



UNIVERSIDAD NACIONAL DE LOJA

Facultad de la Energía, las Industrias y los Recursos
Naturales no Renovables

Asignatura:

Teoría de la Programación

Unidad: 1

Tema:

*Revisión de tutoriales oficiales de instalación de lenguajes de
programación (C, Python o Java).*

Docente:

Ing. Lissette Geoconda López Faicán

Estudiante:

Valeria Idaly Agila Gomez

1859

1. Origen

El lenguaje de programación Java fue desarrollado por James Gosling y su equipo en Sun Microsystems entre los años 1991 y 1995. Su presentación oficial fue en mayo de 1995, junto con el navegador HotJava, que fue la primera aplicación capaz de ejecutar programas Java directamente desde la web.

El objetivo principal del proyecto era crear un lenguaje independiente de la plataforma, es decir, que el mismo programa pudiera funcionar en diferentes sistemas operativos (como Windows, Linux o macOS) sin necesidad de modificar el código. Este principio se conoce como “Write Once, Run Anywhere” (WORA), o “escríbelo una vez, ejecútalo en cualquier lugar”, y fue posible gracias a la introducción del bytecode (un código intermedio que no depende del hardware) y de la Java Virtual Machine (JVM), una máquina virtual encargada de traducir ese bytecode para que el sistema operativo lo entienda [1].

A diferencia de lenguajes como C o C++, Java eliminó los punteros (variables que almacenan direcciones de memoria) para evitar errores graves de seguridad y estabilidad. También añadió un sistema de recolección automática de basura (Garbage Collector), que libera memoria sin intervención del programador. Este proceso analiza qué objetos ya no están siendo utilizados por el programa y automáticamente los elimina de la memoria, evitando fallos como fugas de memoria (memory leaks).

Desde su creación, Java se diseñó para ser orientado a objetos (basado en clases y objetos que modelan entidades del mundo real), seguro, portátil y eficiente, convirtiéndose en una base sólida para el desarrollo de aplicaciones modernas [1], [2].

2. Uso

Java es uno de los lenguajes más utilizados en el mundo debido a su versatilidad, portabilidad y robustez. Su aplicación abarca diversos campos tecnológicos, desde sistemas empresariales hasta inteligencia artificial y desarrollo móvil.

En el ámbito empresarial, Java se usa para desarrollar aplicaciones backend (la parte del sistema que maneja la lógica, bases de datos y servicios internos) mediante plataformas como Spring Framework y Jakarta EE (Enterprise Edition). Estas herramientas permiten construir sistemas distribuidos (conectados por red y ejecutándose en diferentes servidores) de manera eficiente y escalable.

Su fiabilidad ha hecho que bancos, instituciones gubernamentales y grandes empresas lo utilicen para manejar millones de transacciones por segundo, garantizando seguridad, integridad y mantenimiento a largo plazo [1].

En el ámbito móvil, Java fue el lenguaje principal en el sistema operativo Android, que utiliza una versión adaptada de la JVM conocida como Dalvik Virtual Machine (posteriormente reemplazada por ART – Android Runtime). Gracias a esto, millones de aplicaciones móviles fueron creadas en Java, lo que consolidó su dominio en el desarrollo móvil [2].

En el entorno científico y académico, Java se utiliza para desarrollar simulaciones, análisis de datos y programas educativos, ya que su sintaxis clara y su orientación a objetos facilitan el aprendizaje de la programación estructurada. Además, al ser multiplataforma, permite ejecutar los mismos programas en diferentes sistemas, lo cual es ideal para proyectos universitarios o investigaciones colaborativas.

También es ampliamente usado en aplicaciones de escritorio (como herramientas administrativas, gráficas o contables) mediante bibliotecas como Swing y JavaFX, que permiten crear interfaces visuales interactivas con menús, botones y gráficos dinámicos. Otro campo importante es el desarrollo en la nube y los microservicios (pequeños programas independientes que se comunican entre sí). Gracias a su compatibilidad con tecnologías modernas como Docker (contenedores que aíslan aplicaciones) y Kubernetes (plataforma de orquestación de contenedores), Java mantiene un rol esencial en sistemas distribuidos, seguros y altamente disponibles [1].

Java también sirve como plataforma multilenguaje, ya que sobre la JVM se pueden ejecutar otros lenguajes modernos como Kotlin, Scala o Groovy, todos compatibles con las bibliotecas y herramientas de Java, lo que extiende aún más su alcance y adaptabilidad [2].

Además, Java se usa en sistemas embebidos, es decir, pequeños dispositivos electrónicos (como routers, cajeros automáticos o sensores industriales) donde se requiere estabilidad y capacidad de ejecución en entornos de recursos limitados. En estos casos, se utiliza la versión Java Micro Edition (Java ME), optimizada para bajo consumo de memoria y energía.

3. Ventajas

Las principales ventajas técnicas de Java que explican su éxito en el desarrollo de software son las siguientes:

- **Independencia de plataforma:** Gracias al uso del bytecode (código intermedio universal) y de la **JVM**, un mismo programa puede ejecutarse en diferentes sistemas operativos sin cambios. Esto permite una portabilidad total del software [1].
- **Gestión automática de memoria:** El Garbage Collector (recolector de basura) libera memoria ocupada por objetos que ya no se usan, evitando errores como fugas de memoria o fallos del sistema. Este proceso se ejecuta en segundo plano y optimiza el rendimiento general de la aplicación.
- **Seguridad integrada:** Java ejecuta los programas dentro de la JVM, lo que crea un entorno aislado (sandbox) que protege al sistema operativo de posibles daños. Además, cuenta con un verificador de bytecode y un Security Manager que controlan el acceso a recursos críticos, impidiendo que programas maliciosos accedan a memoria o archivos sin permiso [2].
- **Modelo orientado a objetos:** La estructura de Java basada en clases y objetos permite desarrollar sistemas modulares, fáciles de mantener y de escalar. Esto facilita la reutilización de código y mejora la organización del software, reduciendo la complejidad en proyectos grandes.
- **Amplia API estándar:** Java incluye una extensa colección de bibliotecas preinstaladas que facilitan el manejo de archivos, redes, bases de datos, criptografía, interfaces gráficas y concurrencia (la capacidad de ejecutar múltiples tareas al mismo tiempo).
- **Rendimiento optimizado:** La **compilación Just-In-Time (JIT)** convierte el bytecode en instrucciones nativas mientras el programa se ejecuta, mejorando significativamente la velocidad sin perder portabilidad.
- **Ecosistema maduro:** Java cuenta con una de las comunidades más grandes del mundo de la programación. Existen miles de frameworks, herramientas y documentación técnica que garantizan un soporte sólido, además de actualizaciones regulares por parte de Oracle y la comunidad OpenJDK [1].

- **Gran estabilidad y compatibilidad:** las versiones de Java se mantienen estables durante años, lo que asegura que los programas desarrollados sigan funcionando incluso después de actualizaciones del entorno.

4. Limitaciones

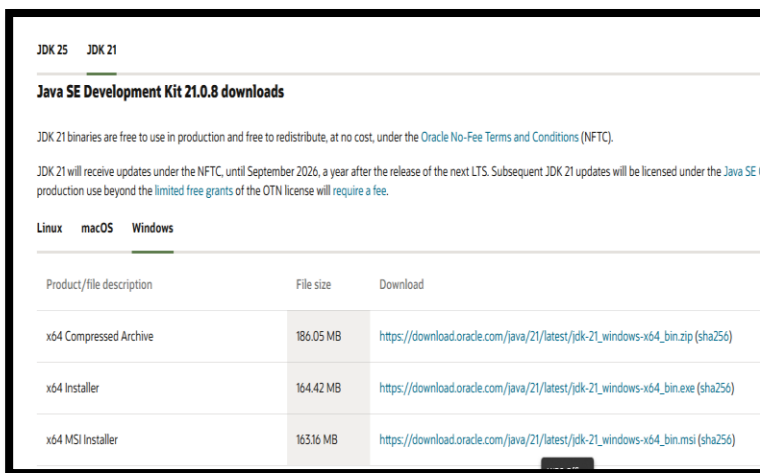
Aunque Java es muy potente, también presenta algunas limitaciones técnicas que deben considerarse en proyectos específicos:

- **Sobrecarga de la JVM:** la **Java Virtual Machine** introduce una capa de ejecución adicional, lo que puede aumentar el consumo de recursos del sistema (CPU y memoria) en comparación con lenguajes que se ejecutan directamente sobre el hardware, como C o C++ [1].
- **Pausas del recolector de basura:** el **Garbage Collector** puede detener momentáneamente el programa para liberar memoria, lo que genera latencia (demoras pequeñas pero perceptibles) en aplicaciones de tiempo real, como videojuegos o sistemas industriales [1].
- **Consumo de memoria elevado:** las aplicaciones Java requieren más memoria RAM debido a la estructura interna de los objetos y al funcionamiento de la JVM, lo que las hace menos adecuadas para dispositivos embebidos (como microcontroladores o sistemas pequeños) [2].
- **Complejidad en la programación concurrente:** aunque Java ofrece herramientas para ejecutar múltiples hilos de procesamiento (multithreading), su mal uso puede generar errores difíciles de detectar como deadlocks (bloqueos mutuos entre procesos) o condiciones de carrera (cuando dos hilos intentan modificar el mismo dato simultáneamente).
- **Compatibilidad entre versiones:** las nuevas versiones del JDK (Java Development Kit) pueden introducir cambios que obligan a modificar código existente, lo que implica un esfuerzo adicional de mantenimiento.
- **Ausencia de control de bajo nivel:** Java no permite manipular directamente la memoria ni acceder al hardware, lo cual limita su uso en sistemas donde se necesita el máximo rendimiento o interacción directa con dispositivos físicos [2].

5. Pasos de instalación

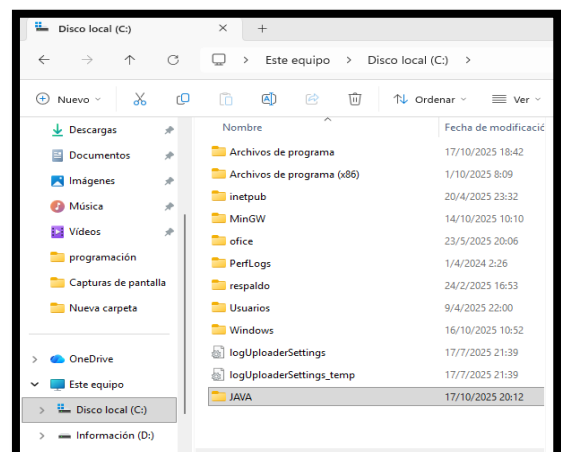
Paso 1: Descargar JDK 21

1. Ingresamos a nuestro navegador de confianza.
2. Accedemos al siguiente enlace oficial de Oracle:
<https://www.oracle.com/java/technologies/downloads/>
3. Buscamos la sección correspondiente a **JDK 21**.
4. Seleccionamos el sistema operativo que utilizamos; en mi caso, **Windows**.
5. Nos dirigimos a la descripción del producto y seleccionamos la opción “**x64 Installer**”.
6. Esperamos a que se complete la descarga.
7. Una vez finalizada, hacemos clic sobre el archivo descargado y seguimos el asistente hasta completar la instalación.



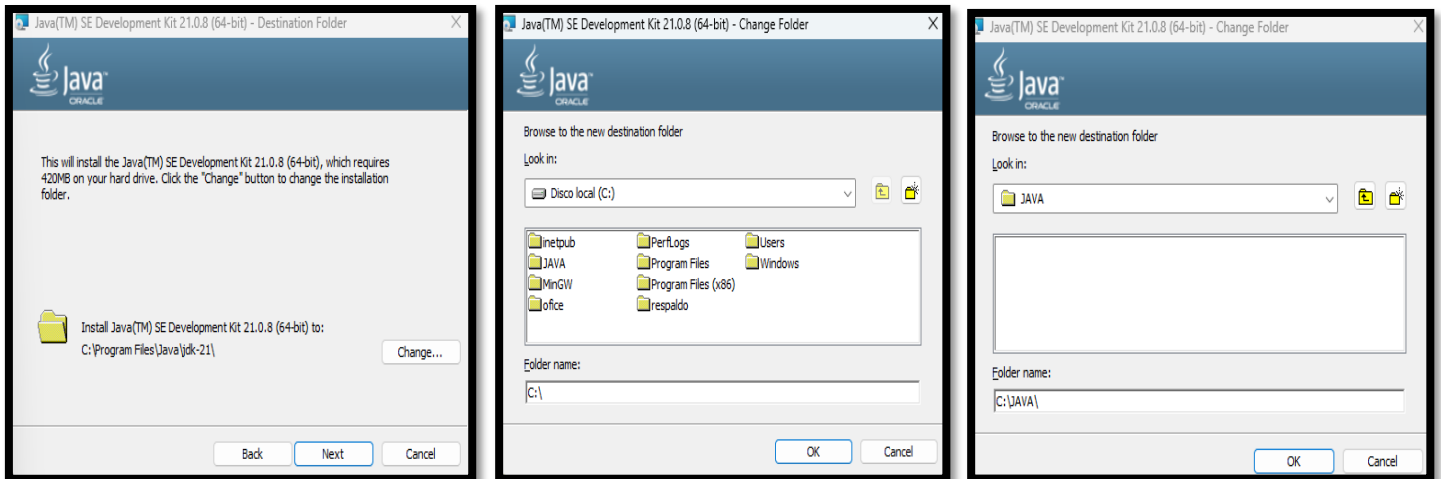
Paso 2: Crear la carpeta de instalación

1. Abrimos el **Explorador de archivos**.
2. Nos dirigimos al **Disco local (C:)**.
3. Creamos una nueva carpeta para instalar Java; en mi caso, la llamé “**JAVA**”.



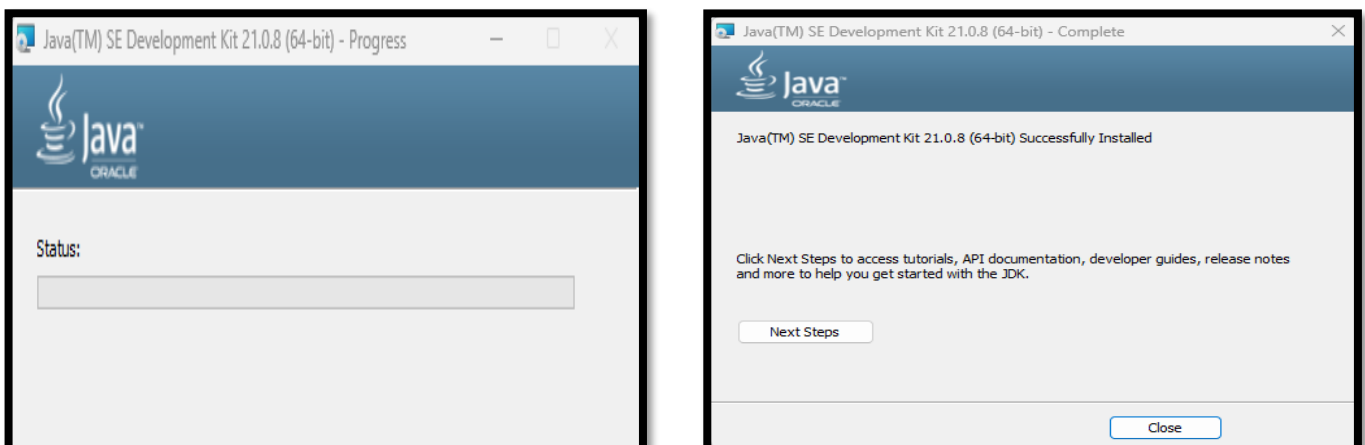
Paso 3: Cambiar la carpeta de instalación

1. Nos dirigimos a la descarga del instalador de JDK 21.
2. Pulsamos el botón para cambiar la carpeta de instalación.
3. Buscamos la carpeta donde vamos a instalar Java; en este caso, **la carpeta “JAVA”** que creamos en el Disco local (C:).
4. Seleccionamos nuestra carpeta y presionamos **“OK”**.



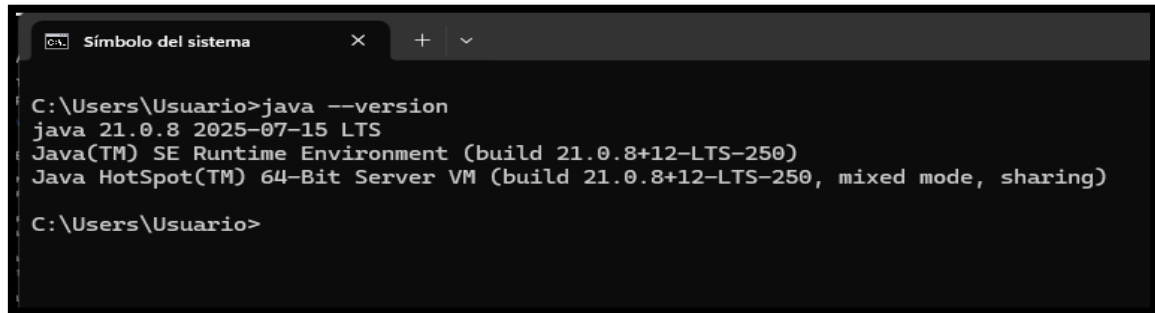
Paso 4: Finalizar la instalación

1. La instalación comenzará a copiar los archivos y configurarse automáticamente.
2. Esperamos a que el proceso termine por completo.
3. Una vez finalizado, cerramos el instalador haciendo clic en **“Close”**.



Paso 5: Verificar la instalación

1. Nos dirigimos a la **Consola (CMD o Terminal)**.
2. Escribimos el siguiente comando: **java --version**
3. Presionamos **Enter** y verificamos que aparezca la versión de Java instalada correctamente.

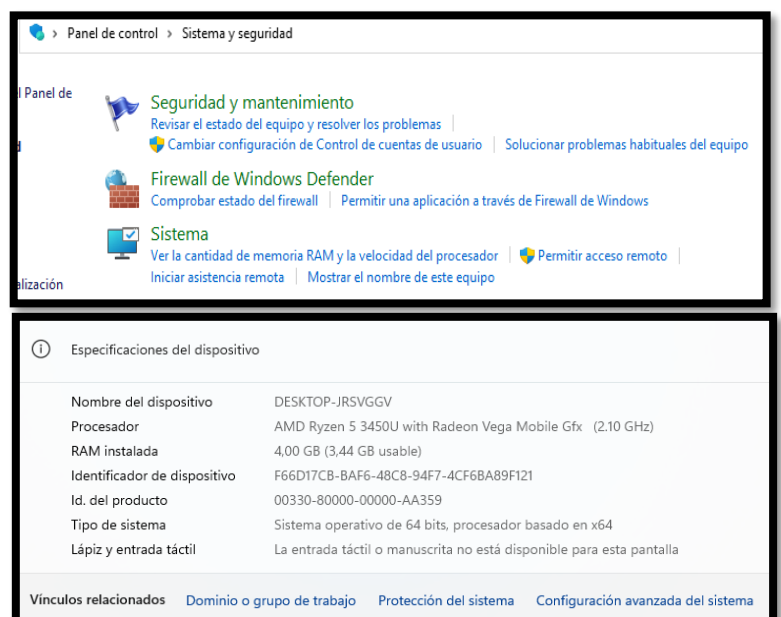
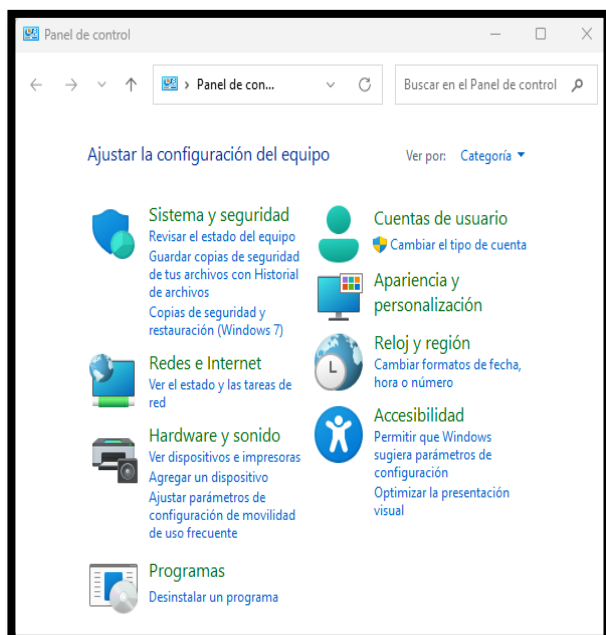


```
C:\Users\Usuario>java --version
java 21.0.8 2025-07-15 LTS
Java(TM) SE Runtime Environment (build 21.0.8+12-LTS-250)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.8+12-LTS-250, mixed mode, sharing)

C:\Users\Usuario>
```

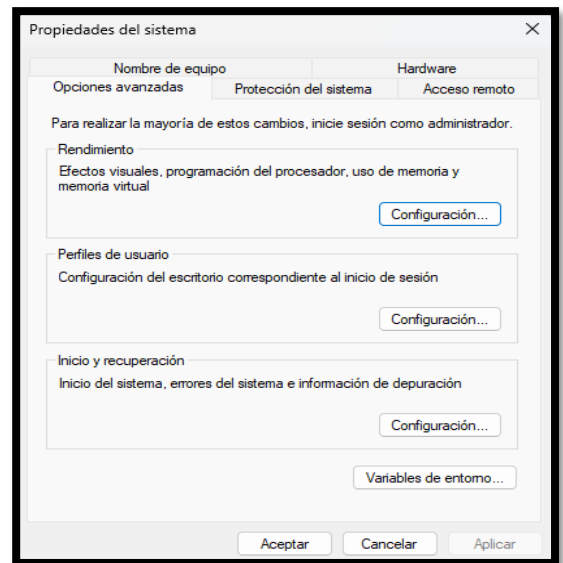
Paso 6: Acceder a la configuración avanzada del sistema

1. Nos dirigimos al **Panel de control**.
2. Seleccionamos **Sistema y seguridad**.
3. Hacemos clic en **Sistema**.
4. Nos dirigimos a **Configuración avanzada del sistema**.



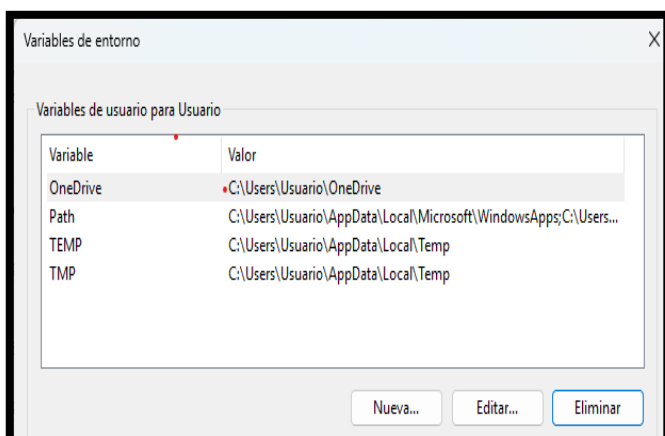
Paso 7: Acceder a Variables de entorno

1. Dentro de **Configuración avanzada del sistema**, se desplegará un cuadro de diálogo que dice **Propiedades del sistema**.
2. Una vez allí, nos dirigimos a **Variables de entorno**.
3. Aparecerá un cuadro con dos secciones: **Variables de usuario para [Usuario]** y **Variables del sistema**.



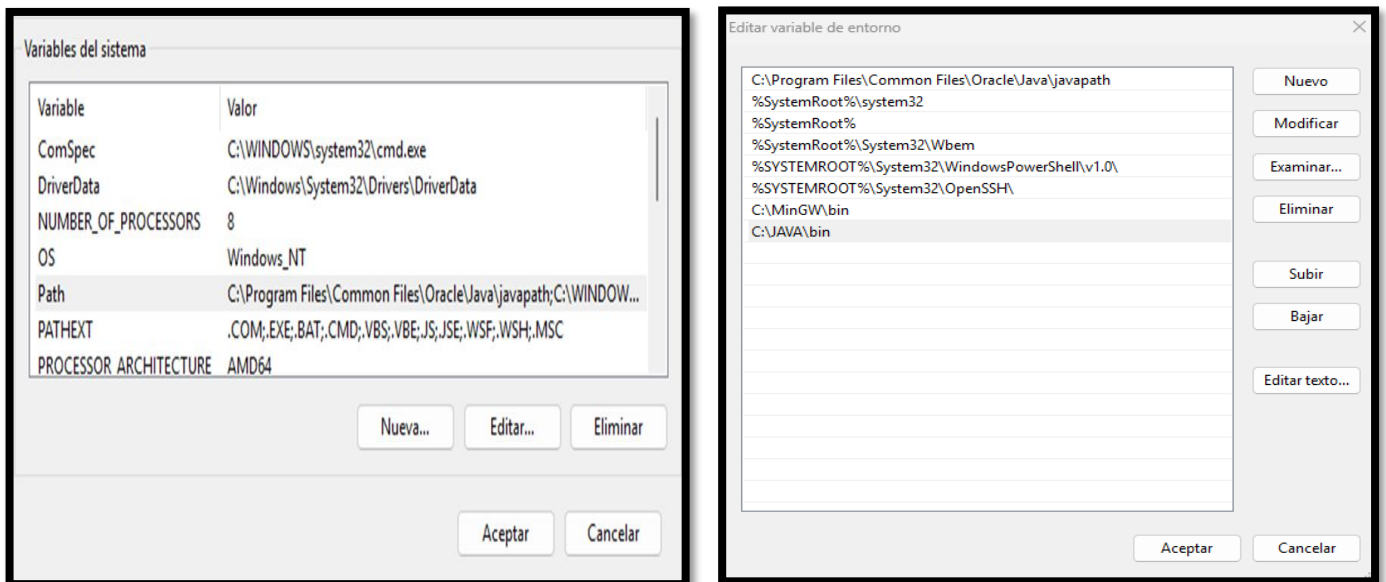
Paso 8: Configurar variables para usuario

1. Nos dirigimos a **Variables de usuario** para nuestro usuario.
2. Hacemos clic en **Nueva**.
3. En **Nombre de la variable**, escribimos: `JAVA_HOME`
4. En **Valor de la variable**, escribimos la ruta donde instalamos Java.
 - Para obtenerla, abrimos la carpeta donde instalamos Java (`C:\JAVA`) y copiamos su dirección desde el explorador de archivos.
 - La ruta debería verse así: `C:\JAVA`
5. Guardamos los cambios haciendo clic en **Aceptar**.



Paso 9: Configurar la variable del sistema

1. Nos dirigimos a **Variables del sistema**.
2. Seleccionamos la variable **Path** y hacemos clic en **Editar**.
3. Hacemos clic en **Nueva**.
4. Ingresamos la ruta completa de la carpeta **bin** de Java.
 - Para obtenerla, abrimos la carpeta donde instalamos Java (C:\JAVA), entramos a la carpeta **bin** y copiamos su dirección desde el explorador de archivos.
 - La ruta que pegamos debería verse así: C:\JAVA\bin
5. Guardamos los cambios haciendo clic en **Aceptar** en todas las ventanas abiertas.



Paso 10: Verificar la instalación de Java

1. Abrimos el **Símbolo del sistema (CMD)**.
2. Ejecutamos los siguientes comandos para comprobar la instalación:
 - java -version → Muestra la versión de Java instalada.
 - javac -version → Muestra la versión del compilador de Java.
3. Si ambos comandos muestran la versión correspondiente, significa que Java está correctamente instalado y configurado.

```

Microsoft Windows [Versión 10.0.26100.6584]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Usuario> java --version
java 21.0.8 2025-07-15 LTS
Java(TM) SE Runtime Environment (build 21.0.8+12-LTS-250)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.8+12-LTS-250, mixed mode, sharing)

C:\Users\Usuario> java -version
java version "21.0.8" 2025-07-15 LTS
Java(TM) SE Runtime Environment (build 21.0.8+12-LTS-250)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.8+12-LTS-250, mixed mode, sharing)

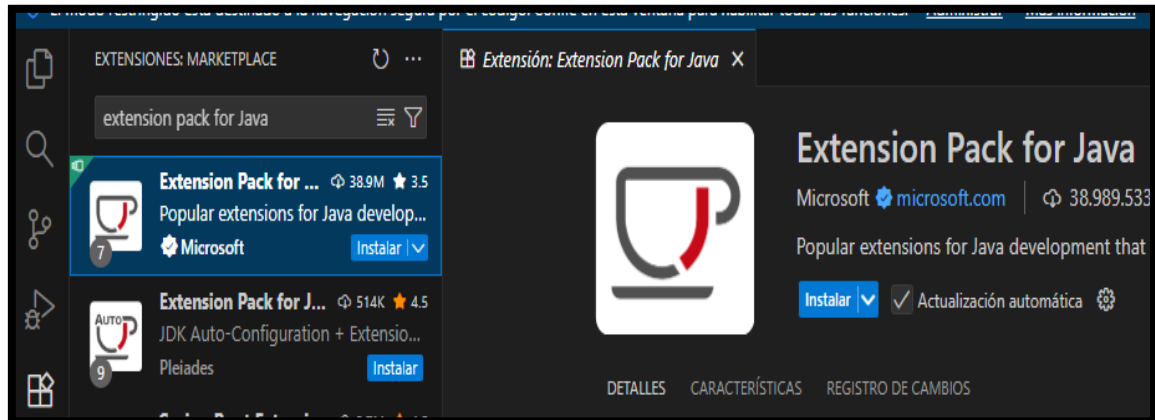
C:\Users\Usuario> javac -version
javac 21.0.8

C:\Users\Usuario>

```

Paso 11: Instalar extensión Pack for Java en Visual Studio Code

1. Abrimos **Visual Studio Code**.
2. Nos dirigimos a la sección de **Extensiones**.
3. En la barra de búsqueda, escribimos: **Extension Pack for Java**
4. Seleccionamos la extensión **Extension Pack for Java** desarrollada por Microsoft.
5. Hacemos clic en **Instalar** y esperamos a que la instalación finalice.
6. Una vez instalada, reiniciamos Visual Studio Code para que todos los cambios tengan efecto.



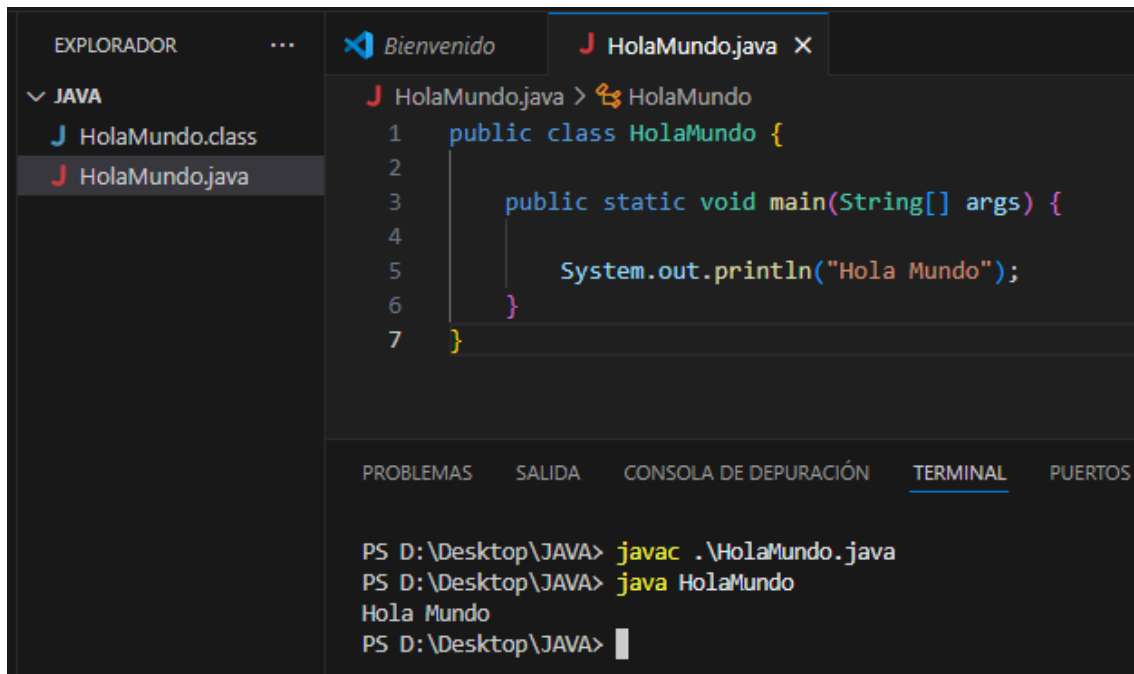
7. Código

En el siguiente código se explica lo siguiente:

- Declarar la **clase principal**.
- Definir el **método main**, que es el punto de entrada del programa.
- Imprimir un mensaje en la consola usando **System.out.println**.

```
public class HolaMundo {  
  
    public static void main(String[] args) {  
  
        System.out.println("Hola Mundo");  
  
    }  
  
}
```

La ejecución del código es el siguiente:



The screenshot shows an IDE with a dark theme. On the left, the 'EXPLORADOR' (Explorer) pane shows a project named 'JAVA' containing two files: 'HolaMundo.class' and 'HolaMundo.java'. The main editor area has a tab for 'HolaMundo.java' and displays the following code:

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

Below the editor, the 'TERMINAL' pane shows the execution commands and output:

```
PS D:\Desktop\JAVA> javac .\HolaMundo.java  
PS D:\Desktop\JAVA> java HolaMundo  
Hola Mundo  
PS D:\Desktop\JAVA> 
```

8. Conclusiones

Se llegó a las siguientes conclusiones:

- Los lenguajes de programación son herramientas esenciales en la resolución de problemas computacionales, ya que permiten traducir el razonamiento humano en instrucciones precisas que la computadora puede ejecutar. A través de ellos se logra estructurar, analizar y automatizar soluciones a diversas necesidades tecnológicas, facilitando el desarrollo de sistemas eficientes y adaptados a

distintos contextos. Además, fomentan el pensamiento lógico y la capacidad de descomponer problemas complejos en pasos más simples y manejables.

- La importancia de los lenguajes de programación radica también en su contribución al avance del conocimiento y la innovación. Gracias a ellos, es posible crear soluciones que optimizan procesos, reducen errores humanos y mejoran la toma de decisiones en múltiples áreas, como la educación, la salud, la industria y la comunicación. Reflexionar sobre su valor nos permite entender que aprender a programar no solo implica dominar una herramienta técnica, sino también adquirir una forma de pensar orientada a la resolución efectiva de problemas y a la mejora continua del entorno tecnológico.

9. Referencias

[1] J. Gosling, "A Review on Java Programming Language," *ResearchGate*, 2023. [En línea]. Disponible en:

https://www.researchgate.net/publication/371166744_A_Review_on_Java_Programming_Language

[2] Instituto Tecnológico Quito, *Dominando Java I*, 2023. [En línea]. Disponible en: https://itq.edu.ec/wp-content/uploads/2023/10/2023-09-29_dominando_java_i.pdf

[3] J. C. Gutiérrez y M. R. Torres, "Java y su relevancia en el desarrollo de software moderno," *Revista Micaela Bastidas*, vol. 7, no. 2, pp. 45–58, 2023. [En línea]. Disponible en: <https://revistas.unamba.edu.pe/index.php/micaela/article/view/143/180>

[4] J. Gosling, "A Review on Java Programming Language," *ResearchGate*, 2023. [En línea]. Disponible en:

https://www.researchgate.net/publication/371166744_A_Review_on_Java_Programming_Language

[5] P. Ramírez, "Análisis de los lenguajes orientados a objetos: ventajas y desafíos de Java," *Revista Aristas*, vol. 5, no. 1, pp. 12–20, 2023. [En línea]. Disponible en: http://revistaaristas.tij.uabc.mx/index.php/revista_aristas/article/view/290/293

Declaración de uso de la IA en la tarea

Para la elaboración de este informe se utilizaron herramientas de inteligencia artificial (IA) con fines de apoyo en la redacción, organización de ideas y revisión de estilo