Carlos Morlans

JUEGO RPG

Índice

i. Descripción de la idea del proyecto.

Para comenzar con la descripción de la idea comentar que soy una persona bastante fan de prácticamente todos los tipos de videojuegos sea más complejos o menos. Partiendo de esto lo primero que se me vino a la cabeza fue hacer un juego simple de avanzar y pegarte con un enemigo, pero claro luego me entraron las dudas si era lo mejor porque podía ser bastante complicado, pero consulté con algunas personas y me animaron a hacerlo.

Después de esta introducción comentar que la idea era y es tener un juego en el que sea un personaje con el nombre que elijas y de ahí empezar el juego encontrándote con distintos tipos de enemigos con sus atributos que en este caso son vida y ataque.

Los enemigos tienen los mismos atributos que el personaje por lo que había que marcar unas estadísticas que aparecen en consola y una vez hecho empezaba el juego pudiendo atacar y huir, esa era mi idea y asi hice.

Si que es verdad, que el varias cosas fueron apareciendo sobre la marcha como el número limitado de huidas que puedes hacer como jugador etc..

ii. Explicación del desarrollo del programa.

Empezando por la clase main (Juego) es donde esta los systems.out que aparecen por pantalla nada mas darle a runear el código en el que te da la bienvenida con intro de la historia etc..

Después marque los array de los enemigos con sus atributos para luego ir recorriendo y aparezcan enemigos aleatorios con su vida y daño.

```
String[] nombresFaciles = { "Slime", "Ogrete", "Esqueleto", "Sombrio" };
int[][] atributosFaciles = { { 42, 12 }, { 70, 10 }, { 35, 15 }, { 60, 15 } };

String[] nombresEnemigos = { "Goblin", "Orco", "Troll", "Zombie" };
int[][] atributosEnemigos = { { 90, 32 }, {130, 30}, { 100, 25}, { 115, 23 } };
```

En el siguiente while es para que si eres nivel menos de nivel 5 aparecen unos enemigos y si eres más del 5 te salgan otros enemigos con más dificultad.

```
while (personaje.getNivel() < 10) {
        if (personaje.getNivel() < 5) {
            lucharContraEnemigo(personaje, nombresFaciles, atributosFaciles, random,
huidasRestantes);
```

```
        } else {
            lucharContraEnemigo(personaje, nombresEnemigos, atributosEnemigos, random,
huidasRestantes);
        }
    }
```

Despues hay mas system.out con más información del juego historia y demas y se añade el jefe final con sus atributos

```
Enemigo jefeFinal = new Enemigo("Willow", 275, 70);
huidasRestantes = 0;
boolean victoria = Combate.empezarBatalla(personaje, jefeFinal, huidasRestantes);
if (!victoria) {
        System.out.println("MORITE MALARAZA");
        System.out.println("Sera la proxima vez");
}
```

Y al final de esta clase aparece el metodo lucharContraEnemigos que se utiliza para la randomizacion de enemigos a la hora de pelear con el boolean para !victroria, si es igual a false en la consola indica “Has sido derrotado”

```
private static void lucharContraEnemigo(Personaje personaje, String[] nombres, int[][]
atributos, Random random, int huidasRestantes) {
        int indice = random.nextInt(nombres.length);
        Enemigo enemigo = new Enemigo(nombres[indice], atributos[indice][0],
atributos[indice][1]);
        //System.out.println("Te encuentras con un " + nombres[indice]);
        boolean victoria = Combate.empezarBatalla(personaje, enemigo, huidasRestantes);
        // !victoria es lo mismo que victoria == false
        if (!victoria) {
                System.out.println("Has sido derrotado...");
                System.exit(0);
        }
}
```

En la Clase Personaje, se comienza marcando los atributos del personaje/jugador y luego se crea el constructor para inicializar los atributos en la clase.

```
// Variables - atributos del personaje
public int vida = 100;
public int daño = 20;
public int nivel = 1;
String nombre;
public int huidas = 3;
// constructor de la clase personaje
public Personaje(int vida, int daño, int nivel, String nombre, int huidas) {
        this.vida = vida;
        this.daño = daño;
```

```
        this.nivel = nivel;
        this.nombre = nombre;
        this.huidas = huidas;
}
```

Además creé un constructor vacío por si más adelante cuando desarrollaba el programa creaba objetos sin valores iniciales específicos.

Luego añadí los get y set de cada uno de los atributos que necesitase añadiendo un método de recibirDaño y subirNivel. Para terminar le metí un método llamado stats para que por consola salieron los atributos actualizados usando los get.

```
public void recibirDaño(int dañoEnemigo) {
                vida -= dañoEnemigo;
        }
public static void subirNivel(Personaje personaje) {
                personaje.setDaño(personaje.getAtaque() + 10);
                personaje.setVida(personaje.getVida() + 20);
                personaje.setNivel(personaje.getNivel() + 1);
        }
public void stats(Personaje personaje) {
                System.out.println("Estadísticas del personaje:");
                System.out.println("Vida del personaje: " + personaje.getVida());
                System.out.println("Ataque del personaje: " + personaje.getAtaque());
                System.out.println("Nivel del personaje: " + personaje.getNivel());
        }
```

La clase Enemigo, es practicamente igual que la clase personaje solo que tiene menos atributos y tiene menos metodos como por ejemplo no tiene el metodo de subirNivel ya que los enemigos no tienen el atributo subir de Nivel, pero si tienen el metodo recibirDaño y stats como el personaje

```
public void recibirDaño(int dañoPersonaje) {
        vida -= dañoPersonaje;
}

public void stats(Enemigo enemigo) {

        System.out.println("Estadisticas del enemigo:");
        System.out.println("Vida del enemigo: " + enemigo.getVida());
        System.out.println("Ataque del enemigo: " + enemigo.getAtaque());

}
```

Como ultima clase Combate (no comento la clase colores en este apartado, lo comentare en el apartado 3), comenzamos importando el escaner porque sea clave a la hora de decidir cada una de las decisiones del jugador.

Empezamos con el método empezarBatalla que empezó siendo void y terminó siendo boolean para saber cuando acaba el juego. este método comenta la salida por consola de las estadísticas del personaje y de los enemigos mas aparte un while en el que se usa mientras yo y el enemigo tengamos más de 0 de vida siga el bucle.

Dentro de este while se encuentra el menú de pelear o huir, a recalcar en uno de los if sirve para que te vaya restando huidas ya que uno de los añadidos es que el jugador tenga 3 huidas por lo que en este if se van restando las huidas posibles del jugador. Con los return true es como si se repitiese todo lo de este método.

```
if (decision == 2) {
        if (jugador.huidas > 0) {
        jugador.setHuidas(jugador.getHuidas() - 1);
        System.out.println("Huiste del combate. Te quedan " + (jugador.getHuidas())+ "
huidas.");
        return true;
        } else {
        System.out.println("No puedes rendirte mas, debes pelear como un hombre");
        return true;
        }
}
```

Siguiendo con el método pelear es donde aparece la lógica del combate, aqui se usan los métodos jugador.subirNivel, jugador.stats y enemigos.stats (con gets) y enemigo.recibirDaño, jugador.recibirDaño. (Adjunto captura ya que era mucho espacio el escribir las partes del método.

```
public static void pelear(Personaje jugador, Enemigo enemigo) {

    System.out.println("Empiezas tu!");
    enemigo.recibirDaño(jugador.getAtaque());
    System.out.println("Has hecho: " + jugador.getAtaque() + " puntos de daño");

    if (enemigo.getVida() <= 0) {

        System.out.println("Le quedan 0 puntos de vida");
        System.out.println("Muy bien, has derrotado al monstruo!");
        jugador.subirNivel(jugador);
        System.out.println("--------------------------------------");
        System.out.println(
                Colores.VERDE + "Subiste de nivel! Ahora eres nivel: " + jugador.getNivel() + Colores.RESET);
        System.out.println("--------------------------------------");
        jugador.stats(jugador);
    } else {
        System.out.println("Al enemigo le quedan " + enemigo.getVida() + " puntos de vida");
        jugador.recibirDaño(enemigo.getAtaque());
        System.out.println("El enemigo te ha hecho: " + enemigo.getAtaque() + " de daño ");

        if (jugador.getVida() <= 0) {
            System.out.println("Eres un pringado has muerto");

        } else {
            System.out.println("Te quedan " + jugador.getVida() + " puntos de vida");
        }

    }
```

Un comentario en esta clase es que cree un método para huir que terminó no siendo útil ya que lo añadí en el if de empezarBatalla.

iii. Notas importantes sobre el diseño y la implementación.

Empecé creando un boceto del posible diagrama de clases que podía tener el programa, para empezar desde ahí teniendo una idea visual de lo que quería implementar.

DIAGRAMA DE CLASES

- Clase: juego con método lucharContraEnemigos.
- Clase: personaje con atributos (vida, daño, nivel, nombre, y huidas) y métodos (atacar, huir , subirNivel, recibirDaño y estadisticas)
- Clase: Enemigo con atributos (vida, daño, nombre) y métodos (atacar, recibirDaño y estadisticas)
- Clase: combate con atributos (turnos) y métodos (empezarBatalla y pelear)

  LAS RELACIONES ENTRE CLASES

  - Juego contiene las demas clases a juego por composición
  - Combate tiene relación de dependencia con personaje y enemigo ya que sin estas clases no sirve.
  - Personaje y Enemigo estan unidos por herencia

Algunas de las implementaciones para que tenga un diseño más atractivo es:

-
```
try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
```
Sirve para parar el hilo 1000 milisegundos y asi no sale todo por consola de golpe y va poco a poco.

-
```
package CLASES;
public class Colores {
        public static final String RESET = "\u001B[0m";
        public static final String ROJO = "\u001B[31m";
        public static final String VERDE = "\u001B[32m";
        public static final String AMARILLO = "\u001B[33m";
```

```
        public static final String AZUL = "\u001B[34m";
}
```

Esta clase está creada para poder añadir colores a los System.out.println para que sea más visual y no siempre del mismo color, y se añade asi.

```
System.out.println(Colores.VERDE + "Subiste de nivel! Ahora eres nivel: " +
jugador.getNivel() + Colores.RESET);
```

Abres el system.out.println y añades el color que quieras(que tengas añadido en la clase color) pones el texto que quieres personalizar y terminas con Colores.RESET para volver al color default el resto de consola, en este caso negro.

-

Use transformador de texto en ascii para darle un poco de encanto en cuanto al "Bienvenido" y al "The end", sumándole colores.

iv. Horas estimadas de trabajo invertidas

Para comentar el número de horas, decir, que ha habido varias etapas (es complicado porque algunas de estas etapas no se como calcularlo ya que ha sido dias de poco en poco):

- **Crear un diagrama de clases básico (boceto) = 1 hora aprox.**
- **Buscar información relevante, vi un par de videos en inglés de cómo podría hacerse (con un nivel más elevado que el mio pero para pillar posibles ideas) = 2 horas aprox.**
- **Preguntar a la I.A dudas que me surgen mientras lo hacía = ? .**
- **Comentar con conocidos para resolver alguna duda que la I.A no pudo solucionar = 3 horas.**
- **Crear el código = entre 17 y 22 horas aproximadamente.**
- **Hacer este documento = 3 horas aprox.**

TOTAL= entre 25 y 30 horas aproximadamente.

v. Problemas o errores encontrados durante el desarrollo y cómo los habéis solucionado.

Algunos de los errores más comunes que he tenido, son errores de sintaxis ya que algunas veces son difíciles de detectar por despiste o lo que sea.

Un error que tuve era sobre huidasRestantes, que la intencion es que fuesen reduciendo de 3 a 0 y cuando lo ejecutaba restaba y se quedaba en 2 por mucho que huyas como en la imagen a continuación

```
Introduce el nombre de tu personaje:
carlos
Comienza tu aventura...
Te encuentras con un Slime
---------------------------------------
Oh no un Slime, que quieres hacer:
---------------------------------------
Estadisticas del personaje:
Vida del personaje: 100
Ataque del personaje: 20
Nivel del personaje: 1
---------------------------------------
Estadisticas del enemigo:
Vida del enemigo: 42
Ataque del enemigo: 12
---------------------------------------
1.pelear:
2.huir:

2

Huiste del combate. Te quedan 2 huidas.
Te encuentras con un Sombrio
---------------------------------------
Oh no un Sombrio, que quieres hacer:
---------------------------------------
Estadisticas del personaje:
Vida del personaje: 100
Ataque del personaje: 20
Nivel del personaje: 1
---------------------------------------
Estadisticas del enemigo:
Vida del enemigo: 60
Ataque del enemigo: 15
---------------------------------------
1.pelear:
2.huir:

2

Huiste del combate. Te quedan 2 huidas.
Te encuentras con un Ogrete
---------------------------------------
Oh no un Ogrete, que quieres hacer:
---------------------------------------
Estadisticas del personaje:
Vida del personaje: 100
Ataque del personaje: 20
Nivel del personaje: 1
---------------------------------------
Estadisticas del enemigo:
Vida del enemigo: 70
Ataque del enemigo: 10
---------------------------------------
1.pelear:
2.huir:
```

La solución a este error, contacte con el profesor que me recomendó en la clase de personaje añadirle los siguientes métodos

```
public int getHuidas() {
	return huidas;
}

public void setHuidas(int huidas) {
	this.huidas= huidas;
}
```

Esto me ayudara a poder poner el getHuidas y se fuese actualizando el int huidas y almacenando a medida que se iba restando de uno en uno, dejando así el if.

```
if (jugador.huidas > 0) {
	jugador.setHuidas(jugador.getHuidas() - 1);
	System.out.println("Huiste del combate. Te quedan " + (jugador.getHuidas())+ "
huidas.");
	return true;
}
```

Otro problema que arregle, es que cree un método en la clase personaje de avanzar que al final termine sin usarlo ya que era poco eficaz y no tenia porque esta en esa clase por lo que en la clase de combate en el método de empezarBatalla se lo añadí para que el código fuese más limpio y óptimo.
Está en modo comentario ya que lo deje en el código para que se vea lo que use y deje de usar como lo que añadi del método huir en el apartado 2.
Imagen de como era:

```
// metodo avanzar
/*
 * public void avanzar() { System.out.println("Avanzas una casilla");
 * System.out.println("Oh no un monstruo, que quieres hacer: ");
 * System.out.println(); // salto de linea como si fuese //n
 * System.out.println("1.pelear: "); System.out.println("2.huir: "); int
 * decision = sc.nextInt();
 *
 * if (decision == 1) { System.out.println("Vamos alla!"); // de aqui pasas al
 * metodo combate
 *
 * } else { System.out.println("buena decision "); avanzar(); // para que se
 * repita el metodo }
 *
 * }
 */
```

imagen de como quedo:

```java
public static boolean empezarBatalla(Personaje jugador, Enemigo enemigo, int huidasRestantes) {

    Scanner scanner = new Scanner(System.in);

    int turno = 1;

    //System.out.println("--------------------------------------");
    System.out.println("Oh no un " + enemigo.getNombre() + ", que quieres hacer: ");
    System.out.println("--------------------------------------");
    jugador.stats(jugador);
    System.out.println("--------------------------------------");
    enemigo.stats(enemigo);

    // mientras yo y el enemigo tengamos mas de 0 de vida seguimos en el bucle
    while (enemigo.getVida() > 0 && jugador.getVida() > 0) {
        System.out.println("--------------------------------------");
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("1.pelear: ");
        System.out.println("2.huir: ");
        System.out.println();

        int decision = scanner.nextInt();

        System.out.println();

        if (decision == 1) {
            System.out.println("Turno " + turno++);
            pelear(jugador, enemigo);
        }

        if (decision == 2) {
            if (jugador.huidas > 0) {
                jugador.setHuidas(jugador.getHuidas() - 1);
                System.out.println("Huiste del combate. Te quedan " + (jugador.getHuidas())+ " huidas.");
                return true;
            } else {
                System.out.println("No puedes rendirte mas, debes pelear como un hombre");
                return true;
            }

        }
    }
```

vi. Cualquier otro detalle relevante sobre el proceso.

En este apartado voy a comentar los aspectos que quería implementar pero no fue posible. La opción de curar y que al subir de nivel halla la probabilidad de recibir una poción, ya que cuando subes de nivel las estadísticas no se te reinician, es decir, si terminas el combate con 80 de vida y el subir de nivel te da 20 no se te reinicia la salud a 100 y se te añade los 20 sino que se te añade los 20 puntos a la vida restante después del combate. Con estas pociones se le podría subir un poco más el nivel de dificultad de los enemigos ya que podrías curarte.

Estas posiciones podrían tener tres tipos: las que curan 10, las que curan 25 y las que curan 50.

Esta opción la implementaré más adelante ya que tengo intención de continuarlo como proyecto personal.