

INF-253 Lenguajes de Programación

Tarea 3: Java

Profesor: José Luis Martí Lara

Ayudante Cátedras: Lucio Fondón Rebolledo

Ayudante Tareas: Gabriel Carmona Tabja - Sebastián Campos Muñoz

3 de junio de 2021

1. Nueva Vida

Después de ayudar a salvar al mundo de los Objecticons, Don Fondon estaba descansando tranquilamente en su silla. Él se estaba tambaleando porque lo hacia relajarse, pero eso le costará caro. Sin darse cuenta se cae bruscamente, gira por el aire y cae en un cofre que por alguna extraña razón estaba abierto, terminando así encerrado. Él encontró una escotilla por donde puede salir, no se cuestiona el porqué hay una escotilla en el cofre, pero sigue ese camino puesto que no puede abrir el cofre. Lamentablemente, Don Fondon está agotado y tiene alucinaciones, entonces cree que es un juego donde al principio debe elegir quien ser. Por eso les pide a ustedes que lo ayuden, puesto que tiene miedo de fallar.


2. The Binding of Don Fondón

La Tarea se tratará en un juego de texto basado en el juego The binding of Isaac en el cuál el jugador deberá ayudar a Don Fondón a salir de un “laberinto” donde tendrá que enfrentarse a enemigos, abrir cofres y obtener items que lo ayudarán en su aventura. El laberinto nunca será igual, y se creará aleatoriamente en cada partida.

3. Estructura

3.1. Personaje

Un personaje es una **clase abstracta**, el cual puede corresponder a **Jugador** y **Enemigo**, donde ellos en común tendrán los siguientes atributos:

- ✓ **Nombre**: Nombre del personaje
- ✓ **Corazones**: La vida total del personaje
- ✓ **Daño**: El daño que hace  oponente (en unidades de corazones)
- ✓ **Velocidad**: Para determinar quien atacará primero

Y tiene el siguiente **método abstracto** que las clases hijas (Jugador y Enemigo) **deberán implementar**:

- ✓ **Ataque**: Realiza un ataque

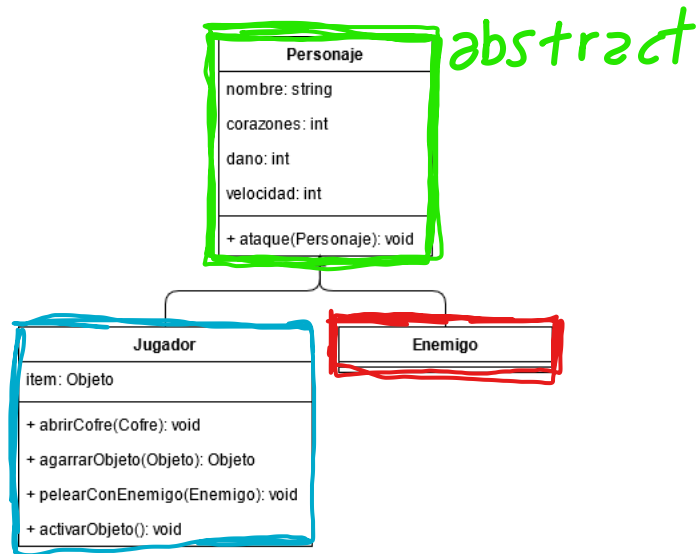


Figura 1: Personaje

3.1.1. Jugador

El jugador corresponde a una clase que tiene como padre a **personaje**, esta clase tendrá un atributo extra que corresponderá a:

Atributo { ✓ **Item**: Un objeto que puede modificar los stats del personaje

Además, tendrá los siguientes métodos:

Métodos {

- **abrirCofre(Cofre)**: Abre un cofre que esté en la habitación actual
- ✓ **agarrarObjeto(Objeto)**: Toma un objeto de la habitación actual y el objeto actual del personaje se dejará en la habitación
- ✓ **pelearConEnemigo(Enemigo)**: Comienza pelea con un enemigo
- ✓ **activarObjeto()**: Activa un objeto

Existirán tres personajes que se podrán elegir al comenzar el juego, cada personaje tiene diferentes características y un objeto al comenzar el juego:

- ① Don Mario:

 - 9 Corazones
 - 3 Daño
 - 3 Velocidad
 - Champiñón Recursivo
- ② Don Sora

 - 8 Corazones
 - 4 Daño
 - 3 Velocidad
 - Firaga Funcional

- ③
- Don NoMuerto
 - 11 Corazones
 - 2 Daño
 - 2 Velocidad
 - Espada Heredada
- ✓

3.1.2. Enemigo

El enemigo corresponde a una clase que tiene como padre a la clase **personaje**, esta clase **no** tiene atributos ni métodos extras a implementar.

Existirán distintos enemigos que tendrán diferentes características y son los siguientes:

- ①
- Leakitu
 - 7 Corazones
 - 1 Daño
 - 2 Velocidad
- ✓
- ②
- Goombation fault
 - 8 Corazones
 - 1 Daño
 - 1 Velocidad
- ✓
- ③
- SiMuerto
 - 6 Corazones
 - 2 Daño
 - 3 Velocidad
- ✓

3.2. Peleas

Las peleas serán 1 vs 1 (sin importar cuantos enemigos existan en la sala) donde comienza el personaje con más velocidad. Sólo se puede combatir con un enemigo a la vez, por lo que si la sala tiene más de un enemigo se combatirá de a 1 en 1, no se combatirá al siguiente enemigo hasta que se acabe con el enemigo actual. El combate es por turnos, en cada turno se puede atacar o usar la activa del objeto en posesión. Un ataque (a menos que se vea modificado por algún objeto) tiene 50 % de probabilidades de acertar y 50 % de probabilidades de fallar, si acierta el ataque le quita corazones al enemigo equivalente al daño del atacante. Para el caso de los enemigos, sólo pueden atacar.

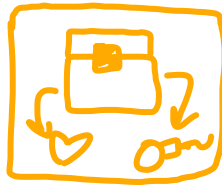
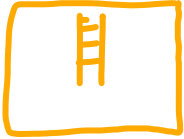
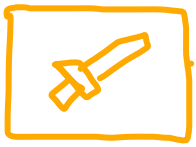
3.3. Salas

salas

El laberinto está formado por cuadrados que tienen cuatro salidas: arriba, abajo, izquierda y derecha. Cada cuadrado puede tener 5 tipos:

- Sala con enemigos: Sala que contendrá desde 1 a 3 enemigos, estos enemigos pueden ser cualquier combinación de los tres enemigos mencionados con anterioridad. La probabilidad de que aparezcan uno, dos o tres enemigos es la misma y la probabilidad de aparición de cada enemigo también es la misma (se pueden repetir enemigos).





- **Sala del tesoro:** Sala del tesoro que contendrá un **ítem** al azar (se especificarán los ítems más adelante)
- **Sala con cofre:** Sala que tiene un **cofre**, los cofres pueden: otorgar **desde 1 a 3 corazones de vida** o **una bomba que explota y te quita un corazón** (cada evento es equiprobable).
- **Sala vacía:** No tiene nada.
- **Meta:** La salida del laberinto, si el jugador llega a este recuadro gana la partida.

Por otro lado, cada tipo de sala estará implementada de la siguiente forma:

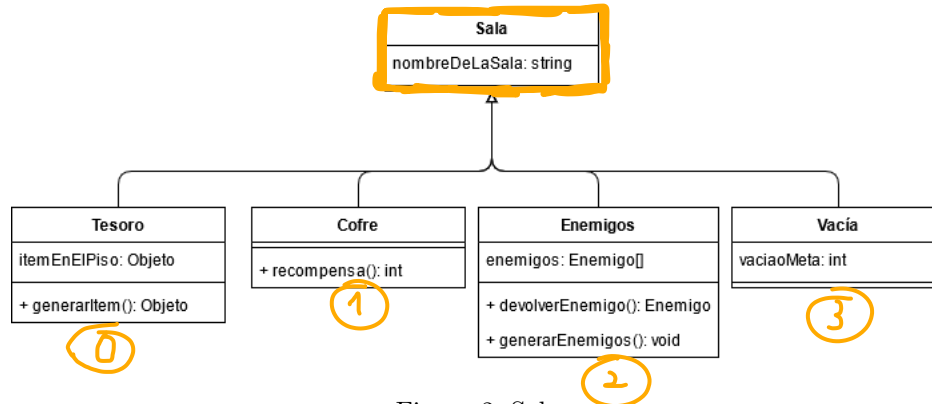


Figura 2: Salas

Dónde estos atributos y métodos se describen a continuación:

- Sala** ✓ **nombreDeLaSala:** Tendrá el nombre de la sala (Tesoro, Cofre, Enemigos y Vacía).
- Tesoro** ✓ **generarItem():** Devolverá un **Objeto** con el ítem que está en la sala.
- Tesoro** ✓ **itemEnElPiso:** **Objeto** que está en el piso de la sala.
- Cofre** ✓ **recompensa():** Devuelve un entero que señala la recompensa del cofre (**-1 para bomba, 1 para un corazón, 2 para dos corazones y 3 para tres corazones**).
- Enemigo** ✓ **enemigos:** **Arreglo** con los **enemigos** de la sala.
- Enemigo** ✓ **generarEnemigos():** **Genera los enemigos de la sala** y los **agrega a la lista de enemigos**.
- Enemigo** ✓ **devolverEnemigo():** Devuelve el último enemigo del arreglo y **lo borra del arreglo**.
- Vacío** ✓ **vaciaoMeta:** Entero que señala el tipo de sala, **0 para vacía 1 para meta**.

3.4. Laberinto

Para facilitar la creación del laberinto, se considerará **un laberinto como una matriz de salas 5x5**. Al iniciar una partida, **se crearán 10 salas dispersas en la matriz de forma aleatoria** donde **cada tipo de sala tiene la misma probabilidad de ocurrir** (con la única excepción de que **debe existir siempre una sola meta** y la **sala de inicio SIEMPRE debe estar vacía**). **TODAS las salas deben ser accesibles**, esto quiere decir que **puedo llegar a cualquier sala si es que recorro el suficiente número de salas**.

	Enemigo	Cofre	Cofre	Tesoro
	Vacio			Enemigo
Enemigo	Tesoro			Meta
Inicio				

Figura 3: Ejemplo de laberinto válido

	Enemigo		Cofre	Tesoro
	Vacio			Enemigo
Enemigo	Tesoro	Meta		
Inicio				Cofre

inaccesibles

Figura 4: Ejemplo de laberinto inválido

Además de las 10 salas, cualquier otro espacio en la matriz será inaccesible. Finalmente, la clase que tendrá la matriz será la siguiente:

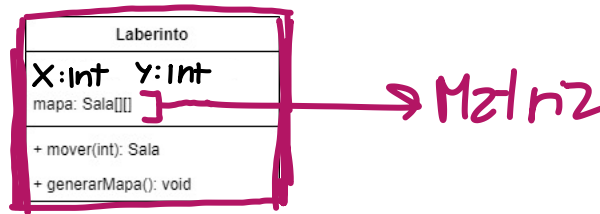
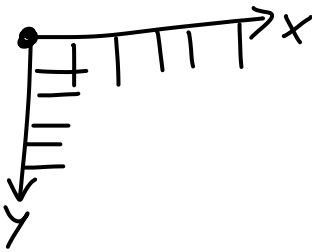


Figura 5: Clase Laberinto

Dónde:

- ✓ posicionActual: La posición en la matriz donde está el personaje. (x) (y)
- ✓ mapa: Matriz de salas.
- ✓ mover(int): Recibe un entero y regresa la sala a la cuál llegó. El entero debe ser 0 hacia arriba, 1 hacia la derecha, 2 hacia abajo y 3 hacia la izquierda. Cada vez que el personaje se mueva debe actualizar posicionActual, excepto en el caso de que llegue a un espacio de la matriz sin sala, en ese caso debe retornar null y no mover al personaje de la casilla actual.
- ✓ generarMapa(): Rellenará la matriz con 10 salas y generará el laberinto.

3.5. Objetos

Los objetos son una interfaz que se dividen en dos tipos de objetos, pasivos y activables. Los pasivos afecta a los corazones, el daño y la velocidad mientras uno tenga el objeto. Los activables son objetos que se para que su efecto se realice deben activarse.

La interfaz posee dos métodos a implementar:

↳ Activar Objeto()

Metodos {
 ✓ Efecto: le aplica el efecto al personaje
 ✓ QuitarEfecto: le quita el efecto al personaje

Los objetos pasivos son los siguientes:

- ① Espada Heredada: este objeto aumenta el ataque del personaje en 1 y disminuye los corazones en 1.

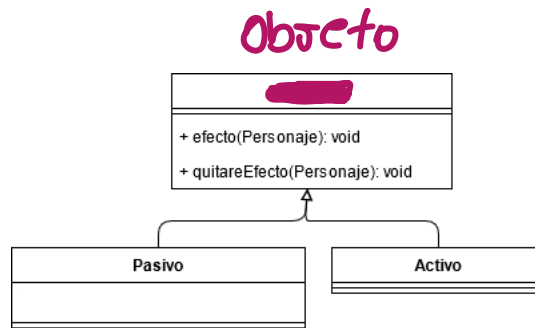


Figura 6: Interfaz objeto y las clases Pasivo y Activo

- ② ■ Firaga Funcional: cuando se realice el ataque, existe un 10 % de probabilidad de que el daño realizado se multiplique por 1,5, redondeado hacia arriba.
- ③ ■ Caparazón de Compilación: este objeto aumenta los corazones en 2 y disminuye el daño en 1.

— Los objetos activos son los siguientes:

- ① ■ Champiñón Recursivo: al activarse quita 2 de vida para que el ataque haga 3 más de daño.
- ② ■ Shine: Al activarse tiene una probabilidad de 10 % de añadir 5 de daño al ataque y perder 2 corazón.

4. Funcionamiento

El funcionamiento principal del programa debe ocurrir en un archivo llamado **Main.java**. El **main** del programa debe permitir jugar el juego (elegir personaje, explorar el mapa, pelear, agarrar objetos, abrir cofres, etc). El programa debe tener un menú por consola con el cual se pueda interactuar y jugar (favor de hacer los menús lo más simple posibles para facilitar la revisión). Además, se debe imprimir por pantalla todo lo que está sucediendo en la partida (a que sala me moví, contra quien estoy peleando, mis stats, que pasa cuando abro un cofre, etc). Por otro lado, se debe poder ver en cualquier momento los stats de tu personaje, tu vida actual y el objeto que tienes equipado. Finalmente, no se revisarán tareas que no posean el main implementado (pueden entregar main incompletos, pero el main debe poder utilizarse para jugar),

5. A entregar

- Main.java
- Laberinto.java
- Sala.java, Tesoro.java, Cofre.java, Enemigos.java y Vacía.java
- Objeto.java, Pasivo.java y Activo.java
- Personaje.java, Jugador.java, Enemigo.java
- makefile
- readme.txt

6. Sobre Entrega

- El código debe venir ordenado.
- Se darán puntos por creatividad :D.
- Cada método debe llevar un comentario que indique nombre de la función, parámetros que recibe, una breve descripción de lo hace y lo que retorna.
- Debe estar presente el archivo **makefile** para que se efectúe la revisión, este debe compilar **TODOS** los archivos. Si su código viene sin makefile (o el makefile no funciona), no se revisará su tarea.
- El trabajo es individual.
- La entrega debe realizarse en tar.gz y debe llevar el nombre: **Tarea3LP_RolIntegrante.tar.gz**
- El archivo README.txt debe contener nombre y rol del alumno e instrucciones detalladas para la compilación y utilización de su programa. **SE DESCONTARÁ SI FALTA ALGO DE LO SEÑALADO.**
- El no cumplir con las reglas de entrega conllevará un descuento máximo de 30 puntos en su tarea.
- La entrega será vía aula y el plazo máximo de entrega es hasta el **jueves 24 de junio a las 23:55 hora aula**.
- Por cada día de atraso se descontarán 20 puntos.
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.

7. Calificación

- Personaje (25 pts):
 - Jugador (10 pts)
 - Enemigo (10 pts)
- Sala (40 pts):
 - Tesoro (10 pts)
 - Cofre (10 pts)
 - Enemigo (10 pts)
 - Vacía (10 pts)
- Objetos (20 pts)
 - Pasivo (10 pts)
 - Activo (10 pts)
- Laberinto (10 pts)
- Main (10 pts)
- Warning (-2 pts c/u, Max 20pts)
- No compila: Recorrección