INF-253 Lenguajes de Programación
Tarea 3: Java

28 de abril de 2023

1. JavaHack

Para esta tarea deberán crear un juego llamado JavaHack utilizando programación orientada a objetos en Java. En este juego los jugadores deberán progresar, matando enemigos, subiendo niveles y obteniendo items.

JavaHack esta conformado por: Un Mundo, Personajes, Un Jugador e Items que pueden ser Equipamiento o Armas. A continuación se presenta un diagrama de clases que describe que debe contener cada uno de estos elementos.

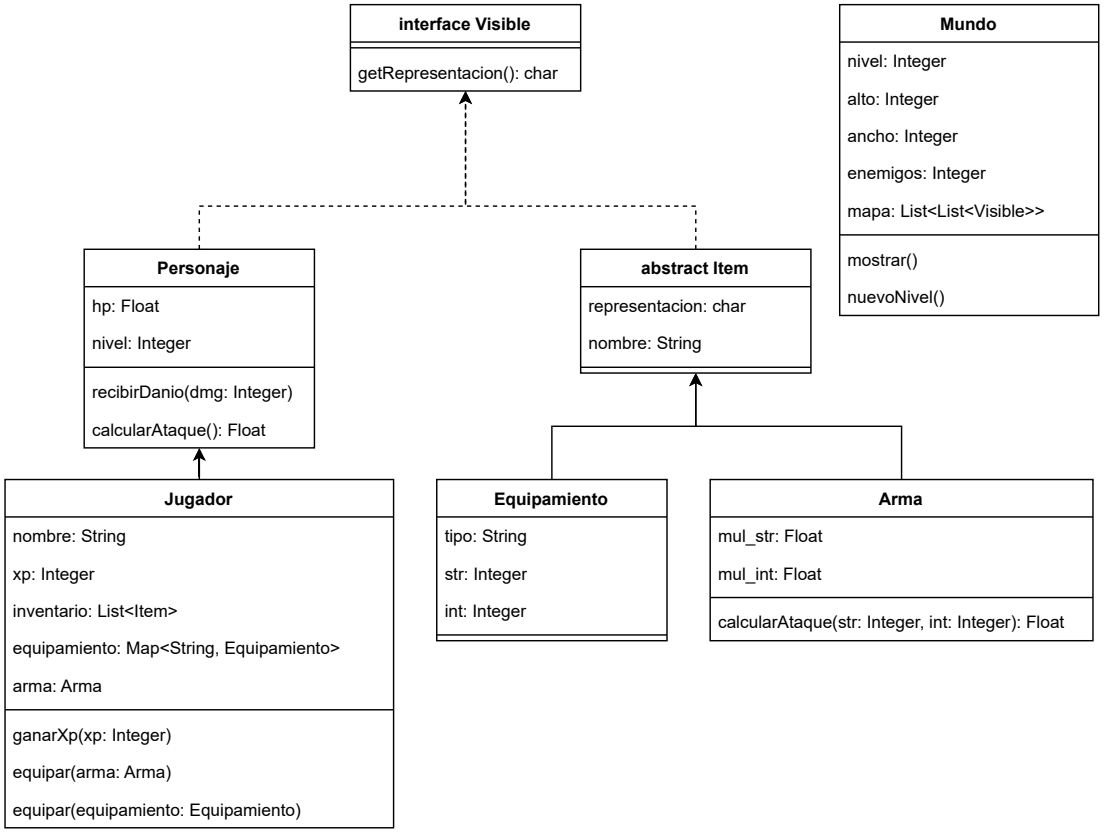

Figura 1: Clases a implementar

1.1. Mundo

- nivel: Nivel del mundo. Determina la vida y daño de los enemigos.
- alto: Alto del mapa.
- ancho: Ancho del mapa.
- enemigos: Cantidad de enemigos en el mapa.
- mapa: Array 2D con los elementos del mundo.
- mostrar(): Imprime el mapa por consola. Utiliza getRepresentacion para saber con que carácter representar cada entidad.
- nuevoNivel(): Aumenta el nivel del mundo en 1, y vuelve a generar un mapa del mismo tamaño.

1.2. Visible

- getRepresentacion(): Retorna un caracter que es utilizado para representar la entidad cuando se muestra el mundo por pantalla.

1.3. Personaje

- hp: Cantidad de salud del personaje. El personaje muere si su hp es menor o igual a 0.
- nivel: Nivel del personaje. Afecta a la hora de calcular cuanto daño realiza.
- getRepresentacion(): Los personajes se representan con la letra 'O'.
- recibirDanio(): Reduce la cantidad de hp del personaje.
- calcularAtaque(): Retorna la cantidad de daño que hace el personaje al atacar. Se calcula como $3 \cdot nivel$.

1.4. Jugador

- xp: Puntos de experiencia del jugador. Cada 100 puntos sube un nivel.
- nivel: Nivel del jugador. Afecta a la hora de calcular cuanto daño realiza.
- inventario: Lista de items que el jugador a adquirido.
- equipamiento: Objetos equipados por el jugador. Pueden ser 1 de 3 tipos Armadura, Botas o Amuleto. El jugador solo puede tener equipado un Equipamiento de cada tipo a la vez.
- arma: Arma equipada por el jugador, se utiliza para calcular el daño que realiza.
- getRepresentacion(): Los jugadores se representan con la letra 'J'.
- calcularAtaque(): Retorna la cantidad de daño que hace el jugador al atacar. Se calcula como $3 \cdot nivel + ataque_{arma}$.
- ganarXp(): Le otorga xp al jugador. Si es necesario incrementa su nivel.
- equipar(Arma): Equipa el arma al jugador.
- equipar(Equipamiento): Equipa un equipamiento al jugador. Si ya hay un equipamiento de ese tipo lo reemplaza.

1.5. Item

- representacion: Caracter con el que se representa el Item cuando esta en el mundo.
- nombre: Nombre del Item.
- getRepresentacion(): Los ítem se representan con el carácter establecido en representación.

1.6. Equipamiento

- tipo: Pueden ser 1 de 3 tipos Armadura, Botas o Amuleto.
- str: Fuerza que otorga el item. Se utiliza para calcular el atk del jugador.
- int: Inteligencia que otorga el item. Se utiliza para calcular el atk del jugador.
- getRepresentacion(): Los ítem se representan con el carácter establecido en representación.

1.7. Arma

- mul_str: Multiplicador de fuerza que otroga el item. Se utiliza para calcular el atk del jugador.
- mul_int: Multiplicador de inteligencia que otroga el item. Se utiliza para calcular el atk del jugador.
- getRepresentacion(): Los ítem se representan con el carácter establecido en representación.
- calcularAtaque(): Calcula el ataque que puede hacer el arma. Se obtiene como $str \cdot mul_{str} + int \cdot mul_{int}$.

1.8. 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.

2. Inicialización del juego

Cuando se inicia el juego se le solicita al usuario ingresar un nombre para su personaje. El jugador inicialmente tendrá:

- hp: 100
- nivel: 1
- xp: 0
- arma: “Espada Básica” con 0.5 en multiplicador de fuerza y 0.25 en multiplicador de inteligencia.

Posterior a la creación del jugador, se creara el mundo. Se le solicitara al usuario ingresar el ancho y alto del mundo a crear. El nivel inicial del mundo es 1. En la posición (0, 0) del mapa se encontrara el jugador. Luego, por cada casilla se calculara un numero aleatorio r entre 0 y 1 de forma que:

- Si $r \leq \text{mín}(0{,}05 + 0{,}01 \cdot nivel, 20)$ entonces en la casilla hay un item aleatorio (queda a su criterio como crearlos!).
- Si $\text{mín}(0{,}05 + 0{,}01 \cdot nivel, 20) < r \leq \text{mín}(0{,}2 + 0{,}01 \cdot nivel, 55)$ entonces en la casilla hay un personaje enemigo.
- En cualquier otro caso la casilla esta vacía.

Posterior a esto comienza el juego.

3. El juego

En cada turno se muestra el mapa y el jugador puede realizar una de las siguientes acciones:

- Ver estadísticas: Muestra el nombre del jugador, su nivel, su xp, su hp, su ataque, su equipamiento y su arma.
- Mover: El jugador puede moverse a una casilla adyacente dentro del mapa. Si la casilla contiene un item, el item se mueve a su inventario y sale del mapa. Si la casilla contiene un personaje enemigo, tanto el enemigo como el jugador reciben daño, si el enemigo muere, el jugador gana xp y el enemigo se elimina del mapa.
- Ver inventario: Se le muestra al jugador su inventario. El jugador puede equipar alguno de los items.
- (Si la cantidad de enemigos es 0) Subir nivel de mundo: Se sube el nivel de mundo y se regenera el mapa. El jugador se vuelve a mover a la casilla (0, 0) y gana $50 + 5 \cdot nivel$ hp adicional.

Pueden cambiar los números si es que el juego es muy difícil/lento/aburrido lo importante es que este implementadas todas las funcionalidades indicadas en este enunciado.

4. Sobre la Entrega

- Se deberá entregar un programa con los siguientes archivos:
  - `JavaHack.java`: Este es el archivo que se debe ejecutar. Aquí se encuentra el método main.
  - `Mundo.java`
  - `Visible.java`
  - `Personaje.java`
  - `Jugador.java`
  - `Item.java`
  - `Equipamiento.java`
  - `Arma.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 sin probarlo.**
- 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: Tarea3LP_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 realizará un descuento.**
- La entrega será vía aula y el plazo máximo de entrega es hasta el **19 de Mayo**.
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.
- **Solo se contestarán dudas realizadas en AULA y que se realicen al menos 48 horas antes de la fecha de entrega original.**

5. Calificación

5.1. Entrega mínima

La entrega mínima consiste de implementar la inicialización del mundo y mostrar por pantalla el mundo inicializado.

5.2. Entrega

- Implementación (20 pts)
  - Inicialización (10 pts)
    - El programa se ejecuta correctamente y se puede acceder al juego, sin embargo el jugador o el mundo no se encuentra inicializado como se indica. (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 tomar algunas de las acciones indicadas en cada turno, pero no se encuentran todas implementadas. (max 5/10 pts)
    - El jugador puede tomar cualquiera de las acciones indicadas en el enunciado. (10/10 pts)
- Mundo (15 pts)
  - Clase y atributos correctamente implementados. (1 pts)
  - Constructor correctamente implementado. (2 pts)
  - Implementa correctamente el método `mostrar`. (6 pts)
  - Implementa correctamente el método `nuevoNivel`. (6 pts)
- Visible (6 pts)
  - Interfaz correctamente creada. (3 pts)
  - Uso correcto de interfaces. (3 pts)
- Personaje (12 pts)
  - Clase y atributos correctamente implementados. (1 pts)
  - Constructor correctamente implementado. (2 pts)
  - Interfaz correctamente implementada. (2 pts)
  - Implementa correctamente el método `recibirDanio`. (4 pts)
  - Implementa correctamente el método `calcularAtaque`. (3 pts)
- Jugador (19 pts)
  - Clase y atributos correctamente implementados. (1 pts)
  - Constructor correctamente implementado. (3 pts)
  - Uso correcto de herencia de clases. (2 pts)
  - Implementa correctamente el método `ganarXp`. (3 pts)
  - Implementa correctamente el método `equipar(Arma)`. (5 pts)
  - Implementa correctamente el método `equipar(Equipamiento)`. (5 pts)

- Item (6 pts)
  - Clase y atributos correctamente implementados. (3 pts)
  - Uso correcto de clase abstracta. (3 pts)
- Equipamiento (10 pts)
  - Clase correctamente correctamente. (2 pts)
  - Uso correcto de herencia de clases. (5 pts)
  - Constructor correctamente implementado. (3 pts)
- Arma (12 pts)
  - Clase correctamente correctamente. (2 pts)
  - Uso correcto de herencia de clases. (5 pts)
  - Constructor correctamente implementado. (3 pts)
  - Implementa correctamente el método `calcularAtaque`. (2 pts)

Se asignara puntaje parcial por funcionamiento parcialmente correcto.

5.3. 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 día, -10 pts dentro de la primera hora)
- Mal nombre en algún archivo entregado (-5 pts c/u)