Programación Orientada a Objetos. Práctica Adicional 3.x1

Tema 3. Tratamiento de Excepciones

Ejercicio 1.

1. Diseña la clase `UrnaExc`, en el paquete `prVotacion`, que represente objetos *urnas* que sólo permitan depositar votos (*afirmativos* o *negativos*) mientras la urna este *abierta*, y sólo permitan consultar el resultado de la votación (si la cuenta de votos afirmativos es mayor o igual que la cuenta de votos negativos) cuando la urna esté *cerrada*.

   La clase ofrecerá los siguientes constructores y métodos públicos:

   - `UrnaExc()`

     Construye un objeto urna con las cuentas de votos afirmativos y negativos a cero, y el estado de votación *abierto*.

   - `boolean estaAbierta()`

     Devuelve el estado de votación de la urna, `true` si la votación está abierta, y `false` en caso contrario.

   - `void cerrarVotacion()`

     Cierra la urna, poniendo el estado de votación a `false`.

   - `void votar(boolean)`

     Si la urna está cerrada, entonces lanza una `RuntimeException`. Sin embargo, si la urna está abierta, entonces aumenta la cuenta de votos afirmativos o negativos según el valor recibido como parámetro.

   - `boolean resultado()`

     Si la urna está abierta, entonces lanza una `RuntimeException`. Sin embargo, si la urna está cerrada, entonces devuelve el resultado de la votación (si la cuenta de votos afirmativos es mayor o igual que la cuenta de votos negativos).

   - `String toString()` *`// @Redefinición`*

     Devuelve la representación textual del objeto actual, según el formato de los siguientes ejemplos:

     - Si la urna está abierta: `(Urna-Abierta)`
     - Si la urna está cerrada, *votos positivos, negativos y resultado*: `(3, 2, true)`

1. Diseña la clase distinguida `PruebaUrnaExc` (principal) que permita probar el funcionamiento de la clase `UrnaExc`. Para ello, debe crear un objeto urna, y procesar los argumentos recibidos por el programa principal. Los comandos recibidos por el programa principal podrán ser los siguientes: **si**, **no**, **cerrar** y **resultado**. La ejecución errónea del algún comando, mostrará un mensaje de error, pero continuará el procesamiento hasta el final de los argumentos recibidos.

   Si se solicita algún otro comando no reconocido, entonces mostrará un mensaje de error, y se terminará el programa.

   Finalmente, bajo cualquier circunstancia, tanto si se han procesado todos los comandos de forma correcta o no, como si se ha abortado la ejecución del programa por comandos no reconocidos, mostrará el resultado final de la votación y las estadísticas asociadas.

   Véase la *Guía de Eclipse* para saber como pasar argumentos al programa principal.

Ejercicio 2.

1. Defina la **excepción comprobada** `UrnaException`, en el paquete `prVotacion`, que sera utilizada para notificar los errores que se produzcan en la manipulación de la urna.
2. Modifique la clase `UrnaExc` del ejercicio anterior para que lance la **excepción comprobada** `UrnaException` cuando se produzca algún error en la manipulación de la urna.
3. Modifique la clase `PruebaUrnaExc` del ejercicio anterior para que gestione adecuadamente la **excepción comprobada** `UrnaException` cuando se produzca algún error en la manipulación de la urna.