INFORME SOBRE GIT

Johan Fernando González Acuña

Centro de Diseño y Metrología

Codificar el Software de Acuerdo con el Diseño

Julio Roberto Galvis

Bogotá D.C

23/10/2024

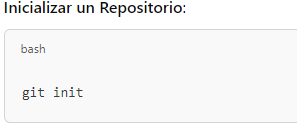
**¿Qué es Git?**

Git es un sistema de control de versiones distribuido que permite a múltiples desarrolladores trabajar en un proyecto de manera simultánea. Fue creado por Linus Torvalds en 2005 y se ha convertido en una de las herramientas más populares para el desarrollo de software.

### Funciones Principales de Git

1. **Control de Versiones**: Permite rastrear los cambios en el código a lo largo del tiempo, facilitando la gestión de diferentes versiones de un proyecto.
2. **Ramas (Branches)**: Git permite crear ramas para trabajar en nuevas características o correcciones sin afectar la versión principal del código. Esto ayuda a mantener el trabajo organizado y permite desarrollar nuevas funcionalidades de forma aislada.
3. **Fusión (Merge)**: Una vez que se ha completado el trabajo en una rama, se puede fusionar con la rama principal (generalmente main o master). Git ofrece herramientas para resolver conflictos que puedan surgir durante este proceso.
4. **Deshacer Cambios**: Git proporciona varias maneras de deshacer cambios, como git checkout para revertir a una versión anterior o git reset para deshacer commits recientes.
5. **Historial de Commits**: Cada cambio en el proyecto se registra con un "commit", que incluye un mensaje descriptivo. Esto permite a los desarrolladores ver el historial del proyecto y entender qué cambios se realizaron y por qué.
6. **Colaboración**: Git facilita la colaboración entre múltiples desarrolladores, permitiendo que cada uno trabaje en su propio espacio sin interferir con los demás.
7. **Almacenamiento Remoto**: Git puede trabajar con repositorios remotos, lo que permite compartir y sincronizar el código con otros desarrolladores a través de plataformas como GitHub, GitLab o Bitbucket.

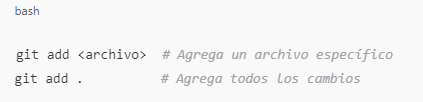
**Flujo de Trabajo Básico con Git:**

****

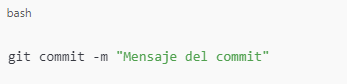
**Clonar un Repositorio**:

****

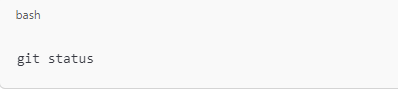
**Agregar Cambios**: Después de realizar cambios en los archivos, se deben agregar al área de preparación:

****

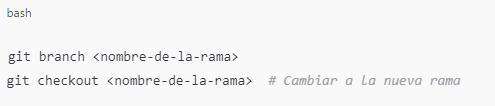
**Realizar un Commit**: Guardar los cambios en el repositorio local:

****

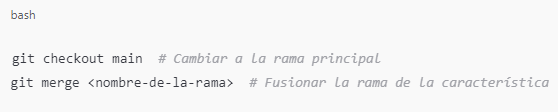
**Verificar el Estado del Repositorio**: Para ver los archivos modificados y el estado actual:

****

**Crear una Nueva Rama**: Para desarrollar una nueva característica:

****

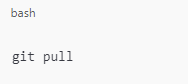
**Fusionar Ramas**: Una vez completada la nueva característica:

****

**Subir Cambios a un Repositorio Remoto**: Para compartir los cambios con otros:

****

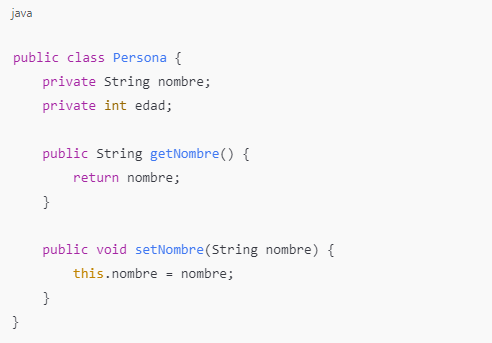
**Actualizar el Repositorio Local**: Para obtener los últimos cambios del repositorio remoto:

****

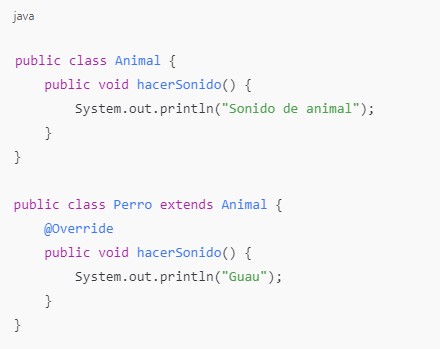
### ANEXO, Programación orientada a objetos aplicada en JAVA: ¿Qué es la Programación Orientada a Objetos (POO)?

La Programación Orientada a Objetos es un paradigma de programación que utiliza "objetos" para representar datos y métodos que operan sobre esos datos. Este enfoque facilita la organización y gestión del código, promoviendo la reutilización y la escalabilidad.

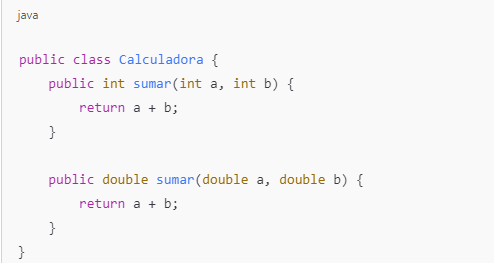
**Principios Fundamentales de la POO**

1. **Encapsulamiento**:
   * Consiste en ocultar los detalles internos de un objeto y exponer solo lo necesario a través de una interfaz pública. Esto se logra mediante modificadores de acceso (como private, public y protected).
   * 

**Herencia**:

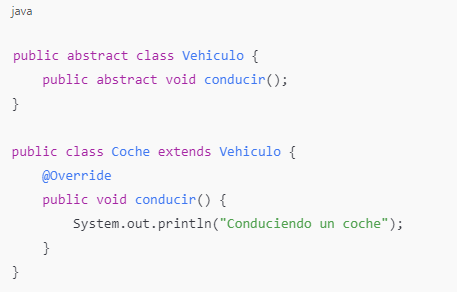
1. Permite crear nuevas clases a partir de clases existentes. La nueva clase (subclase) hereda atributos y métodos de la clase base (superclase), promoviendo la reutilización del código.
   * 

**Polimorfismo**:

1. Permite que una variable, función o método tenga múltiples formas. En Java, esto se puede lograr a través de la sobrecarga de métodos y la sobrescritura de métodos.
2. 

**Abstracción**:

* Es el proceso de simplificar la complejidad al ocultar los detalles innecesarios y mostrar solo la funcionalidad relevante. En Java, esto se puede lograr mediante clases abstractas e interfaces.



**Ventajas de la POO en Java**

* **Reutilización de Código**: Las clases se pueden reutilizar y extender, reduciendo la duplicación del código.
* **Facilidad de Mantenimiento**: El encapsulamiento permite hacer cambios internos sin afectar a otras partes del programa.
* **Modularidad**: La POO promueve la creación de componentes modulares, facilitando el trabajo en equipo y la colaboración.
* **Claridad y Organización**: Los objetos y clases ayudan a modelar problemas del mundo real de manera más intuitiva.



### Conclusiones

Git es una herramienta fundamental para el desarrollo de software moderno, que permite a los equipos gestionar su trabajo de manera eficiente y colaborativa. Su capacidad para manejar versiones, fusionar cambios y trabajar con ramas lo convierte en una opción ideal para proyectos de cualquier tamaño.