|  |
| --- |
| **Computer Engineering Department - ITU** |
| **CE101L: Object-Oriented Programming Lab** |

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated:** |
| **Teaching Assistant: Zain** | **Semester: Spring 2023** |
| **Lab Engineer: Rana Hamza Shakil** | **Batch: BSCE2022** |

# **Lab 14 B. Problem-Based Learning in Java**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
|  |  |  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Objective**

The objective of this lab is to observe the basic knowledge of programming in C++.

## **Equipment and Component**

|  |  |  |
| --- | --- | --- |
| **Component Description** | **Value** | **Quantity** |
| Computer | Available in lab | 1 |

## **Conduct of Lab**

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA), and lab attendants.

## **Theory and Backgrou****nd**

Java classes are defined using the keyword "class" followed by the class name, which should be in camel case. The variables and methods are defined within the curly braces {} of the class. A class can also have access modifiers such as public, private, and protected, which control the accessibility of the variables and methods to other classes.

To create an object of a class in Java, the "new" keyword is used followed by the class name and the constructor. Once an object is created, its properties can be accessed and modified using dot notation, where the object name is followed by a dot and the name of the variable or method. Overall, classes in Java provide a powerful mechanism for creating modular, reusable, and maintainable code. They enable programmers to encapsulate complex data and logic into easy-to-use objects, making programming more intuitive and efficient.

**Lab Task**

**Task A [Marks: 5]**

Please follow the following steps before starting the below tasks:

1. Create a ‘.Java’ file to define your Java classes and methods, and a ‘Main’ class with a ‘main’ method to run your program. If task requires the use of external libraries or dependencies, you may also need to include one or more ‘.jar’ files in your task. For

2. In order to create multi-file program, you need to organize your code into packages and include one or more directories with ‘.Java’ files. In this case, you would also need to include a ‘package-info.java’ file in each package to define the package.

**Task B: [Marks: 35]**

Design a program in Java that converts temperatures from Fahrenheit to Celsius and also calculates various statistics based on the converted temperatures using the following steps:

1. Read a series of temperatures in Fahrenheit from the user using Scanner class. The user can enter any number of temperatures, and the program should continue to prompt the user for input until they enter a non-numeric value.
2. Convert each temperature from Fahrenheit to Celsius using the formula Celsius = (Fahrenheit - 32) \* 5/9 and store the converted temperatures in an ArrayList.
3. Calculate and print the following statistics based on the converted temperatures: (a). The minimum temperature. (b) The maximum temperature (c) The average temperature (d) The median temperature (e) The standard deviation of the temperatures

Note: To calculate the median and standard deviation, you will need to implement appropriate algorithms in Java. You can use the formulas for these calculations found online or in a statistics textbook.

1. Print the converted temperatures and their corresponding Fahrenheit values side-by-side in a tabular format. The table should include the original Fahrenheit values, the converted Celsius values, and a label indicating whether the temperature is "cold" (below 0°C), "normal" (between 0°C and 100°C), or "hot" (above 100°C).

The program should handle invalid input and display an appropriate error message if the user enters an invalid value. Additionally, the program should be well-structured and follow good programming practices, such as using appropriate data structures and separating code into functions or classes where appropriate. Finally, the program should include comments explaining the purpose and behaviour of each major section of code.

|  |
| --- |
| // Paste your code here |

### Assessment Rubric for Lab

**Method for assessment:**

Lab reports and instructor observation during lab sessions. Outcome assessed:

a. Ability to conduct experiments, as well as to analyze and interpret data (P) b. Ability to function on multi-disciplinary teams (A)

c. Ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (P)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Performance metric** | **Task** | **CLO** | **Description** | **Max marks** | **Exceeds expectation** | **Meets expectation** | **Does not meet expectation** | **Obtained marks** |
| 1. Realization of experiment (a) | 1 | 1 | Functionality | 40 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed (35-40) | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed (20-34) | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed (0-19) |  |
| 2. Teamwork (b) | 1 | 3 | Group Performance | 5 | Actively engages and cooperates with other group member(s) in effective manner (4-5) | Cooperates with other group member(s) in a reasonable manner but conduct can be improved (2-3) | Distracts or discourages other group members from conducting the experiment (0-1) |  |
| 3. Conducting experiment (a, c) | 1 | 1 | On Spot Changes | 10 | Able to make changes (8-10) | Partially able to make changes (5-7) | Unable to make changes (0-4) |  |
| 1 | 1 | Viva/Quiz | 10 | Answered all questions (8-10) | Few incorrect answers (5-7) | Unable to answer all questions (0-4) |  |
| 4. Laboratory safety and disciplinary rules (a) | 1 | 3 | Code commenting | 5 | Comments are added and does help the reader to understand the code (4-5) | Comments are added and does not help the reader to understand the code (2-3) | Comments are not added (0-1) |  |
| 5. Data collection (c) | 1 | 3 | Code Structure | 5 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap (4-5) | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables (2-3) | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy (0-1) |  |
| 6. Data analysis (a, c) | 1 | 4 | Algorithm | 20 | Solution is efficient, easy to understand, and maintain (15-20) | A logical solution that is easy to follow but it is not the most efficient (6-14) | A difficult and inefficient solution (0-5) |  |
| 7. Computer use (c) | 1 | 2 | Documentation & GitHub Submissions | 5 | Timely (4-5) | Late (2-3) | Not done (0-1) |  |
|  | Max Marks (total): | | | 100 | Obtained Marks (total): | | |  |

Lab Engineer Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_