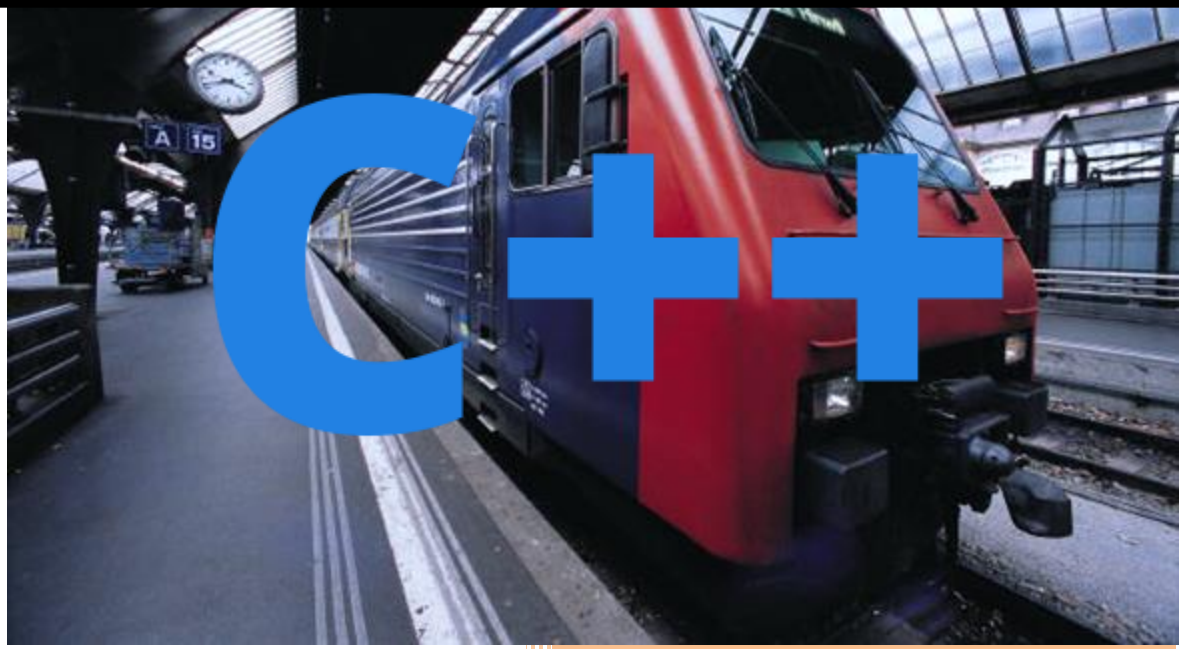
Build Your IT Skill

2020

ណែនាំស្តាប់ពី Inheritance Of OOP

Single Inheritance Multiple Inheritance



ណែនាំអោយស្គាល់ពី Inheritance

I. ដូចម្តេចទៅដែលហៅថា Inheritance?

Inheritance គឺជាដំណើរនៃការកកើត Class ថ្មីមួយចេញពី Class ដែលមានស្រាប់ ដែល Class ថ្មី គេហៅថា Sub Class ឬ Derived Class និង Class មាន ស្រាប់គេហៅថា Base Class ឬ Super Class។

ទំរង់ទូទៅ៖

```
1
2  class Super_Class{
3
4      Data Member
5
6      Function Member
7       .........
8
9  };
10
11 class Sub_Class: public/protected/private{
12
13     Data Member
14
15     Function Member
16      .........
17 };
18
19
```

នៅក្នុង Inheritance គេបែងចែកជា ២ប្រភេទគឺ៖

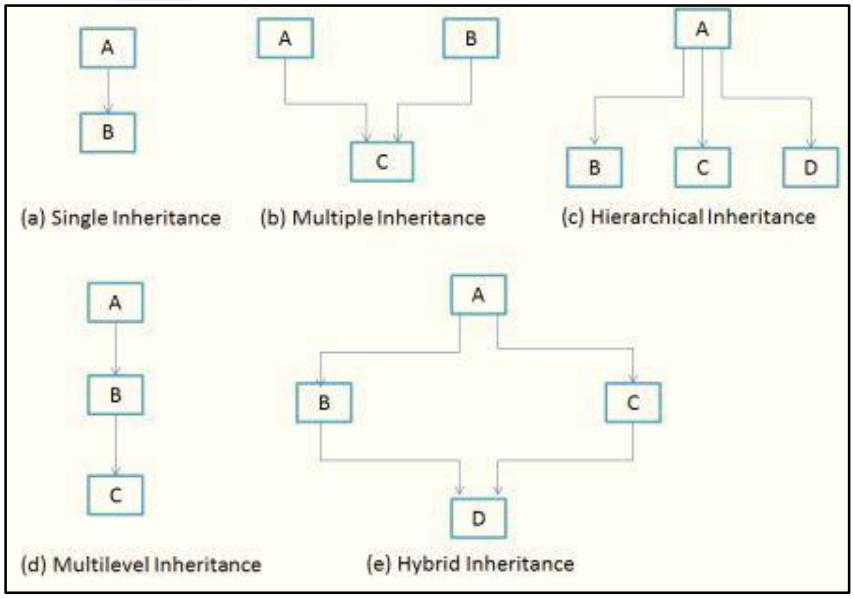
១) Single Inheritance

២) Multiple Inheritance

៣) Hierarchical

៤) Multilevel

៥) Hybrid

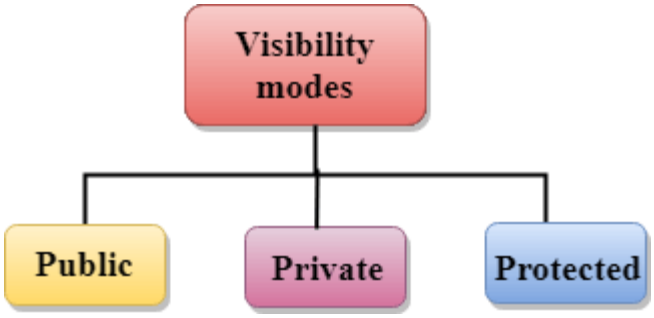


តែការបែងចែកជាចំណែកធំៗនៃ Inheritance គេបែងចែកជាពីរសំខាន់គឺ៖

១) Single Inheritance

២) Multiple Inheritance

ខាងក្រោមគឺជាកំរិតនៃ Level របស់ Inheritance៖



អ្នកត្រូវចាំថាកំរិតនៃការ Accessing Private គឺមិនអាចប្រាស់បានសំរាប់ទំរង់ទាំងអស់ នៃ Inheritance ៖

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

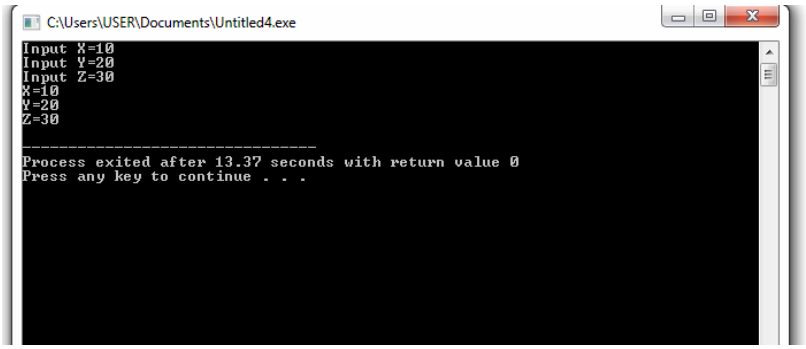
1.1. Single Inheritance: គឺជាប្រភេទ Inheritance ដែលមាន Base Class មួយ និង Sub Class អាចមានឲ ឬ ច្រើន។

ឧទាហរណ៏ ១

```
#include<iostream>
using namespace std;
class Test1{
    protected:
     int x;
     int y;
};
class Test2:public Test1{
    private:
    int z;
    public:
        void Input()
        {
            cout<<"Input X=";cin>>x;
            cout<<"Input Y=";cin>>y;
            cout<<"Input Z=";cin>>z;
        }
        void Output()
        {
            cout<<"X="<<x<<endl;
            cout<<"Y="<<y<<endl;
            cout<<"Z="<<z<<endl;
        }
};
  int main()
  {
     Test2 obj2;
     obj2.Input();
     obj2.Output();
  }
```

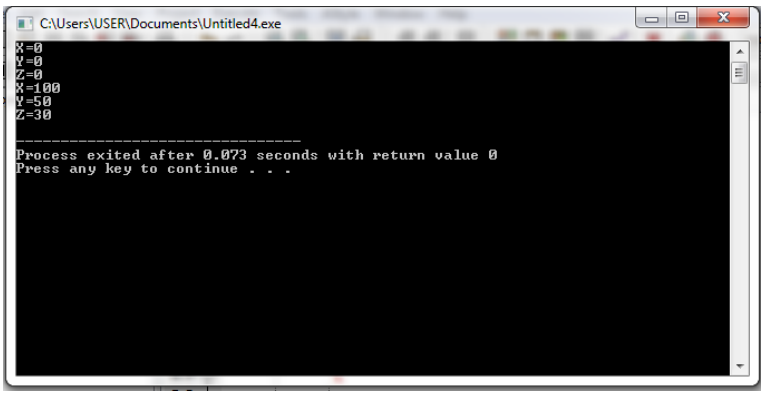
`

លទ្ធផលទទួលបាន៖



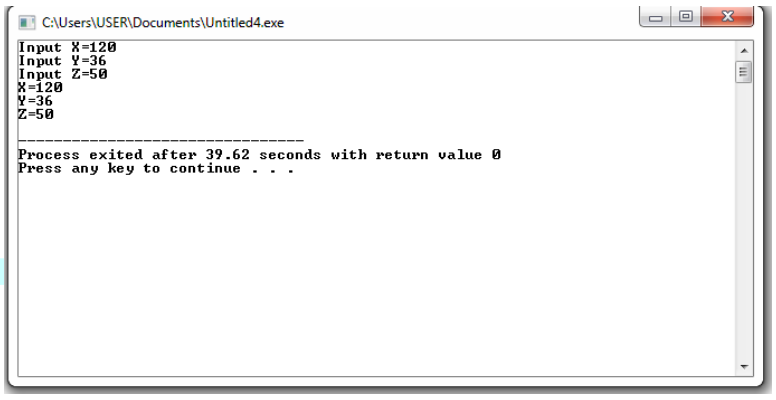
ឧទាហរណ៏ ២៖

```
#include<iostream>
using namespace std;
class Test1{
    protected:
     int x;
     int y;
     public:
      Test1()
      {
        x=0;
        y=0;
      }
      Test1(int a,int b)
      {
        x=a;
        y=b;
      }
};
class Test2:public Test1{
    private:
    int z;
    public:
        Test2()
        {
            Test1:Test1();
            z=0;
        }
        Test2(int a,int b,int c)
        {
            x=a;
            y=b;
            z=c;
        }
        void Input()
        {
            cout<<"Input X=";cin>>x;
            cout<<"Input Y=";cin>>y;
            cout<<"Input Z=";cin>>z;
        }
        void Output()
        {
            cout<<"X="<<x<<endl;
            cout<<"Y="<<y<<endl;
            cout<<"Z="<<z<<endl;
        }
};
  int main()
  {
    Test2 obj2;
    obj2.Output();
    Test2 obj3(100,50,30);
    obj3.Output();
  }
```



ឧទាហរណ៏ ៣៖

```
#include<iostream>
using namespace std;
class Test1{
    protected:
     int x;
     int y;
     public:
      Test1()
      {
        x=0;
        y=0;
      }
      Test1(int a,int b)
      {
        x=a;
        y=b;
      }
     void Input()
     {
          cout<<"Input X=";cin>>x;
          cout<<"Input Y=";cin>>y;
      }
      void Output()
      {
        cout<<"X="<<x<<endl;
        cout<<"Y="<<y<<endl;
      }
};
class Test2:public Test1{
    private:
    int z;
    public:
        Test2()
        {
          Test1:Test1();
          z=0;
        }
        Test2(int a,int b,int c)
        {
            x=a;
            y=b;
            z=c;
        }
        void Input()
        {
          Test1::Input();
          cout<<"Input Z=";cin>>z;
        }
        void Output()
        {
          Test1::Output();
          cout<<"Z="<<z<<endl;
        }
};
  int main()
  {
     Test2 obj2;
     obj2.Input();
     obj2.Output();

  }
```

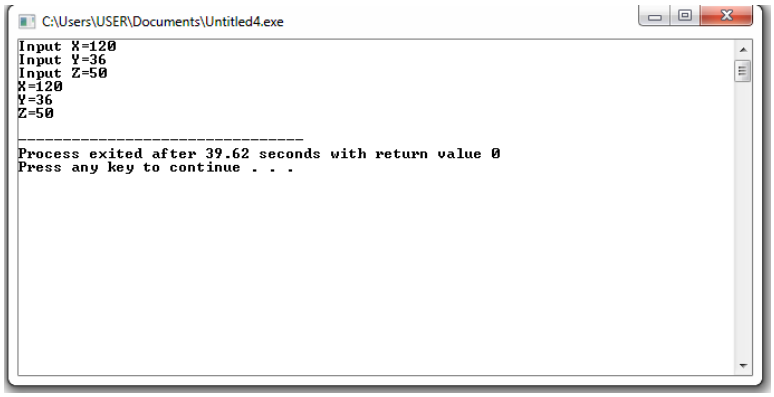


ឧទាហរណ៏ ៤៖

```
#include<iostream>
using namespace std;
class Test1{
    protected:
     int x;
     int y;
     public:
      Test1()
      {
        x=0;
        y=0;
      }
      Test1(int a,int b)
      {
        x=a;
        y=b;
      }
     void Input()
     {
            cout<<"Input X=";cin>>x;
            cout<<"Input Y=";cin>>y;
      }
      void Output()
      {
        cout<<"X="<<x<<endl;
        cout<<"Y="<<y<<endl;
      }
};
class Test2:public Test1{
    private:
    int z;
    public:
        Test2()
        {
            Test1:Test1();
            z=0;
        }
        Test2(int a,int b,int c)
        {
            x=a;
            y=b;
            z=c;
        }
        void Input()
        {
            Test1::Input();
            cout<<"Input Z=";cin>>z;
        }
        void Output()
        {
            Test1::Output();
            cout<<"Z="<<z<<endl;
        }
};
```

```
55  class Test3:private Test1{
56      private:
57       int z1;
58      public:
59        Test3()
60        {
61          Test1:Test1();
62          z1=0;
63        }
64        Test3(int a,int b,int c)
65        {
66              x=a;
67              y=b;
68              z1=c;
69        }
70          void Input()
71          {
72              Test1::Input();
73              cout<<"Input Z1=";cin>>z1;
74          }
75          void Output()
76          {
77              Test1::Output();
78              cout<<"Z1="<<z1<<endl;
79          }
80  };
81    int main()
82    {
83      Test3 obj3;
84      obj3.Output();
85      obj3.Input();
86      obj3.Output();
87    }
```

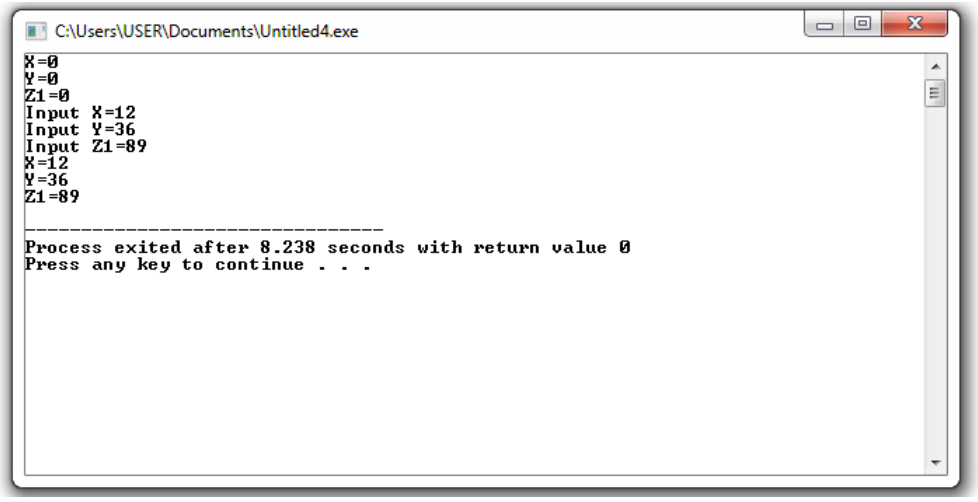
លទ្ធផលទទួលបាន៖



ឧទាហរណ៏ ៥៖

```cpp
#include<iostream>
using namespace std;
class Test1{
    protected:
     int x;
     int y;
     public:
      Test1()
      {
        x=0;
        y=0;
      }
      Test1(int a,int b)
      {
        x=a;
        y=b;
      }
     void Input()
     {
          cout<<"Input X=";cin>>x;
          cout<<"Input Y=";cin>>y;
      }
      void Output()
      {
        cout<<"X="<<x<<endl;
        cout<<"Y="<<y<<endl;
      }
};
class Test2:public Test1{
    private:
    int z;
    public:
        Test2()
        {
            Test1:Test1();
            z=0;
        }
        Test2(int a,int b,int c)
        {
            x=a;
            y=b;
            z=c;
        }
        void Input()
        {
            Test1::Input();
            cout<<"Input Z=";cin>>z;
        }
        void Output()
        {
            Test1::Output();
            cout<<"Z="<<z<<endl;
        }
};
```

```
55  class Test3:private Test2{
56      private:
57       int z1;
58      public:
59        Test3()
60        {
61          Test1:Test1();
62          z1=0;
63        }
64        Test3(int a,int b,int c)
65        {
66              x=a;
67              y=b;
68              z1=c;
69        }
70          void Input()
71          {
72              Test1::Input();
73              cout<<"Input Z1=";cin>>z1;
74          }
75          void Output()
76          {
77              Test1::Output();
78              cout<<"Z1="<<z1<<endl;
79          }
80  };
81    int main()
82    {
83       Test3 obj3;
84       obj3.Output();
85       obj3.Input();
86       obj3.Output();
87    }
```



នៅក្នុង Single Inheritance **គេបែងចែកជា ពីរប្រភេទទៀត គឺ៖**

១) Direct Class(Sub Class to direct Base Class)

២) InDirect Class(Sub Class to Sub Class to Base Class)

1.2. Multiple Inheritance: គឺជាប្រភេទ Inheritance មួយប្រភេទទៀតដែលគេអាចបង្កើតនូវ Base Class ពីរ ឬច្រើន បន្ទាប់មក Sub Class មួយរឺ ច្រើនអាចទាញយកទិន្នន័យពី Base Class ទំនោះបាន។

ឧទាហរណ៏ ១៖ ចូរបង្កើតនូវ Base Class ពីរ និង Sub Class ចំនួន ១ដោយទាញទិន្នន័យពី Base Class ទាំងពីរ?

```cpp
#include<iostream>
using namespace std;
class Test1{
    protected:
    int a;
    int b;
};
class Test2{
    protected:
     float c;
     float d;
};
class Test3:private Test1,private Test2{
    public:
    Test3()
    {
        a=0;
        b=0;
        c=0;
        d=0;
    }
    Test3(int a1,int b1,float c1,float d1)
    {
        a=a1;
        b=b1;
        c=c1;
        d=d1;
    }
    void Output()
    {
        cout<<"A="<<a<<endl;
        cout<<"B="<<b<<endl;
        cout<<"C="<<c<<endl;
        cout<<"D="<<d<<endl;
    }
};
int main()
{ Test3 obj3;
  obj3.Output();
  cout<<".....................\n";
  Test3 obj4(12,52,100,60);
  obj4.Output();

}
```

```
C:\Users\USER\Documents\Untitled4.exe
A=0
B=0
C=0
D=0
.....................
A=12
B=52
C=100
D=60

--------------------------------
Process exited after 0.01664 seconds with return value 0
Press any key to continue . . .
```

លំហាត់អនុវត្តន៍

១) ចូរបង្កើត Class មួយឈ្មោះ Person ដែលមាន Data member ដូចជា ID(int), Name(String), Sex(String), DOB(String), Constructor ពីរគឺ Person(), Person(_,_,_,_) និង Method ចំនួនពីរ ដូចជា void Input() និង void Output() បន្ទាប់មកបង្កើតobject មួយ?

២) តពីលេខ១ ចូរបង្កើត class ចំនួនពីរទៀតគឺ Employee និង Students ហើយហៅ ទិន្នន័យពី Class Person មកប្រើបន្ទាប់មកបង្កើត Object នៃ Class ទាំងពីរ មកប្រើប្រាស់វា។

គេអោយ Base Class ដូចខាងក្រោម៖

```cpp
#include<iostream>
using namespace std;
 class Person{
    protected:
     int code;
     string name;
     string sex;
     string dob;
    public:
    Person()
    {
        code=0;
        name="N/A";
        sex="N/A";
        dob="dd/mm/yyyy";
    }
    Person(int i,string n,string s,string d)
    {
        code=i;
        name=n;
        sex=s;
        dob=d;
    }
    void Input()
    {
        cout<<"Input ID=";cin>>code;
        cout<<"Input name=";cin>>name;
        cout<<"Input Sex=";cin>>sex;
        cout<<"Input DOB=";cin>>dob;
    }
    void Output()
    {
        cout<<"ID="<<code<<endl;
        cout<<"Name="<<name<<endl;
        cout<<"Sex="<<sex<<endl;
        cout<<"DOB="<<dob<<endl;
    }
};
```

1.3. Hybrid Inheritance: គឺជាប្រភេទ នៃ Inheritance មួយបែបទៀតរបស់ OOP ក្នុង C++ ដែលវាអាចអនុញ្ញាតិអោយមានការទាញយកទិន្នន័យពី Base Class បន្តគ្នាពោលពី Base Class មួយទៅកាន់ Base Class មួយផ្សេងទៀត រហូតដល់ Derived Class។

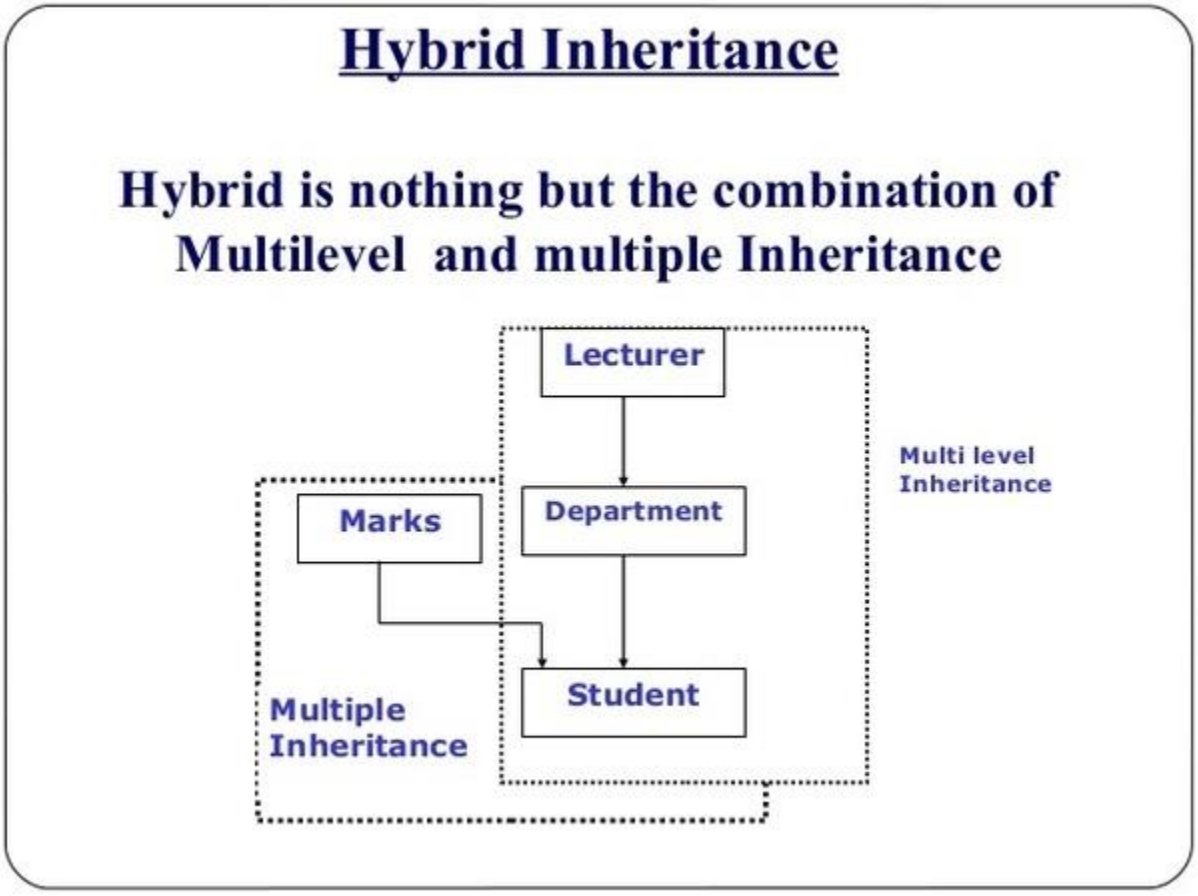
```
Syntax:


class A
{
     .........
};
class B : public A
{
     ..........
} ;
class C
{
     ...........
};
 class D : public B, public C
{
     ...........
};

```

ឧទាហរណ៏ ១៖

```
#include <iostream>
using namespace std;
class A
{
    public:
    int x;
};
class B : public A
{
    public:
    B()      //constructor to initialize x in base class A
    {
       x = 10;
    }
};
class C
 {
    public:
    int y;
    C()   //constructor to initialize y
    {
        y = 4;
        }
};
```

```
25  class D : public B, public C   //D is derived from class B and class C
26  {
27      public:
28      void sum()
29      {
30          cout << "Sum= " << x + y;
31      }
32  };
33
34  int main()
35  {
36          D obj1;          //object of derived class D
37      obj1.sum();
38      return 0;
39  }                   //end of program
```

លំហាត់អនុវត្តន៍



Good Luck!