Ejercicios

Leer atentamente y crear las clases, propiedades y métodos que se piden en cada ejercicio.

Ejercicio 1

Crear una clase Personaje

Propiedades

- private string $nombre;
- private int $ataque;
- private int $defensa;
- private int $vidas;

Métodos

public function __construct(string $nombre, int $ataque, int $defensa)

Recibe como parámetros nombre, ataque y defensa; y setea las propiedades con los valores que le corresponden. En el caso de las vidas setearlas en 3.

public function getNombre():string

Retornar el nombre del personaje.

public function getVidas():int

Retornar la cantidad de vidas del personaje.

public function getAtaqueRand():int

Retornar un número aleatorio (usar la función rand()) entre 0 y el ataque del personaje.

public function getDefensaRand():int

Retornar un número aleatorio (usar la función rand()) entre 0 y la defensa del personaje.

public function restarVida():void

Decrementar la cantidad de vidas del personaje en 1 (uno)

public function curar():void

Setear el valor de vidas a 3.

private function atacar(Personaje $enemigo):bool

Recibe como parámetro un objeto Personaje. Si el método getAtaqueRand() de **$this** devuelve un número mayor al que devuelve el método getDefensaRand() de **$enemigo**, entonces llamar al método **restarVida()** de **$enemigo** y retornar un valor true, caso contrario false.

private function defender(Personaje $enemigo):bool

Recibe como parámetro un objeto Personaje. Si el método getDefensaRand() de **$this** devuelve un número mayor o igual al que devuelve el método getAtaqueRand() de **$enemigo**, entonces retornar un valor true, caso contrario llamar al método **restarVida()** de **$this** y retornar false.

public function pelear(Personaje $enemigo):bool

Recibe como parámetro un objeto Personaje. Generar un valor binario, por ejemplo un rand() entre 1 y 2, y:

- Si el resultado es 1 (uno) entonces llamar al método atacar(Personaje $enemigo), retornar su resultado.
- Sino llamar al método defender(Personaje $enemigo), retornar su resultado.

Test:

```php
$superman = new Personaje('Superman', 98, 70);

$batman = new Personaje('Batman', 70, 89);


for($i = 0; $i < 3; $i++)

{

    $superman->pelear($batman);

}


if($superman->getVidas() > $batman->getVidas()){

    echo 'Ganó Superman';

}else if($superman->getVidas() < $batman->getVidas()){

    echo 'Ganó Batman';

}else{

    echo 'Empate';

}
```

Ejercicio 2

Crear una clase abstracta Vehículo con dos propiedades:

Propiedades

- private int $cant_ruedas;
- private int $cant_puertas;

Métodos

public function __construct(int $cant_ruedas, int $cant_puertas)

Recibe los valores de ambos parámetros y setea la cantidad de ruedas y puertas.

Además crear otras tres clases: Auto, Moto y Camión. Éstas deben heredar de la clase Vehículo, pero además deberán tener sus propios constructores que pisen el constructor de la clase padre en donde:

- Auto: tendrá un constructor en donde se debe pasar como parámetro la cantidad de puertas. La cantidad de ruedas debe setearse con el valor 4.
- Moto: no tendrá parámetros en el constructor. La cantidad de ruedas debe setearse en 2 y las puertas en 0 (cero)
- Camión: tendrá un constructor en donde se debe pasar como parámetro la cantidad de ruedas. La cantidad de puertas debe setearse con el valor 2.

Ejercicio 3

Crear una clase TestPersonajes. Incluir con require_once la clase Personaje que hicimos en el primer ejercicio.

Propiedades

Ninguna.

Métodos

public static function simularPelea(Personaje $personaje1, Personaje $personaje2):Personaje

Recibe como parámetros dos objetos de tipo Personaje. Entrar en un bucle en donde se ejecutará $personaje1->pelear($personaj2) hasta que uno de los dos se quede sin vida. Retornar al personaje ganador.

public static function simularMultPelea(Personaje $personaje1, Personaje $personaje2, int $intentos):array

Repite varias veces el método anterior (simularPelea(Personaje $personaje1, Personaje $personaje2)) Curar a los personajes antes de cada enfrentamiento.

Recibe como parámetros los mismos objetos de tipo Personaje y la cantidad de veces que se ejecutará lo mencionado en el párrafo anterior.

Al final deberá devolver un array asociativo con:

- La cantidad de peleas ganas de **$personaje1**

- La cantidad de peleas ganadas de **$personaje2**
- El porcentaje de victorias de **$personaje1**
- El porcentaje victorias de **$personaje2**

Test:

```php
$superman = new Personaje('Superman', 98, 70);

$chapulin = new Personaje('Chapulin colorado', 30, 25);



$resultado = TestPersonajes::simularMultPelea($superman, $chapulin, 300);

echo '<pre>';

var_dump($resultado);

exit;
```