Práctica 5. Validación, Serialización y Documentación JavaDoc.

En esta última práctica guiada se abordará la validación del código, utilizando en algunos casos el manejo de excepciones, y se almacenará en memoria secundaria el objeto donde se manejan todos los recursos del sistema (serialización).

5.1. Primera parte. Manejo de excepciones

5.1.1. Validación de la selección de opciones

Modificar el método **readInt** para que propague una excepción del tipo **NumberFormatException,** en el caso de que la cadena de caracteres (String) que se introduzca no se corresponda con un número entero, de tal forma que si se lanza esta excepción se le indique al usuario que debe introducir un número válido y se vuelva a leer la entrada. Por esta razón, allí donde se considere oportuno, se deberá tratar la excepción mediante un bloque try-catch.

5.1.2. Validación del formato de la respuesta del usuario.

Añadir un nuevo método booleano denominado **validaFormato** al interfaz Jugable y que por tanto deberán implementar todas las clases que implementen dicho interfaz. Este método recibe como parámetro una cadena de caracteres (String) que representa la entrada del usuario como respuesta al acertijo. Si el formato es correcto devuelve true en caso contrario devuelve false. En el caso de la clase JuegoAdivinaNumero el método intentará convertir (parsear) la cadena (String) introducida y comprobar si es un número entero. Si no lo es, saltará una excepción y el método retornará el valor false. En el caso de la clase JuegoAhorcadoInglés, el método **validaFormato** retornará true si la cadena (String) representa una única letra minúscula y false en caso contrario. La introducción del método **validaFormato** supondrá la modificación del método **jugar** de la clase Menu. Ahora la solicitud de una respuesta al usuario se deberá incluir dentro de un bucle, que se anidará en el anterior, y que repetirá la pregunta al usuario hasta que este introduzca una respuesta con el formato correcto.

5.1.3 Validación de la entrada del diccionario mediante excepciones de usuario.

El constructor de las clases no puede devolver ningún código de retorno para indicar si ha funcionado bien o no. Una opción es que genere una excepción en caso de fallo.

Crear la Clase JuegoAhorcadoInglesExcepcion

- Estará en el paquete excepciones.
- Extiende la clase **Exception**.
- Su constructor toma como parámetro una cadena de caracteres con la descripción del motivo de la excepción.

Crear un método denominado chequeaPalabra dentro de la clase JuegoAhorcadoIngles.

- Este método, que será privado, recibe por parámetro la palabra seleccionada (String) y comprueba si contiene algún carácter que no sea una letra

minúscula. Si es así, arroja una excepción de tipo JuegoAhorcadoInglesExcepcion. En caso contrario no hace nada.

Modificar el método reiniciaPartida de la clase JuegoAhorcadoIngles

- Este método debe ser modificado para que invoque el método **chequeaPalabra** y trate la posible excepción que lanza dicho método. En este sentido, se deberá tratar la excepción de tal modo que, si la palabra es errónea el código vuelva a extraer una palabra aleatoria del diccionario. Esta operación se deberá repetir hasta obtener una palabra válida.

Otras consideraciones

- Probar a compilar y ejecutar el programa empleando como palabras a adivinar algunas con número y otras sin ningún número.

5.2. Segunda parte. Serialización.

- Modificar la clase MyInput para que incluya los dos siguientes métodos:

```
public static <A> void serialize(A a, String nombreFichero) {
        System.out.println("Serializando...");
        try {
                FileOutputStream fos = new FileOutputStream(nombreFichero) ;
                ObjectOutputStream oos = new ObjectOutputStream(fos) ;
                oos.writeObject(a) ;
        } catch (Exception e) {
                System.err.println("Problem: "+e) ;
        }
  }

  public static <A> A deserialize(String nombreFichero) {
        System.out.println("DeSerializing...");
        try {
                FileInputStream fis = new FileInputStream(nombreFichero) ;
                ObjectInputStream iis = new ObjectInputStream(fis) ;
                return (A) iis.readObject() ;
        } catch (Exception e) {
                System.err.println("Problem: "+e) ;
        }
        return null;
    }
```

- Modificar las clases JuegoConVidas y JuegosReunidos para que implementen el interfaz Serializable.
- Modificar la clase principal con el siguiente método **main**:

```
public static void main(String[] args) {
      JuegosReunidos jr=MyInput.deserialize("juegos.dat");
      if (jr==null)
            jr= new JuegosReunidosExtendidos(MyInput.leeFichero("diccionario.txt"));
      Menu mp=new MenuExtendido(jr);
      mp.ejecuta();
      MyInput.serialize(jr,"juegos.dat");
   }
```