Advanced Programming 2015 – Year 2
Labwork 2: (5% - or 50 points out of 500 points for labwork this semester)

NOTE: ALL LABS TO BE COMPLETED IN PROJECTS USING ECLIPSE

Part 1 – Getting familiar with try – catch - finally (6 points)

Create an Eclipse Project called **Lab2Part1**. Create a class called **TestTryCatchFinally**. Write a try catch and finally block of code in the class. You can catch the **NullPointerException** exception in the catch block and make the catch block execute by deliberately setting a variable in the try block to **null** and attempt to use the variable (e.g. String s = null; s.toString()). Place an output statement in the finally block to verify that it has been called. Call **printStackTrace()** (e.printStackTrace()) within each of the catch blocks.

Required activities and marking guideline:

- Implemented try block (2 points)
- Implement catch NullPointerException (2 points)
- Implement finally block with output message (2 points)

Part 2 – Multiple catches for one try (10 points)

Create an Eclipse Project called **Lab2Part2**. Create a simple application called **TestMultipleCatches**. Write a try catch block in the main method so that ALL of the following exceptions are caught in three different catch blocks: **ArrayIndexOutOfBoundsException, NumberFormatException, Exception**. Ensure that you place the most generic exception as the last catch block. Add a simple test output statement to each of the catch blocks so that you can identify if the catch block was executed, e.g., "The Class Cast catch statement was executed". Add **printStackTrace()** to each catch block. Place code in the try block to test at least one of the exceptions listed (e.g. you could create an array of size four and then try to access a non-existant fifth index [4] to generate an **ArrayIndexOutOfBoundsException**, or you could try to use Interger.parseInt on a string like "s" which will cause a **NumberFormatException** [where "s" is a string that can't be formatted as an **int** as it's not a number]).

Required activities and marking guideline:

- Implement the try (2 points)
- Write all three catches with stack trace [2 points each] (6 points)
- Insert sample error and test catch block (2 points)

Part 3 Declaring Exceptions (10 points)

Create an Eclipse Project called **Lab2Part3**. Create a class called **TestDeclaringExceptions**. In this class create two methods. Create a static method called reverseString(String s) to reverse a string and output a reversed string passed in reverse; also the reverseString method must be declared to **throw a NullPointerException**. Create a second static method called openFile(String fileName) which will accept the name of a file and attempt to test whether the file exists [File file = new File(fileName); use file.exists() to test file existence]: also the openFile method must be declared to **throw IOException**. Now call both methods from the main method (note: you will be forced to place a try and catch around one the methods you have written).

Required activities and marking guideline:

- Implement reverseString method (2 points)
- Declare reverseString method to throw NullPointerException (1 point)
- Implement openFile method (2 points)
- Declare openFile method to throw IOException (1 point)
- Call both methods from the main (2 points)
- Add necessary try catch to method calls in main (1 point)
- Use comments to explain why one of the methods called in the main required a try block (or throw) using regular Java comments in the code (1 point)

Part 4 Putting it all together – GUI with exception handling (26 points)

Create an Eclipse Project called **Lab2Part4**. This lab is intended to test the following exception classes all within the same try catch finally block: **NullPointerException, MalFormedURLException, IOException, Exception**. Create a class called **TestFourExceptionsGUI**. Make the class a JFrame and implement the ActionListener interface. Add FOUR JButton's to the JFrame (use any layout you wish to use that looks reasonable). The button texts should be set to the following: "Test IO Exception", "Test URL Exception", "Test Null Pointer Exception", "Test General Exception". Add the listeners to the buttons. Add a method to the class called testException which accepts FOUR parameters: **(String string, String filename, String url, boolean generalExceptionActivated)**. Add a try catch finally block to the method called testException. Add the following code inside the try block of the testException method:

```
str.toCharArray(); //Null string potential error
new FileReader(filename); //Unknown filename potential error
new URL(url); //Badly written URL potential error
if(generalExceptionActivated) { //Potential error
      this.clone(); //Will be caught as a general error!
}
```

Add a four catch blocks for the above try block as follows (**e** is the parameter to the catch block in each case) – use **JOptionPane** class for messages:

- **MalformedURLException** with a **Message dialog box** that appears with the following message should the exception be caught: "A URL has been badly written" + e.getMessage()
- **IOException** with a **Message dialog box** that appears with the following message should the exception be caught: "An IO Exception has been caught" + e.getMessage()
- **NullPointerException** with a **Message dialog** box that appears with the following message should the exception be caught: "A Null Pointer Exception has been caught" + e.getMessage()
- **Exception** with the a **Message dialog box** that appears with the following message should the exception be caught: "Some general unidentified Exception has been caught" + e.getMessage()

Also add a **finally** block that will display in a **message dialog** "The finally block has been called". Additionally add a blank file called "Real.txt" to the project directory of the Lab2Part4 Eclipse project directory (this will be the valid file that the FileReader will accept to avoid an IOException).

Now implement the actions for each of the buttons so that each button when pushed will cause the specific exceptions to be caught. This will be work as follows:

- If the “Test IO Exception” button is pushed call the testException method with the following parameters ("Hi", "Whatever.txt", "http://www.itb.ie", false) [This will cause an **IOException** as there is no file called “Whatever.txt” in the project]
- If the “Test Null Pointer Exception” button is pushed call the testException method with the following parameters (null, "Real.txt", "http://www.itb.ie", false) [This will cause a **NullPointerException** as the string is set to null]
- If the “Test URL Exception” button is pushed call the testException method with the following parameters ("Hi", "Real.txt", "ht//www.itb.ie", false) [This will cause a **MalformedURLException** as the URL "ht//www.itb.ie" is not a correctly formatted URL]
- If the “Test General Exception” button is pushed call the testException method with the following parameters ("Hi", "Real.txt", "http://www.itb.ie", true) [This will be caught by the general exception catch block as it is not one of the specific exception types caught]

Add Javadoc to the class and Jar and execute the program.

Required activities and marking guideline:

- Create GUI with buttons added (2 points)
- Add listeners and handlers (2 points)
- Write the testException method [2 marks per catch block] (10 points)
- Implement button actions [2 marks each per action] (8 points)
- Jar the project (include Javadoc in the jar) (2 points)
- Run the project from the Jarfile and test all exceptions (2 points)