 **Universidad Mariano Gálvez**

**Facultad de Ingeniería en Sistemas de Información**

**PROGRAMACION I**

**PARICAL I**

**01 de MAR. de 2025**

**Nombre: Emanuel José Jiménez Cojolón**

**Carné: 0905-23-10682**

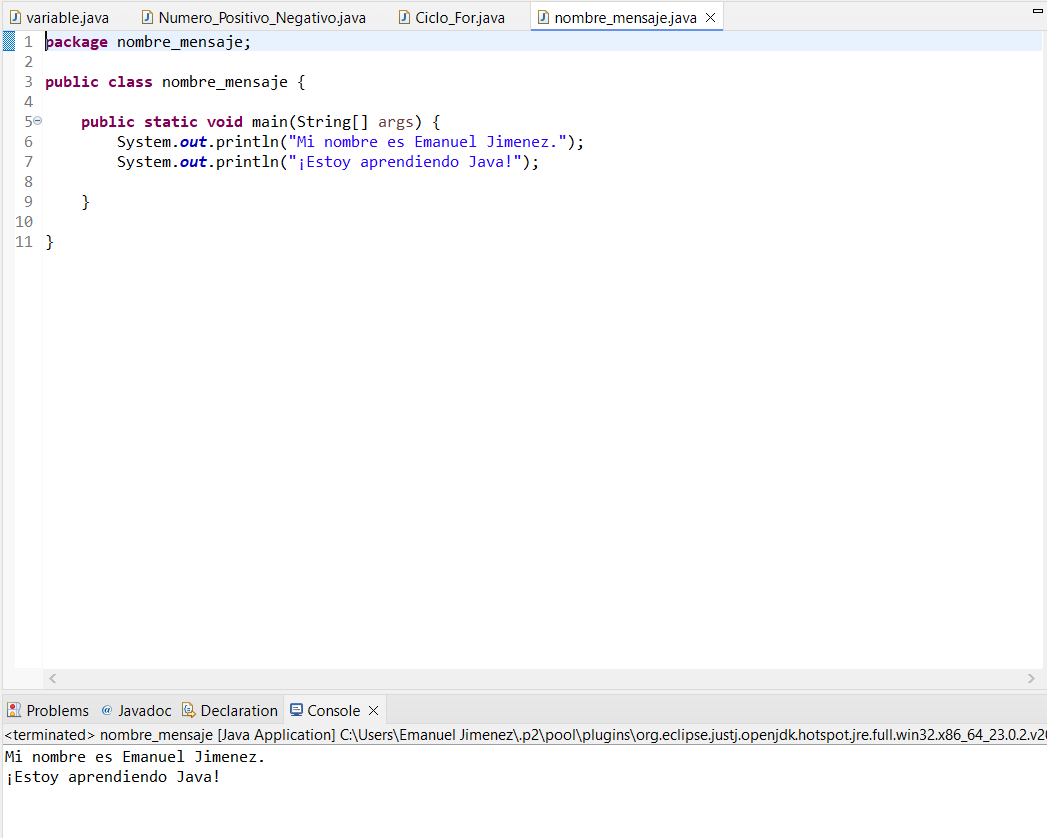
***SERIE I: Deberá dar una explicación clara y concisa de cada uno de los temas que se plantean a continuación, respalde su explicación con un ejemplo práctico. Redacte una presentación para responder a cada pregunta.***

1. **Fundamentos de Java a) Explica en tus palabras qué es Java y para qué se utiliza. b)**

**Escribe un programa en Java que imprima tu nombre y el mensaje: "¡Estoy aprendiendo Java!".**

Java es un lenguaje de programación de alto nivel, orientado a objetos, creado por Sun Microsystems en 1995 (ahora parte de Oracle). Es conocido por su portabilidad, ya que los programas en Java pueden ejecutarse en cualquier dispositivo que tenga instalada la Máquina Virtual de Java (JVM).

Se utiliza para desarrollar aplicaciones en diversas áreas como:

* **Aplicaciones de escritorio**: Programas que se ejecutan en computadoras personales.
* **Aplicaciones web**: Servicios accesibles a través de navegadores.
* **Aplicaciones móviles**: Principalmente en dispositivos Android.
* **Sistemas embebidos**: Software para dispositivos electrónicos específicos.

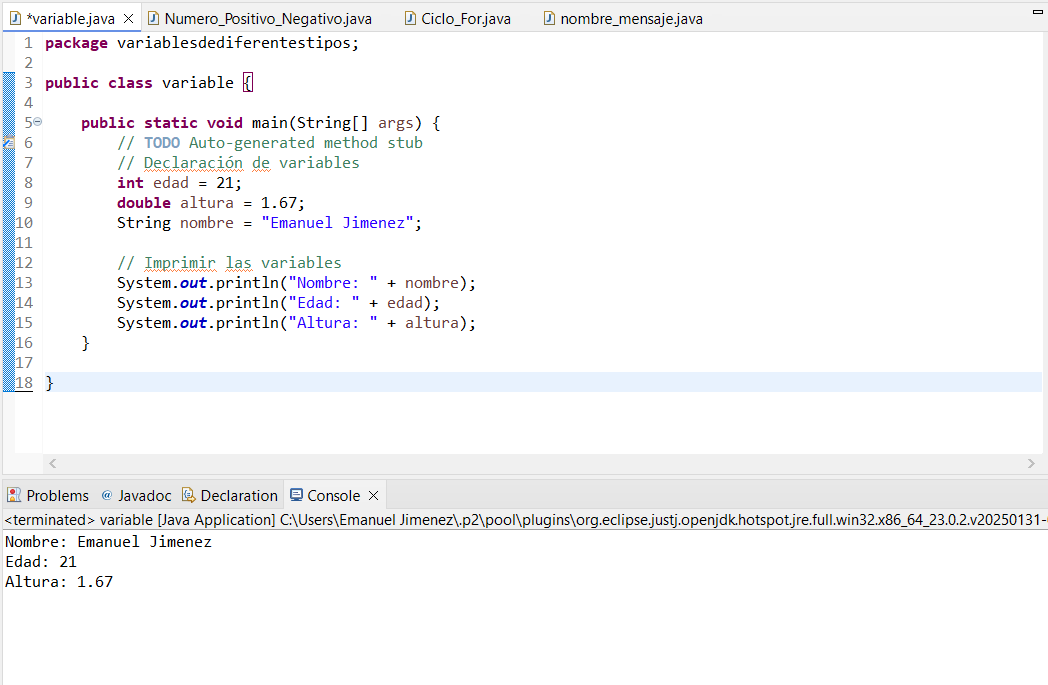
1. **Variables y tipos de datos a) Investiga qué son las variables en Java y menciona tres tipos de datos con ejemplos. b)**

**Escribe un programa que declare tres variables de diferentes tipos y las imprima en pantalla.**

En Java, una variable es un contenedor que almacena un valor. Cada variable tiene un tipo de dato que define qué clase de valor puede contener.

**Tipos de datos en Java:**

1. **int**: Almacena números enteros (Ejemplo: 1, 45, -23).
2. **double**: Almacena números decimales (Ejemplo: 3.14, -0.001).
3. **String**: Almacena cadenas de texto (Ejemplo: "Hola", "Java").

****

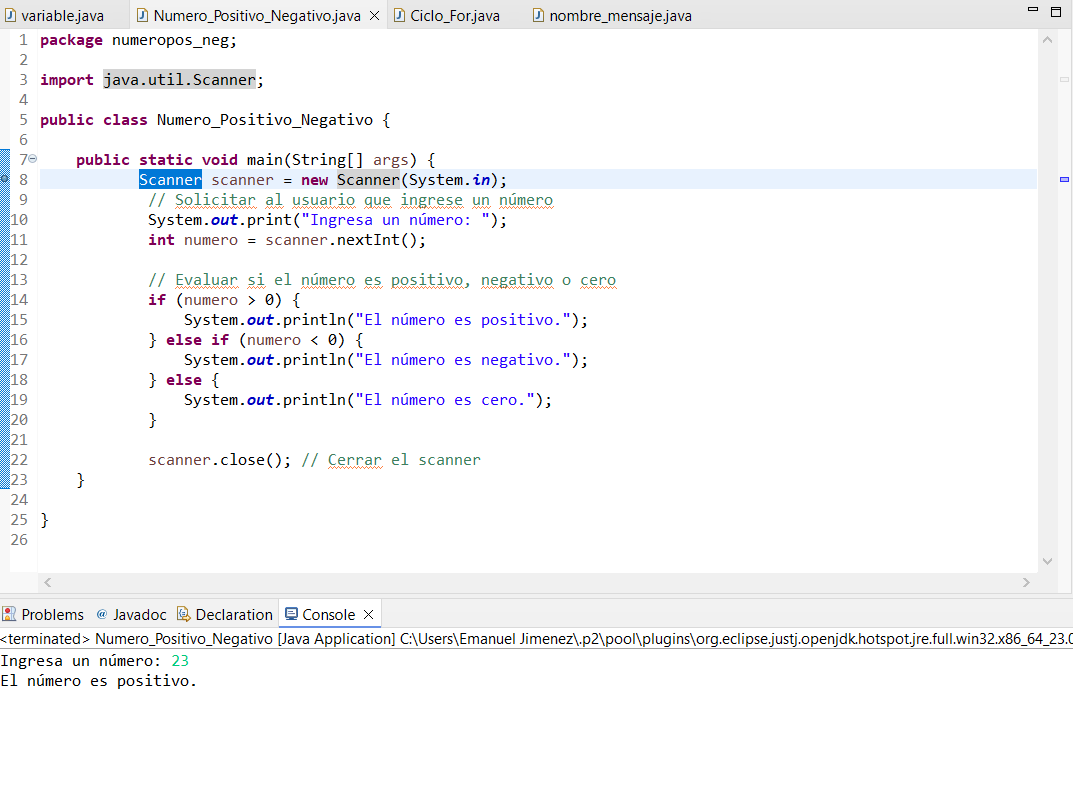
1. **Condiciones y estructuras de control a) ¿Qué son las estructuras de control en Java? Explica if, else if, else. b)**

**Escribe un programa que solicite al usuario ingresar un número y determine si es positivo, negativo o cero.**

Las estructuras de control permiten que un programa tome decisiones y altere el flujo de ejecución en función de condiciones específicas.

**Estructuras de control if, else if, else:**

* **if**: Ejecuta un bloque de código si una condición es verdadera.
* **else if**: Evalúa condiciones adicionales si el if es falso.
* **else**: Ejecuta un bloque de código si ninguna de las condiciones anteriores se cumple.



1. **Ciclos e iteraciones a) Explica la diferencia entre los ciclos for, while y do-while con ejemplos. b)**

**Escribe un programa que imprima los números del 1 al 10 usando un ciclo for.**

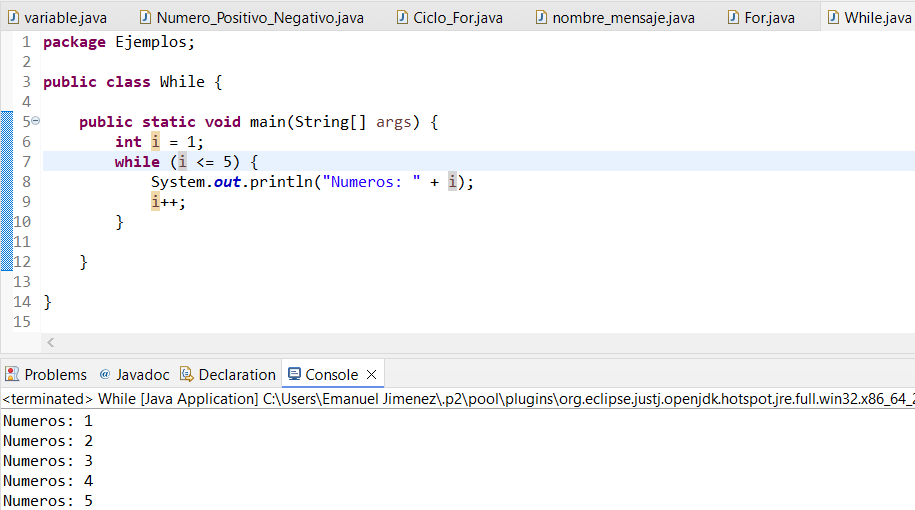
FOR

Se usa cuando sabemos cuántas veces se ejecutara el bloque de código, también cuenta con tres partes; Inicialización, condición y actualización



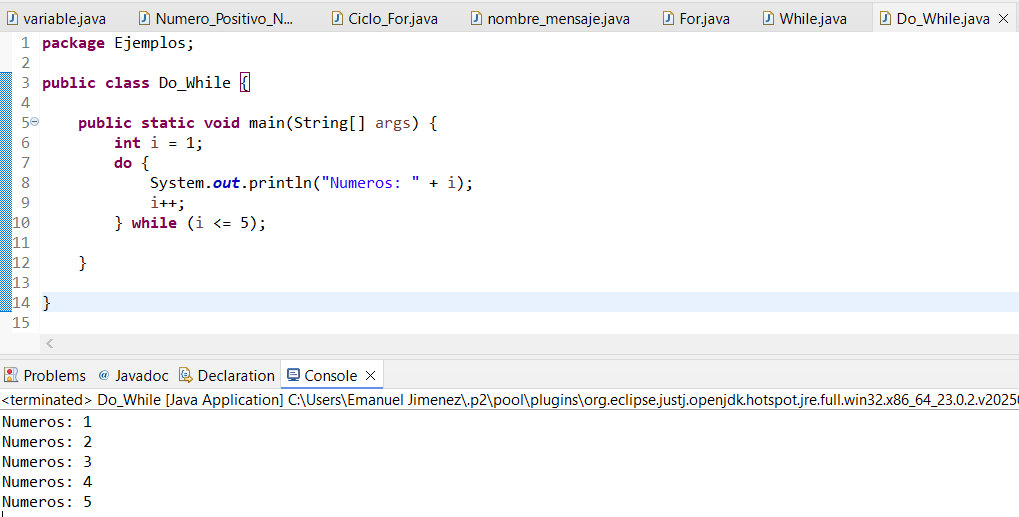
Wile

Se utiliza cuando no sabemos cuántas veces se repetirá el bloque, pero necesitamos verificar una condición antes de cada iteración.



Do-While

Similar al While, pero garantiza que el bloque de código se ejecutará al menos una vez



***SERIE II: Tome en cuenta el código de la primera serie para usarlo en esta práctica.***

* 1. **¿Qué es Git y para qué se usa?**

Git es un sistema de control de versiones distribuido que permite a los desarrolladores rastrear los cambios en su código fuente, colaborar de manera eficiente y mantener un historial detallado de todas las modificaciones. Fue creado por Linus Torvalds en 2005 para el desarrollo del kernel de Linux.

**¿Para qué se usa Git?**

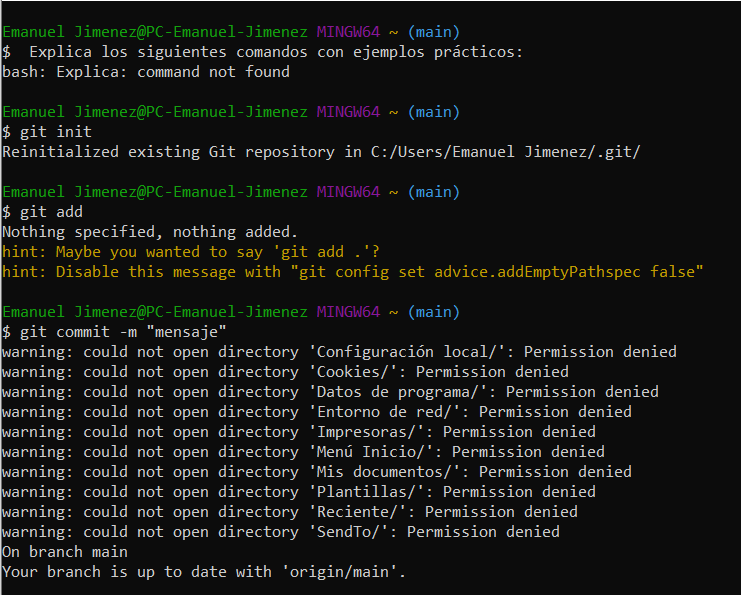
1. **Control de versiones**: Permite llevar un registro de cada cambio realizado en el código, facilitando la reversión a versiones anteriores si es necesario.
2. **Colaboración**: Varios desarrolladores pueden trabajar simultáneamente en el mismo proyecto sin conflictos, utilizando ramas (branches) para separar funcionalidades.
3. **Historial de cambios**: Mantiene un historial completo de todos los cambios, con detalles como el autor, fecha y descripción de la modificación.
4. **Trabajo distribuido**: Cada desarrollador tiene una copia completa del proyecto, lo que permite trabajar sin conexión y sincronizar los cambios más tarde.

**Comandos básicos de Git:**

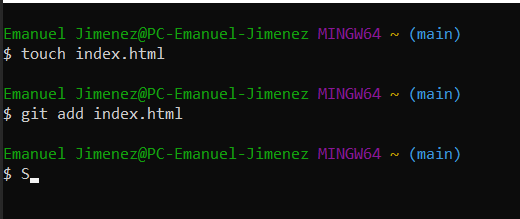
* git init: Inicializa un nuevo repositorio Git.
* git clone [URL]: Clona un repositorio existente desde una URL.
* git add [archivo]: Agrega un archivo al área de preparación.
* git commit -m "mensaje": Guarda los cambios en el historial con un mensaje descriptivo.
* git status: Muestra el estado actual del repositorio.
* git push: Envía los cambios al repositorio remoto.
* git pull: Descarga y fusiona los cambios del repositorio remoto.
  1. **Explica los siguientes comandos con ejemplos prácticos:**
* **Git init**
* **git add**
* **git commit -m "mensaje"**
* **git status**
* **git log**

1. GIT INIT

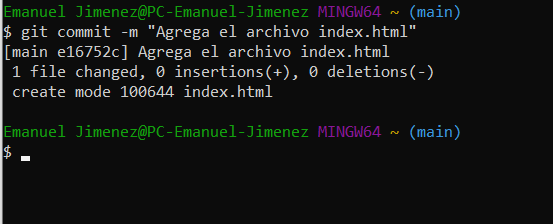
Este comando inicializa un nuevo repositorio GIT en un directorio. Es el primer paso para rastrear los cambios de un proyecto.



1. GIT ADD

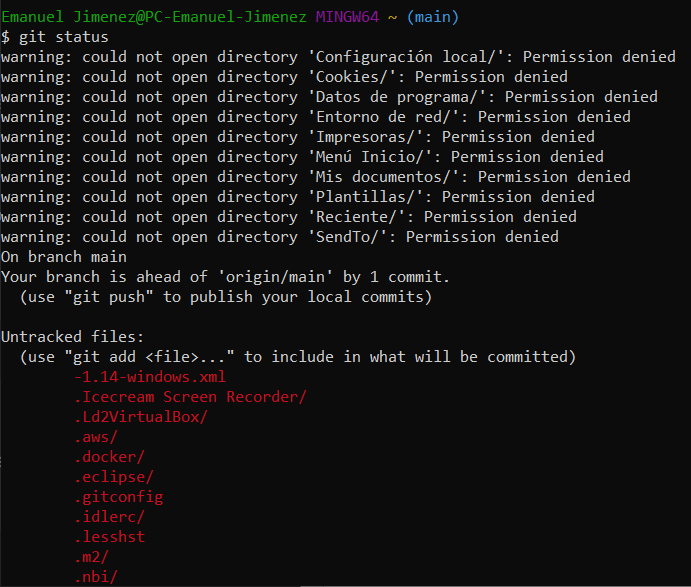
Este comando agrega archivos al área de preparación (staging area), lo que indica que están listos para ser confirmados (committed).

1. GIT COMMIT –M “MENSAJE ”

Registra los cambios del área de preparación en el historial del repositorio. El mensaje describe los cambios realizados.

1. GIT STATUS

Muestra el estado actual del repositorio: que archivo están modificados en el área de preparación o no rastreados.



1. GIT LOG

Muestra el historial de confirmaciones (Commits) del repositorio.



* 1. ***Que es GitHub y cuál es su relación con Git?***

GitHub es una plataforma de desarrollo colaborativo que permite alojar repositorios Git en la nube. Proporciona herramientas para el control de versiones, la revisión de código, la gestión de proyectos y la colaboración en equipo.

### Relación entre Git y GitHub:

* **Git** es un sistema de control de versiones que se utiliza localmente para rastrear cambios en el código.
* **GitHub** es un servicio en línea que permite almacenar y compartir repositorios Git.

GitHub facilita la colaboración en proyectos al permitir que múltiples desarrolladores trabajen en el mismo código, gestionen cambios y resuelvan conflictos. También ofrece funciones como:

* **Pull requests**: Propuestas para fusionar cambios en el código.
* **Issues**: Registro de errores y tareas pendientes.
* **Acciones de GitHub**: Automatización de flujos de trabajo (CI/CD).