Inheritance

The mechanism that allows us to extend the definition of a class without making any physical changes to the existing class is inheritance.

Inheritance lets you create new classes from existing class. Any new class that you create from an existing class is called **derived class**; existing class is called **base class**.

The inheritance relationship enables a derived class to inherit features from its base class. Furthermore, the derived class can add new features of its own. Therefore, rather than create completely new classes from scratch, you can take advantage of inheritance and reduce software complexity.

Forms of Inheritance

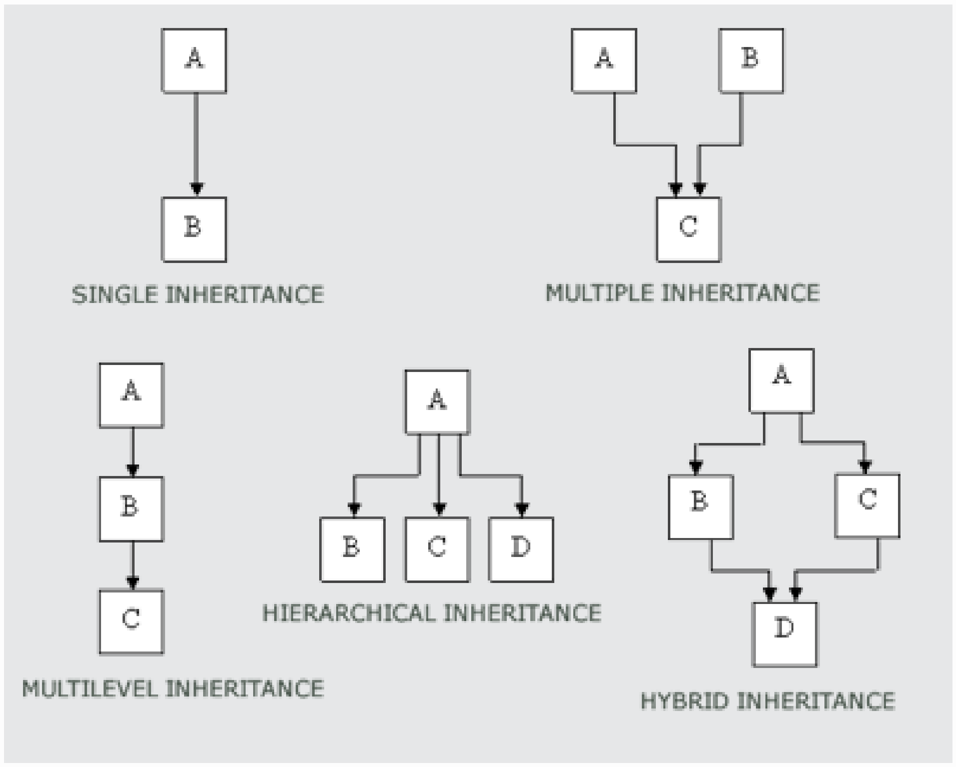
Single Inheritance: It is the inheritance hierarchy wherein one derived class inherits from one base class.

Multiple Inheritance: It is the inheritance hierarchy wherein one derived class inherits from multiple base class(es)

Hierarchical Inheritance: It is the inheritance hierarchy wherein multiple subclasses inherit from one base class.

Multilevel Inheritance: It is the inheritance hierarchy wherein subclass acts as a base class for other classes.

Hybrid Inheritance: The inheritance hierarchy that reflects any legal combination of other four types of inheritance.



In order to derive a class from another, we use a colon (:) in the declaration of the derived class using the following format :

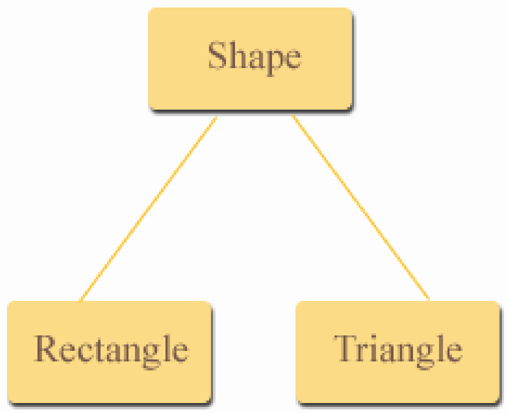
```
class derived_class: memberAccessSpecifier base_class
{
        ...
};
```

Where derived_class is the name of the derived class and base_class is the name of the class on which it is based. The member Access Specifier may be public, protected or private. This access specifier describes the access level for the members that are inherited from the base class.

Member Access Specifier	**How Members of the Base Class Appear in the Derived Class**
Private	Private members of the base class are inaccessible to the derived class.
	Protected members of the base class become private members of the derived class.
	Public members of the base class become private members of the derived class.
Protected	Private members of the base class are inaccessible to the derived class.
	Protected members of the base class become protected members of the derived class.
	Public members of the base class become protected members of the derived class.
Public	Private members of the base class are inaccessible to the derived class.
	Protected members of the base class become protected members of the derived class.
	Public members of the base class become public members of the derived class.

In principle, a derived class inherits every member of a base class except constructor and destructor. It means private members are also become members of derived class. But they are inaccessible by the members of derived class.

Following example further explains concept of inheritance :



```
class Shape
{
protected:
       float width, height;
public:
       void set_data (float a, float b)
       {
              width = a;
              height = b;
       }
};

class Rectangle: public Shape
{
public:
       float area ()
       {
              return (width * height);
       }
};

class Triangle: public Shape
{
public:
       float area ()
       {
              return (width * height / 2);
       }
};
```

```
int main ()
{
       Rectangle rect;
       Triangle tri;
       rect.set_data (5,3);
       tri.set_data (2,5);
       cout << rect.area() << endl;
       cout << tri.area() << endl;
       return 0;
}
```

output :

```
15
5
```

The object of the class Rectangle contains :
width, height inherited from Shape becomes the protected member of Rectangle.

set_data() inherited from Shape becomes the public member of Rectangle
area is Rectangle's own public member

The object of the class Triangle contains :
width, height inherited from Shape becomes the protected member of Triangle.
set_data() inherited from Shape becomes the public member of Triangle
area is Triangle's own public member

set_data () and area() are public members of derived class and can be accessed from outside class i.e. from main()