Nombre: ..... Fecha: / 01 / 2009

Grupo: 1 ☐ 2 ☐ 3 ☐ 4 ☐

PRÁCTICA 11
DECLARACIÓN, CONSTRUCCIÓN, Y GESTIÓN DE EXCEPCIONES EN JAVA. EJEMPLOS DE USO

En esta práctica vamos a introducir el trabajo con excepciones. Para ello diseñaremos algunos ejemplos de situaciones anómalas en programación que intentaremos gestionar por medio de algunas de las excepciones de la librería de Java.

Parte 1. Capturando excepciones propias de la librería de Java.

1. Crea un fichero "Principal_prac11.java". Crea un método "main" que cree un objeto de la clase "String" de nombre cadena. Inícialo con el valor "null". Muestra por pantalla el valor de la invocación "cadena.length()".

¿Qué ha sucedido al compilar el programa?

¿Qué ha sucedido al ejecutar el programa?

2. Vamos a intentar ahora capturar la excepción que ha tenido lugar en el anterior ejercicio. Para ello debes encerrar el comando que ha producido la excepción en un bloque:

```
try {
        ...
}
```

Asimismo, para poder capturar la excepción, deberás crear a continuación del bloque "try {...}" un bloque como el siguiente en el que decidas lo que vas a hacer con la misma:

```
 catch (TipoDeLaExcepcion nombre) {
        ...
}
```

Haz que el bloque "catch (TipoDeLaExcepcion nombre){...}" muestre un mensaje por pantalla con la siguiente cadena:

"Se ha intentado llamar a un metodo sobre un objeto que contenia una referencia a null"

A continuación muestra también la siguiente cadena:

System.out.println (nombre.toString());

3. Vuelve a la línea donde has inicializado la cadena de caracteres como "nulll" e inicialízala con un nuevo valor, "ejemplo". ¿Qué sucede ahora al compilar y ejecutar tu código? Como puedes observar, la presencia de excepciones en un programa hace que el mismo pueda tomar diferentes flujos de ejecución dependiendo de que en el código se produzcan ciertas excepciones o no.

4. Dentro del bloque "try {...}" que tenías programado, y después de haber mostrado por pantalla la longitud de la cadena "cadena", realiza la siguiente operación.

double x = Double.parseDouble (cadena);

Puedes encontrar más información sobre la función de este método en http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Double.html.

¿Qué ha sucedido al compilar dicho código?

¿Qué ha sucedido al ejecutar dicho código?

5. Crea un nuevo bloque "catch (TipoDeLaExcepcion identificador){...}" a continuación del "catch (...){...}" que ya habías programado que se encargue de capturar la excepción que has obtenido en el ejercicio anterior.

En el bloque "catch (...){...}" introduce tres comandos. Uno primero que muestre por pantalla el mensaje:

"Se ha producido un error en la conversion de la cadena de caracteres en un numeral"

A continuación realiza las invocaciones:

```
System.out.println (identificador.getMessage());
identificador.printStackTrace();
```

Puedes comprobar la función de los métodos "getMessage(): String" y "printStackTrace(): void" respectivamente en http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Throwable.html#getMessage() y http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Throwable.html#printStackTrace().

6. De nuevo, cambia el valor de "cadena" en su inicialización y haz que el mismo sea "8.751". Asimismo, añade una nueva orden dentro del bloque "try {...}" que contenga la siguiente orden:

```
System.out.println ("El valor del double es " + x);
```

¿Qué ha sucedido con la excepción que encontrábamos antes?

7. Vamos a ver ahora cómo podemos escribir el valor de la cadena anterior en un fichero. Dentro del bloque "try {...}" añade un nuevo comando que realice la siguiente orden (para ello debes incluir las librerías del sistema "java.io.*"):

```
File fichero = new File("entrada_salida_practica11.txt");
```

Puedes encontrar más información sobre la librería "File" en http://java.sun.com/j2se/1.5.0/docs/api/java/io/File.html.

A continuación, utiliza el método "createNewFile(): boolean" de la clase "File" para que el fichero se cree también en tu sistema:

```
//Primero utilizamos el método "delete (): boolean" para asegurarnos
//de que el fichero no existe con antelación
fichero.delete();
fichero.createNewFile();
```

Compila el código. ¿Qué ha sucedido?

8. Crea un nuevo bloque "catch (TipoDeLaExcepcion identificador) {...}" a continuación de los que ya has creado que te permita capturar la excepción que puede surgir al usar el método "createNewFile(): boolean".

Haz que en el mismo se muestre la cadena:

"Ha ocurrido un error de entrada y salida al crear el fichero"

Observa que en la carpeta en la que está tu proyecto ha aparecido un fichero de nombre "entrada_salida_practica11.txt".

9. Nuestra siguiente tarea ahora va a ser intentar volcar información a dicho fichero.

Hasta ahora hemos conseguido crear un fichero, pero para poder escribir en el mismo debemos crear un flujo de entrada de datos. Una de las clases que podemos utilizar para lo mismo es "FileWriter" (http://java.sun.com/j2se/1.5.0/docs/api/java/io/FileWriter.html). Además de asociar un flujo al fichero, por medio de la asociación de un "búfer" podremos mejorar las operaciones de escritura (para ello se utiliza la clase "BuffererWriter" http://java.sun.com/j2se/1.5.0/docs/api/java/io/BufferedWriter.html).

Los anteriores pasos los puedes llevar a cabo en un solo comando por medio de la orden:

BufferedWriter fichero_escribir = new BufferedWriter (new FileWriter (fichero));

10. Una vez has creado el objeto "fichero_escribir", observa los métodos que ofrece la clase "BufferedWriter" para volcar información al mismo (http://java.sun.com/j2se/1.5.0/docs/api/java/io/BufferedWriter.html).

Ejecuta las siguientes órdenes dentro del bloque "try {...}":

```
fichero_escribir.write("Hemos abierto el fichero para escritura");
fichero_escribir.newLine();
fichero_escribir.write("En el mismo solo podemos volcar cadenas de caracteres");
fichero_escribir.newLine();
fichero_escribir.write (cadena);
fichero_escribir.newLine();
Double aux = new Double (Math.random());
fichero_escribir.write (aux.toString());

//La siguiente orden hace que el "búfer" se vacíe hacia el fichero
fichero_escribir.flush();
//La siguiente orden hace que se cierre la conexión de escritura con el fichero
fichero_escribir.close();
```

Comprueba el contenido del fichero "entrada_salida_practica11.txt".

Parte 2. Generando métodos que lanzan excepciones propias.

11. Vamos a definir ahora un método auxiliar que sea capaz de leer el fichero anterior y recuperar el último de los valores "double" que contenga el mismo. Define un método auxiliar de nombre "escanea_fichero (File): double"

En el mismo vamos a asociar un "Scanner" (recuerda importar la librería "java.util.Scanner") con el fichero que nos pasan como parámetro:

```
Scanner lectura_fichero = new Scanner (fichero);
```

Si observas la especificación del método anterior en http://java.sun.com/j2se/1.5.0/docs/api/java/util/Scanner.html#Scanner(java.io.File) comprobarás que el mismo puede lanzar una excepción de tipo "FileNotFoundException", en caso de que el fichero usado como parámetro no exista. Modifica la cabecera de "escanea_fichero (File): double" para que propague dicha excepción, en lugar de gestionarla.

Una posible programación del cuerpo del método sería:

```
Scanner lectura_fichero = new Scanner (fichero);

Double un_double = new Double (0.0);

while (lectura_fichero.hasNext()){
        try {
                un_double = new Double (lectura_fichero.nextLine());
        }
        catch (NumberFormatException nfe){
                System.out.println ("El valor leido no era un double");
        }
}

return un_double;
```

12. Incluye, dentro del bloque "try{…}" que tienes en tu función "main", una llamada al método anterior pasándole como parámetro el fichero que hemos creado:

```
System.out.println ("Un double en el fichero " + escanea_fichero (fichero));
```

Considera la siguiente cuestión:

¿Qué ha pasado con la excepción "FileNotFoundException" que lanzaba nuestro método "escanea_fichero (File): double"? ¿Por qué nos hemos tenido que capturarla en el "main"? La información en http://java.sun.com/j2se/1.5.0/docs/api/java/io/FileNotFoundException.html podría servirte para responder la pregunta.

13. Vamos a ver ahora un ejemplo de gestión y propagación simultánea de una excepción. En el bloque "catch (...){...}" del ejercicio anterior realiza la siguiente modificación:

```
catch (NumberFormatException nfe){
        System.out.println ("El valor leido no era un double");
        throw nfe;
}
```

Compila y ejecuta tu programa de nuevo. ¿Observas algún cambio en el resultado de la ejecución del mismo?

Aparte de capturar la excepción dentro de "escanea_fichero(File): double", la hemos propagado al método que lo invoca (en esta caso el "main") cambiando el flujo de ejecución del programa.

Deja comentada la línea que hemos añadido en este ejercicio, "//throw nfe;".

14. Vamos a definir un segundo método auxiliar que lance excepciones. Define un método "raiz_cuadrada (double): double" que realice la siguiente función:

Si el valor que recibe es mayor que 0, devolverá la raíz cuadrada del mismo (la función "Math.sqrt (double): double" en http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Math.html#sqrt(double) te puede resultar de utilidad).

Si el valor es menor que 0, genera y lanza una excepción de tipo "Exception" con el mensaje "Has intentado calcular la raiz de un numero negativo". Ten en cuenta que la cabecera del método "raiz_cuadrada (double): double" debe incluir el hecho de que puede lanzar una excepción de tipo "Exception".

15. Recupera el valor "double" de "escanea_fichero (fichero)" en el bloque "try {...}" de la función "main". Su valor siempre estará entre 0 y 1, así que nunca se produciría la excepción que hemos programado ahora. Réstale 0.5 al mismo. Ahora invoca al método "raiz_cuadrada (double): double"

Como podrás observar, el compilador te pide que gestiones la excepción correspondiente. Crea un nuevo bloque "catch (TipoDeLaExcepcion ...){...}" después de los bloques "catch (...){...}" anteriores que te permita capturarla.

Dentro del bloque "catch(...){...}" invoca a los métodos "getMessage(): String" y "toString(): String" para observar su comportamiento.

16. Finalmente, vamos a intentar programar una excepción propia. En el mismo fichero "Principal_prac11.java" define una clase de nombre "NumeroNegativoExcepcion", que herede de la clase "Exception" (http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Exception.html). Define dos constructores en la misma a imagen de los propios de la clase "Exception" (recuerda que puedes hacer uso de los de la clase "Exception" por medio del comado "super (...)").

Modifica ahora el método del ejercicio 15 convenientemente para que lance la excepción "NumeroNegativoExcepcion", en lugar de "Exception". Modifica también el bloque "catch (...){...}" en el que se gestionaba la misma.

ENTREGAR el archivo *Principal_prac11.java*. El archivo debe contener **todos los cambios** realizados durante los ejercicios de la práctica. La entrega es a través de Belenus en el repositorio de Julio Rubio. También deben contener unas cabeceras que permitan identificarte:

```
/* Nombre: ___
     Grupo: _____
     Nombre del fichero: ____
     Descripción: _______*/
```