Lab Manual

CS112L – Object Oriented Programming Lab
Lab No: **11**
Topic: **Inheritance**

Class: **BSGM**
Semester: **II**
Session: **Spring, 2022**
Instructor: **Ms. Saira Qamar**

Lab Date: **May 10h, 2022**
Lab Time: **11:40hrs – 2:40hrs**

Air University Islamabad
FACULTY OF COMPUTING & ARTIFICIAL INTELLIGENCE
Faculty of Computing and AI

Instructions

Submission: Use proper naming convention for your submission file. Name the submission file as LabNO_ROLLNUM (e.g. Lab01_00000). Submit the file on Google Classroom within the deadline. Failure to submit according to the above format would result in deduction of 10% marks. Submissions on the email will not be accepted.

Plagiarism: Plagiarism cases will be dealt with strictly. If found plagiarized, both the involved parties will be awarded zero marks in the assignment, all of the remaining assignments, or even an F grade in the course. Copying from the internet is the easiest way to get caught!

Deadline: The deadlines to submit the assignment are hard. Late submission with marks deduction will be accepted according to the course policy shared by the instructor. Correct and timely submission of the assignment is the responsibility of every student; hence no relaxation will be given to anyone.

Comments: Comment your code properly. Bonus marks (maximum 10%) will be awarded to well comment code. Write your name and roll number (as a block comment) at the beginning of the solution to each problem.

Tip: For timely completion of the assignment, start as early as possible. Furthermore, work smartly - as some of the problems can be solved using smarter logic.

- Note: Follow the given instructions to the letter, failing to do so will result in a zero.

Objectives

In this lab, you will learn about type Conversion and following:

- Inheritance in Classes
- Multiple Inheritance
- Multilevel Inheritance
- Hybrid Inheritance
- Accessing Base Class Members
- Protected Members
- Allocation in Memory
- Sequence of Constructor and Destructor

Concepts

1. Inheritance in Classes

Inheritance is one of the core features of an object-oriented programming language. It allows software developers to derive a new class from the existing class. The derived class inherits the features of the base class (existing class). There are various types of inheritance in OOP C++ programming which will be discussed in later Sections. The simple example of Inheritance:

```
//Base class
class Parent
{
    public:
      int id_p;
};
// Sub class inheriting from Base Class(Parent)
class Child : public Parent
{
    public:
      int id_c;
};
//main function
int main()
{
    Child ch_obj;
    // An object of class child has all data members
    // and member functions of class parent
    ch_obj.id_c = 7;
    ch_obj.id_p = 91;
    cout << "Child id is " <<  ch_obj.id_c << endl;
    cout << "Parent id is " <<  ch_obj.id_p << endl;
    return 0;
}
```

Output:

```
Child id is 7
Parent id is 91
```

2. Multi-Level Inheritance:

In C++ programming, not only you can derive a class from the base class but you can also derive a class from the derived class. This form of inheritance is known as **multilevel** inheritance.

```
class A {
... .. ... };

class B: public A {
... .. ... };

class C: public B {
... ... ... };
```

The following is an **example** of Multi-level Inheritance:

```
class A {
    public:
      void display()
      {
          cout<<"Base class content.";
      }
};

class B : public A {};

class C : public B {};

int main() {
    C obj;
    obj.display();
    return 0;
}
```

Output:

```
Base class content.
--------------------------------
```

3. Multiple Inheritance:

Multiple Inheritance is a feature of OOP where a class can inherit from more than one classes i.e. one **sub class** is inherited from more than one **base classes**. The following is syntax to demonstrate the concept behind:

```
class subclass_name : access_mode base_class1, access_mode base_class2, ....
{
//body of subclass
};
```

The following is an **example** of Multiple Inheritance:

```
// first base class
class Vehicle {
  public:
    Vehicle()
    { cout << "This is a Vehicle" << endl; }
};
// second base class
class FourWheeler {
  public:
    FourWheeler()
    { cout << "This is a 4 wheeler Vehicle" << endl; }
};
// sub class derived from two base classes
class Car: public Vehicle, public FourWheeler {
};
// main function
int main()
{
    // creating object of sub class will
    // invoke the constructor of base classes
    Car obj;
    return 0;
}
```

Output:

```
This is a Vehicle
This is a 4 wheeler Vehicle
```

4. Hybrid Inheritance:

In this type of inheritance, **more than one sub class is inherited from a single base class**. i.e. more than one derived class is created from a single base class. The following is an example:

```
// base class
class Vehicle
{
  public:
    Vehicle()
    { cout << "This is a Vehicle" << endl; }
};
//base class
class Fare
{
    public:
    Fare()
    { cout<<"Fare of Vehicle\n";    }
};
// first sub class
class Car: public Vehicle
{  };
// second sub class
class Bus: public Vehicle, public Fare
{  };
int main()
{
    // creating object of sub class will
    // invoke the constructor of base class
    Bus obj2;
    return 0;
}
```

Output:

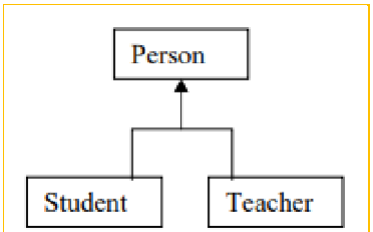
```
This is a Vehicle
Fare of Vehicle

--------------------------------
```

5. Accessing Base Class Members:

Public members of **base class** become public member of **derived class**. **Private members** of **base class** are not accessible from outside of **base class**, even in the derived class (Information Hiding)

Example:

In the code given below Student and Teacher classes has been **derived** from single Person class.




```
class Person {
        string name;
        int age;
        /*...*/
public:
        string GetName() const;
        int GetAge() const;
        /*...*/
};
```

```
class Teacher : public
        Person {
        string dept;
        int course;
        /*...*/
public:
        string GetDept() const;
        int GetCourse() const;
        void Print() const;
        /*...*/
};
```

```
class Student : public
        Person {
        int semester;
        int rollNo;
        /*...*/
public:
        int GetSemester() const;
        int GetRollNo() const;
        void Print() const;
        /*...*/
};
```

```
void Student::Print()
{
        cout << name << " is in" << " semester " << semester;
}
//corrected Code :
void Student::Print()
{
        cout << GetName() << " is in semester " << semester;
}


int main() {
        Student stdt;
        stdt.semester = 0;//error
        stdt.name = NULL; //error
        cout << stdt.GetSemester();
        cout << stdt.GetName();
```

```
        return 0;
}
```

6. Protected Members:

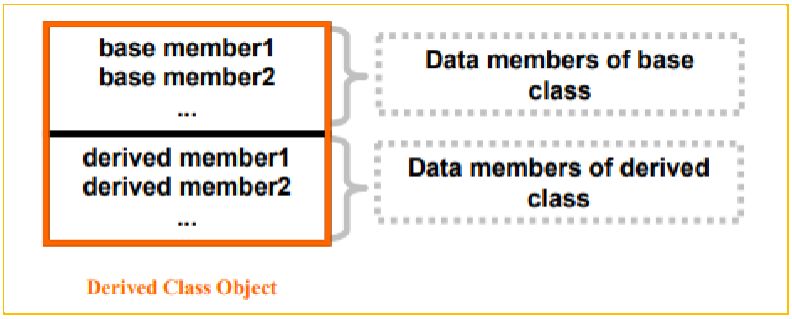
Protected access specifier ensures that member(s) of **base class** is accessible in **derived class** and NOT outside of this class. **Protected members** of a class can not be accessed outside the class but only in the derived class of that class. Protected members of base class become protected members of derived class.

Example:

```
class Date {
        /* . . . */
protected:
        bool IsLeapYear(int);
};
class SpecialDate : public Date {
        /* . . . */
public:
        void AddSpecialYear(int i) {
                /* . . . */
                        if (day == 29 && month == 2
                                && !IsLeapyear(year + i)) { //ERROR!
                        }
        }
};
int main() {
        Date aDate;
        aDate.IsLeapYear(); //Error
        return 0;
}
```

7. Allocation in Memory:

The object of **derived class** is represented in memory as follows:



Every object of **derived class** has an anonymous object of **base class**.

8. Sequence of Constructor/Destructor:

The anonymous object of **base class** must be initialized using constructor of **base class.** When a **derived class** object is created the **constructor** of **base class** is **executed before** the **constructor** of **derived class**.

Example:

```
class Parent {
public:
        Parent() {
                cout <<
                        "Parent Constructor...";
        }
};
class Child : public Parent {
public:
        Child() {
                cout <<
                        "Child Constructor...";
        }
};

int main() {
        Child cobj;
        return 0;
}
```

Output:

```
Parent Constructor...
Child Constructor...
```

Additional Information to Consider:

- If default constructor of base class does not exist then the compiler will try to generate a default constructor for base class and execute it before executing constructor of derived class.
- If the user has given only an overloaded constructor for base class, the compiler will not generate default constructor for base class and result in ERROR in object is created with default constructor.

Lab Tasks

1. (Simple Inheritance) Write a C++ program to design a base class Person (name, address, phone_no). Derive a class Employee (eno, ename) from Person. Derive a class Manager (designation, department name, and basic-salary) from Employee. Write a menu driven program to:
   - Accept all details of 'n' managers.
   - Display manager having highest salary.

2. (Multiple Inheritance) Write a C++ program to read and print employee information using multiple inheritance.

3. PakWheel required a system that manages their vehicles. Vehicles are of two types: Car and Truck. For the Car extra feature is “Sitting Capacity” and for Truck it is Loading capacity in weight. Each Vehicle contains clustered Vehicle Body Feature (Color, Type and Material). (Implement this problem using hybrid inheritance). Use constructor for Initialization.

4. (Multi-level Inheritance) Write a C++ program to calculate the percentage of a student using multi-level inheritance. Accept the marks of three subjects in base class. A class will be derived from the above mentioned class which includes a function to find the total marks obtained and another class derived from this class which calculates and displays the percentage of the student.

5. You are creating an Animal System for a Zoo in Islamabad. You are required to keep all kinds of information related to all animals.

   Create a base Animal Class containing basic animal detail (petName, Specie Name, Weight, Food) and child class (named on Animal’s Name) should contain custom animal detail in parameters. For Instance, custom attributes of Snake and Lion are given below.
   - Snake (Length, IsPoisons)
   - Lion (NumberFeet, Speed, IsDangerous)
   - Eagle (-------- Add 3 Custom Features --------)