Department of Computer and Software Engineering- ITU

CE101L: Object Oriented Programming Lab

Course Instructor: Nadir Abbas	**Dated: 25/02/2025**
Teaching Assistant: Zainab, Nahal, Bilal	**Semester: Spring 2025**
Session: 2024-2028	**Batch: BSCE24**

Lab 6. Implementation Of Inheritance And Its Levels

Name	Roll number	Obtained Marks/35

Checked on: _____

Signature: _____

Objective

The objective of this lab is to help students understand and apply concepts of Inheritence.

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

Understanding Inheritance

Inheritance is a mechanism that allows one class (derived class) to inherit attributes and methods from another class (base class). It promotes **code reusability** and establishes a hierarchical relationship among classes.

Types of Inheritance

1. **Single Inheritance**
    - In single inheritance, a derived class inherits from only one base class.
    - This is the simplest form of inheritance, where the child class extends the functionality of its parent.
    - Example: A Car class inheriting from a Vehicle class.
2. **Multi-level Inheritance**
    - In multi-level inheritance, a derived class inherits from another derived class, forming a chain of inheritance.
    - This allows a class to inherit properties from multiple levels in a hierarchy.
    - Example: A SportsCar class inheriting from Car, which in turn inherits from Vehicle.
3. **Hierarchical Inheritance**
    - In hierarchical inheritance, multiple derived classes inherit from the same base class.
    - This allows multiple child classes to share common properties and behaviors defined in the base class.
    - Example: Both Car and Bike classes inheriting from a Vehicle class.
4. **Multiple Inheritance**
    - In multiple inheritance, a derived class inherits from more than one base class.
    - This allows a class to combine functionalities from multiple sources but can introduce complexity due to potential conflicts.
    - Example: A Professor class inheriting from both Teacher and Researcher classes.

Tasks

Task 1: Accept the assignment posted in Google Classroom and after accepting clone the repository to your computer for this ensure you have logged into github app with your account.

Task 2: Solve the given problems written after task instructions, write code through IDE like CLion

Task 3: Ensure your code/solution is in the cloned folder.

Task 4: Commit and Push the changes through the Github App

Write code in functions, in files **functions.cpp**, **functions.h** and **main.cpp** after completing each part, verify through running code using **"make run"** on Cygwin.

Question 1: Creating the Class Hierarchy

Create the following classes demonstrating **multi-level** and **hierarchical inheritance**:

1. **Base Class: Vehicle**
    - Data Members:
        - string brand
        - int year
    - **Vehicle(string b, int y);** Constructor to initialize brand and year.
    - Function: void displayInfo(); (to print vehicle details)
    - **Getters and Setters:**
        - string getBrand();
        - void setBrand(string b);
        - int getYear();
        - void setYear(int y);
2. **Derived Class: Car (inherits from Vehicle)**
    - Data Members:
        - int seats
    - **Car(string b, int y, int s);** Constructor to initialize brand, year, and seats.
    - Function: void showCarDetails(); (to display car details)
    - **Getters and Setters:**
        - int getSeats();
        - void setSeats(int s);
3. **Derived Class: Bike (inherits from Vehicle)**
    - Data Members:
        - bool hasGear
    - **Bike(string b, int y, bool g);** Constructor to initialize brand, year, and gear status.
    - Function: void showBikeDetails(); (to display bike details)
    - **Getters and Setters:**
        - bool getHasGear();
        - void setHasGear(bool g);
4. **Derived Class: SportsCar (inherits from Car)**
    - Data Members:

    - double topSpeed
- **SportsCar(string b, int y, int s, double ts);** Constructor to initialize brand, year, seats, and top speed.
- Function: void showSportsCarDetails(); (to display sports car details)
- **Getters and Setters:**
    - double getTopSpeed();
    - void setTopSpeed(double ts);

Question 2: Adding a Function in the Vehicle Class

Function to Implement:

int getVehicleAge();

- Implement this function in the Vehicle class.
- It should return the age of the vehicle based on the current year (assume the current year is 2025 for calculations).

Question 3: Adding a Function in the Car Class

Function to Implement:

void modifySeats(int newSeats);

- This function should modify the seats of the car.
- Ensure that newSeats has a positive value before updating.

Question 4: Adding a Function in the Bike Class

Function to Implement:

bool isGearBike();

- This function should return true if the bike has gears, otherwise false.

Question 5: Adding a Function in the SportsCar Class

Function to Implement:

void upgradeSpeed(double increase);

- This function should increase the topSpeed of the sports car by the given value.
- Ensure that the speed increase is a positive number.

Question 6: UML Diagram Submission

- Create a UML diagram representing the class hierarchy.
- Indicate inheritance relationships clearly.
- Save the UML diagram as an image file (e.g., uml_diagram.png).
- Place the UML diagram inside the submission folder.

Assessment Rubric for Lab

Performance metric	CLO	Able to complete the task over 80% (4-5)	Able to complete the task 50-80% (2-3)	Able to complete the task below 50% (0-1)	Marks
1. Realization of experiment	1	Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed.	Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed.	Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed.	
2. Conducting experiment	1	Able to make changes and answered all questions.	Partially able to make changes and few incorrect answers.	Unable to make changes and answer all questions.	
3. Computer use	2	Document submission timely.	Document submission late.	Document submission not done.	
4. Teamwork	3	Actively engages and cooperates with other group member(s) in effective manner.	Cooperates with other group member(s) in a reasonable manner but conduct can be improved.	Distracts or discourages other group members from conducting the experiment	
5. Laboratory safety and disciplinary rules	3	Code comments are added and does help the reader to understand the code.	Code comments are added and does not help the reader to understand the code.	Code comments are not added.	
6. Data collection	3	Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap.	Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables.	Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy.	
7. Data analysis	4	Solution is efficient, easy to understand, and maintain.	A logical solution that is easy to follow but it is not the most efficient.	A difficult and inefficient solution.	
Total (out of 35):					