**Lab 08**

**Class Hierarchies(Multi-Level & Multiple Inheritance)**

1. **Objectives**

To familiarize the students with class hierarchies, multiple inheritance and multilevel inheritance .

1. **Outcome**

After this lab the students should be able to differentiate between public and private

inheritance. They should learn the role of constructor and destructors in derived classes. They should be able to implement multilevel inheritance and multiple inheritance.

1. **Introduction** 
   1. **Public and private inheritance**

The public keyword in the inheritance syntax means that publicly accessible members inherited from the base class stay publicly accessible in the derived class. But for private keyword in the inheritance syntax means that accessible members inherited from the base become private members of derived class.

* 1. **Constructors and Destructors in Derived Classes**

Derived classes do not inherit constructors or destructors from their base classes, but they do call the constructor and destructor of base classes. When an object of a derived class is created, the base class’s constructor is executed first, followed by the derived class’s constructor. When an object of a derived class is destroyed, its destructor is called first, then that of the base class.

* 1. **Multiple inheritance and Multi-level inheritance**

Classes can be derived from other classes that are already derived from other classes. This creates multi-level Inheritance

**Syntax:**

**class A**

**{ public: int x;**

**};**

**class B : public A**

**{ public:**

**int y;**

**};**

**class C : public B { };**

A Class can be derived from more than one base class. This is called *Multiple inheritance.*

**Syntax:**

**class A**

**{ }; class B**

**{ };**

**class C: public A, public B**

**{ };**

1. **Examples** 
   1. **This example demonstrates the Public and Private Inheritance between the classes.**

#include <iostream> **class base**

**{**

|  |  |  |  |
| --- | --- | --- | --- |
| **private:** | |  |  |
| **int pridataA; protected:** | |  | **//private data member** |
| **int prodataA; public:** | |  | **//protected data member** |
| **int pubdataA;**  **};** | |  | **//public data member** |
| **class derived1: public base**  **{**  **public:**  **void funct()**  **{ int a;** | |  | **//publically - derived class** |
| **a=pridataA;** | |  | **//error: not accessible** |
| **a=prodataA;** | |  | **//ok** |
| **a=pubdataA;**  **} };** | |  | **//ok** |
| **class derived2: private base** | |  | **//privately - derived class** |
| **{**  **public:**  **void funct()**  **{ int a;** |  |
| **a=pridataA;** |  |  | **//error: not accessible** |
| **a=prodataA;** |  |  | **//ok** |
| **a=pubdataA;**  **} };**  **int main()**  **{ int a; derived1 objd1;** |  |  | **//ok** |
| **a=objd1.pridataA;** |  |  | **//error: not accessible** |
| **a= objd1.prodataA;** |  |  | **// error: not accessible** |
| **a= objd1.pubdataA; derived2 objd2;** |  |  | **//ok (base public to derived1)** |
| **a=objd2.pridataA;** |  |  | **//error: not accessible** |
| **a=objd2.prodataA;** |  |  | **// error: not accessible** |

**a=objd2.pubdataA; // error: not accessible (base private to derived2) return 0;**

**}**

* 1. **This example will explain the process of multiple inheritance.**

**Example 1:**

#include<iostream>

#include<conio.h>

using namespace std;

**class student**

**{ protected:**

**int rno,m1,m2;**

**public:**

**void get()**

**{**

**cout<<"Enter the Roll no :";**

**cin>>rno;**

**cout<<"Enter the two marks :";**

**cin>>m1>>m2;**

**}**

**};**

**class sports**

**{ protected:**

**int sm; *// sm = Sports mark***

**public:**

**void getsm()**

**{**

**cout<<"\nEnter the sports mark :";**

**cin>>sm;**

**}**

**};**

**class statement:public student,public sports**

**{ private:**

**int tot,avg; public:**

**void display()**

**{**

**tot=(m1+m2+sm);**

**avg=tot/3;**

**cout<<"\n\n\tRoll No : "<<rno<<"\n\tTotal \t:"<<tot; cout<<"\n\tAverage : "<<avg;**

**}**

**};**

**int main()**

**{**

**statement obj; obj.get(); obj.getsm(); obj.display(); getch();**

**}**

**4.3.This example demonstrates the concept of multilevel inheritance between the classes.**

**#include<iostream>**

**using namespace std;**

**class base1**

**{**

**protected: int number1; public:**

**void showA()**

**{**

**cout<<"enter a value for first number:"<<endl;**

**cin>>number1;**

**} };**

**class base2:public base1**

**{ protected:**

**int number2; public:**

**void showB()**

**{**

**cout<<"enter value for second number:"; cin>>number2;**

**}**

**};**

**class derive: public base2**

**{**

**public:**

**void showC()**

**{**

**showA(); *//accessing base1's function***

**showB(); *//accessing base2's function***

**cout<<"number1\*number2 ="<<number1\*number2;**

**}**

**};**

**int main()**

**{ derive ob1; ob1.showC(); cout<<endl; system("pause");**

**}**

1. **Lab Tasks** 
   1. Create a class Person having *name, age* and *gender* as its data members. Create another class Department which has *DepartmentName* and *ProgramName* as it data members. Derive a class Student from class Person and class Department which has student *ID*,*grade* and *number of courses* as its member variables.
      1. Write set and get functions to enter and display the data members.
      2. Write main function to implement these classes. Enter the student data to show multiple inheritance.
   2. Design a class named Employee. The class should keep the following information in

•Employee name

•Employee number

•Hire date

Write one or more constructors and the appropriate accessor and mutator functions for the class. Next, write a class named ProductionWorker that is derived from the Employee class. The ProductionWorker class should have member variables to hold the following information:

•Shift (an integer)

•Hourly pay rate (a double )

The workday is divided into two shifts: day and night. The shift variable will hold an integer value representing the shift that the employee works. The day shift is shift 1, and the night shift is shift 2. For night shift hourly rate will be doubled, write down a function called salary to calculate the total salary of worker. Write one or more constructors and the appropriate accessor and mutator functions for the class. Demonstrate the classes by writing a program that uses a ProductionWorker object, ask the user for how many workers he wants to store the data and then display the recorded data.

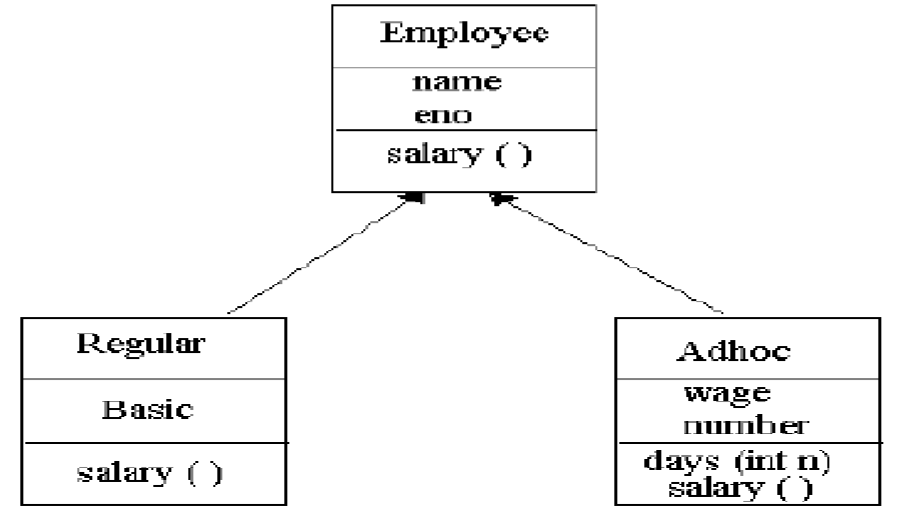
* 1. **Consider the following classes**

Create a class Date having day, month & year as its data members**.** Create another class called Time with its data members as hours, minutes & seconds. Write down the following functions for both classes:

1. void display(); // to displays the data
2. get() function // to accesses the data members
3. void set(); // to sets the values of data members

Define a class DateandTime from above two classes which displays both date and time.

1. Define an instance object of class DateTime called Watch.
2. Write a main () function that would initialize the values through the constructor functions, and then allows them to be reset through the set () functions. Be sure and display the results following the constructor before you use the set functions.
3. Through the use of the display () function, the time and date are to be displayed. Note that the display () functions in all three classes need to be defined, as well as the constructor and all the access functions.
4. **Home Tasks** 
   1. An organization has two types of employees: regular and adhoc. Regular employees get a salary which is basic + DA + HRA where DA is 10% of basic and HRA is 30% of basic.Adhoc employees are daily wagers who get a salary which is equal to Number \* Wage.
      1. Define the classes shown in the following class hierarchy diagram:



* + 1. Define the constructors. When a regular employee is created, basic must be a parameter.

When adhoc employee is created wage must be a parameter.

* + 1. Define the destructors.
    2. Define the member functions for each class. The member function days ( ) updates number of the Adhoc employee.
    3. Write a test program to test the classes.
  1. Write a class LocalPhone that contains an attribute *phone* to store a local telephone number. The class contains member functions to input and display phone number. Write a child class NatPhone for national phone numbers that inherits LocPhone class. It additionally contains an attribute to store *city code*. It also contains member functions to input and show the city code. Write another class IntPhone for international phone numbers that inherit NatPhone class. It additionally contains an attribute to store *country code*. It also contains member functions to input and show the country code. Test these classes from main() by creating objects of derived classes and testing functions in a way that clear concept of multi-level Inheritance.