|  |
| --- |
| **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 8B. Association vs Composition in Object-Oriented Programming**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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**

In C++, association and composition are two types of relationships between classes. Here are the main differences between the two:

Definition: Association is a relationship between two classes where one class has a reference to another class, but the two classes can exist independently. Composition is a relationship between two classes where one class owns the other class, and the two classes have a dependent relationship - if the owning class is destroyed, the owned class is also destroyed.

1. Object creation: In association, objects of each class can be created independently of each other. In composition, the owned class is created and destroyed by the owning class.
2. Multiplicity: In association, the relationship can be one-to-one, one-to-many, or many-to-many. In composition, the owned class can only be part of one owning class at a time.
3. Dependency: In association, there is no dependency between the two classes. In composition, the owned class is dependent on the owning class.
4. Memory management: In association, memory management is simpler as the two classes can be created and destroyed independently. In composition, memory management can be more complex as the owned class is created and destroyed by the owning class, and proper deletion of the owned class must be ensured to avoid memory leaks.

In summary, the association is a looser relationship between two classes where each class can exist independently, while the composition is a tighter relationship where one class owns the other and the two classes are dependent on each other.

**Lab Task**

**Task A [Marks: 5]**

Please follow the following steps before starting the 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 the class and other driving codes.

**Task B: Association Vs Composition [Marks: 35]**

**Step 1: Implementing Association Relationships**

Students will create two classes, Book and Borrower, in a library system. The Book class should have attributes for the book's title, author, number of pages, and a reference to a Borrower object to establish an association relationship. The Borrower class should have an attribute for the borrower's name and a vector of pointers to Book objects to establish the association relationship. Students will create objects of both classes and test the association relationship by accessing the methods and attributes of each class through the other.

1. Methods for Book Class: getters and setters for title, author, and number of pages; getBorrower() to retrieve the reference to the Borrower object associated with the book
2. Methods for Browser Class: getters and setters for the name; addBook() and removeBook() to manage the vector of pointers to Book objects; getBooks() to retrieve the list of Book objects associated with the borrower

**Step 2: Implementing a composition relationship between two classes**

Students will create two classes, Restaurant and Menu, in a restaurant system. The Restaurant class should have an attribute for the restaurant's name and a Menu object to establish a composition relationship. The Menu class should have an attribute of menu items. Students will implement methods in the Restaurant class to add and remove items from the menu, and to get the list of menu items. They will test the composition relationship by creating and destroying objects of the Restaurant and verifying that the objects of the Menu are also created and destroyed accordingly.

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