Laboratorio #1

Git, debug, plug-in, & JUnit Parte I - Pre-laboratorio

1. Cuestionario

- **Defina la diferencia entre un Sistema de Control de Versiones Centralizado y Distribuido.** (1 punto)

  R:

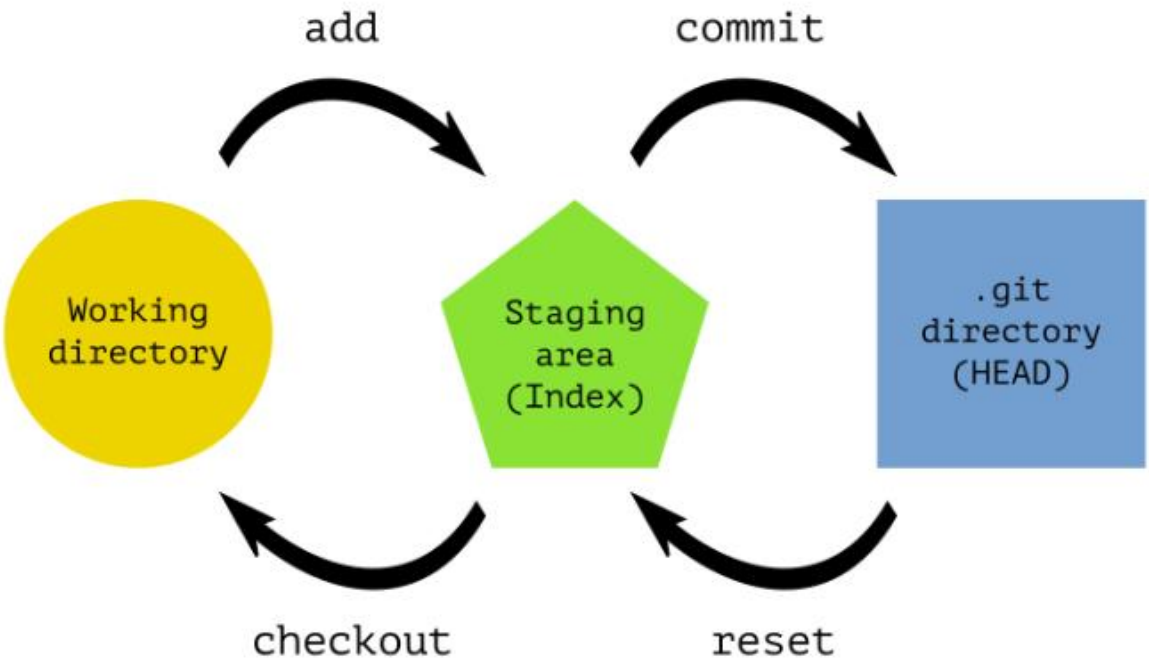
Centralizado	Distribuido
El sistema servidor es un repositorio, como los que mantienen los clientes, pero perfectamente sincronizado y sin que dé lugar a conflictos. Es la copia maestra de los datos.	Dispone de forma distribuida la información del repositorio al completo, tanto de forma local, como a través de los demás componentes del grupo.
Cuando un sistema web quiere hacer un listado, puede tomar los datos de este servidor y siempre serán seguros, por lo que no tendrá que resolver conflictos, ni tendrá que hacer mezclas.	Cada cambio se va replicando entre los demás equipos distribuidos, a modo de que puedan emplear esos datos y actualizarlos en sus sistemas.
Una copia local debe de poder mezclarse con el repositorio central cuando queramos publicar un conjunto de cambios o cuando queramos tomar la última versión publicada en concordancia con nuestra copia local.	El sistema de control de versiones distribuido ha sido pensado con la forma de trabajo basada en ramas, unión central en una sola versión (trunk) y liberaciones (o tags). Con lo que cada rama puede identificarse como cada copia distribuida que se use.

- **Explique en qué consisten los estados de Git, y de un ejemplo que ilustre los mismos.** (1 punto)

  R:

  Git tiene tres estados principales en los que se pueden encontrar los archivos:
  1.- **Confirmado** (committed): significa que los datos están almacenados de manera segura en tu base de datos local.
  2.- **Modificado** (modified): significa que has modificado el archivo pero todavía no lo has confirmado a tu base de datos.
  3.- **Preparado** (stagedArchivos modificados que están listos para formar parte del repositorio a través de la confirmación.

  Ejemplo: para el estado de modificación, es simplemente modificar un archivo dentro del directorio de trabajo por ejemplo cualquier modificación a hola.txt el cual se encuentra dentro del directorio de trabajo. Para el estado preparado es preparar el archivo hola.txt para que sea confirmado. En el estado de confirmación se almacena el archivo hola.txt en el repositorio git.



- **Explique el término *debugging* y su importancia.** (1 punto)

  R:

  El debuggin es un análisis paso a paso sobre el código de un programa el cual no está arrojando un resultado esperado, esto con el fin de encontrar el error en tiempo de ejecución. Su importancia es que facilita en gran medida la búsqueda y

solución de errores en el código, además de que permite monitorear e identificar fácilmente la perdida de los datos durante la ejecución.

- **Defina el término *plug-in*.** (1 punto)

  R:

  Es una aplicación que añade nuevas funcionalidades o características específicas a un software.

- **Defina *JUnit 5* y su composición. Explique brevemente cada uno de sus componentes.** (1 punto)

  R:

  JUnit es un conjunto de clases (framework) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

  JUnit es también un medio de controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación.

  JUnit 5 está compuesto por:

  1.- **JUnit Platform** es la base que nos permite el lanzamiento de los frameworks de prueba en la JVM y, entre otras cosas, también es el encargado de proporcionarnos la posibilidad de lanzar la plataforma desde línea de comandos y de los plugins para Gradle y Maven.

  2.- **JUnit Jupiter** es el que más utilizaremos a la hora de programar. Nos permite utilizar el nuevo modelo de programación para la escritura de los nuevos tests de JUnit 5.

  3.- **Junit Vintage** es el encargado de los tests de Junit 3 y 4.