**Experiment No. 3**

**Title:** **Implementation of multilevel inheritance**

**Batch: B2 Roll No.: 16010421091**

**Experiment No.:3**

**Aim**: Write a C++ program to define classes ‘Student’ , ‘CAMarks’ and ‘Total marks’ as per the given description. Define class ‘Student’ having roll number of student as attribute. Accept roll number of student from user inside the constructor and define display( ) function to print the roll number of student. Next define class ‘CAMarks’ class which inherits ‘Student’ class. ‘CAMarks’ class stores course id, ISE marks and IA marks of a student inputted by the user. Define calculateCAMarks( ) function in the class , which will calculate the total CA marks of the student. Override display( ) unction to print the ISE, IA and total CA marks of the student. Finally define ‘Total marks’ class which inherits ‘CAMarks’ class. ’Total Marks’ class stores ESE marks scored by the student in the same course which is inputted by the user. Define calculateTotalMarks( ) function in the class , which will calculate the total of CA marks and ESE marks of the student. Override display( ) function to print all the details and all the marks of the student. Inside main ( ) function, using ‘Total Marks’ class demonstrate the calculation of total marks of a student and display all the information of a student.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Resources needed: Text Editor, C++ compiler**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

### Theory:

**Inheritance:**

Inheritance is the process by which objects of one class acquired the properties of objects of another classes. The concept of inheritance provides the idea of reusability.This means that we can add additional features to an existing class without modifying it. This is possible by deriving a new class from the existing one. The new class will have the combined feature of both the classes. The existing class is called base class and the new class derived from the existing class is called the base class.

There are five types of inheritance

1. Single level Inheritance: It has one base class and one derived class.
2. Multilevel Inheritance: It has one base and one derived class. The derived class is a base class for another derived class.
3. Multiple Inheritance: It has one derived class which inherits more than one base classes.
4. Hierarchical Inheritance: It has one base class and more than one derived classes.
5. Hybrid Inheritance: It is any combination of the above four types of Inheritance .

The syntax for declaring the derived class is as follows

class derived-class-name: visibility-mode base-class-name

{ //members of derived class

};

The private members of a base class are never inherited in the derived class.

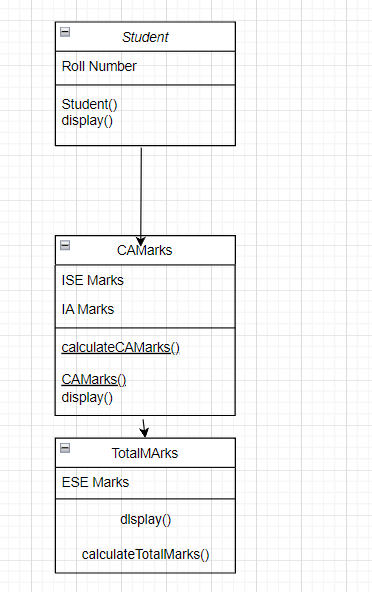
The visibility-mode is optional and it is private by default.

1. When the visibility mode is private, public and protected members of the base class become private members of the derived class.
2. When the visibility mode is protected, public and protected members of the base class become protected members of the derived class.
3. When the visibility mode is public, public and protected members of the base class become respectively public and protected members of the derived class.

**Function Overriding**

If we inherit a class into the derived class and provide a definition for one of the base class's function again inside the derived class, then that function is said to be overridden, and this mechanism is called Function Overriding. For function overriding, function that is redefined must have exactly the same declaration in both base and derived class, that means same name, same return type and same parameter list.

**Class Diagram:**



**Results: (Program with snapshot of output)**

**#include<iostream>**

**using namespace std;**

**class Student**

**{**

**public:**

**int roll;**

**Student()**

**{**

**cout<<"Please enter your roll no.: ";**

**cin>>roll;**

**}**

**void display()**

**{**

**cout<<"Roll no. is "<<roll<<endl;**

**}**

**};**

**class CAMarks : public Student**

**{**

**public:**

**int courseid,ise,ia;**

**CAMarks()**

**{**

**cout<<"Enter CourseID: ";**

**cin>>courseid;**

**cout<<"Enter ISE Marks: ";**

**cin>>ise;**

**cout<<"Enter IA marks: ";**

**cin>>ia;**

**}**

**void calculateCAmarks()**

**{**

**cout<<"Your CA marks are: "<<ise+ia;**

**}**

**void display()**

**{**

**cout<<"Your ISE marks are: "<<ise<<endl;**

**cout<<"Your IA marks are: "<<ia<<endl;**

**cout<<"Your Total CA marks are: "<<ise+ia<<endl;**

**}**

**};**

**class TotalMarks : public CAMarks**

**{**

**public:**

**int courseid;**

**int ese;**

**TotalMarks()**

**{**

**cout<<"Enter ESE marks: ";**

**cin>>ese;**

**}**

**void calculateTotalMarks()**

**{**

**cout<<"Your CA marks are: "<<ise+ia<<endl;**

**cout<<"Your ESE marks are: "<<ese<<endl;**

**}**

**void display()**

**{**

**cout<<"Your ISE marks are: "<<ise<<endl;**

**cout<<"Your IA marks are: "<<ia<<endl;**

**cout<<"Your CA marks are: "<<ise+ia<<endl;**

**cout<<"Your ESE marks are: "<<ese<<endl;**

**}**

**};**

**int main()**

**{**

**TotalMarks s3;**

**s3.display();**

**s3.calculateTotalMarks();**

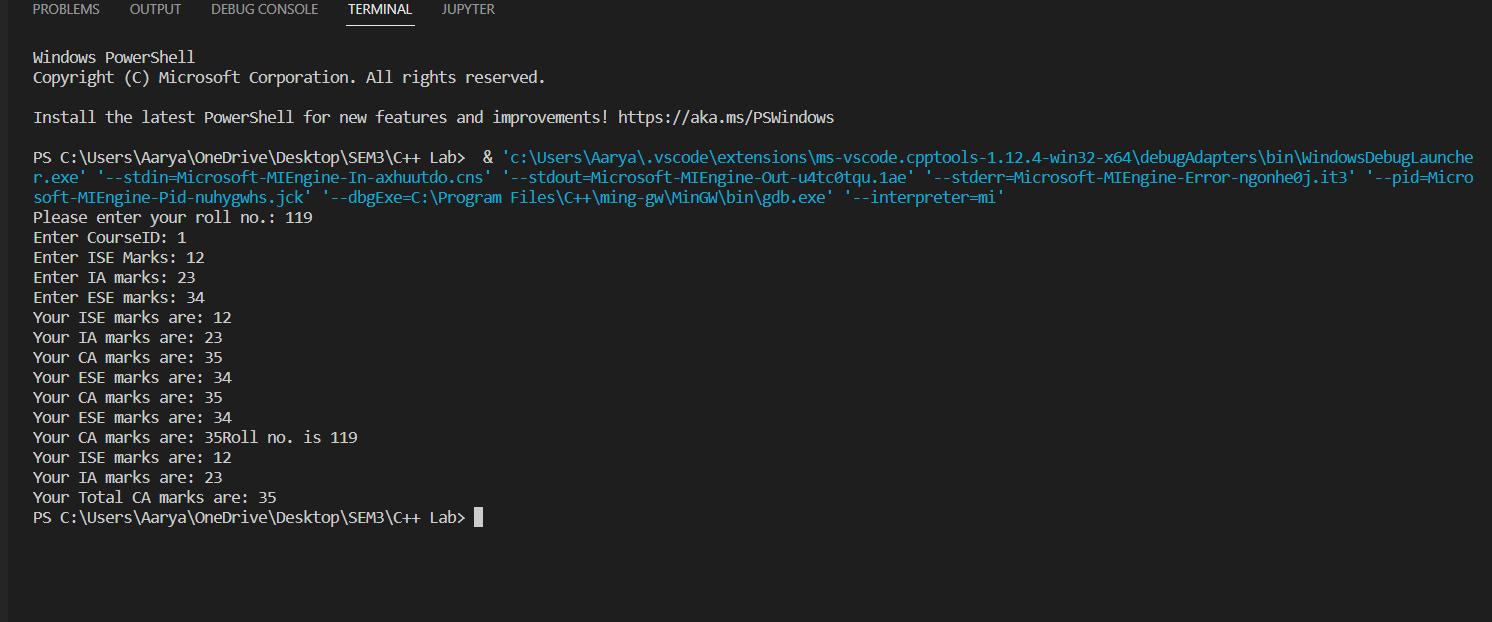
**s3.calculateCAmarks();**

**s3.Student::display();**

**s3.CAMarks::display();**

**}**

**OUTPUT:**

****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Test Cases (minimum 5 test cases required):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr. No.** | **Sample Input** | **Sample Output** | **Description** | **Test Case Type (general/special)** | **Pass/Fail** |
| **1** | **119,12,13,20** | **119,25,35** |  | **General** | **General** |
| **2** | **091,12,23,12** | **091,35,41** |  | **General** | **Pass** |
| **3** | **138,12,34,65** | **138,46,78.5** |  | **General** | **Pass** |
| **4** | **137,14,15,15** | **137,29,36.5** |  | **General** | **Pass** |
| **5** | **126,10,14,45** | **126,24,46** |  | **General** | **Pass** |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Questions:**

1. Explain virtual base class with the help of a suitable example.

Virtual base classes are used in virtual inheritance in a way of preventing multiple “instances” of a given class appearing in an inheritance hierarchy when using multiple inheritances.

Need for Virtual Base Classes:

Consider the situation where we have one class A .This class is A is inherited by two other classes B and C. Both these class are inherited into another in a new class D as shown in figure below.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Outcomes:**

CO2. Implement the principles of Data Abstraction Inheritance & Polymorphism.

**Conclusion: (Conclusion to be based on the outcomes achieved)**

We can conclude that we have learnt about multi-level inheritance.

**Grade: AA / AB / BB / BC / CC / CD /DD**

Signature of faculty in-charge with date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**References:**

**Books/ Journals/ Websites:**

1. E Balagurusamy, Object oriented Programming with C++, Tata McGraw-Hill, 8th Edition September 2020
2. Herbert Schildt, C++: The Complete Reference, McGraw Hill Education, 4th edition, July 2017
3. Jeff Langr, Modern C++ Programming with Test-Driven Development : Code Better,Sleep Better, O′Reilly, 1st edition, November 2013
4. <https://docs.microsoft.com/en-us/cpp/cpp/?view=msvc-170>