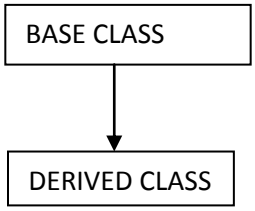
What is Inheritance ?

Another important feature of Object-oriented programming is inheritance. Inheritance allows us to create a class from another class.Using inheritance its possible to reuse the code functionality of one class in another.

A new (derived) class, can use the data members and member functions of the base class.The new class can access the functions of the base class and it can also add some functions in its .The derived class can use functions of base class,we will have a concept of reusability through inheritance. The new class should inherit the members of an existing class.

This existing class from which the new class is derived is called the **base** class, and the new class is referred to as the **derived** class.

The idea of inheritance implements the **is a** relationship.



Base class can also be called as super class and derived class is called as sub class.

Syntax for deriving a class is

derived class name : access specifier base class name

e.g.

class B : public class A

where B is derived class and A is base class.

Where access-specifier can be **public, protected,** or **private**

Following table shows different access specifers and their accesses to the class /derived class and outside the class

Access	**public**	**protected**	**private**
Same class	yes	yes	yes
Derived classes	yes	yes	no
Outside classes	yes	no	no

Type of Inheritance:

When deriving a class from a base class, the base class may be inherited through **public, protected** or**private** inheritance.

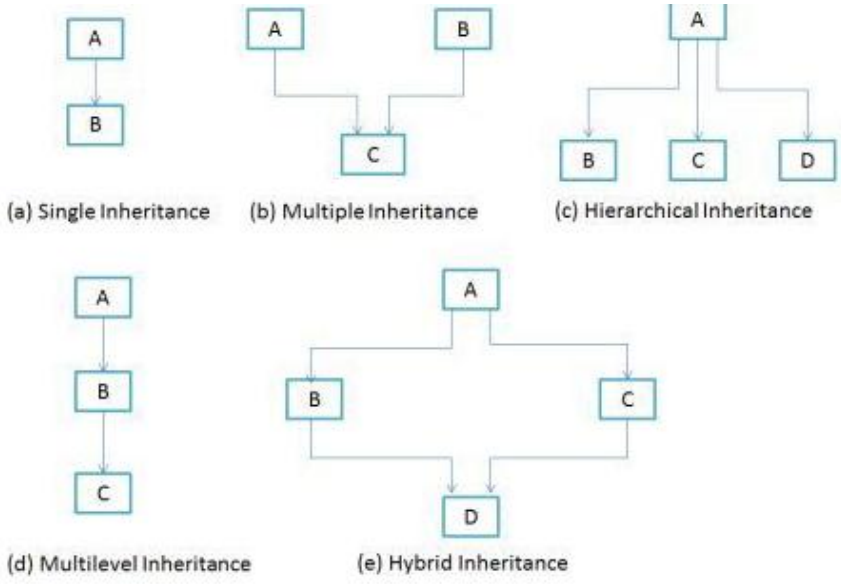
a. **Public Inheritance:** When deriving a class from a **public** base class, **public** members of the base class become **public** members of the derived class and **protected** members of the base class become **protected** members of the derived class. A base class's **private** members are never accessible directly from a derived class.
b. **Protected Inheritance:** When deriving from a **protected** base class, **public** and **protected**members of the base class become **protected** members of the derived class.
c. **Private Inheritance:** When deriving from a **private** base class, **public** and **protected** members of the base class become **private** members of the derived class.

We hardly use **protected** or **private** inheritance, but **public** inheritance is commonly used.

A class can be derived from one or more number of base classes

Types of Inheritance those are provided by C++ are as followings:

a. Single Inheritance
b. Multilevel Inheritance
c. Multiple Inheritance
d. Hierarchical Inheritance
e. Hybrid Inheritance



NOTE :-

A derived class /sub class can inherits all base class functions except the following:

1. Constructors, destructors and copy constructors of the base class.
2. Overloaded operators of the base class.
3. The friend functions of the base class.

Example

```
class A
{
Private: int a;
public:
fun1()
{
cout<<"\n this is class A";
}
};
class B:public class A
{
private: int b;
Public:
fun2()
{
cout<<"\n this is class B";
}
};
```

```
int main()
{
B b1;
b1.fun1();
b1.fun2();
return 0;
}
```

Output: this is class A
this is class B

Member Function Overriding in Inheritance

If a base class and derived class can have member function. If an object is created of derived class and if a member function is called then the member function in derived class is only invoked, i.e., the member function of derived class overrides the member function of base class.

Please refer to Virtual Functions topic

Try to write the following programs using inheritance
e.g

1. Find area of different objects
2. Calculate salary of 5 employees using single level inheritance
3. Program to print marksheet using multilevel inheritance
4. Program to find the total by adding sports marks using multiple inheritance