|  |
| --- |
| **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 9B. Singleton Implementation in C++**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 Background**

Singleton is a design pattern in C++ that restricts the instantiation of a class to a single instance and provides a global point of access to that instance. It ensures that only one instance of the class is created, and provides a way to access that instance throughout the program. The Singleton pattern is commonly used for managing resources that should only have one instance, such as a database connection pool, or for providing a global configuration object that can be accessed by multiple parts of the program. The Singleton pattern is implemented using a private constructor, a private static member variable that holds the single instance of the class, and a public static member function that provides access to that instance.

**Lab Task**

**Task A [Marks: 5]**

Please follow the following steps before starting below tasks:

1. Create a separate header file (**.h file**) for each class declaration of data members and member functions

2. Create a separate source file (**.cpp file**) for the implementation of the class member functions.

3. Create **main.cpp** file for creating objects of class and other driving code.

**Task B: Inheritance vs Aggregation [Marks: 35]**

Implement a Singleton class in C++ that can be used to manage the logging functionality of a program. The Singleton should allow only one instance to be created and should provide a way to log messages to a file.

**Step 1: Define the Singleton class**

* Create a header file for the Singleton class, defining a private constructor and a private static member variable that holds the single instance of the class. Define a public static member function that provides access to the single instance of the class, and ensures that only one instance is created.

**Step 2: Implement the Singleton class**

* In the implementation file for the Singleton class, define the private constructor that initializes the logging file. Define the static member variable that holds the single instance of the class, and ensure that only one instance is created by checking if the instance is already initialized.
* Define the static member function that provides access to the single instance of the class, creating the instance if it doesn't already exist.
* Define a public member function that can be used to log messages to the file.
* Add an array dynamically created for size 1 initially of Person type class. Create a Person class with attributes, first name and last name and age

**Step 3: Use the Singleton class to log messages**

* Create a main function that retrieves the single instance of the Singleton class and uses it to log some messages.
* Test the Singleton by attempting to create multiple instances of the Singleton class and verifying that they all refer to the same logging file.

**Step 4: Create a Menu Class**

* Create a Menu class, make a function in it, which gives an option to add new person, find person, delete existing and modify existing.
* Use singleton class to update content without composition or aggregation.

|  |
| --- |
| // 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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_