

Sistema de Personajes para un Juego de Rol

En este ejercicio, deberás diseñar un sistema de personajes para un **juego de rol (RPG)** utilizando **Programación Orientada a Objetos (POO)** en **PHP**. Aplicarás conceptos clave como **herencia**, **interfaces**, **clases abstractas**, y **métodos y propiedades estáticas**. El sistema debe simular diferentes tipos de personajes, habilidades, y características especiales de cada uno.

Entorno de ejecución:

Descarga el proyecto:

<https://github.com/acrerosj/DSW25-Rolgame>

Este proyecto ya incluye el devcontainer con la instalación de composer y se autoejecuta “composer install” con lo que se instalan las librerías necesarias.

Crea la carpeta “public” y “src”.

Abre en el contenedor.

Pruebas (testing):

El proyecto incluye testing en PHPUnit para las clases que se piden en los apartados del 1 al 8.

Para su ejecución ya se ha instalado en el devcontainer una extensión para ver las pruebas:



Aquí podrás ir probando los tests de las clases.

Requisitos:

1. Clase abstracta **Personaje**:

- Crea una clase abstracta llamada **Personaje**, que represente a cualquier personaje genérico del juego.
- Propiedades:
 - **nombre** (cadena de texto)
 - **nivel** (entero)
 - **puntosDeVida** (entero)
- Métodos:
 - Un método abstracto llamado **atacar()**, que deberá ser implementado por las clases derivadas. Y devuelve un entero.

- Un método abstracto llamado `defender(daño)`, que deberá ser implementado por las clases derivadas. Tiene un parámetro “daño” que es un entero y devuelve un entero.
- Un método llamado `subirNivel()`, que aumenta el nivel del personaje en 1. No devuelve nada.
- Tiene un constructor que recibe los parámetros nombre, nivel y puntosDeVida en ese orden.

2. Clase **Guerrero**:

- Hereda de la clase `Personaje`.
- Propiedad adicional:
 - `fuerza` (entero)
- Métodos:
 - Implementa el método `atacar()`, que simule un ataque físico basado en la fuerza del guerrero. El resultado es el nivel * la fuerza.
 - Implementa el método `defender()`, que reduzca el daño recibido usando su fuerza. Al daño inicial se le resta la mitad de la fuerza del guerrero.

3. Clase **Mago**:

- Hereda de la clase `Personaje`.
- Propiedad adicional:
 - `mana` (entero)
- Métodos:
 - Implementa el método `atacar()`, que simule un ataque mágico utilizando puntos de mana. El ataque es la mitad del mana
 - Implementa el método `defender(dañoInicial)`, que use magia para reducir el daño o anularlo. Al daño inicial se le resta una quinta parte del mana.

4. Interfaz **Curable**:

- Crea una interfaz llamada `Curable`, que defina el método:
 - `curar()`: Implementado por las clases que lo usen, permitirá curar puntos de vida a un personaje.

5. Clase **Clerigo**:

- Hereda de la clase `Personaje` e implementa la interfaz `Curable`.
- Propiedades adicionales:
 - `poderCurativo` (entero)
- Métodos:
 - Implementa `atacar()` para realizar un ataque sagrado basado en su poder curativo.
 - Implementa el método `curar()`, que permita al clérigo restaurar puntos de vida a sí mismo o a otros personajes.
 - Implementa `defender(dañoInicial)` para reducir el daño recibido utilizando poder curativo dividido entre 2.

6. Clase **Partida**:

- Crea una clase llamada `Partida` que contenga los personajes y métodos.

- Implementa el método `agregarPersonaje(Personaje)` que añade el personaje a la partida.
- Implementa un método llamado `obtenerPersonajes()` que devuelva el array de personajes.
- Implementa el método `eliminaPersonaje(Personaje)` que elimina el personaje de la partida.
- Implementa un método llamado `obtenerPersonajesPorClase(class)` que devuelva el array de personajes que pertenecen a dicha clase..

7. Método estático **Lucha**:

- En la clase `Personaje`, implementa un método estático llamado `lucha(Personaje1, Personaje2)` que implementa una lucha entre los dos:
 - `Personaje1` ataca, `Personaje2` se defiende y se le restan los puntos de daño final.
 - Viceversa.

8. Clase **Partida**:

- Implementa el método `eliminaMuertos()` que elimina los personajes cuyo `puntosDeVida` sean igual o menor que 0.