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. 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](https://github.com/organizations/TallerGIT/settings/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  GameOverException  GoldModel  Position  SnakeModel | GUIView  GameTile  GameView  RectangularTile  RoundTile | GameController | Constants  GameFactory  IGameFactory  Main |

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

El proyecto sí parece haber sido desarrollado según el patrón MVC, ya que sí hay una distinción clara entre las clases que controlan la interfaz gráfica (vistas), las clases que sólo son contenedores de datos (modelos) y las clases que implementan la lógica del juego y que actúan de vínculo entre los modelos y vistas (controladores).

# 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):
   1. Ventana principal:
      1. Cambiar el texto del botón de “Start Game” a “Let’s Go!!!”.
   2. Juego Gold:
      1. Por cada ficha recolectada, asigne 2 puntos en lugar de 1.
2. **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).
3. **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.
4. **Integrante1( otra vez )**:
   1. Juego Gold:
      1. Cambie el color de relleno de las fichas a verde (green).
      2. Por cada ficha recolectada, asigne 4 puntos en lugar de 2.
   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 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?

Al inicio se tuvo algunos inconvenientes de sincronización con las cuentas de GitHub, no obstante, fue sencillo encontrar las clases y los atributos que poseían la información a modificar puesto que la distribución de las mismas y su identificación (el nombre de la clase), proporcionaban una idea de la información que contenía por lo que navegar en el código fue sencillo.

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

* GameController
* GameView
* GameModel

Si bien es cierto cada clase es importante, puesto que poseen la estructura de cada objeto que interviene en el programa. Sin embargo, considerando que el programa aplica el modelo MVC, es de entender que aquellas clases que controlan la comunicación entre los diferentes objetos del programa y entre cada componente son las descritas al inicio de esta respuesta.

1. ¿Qué aspectos ayudaron a realizar los cambios?

Principalmente fue posible reconocer los atributos gracias a que éstos estaban debidamente comentados.

Por otra parte, el nombre de las clases proporciona una idea la información que puede contener.

También resultó de gran ayuda el hecho de que las variables están definidas de manera intuitiva, es decir que proyectan una idea del valor que posee el atributo.

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

Considerando que previamente establecimos que el proyecto parece haber sido desarrollado en base al modelo MCV, se opina que:

Cumple con el principio de modularización ya que las clases manejan de manera independiente la información sin afectar a otras. También cumple con el concepto de abstracción ya que proporciona fácilmente una idea de los elementos principales del proyecto. Por otra parte cumple también el principio de separación de la interfaz y la implementación. No obstante para medir la suficiencia y completitud del proyecto creemos que es necesario analizar los requerimientos iniciales de la propuesta, es decir los alcances del mismo, ya que por el momento creemos que el proyecto es completamente funcional. También cumple el principio de cohesión ya que cada módulo tiene una función particular dentro del proyecto, además cumple el principio de acoplamiento porque existen clases como “Constants” que proporcionan información sobre las dimensiones de la ventana, alterando el comportamiento de una clase diferente.

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

Algo que podría mejorar de manera significativa es clasificar las clases en packages, de manera que se puede representar el modelo MVC y que el acceso a cada componente tenga un comportamiento más intuitivo en la lectura del código.

# 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.