UNIT -4

Inheritance – What is inheritance and its types

Inheritance

Inheritance is such a mechanism by which a new class is created from an old class. By this the properties of old class can be used in new class. To use the properties of the old class in the new class, the old class has to be inherited. And derivation public, private and protected are used to inherit the class.

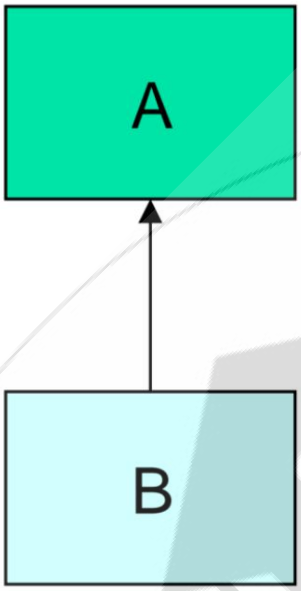
In inheritance, the old class is called the base class or parent class or super class, which also has the property of the class. And the new class is called child class or derived class or sub class, by which class the property is also known.

Types of inheritance

There are five types of inheritance-

1. Single
2. Multiple
3. Multilevel
4. Hierarchical
5. Hybrid

1.Single inheritance- When a class is inherited by another class, then that inheritance is called single inheritance. In this the property of one class is accessed by another class. In this there is a super class and a sub class.



Syntax :- class A {

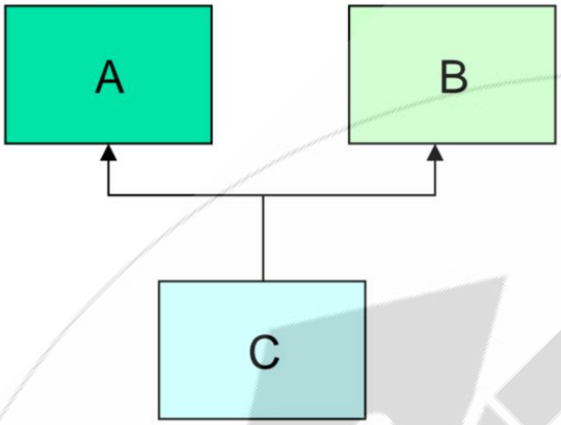
};

Class B : public A

{

};

Multiple inheritance- When more than one classes are inherited by a single class, that inheritance is called multiple inheritance. In this the property of more than one class is inherited by one class. It has more than one super class and one sub class. **Java does not support multiple inheritance.**



Syntax :- Class A {

};

Class B

{

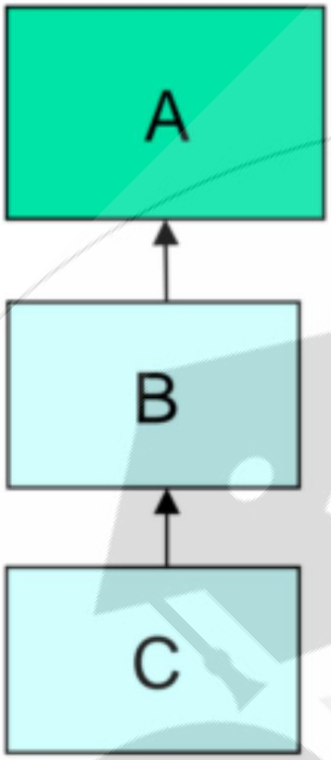
};

Class C : public A , public B

{

};

Multilevel inheritance- When more than one class inherits each other in one level, then that inheritance is called multilevel inheritance. In this a class inherits another class and that class which inherits the class becomes its sub class and the same sub class is inherited by another class. Similarly all classes inherit each other.



Syntax:- Class A {

};

Class B : public A

{

};

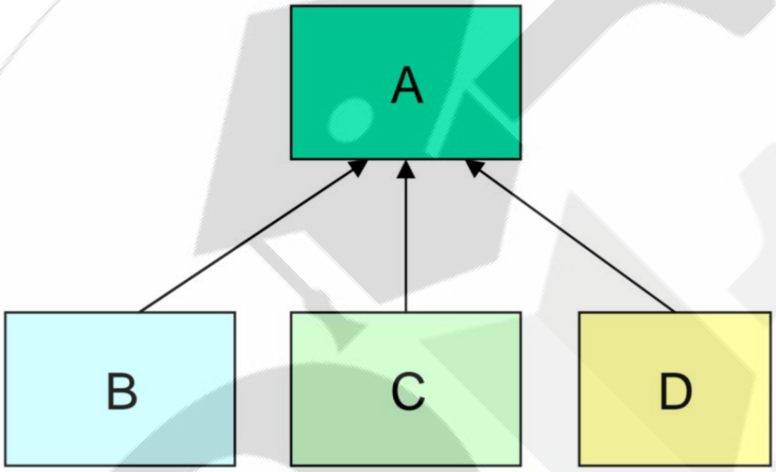
Class C : public B

{

};

4.Hierarchical inheritance-

When a base class is inherited by more than one sub class then that inheritance is called hierarchical inheritance. In this the property of one class is accessed by more than one class. In hierarchical inheritance there is a base class and more than one sub class. This is the opposite of multiple inheritance.



Syntax:- Class A {

};

Class B : public A

{

};

Class C : public A

{

};

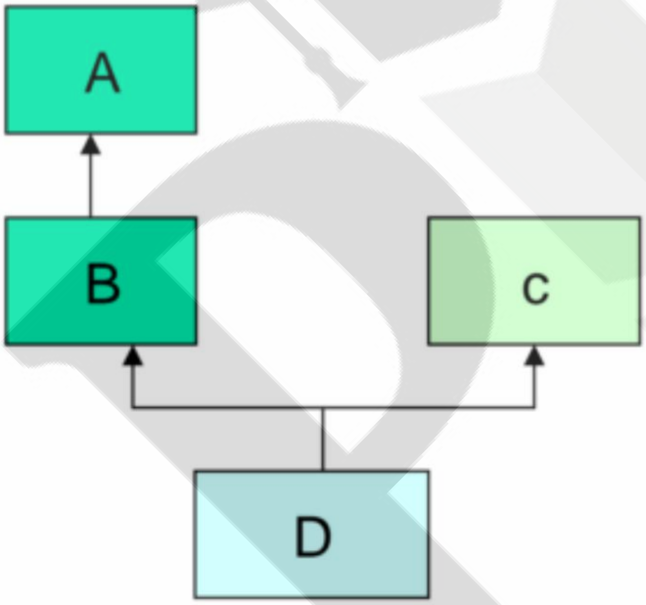
Class D : public A

{

};

5.Hybrid inheritance-

When we mix any two types of inheritance in our program, it is called hybrid inheritance. This inheritance is a combination of a genetic inheritance. That is, two or more than two inheritances are made of chemistry.



image

Syntax :- Class A {

};

Class B : public A

{

```
};

Class C

{

};

Class D : public B  , public C

{

};
```

Inheritance in C++ in hindi

This is an important concept of OOPs. As we know that we can reuse the codes to perform the same task in C++. The same happens in C++ inheritance.

In inheritance, we add the property and behavior of one class to another class without modifying the class. In this case, some of the properties or behavior of the second class are already there, but by inheritance the second class also receives the property and behavior of the first class. The class which is inherited from is called base or parent class whereas the class from which it is inherited is called derive or child class.

You can understand it in this way,

As in the real world, children inherit properties and behaviors from their parents, some of which are their own (such as their name) and some are inherited from their parents (such as their surname).

There can be more than one base class and derived class in a program, in this case, one derived class will be the base class for another and the same base class will be the derived class for any other class.

Live by example

In the same way that grandfather will be the base for the father whereas father will be the base for the child. But father will derive for grandfather but child will derive for father.

How to inherit a base class into derive-class in C++?

Its syntax is given below-

syntax

```
class derive-class-name : visibility-mode base-class-name
```

Here visibility-mode refers to the access specifier. It can be of private, public and protected type.

It means, in C++ we can inherit a class in three ways. So before introducing inheritance, we will discuss about these methods.

Access specifiers for inheritance in C++

- C++ inheritance Private mode
- C++ inheritance Public mode •
C++ inheritance Protected mode

C++ inheritance Private Mode

When a class is inherited with private mode, all the members of the base class become private members for the derived class, which can be accessed only by the members of the derived-class (public members of the derived class, not public objects of the derived class members of the base class).

Keep in mind, private members will not be accessed from any object nor from derived class members (private or public).

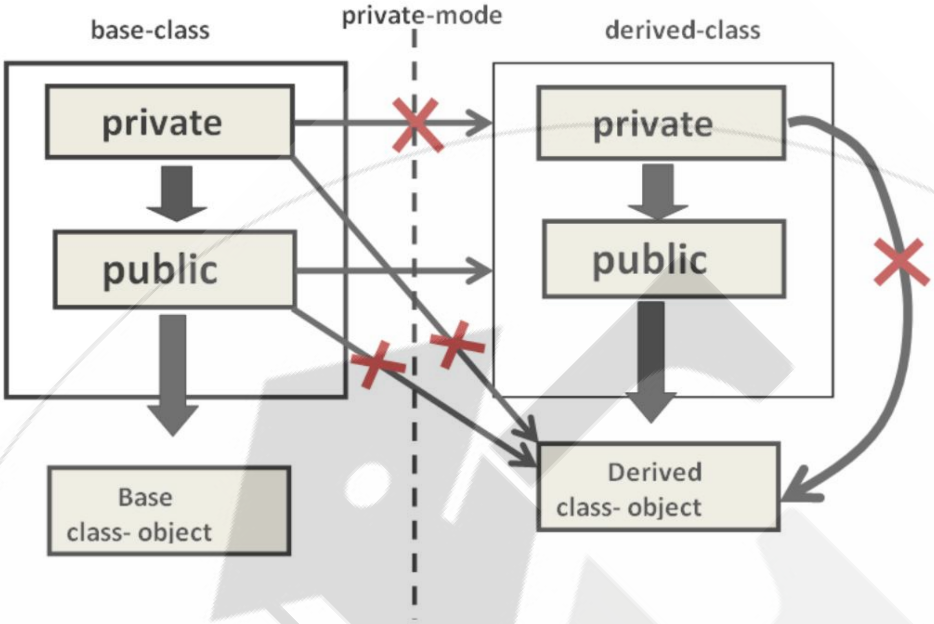
```
class derive class-name : private base-class-name

    {

        derive class members;

    }
```

Since the private specifier is by default, the private keyword will be optional here. messenger,

```
class derive-class-name : base-class-name

      {

            derive class members;

      }
```

: Private member of a class cannot be inherited. Private member and private mode both are different.

You can understand this with the help of given diagram –



C++ inheritance Public Mode

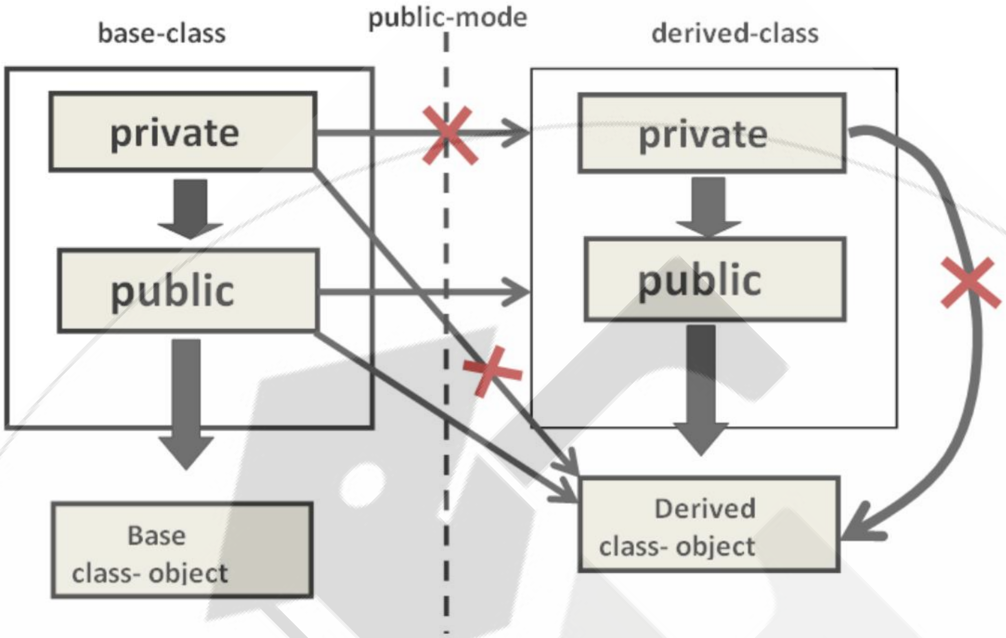
When a class inherits from the public specifier, then the public members of the base class can be accessed by both the derived class members and its objects. Also they can be inherited in the next class as well.

Here we can access public members of base class in two ways (from derive class members and its object)

Its syntax is given below-

```
class derive-class-name : public base-class-name

   {

            derive class members;

   }
```

The diagram given below explains it-



But if we want to access private members of base class then we replace private mode with protected mode-

C++ inheritance Protected Mode

As we know that private members of a class cannot be inherited. Means a member of a derive class does not have access to a private member of the base class. can.

If we want to access all the members then we have to declare private members to public. But by doing this the feature of data-hiding is not available.

To overcome this problem, a third specifier has been given in C++ which is called protected specifier. Declared in protected specifier, members of derived class can be accessed whereas objects cannot access these members In this way the feature of data hiding is also available and the rules of OOPs are not violated.

Its syntax has been given-

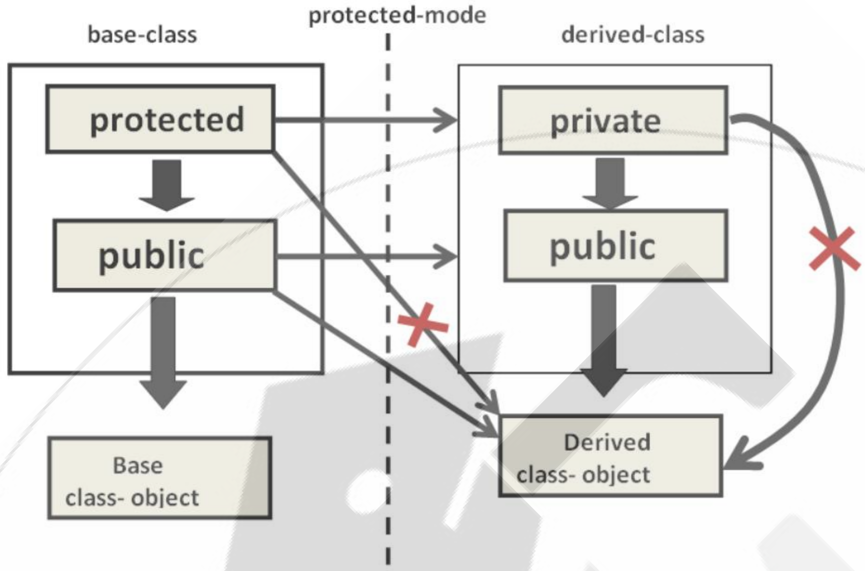
```
class derive-class-name : protected base-class-name
```

meaning,

```
class class_name

{

    private:

          private member;. // Accessible only same class member

    protected:

        protected member; // accessible derive class

    public:

        public member; //accessible own class and derive

};
```

Protected members can be accessed from derived class members and derived class objects.

You can understand this from the diagram given below –



To understand better, compare the above three diagrams with each other.

Type of inheritance in C++ hindi

There are 5 types of inheritance found in C++.

- C++ Single level inheritance • C++ multilevel inheritance
- C++ multiple inheritance • C++ Hierarchical inheritance • C++ Hybrid inheritance

Single Level Inheritance

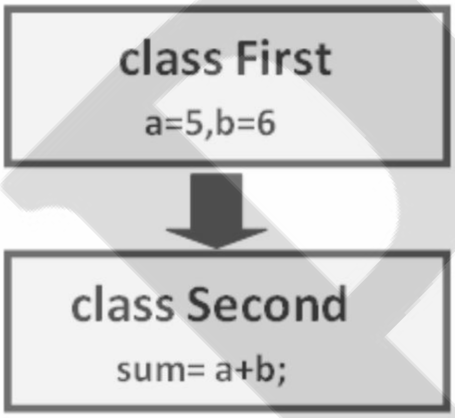
In this type of inheritance we give two classes in which the property of base class is inherited in one derived class.

Its syntax is given below-syntax

```
class derive-class-name : base-class-name

 {

     member of derive class;

     access base class member;

 }
```

Example

You can understand this with the help of the diagram given below –



Its program is given below-

```
#include<iostream.h>

#include <conio.h>

#include<stdio.h>



class first                //base class
```

```
{
   protected:
   int a,b;
   public:
   void get_num(int x,int y)
   {
       a = x;
       b = y;
   }
}; // first class terminate


class second:public first // class first inherit
{
     public:
     void get_sum()
     {
       int sum;
       sum = a+b;
       cout<<"Total: "<<sum;
     }
}; // second class terminate



void main() // main function start here

{

    int a,b;

    clrscr();


    second obj; //class second object declare


    cout<<"Enter two number: ";

    cin>>a>>b;


    obj.get_num(a,b); // class first member call

    obj.get_sum();              // class second member call


    getch();

}
```

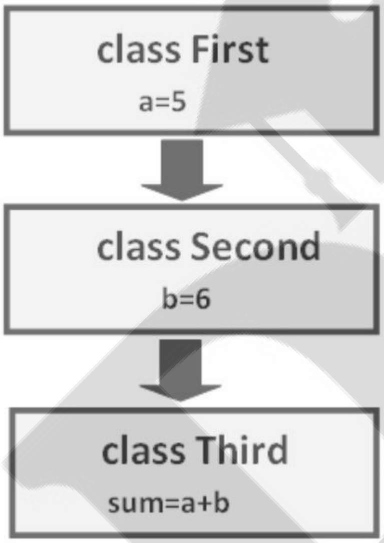
OUTPUT

```
Enter two number: 5 6

Total: 11
```

Multi-Level Inheritance

In this type of inheritance, there can be two or more classes, in this one class is inherited from the second class and the second class is inherited from the third class and so on.

Example



Here is the program,

```
#include<iostream.h>

#include <conio.h>

#include<stdio.h>



class first                    // base class

{
    protected:

    int a,b;

    public:

    void get_num(int x,int y)

    {

         a = x;

         b = y;

    }

}; // first class terminate


class second:public first                    // class first inherit in class second

{

    protected:

    int sum;

    public:

    void get_sum()

    {

        sum = a+b;

    }

}; //second class terminate
```

```
class third:public second                    // class second inherit in class third

{

    public:

    void show_sum()

    {

        cout<<"Total: "<<sum;

    }

}; // third class terminate


void main() // main function start here

{

    int a,b;

    clrscr();


    third obj; // class third object declare


    cout<<"Enter two number: ";

    cin>>a>>b;
    obj.get_num(a,b);  // class first member call

    obj.get_sum();                      // class second member call

    obj.show_sum();                    // class third member call



    getch();

}
```

OUTPUT

```
Enter two number: 5 6

Total: 11
```

Explanation:

program   In class first two variables are declared and class second adds these two numbers whereas in class third their total is printed. You can see this in the above diagram.

Multiple Inheritance

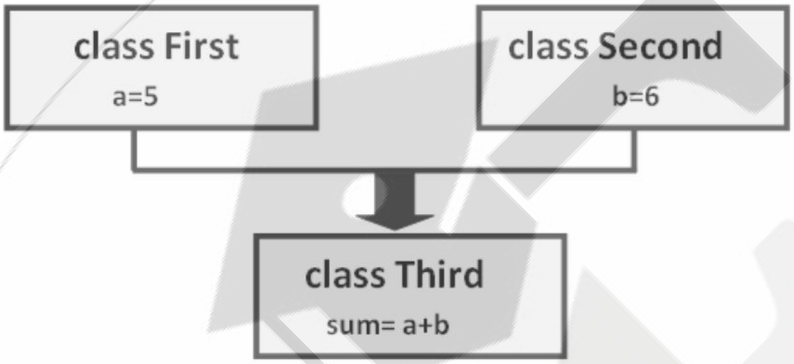
In this, the property of many classes is inherited in only one class. In a way it can be said that it has more than one base classes.

Its syntax is given as follows-

SYNTEX

```
class derive-class-name : mode base1-class-name,mode base2-class-name,…n

    {
        member of derive class;


    }
```

Example:-



Its program is given below-

```
#include<iostream.h>

#include <conio.h>

#include<stdio.h>



class first                    // base class first

{

    protected:

    int a;

    public:

    void get_num(int x)
    {

            a = x;

    }

}; // first class terminate



class second // base class second

{

      protected:

      int b;

      public:

      void get_num1(int y)

      {

              b = y;

      }

}; //second class terminate



class third:public first,public second  second              // class first and class
inherit in class third

{

      int sum;
    public:

    void show_sum()

    {

          sum = a+b; // access first and second class member

          cout<<"Total: "<<sum;

    }

}; // third class terminate

void main() // main function start here

{

    int a,b;

    clrscr();


    third obj;                  // class third object declare

    cout<<"Enter two number: ";

    cin>>a>>b;

    obj.get_num(a);             // class first member call

    obj.get_num1(b);             // class second member call

    obj.show_sum();              // class third member call


    getch();
```

```
}
```

OUTPUT

```
Enter two number: 5 6

Total: 11
```

Example:-

The above diagram explains this program.

Here we are taking two number from user first number for first class

```
obj.get_num(a);
```

And give the second number for the second class

```
obj.get_num1(b);
```

But in the third class only one task has been performed for both the numbers.

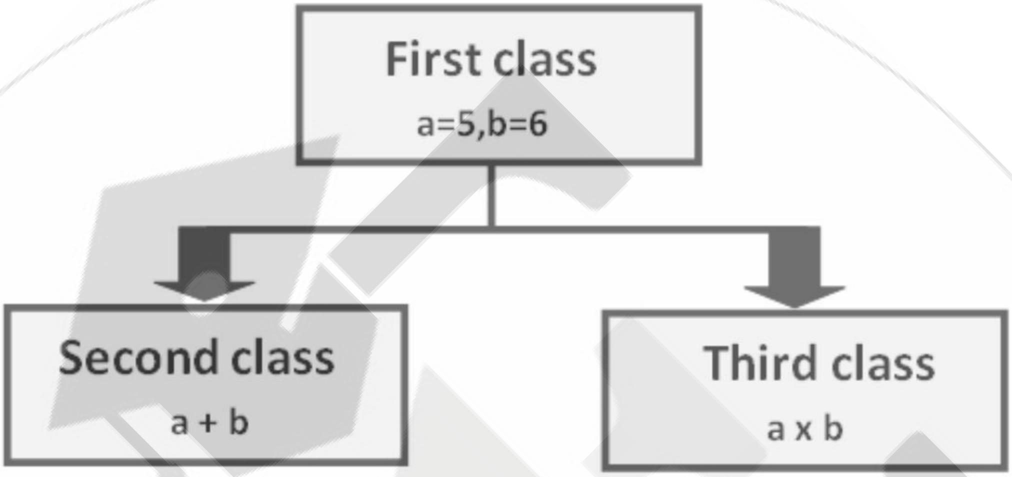
program   In me only object of class third has been declared and from this object both
been done. inherit or access member (get _num) of base class in third classJakaya has

Hierarchical Inheritance in C++ Hindi

This is in contrast to multiple-inheritance in which there is a base class whose properties are inherited by more than one derive class.

Its syntax is given below-

```
class derive-class-name : mode base1-class-name, mode base-class-name, ….n

 {
      member of derive class;

  }
```



Example

Here is the program,

```
#include<iostream.h>

#include <conio.h>

#include<stdio.h>



class first // base class first

{

   protected:

     int a,b;

   public:
     void get_num(int x, int y)

     {

        a = x;

        b = y;

     }

}; // first class terminate


class second:public first // class first inherit in class second

{

     int sum;

   public:

     void show_sum()

     {

        sum = a+b; // access first class member

        cout<<“\nSecond class:”;

        cout<<“\nTotal: “<<a<<“+”<<b<<” = “<<sum<<endl;

     }

}; //second class terminate


class third:public first // class first inherit in class third
```

```
{

     int mul;

   public:

     void show_mul()

     {

        mul = a*b; // access second class member

        cout<<“\nThird class:”;

        cout<<“\nMultiplicatoin: “<<a<<“x”<<b<<” = “<<mul;

     }

}; // third class terminate


void main() // main function start here

{

   int a,b;

   clrscr();


   second obj1; // class second obj1 declare

   third obj2; // class third obj2 declare


   cout<<“Enter two number: “;
   cin>>a>>b;



   obj1.get_num(a,b); // for second class

   obj2.get_num(a,b); // for third class

   cout<<“First class : “;

   cout<<"\nNumber are: "<<a<<” “<<b<<endl;



   obj1.show_sum(); // class second

   obj2.show_mul(); // class third



   getch();

}
```

OUTPUT

```
Enter two number: 5 6



First class:

  Number are: 5 6



Second class:

Total: 5+6 = 11
```

```
Third class:

Multiplication: 5x6 = 30
```

Explanation:

In the above diagram you can see that variables are declared in the first class whereas two different operations have been performed in the second class sum and in the third class multiply. Now here in void main the objects of class second and class third have been declared. Now from the object of first class second to the member of first class- access

```
obj1.get_num(a,b);
```

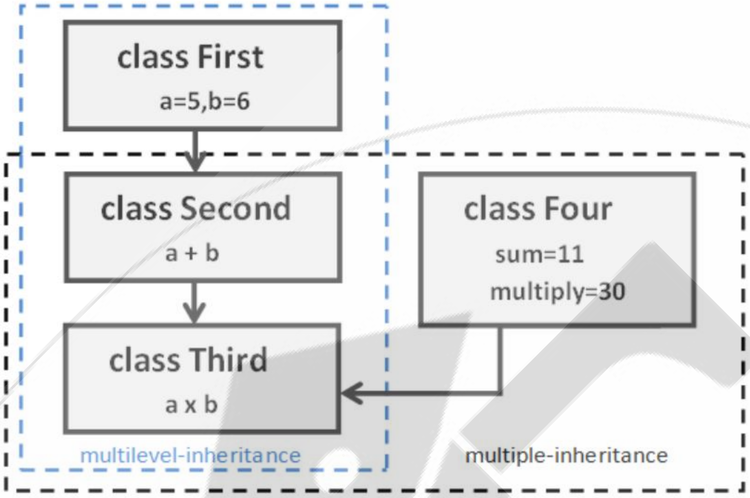
When the member of class first is accessed from the object of class third-

```
obj2.get_num(a,b);
```

Here the same function is accessed twice from different object.different object means different task (sum from first class and multiply from third class)

Hybrid Inheritance in C++ hindi

Hybrid inheritance is a collection of multiple and multi-level inheritance. It has more than one base class and more than one derive class-



Its program is given below-

```
#include<iostream.h>

#include <conio.h>

#include<stdio.h>



class first

 {

   protected:

     int a,b;

   public:

 void get_num(int x, int y)

     {

       a = x;
```

```
    b = y;

   }

 }; // first class terminate


class second:public first

 {

  protected:

    int sum;

  public:

   void get_sum()

    {

     sum = a+b; // access first class data-member

    }

}; //second class terminate


class third

{

 protected:

  int m,n, mul;

 public:
    void get(int x,int y)

    {

        m = x; // access second class member

        n = y; // access second class member

        mul = m*n;

    }

  }; // third class terminate



class fourth:public second,public third

{

  public:

    void show_all()

    {

      cout<<"\nTotal: "<<a<<"+"<<b<<" = "<<sum;

      cout<<"\nMultiplicatoin: "<<m<<"x"<<n<<" = "<<mul;

    }

}; //fourth class terminate



void main()

  {
   int a,b;

   clrscr();



 fourth obj; // fourth class object declare

 cout<<"Enter two number: ";

 cin>>a>>b;



 obj.get_num(a,b); // class first member

 obj.get(a,b); // class second member call

 obj.get_sum(); // class second member call

 obj.show_all(); // class fourth member call



 getch();

 }
```

OUTPUT

```
Enter two number: 5 6

Total: 5+6 = 11

Multiplication: 5x6 = 30
```

Explanation: