Ejercicio 2 — Herencia (Juego de Rol sencillo)

Objetivo

Vas a programar un pequeño simulador de combate entre personajes de rol utilizando herencia en Java. El ejercicio está pensado para practicar el diseño de clases, constructores, herencia, atributos comunes y polimorfismo.

Indicaciones

• La primera clase debe llamarse Personaje y debe ser abstracta.

• A partir de ella crearás varias subclases, por ejemplo: Guerrero, Mago, Arquero (puedes inventar más si lo deseas).

• Todos los personajes deben tener al menos un nombre y una cantidad de vida. Todos empiezan con 100 puntos de vida.

• Cada subclase tendrá sus propios valores de ataque y defensa (pueden ser constantes dentro de cada subclase).

La fórmula del ataque y cómo se calcula el daño

Cuando un personaje ataca a otro, el daño se calcula así:

daño = ataque_del_atacante – defensa_del_objetivo

Si el resultado es menor que 0, el daño se considera 0. En Java:

daño = Math.max(0, atacante.getAtaque() - objetivo.getDefensa());
Una vez calculado el daño, se resta de la vida del objetivo:

vida_objetivo = vida_objetivo – daño
La vida nunca puede bajar de 0. Si el resultado es negativo, debe fijarse a 0.

Casos especiales

• Si intentas atacar a un personaje que ya está muerto, debe mostrarse un error o lanzarse una excepción.

• Si un personaje muerto intenta atacar, también debe mostrarse un error.

• Al llegar a 0 de vida, un personaje se considera muerto y debe informarse en la consola.

Consejos

Pon en la clase abstracta Personaje lo que sea común (nombre, vida, atacar, mostrar estado). Deja en cada subclase lo que sea específico (valores de ataque y defensa, frases personalizadas al atacar). Diseña solo los métodos que veas necesarios para que el combate funcione y el resultado se entienda.

Ejemplo orientativo de clase Partida

```
public class Partida {
  public static void main(String[] args) {
    // Creamos dos personajes
    Personaje ana = new Guerrero("Ana");   // Ataque 25, Defensa 15
```

```
        Personaje luis = new Mago("Luis");     // Ataque 30, Defensa 5

        // Imprimir sus características iniciales
        System.out.println(ana);
        System.out.println(luis);

        // Simulamos varios ataques
        ana.atacar(luis);
        luis.atacar(ana);
        ana.atacar(luis);
        ana.atacar(luis);
        ana.atacar(luis);  // Aquí debería morir Luis

        // Intentar atacar de nuevo a un muerto
        ana.atacar(luis);

        // Intentar que un muerto ataque
        luis.atacar(ana);
    }
}
```

Ejemplo de salida en consola

[Guerrero] Ana — Vida: 100 | Ataque: 25 | Defensa: 15
[Mago] Luis — Vida: 100 | Ataque: 30 | Defensa: 5

[Guerrero] Ana ataca a [Mago] Luis
Daño: 20 (25 - 5) → Vida de Luis ahora: 80

[Mago] Luis ataca a [Guerrero] Ana
Daño: 15 (30 - 15) → Vida de Ana ahora: 85

[Guerrero] Ana ataca a [Mago] Luis
Daño: 20 (25 - 5) → Vida de Luis ahora: 60

[Guerrero] Ana ataca a [Mago] Luis
Daño: 20 (25 - 5) → Vida de Luis ahora: 40

[Guerrero] Ana ataca a [Mago] Luis
Daño: 20 (25 - 5) → Vida de Luis ahora: 20

[Guerrero] Ana ataca a [Mago] Luis
Daño: 20 (25 - 5) → Vida de Luis ahora: 0
[Mago] Luis ha muerto.

[Guerrero] Ana ataca a [Mago] Luis
Error: No puedes atacar a un personaje que ya está muerto.

[Mago] Luis intenta atacar a [Guerrero] Ana
Error: Un personaje muerto no puede atacar.

Entrega esperada

Una jerarquía de clases en Java con una clase abstracta Personaje y varias subclases (Guerrero, Mago, Arquero, …). Una clase con un main que simule una partida como la del ejemplo, mostrando ataques, reducción de vida, muertes y errores en consola.