Proyecto final

Modelado y programación

Objetivo: *El objetivo de este proyecto es resolver un problema propuesto considerando y valorando el uso de patrones de diseño de software.*

Problema:
Diseña los diagramas UML* e implementa la solución para el siguiente problema usando los patrones de diseño vistos en clase que creas necesarios sin que tenga problemas de diseño como son Rigidez, Viscosidad, Fragilidad, Inmovilidad:

Se está desarrollando un videojuego de estrategia militar y se te ha encargado implementar algunos módulos del mismo.

El juego consiste en un mundo donde el jugador controla varios pelotones a la vez. El objetivo de este juego será eliminar a un enemigo con puntos de vida únicos y a cierta distancia inicial de los pelotones. Los pelotones están conformados por soldados, y cada soldado ejerce una única especialidad.Todos los soldados pueden moverse, atacar y reportarse con su comandante, pero cada uno lo hace de forma diferente dependiendo de su especialidad:

- Infantería:
  Se mueve a pie. Movimiento normal.
  Ataca con una pistola. Hace un daño normal.
  Se reporta como infante.
- Caballería:
  Se mueve en su caballo. Movimiento rápido.
  Ataca con un mosquete. Hace un daño alto.
  Se reporta como caballero.
- Artillería:
  Se mueve junto con un cañón. Movimiento lento.
  Ataca con un cañón. Hace un daño muy alto. Se daña a sí mismo ligeramente con cada ataque.
  Se reporta como artillero.

Además de la especialidad, todos los soldados tienen puntos de vida, un id y un nombre.

(HINT: Piensen cómo es que cada soldado individual debe hacer cada acción respetando esas reglas de un soldado en general.)

Todos los soldados deben obedecer las órdenes de un comandante de pelotón. El comandante puede ejercer cualquiera de las 3 especialidades, por lo que no hay restricción para el líder de un pelotón.

Los comandantes obedecen las instrucciones del jugador y la comunican a sus subordinados.
(HINT: Piensen cómo deben organizarse los soldados en el sistema.
HINT 2: Piensen cómo se pueden comunicar los soldados y hasta qué nivel deben comunicarse dado el hint anterior.)

Al inicio del juego, se le ofrece un menú al usuario para crear un ejército. Existen 3 ejércitos disponibles compuestos de la siguiente manera:

- Explorador: Se conforma con 3 pelotones.
  Pelotón 1 tienen 6 soldados de infantería.
  Pelotón 2 se compone con 1 artillero y 2 miembros de la caballería.
  Pelotón 3 se compone con 6 soldados, todos miembros de la caballería.
- Default: Se conforma con 3 pelotones.
  Pelotón 1 tienen 6 soldados, todos miembros de la infantería.
  Pelotón 2 tiene 3 artilleros.
  Pelotón 3 tiene 6 miembros de la caballería.
- Kamikaze: Se conforma de 2 pelotones.
  Pelotón 1 y 2 tienen 5 soldados cada uno, todos miembros de la infantería.
  Pelotón 3 tiene 5 soldados, todos miembros de la caballería.

(HINT: Piensen cómo se puede organizar la creación de estos ejércitos de forma unificada.)

El jugador debe poder controlar de forma centralizada a todos los pelotones. Es decir que el jugador cuenta con una consola unificada para manipular a su ejército.
(HINT: Piensen cómo se puede unificar las distintas tareas.)

La simulación debe permitir al jugador interactuar con los pelotones. Debe incluirse un elemento enemigo con el que los soldados deben "pelear". No es necesario que la simulación sea detallada ni refinada, pero debe mostrarse en pantalla que un soldado X se está moviendo/atacando/reportándose. El enemigo contará con un atributo que represente sus puntos de vida. La simulación terminará cuando el ejército reduzca a 0 los puntos de vida del enemigo.

Consideraciones:
-No es requisito que el enemigo pueda atacar.
-No es requerida una representación gráfica del juego. Es más que suficiente una representación en texto en terminal.

Requerimientos:

- El programa debe iniciar con una pantalla que te permite elegir entre los ejércitos disponibles, con las características de cada soldado. Cuando se elige el ejército, se debe imprimir la información de cada soldado.

- Una vez personalizado el ejército y revisado la lista de soldados, se presenta una pantalla que indica los puntos de vida del enemigo, la distancia inicial y te presente acciones a ordenar para todos los pelotones. Estas acciones son atacar, moverse hacia el enemigo o reportarse.
- La acción que se elija debe ser vista por los comandantes y enviada a todos sus soldados. Una vez seleccionada la acción debe mostrar en pantalla que cada soldado realizó la acción que le corresponde.
- La acción atacar, hará que todos los soldados que se encuentren a distancia 0 del enemigo, hagan daño.
    - Los soldados cuyo HP baje a 0, no podrán atacar.
- La acción moverse hará que los soldados que no estén a distancia 0 del enemigo, se acerquen a él, e indicarán a la distancia que quedan del enemigo al hacerlo. Los soldados que ya estén a distancia 0 del enemigo solo dirán “ya estoy al lado del enemigo”.
- La acción “reportarse”, indicará el cargo de cada soldado, su id, la distancia a la que se encuentran del enemigo y el HP de cada uno.
- Cada que el enemigo recibe daño, se deben mostrar sus puntos de vida restantes, que no deben bajar más allá de 0.
- Al terminar de mostrar los mensajes correspondientes de cada acción, se debe mostrar de nuevo el menú para realizar otra acción, hasta que la simulación termine.
- Cuando los puntos de vida del enemigo lleguen a 0, la simulación termina, o se reinicia, luego de mostrar un mensaje de felicitación.

UML:

- Diagrama de clases: Para esta entrega, deberán enviar el diagrama de clases de forma separada. Dependiendo de sus diseños, hagan un diagrama de clases por módulo aunque se repitan clases en varias imágenes.
- Diagrama de secuencias. Debe representar la propagación de órdenes entre el usuario, la interfaz, el comandante y un soldado.

Evaluación:

- ❖ Diagramas UML 2.5
- ❖ Justificación 2.5
- ❖ Implementación 2.5
- ❖ Requerimientos 2.5

Lineamientos

- La entrega se realizará en equipos de a lo más **3** personas.
- Se deberá entregar un archivo pdf llamado README el cual debe incluir:
    - Nombres de los integrantes
    - Números de cuenta
    - Una breve descripción del proyecto y cómo ejecutarlo. **Una justificación** de cuáles patrones decidieron usar y la razón de su uso (mínimo una cuartilla).

En caso de haber resuelto alguna parte del proyecto sin usar un patrón también debe justificarse.

- En caso de que existan anotaciones o consideraciones sobre su implementación deben incluirlas. Si entregan el proyecto incompleto, deben anotar en el readme las partes que faltan. Si estas notas no se incluyen, no habrá consideraciones al evaluar sus entregas.
- Se deben entregar en imágenes los diagramas que se les pidan. No se revisarán archivos dia, xml u otro formato que no sea de imágen (jpg, jpeg, png)
- **Su implementación estará hecha en java y <u>TODO</u> el código debe estar adecuadamente documentado**.
- El formato de entrega es el siguiente:
    - ❖ Un archivo zip que incluirá lo siguiente:
        - ➢ Dentro del archivo se incluirá el documento pdf titulado README
        - ➢ A la misma altura del pdf, se guardarán los diagramas solicitados.
        - ➢ Una carpeta titulada src que incluirá solamente los archivos .java de su implementación

  Si no se respeta este formato de entrega y el readme no justifica este cambio, se descontarán puntos de su calificación.
  El nombre del zip tendrá el siguiente formato:
  Proyecto01_ApellidoPaternoIntegrante1_ApellidoPaternoIntegrante2_ApellidoPaternoIntegrante3.zip
- Si se encuentran implementaciones copiadas, se repartirá la calificación entre los equipos.
- **Si el código no tiene comentarios de javadoc, se restará un punto.**