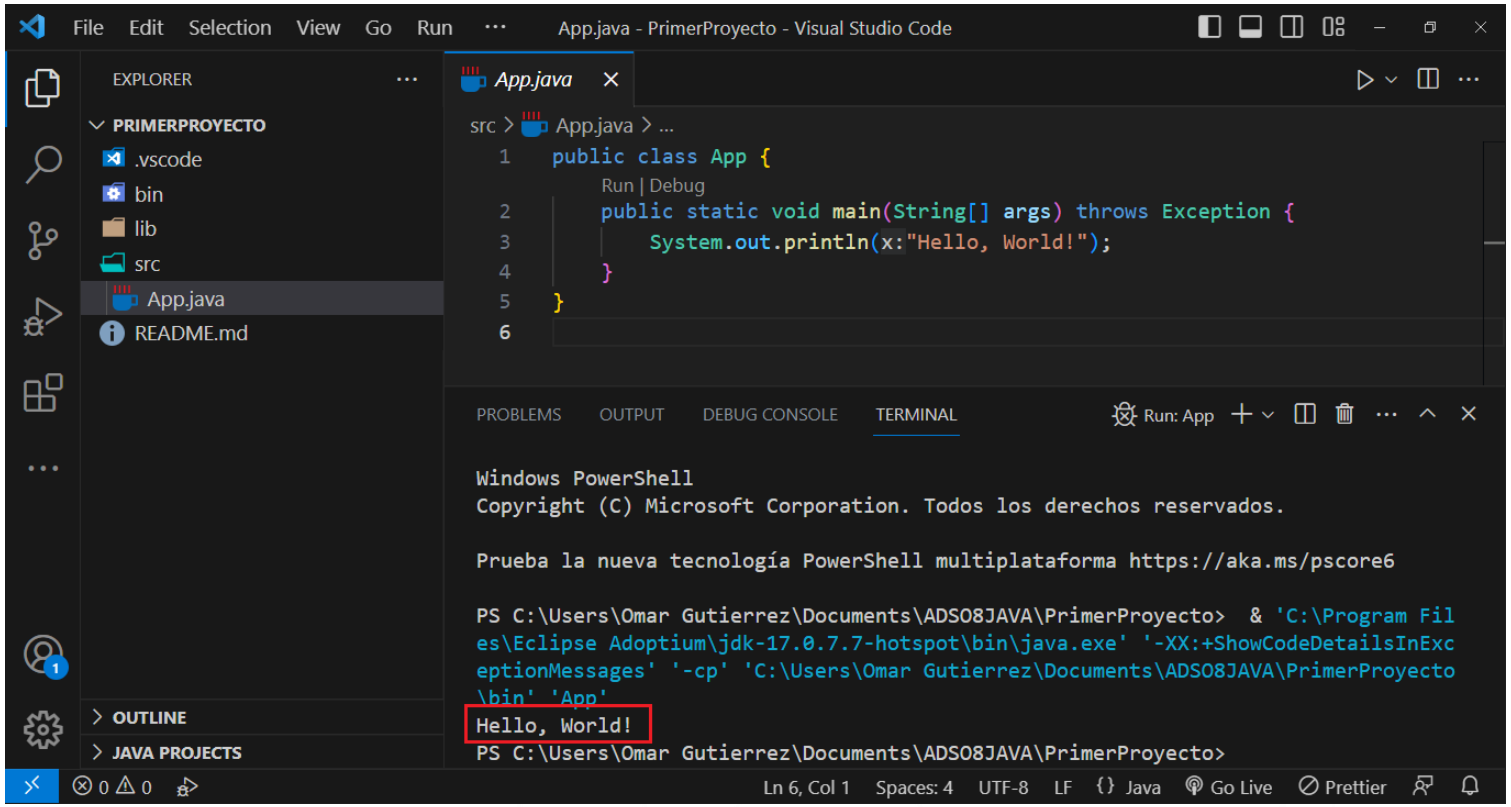
4. Crear el primer Proyecto

1. Una vez abierto el archivo App.java, procedemos a ejecutarlo en la opción Run Java



4. Crear el primer Proyecto

1. Nos debe mostrar nuestra salida de *Hello World !* indicando que nuestro ambiente de Java se encuentra correctamente instalado.



Lenguaje Java

Comentarios

```
// Este es un comentario de linea

/*
  Este es un comentario de bloque
  Todo entre estos símbolos es ignorado
 */
```

Sentencias

- Una sentencia es una orden que se le da al programa para realizar una tarea específica.
- Las sentencias acaban con punto y coma (;). Este carácter separa una sentencia de la siguiente.

```
int i=1;
import java.awt.*;
System.out.println("El primer programa");
```

Bloques de código

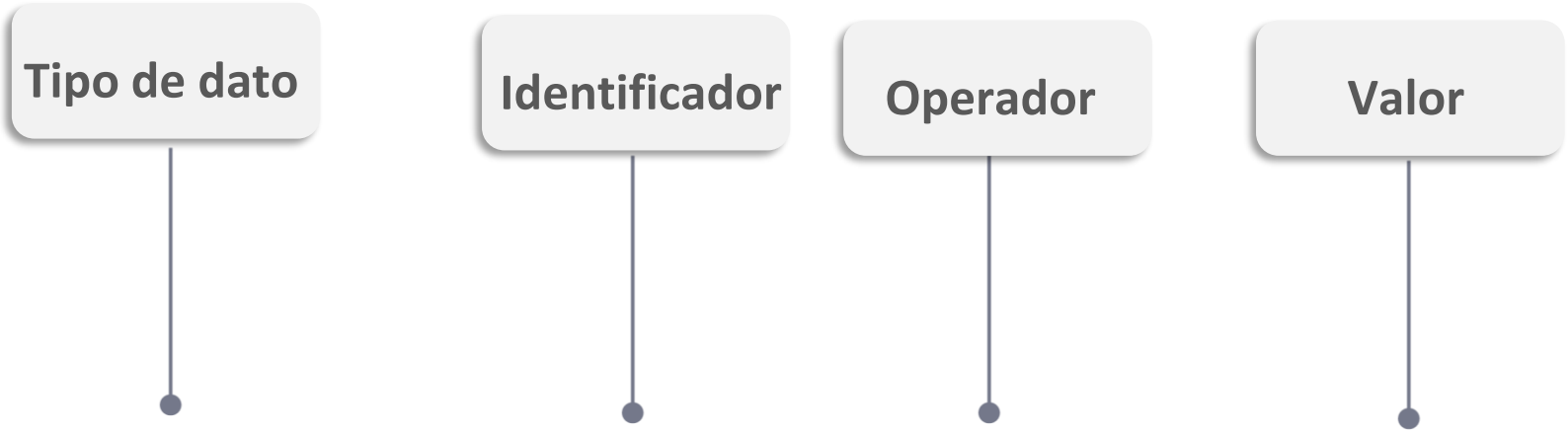
- Un bloque de código es un grupo de sentencias que se comportan como una unidad.
- Un bloque de código está limitado por las llaves de apertura { y cierre }.

```
{
    saludo="Hola mundo";
    System.out.println(saludo);
}
```

Variables

- Es un nombre que se asocia con una porción de la memoria del ordenador, en la que se guarda el valor asignado.
- **Todas las variables han de declararse antes de usarlas**, la declaración consiste en una sentencia en la que figura, el **tipo de dato** y el ***nombre*** que asignamos a la variable.

OJO: ***El tipo de dato de la variable no cambiará después de creada.***



Variables Identificador – Nombre de la variable

Es una secuencia de símbolos que se utilizan como nombres de variables, funciones, arreglos y otras estructuras de los lenguajes de programación.

- ✓ En Java se utiliza una secuencia de caracteres alfanuméricos del alfabeto inglés junto con el guion bajo (_), tales que su primer símbolo no es un dígito.
- ✓ Los identificadores tienen una longitud recomendada de 4 a 15 caracteres.
- ✓ No se pueden utilizar como identificadores las **palabras reservadas de Java**.

Validos
counter
x
diaSemana
tamano
height
Var_1
var_2
_suma
resta2

No Validos
String
1x
díaSemana
tamaño
alto/1
Var-1
for
6_suma
p@z

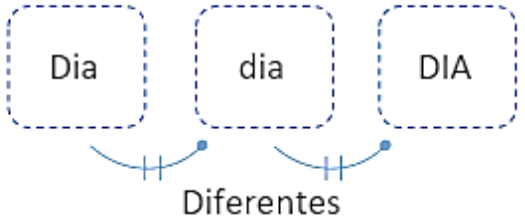
Variables

- Una vez declarada se le podrá asignar valores.

```
int x=0;
String nombre="Angel";
double a=3.5;
boolean bNuevo=true;
int[] datos;
```

Sensibilidad

Los identificadores son sensibles a mayúsculas y minúsculas.



Tipos de datos

TIPO	POR DEFECTO	TAMAÑO	VALORES EXTREMOS
boolean	false	1 bit	true, false
byte	0	8 bits	-128 a 127
char	\u0000	16 bits	'\u0000' a '\uffff'
short	0	16 bits	-32.768 a 32.767
int	0	32 bits	-2.147.486.648 a 2.147.486.647
long	0L	64 bits	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
float	0.0F	32 bits	-3.402823e38 a 3.402823e38
double	0.0	64 bits	-1.79769313486232e308 a 1.79769313486232e308

Tipos de datos – caracteres

- Símbolos definidos por el ASCII (*American Standard Code for Information Interchange*).
- La variable **char** solo puede contener un símbolo a la vez.
- Se debe usar la comilla simple para representar caracteres char, por ejemplo: 'a'.
- Existen símbolos que no tienen una representación de un carácter en el sistema, estos se representan con un el símbolo especial \ (back slash).

Algunos ejemplos con sus respectivos valores:

\n : Nueva línea.
\t : Tabular horizontal.
\\ : Back slash.
\" : Comillas.
\v : Tabulador vertical.
\? : Signo de interrogación.
\b : Retroceso.
\' : Apóstrofo.

La clase String

- Desde el punto de vista de la programación cotidiana, uno de los tipos de datos más importantes de Java es **String**.
- String define y admite cadenas de caracteres.
- En algunos otros lenguajes de programación, una cadena o string es una matriz o array de caracteres. Este no es el caso con Java. **Los Strings son objetos.**

```
"Hola Mundo"
```

Estructura de Salida - Escritura

Para la salida por consola o pantalla en Java los métodos más utilizados son **System.out.print()** y **System.out.println (),**

System.out.print():

- El método print de System.out se utiliza para imprimir los valores proporcionados en la misma línea, sin agregar un salto de línea al final.
- Puede aceptar diferentes tipos de datos, como cadenas, enteros, decimales y más.
- Puedes utilizar concatenación de cadenas o separar los elementos con comas para imprimir múltiples valores en la misma línea.

```
int edad = 25;
System.out.print("Mi edad es: ");
System.out.print(edad);
```

Estructura de Salida - Escritura

System.out.println ():

- El método println de System.out se utiliza para imprimir los valores proporcionados en una línea nueva, agregando un salto de línea al final.
- Al igual que print, println también puede aceptar diferentes tipos de datos y realizar concatenación de cadenas o separación con comas.

```
String nombre = "Juan";
int edad = 25;
System.out.println("Nombre: " + nombre);
System.out.println("Edad: " + edad);
```

EXPRESIONES Y OPERADORES

Operación de asignación

Recordemos:

Esta operación permite dar valor a variables.

Representación algorítmica: ←

En Java, al igual que en Python la operación de asignación se representa con el signo =

Pseudocódigo	Java
A ← 5	A = 5
B ← A+2	B = A+2
A ← 7	A = 7

Operadores aritméticos

$$2*c^3+b*a*\sqrt{3*a+b^2}$$

- Una expresión consta de operadores y operandos.
- Según sea el tipo de datos que manipulan, se clasifican en:
  - Aritméticas
  - Relacionales
  - Lógicas

Operadores aritméticos

Operador	Nombre	Ejemplo	Java
+	Suma	A + 4	+
-	Resta	A – 4	-
*	Multiplicación	B * 2	*
/	División	C / 3	/
Mod	Modulo (residuo de la división entera)	15 mod 2 = 1	15 % 2
Div	Cociente de la división entera	15 div 2 = 7	En Java no existe, la división de dos números enteros da un resultado entero: 15 / 2 = 7
^	Potencia	B ^ 3	Math.pow(base, potencia) Math.pow(B, 3)

Operadores relacionales

Operador	Nombre	Ejemplo	Java
>	Mayor que	4 > 2	>
<	Menor que	3 < 10	<
>=	Mayor o igual	5 >= 5	>=
<=	Menor o igual	7 <= 9	<=
=	Igual	3 = 2	==
<>	Diferente	9 <> 7	!=

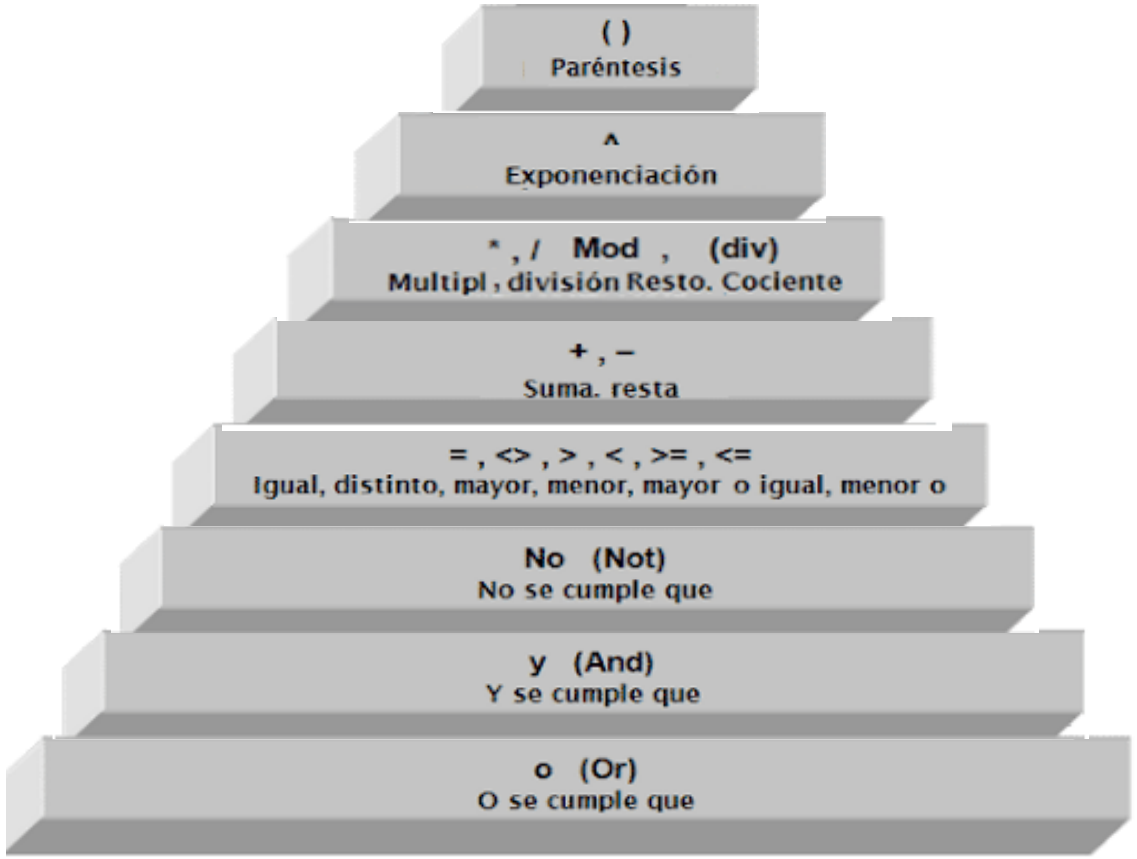
Operadores lógicos

Operador	Nombre	Ejemplo	Java
And	Y	(3 > 5) & (4 < 10)	**&&**
Or	O	(3 < 5) | (4 < 10)	**||**
NO	Negación	~(6 = 6)	**!**

- El condicional O será cierto si alguna de sus partes es cierta.
- El condicional Y será cierto si todas sus partes son ciertas

P	Q	P v Q	P ^ Q
V	V	V	V
V	F	V	F
F	V	V	F
F	F	F	F

Prioridad de todos los operadores



Estructura de Entrada - Lectura

Java:

Se debe importar la librería **java.util.Scanner**; luego se define una variable que permitirá realizar la captura de los datos por teclado.

```
Scanner leer = new Scanner(System.in);
```

Dependiendo del tipo de dato, estos se leen utilizando nextTipo (donde Tipo puede ser: Int, Float, Double, Line) de la siguiente forma:

```
a = leer.nextInt();
b = leer.nextInt();
```

Ejemplo

```java
import java.util.Scanner;

public class LeerNumeroPorConsola {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Ingresa un número: ");
        int numero = scanner.nextInt();

        System.out.println("El número ingresado es: " + numero);

        scanner.close();
    }
}
```

Cast Conversion entre tipos de Datos

String (cadena) a int (entero):

```
String enteroString = '5';
int entero = Integer.parseInt(enteroString);
System.out.println(entero);
```

int (entero) a String:

```
int entero = 5;
String enteroString = Integer.toString(entero);
System.out.println(enteroString);
```

String a double:

```
String doubleString = '8342342';
double aDouble = Double.parseDouble(aString);
System.out.println(aDouble);
```

Double a String:

```
double d = 8342342;
String aString = Double.toString(d);
System.out.println(aString);
```

Errores comunes

1. Olvidar un punto y coma al final de una instrucción o sentencia.
2. No cerrar llaves de algún bloque de código, método, clase o en alguna estructura de control.
3. Colocar el mismo nombre a variables con diferente tipo.
4. Asignar una variable con tipo de dato diferente. Ejemplo: se declara una variable de tipo String y se le asigna una variable de tipo int, en este caso el compilador arroja un error de conversión de tipos.
5. Ingreso de valores diferentes a los que la aplicación recibe.
6. Acceder a una posición que no existe en un arreglo.
7. Almacenar cadenas donde se debe almacenar números.
8. Divisiones por cero.

Ejercicios Básicos Java

1. Hacer una programa que calcule el área y el perímetro de un círculo. Mostrar por terminal. Mostrar el área y perímetro redondeado a dos cifras decimales.

2. Hacer una programa que calcule el volumen de un cubo regular. Mostrar por terminal

3. Hacer una programa que intercambie el valor de dos variables.

4. Hacer una programa que lea var1 y var2 y calcule valor redondeado a dos cifras decimales

valor= (var1*var2)/(var1+var2)