Advanced Programming 2025 – Year 2
Labwork 2: (3% - or 50 points out of 300 points for labwork this semester)

NOTE: ALL LABS TO BE COMPLETED IN PROJECTS USING ECLIPSE OR EQUIVALENT IDE (NO MORE TEXTPAD, EVER EVER EVER!!!)

TOPICS INCLUDE: try, catch, finally, exception handling and declaring

IMPORTANT NOTES:

- **NO COPYING PERMITTED AND ZERO MARKS WILL APPLY TO COPIED WORK. FURTHER ACTION MAY BE TAKEN AGAINST STUDENTS THAT HAVE BEEN FOUND TO COPY WORK.**

- **ASSESSMENT WILL INVOLVE ONE-TO-ONE QUESTIONS ABOUT YOUR SUBMITTED WORK. A COMPLETED SELF-ASSESSMENT SHEET WILL BE USED TO GUIDE THE ASSESSMENT. USE COMMENTS IN YOUR CODE TO ENSURE YOU DON'T FORGET WHY YOU WROTE CODE YOU MAY LATER BE ASKED ABOUT.**

- **ALL WORK MUST BE SUBMITTED TO MOODLE BY DATES SPECIFIED (SUBMISSION DEADLINES WILL BE POSTED ON MOODLE).**

- **MANY OF THE TASKS ASSIGNED BELOW CAN BE COMPLEX AND\OR THE DESCRIPTIONS MAY REQUIRE FURTHER CLARIFICATIONS. PLEASE USE THE AVAILABLE LAB TIMES TO ASK FOR CLARIFICATIONS AND ADVICE\HINTS ON THE TASKS BELOW.**

- **YOU MUST USE AN IDE TO COMPLETE TASKS (e.g., Eclipse or IntelliJ or NetBeans or similar). Support and help provided for Eclipse primarily.**

- **CHATGPT and other similar AI tools that can code simple solutions are NOT PERMITTED. THEY DO NOT TEACH YOU HOW TO BECOME A GOOD PROGRAMMER EITHER!**

Part 1 – Getting familiar with try – catch - finally (10 points)

Create an Eclipse Project called **Lab2Part1**. Create a class called **TryCatchTest**. Write a try catch and finally block of code in the class. You can catch the **StringIndexOutOfBoundsException** exception in the catch block. Make this error happen by creating a String with your surname\family name (e.g. String s = "Kelly") and try to access a character beyond the length of the string (e.g. char c = s.charAt(7) : This is more than 'K''e''l''l''y' string holds and will throw the exception for string out of bounds). Place an output statement in the finally block to verify that the finally block has been called. Call **printStackTrace()** (e.printStackTrace()) within the catch block (where 'e' is the catch block variable). Output your surname in the finally block using System.out.println.
Add a second try block to the class that has no catch (i.e., try and finally only). Output some sample text in the try block (e.g. "Test try with out catch") and also output some sample text from the finally block (e.g. "There's no catch").
Put this in a **jarfile** and run it from the jarfile.

Required activities and marking guideline:

- Implemented try block with deliberate error string out of bounds (2 points)
- Write catch StringIndexOutOfBoundsException for YOUR name (2 points)
- Implement finally block with an output message (2 points)
- Call print stack trace (study the output trace, understand it?) (1 point)
- Create and the try without a catch block with text outputs (2 points)
- Jar the project and run it from the jarfile (1 point)

Part 2 Declaring Exceptions – Checked Versus Unchecked (10 points)

Create an Eclipse Project called **Lab2Part2**. Create a class called **CheckedVersusUncheckedExceptions**. In this class create two methods. Create a static method called capitalizeString(String s) to convert a String that's passed to ALL CAPS. The capitalizeString 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]. The openFile method must be declared to **throw IOException**. Call both methods from the main method (note: you will be forced by the compiler to place a try and catch around one the methods you have written due to it being a checked exception).

Required activities and marking guideline:

- Implement capitalizeString method (2 points)
- Declare capitalizeString 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 (1 point)
- Add necessary try catch or throws to method calls in main (1 point)
- Add comments to your code to explain why one of the methods called in the main required a try block (or throw) and the other method did not (use regular Java comments in the code – no comment, no marks!) (2 points)

Part 3 – try catch with multiple exceptions (20 points – 10 for Part a, b)

a) Create an Eclipse Project called **Lab2Part3a**. Create a simple application called **MultipleCatchBlocks**. Write a *try catch* block in the main method so that ALL of the following exceptions are caught in three different *catch* blocks [but only **ONE** try block]: **ArrayIndexOutOfBoundsException, ArithmeticException, Exception**. Ensure that you place the most generic exception as the last catch block (normally Eclipse will prevent you doing otherwise!). 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., "You are trying to access beyond the array bounds". 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 dividing a number by zero (to create **ArithmeticException**). Test all of the catches in turn (comment out and comment back in the could errors to execute different catch blocks).

Required activities and marking guideline:

- Implement the try (3 points)
- Write all three catches with stack trace and output (4 points)
- Insert sample error and test catch block (3 points)

b) Create an Eclipse Project called **Lab2Part3b**. Create a simple application called **MultiCatchUsingOr**. Write a try catch block in the main method so that ALL of the following exceptions are caught in **THE SAME** catch block [but only ONE try block is to be used!]: catch **NullPointerException** and **StringIndexOutOfBoundsException** in the same catch block using the OR statement notation, e.g., '|'. You can cause either of these exceptions to catch if you set the string in the try block to **null** or attempt to output a character using **charAt** in an index greater than the string size (like **Part 1** above). In both cases of catching the exception print the stack trace (**printStackTrace()**).

- Implement the try (2 points)
- Catch two exceptions in same catch block using '|' (the OR) (4 points)
- Test EACH catch works using strings with null and charAt (2 points)
- Print the stack trace (you see the null and bounds exceptions) (2 points)

Part 4 Putting it all together – GUI with exception handling (20 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: "The URL passed is not correctly formatted " + e.getMessage()
- **IOException** with a **Message dialog box** that appears with the following message should the exception be caught: "There has been an input\output error " + e.getMessage()
- **NullPointerException** with a **Message dialog** box that appears with the following message should the exception be caught: "An null object has been passed to the method " + e.getMessage()
- **Exception** with the a **Message dialog box** that appears with the following message should the exception be caught: "An unidentified error has occurred " + 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 generate the Javadoc. Run the program a few times and test the exceptions respond correctly (Real.txt needs to be accessible in the correct location for the IDE or the URL\File IO error may not appear).

Required activities and marking guideline:

- Create GUI with buttons added (4 points)
- Add listeners and handlers (4 points)
- Write the testException method [1 mark per catch block] (4 points)
- Implement button actions [1 mark each per action] (4 points)
- Javadoc the project and generate the Javadoc (4 points)