Nombre: _____

Certamen I ELO-329 Diseño y Programación Orientada a Objetos

En este certamen usted no podrá hacer preguntas. Si algo no está claro, indíquelo en su respuesta, haga una suposición razonable y resuelva conforme a ella.

Segunda parte, **con apuntes**:

Segunda pregunta (35 pts.)

Se le ha encargado el desarrollo del juego "**Mago adventures**". Para ello se le entregan dos archivos disponibles en la página de Aula:

- Metodos.java el cual contiene una interfaz (interface) a ser implementada por los personajes del juego.
- Juego.java el cual contiene la lógica del juego.

A. (30 pts.) Descargue y utilice el código entregado y cree las siguientes clases:
   i) (10 pts.) Clase Personaje, la cual implementa la interfaz Metodos, manteniendo el método atacar() como método abstracto. Adicionalmente incluye el nombre y los puntos de vida del personaje como características.
   ii) (10 pts.) Clase Enemigo, la cual hereda de la clase Personaje. Esta clase posee adicionalmente el atributo fuerza, que representa la fuerza con que un enemigo golpea a un mago.
   iii) (10 pts.) Clase Mago, la cual hereda de la clase Personaje. Esta clase posee adicionalmente el atributo poder, el que representa el poder con que un mago ataca a sus enemigos. Además, cuenta con el método aumentarVida(), el cual aumenta la vida de dicho mago de 1 a 5 puntos de manera aleatoria.

B. (5 pts.) Modifique el archivo Juego.java y, a través de manejo de excepciones, evite que el programa falle ante el ingreso de una acción diferente de un número entero para un mago (por ejemplo, acción 'a' en lugar de 1 o 2). En caso de ocurrir una excepción, el programa continúa como si se hubiera ingresado la acción -1 (se pierde el turno).

Respuesta:

A)

```
// Personaje.java

// Definición correcta 3pts
public abstract class Personaje implements Metodos {
 private String nombre;
 protected int vida;

// Constructor 1pt
 public Personaje(String nombre, int vida){
 this.nombre = nombre;
 this.vida = vida;
```

Nombre: _____

```
 }

 // Getters 1pt
 public String getNombre(){
 return this.nombre;
 }

 public int getVida(){
 return this.vida;
 }

 // Definir atacar() como método abstracto 2pts
 public abstract void atacar(Personaje objetivo);

 // Métodos recibirDano() y estaVivo() 3pts
 public void recibirDano(int cantidad){
 this.vida -= cantidad;
 }

 public boolean estaVivo(){
 return vida>0;
 }
}


// Mago.java

import java.util.Random;

// Definición correcta 3pts
public class Mago extends Personaje{
 private int poder;

 // Constructor 2pts
 public Mago(String nombre, int vida, int poder){
 super(nombre, vida);
 this.poder = poder;
 }

 // Método atacar() 2pts
 public void atacar(Personaje objetivo){
 objetivo.recibirDano(poder);
 }

 // Metodo aumentarVida() 3pts
 public void aumentarVida(){
 Random random = new Random();
 super.vida += random.nextInt(5)+1;
 }
}


// Enemigo.java
```

Nombre: _____

```
// Definición correcta 6pts
public class Enemigo extends Personaje {
 private int fuerza;

 // Constructor 2pts
 public Enemigo(String nombre, int vida, int fuerza){
 super(nombre, vida);
 this.fuerza = fuerza;
 }

 // Constructor 2pts
 public void atacar(Personaje objetivo){
 objetivo.recibirDano(fuerza);
 }
}
```

B)

```
// Excepción

int accion;
 try{ // Captura excepción 2 pts
 accion = s.nextInt();

// Maneja excepción
 }catch(InputMismatchException e1){

// accion con valor -1
 accion = -1;// Retorna 1
 }
```