INHERITANCE (reuse)

IS-A RELATIONSHIP

THE TECHNIQUE OF BUILDING NEW CLASSES FROM THE

EXISTING CLASSES IS CALLED INHERITANCE.

existing classes -> base class -> father

|

|

|

new classes -> derived class -> son

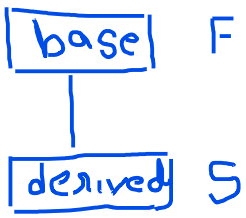
TYPES OF INHERITANCE

1. **SINGLE INHERITANCE**
2. **MULTILEVEL INHERITANCE**
3. **MULTIPLE INHERITANCE**
4. **HYBRID INHERITANCE**
5. **MULTIPATH INHERITANCE**
6. **HIERARCHICAL INHERITANCE**

ACESSS MODIFIER (VISIBILITY)

public	protected	private
main() ✓	main() X	main() X
fun() ✓	fun() X	fun() X
	public ✓ (inherit class)	public ✓ (same class)

// SINGLE INHERITANCE




```
#include<iostream>
using namespace std;
class   base
 {
        protected : int  a , b;

        public    :    void   get()
                       {
                         cout<< " ENTER TWO NOS " << endl;
                         cin>> a >> b;
                       }
 };
```

```
class   derived :  public  base
{
        public   : void  out()
                   {
                        cout << a << endl << b << endl;
                    }
};

int    main()
{

      derived p;
      p . get();
      p . out();
}
```

CALLING PRIVATE MEMBER FUNCTION

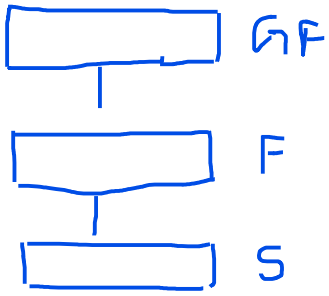
```
#include<iostream>
using namespace std;
class   base
{
      protected :  int  a , b ;

      public    :    void  get()
                     {
                          cout<< " ENTER TWO NOS " << endl;
                          cin>>a>>b;
                     }
};
```

```
class   derived :   private base
{
        public  :   void out()
                    {
                        get(); // calling  private  member  functions
                        cout << a << endl << b << endl;
                    }
};

int   main()
{
      derived p;
      p . out();
}
```

---

class members -->	public	protected	private
inherit --> public	public	protected	private
inherit --> protected	protected	protected	private
inherit --> private	private	private	private

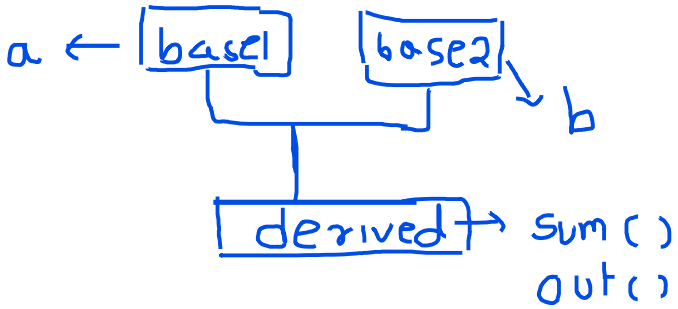
MULTILEVEL INHERITANCE




```
#include<iostream>
using namespace std;

class   base
{
        protected  :  int   a , b ;

         public    :   void  get()
                       {
                          cout<<" ENTER TWO NOS"<<endl;
                          cin >> a>> b;
                       }
};

class   derived1   :   public  base
{
        protected  :  int  c ;

         public    :  void sum()
                       {
                              c = a + b;
                        }
};
class  derived2    :   public  derived1
 {
        public     :
```

```
                void    out()
                {
                        cout<< " sum = " << c << endl;
                }
        };
        int    main()
        {
                derived2  p;

                p. get();

                p. sum();

                p. out();
        }
```

MULTIPLE INHERITANCE



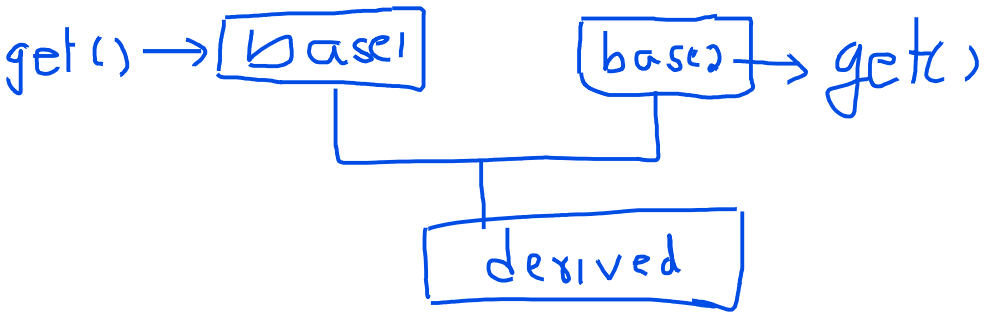

```
#include<iostream>
using namespace std;

class    base1
{
        protected  :  int  a;
        public    :     void   get1()
                        {
                                cout << " ENTER a = "<< endl;
                                cin >> a;
                        }
```

```
};
class   base2
 {
        protected   :  int  b;
        public      :  void  get2()
                       {
                           cout << " ENTER b = " << endl;
                           cin >> b ;
                       }
 };

class    derived     :  public base1 , public  base2
{
         protected :  int  c ;

          public    :
                        void sum()
                        {
                            c = a + b;
                        }
                       void out()
                       {
                            cout << " SUM = " << c << endl;
                       }
 };

int    main()
{
```

```
        derived p;

        p . get1();

        p . get2();

        p . sum();
        p . out();
}
```

---

ambiguity error



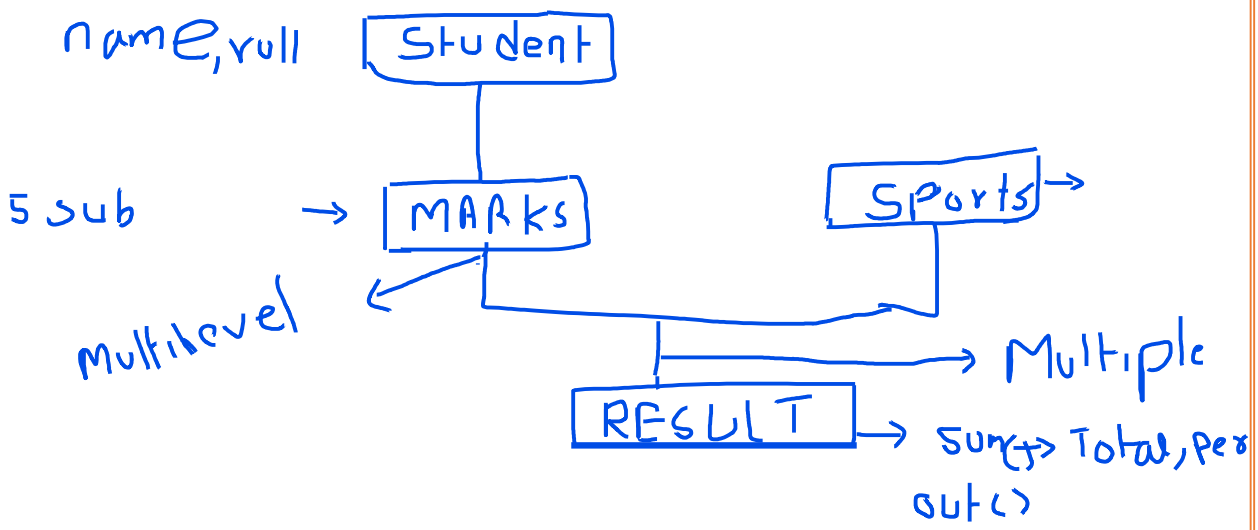
SOLVE AMBIGUITY ERROR

```
int   main()
{
      derived p;
```

```
        p . base1 :: get();

        p . base2 :: get();

        p . sum();
        p . out();
}
```

HYBRID INHERITANCE

(multilevel + multiple)

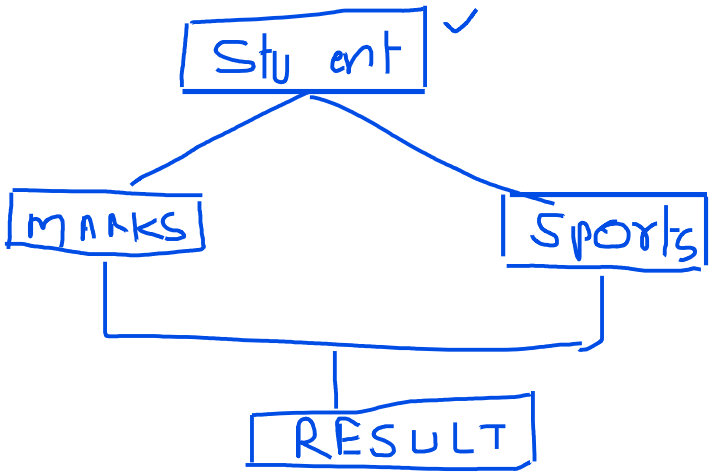


1. **class marks : public student**

2. **class result : public marks , public sports**

MULTIPATH INHERITANCE

(multilevel + multiple)



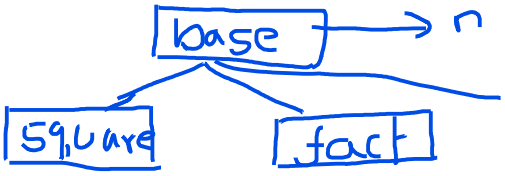
daigram

1. **class marks : virtual public student**

2. **class sports : virtual public student**

3. **class result : public marks , public sports**

HIERARCHICAL INHERITANCE



1. **class fact : public base**

2. **class square : public base**

// HIERARCHICAL INHERITANCE

```
#include<iostream>
using namespace std;

  class base
  {
       protected : int n;

       public :
              void get()
              {
```

```
                cout << "enter no " << endl;
                cin >> n;
            }
};

class square :  public base
{
    public:
            void out()
            {
                cout << n*n << endl;
            }
};

class fact  :  public base
{
  protected :  int f;

  public :
            fact()
            {
               f = 1;
            }
            void   cal()
            {
                for( int i=1 ;i <=n ; i++)
                {
                      f= f * i;
```

```
        }
    }
    void   out()
    {
        cout<< "fact=" << f << endl;
    }
};

int   main()
{

        square s;

        s. get();

        s. out();

        fact p;

        p. get();

        p. cal();

        p. out();


}
```

```
/*
-----------------------------------------------------------------
```

sequence of constructor and destructor

single inheritance

constructor	destructor
base	derived
derived	base

multilevel inheritance

constructor	destructor
base	derived2
derived1	derived1
derived2	base

multiple inheritance

a) class derived : public base1 , public base2

constructor destructor

base1 derived

base2 base2

derived base1

b) class derived : public base2 , public base1

constructor destructor

base2 derived

base1 base1

derived base2

***/**