Taller Git - MVC

# 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. Dentro de la página de la Organización debe crear un nuevo repositorio.
3. Dentro de este repositorio debe agregar a los demás integrantes del grupo (**Collaborators**).
4. A cada integrante debe darle permiso para poder modificar(**write**) el código del repositorio.
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.

# 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).
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 |
| GoldModel | GUIView | GameFactory | Constants |
| SnakeModel | GameView | GameController | GameTile |
| Position |  |  | RectangularTile |
|  |  |  | RoundTile |
|  |  |  | IGameFactory |
|  |  |  | GameOverException |
|  |  |  | GameModel |

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

Si, debido a que se pueden clasificar las distintas clases en Modelos, Vistas y Controladores. Las clases Modelos almacenan los datos de los objetos a utilizar en el juego e información esencial de cada uno de ellos. Las clases Vistas se encargan de la correcta presentación de la interfaz grafica al usuario. Finalmente, las clases Controladoras se encarga de que existe una comunicación efectiva entre las clases Modelos y las Vistas.

# Parte 2

Realicen las siguientes modificaciones al código de acuerdo, tomando uno solo rol:

1. **Integrante2**:
   1. Ventana principal:
      1. Cambiar las dimensiones de la ventana a 15x15.
      2. Cambiar el texto del botón “Start Game” a “Let’s Play”
      3. Cambiar el color de fondo de la pantalla principal de blanco a gris (lightGray)
   2. Juego Snake:
      1. Para el juego ‘Snake’ cambie el color de la serpiente a verde (GREEN).
2. **Integrante3**:
   1. Ventana principal:
      1. Cambiar el texto del botón “Start Game” a “Empezar”
      2. Cambiar el color de fondo de la pantalla principal de blanco a celeste.
   2. Juego Gold:
      1. Cambie el color de relleno de las fichas a verde (green) y el color del borde a azul (blue).
      2. Por cada ficha recolectada, asigne 3 puntos en lugar de 1.
3. **Integrante1**(creó el repositorio):
   1. Juego Gold:
      1. Cambie el color de relleno de las fichas a verde (green).
      2. Por cada ficha recolectada, asigne 2 puntos en lugar de 1.
   2. Juego Snake:
      1. 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 2, integrante 3, integrante 1). Esto es para tratar de generar un conflicto de integración de código.

# 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, los cambios fueron sencillos de identificar debido a que las clases estaban bien comentadas.

1. ¿Cuáles considera usted que son los archivos/clases ‘más importantes?

Los archivos “Modelo” son los mas importantes porque contienen la información general de como van a ser instanciados estos objetos. Por ejemplo, GameModel,GoldModel,SnakeModel,Main,etc…

1. ¿Qué aspectos ayudaron a realizar los cambios?

Las funciones y atributos de las clases estaban bien comentados y se podía leer fácilmente el código. Asi mismo, había un correcto flujo entre una clase y otra, evitando ambigüedades.

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

Si, se presentan claramente los principios del Model View Controller Patterns en el programa, haciendo mas fácil la lectura y manipulación de este.

1. ¿Qué cambios haría para mejorar la arquitectura de la aplicación?

Trataria de reducir la cantidad de clases que existen.

# 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 ultimo commit del repositorio remoto.