UNIVERSIDAD NACIONAL DE LOJA

ASIGNATURA:

Programación Orientada a Objetos

DOCENTE:

Ing. Coronel Romero Edison Leonardo

INTEGRANTES:

- **FERNANDO VASQUEZ**
- **MAYVELIN PINTA**
- **MIGUEL LUNA**

LOJA- ECUADOR

2023 - 2024

«Todo el mundo te dice lo que es mejor para ti. No quieren que busques respuestas, sino que creas las suyas. Deja de coleccionar información del exterior y empieza a buscarla por ti mismo en tu interior.»

TAREA DE PROGRAMACIÓN

1. Realiza un pequeño proyecto de prueba en la herramienta elegida y describe en un documento breve las ventajas y desventajas de la herramienta seleccionada.

TEMA: APACHE NETBEANS

Apache NetBeans es un IDE de desarrollo de código abierto y multiplataforma que brinda soporte para una amplia gama de lenguajes de programación, incluido Java. Tiene una interfaz intuitiva y ofrece numerosas características y funcionalidades que facilitan el desarrollo de aplicaciones.

- **Características principales de Apache NetBeans:**
  - Edición de código inteligente con resaltado de sintaxis y autocompletado.
  - Depuración y pruebas unitarias integradas.
  - Administración de proyectos y bibliotecas.
  - Soporte para el desarrollo web con tecnologías como HTML, CSS y JavaScript.
  - Herramientas para el desarrollo de interfaces de usuario, incluido un diseñador de GUI.
  - Integración con sistemas de control de versiones como Git.
  - Compatibilidad con extensiones y complementos para ampliar sus funcionalidades.
- **Ventajas de Apache NetBeans:**
  - Interfaz de usuario intuitiva y fácil de usar.
  - Amplio conjunto de características y herramientas para el desarrollo Java.
  - Compatibilidad con una variedad de tecnologías y frameworks.
  - Comunidad activa y soporte de Apache Software Foundation.
- **Desventajas de Apache NetBeans:**
  - En comparación con otras opciones como IntelliJ IDEA, puede tener una menor cantidad de complementos disponibles.
  - Puede tener un rendimiento ligeramente inferior al trabajar con proyectos grandes o complejos.
- **Funcionalidades Apache NetBeans**
  - **Edición de código inteligente:** NetBeans ofrece un editor de código potente y amigable que proporciona funciones de resaltado de sintaxis, autocompletado y corrección automática. Además, cuenta con herramientas de refactorización que facilitan la modificación y reestructuración del código.
  - **Depuración y pruebas unitarias:** NetBeans incluye un depurador integrado que permite rastrear y solucionar problemas en el código paso a paso. También proporciona soporte para ejecutar y crear pruebas unitarias, lo que facilita la detección de errores y el desarrollo de código más robusto.
  - **Administración de proyectos y bibliotecas:** NetBeans facilita la creación y gestión de proyectos Java. Proporciona asistentes para crear nuevos proyectos, importar proyectos existentes y configurar bibliotecas y dependencias. Además, ofrece una vista estructurada de los proyectos para una fácil navegación y organización de los archivos.
  - **Desarrollo web:** NetBeans incluye herramientas para el desarrollo web con soporte para HTML, CSS y JavaScript. Proporciona un editor web con funciones de autocompletado y resaltado de sintaxis, así como un depurador web para depurar aplicaciones web en el navegador.
  - **Diseño de interfaces gráficas:** NetBeans cuenta con un diseñador de interfaces gráficas (GUI Builder) que permite crear interfaces de usuario arrastrando y soltando

componentes visuales. Esto facilita la creación rápida y visual de interfaces de usuario para aplicaciones de escritorio.

- **Integración con sistemas de control de versiones:** NetBeans tiene integración nativa con sistemas de control de versiones como Git. Permite realizar operaciones básicas de control de versiones, como confirmar cambios, actualizar el repositorio y fusionar ramas. Además, proporciona una interfaz gráfica para administrar el control de versiones dentro del IDE.
- **Extensiones y complementos:** NetBeans es altamente extensible a través de complementos y extensiones. Puedes agregar funcionalidades adicionales mediante la instalación de complementos desde el Centro de Actualizaciones de NetBeans. Hay una amplia variedad de complementos disponibles, desde herramientas de análisis de código hasta integración con frameworks específicos.
- **Ejemplo:**

```
1
2    package com.mycompany.sumar;
3
4    import java.util.*;
5
6
7    /**
8     *
9     * @author mayvelinpuglla, Miguel Luna, Fernando Vasquez
10    */
11   public class SumarDosNumeros {
12
13       public static void main (String[] args)
14       {
15           double num1,num2,resultado;
16
17           Scanner Teclado = new Scanner (System.in);
18
19           System.out.println ("Ingresar  numero 1");
20           num1=Teclado.nextDouble();
21
22           System.out.println ("Ingresar numero 2");
23           num2=Teclado.nextDouble();
24
25           resultado=num1+num2;
26
             System.out.println("El resultado de la suma es: "+resultado);
28
29
30       }
31
32       }
33
```

Resultado:

```
Ingresar  numero 1
138
Ingresar numero 2
210
El resultado de la suma es: 348.0
```

2. **Explica cómo funciona el control de versiones y cómo puede ayudar en el desarrollo colaborativo de software.**
   El control de versiones es un sistema que registra y administra los cambios realizados en los archivos de código fuente de un proyecto de software a lo largo del tiempo. En el desarrollo de software orientado a objetos, se utilizan diferentes archivos de código que contienen las clases, métodos y atributos que definen los objetos y su comportamiento.
   Existen diversas herramientas de control de versiones, siendo Git una de las más populares. Y funciona de la siguiente manera:

- **Repositorio:** El control de versiones se basa en la idea de tener un repositorio centralizado que almacene todos los archivos del proyecto, incluyendo su historial de cambios.
- **Inicialización del repositorio:** Para comenzar a utilizar el control de versiones en un proyecto, se inicializa un repositorio en la ubicación del proyecto. Esto crea una estructura interna en la cual se almacenarán los archivos y se registrará el historial de cambios.
- **Commits:** Un commit, también conocido como confirmación, es una acción que guarda una instantánea del estado actual de los archivos en el repositorio. Al realizar un commit, se registra la versión actual de los archivos y se almacena un mensaje descriptivo que indica los cambios realizados.
- **Ramas:** El desarrollo colaborativo a menudo involucra trabajar en diferentes aspectos del proyecto al mismo tiempo. Las ramas permiten crear líneas de desarrollo independientes para abordar características o correcciones de errores sin afectar la versión principal del proyecto. Cada rama tiene su propio conjunto de commits y se puede fusionar con otras ramas cuando los cambios están listos para ser incorporados al proyecto principal.
- **Fusiones:** La capacidad de fusionar ramas es una característica clave del control de versiones que facilita el trabajo colaborativo en un proyecto. Cuando se completa una función o corrección en una rama, se puede fusionar con otra rama, como la rama principal, para incorporar los cambios. Esto asegura que el proyecto principal se actualice con las últimas modificaciones realizadas por diferentes colaboradores.
- **Resolución de conflictos:** En el desarrollo colaborativo, es posible que dos o más personas realicen cambios en la misma parte de un archivo. Esto puede generar conflictos durante la fusión. El control de versiones proporciona herramientas para resolver estos conflictos, permitiendo a los colaboradores revisar y combinar manualmente los cambios, o bien, aceptar una versión específica de los cambios realizados.
- **Historial y trazabilidad:** El control de versiones mantiene un historial completo de todos los cambios realizados en el proyecto. Esto permite a los desarrolladores rastrear quién hizo qué cambios, cuándo y por qué. La trazabilidad del historial es valiosa para identificar errores, revertir cambios no deseados o comprender la evolución del proyecto a lo largo del tiempo.
- **Etiquetas o tags:** Además de los commits, las herramientas de control de versiones permiten etiquetar o marcar puntos específicos en la historia del proyecto. Las etiquetas son útiles para marcar versiones estables, hitos importantes o lanzamientos específicos del software. Proporcionan una forma rápida y fácil de acceder a versiones específicas y brindan una referencia clara de los puntos clave en el historial del proyecto.
- **Gestión de ramas:** Las ramas no solo se utilizan para desarrollar características o corregir errores, sino también para gestionar diferentes entornos, como pruebas, desarrollo y producción. Por ejemplo, puedes tener una rama "develop" para integrar y probar características nuevas y una rama "master" para la versión estable y desplegarla en

producción. Esto permite un flujo de trabajo más organizado y garantiza que los cambios se prueben antes de su lanzamiento.

- **Control de acceso y permisos:** Las herramientas de control de versiones permiten definir niveles de acceso y permisos para los colaboradores del proyecto. Esto es importante en proyectos colaborativos, ya que no todos los desarrolladores pueden tener los mismos derechos de modificación y fusión. Se pueden establecer restricciones para garantizar que los cambios se revisen antes de fusionarlos en la rama principal, lo que ayuda a mantener la calidad y la integridad del código base.
- **Integración continua y entrega continua:** El control de versiones se combina a menudo con prácticas como la integración continua (CI) y la entrega continua (CD). La CI implica fusionar y probar regularmente los cambios en una rama compartida, lo que ayuda a identificar rápidamente los conflictos y los errores. La CD automatiza la entrega del software a través de la construcción, prueba y despliegue automatizados. El control de versiones proporciona la base para estas prácticas, permitiendo una gestión eficiente de los cambios y una trazabilidad clara de los resultados de las pruebas y los despliegues.
- **Colaboración distribuida:** Una ventaja importante del control de versiones es su capacidad para facilitar la colaboración distribuida. Los colaboradores pueden trabajar en el proyecto desde diferentes ubicaciones geográficas y realizar cambios independientes en sus propias copias locales. Luego, pueden sincronizar y fusionar sus cambios en el repositorio central, lo que permite la colaboración en tiempo real y el desarrollo conjunto sin conflictos.
- **Revertir cambios:** El control de versiones permite revertir cambios no deseados o errores. Si se introduce un error crítico o se realizan modificaciones que no funcionan como se esperaba, es posible retroceder en el historial y restaurar una versión anterior del proyecto. Esto proporciona una forma segura de deshacer cambios y volver a un estado funcional anterior.

  Así mismo, el control de versiones proporciona una forma estructurada de administrar y rastrear los cambios en los archivos de código fuente. Permite trabajar en diferentes aspectos del proyecto de manera simultánea, fusionar cambios de manera eficiente, resolver conflictos y mantener un historial completo de versiones. Estas características hacen que el control de versiones sea esencial para mejorar la colaboración y la gestión del código en proyectos de software orientados a objetos.

REFERENCIAS:

Calendamaia," Genbeta",9 Enero 2014 https://www.genbeta.com/desarrollo/netbeans-1.

OpenAI. (2021). ChatGPT. Retrieved from https://openai.com/

LINK REPOSITORIO:

https://github.com/Miguelin04/Seguimiento-Competencias-Deportivas.git