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

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 14/04/2022** |
| **Teaching Assistant: Aqsa Khalid** | **Semester: Spring 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 6B. Use of Inheritance Hierarchies and Multiple Inheritance in Classes and Objects**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
| NIMRA MAQBOOL | BSCE21012 |  |  |  |

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

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

## **Objective**

The objective of this lab is to observe the basic knowledge of programming classes 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**

A **multi**-**level** **inheritance** is a concept in which a class can also be derived from one class, which is already derived from another class. In **multiple** **inheritance**, a class can also be derived from more than one base class.

Suppose the same function is defined in both the derived class and the based class. Now if we call this function using the object of the derived class, the function of the derived class is executed. This is known as **function** **overriding** in C++. The function in derived class overrides the function in base class.

**Lab Task**

**Task A: Multi Level Inheritance and Function Overriding [Marks: 20]**

In this task, you are required to create three classes’ **Staff (Parent class)**, **Department (Child class)**, and **Member (Grandchild class)** with the following data members and member functions,

***Public Data Members of class Staff such as:***

salary(int)

***Public Member Functions of class Staff such as:***

**//overriden function**

**void base\_public() –** It will print “I am a function from base class”.

***Public Member Functions of class Department such as:***

**void setsalary(int) –** It will initialize data member salary.

***Public Member Functions of class Member such as:***

**//overriding function**

**void base\_public() –** It will print “I am a function from grandchild class”.

**int getsalary() –** It will return salary.

Create a **UML diagram.**

Do the following operations in main function:

1. Create object of class **Staff** and call member function **void base\_public()**.

2. Create object of class **Member** (grandchild class) and call member function **void base\_public()**, **void setsalary(int s)**, **void grandchild\_public(), int getsalary()** and print salary**.**

|  |
| --- |
| Diagram  Description automatically generated  **Functions.h:**  class staff { public:  int salary; //making public variables  void base\_public(); //making public function };  class department : public staff { public:  void setSalary(int number); //making public function }; class member : public department{ public:  void base\_public(); //making public functions  int getSalary(); };  **function.cpp:**  #include "Functions.h" #include <iostream> using namespace std; void staff::base\_public(){  cout<<"I AM A FUNCTION FROM BASE CLASS"<<endl; //displaying } void department::setSalary(int number){ //giving parameters  cout<<"enter number = "; //taking number from user  cin>>number;  salary=number; //storing it to salary } void member::base\_public(){  cout<<"I AM A FUNCTION FROM GRANDCHILD CLASS"<<endl; //displaying } int member::getSalary(){  cout<<"SALARY = "<<salary<<endl; //displaying a salary  return salary; //returning }  **Main.cpp:**  int main() {  int opt; //declaring  do{  cout<<"WHICH TASK DO YOU WANT TO PERFORM ?"<<endl;  cout<<"1.TASK 1"<<endl;  cout<<"2.TASK 2"<<endl; //taking options to perform tasks  cout<<"3.EXIT"<<endl;  cin>>opt; //taking input  if(opt==1){  staff stf; //making obj  stf.base\_public(); //calling  member m; //making object  m.base\_public();  int number; //initializing  m.setSalary(number);  m.base\_public(); //calling  m.getSalary();  }  **output:**  **Text  Description automatically generated** |

**Task B: Multiple Inheritance [Marks: 20]**

In this task, you are required to create two parent classes’ **Parent\_One (Parent class)**, **Parent\_Two (Parent class)**, and one child class **Derived\_Class (Child class)** with the following data members and member functions,

***Public/Protected Data Members of class Parent\_One such as:***

a, b(int)

***Public Member Functions of class Parent\_One such as:***

**void set() –** It will initialize a, b with any integer.

***Public/Protected Data Members of class Parent\_Two such as:***

x, y(int)

***Public Member Functions of class Parent\_Two such as:***

**void set\_input (int) –** It will take input x, y.

***Public Member Functions of class Derived\_Class such as:***

**void show() –** It will print addition of a,b(Parent\_One class data members) and x,y(Parent\_Two class data members).

Create a **UML diagram**.

Do the following operations in main function:

1. Create object of class ***Derived\_Class*** and call member functions such as **void set(), void set\_input()**, **void show().**

|  |
| --- |
| Diagram  Description automatically generated  **Parent.h:**  #include <iostream> using namespace std; class parent\_one { public:  int a;  int b; //declaring  void set(int num1 ,int num2); //declaring function }; class parent\_two{ public:  int x; //declaring  int y;  void setInput(int number1,int number2); //declaring function }; class derived : public parent\_one , public parent\_two{ public:  void show(); //declaring };  **Parent.cpp:**  #include "parents.h" #include <iostream> using namespace std; void parent\_one::set(int num1 ,int num2){  cout<<"Enter first number = ";  cin>>num1;  cout<<"enter second number = "; //taking numbers from the user  cin>>num2;  a=num1; //storing it to a and b  b=num2; } void parent\_two::setInput(int number1,int number2){  cout<<"enter first number = ";  cin>>number1; //taking numbers  cout<<"enter second number = ";  cin>>number2;  x=number1; //storing it to x and y  y=number2; } void derived::show(){  int sum;  sum=a+b; //taking sum of a and b  int sum1;  sum1=x+y; //taking sum of x and y  cout<<" SUM OF PARENT A = "<<sum<<endl; //displaying sum  cout<<" SUM OF PARENT B = "<<sum1<<endl; }  **Main.cpp:**  int main() {  int opt; //declaring  do{  cout<<"WHICH TASK DO YOU WANT TO PERFORM ?"<<endl;  cout<<"1.TASK 1"<<endl;  cout<<"2.TASK 2"<<endl; //taking options to perform tasks  cout<<"3.EXIT"<<endl;  cin>>opt; //taking input  if(opt==1){  staff stf; //making obj  stf.base\_public(); //calling  member m; //making object  m.base\_public();  int number; //initializing  m.setSalary(number);  m.base\_public(); //calling  m.getSalary();  }  if(opt==2){  derived d; //making object  int num1; //declaring  int num2;  int number1;  int number2;  d.set(num1,num2); //calling  d.setInput(number1,number2);  d.show();  }  if(opt==3){  cout<<"YOU CHOOSE TO EXIT.."<<endl; //displaying  exit(5); //exiting  }  }  while(opt>=1 && opt<=3);    return 0; }  **output:**  **Text  Description automatically generated** |

#### **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 | 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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_