Taller Git - MVC

Integrantes:

- Doménica Barreiro
- Karla Castro
- Robert Moreno
- Paulette Vázquez

Objetivos:

- Analizar la estructura de un proyecto implementado a partir del patrón arquitectónico MVC.
- Realizar modificaciones al proyecto de forma distribuida utilizando GitHub.
- Resolver conflictos de integración de código utilizando diferentes estrategias.

Antecedentes

Cada grupo debe crear una organización dentro de Github y crear un nuevo repositorio en donde todos los integrantes puedan tener permisos para realizar cambios (rw). La primera parte de este taller la deben realizar en conjunto, mientras que la segunda parte debe ser dividida entre los integrantes y deben subir sus avances al repositorio remoto.

Pasos previos

1. Un integrante debe crear una organización en GitHub con un nombre cualquiera para este taller.
2. Agregar a los integrantes del grupo a la Organización enviándoles invitación. (Cada integrante debe abrir su correo y aceptar la invitación)
3. Dentro de la página de la Organización debe crear un nuevo repositorio.
4. Dentro de la configuración de la organización ir a "**Member privileges**" y en "**Organization members**" darles permiso "**Write**" a los miembros.
5. Después, debe clonar el repositorio en un directorio de la computadora local.
6. Descomprimir y copiar el código del proyecto Snake dentro del repositorio local.
7. Finalmente, debe agregar los nuevos archivos al repositorio remoto.

Comandos importantes

- Conocer el estado de su repositorio: **git status**
- Clonar localmente un repositorio remoto: **git clone <repositorio_remoto>**
- Agregar todos los cambios realizados al stage: **git add .**
- Guardar todos los cambios agregados al stage: **git commit -m "Comentarios de los cambios realizados"**
- Enviar al repositorio remoto todos los commit sin enviar: **git push origin master**
- Descargar los nuevos cambios desde el repositorio remoto: **git pull**
- Si hay un usuario grabado en su computadora y desea utilizar otro: **git config --local credential.helper ""**

Parte 1

1. Abrir su correo asociado a Github y aceptar la invitación para colaborar en el proyecto.
2. Clone en un directorio de su computadora el repositorio remoto. (Esto deben realizarlo todos los integrantes en cada computadora)
3. Importe el proyecto al IDE de su preferencia (Se recomienda usar Eclipse, pero se puede importar desde Netbeans).
4. Observe y analice la estructura del proyecto, complete lo siguiente.
    - **Según su criterio, ¿cómo clasificaría a las clases/archivos que conforman el proyecto?:**

Modelos	Vistas	Controladores	Auxiliares
GameModel GoldModel SnakeModel GameTile RectangularTile RoundTile Position	GUIView GameView	GameController GameFactory	Constants GameOverException Main IGameFactory

    - **¿Considera usted que el proyecto fue implementado siguiendo los principios del patrón MVC?**

      Sí, la estructura que posee el proyecto divide el código fuente en clases cuyas funciones son específicas ya sea en la visualización, la lógica y la respuesta a las acciones del usuario.

Parte 2

Cada integrante debe clonar en su computadora el repositorio remoto y luego todos deben trabajar paralelamente con esta versión. Luego, el orden para enviar los cambios realizados debe ser el siguiente:

1. **Integrante1**(creó el repositorio):
    a. Ventana principal:
        i. Cambiar el texto del botón de “Start Game” a “Let’s Go!!!”.
    b. Juego Gold:
        i. Por cada ficha recolectada, asigne 2 puntos en lugar de 1.
2. **Integrante2**:
    a. Ventana principal:
        i. Cambiar las dimensiones de la ventana a 15x15.
        ii. Cambiar el texto del botón “Start Game” a “Let’s Play”
        iii. Cambiar el color de fondo de la pantalla principal de blanco a gris (lightGray)
    b. Juego Snake:
        i. Para el juego ‘Snake’ cambie el color de la serpiente a verde (GREEN).

3. **Integrante3**:
    a. Ventana principal:
        i. Cambiar el texto del botón “Start Game” a “Empezar”

ii. Cambiar el color de fondo de la pantalla principal de blanco a celeste.

b. Juego Gold:
   i. Cambie el color de relleno de las fichas a verde (green) y el color del borde a azul (blue).
   ii. Por cada ficha recolectada, asigne 3 puntos en lugar de 1.

4. **Integrante4**:
   a. Juego Gold:
      i. Cambie el color de relleno de las fichas a verde (CYAN).
      ii. Por cada ficha recolectada, asigne 4 puntos en lugar de 2.
   b. Juego Snake:
      i. Aumente el número de frutas de 1 a 3.

Nota: Tengan en cuenta que cada integrante debe realizar los cambios que le corresponde, pero al finalizar deben subir en el orden (integrante 1, integrante 2, integrante 3, integrante 1). Esto es para tratar de generar un conflicto de integración de código con cada integrante.

Parte 3

Responda a las siguientes interrogantes (**Solo uno del grupo debe subir este archivo al GitHub con las respuestas grupales**):

1. **¿Le resultó complicado realizar los cambios solicitados?**

   No, la mayor complicación dentro de esta actividad fue entender el código y encontrar las clases correspondientes ya que todas se encontraban dentro de un mismo paquete; sin embargo, lo descriptivo que fueron los nombres de las clases, y el modelo que posee permitieron agilizar este proceso de búsqueda y actualización.

2. **¿Cuáles considera usted que son los archivos/clases 'más importantes'?**
   Los modelos del juego, estos son la base principal para el funcionamiento del programa. A pesar de contar con la estructura visual e incluso las herramientas que controlan las interacciones, si no existe un esquema mediante el cual la información se pueda representar o mostrar no funciona.

3. **¿Qué aspectos ayudaron a realizar los cambios?**

   Al no saber de antemano de qué trataba el proyecto o su estructura, facilitó mucho que el nombre de cada archivo fuera bastante descriptivo.

4. **Luego de haber explorado el código, ¿considera usted que se respetan los principios de MVC?**

   Sí, el código estaba estructurado de tal forma en que ciertas clases cumplían con roles específicos, ya sea generando gráficamente lo que se debía mostrar al jugador, manejando la lógica del juego y manejando la interacción entre el programa y el jugador con los debidos controladores.

5. **¿Qué cambios haría para mejorar la arquitectura de la aplicación?**
   Empaquetar las clases clasificándolas según la función que cumplen, es decir siguiendo los principios del patrón MVC, ayudaría en mantener más organizado las

componentes del proyecto y facilitaría el acceso y entendimiento del código; lo que implícitamente mejora el proceso de escalabilidad y mantenibilidad del programa.

Entregables

1. En el repositorio de GitHub debe estar el código con los cambios requeridos.
2. En el repositorio de GitHub debe estar subido este archivo con las respuestas de todo el grupo.
3. En Sidweb debe subir el enlace del último commit del repositorio remoto.