Programación II

Ejercicio Evaluable 2: Mi Bonoloto

Dpto. LSIIS. Unidad de Programación

Objetivo: El objetivo de este ejercicio entregable es la familiarización del alumno con la herencia y las excepciones de Java.

Entrega: El ejercicio se entregará a través del servidor web de la asignatura:

http://triqui2.fi.upm.es/entrega/

El ejercicio entregado debe compilar en la versión 1.8 del J2SE de Oracle/Sun. Por el hecho de que el ejercicio sea admitido, eso no implicará que el ejercicio esté aprobado.

Grupos: El ejercicio se podrá realizar en grupos de dos alumnos o de forma individual. Se proporcionará al alumno el código que se necesita importar para que tras la entrega se pueda validar la corrección de los datos del autor o autores. Al comienzo del código realizado se deberán completar los datos del alumno o alumnos según este formato:

```
import anotacion.*;
@Programacion2 (
        nombreAutor1 = "nombre",            // (del alumno 1)
        apellidoAutor1 = "apellido1 apellido2",  // (del alumno 1)
        emailUPMAutor1 = "usr@alumnos.upm.es",   // (del alumno 1)
        nombreAutor2 = "nombre",            // (del alumno 2 si lo hay)
        apellidoAutor2 = "apellido1 apellido2",  // (del alumno 2 si lo hay)
        emailUPMAutor2 = "usr@alumnos.upm.es"    // (del alumno 2 si lo hay)
)
public class Loto {

        // .... rellenar aquí el código solicitado en el ejercicio

} // de clase Loto
```

Evaluación: En el momento de entregar el ejercicio en la web de la asignatura, será sometido a una serie de pruebas. Para que la entrega sea admitida, tendrá que superar las pruebas obligatorias. Dependiendo de las pruebas que consiga superar el ejercicio, se obtendrá una nota, que se mostrará al alumno tras realizar la entrega.

El baremo concreto para la evaluación del ejercicio figura al final del enunciado.

Problema a Resolver

Se necesita desarrollar una aplicación de juegos de azar de tipo Bonoloto. En estos juegos se trata de hacer una apuesta que consiste en la elección de una combinación de números en un rango de valores posibles con la finalidad de acertar una combinación de números llamada combinación ganadora. Según se consiga acertar más o menos números, así será el premio ganado.

Se define una *Loto* como un juego de azar que consiste en acertar *n* números diferentes en el rango [*min*, *max*]. Se supone que esto se va a cumplir para todas las lotos. A partir de esta definición general podemos definir juegos de azar más específicos particularizando una Loto y añadiendo más atributos y métodos. Su especificación e implementación se dan en el archivo *Loto.java*.

En los juegos de azar existentes actualmente, como la *Lotería Primitiva*, la *Bonoloto* o el *EuroJackpot*, además de los *n* números hay que acertar uno o más números Extra que tienen diferentes significados, rangos de valores y nombres según el juego de que se trate: *complementario*, *sol* o *reintegro*. Definimos un *Extra* como el *número* (int) y los límites *min* (int) y *max* (int) que marcan el rango en el que se mueve [*min*, *max*]. La especificación e implementación de esta clase *Extra* se dan en el archivo *Extra.java*.

Se define la *Primitiva* como un juego de azar que consiste en acertar 6 números diferentes entre 1 y 49, más dos Extras que son el *complementario* (un número entre 1 y 49) y el *reintegro* (otro número entre 0 y 9). Definimos esta clase como subclase de la clase *Loto*. Su especificación e implementación se dan en el archivo *Primitiva.java*.

Se define un *Sorteo* como una tupla formada por una *Loto* y la *Fecha* en el que tuvo lugar. Su especificación e implementación se dan en el archivo *Sorteo.java*. Asimismo se da la especificación e implementación de la clase *Fecha* en el archivo *Fecha.java.*

Se tiene un registro del *Histórico* de sorteos como una lista de Sorteos de los diferentes tipos de Loto. La especificación e implementación de la clase *Historico* se facilita en el archivo *Historico.java.*

Se pide:

1. Implementa el constructor de la clase *Loto,* encargado de lanzar las excepciones *BadSize* y *FueraDeRango.*
   - Si la longitud del array *numeros* es distinta de *numerosSorteo*, lanzará la excepción *BadSize.*

- Si no están todos los números del array en el rango [*min*, *max*], lanzará la excepción *FueraDeRango. S*e necesita desarrollar la función *todosEnRango,* que también se pide, y cuya cabecera se facilita en la clase Loto.
- En caso contrario, construirá la *Loto*.

2. Implementa el manejador de excepciones de Primitiva *consPrimitiva* en la clase Primitiva. Esta operación crea una Primitiva llamando al constructor de esta clase y manejando las excepciones *BadSize* y *FueraDeRango.* Se recomienda utilizar como referencia el método *consLoto* de la clase Loto.
3. Implementa la función *frecuenciasReintegro* en la clase Historico.

Para probar los métodos añadidos a las clases Loto, Primitica e Historico se proporcionan los JUnitTests LotoJUnitTest, PrimitivaJUnitTest y HistoricoJUnitTest, respectivamente.

Nota: Lee con atención la especificación de cada operación para entender lo que tiene que hacer.

Diagramas

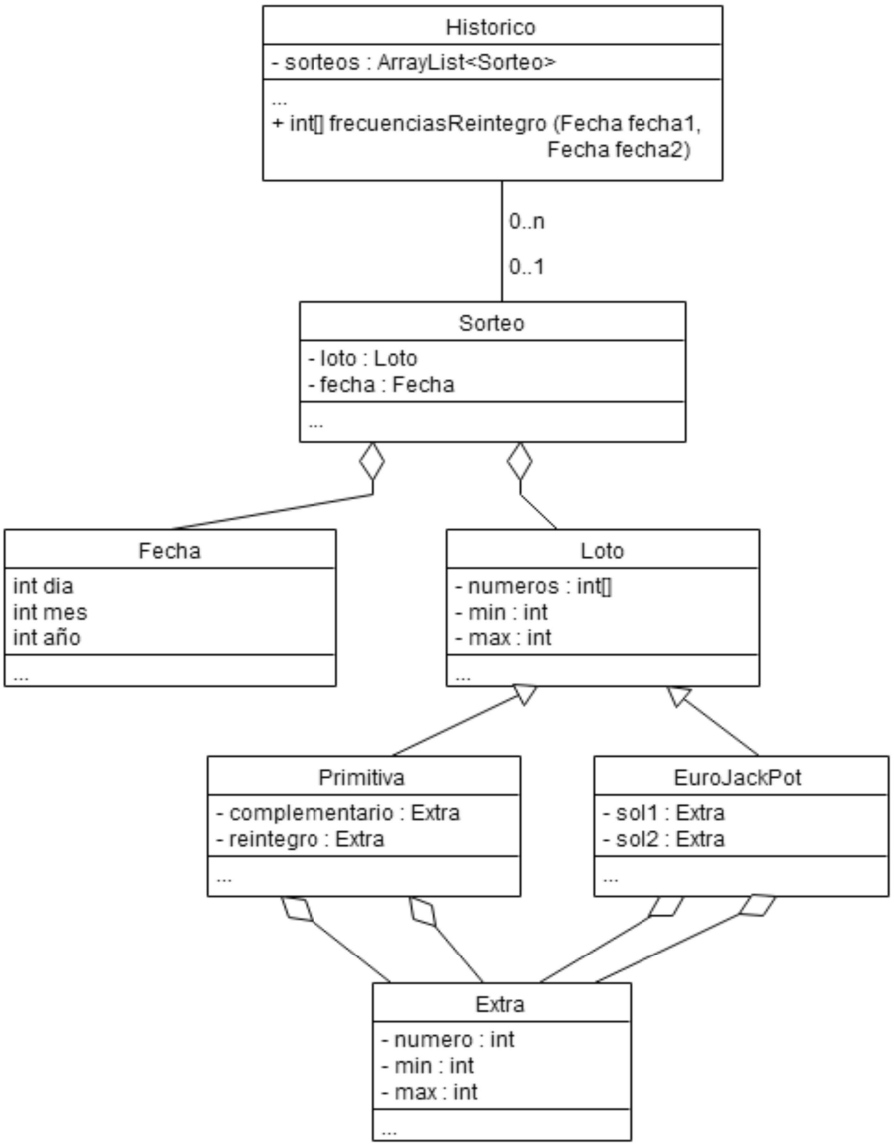

Figura 1: Diagrama UML

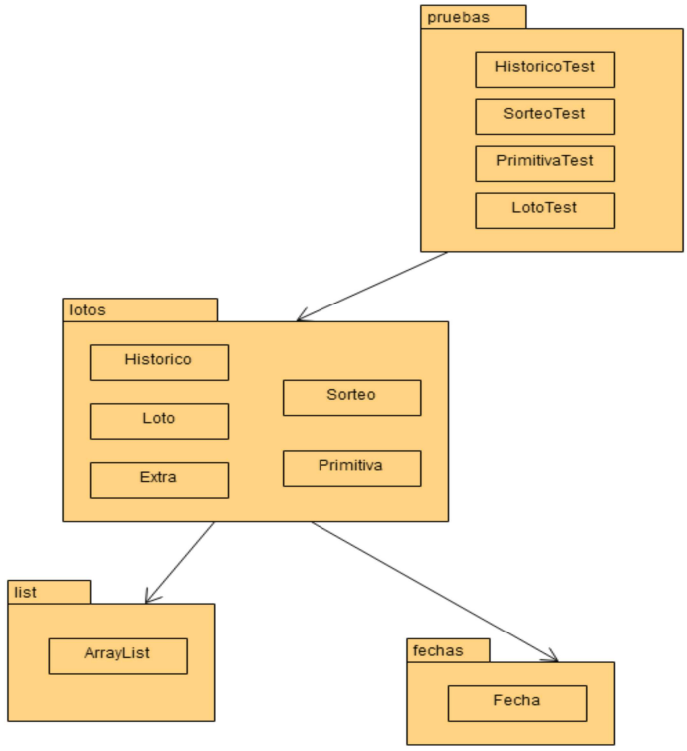

Figura 2: Diagrama de paquetes

Valoración

La **valoración específica** de este entregable será:

- Apartado 1: 4 puntos.
- Apartado 2: 2 puntos.
- Apartado 3: 4 puntos.

Se descontarán puntos según la **valoración general** a todos los trabajos prácticos de la asignatura.