

**Single Inheritance:** In single inheritance, a class is allowed to inherit from only one class. i.e. one sub class is inherited by **one base class only**.

```
class subclass_name : access_mode base_class
{
    //body of subclass
};
```

**Multiple Inheritance:** Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes. i.e. one sub class is inherited from **more than one base classes**.

```
class subclass_name : access_mode base_class1, access_mode base_class2,
....
{
    //body of subclass
};
```

### 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.

```
// 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
};

#include <iostream>
using namespace std;
// base class
class Vehicle {
public:
    Vehicle()
    {
        cout << "This is a Vehicle" << endl;
    }
};
// sub class derived from two base classes
class Car: public Vehicle {
};

// main function
int main()
{
    // creating object of sub class will
    // invoke the constructor of base classes
    Car obj;
    return 0;
}

```