# Week 8: Lecture 16

## Inheritance in C++

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.

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

```
class deived_class_name : access_mode base_class_name
{
  //body of subclass
};
```

Where deived_class_name is the name of the derived class and base_class_name 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.

**Modes of Inheritance**

1. **Public mode**: If we derive a sub class from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class.

2. **Protected mode**: If we derive a sub class from a Protected base class. Then both public member and protected members of the base class will become protected in derived class.

3. **Private mode**: If we derive a sub class from a Private base class. Then both public member and protected members of the base class will become Private in derived class.

**Note :** The private members in the base class cannot be directly accessed in the derived class, while protected members can be directly accessed. For example, Classes B, C and D all contain the variables x, y and z in below example. It is just question of access.

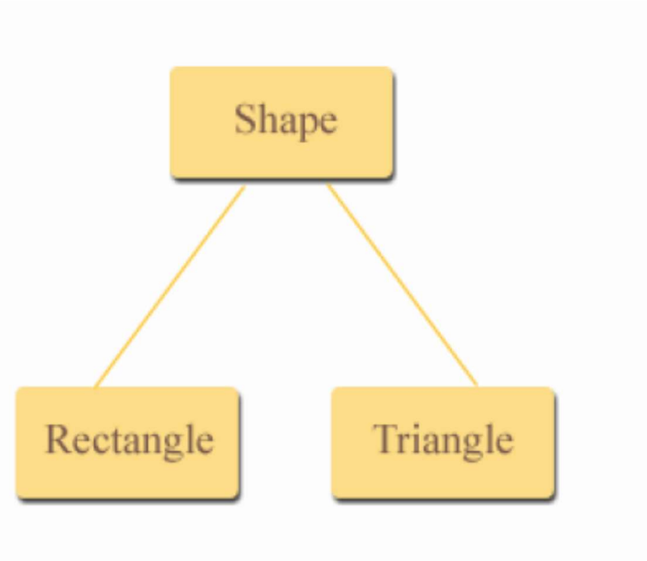
The below table summarizes the above three modes and shows the access specifier of the members of base class in the sub class when derived in public, protected and private modes:

| Base class member access specifier | Type of Inheritence | | |
|---|---|---|---|
| | Public | Protected | Private |
| Public | Public | Protected | Private |
| Protected | Protected | Protected | Private |
| Private | Not accessible (Hidden) | Not accessible (Hidden) | Not accessible (Hidden) |

```
// C++ Implementation to show that a derived class
// doesn’t inherit access to private data members.
// However, it does inherit a full parent object
class A
{
public:
    int x;
protected:
    int y;
private:
    int z;
};

class B : public A
{
    // x is public
    // y is protected
    // z is not accessible from B
};

class C : protected A
{
    // x is protected
    // y is protected
    // z is not accessible from C
};

class D : private A    // 'private' is default for classes
{
    // x is private
    // y is private
    // z is not accessible from D
};
```

*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().