C++ Inheritance

Inheritance is a process in which one object acquires all the properties and behaviours of its parent object automatically. In this way, we can reuse, extend, modify the attributes and behaviors which are defined in other class.

→ In C++, class which inherits the members of another class is called derived class.

→ class whose members are inherited is called base class.

→ Derived class is specialised class for base class.

Advantage of C++ inheritance

↳ Code Reusability → We can reuse the members of an already defined class

Types of Inheritance

C++ supports five types of inheritance:

(a) Single inheritance (b) Multiple inheritance
(c) Hierarchical Inheritance (d) Multilevel inheritance
(e) Hybrid inheritance

→ Derived class syntax

```
class derived_class_name :: visibility mode  base_class_name
{
    // body
}
```

visibility mode - This specifies whether features of the base class are publicly inherited or privately inherited. It can be public or private

→ When the base class is privately inherited by the derived class, public members of the base class become private members of derived class. Thus, public members of the base class are not accessible by the objects of the derived class but only by the member functions of derived class.

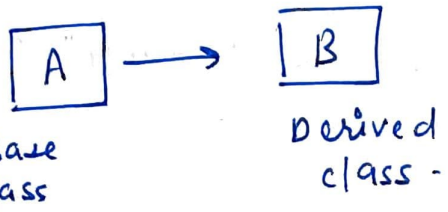
→ When the base class is publicly inherited by the derived class, public members of the base class also become the public members of the derived class Thus, public members of the base class are accessible by the objects of the derived class as well as by the member functions of the base class

Note :- In C++, default mode of visibility is private

- The private members of the base class are never inherited.

⇒ Single inheritance

It is defined as the inheritance in which a derived class is inherited from only one base class.



ex.

```
class Account {
        public:
        float salary = 60000;
    };
class Programmer : public Account {
        public:
            float bonus = 5000;
    };
int main(){
        Programmer p1;
        cout << "Salary" << p1.salary << endl;
        cout << "Bonus" << p1.bonus << endl;
}.
```

Output

Salary 60000
Bonus 5000

} salary member was inherited from Account class.

→ Inheriting methods (single level inheritance)

ex.

```
class Animal {
    public:
        void eat() {
            cout << "Eating" << endl;
        }
};

class Dog : public Animal {
    public:
        void bark() {
            cout << "Barking" << endl;
        }
};

int main() {
    Dog d1;
    d1.eat();
    d1.bark();
}
```

Output

Eating
Barking

ex.

```
class A {
    int a = 4;
    int b = 5;

    public:
        int mul() {
            int c = a x b;
            return c;
        }
    };

class B : private A {

    public:
        void display() {
            int result = mul();
            cout << "multiplication of a and b is " << result;
        }
    };

int main() {

    B b;
    b.display();

}
```

output

multiplication of a and b is 20.

mul() is inherited method and can access private members of base class.

As class A were privately inherited in previous example, the function mul() of class 'A' cannot be accessed by object of class B. It can only be accessed by member function of class B.

→ visibility modes

(i) Public :→ when a member is declared public, it is accessible to all functions of the program.

(ii) Private : when the member is declared as private, it is accessible within the class only.

(iii) Protected : when member is declared protected, it is accessible within its own class as well as the class immediately derived from it.

visibility of Inherited members

Base class visibility	Derived class visibility		
	Public	Private	Protected
Private	Not inherited	Not inherited	Not inherited
Protected	Protected	Private	Protected
Public	Public	Private	Protected