INF-253 Lenguajes de Programación
Tarea 3: Java

5 de octubre de 2022

1. Certamen

Tras exitosamente superar tus certámenes con la ayuda de la herramienta que programaste en la tarea anterior, te encuentras finalmente con algo de tiempo libre en tus manos, y decides utilizarlo para continuar con uno de tus hobbies, hacer videojuegos!

2. Java Quest

Para esta tarea deberán crear el juego Java Quest utilizando programación orientada a objetos en Java. En este juego los jugadores deben seleccionar un camino para avanzar por una serie de encuentros de distintos tipos con el objetivo de derrotar al jefe final.

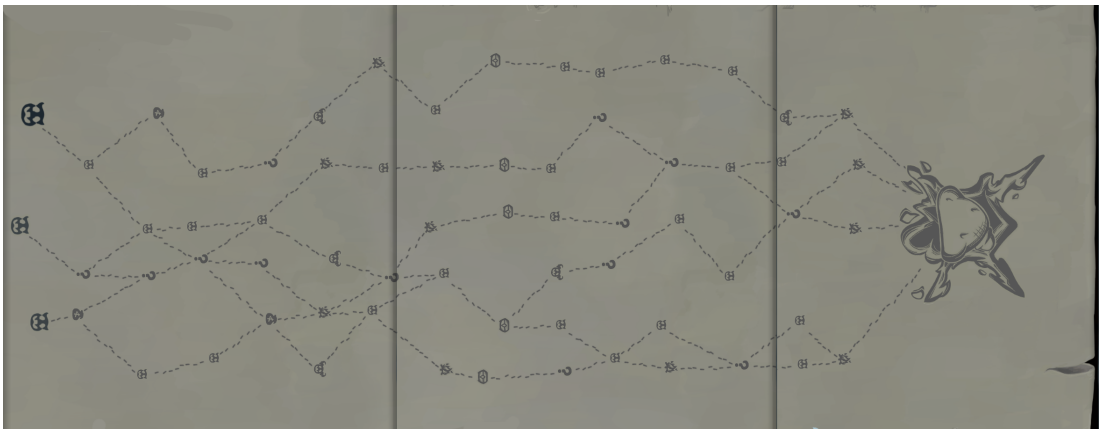

Figura 1: Ejemplo de un mapa de encuentros en un juego similar (Slay the Spire).

3. Definición

Para crear Java Quest deberán implementar (y utilizar) las siguientes clases: Personajes, Jugador, Items, Mapa, y los distintos tipos de nodos que pueden haber en un mapa. A continuación se presenta un diagrama de clases que describe que debe contener cada una de estas.

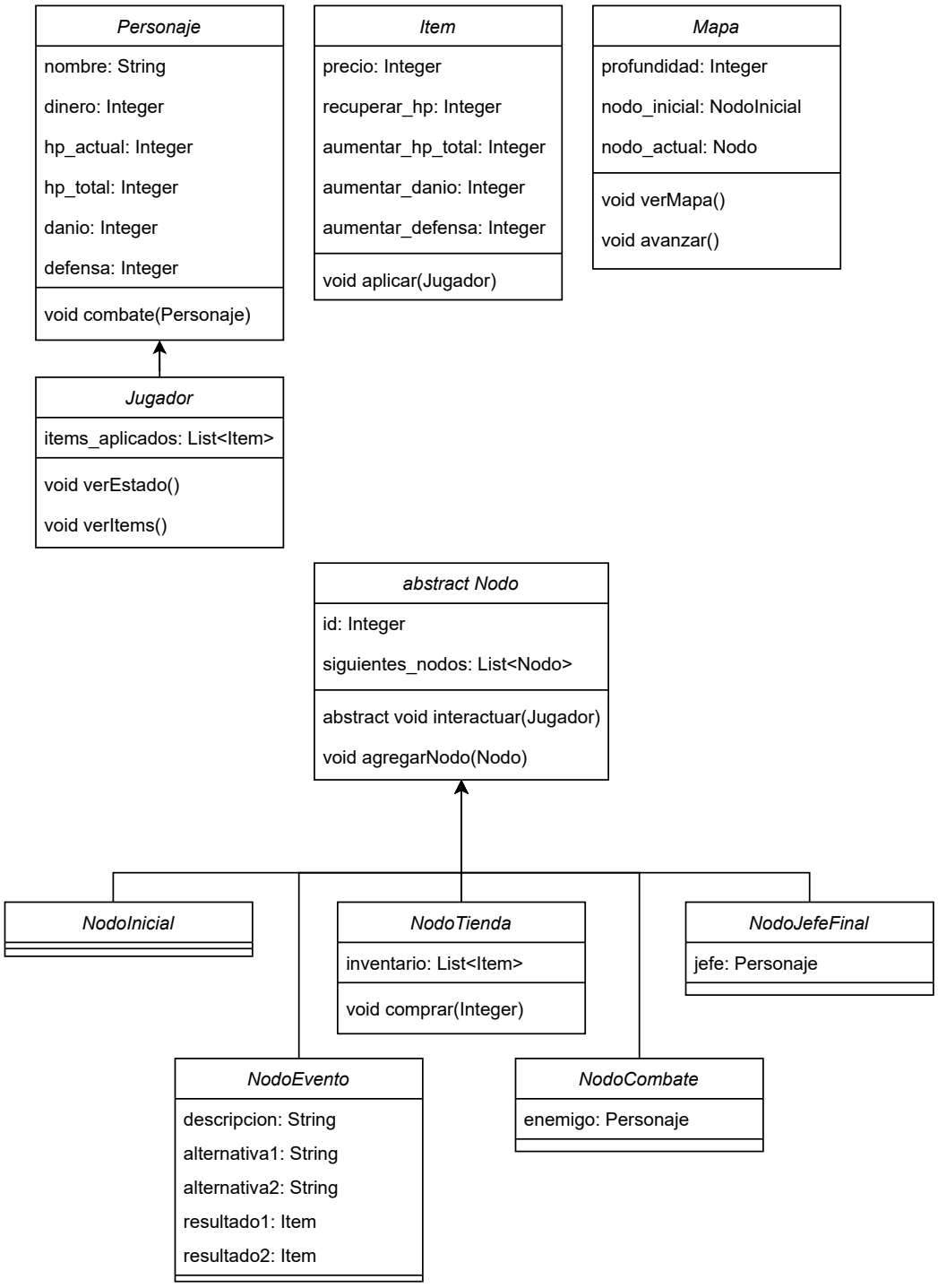

Figura 2: Clases a implementar

3.1. Personaje

- nombre: Nombre del personaje.

- dinero: Cuanto dinero tiene el personaje. En el caso del jugador representa cuanto puede gastar, en el caso de un enemigo corresponde a la cantidad de dinero que le otorga al jugador al ser derrotado.
- hp_actual: Cantidad de vida que tiene actualmente el personaje, no puede exceder la hp_total.
- hp_total: Cantidad de vida máxima que puede tener el personaje.
- danio: Cantidad de vida que le resta a su oponente al combatir (antes de aplicar defensa).
- defensa: Cantidad de daño evitado por golpe al combatir.
- combate: Realiza el combate entre dos personajes. Durante un combate se selecciona un personaje aleatoriamente para comenzar y posteriormente se alternan al atacar. Durante un ataque, el personaje defensor pierde vida equivalente al daño de su oponente menos la defensa del defensor. *El combate termina una vez la hp_actual de uno de los personajes sea menor o igual a 0.*

3.2. Jugador

Extiende la clase Personaje.

- items_aplicados: Una lista con los items que el personaje a adquirido durante su aventura.
- verEstado: Muestra al usuario los atributos actuales del jugador (nombre, dinero, hp_actual, hp_total, danio y defensa)
- verItems: Muestra al usuario los items que a adquirido.

3.3. Item

- precio: Corresponde a cuanto dinero cuesta adquirir un item desde la tienda.
- recuperar_hp: Cantidad de hp que recupera al jugador.
- aumentar_hp_total: Cantidad de hp_total que aumenta al jugador. Un aumento de hp_total no afecta a la hp_actual.
- aumentar_danio: Cantidad de daño que aumenta al jugador.
- aumentar_defensa: Cantidad de defensa que aumenta al jugador.
- aplicar: Agrega el item a la lista de items aplicados del jugador y aplica sus estadísticas.

Si lo desean los atributos que no son el precio pueden incluir valores negativos. En caso de que ocurra que hp_total es menor que hp_actual, se debe dejar hp_actual igual a hp_total (lo que puede finalizar el juego si hp_actual es menor o igual a 0).

3.4. Mapa

- profundidad: Cantidad de "pisos"que tiene el mapa (no incluye el nodo inicial ni el nodo del jefe final).
- nodo_inicial: Nodo inicial del mapa.
- nodo_actual: Nodo en el que se encuentra el jugador.

- verMapa: Permite al usuario ver el mapa completo (queda a su criterio si se muestra el mapa completo, se muestra nivel a nivel o de otra forma).
- avanzar: Le muestra al usuario los nodos a los que puede avanzar, le pide seleccionar uno y hace al jugador interactuar con ese nodo.

3.5. Nodo

- id: Identificador, opcional, pero puede ser útil para generar el mapa.
- siguientes_nodos: Lista de nodos a los cuales se puede avanzar desde el actual.
- agregarNodo: Agrega un nodo a la lista de siguientes nodos.

3.6. NodoInicial

- interactuar: Le muestra al usuario una introducción al juego (Instrucciones y pueden incluir una historia).

3.7. NodoEvento

- descripción: Pequeña historia que describe un encuentro del jugador en su aventura.
- alternativas: Dos opciones que puede tomar en el encuentro descrito.
- recompensas: Que le sucede al jugador tras tomar una de las alternativas.
- interactuar: Le muestra al usuario la descripción del evento y las alternativas que puede seleccionar (No se muestran explícitamente las recompensas asociadas a cada alternativa). Cuando el usuario selecciona una alternativa se le aplica la recompensa correspondiente.

3.8. NodoTienda

- inventario: Lista de items disponibles para comprar.
- comprar: Le aplica el item en la posición indicada al jugador.
- interactuar: Le muestra los items en el inventario de la tienda al usuario y cuanto dinero tiene a su disposición. Le permite comprar tantos items como quiera.

3.9. NodoCombate o NodoJefeFinal

Se implementan de la misma forma pero JefeFinal se inicia con un enemigo más poderosos.

- enemigo/jefe: Enemigo al cual se debe enfrentar el jugador.
- interactuar: Realiza el combate informándole al usuario los resultados de cada ataque realizado y determina al ganador.

3.10. Otros métodos y atributos

Todos los métodos y atributos mencionados deben estar implementados en la tarea, pero además deberán incluir constructores para todas las clases que los necesiten, getters y setters donde estimen necesario (**no deben acceder a los atributos directamente desde fuera de una clase, estos deben ser obligatoriamente privados**) y métodos o atributos adicionales si así lo desean.

4. Inicialización del juego

Cuando se inicia el juego se le solicita al usuario ingresar un nombre para su personaje, y se inician sus estadísticas con:

- dinero = 500
- hp_actual = 20
- hp_total = 20
- daño = 5
- defensa = 1

Posterior a esto se genera un mapa de forma aleatoria para esto se deja a su disposición el siguiente código el cual dada una profundidad, genera aleatoriamente las aristas de un mapa. Si lo desean pueden modificar este código o generar el mapa de otra forma que genere un nodo inicial, un nodo final y múltiples formas de llegar a este ultimo.

Los tipos de nodos intermedios también son determinados aleatoriamente. Cada nodo tiene intermedio tiene una probabilidad dada de ser de uno de los tipos:

- NodoEvento: 30 %
- NodoTienda: 10 %
- NodoCombate: 60 %

En los nodos de tipo Evento se inicializan con un set de descripción, alternativas y recompensas al azar. Las posibles descripciones, alternativas y estadísticas de items son arbitrarias y quedan a su criterio o creatividad.

En los nodos de tipo Tienda se deben inicializar con una cantidad aleatoria el inventario con entre 5 y 8 items, estos items pueden ser creados con estadísticas al azar (intentando que tenga valores razonables para poder progresar en el juego) o con estadísticas seleccionadas aleatoriamente de una lista de items ya creados.

Para los nodos de Combate y JefeFinal, se debe crear un enemigo aleatorio siguiendo el mismo criterio que para los items de la tienda (estadísticas aleatorias pero que permitan jugar o seleccionar aleatoriamente uno de varios personajes ya creados).

Una vez generado todo, al jugador se le muestra una introducción al juego.

5. El juego

Una vez iniciado el jugador y el mapa, el resto del juego es simple. En cada momento el jugador puede tomar una de las siguientes acciones: ver el mapa, ver sus estadisticas, ver sus items o avanzar. El juego termina una vez el jugador no puede seguir luchando (hp_actual menor o igual a 0) o el jugador derrota al jefe final y se le muestra un mensaje felicitándolo.

6. Sobre la Entrega

- Se deberá entregar un programa con los siguientes archivos:
  - `JavaQuest.java`: Este es el archivo que se debe ejecutar. Aquí se encuentra el método main.
  - `Personaje.java`
  - `Jugador.java`
  - `Item.java`
  - `Mapa.java`
  - `Nodo.java`
  - `NodoInicial.java`
  - `NodoEvento.java`
  - `NodoTienda.java`
  - `NodoCombate.java`
  - `NodoJefeFinal.java`
  - `makefile`: **Asegúrense de que su makefile funcione y especifiquen desde que directorio debe ser ejecutado si es relevante, no compilen solamente utilizando su IDE favorito y esperen que su makefile simplemente funcione.**
- Los ayudantes correctores pueden realizar descuentos en caso de que el código se encuentre muy desordenado.
- Las funciones implementadas y que no esten en el enunciado deben ser comentadas de la siguiente forma. **SE HARÁN DESCUENTOS POR FUNCIÓN NO COMENTADA**

```
/**
* Descripcion de la funcion
*
* @param a: Descripcion del parametro a
* @param b: Descripcion del parametro b
*
* @return c: Descripcion del parametro c
*/
```

- Debe estar presente el archivo MAKEFILE para que se efectué la revisión, este debe compilar el programa completo.
- Si el makefile no está bien realizado, la tarea no se revisará
- Se debe trabajar de forma individual obligatoriamente.
- **La entrega debe entregarse en .tar.gz y debe llevar el nombre: Tarea1LP_RolAlumno.tar.gz**
- **El archivo README.txt debe contener nombre y rol del alumno e instrucciones detalladas para la correcta utilización de su programa. De no incluir README se realizara un descuento.**
- La entrega será vía aula y el plazo máximo de entrega es hasta **28 de Octubre**.

- **Por cada hora de atraso se descontaran 20 pts.**
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.
- **Solo se contestaran dudas realizadas en AULA y que se realicen al menos 48 horas antes de la fecha de entrega original.**

7. Calificación

7.1. Entrega

- Implementación (20 pts)
  - Inicialización (10 pts)
    - El programa se ejecuta correctamente y se puede acceder al juego, sin embargo el jugador, el mapa o los nodos no se encuentran inicializados como se indica en el enunciado. (max 5/10 pts)
    - El programa se ejecuta correctamente, se puede acceder al juego y este se encuentra inicializado como se indica en el enunciado. (10/10 pts)
  - Turnos (10 pts)
    - El jugador puede avanzar por el mapa escogiendo un camino e interactuando con los nodos por los que pasa, pero no puede ver el mapa, sus estadísticas, sus items o algunas de las interacciones se realizan de forma incorrecta. (max 5/10 pts)
    - El jugador puede avanzar por el mapa escogiendo un camino e interactuando con los nodos por los que pasa. Se realizan todas las interacciones de forma correcta y se pueden realizar todas las acciones descritas en el enunciado. (10/10 pts)
- Personaje (9 pts)
  - Clase y atributos correctamente implementados. (1 pts)
  - Constructor correctamente implementado. (2 pts)
  - Implementa correctamente el método `void combate(Personaje)`. (6 pts)
- Jugador (8 pts)
  - Clase y atributos correctamente implementados. (1 pts)
  - Uso correcto de herencia de clases. (3 pts)
  - Constructor correctamente implementado. (2 pts)
  - Implementa correctamente el método `void verEstado()`. (1 pts)
  - Implementa correctamente el método `void verItems()`. (1 pts)
- Item (6 pts)
  - Clase y atributos correctamente implementados. (1 pts)
  - Constructor correctamente implementado. (2 pts)
  - Implementa correctamente el método `void aplicar(Jugador)`. (3 pts)
- Mapa (14 pts)
  - Clase y atributos correctamente implementados. (1 pts)
  - Constructor correctamente implementado. (5 pts)

- Implementa correctamente el método `void verMapa()`. (3 pts)
- Implementa correctamente el método `void avanzar()`. (5 pts)

- Nodo (8 pts)
  - Clase y atributos correctamente implementados. (1 pts)
  - Uso correcto de clase abstracta. (3 pts)
  - Constructor correctamente implementado. (2 pts)
  - Define correctamente el método `asbtract void interactuar(Jugador)`. (1 pts)
  - Implementa correctamente el método `void agregarNodo(Nodo)`. (1 pts)
- NodoInicial (4 pts)
  - Clase correctamente correctamente. (1 pts)
  - Uso correcto de herencia de clases (implementación método abstracto). (3 pts)
- NodoEvento (8 pts)
  - Clase y atributos correctamente implementados. (1 pts)
  - Uso correcto de herencia de clases (implementación método abstracto). (5 pts)
  - Constructor correctamente implementado. (2 pts)
- NodoTienda (11 pts)
  - Clase y atributos correctamente implementados. (1 pts)
  - Uso correcto de herencia de clases (implementación método abstracto). (5 pts)
  - Constructor correctamente implementado. (2 pts)
  - Implementa correctamente el método `void comprar(Integer)`. (3 pts)
- NodoCombate (6 pts)
  - Clase y atributos correctamente implementados. (1 pts)
  - Uso correcto de herencia de clases (implementación método abstracto). (3 pts)
  - Constructor correctamente implementado. (2 pts)
- NodoJefeFinal (6 pts)
  - Clase y atributos correctamente implementados. (1 pts)
  - Uso correcto de herencia de clases (implementación método abstracto). (3 pts)
  - Constructor correctamente implementado. (2 pts)

Se asignara puntaje parcial por funcionamiento parcialmente correcto.

7.2. Descuentos

- Falta de comentarios (-10 pts c/u Max 30 pts)
- Código no compila (-100 pts)
- Warning (c/u -5 pts)
- Falta de README (-20 pts)

- Falta de alguna información obligatoria en el README (-5 pts c/u)
- Falta de orden (entre -5 y -20 pts dependiendo de que tan desordenado)
- Día de atraso (-20 pts por cada hora de atraso)
- Mal nombre en algún archivo entregado (-5 pts c/u)