PRÁCTICA EN GRUPO - RPG POR TURNOS SIMPLIFICADO

"Aventura en la Mazmorra"

1º DAW – Programación

Descripción del Proyecto

Vais a crear un juego de rol por turnos simplificado donde un equipo de héroes se enfrenta a enemigos en diferentes salas de una mazmorra. El objetivo es derrotar a todos los enemigos y sobrevivir.

Esta es vuestra oportunidad de crear un JUEGO REAL. ¡Poneos creativos!

Mecánica Básica del Juego

1. El jugador comienza con un **equipo de 3 héroes** (puede elegirlos)
2. Hay **5 salas** en la mazmorra, cada una con enemigos
3. En cada sala se produce un **combate por turnos**:
    - Los héroes atacan primero (en orden)
    - Luego atacan los enemigos
    - Se repite hasta que todos los enemigos o todos los héroes mueran
4. **Entre salas**, los héroes pueden:
    - Usar pociones para curarse
    - Ver su inventario
    - Descansar (recuperan algo de vida)
5. Si todos los héroes mueren → **GAME OVER**
6. Si completan las 5 salas → **¡VICTORIA!**

Estructura Simplificada de Clases

1. Enums (0.5 puntos)

```
enum TipoHeroe { GUERRERO, MAGO, ARQUERO }
enum TipoEnemigo { GOBLIN, ORCO, DRAGON }
enum TipoItem { POCION_PEQUENA, POCION_GRANDE, ELIXIR }
```

2. Clase Personaje (abstracta) (1 punto)

Atributos comunes:

- `String nombre`
- `int puntosVidaMax`
- `int puntosVidaActual`
- `int ataque`
- `int defensa`
- `boolean vivo` (true si puntosVidaActual > 0)

Métodos:

- `void atacar(Personaje objetivo)` : calcula daño y lo aplica al objetivo
  - Daño = ataque del atacante - defensa del objetivo (mínimo 1 de daño)
- `void recibirDanio(int danio)` : resta vida, si llega a 0 o menos, marca vivo = false
- `void curar(int cantidad)` : suma vida sin pasar del máximo
- `boolean estaVivo()` : devuelve si vivo == true
- Getters, setters, toString

Método abstracto:

- `void usarHabilidadEspecial(Personaje objetivo)` : cada tipo de personaje tendrá su habilidad

3. Clase Heroe (hereda de Personaje) (1.5 puntos)

Atributos adicionales:

- `TipoHeroe tipo`
- `int nivel` (empieza en 1)
- `int experiencia` (empieza en 0)
- `ArrayList<Item> inventario`

Constructor:

- Según el TipoHeroe, establece stats diferentes:
  - **GUERRERO**: Vida alta (100), Ataque medio (20), Defensa alta (15)
  - **MAGO**: Vida baja (60), Ataque alto (30), Defensa baja (5)
  - **ARQUERO**: Vida media (80), Ataque medio (25), Defensa media (10)

Métodos:

- `void usarHabilidadEspecial(Personaje objetivo)` :
  - **GUERRERO**: "Golpe Poderoso" - hace el doble de daño
  - **MAGO**: "Bola de Fuego" - hace daño a todos los enemigos (se le pasa una lista)
  - **ARQUERO**: "Disparo Preciso" - ignora la defensa del enemigo
- `void ganarExperiencia(int exp)` : suma experiencia, si llega a 100 sube de nivel
- `void subirNivel()` : aumenta stats (+20 vida, +5 ataque, +3 defensa), reinicia exp a 0
- `void usarItem(Item item)` : usa una poción del inventario

4. Clase Enemigo (hereda de Personaje) (1 punto)

Atributos adicionales:

- `TipoEnemigo tipo`
- `int expOtorgada` (experiencia que da al morir)

Constructor:

- Según el TipoEnemigo, establece stats diferentes:
  - **GOBLIN**: Vida baja (30), Ataque bajo (8), Defensa baja (3), Exp: 20
  - **ORCO**: Vida media (60), Ataque medio (15), Defensa media (8), Exp: 40
  - **DRAGON**: Vida alta (150), Ataque alto (25), Defensa alta (12), Exp: 100

Métodos:

- `void usarHabilidadEspecial(Personaje objetivo)` :
  - **GOBLIN**: "Golpe Rápido" - ataca dos veces seguidas con daño reducido
  - **ORCO**: "Grito de Guerra" - aumenta su ataque temporalmente
  - **DRAGON**: "Aliento de Fuego" - daña a todos los héroes

5. Clase Item (1 punto)

Atributos:

- `String nombre`
- `TipoItem tipo`
- `int valorCuracion`

Constructor:

- Según el TipoItem:
  - **POCION_PEQUENA**: cura 30 puntos
  - **POCION_GRANDE**: cura 60 puntos
  - **ELIXIR**: cura completamente

Métodos:

- `void usar(Heroe heroe)` : aplica la curación al héroe
- toString, getters

6. Clase Sala (1.5 puntos)

Representa una sala de la mazmorra con enemigos.

Atributos:

- `int numeroSala`
- `ArrayList<Enemigo> enemigos`
- `boolean completada`

Constructor:

- Recibe el número de sala
- Genera enemigos aleatorios según la dificultad:
  - Salas 1-2: 2-3 Goblins
  - Salas 3-4: 1-2 Orcos + 1 Goblin
  - Sala 5 (JEFE): 1 Dragón + 2 Orcos

Métodos:

- `void generarEnemigos()` : crea los enemigos de la sala
- `boolean todosEnemigosmuertos()` : verifica si no quedan enemigos vivos
- `ArrayList<Enemigo> getEnemigosVivos()` : devuelve solo los enemigos con vida
- toString que muestre los enemigos

7. Clase Combate (2 puntos)

Gestiona la lógica de un combate en una sala.

Atributos:

- `ArrayList<Heroe> heroes`
- `Sala sala`
- `int turno`

Métodos:

- `void iniciarCombate()` : bucle principal del combate
  - Mientras haya héroes y enemigos vivos:
    1. Mostrar estado (vida de todos)
    2. Turno de héroes (cada héroe ataca)
    3. Turno de enemigos (cada enemigo ataca)

4. Incrementar contador de turnos

- `void turnoHeroes()` : cada héroe vivo elige objetivo y ataca
  - **SIMPLIFICACIÓN**: Los héroes atacan al primer enemigo vivo automáticamente
  - O puedes pedir al jugador que elija objetivo
- `void turnoEnemigos()` : cada enemigo vivo ataca a un héroe aleatorio
- `void mostrarEstadoCombate()` : muestra vida de héroes y enemigos
- `boolean combateTerminado()` : true si todos de un bando están muertos
- `void distribuirRecompensas()` : si ganan, reparte exp entre héroes vivos

8. Clase Juego (2 puntos)

Clase principal que gestiona todo el juego.

Atributos:

- `ArrayList<Heroe> equipo` (los 3 héroes del jugador)
- `ArrayList<Sala> salas` (las 5 salas de la mazmorra)
- `int salaActual` (0-4)
- `boolean juegoTerminado`

Métodos:

- `void inicializarJuego()` :
  - Crear las 5 salas
  - Dejar que el jugador elija sus 3 héroes (o crear equipo predefinido)
  - Darles algunas pociones iniciales
- `void jugar()` : bucle principal del juego
  - Mientras no esté terminado:
    1. Mostrar menú entre salas (si no es la primera)
    2. Entrar en la sala actual
    3. Iniciar combate
    4. Verificar resultado (victoria/derrota)
    5. Avanzar a siguiente sala
- `void menuEntreSalas()` :

    - Mostrar opciones: Ver equipo / Usar pociones / Descansar / Continuar
- `void verificarEstadoJuego()` :
    - Si todos los héroes muertos → GAME OVER
    - Si completó sala 5 → VICTORIA
- `void mostrarResultadoFinal()` : pantalla de victoria o derrota

9. Clase Principal (0.5 puntos)

Método main:

```
public static void main(String[] args) {
    Juego juego = new Juego();
    juego.inicializarJuego();
    juego.jugar();
}
```

Simplificaciones Propuestas

Para hacerlo más manejable en grupo:

1. **Combate automático para héroes**: Los héroes atacan al primer enemigo vivo automáticamente
    - Opcional: Permitir elegir objetivo manual para más estrategia
2. **Habilidades especiales automáticas**: Se usan cada X turnos automáticamente
    - Opcional: Dar opción al jugador de cuándo usarlas
3. **Generación de salas fija**: Enemigos predefinidos por sala
    - Opcional: Generación aleatoria con rangos
4. **Inventario compartido**: Todo el equipo comparte las pociones
    - Opcional: Cada héroe tiene su propio inventario

5. **Sin interfaz gráfica**: Todo por consola con texto
    - Opcional: Añadir ASCII art para hacerlo más visual

Ideas para Ampliar (Opcional - Para Nota Extra)

Si el grupo es rápido y quiere más desafío:

- **Sistema de equipo**: Armas y armaduras que modifican stats
- **Más tipos de héroes**: Clérigo (cura), Ladrón (críticos), etc.
- **Tienda entre salas**: Comprar pociones y equipo con oro
- **Elementos**: Fuego, Agua, etc. con fortalezas/debilidades
- **Boss final más complejo**: Con múltiples fases
- **Sistema de guardado**: Guardar partida en archivo
- **Estadísticas**: Llevar registro de turnos, daño total, etc.

Distribución de Puntos

Componente	Puntos
Enums	0.5
Clase Personaje (abstracta)	1.0
Clase Heroe	1.5
Clase Enemigo	1.0
Clase Item	1.0
Clase Sala	1.5
Clase Combate	2.0
Clase Juego	2.0
Clase Principal	0.5
TOTAL	**10.0**

Extras opcionales: +1 punto por funcionalidades adicionales bien implementadas

👥 Recomendaciones para Trabajo en Grupo

Organización sugerida (grupo de 3-4 personas):

Persona 1 - "Sistema de Personajes":

- Clase Personaje (abstracta)
- Clase Heroe
- Clase Enemigo

Persona 2 - "Sistema de Combate":

- Clase Combate
- Clase Sala

Persona 3 - "Sistema de Juego":

- Clase Juego
- Clase Principal
- Menús e interfaz de usuario

Persona 4 - "Items y Testing" (si hay 4):

- Clase Item
- Enums
- Testing general
- Documentación

Workflow recomendado:

1. **Día 1**: Reunión inicial, diseñar estructura juntos, repartir tareas
2. **Día 2-3**: Cada uno trabaja en su parte, comunicación constante
3. **Día 4**: Primera integración, pruebas
4. **Día 5**: Ajustes, testing, documentación
5. **Día 6**: Presentación y entrega

Herramientas:

- **Git/GitHub**: Para trabajar juntos sin pisarse el código
- **Discord/WhatsApp**: Para comunicación
- **Google Docs**: Para documentación compartida

Ejemplo de Flujo de Juego

```
=== AVENTURA EN LA MAZMORRA ===

Elige tus 3 héroes:
1. Guerrero (Vida: 100, Ataque: 20, Defensa: 15)
2. Mago (Vida: 60, Ataque: 30, Defensa: 5)
3. Arquero (Vida: 80, Ataque: 25, Defensa: 10)

Héroe 1: 1 (Guerrero - Thorin)
Héroe 2: 3 (Arquero - Legolas)
Héroe 3: 2 (Mago - Gandalf)

¡Equipo formado!
Recibes: 3 Pociones Pequeñas

=== SALA 1 ===
¡Aparecen enemigos!
- Goblin (Vida: 30, Ataque: 8)
- Goblin (Vida: 30, Ataque: 8)
- Goblin (Vida: 30, Ataque: 8)

=== COMBATE ===

--- TURNO 1 ---
Thorin ataca a Goblin → Daño: 17 (Goblin: 13/30 HP)
Legolas ataca a Goblin → Daño: 22 (Goblin: -9/30 HP) ¡Goblin derrotado
Gandalf ataca a Goblin → Daño: 27 (Goblin: 3/30 HP)

Goblin ataca a Thorin → Daño: 1 (Thorin: 99/100 HP)
Goblin ataca a Legolas → Daño: 1 (Legolas: 79/80 HP)

--- TURNO 2 ---
...

¡VICTORIA!
Has ganado 60 puntos de experiencia
```

```
=== ENTRE SALAS ===
1. Ver estado del equipo
2. Usar poción
3. Descansar (recupera 20 HP a todos)
4. Continuar a siguiente sala

Elige: _
```

Criterios de Evaluación

- **Funcionalidad (40%)**: El juego funciona correctamente
- **POO (30%)**: Buen uso de herencia, polimorfismo, encapsulación
- **Código limpio (20%)**: Tabulación, nombres claros, comentarios
- **Creatividad (10%)**: Nombres, mensajes, detalles que hacen el juego más divertido