Preguntas Examen Semana 03:

1.-Git plantear escenario, hacer ramas, sobre el mismo archivo hay conflicto y como se resuelve.

R: Se tiene un proyecto donde se planea hacer un videojuego sobre coches, para su realización usaremos un controlador de versiones el cual será git, para esto crearemos un repositorio llamado videojuego, para hacer esto usaremos el comando **git init** para crear la rama master, en esta se hara un archivo txt con todas las especificaciones que se requiere.

Después añadiremos este archivo al index de git con el comando **git add #nombre_del_archivo**, este comando lo que hace hacer una copia del archivo que tenemos de forma local y añadirla al index de git, después haremos un commit con el comando **git commit -m "mensaje del commit",** para asi que git lo deje registrado en su base.

Ahora cada integrante estará a cargo de una especificación del videojuego a lo cual a partir de la rama master que es la principal crearan ramas diversas las cuales llevaran por nombre su especificación. Para crear estas ramas ocuparemos el comando **git Branch #nombre_de_la_rama,** usaremos este comando para crea las n ramas de las especificaciones del videojuego. Una vez creadas listaremos todas las ramas existentes para verificar que no falte o hayamos omitido alguna con el comando **git Branch,** nos listará todas las ramas y marcara con un (*) asterisco la rama donde estamos posicionados.

```
lucho@DESKTOP-8LMTQUM MINGW64 ~/Mis documentos/Tutoriales/testgit (master)
$ git branch
  development
  development2
* master
  nueva-rama
```

Para cambiarnos entre ramas ocuparemos el comando **git switch #nombre_de_la_rama** y volveremos a listar las ramas para verificar que se haya echo el cambio.

En cada rama cada integrante creará un txt con el nombre y descripción de su funcionalidad asignada, lo añadirán y harán commit de estos mismos para persistir los cambios.
A continuación al hacer la rama de cada especificación como se hizo de la rama master tendrán en sus ramas el archivo txt principal en el cual en cada rama se modificará para cambiar el estatus de cada especificación una vez terminada.
Terminando esto harán un merge para juntar todos los archivos en la rama master con el comando **git merge #nombre_rama_a_juntar** pero primero deberemos de estar posicionados en la rama master.

Pero todos al haber modificado el archivo txt habrá un error al hacer el merge en este archivo dados que ha sido modificado por varias personas con lo cual nos saldrá el error:

error: Entry '<fileName>' would be overwritten by merge. Cannot merge. (Changes in staging are a)

Para solucionar este conflicto lo que haremos será abrir el archivo en cuestión y nos mostrará el siguiente contenido:

Donde en el head es donde empieza el conflicto y es separado mediante **=======** los cambios hechos por ti y los cambios hechos por otra persona, para resolver esto se debe de eliminar los comentarios y decidir el como va a quedar el archivo, una vez que se haya hecho esto podemos guardar el archivo y realizar de nuevo el **git add, git commit y el merge.**

Así es como se solucionaría el conflicto.

2.-Hacer programa del polimorfismo con lambdas con perímetro y áreas de las figuras Rectángulo y Triángulo:

R: Se anexa el programa correspondiente.

3.- Explica en que consiste la inyección de dependencias:

R: La inyección de dependencias nos permite módulos no acoplados o dependientes de otros, esto hace que los componentes usan las funcionalidades definidas en las interfaces.

Es útil debido a que nos permite cambiar fácilmente el comportamiento de una aplicación al cambiar los componentes que se implementó en las interfaces, así como son más fáciles de aislar para pruebas unitarias.

Existen dos tipos: por constructor o por uso de propiedades.

La Inyección de Dependencia nos permite configurar cómo se crean los objetos.

-Ejemplo:

Se tiene a un piloto de avión comercial, imagine que cada vez que tiene que pilotear tuviera que llenar el tanque de combustible, preparar la pista, verificar que todo está listo para el despegue, sabemos que no es humanamente posible, por eso existen distintas personas trabajan en conjunto para que sea posible el despegue. Entonces si adaptamos esta situación para el aterrizaje:

-El aeropuerto hace la función de Contenedor de dependencia, pues se encarga de gestionar todo el proceso para que los vuelos puedan tener lugar. Los controladores aéreos, hacen la función de Framework, pues se encargan de inyectar las dependencias a los módulos dependientes y gestionar los recursos para que los módulos puedan funcionar correctamente, las interfaces son los lineamientos que se necesitan para que el avión pueda aterrizar. Todos elementos en conjunto conforman la inyección de dependencia y la inversión de control.

-Cuando un piloto va aterrizar en una pista, necesita tener el espacio para poder descender y que el transito actual del aeropuerto no interfiera con su aterrizaje. Pero él no va a comunicarse directamente con el controlador de Tierra o el controlador de Ruta o Área. En lugar de ello, se comunicará con la torre de control quien se encarga de gestionar todo el proceso y darle toda la información que necesita para el aterrizaje.

4.-Dibujar el diagrama se servicios rest:

R: Se anexa diagrama.

5.- Hacer un servicio Rest:

R: Se anexa el programa en cuestión.