Advanced Programming 2025 – Year 2
Labwork 3: (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: Exception handling and declaring, custom exceptions, JDBC

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 – Try-with-resources (10 points)

Create an Eclipse Project called **Lab3Part1**. Add a text file to the project directory called *myFavPastimes.txt* which contains a list with the names of your favorite pastimes\hobbies (pick something even if you don't have pastimes) – include at least THREE pastimes in the text file. Create a File variable called **pastimeFile** that references the *myFavPastimes.txt* file. Write a small GUI JFrame with a label or text area and one button that says "Print My Pastimes From File". Write a program to read all of your favourite pastimes list from the **pastimeFile**. **Use the *try-with-resources* approach** to read the file and output the contents of the file to the label or text area in the JFrame. [NOTE: You must use try-with-resources approach to receive marks for this exercise – see the lecture notes!!]

- Create file and add at least THREE pastimes (1 point)
- Read ALL of the information from the file (3 points)
- Use try-with-resources (not a regular try block!!!) (3 points)
- Implement listener and handler for button (1 point)
- Display the contents of the file to the frame (2 points)

Part 2 – Retry an action in catch block (10 points)

Create an Eclipse Project called **Lab3Part2**. Create a simple JFrame GUI with a JButton called **inputButton.** Create an array in your program called **cityArray** with the contents defined as {"Cork","Dublin","Belfast","Galway"}. Create and add a **JLabel** with the String "Please enter the array index you wish to output: ". Use a **JTextfield** called **inputField** to input the index number of the array requested by the user. Create and add a **JLabel** called **outputLabel** to output the contents of the array at the index requested by the user, e.g., if the user enters '0' output "Cork" to the output label, e.g., outputLabel.getText(array[0]). Add the listeners necessary to get the index and display the contents of the array and display in the output field. Place a **try..catch** block around the code to output the contents of the array: catch an **ArrayIndexOutOfBoundsException**. If this exception is caught (requesting beyond the array size!!) use an option pane **input dialog** to give the user a second chance to try and output within the array bounds and show the output in the **outputLabel**, e.g., message via the input dialog "You attempted to access beyond the limits of the array, please enter a number between 0 and 3". If the user fails to keep within limits again you do not have to deal with that! Jar the project and test the running of the project from the jarfile. Javadoc the project.

Required activities and marking guideline:

- GUI created and layout reasonable (2 points)
- Listeners working (2 points)
- actionPerformed with try and catch (2 points)
- Retry works with input dialog (2 points)
- Javadoc (1 point)
- Jar and run the GUI from the Jar (1 point)

Part 3 – Write custom exceptions, throw and handle them (15 points)

Create an Eclipse Project called **Lab3Part3**. Write a class called **MyNetworkChecker**that will attempt to verify the network and balance for your mobile phone and throw exceptions using exception handling if there's an issue. This class will store your network and account balance and will use exception handling when the network is incorrect or when the balance is too low (e.g., below 1 minimum).

Create TWO custom exception classes called **WrongNetworkException** and **BalanceBelowLimitException**. Put these exceptions in two separate classes in the same package called *exception*.

Write TWO static methods within the **MyNetworkChecker**class. The first method will be called **checkMyMobileNetwork(String inputNetwork)**. This method must declare **(throws)** the custom **WrongNetworkException** exception. The second method will be called **checkMyBalance(int balance)**. This method must declare **(throws)** the custom **BalanceBelowLimitException** exception.

Implement the **checkMyMobileNetwork** method so that it matches the string entered with your network name (e.g. Vodafone, Three etc. whatever you have set it to in the class), if the check succeeds output a message to verify the network passed is correct. If the string passed for the network is checked and it is NOT the same as your network throw a WrongNetworkException using the keyword **throw**.

Implement the **checkMyBalance** method so that it takes in an integer value representing the current balance in the account. If it is above minimum output a message to verify the balance is fine. If the integer passed for the balance is BELOW the minimum allowed throw a BalanceBelowLimitException using the keyword **throw**.

Test the calling of the **checkMyMobileNetwork** AND **checkMyBalance** methods in the main (test the passing of the incorrect phone make to the network method so that the exception is thrown and try passing a balance that is too low also). Include *printStaceTrace()* in the catch blocks. Fully Javadoc the project. Jar the project and test the running of the project from the jarfile.

Required activities and marking guideline:

- Create the WrongNetworkException class in package *exception* (2 points)
- Create the BalanceBelowLimitException class in package *exception* (2 points)
- Create the checkMyMobileNetwork method and declare exception (2 points)
- Create the checkMyBalance method and declare exception (2 points)
- Test EACH methods in the main (try-catches and stack trace etc.) (4 points)
- Javadoc ALL classes and methods and generate (2 points)
- Jar the project and run the Jarfile (1 point)

Part 4 JDBC CRUD Operations (15 points)

Create an Eclipse Project called **Lab3Part4**. Create a new Database using a database to store your favorite songs and set a *username* and *password* for access to the database. Write four methods in total to carry out CRUD operations on your database using JDBC. Specifically, carry out the following operations on the database using four JDBC-based methods:

- Create a database table with at least two attributes one them being an ID (primary key)
- Add some data to the table, at least two sample records should be added to the table
- Once data has been added to the table, read that data back in and output it (either to default output device (System.out) or via a GUI, your choice!)
- Delete one of the records from the database and repeat the output to ensure it has been deleted

Test all of your methods to ensure the work successfully. Javadoc the methods and generate the Javadoc.

Required activities and marking guideline:

- Create the DB complete with user *username* and *password* (1 point)
- Write and test method to create a table using JDBC (3 points)
- Write and test method to add data to the table using JDBC (3 points)
- Write and test method to read data from the table using JDBC (3 points)
- Write and test method to delete a record from table using JDBC (3 points)
- Javadoc methods and classes properly and generate the Javadoc (2 points)