Proyecto Final

Modelado y Programación

Integrantes.

- Ocampo Villegas Roberto, 316293336.
- Liprandi Cortes Rodrigo, 317275605.

Descripción del proyecto.

Para este proyecto creamos una parte de un juego de estrategia militar, donde nosotros controlamos a un ejército de soldados y tenemos como objetivo acabar con un enemigo.

Justificación de los patrones utilizados.

- Strategy: El patrón de diseño Strategy se encuentra implementado en las interfaces AttackBehaviour.java, MovementBehaviour.java, ReportBehaviour.java y las clases que las implementan, así como en Soldado.java, Artilleria.java, Caballeria.java e Infanteria.java. El patrón fue utilizado debido a que contábamos con clases similares con los mismos métodos pero que hacían diferentes cosas, es decir, tenían las mismas acciones pero con diferentes comportamientos.

- Observer: El patrón de diseño Observer se encuentra implementado en las clases de Ejercito.java y Comandante.java, mas especificamente, en Ejercito.java implementamos la Interfaz Sujeto, y en Comandante.java implementamos Observer.
  En un principio se iba a implementar Composite para Ejercito.java también pero esto hubiera hecho que Ejército tuviera que heredar de la clase Soldado, lo cual está muy mal debido a que no necesitamos que Ejército tenga comportamientos de un Soldado, además de que el código quedaba confuso.
  Observer nos permite de forma sencilla notificar a los Comandantes sobre una nueva orden desde un objeto Ejército sin tener ningún problema de los ya mencionados.

- Composite: El patrón de diseño Composite se encuentra implementado en la clase abstracta Soldado.java, y en las clases hijas de este, que son Comandante.java, Artillero.java, Infanteria.java y Caballeria.java.

Se utilizó este patrón debido a que queríamos plantear una jerarquía entre los diferentes tipos de soldado, siendo los comandantes lo de mayor jerarquía y los demás soldado de menor.
Los objetos compuestos serían los comandantes, los cuales tienen como atributo una lista de Soldados, y los objetos individuales sería cualquiera de los tres tipos de soldados.
Con este patrón se vuelve fácil implementar el método ordenar(), ya que podemos implementarlo tanto el los comandantes como en los soldados.
El método en un comandante va a hacer que este realice una acción y haga que los soldados que se encuentran en su lista también realicen la operación ordenar().
En los soldados el método ordenar() solo va a hacer que realicen una acción.

- Factory: El patrón de diseño Factory se encuentra implementado en las clases FabricaEjercitos.java, Ejercito.java, EjercitoExplorador.java, EjercitoDefault.java y EjercitoKamikaze.java. La razón para usar este patrón es para hacer más flexible y unificar la creación de los ejércitos y así no depender completamente de las clases de los productos concretos (los diferentes tipos de ejércitos). El uso del patrón también nos salva de caer en problemas de diseño como inmovilidad y rigidez ya que nos da la oportunidad de poder hacer ampliaciones, en este caso agregar nuevos tipos de ejércitos, sin tener que modificar en un mayor grado nuestro código.

- Facade: El patrón de diseño Facade se encuentra implementado en la clase Menu.java. La razón de usar Facade en el proyecto es que con este patrón podemos “enmascarar” las clases usadas en el proyecto en una clase (Menu.java) para así poder unificar las distintas tareas que estas clases hacen y así contar con una consola unificada para el usuario. También impide que la clase Main.java sea una “clase gorda” y de esta manera hizo el mantenimiento de la simulación más sencilla.

Consideraciones.

- El proyecto se ejecuta ingresando a la carpeta src desde la terminal y utilizando el comando javac *.java y luego java Main.
- Para seleccionar un ejército presionamos un número del 1 al 3, dependiendo del ejército que queramos.

- Después de seleccionar al ejército habrá una pequeña interfaz donde se nos da la vida del enemigo y las órdenes que les podemos dar al ejército, para seleccionar una orden presionamos un número del 1 al 3.
- El programa solo acabará cuando la vida del monstruo llegue a 0.