
National University
of computer and emerging sciences

# CL-1004
# Object Oriented Programming- Lab
# Spring' 2023
# BS-SE

# Lab Manual 10

## Understanding Access Specifier's Working in Inheritance

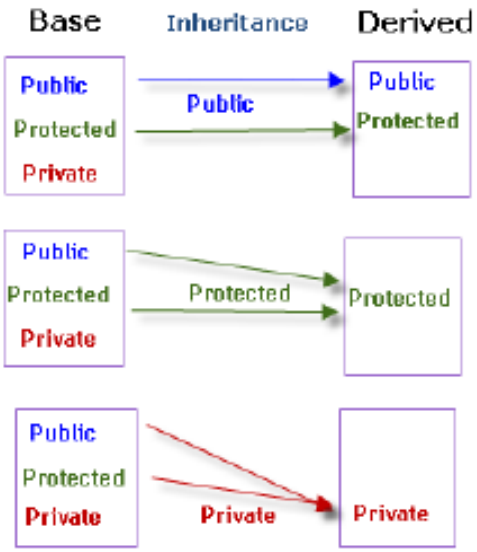


Fig: **Visibility after inheritance**

```
class Base {
  public:
    int x;
  protected:
    int y;
  private:
    int z;
};

class PublicDerived: public Base {
  // x is public
  // y is protected
  // z is not accessible from PublicDerived
};

class ProtectedDerived: protected Base {
  // x is protected
  // y is protected
  // z is not accessible from ProtectedDerived
};

class PrivateDerived: private Base {
  // x is private
  // y is private
  // z is not accessible from PrivateDerived
};
```

## Example: Multiple VS Multi-Level Inheritance

```
class A{
    public:
        int x;
        void getx(){
            cout << "Enter value of x: ";
            cin >> x;
        }
};
class B{
    public:
        int y;
        void gety(){
            cout << "Enter value of y: ";
            cin >> y;
        }
};
class C : public A, public B{
    public:
        void sum(){
            cout << "Sum = " << x + y;
        }
};
int main(){
    C obj;
    obj.getx();
    obj.gety();
    obj.sum();
}
```

```
class A{
    public:
        int x;
        void getx(){
            cout << "Enter value of x: ";
            cin >> x;
        }
};
class B: public A{
    public:
        int y;
        void gety(){
            cout << "Enter value of y: ";
            cin >> y;
        }
};
class C : public B{
    public:
        void sum(){
            cout << "Sum = " << x + y;
        }
};
int main(){
    C obj;
    obj.getx();
    obj.gety();
    obj.sum();
}
```

# Problem 1:

We want to store the information of different vehicles. Create a class named Vehicle with two data member named

- mileage (double)
- Price (double).

  Create its two subclasses

  - Car with data members to store ownership
    - cost,
    - warranty (by years)
    - seating capacity
    - fuel type.
  - Bike with data members to store the
    - number of cylinders
    - number of gears
    - cooling type(air, liquid or oil)
    - wheel type(alloys or spokes)
    - fuel tank size(in inches)

Make another two subclasses Audi and Ford of Car, each having a data member to store the model type.

Next, make two subclasses Bajaj and TVS, each having a data member to store the make-type. Now, store and print the information of an Audi and a Ford car (i.e. model type, ownership cost, warranty, seating capacity, fuel type, mileage and price.) Do the same for a Bajaj and a TVS bike.

# Problem 2:

In this task, we would try to find out areas of various shapes using inheritance. You need to create a class name ‘Base’. Define two protected data members of float type for length and width – ‘Length or Base’ and ‘Width or Height’. Now you need to define constructor(s), setter(s) and a getter for the base class.

Next, you need to define three child classes, ‘Rectangle’, ‘Square’ and ‘Triangle’. Every class must contain a constructor that invokes the parent class constructor whenever a child instance is created. Following the same approach, create the particular getter, setter(s) and area functions for each of the class. Area of a class must return a value accordingly. Use a suitable inheritance.

```
class Base
    • Base(float, float){}
    • void baseGetter() const{}
class Rectangle
    • Rectangle(float, float);
    • void rectangleGetter(){}
    • void rectangleSetter(){}
    • void rectangleSetter(float, float){}
    • float areaOfRectangle(){}
class Square
    • Square(float, float);
    • void squareGetter(){}
    • void squareSetter(){}
    • void squareSetter(float, float){}
    • float areaOfSquare(){}
class Triangle
    • Triangle(float, float);
    • void triangleGetter(){}
    • void triangleSetter(){}
    • void triangleSetter(float, float){}
    • float areaOfTriangle(){}
```

## Problem 3:

In this task, we would implement and learn the usage of protected and public inheritance using one parent class ‘Staff’ and two child classes ‘Faculty’ and ‘NonFaculty’. ‘Faculty’ would use protected inheritance with ‘Staff’ and ‘NonFaculty’ would use public inheritance with ‘Staff’. Create the staff class and include the following protected data members in it – name, cnic, dob(used for year of birth only) and contact. Define the required constructor(s), getters and setter(s) – define a separate getter for each of the data member.

- **void staffGetter();**
- **char* getName();**
- **char* getCnic();**
- **short getDob();**
- **short getAge();**
- **unsigned int getContact();**

Create the child class ‘Faculty’ first and include the following data members as its protected data members – payScale, roomNumber, subject and rank. Define the necessary constructor(s), getters and setter(s) – define a separate getter for each of the data member.

- **void facultyGetter();**
- **short getPayScale();**
- **short getRoomNumber();**
- **char* getSubject();**
- **char* getRank():**

Create the second child class ‘NonFaculty’ and include the following data members as its protected members – duty, startTiming and endTiming. Do the same, define constructors, getters and setters – define a separate getter for each of the data member.

- **void nonFacultyGetter();**
- **char* getDuty();**
- **char* getStartTiming();**
- **char* getEndTiming();**