Práctica 2. Herencia y Polimorfismo

En esta práctica se introducen los conceptos de herencia y polimorfismo. La herencia permite que nuevas clases puedan aprovechar código ya implementado por clases anteriores. El polimorfismo, o dynamic binding, permite al sistema distinguir entre métodos que tienen el mismo nombre, pero distintas implementaciones, según la clase en la que esté implementado.

2.1. Primera Parte

En este ejercicio se va a implementar un juego basado en vidas en el que el usuario tenga que adivinar un número que conoce el programa. El código correspondiente a cada clase que se implemente deberá estar en ficheros separados con el mismo nombre que la clase.

Modificar la clase JuegoConVidas:

- Redefinir el atributo **record** para que se convierta en una variable no estática.
- Convertir el método **muestraVidasRestantes** en un método privado.
- Añadirle un método abstracto **juega** que tome como parámetro una cadena de caracteres (String) y que retorna un booleano.
- Ahora la clase **JuegoConVidas** pasa a ser una clase abstracta por lo que ya no se podrán crear instancias de ésta.

Crear la clase JuegoAdivinaNumero

- Deriva de la clase **JuegoConVidas** (Fig1).
- Tiene un constructor que toma dos parámetros de tipo entero. El primero es el número de vidas que, a su vez, se lo pasará al constructor de la clase base. El segundo parámetro representa el número a adivinar y esta acotado entre 0 y 10.
- Implementa un método booleano denominado **validaNumero** que toma como parámetro un número entero. Este método retorna true si el número introducido es un número entre 0 y 10. En caso contrario, muestra por consola un mensaje indicando que el número esta fuera de ese intervalo y retorna false.
- Implementa el método **juega**:
  - ✓ Convierte en un entero la cadena de caracteres (String) recibido como parámetro.
  - ✓ Valida si el número está comprendido entre 0 y 10. Para ello, usa la función **validaNumero** creada anteriormente. Si el número esta fuera de rango, **juega** retorna true. En caso de que se cumpla, se comprueba si el número recibido por parámetro coincide con el número a acertar:
    - Si son iguales: a) muestra por consola el mensaje: “Acertaste!!”, b) actualiza la variable record y c) retorna False.
    - Si es diferente, comprueba si el número a acertar es mayor o menor al que se pasó por parámetro mostrando por consola dicha información. A continuación, el método juega retorna el booleano obtenido de invocar el método **quitaVida** heredado.

Uso de la clase MyInput

Se utilizará la clase MyInput que aparece descrita en la práctica 0 para la entrada de datos por teclado. Esta clase deberá estar ubicada en otro paquete del mismo proyecto denominado "Entrada" que deberá ser importado por aquellas clases que lo utilicen.

Crear la clase principal

Contiene el método main donde:

1. Se crea una instancia de la nueva clase JuegoAdivinaNumero.
2. Se invoca el método reiniciaPartida.
3. Se define un bucle donde se solicita al usuario que introduzca un número entero de 0 a 10, y cuya salida viene gobernada por el valor de retorno del método juega.

2.2. Segunda Parte

A partir del juego anterior, se añadirán dos juegos más, uno donde se adivinen números pares y otro de números impares.

Crear la clase JuegoAdivinaNumeroPar

- Deriva de la clase JuegoAdivinaNumero (Fig1).
- Redefine el método **validaNumero**. Este método vuelve a validar si el número está entre 0 y 10 y a continuación comprueba si el número es par devolviendo true. Si el número es impar, muestra un mensaje de error por pantalla y devuelve false.

Crear la clase JuegoAdivinaNumeroImpar

- Deriva de la clase JuegoAdivinaNumero (Fig1).
- Redefine el método **validaNumero.** Este método vuelve a validar si el número está entre 0 y 10 y a continuación comprueba si el número es impar devolviendo true. Si el número es par, muestra un mensaje de error por pantalla y devuelve false.

Modificar la clase principal

- Crea un método estático denominado **jugar** que no retorna nada y recibe como parámetro una referencia de tipo JuegoConVidas. Dentro de este método se invoca el método **reinciaPartida** a través de la referencia del tipo JuegoConVida y a continuación, se define un bucle donde se solicita al usuario que introduzca un número entero de 0 a 10, y cuya salida viene gobernada por el valor de retorno del método **juega**.
- El método main crea una instancia de cada uno de los tres juegos creados: JuegoAdivinaNumero, JuegoAdivinaNumeroPar y JuegoAdivinaNumeroImpar. El número de vidas de cada juego será 5 y como número a adivinar un número cualquiera, otro par y otro impar respectivamente, todos comprendidos entre el 0 y el 10.
- Invocar el método jugar tres veces, pasándole como parámetro cada una de las instancias creadas.
- Ejecutar el programa y comprobar el funcionamiento de cada juego.

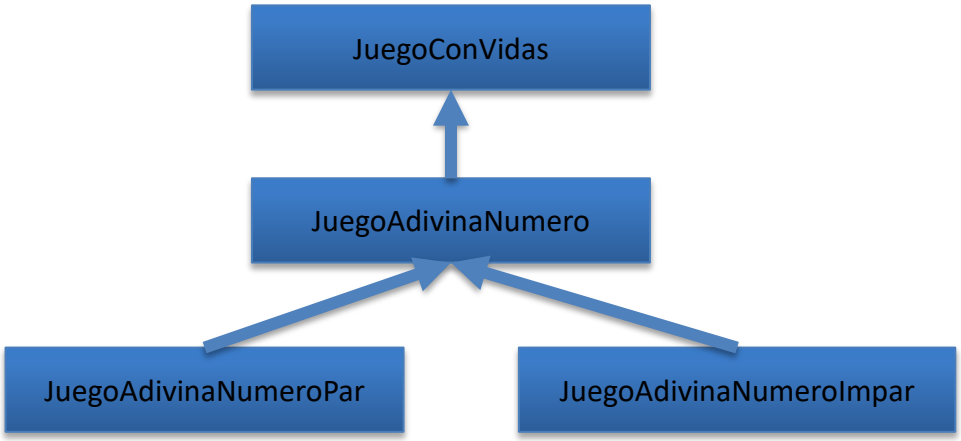

Fig1. Jerarquía de clases resultante