Universidad Nacional de La Matanza

DEPARTAMENTO DE INGENIERÍA

PARADIGMAS DE PROGRAMACIÓN

TP – HÉROES Y VILLANOS

Integrantes:

- *Di Rocco, Sebastián.*
- *Manghi Sheck, Santiago.*
- *Rodriguez Fenske, Axel.*
- *Toloza, Alan.*
- *Zamudio, Verónica.*
- *Zitzman, Yair.*

Introducción

En este documento se encuentra el informe sobre el segundo trabajo práctico de Paradigmas de Programación “Héroes y Villanos”. Se detallará las estrategias utilizadas para el proyecto, los archivos, distribución de tareas y conclusión del trabajo.

Estrategias utilizadas:

A continuación, se detallará las estrategias y documentación de las diferentes secciones relevantes para la creación del código.

LECTURA Y ESCRITURA DE ARCHIVOS:

Creación de clases para la lectura y escritura de los archivos personajes y ligas, con la información almacenada en listas y mapas.

En un principio se consideró utilizar el patrón de diseño Template Method, con el objetivo de no repetir líneas de código, ya que son similares en ambas clases. Se desestimó esta opción debido a la complejidad de la misma. Se optó por crear las clases teniendo en cuenta la teoría de entrada/salida propuesta en la materia. Sencillez y claridad de código, a cambio de algunos bloques de código similares.

COMPOSITE:

Decidimos utilizar el patrón de diseño Composite, ya que este patrón nos permitió trabajar con los personajes y las ligas (composición de personajes) de una manera uniforme, sin la necesidad de distinguir los objetos individuales de las composiciones. Todo esto nos permitió simplificar la complejidad y tener un código más claro y flexible.

REALIZACIÓN DE COMBATES:

El usuario podrá optar entre Liga vs Liga, Jugador vs Liga o Jugador vs Jugador.

Para el diseño del combate, se tuvo en cuenta el tipo de personaje, es decir, sí es Héroe o Villano, ya que no podrán luchar jugadores del mismo tipo. Se validó la existencia del personaje/liga y a cuál bando pertenece. En caso de no existir o su competencia ser del mismo bando, no se realizará el combate. Caso contrario, se solicita la característica para la cual se desea enfrentar, informando quien será el ganador de la misma.

REPORTES:

En una primera instancia no contemplamos el uso de archivos a la hora de informar los resultados de ambos puntos, ya que desconocíamos a que se refería en sí mismo un reporte.

Con el propósito de poder documentar un evento especifico, ya sea para listar personajes que cumplan cierta característica o personajes o ligas que venzan a un personaje, se decidió crear la Clase reportes, de modo que se pueda generar un historial de archivos de texto para las diferentes consultas que se realicen a lo largo del juego.

Descripción de archivos:

A continuación, se detallan todos los archivos utilizados para el proyecto.

Archivo	**Descripción**
ArchivoLigas.java	Dedicado a la gestión de entrada y salida de ligas de héroes y villanos
ArchivoPersonajes.java	Dedicado a la gestión de entrada y salida de personajes individualmente
Característica.java	Enum que posee las características que contiene un Competidor
Competidor.java	Clase abstracta encargada de la realización de combates de los personajes
CompetidorComparador.java	Dedicado a la comparación de competidores
Liga.java	Clase abstracta derivada de Competidor encargada del manejo de ligas
Main.java	Menú de consola inicial para manejo de necesidad del usuario y funciones genéricas que son utilizadas por los submenús.
AdministracionDePersonajes.java	Menú de consola dedicado a la ceración, gestión y guardado de unidades simples
AdministracionDeLigas.java	Menú de consola dedicado a la ceración, gestión y guardado de las Ligas de personajes.
RealizacionDeCombates.java	Menú de consola dedicado a gestión de los combates.
Reportes.java	Dedicado a manejar (mediante consola) y guardar los reportes
Unidad.java	Clase derivada de Competidor encargada del manejo de personajes individuales

CompetidorTest.java	Prueba las características de la clase Competidor
Enfretamientos.java	Prueba los comportamientos de cada tipo de combate
LigaTest.java	Prueba las características de la clase Liga
UnidadTest.java	Prueba las características de la clase Unidad

Distribución del trabajo:

En un principio se realizó una reunión grupal, donde se expuso las ideas principales para encarar el proyecto. A partir de la creación de la clase Main (menú de juego), se decidió dividir las tareas a los distintos integrantes del grupo.

A medida que se fue completando las tareas, cada integrante subía las partes del código a la plataforma GitHub con el propósito de poder modificar y actualizar el proyecto. Al mismo tiempo se han realizado pruebas de caja blanca y al completarse todo el trabajo se realizó una validación general de todo el proyecto.

Se han llevado a cabo reuniones virtuales a lo largo del trabajo con el propósito de observar el progreso y, en caso de ser necesario, ayudar al compañero con la tarea que se le había propuesto.

Tareas:

Composite: Axel / Sebastián.

Reportes: Verónica.

UML: Verónica.

Archivos Entrada/Salida: Yair.

Tests y casos de pruebas: Alan, Santiago.

Validaciones: Yair, Alan, Axel.

Ligas: Axel.

Main: Sebastián.

Combates: Santiago.

Informe: Alan, Verónica, Santiago.

Conclusión

Durante el desarrollo del proyecto, hemos notado que la planificación inicial para el entendimiento del problema nos permitió avanzar sin dificultades a lo largo del trabajo. La división de tareas nos permitió cumplir un rol especifico. Incluso sí algún integrante enfrentaba alguna dificultad para la resolución de la tarea que se le asigno, en las siguientes reuniones grupales se resolvía en forma conjunta.

La investigación que se llevó a cabo nos permitió profundizar algunos temas del lenguaje de programación Java, así como también aprender de algunos temas que no contempla la materia, ya que excede a la misma.

Entre las dificultades que hemos encontrado, podríamos destacar la dificultad de entender código escrito por otro compañero y la administración del tiempo. Sin embargo, creemos que el trabajo fue una buena experiencia para aplicar los conocimientos aprendidos en la materia sobre Java y poder familiarizarse aún más con el Paradigma Orientado a Objetos.