

ARGH! - Juego de Rol

Detalles de Implementación

Requisitos de POO:

1. Relaciones entre Clases:

Composición: La clase *Argh* contiene objetos de *Personaje*, *Enemigo*, *Tendera* y *Producto*.

```
Tendera tendera = new Tendera(items:null, abierto:false);
```

```
● ● ●
1 Enemigo[] listaEnemigosMarinos = {
2     new EnemigoMarinoComun(100, 10, 10, 20, 30, 100, 25, 0, 20, false, 15, "Amonite"),
3     new EnemigoMarinoComun(100, 15, 15, 15, 20, 100, 25, 0, 15, false, 20, "Trilobite"),
4     new EnemigoMarinoComun(100, 20, 20, 10, 10, 100, 25, 0, 10, false, 10, "Zooplacton")
5 };
```

Para los enemigos, los hemos contenido en un array, para poder usar su índice y generarlos de forma aleatoria.

```
● ● ●
1 private static Producto minipocion = new Producto("Minipoción", 0, 20);
2     private static Producto pocion = new Producto("Poción", 0, 30);
3     private static Producto superpucion = new Producto("Superpoción", 0, 60);
```

En el caso de los productos, se han creado como atributos de la clase Argh para poder acceder a ellos desde cualquier función, esté o no dentro del main.

Agregación: Los personajes tienen objetos en el inventario. A su vez, el inventario es un array de productos con 7 posiciones, es decir, el jugador puede llevar hasta 7 objetos diferentes a la vez (El inventario es creado como atributo de la clase personaje).

```
protected Producto[] inventario = new Producto[7];
```

Uso: Con un sistema de combate con interacciones entre personajes en los que se usan y modifican otras clases.

2. Herencia:

- **Jerarquía de enemigos:** *EnemigoMarino* y *EnemigoTerrestre* heredan de *Enemigo*, al igual que *EnemigoMarinoComun* y *EnemigoMarinoJefe* heredan de *EnemigoMarino*; y *EnemigoTerrestreComun* y *EnemigoTerrestreJefe* heredan de *EnemigoTerrestre*.
- **Tipos de personajes:** Diferentes clases de personajes heredan de *Personaje*: Grumete y Capitán.

3. Polimorfismo:

- Métodos de ataque sobrescritos en las subclases de enemigos y personajes. Además de diferentes atributos para cada tipo de enemigo.

```
● ● ●  
1 abstract public void ataque1(Personaje personaje);  
2 abstract public void ataque2(Personaje personaje);  
3 abstract public void ataque3(Personaje personaje);  
  
public class EnemigoMarinoComun extends EnemigoMarino {  
    @Override  
    public void ataque1(Personaje personaje) {
```

- Constructores de subclases creados como el tipo padre/madre.

```
● ● ●  
1 Personaje grumete = new PersonajeGrumete(nombre, genero,  
    100, 20, 20, 20, 20, 0, 0, 1000000, 3, 0, 0, false,  
    inventario, null, false, false, false, rolSeleccionado,  
    contadorCocinero, contadorArtillero, contadorCubierta);
```

4. Clases Abstractas:

- La clase base *Enemigo* es abstracta.

```
abstract public class Enemigo {
```

- La clase base *Personaje* es abstracta.

```
public abstract class Personaje {
```

Al ser abstractas, no se han instanciado los constructores de estas clases en ningún momento de la aplicación.

Estructura de Datos con Arrays:

- Listas de enemigos según tipo (*listaEnemigosTerrestres*, *listaEnemigosMarinos*).

```
● ● ●  
1 Enemigo[] listaEnemigosMarinos = {  
2     new EnemigoMarinoComun(100, 10, 10, 20, 30, 100, 25, 0, 20, false, 15, "Amonite"),  
3     new EnemigoMarinoComun(100, 15, 15, 15, 20, 100, 25, 0, 15, false, 20, "Trilobite"),  
4     new EnemigoMarinoComun(100, 20, 20, 10, 100, 25, 0, 10, false, 10, "Zooplacton")  
5};
```

- Sistema de inventario basado en arrays.

```
protected Producto[] inventario = new Producto[7];
```

Interfaz de Terminal:

1. Selección de Personaje:

- Los jugadores pueden elegir entre *Capitán* o *Grumete*, basados en un modo de dificultad personalizados para cada uno de ellos (difícil/fácil respectivamente).
- Cada personaje tiene atributos predefinidos.

```
protected int experiencia = 0; // Experiencia que aumenta y sirve para subir de nivel.  
protected int nivel = 0; // Nivel del personaje, según el nivel, más fuerte se hace.  
protected int monedas = 0; // El dinero que tiene el jugador para poder comprar en la tienda.  
protected int grumetesRestantes; // La cantidad de oportunidades (vidas) restantes que le queda al jugador.  
protected int islasConquistadas = 0; // El número de islas conquistadas (ha vencido al jefe).  
protected int enemigosDerrotados = 0; // Número de enemigos comunes derrotados (estadística).  
protected boolean estaEnCombate = false; // Cuando entra en combate, este atributo empieza a valer "true".  
protected boolean pedoActivado = false; // Pasa a true cuando activan el objeto pedo.  
protected boolean estaSомнoliento = false; // Detecta si el jugador tiene el estado "sомнoliento".  
protected boolean estaSangrando = false; // Detecta si el jugador tiene el estado "sangrado".  
protected int barrilesDisponibles = 0; // Barriles disponibles para la apertura del jugador.
```

2. Sistema de Combate:

- Combate por turnos con diferentes opciones para poder elegir en cada turno (atacar, estadísticas, usar objetos o huir).
- Comparación de valores de ataque y defensa. Se comparan los valores con una fórmula matemática para averiguar el valor de daño que infinge el enemigo o jugador al rival.



```
1 personaje.setVida(personaje.getVida() - (int) (0.5 * (tipoDaño - personaje.getTipoResistencia()) + (dañoBase * caracteristica / 100)));
```

- Uso de objetos durante el combate.



```
1 private static void gestionarUsoObjeto(Personaje personajeActivo, Scanner sc) {  
2     boolean dentroInventario = true;  
3     while (dentroInventario) {  
4         int opcionObjeto = obtenerOpcionInventario(personajeActivo, sc);  
5         if (opcionObjeto != 8) {  
6             limpiarPantalla();  
7             if (usarObjetoCombate(personajeActivo.getInventario(), opcionObjeto, personajeActivo, sc)) {  
8                 dentroInventario = false;  
9                 break;  
10            }  
11        } else {  
12            dentroInventario = false;  
13            limpiarPantalla();  
14        }  
15    }  
16}
```

- Sistema de vida con condiciones de victoria/derrota con mensajes personalizados para cada uno de los derrotados

```
public static void muerteCombate(Personaje personajeActivo, Scanner sc) {
```

3. Información del Personaje:

- Visualización de estadísticas tanto fuera como dentro del combate, con diferentes mensajes para cada uno de ellos.



```
1 public void estadisticas() {
2     System.out.println("████████████████████████████████");
3     System.out.println("||          A R G H          ||");
4     System.out.println("||          ESTADÍSTICAS JUGADOR ||");
5     System.out.println("||          ██████████");
6     System.out.println("    Vida: " + vida + "                ");
7     System.out.println("    Daño Físico: " + dañoFisico + "            ");
8     System.out.println("    Daño Mágico: " + dañoMagico + "            ");
9     System.out.println("    Resistencia Física: " + resistenciaFisica + "        ");
10    System.out.println("    Resistencia Mágica: " + resistenciaMagica + "        ");
11    System.out.println("    Velocidad: " + velocidad + "            ");
12    System.out.println("    Nivel: " + nivel + "                ");
13    System.out.println("    Experiencia: " + experiencia + "/100 (" + (100 - experiencia) + " para subir)  ");
14    System.out.println("    Monedas: " + monedas + "            ");
15    System.out.println("    Objeto Equipado: " + (objetoEquipado != null ? objetoEquipado.getNombre() : "Ninguno"));
16    System.out.println("████████████████████████████████");
17 }
18 }
```

- Gestión del inventario, en el que poder equipar y desequipar un objeto a conveniencia para usarlo en un combate, en el que se detecta cada tipo de objeto por separado con su cantidad correspondiente.

```
public static void menuInventario (Personaje personaje) {
```

- Sistema de experiencia y niveles. En el que los enemigos proporcionan experiencia al jugador tras ser derrotados, haciendo que pueda subir de nivel, aumentando todas sus estadísticas.



```
1 if (enemigo.getVida() <= 0) {
2     // Damos las recompensas al jugador
3     personajeActivo.setMonedas(personajeActivo.getMonedas() + enemigo.getDineroDado());
4     personajeActivo.setExperiencia(personajeActivo.getExperiencia() + enemigo.getExperienciaDada());
5     while (personajeActivo.getExperiencia() >= 100) {
6         personajeActivo.setExperiencia(personajeActivo.getExperiencia() - 100);
7         personajeActivo.setNivel(personajeActivo.getNivel() + 1);
8         if (personajeActivo.getExperiencia() < 100) {
9             break;
10        }
11    }
12 }
```

- Sistema de cajas de botín, proporcionadas por los jefes. Te pueden aportar cualquier objeto de forma aleatoria.

```

● ● ●
1  if (enemigo instanceof EnemigoMarinoJefe || enemigo instanceof EnemigoTerrestreJefe) {
2      personajeActivo.setBarrilesDisponibles(personajeActivo.getBarrilesDisponibles() + 1);
3

```

4. Flujo del Juego:

- Navegación mediante menús de forma íntegra en terminal.
- Progresión de la historia.
- Encuentros de combates infinitos de forma aleatoria tras la finalización del modo historia.
- Sistema de tienda.
- Sistema de casino.
- Sistema de ayudas.

Sistema de Personajes:

1. Atributos:

- Vida (*vida*).
- Daños, con diferenciación entre daños mágicos y físicos con sus respectivos ataques.
- Resistencias, al igual que los daños, existe una diferenciación entre mágica y física.
- Velocidad, importante para el flujo del combate (orden de los turnos).
- Sistema de niveles, para aumentar las estadísticas del personaje o enemigo según el mismo.

2. Métodos de Combate:

- Tres ataques diferentes por personaje, con diferenciación entre ataques mágicos, físicos y de estado.
- Efectos de estado (sangrado, somnolencia), que afectan en la batalla.
- Uso de objetos dentro de combate, con actualización de las estadísticas a tiempo real.

3. Objetos:

- Pociones de curación.
- Armas, para aumentar ambos tipos de daño.
- Armaduras, para aumentar ambos tipos de resistencias.
- Objetos especiales, que proporcionan distintas habilidades para ayudarte en la batalla.

Características Adicionales:

1. Sistema de Tienda:

- Compra de objetos.
- Sistema de moneda y economía.
- Sistema de casino en el que apostar monedas.
- Sistema de cajas de botín para conseguir objetos y monedas.

2. Elementos Narrativos:

- Un modo con historia progresiva.
- Sistema de mundos y niveles.
- Batallas contra jefes.
- Diferentes entornos y regiones.

3. Niveles de Dificultad:

- Normal (*Grumete*).
- Extremo (*Capitán*).

FUNCIONAMIENTO DEL JUEGO

Antes de comenzar, debes elegir tu nivel de dificultad: Normal, donde jugarás como un grumete, que posee 3 diferentes roles y tendrás hasta 3 oportunidades para completar el juego, o Extremo, donde serás un capitán enfrentando un mayor desafío y no siendo capaz de continuar el juego tras perder una batalla.

Durante los combates, podrás realizar diferentes acciones estratégicas: atacar utilizando tres ataques distintos, usar objetos de tu inventario para curarte o mejorar tus estadísticas, consultar tu estado y estadísticas para evaluar tu situación o intentar huir si el combate se vuelve demasiado difícil (Si tienes suerte).

Fuera de los combates, tendrás la posibilidad de avanzar a la siguiente zona o mundo, gestionar tu inventario para equiparte mejor, visitar la tienda para comprar objetos y mejoras o acceder al menú principal para otras opciones, como consultar tus ataques, estadísticas o nivel, además de echarte unas partiditas al casino o abrir unos cuantos barriles para conseguir un botín misterioso.

El juego posee un modo historia, en el que irás avanzando en la piel de un pirata en busca del mayor tesoro nunca antes encontrado. Al finalizar este modo, no te desanimes, aún queda más Argh! para tí; puedes disfrutar del modo de combate infinito, en el que te enfrentarás a enemigos y jefes de forma aleatoria que se ajustarán a tu nivel para un desafío mayor; o si lo tuyo no son los combates, aún tendrás la posibilidad de seguir disfrutando el casino, incluso con más dinero que nunca!

¡Embárcate en esta aventura con tus compañeros piratas y domina los mares! **ARGH!**

