


- 
- OBJECT
 - ORIENTED
 - PROGRAMMING
 - IN
 - C++.



IN THE OOPS WE WILL PERFORMS VARIOUS
COMPONENTS.

1)CLASS

2)OBJECT

3)ABSTRACT

4)ENCAPCULATION

5)INHERITANCE

CLASS

Class is a collection of member data and member function.

Member data is a collection of attribute in a class.

Member function is a collection of operation .It is perform in the class.

SYNTAX OF CLASS

```
Class classname
```

```
{
```

```
Private:
```

```
    All member data
```

```
Public:
```

```
    All member function
```

```
};
```

EXAMPLE OF CLASS WITH THE HELP OF SUM OPERATION

```
#include<conio.h>
#include<iostream.h>
Class sum
{
Private:
    int a,b,c;
Public:
    void input()
    {
Cout<<"enter the value of a and b variable";
Cin>>a>>b;
    }
```

```
Void display()
{
C=a+b;
Cout<<"value of sun is"<<c;
}
};
Void main()
{
Sum s;
Clrscr();
s.input();
s.output();
Getch();
}
```

OBJECT OBJECT

Object is a real entity world. With the help of OBJECT we can call the member data and member function. It is just like a membership.

Syntax of object:

```
class name      classobject;
```

```
Classobject.mamner function();
```

EXAMPLE OF CLASS AND OBJECT WITH THE HELP OF SUM OPERATION

```
#include<conio.h>
#include<iostream.h>
Class sum
{
Private:
    int a,b,c;
Public:
    void input()
    {
    Cout<<"enter the value of a and b variable";
    Cin>>a>>b;
    }
```

```
Void display()
{
C=a+b;
Cout<<"value of sun is"<<c;
}
};
Void main()
{
Sum s; // Here (s) is a object . It is calling the member data and member function.
Clrscr();
s.input();
s.output();
Getch();
}
```

ABSTRACT

An abstract class is, conceptually, a class that cannot be instantiated and is usually implemented as a class that has one or more pure virtual (abstract) functions.

pure virtual function is one which **must be overridden** by any concrete (i.e., non-abstract) derived class. This is indicated in the declaration with the syntax "`= 0`" in the member function's declaration.

ENCAPCULATION

Encapsulation is the packing of data and functions into a single component. The features of encapsulation are supported using classes in most object-oriented programming languages, although other alternatives also exist. It allows selective hiding of properties and methods in an object by building an impenetrable wall to protect the code from accidental corruption.

Best example of Encapsulations is class because it have the multiple of member data and member function in the same class.

INHERITANCE

With the help of inheritance we can reuse the data one class to another class .

Which inherits the properties of base class is known as parent class/base class/super class.

Which inherits the properties of parent class is known as sub class/child class/inherited class.

Inheritance separates into many classes...

- 1) Single Inheritance
- 2) Hierarchical Inheritance
- 3) Multi Level Inheritance
- 4) Hybrid Inheritance
- 5) c

Single Inheritance

when a single derived class is created from a single base class then the inheritance is called as single inheritance.

It is known as single inheritance class.

BASE CLASS



DERIVED
CLASS

SYNTAX OF SINGLE INHERITANCE

```
class a
{
Public:
    all member data;
Public:
    all member function;
};
class b:public a
{
Public:
    all member data;
Public:
    all member function;
};
```

EXAMPLE OF SINGLE INHERITANCE

```
#include<iostream.h>
#include<conio.h>
Class sum
{
Public:
    Int a=5,b=5,c;
Public:
    void display()
    {
        c=a+b;
```

```
    cout<<"value of sum is"<<c;  
}
```

```
};
```

```
class sub: public sum  
{
```

```
Public:
```

```
    Int d;
```

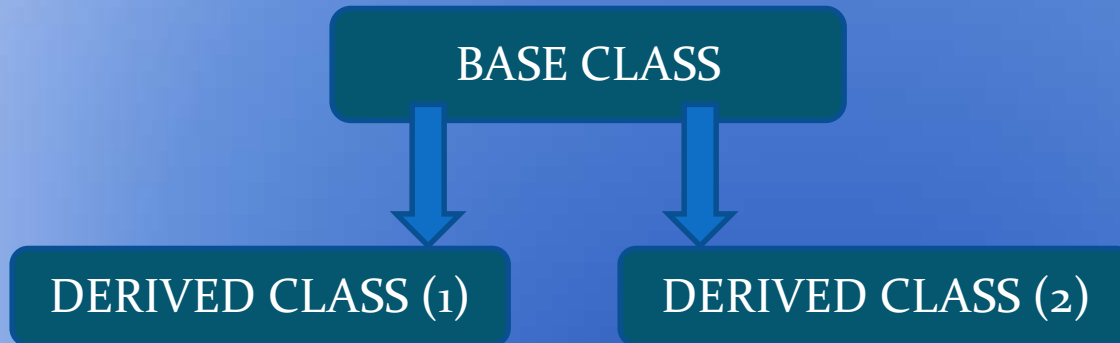
```
Public:
```

```
    void output()
```

```
        {  
            d=a-b;  
            cout<<"the value of subtraction is"<<d;  
        }  
};  
void main()  
{  
    clrscr();  
    sub s;  
    s.display();  
    s.output();  
    getch();  
}
```

Hierarchical Inheritance


when more than one derived class are created from a single base class, then that inheritance is called as hierarchical inheritance.



SYNTAX OF HIEARCHICAL INHERITANCE

```
Class a
{
Public: all member data;
Public: all member function;
};
```

```
Class b:public a
{
Public: all member data;
Public: all member function;
};
```



```
Class c:public a
{
Private: all member data;
Public: all member function;
};
```

EXAMPLE OF HIARCHICAL INHERITANCE

```
#include<conio.h>
#include<iostream.h>
Class a
{
    public :
        int a=10,b=2,c,d,e;
Public:
    void output()
    {
```

```
        c=a-b;
        Cout<<"value of sum is"<<c;
    }
};
class b:public a
{
    Public:
        void display()
    {
```

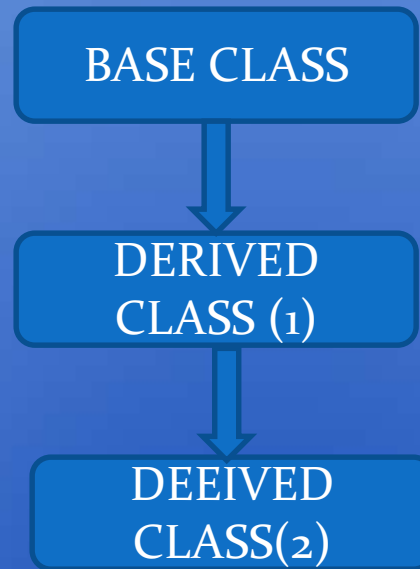
```
d=a-b;
Cout<<"value of subtraction is;
Cout<<d;
}
};
Class c:public
{
Public:
        void coutput()
{
```

```
e=a*b;
Cout<<"multiplication is"<<e;
}
};
void main()
{
Clrscr();
        a    l;
l.output();
```

```
l.display();  
l.coutput();  
getch();  
}
```

Multi Level Inheritance

when a derived class is created from another derived class, then that inheritance is called as multi level inheritance.



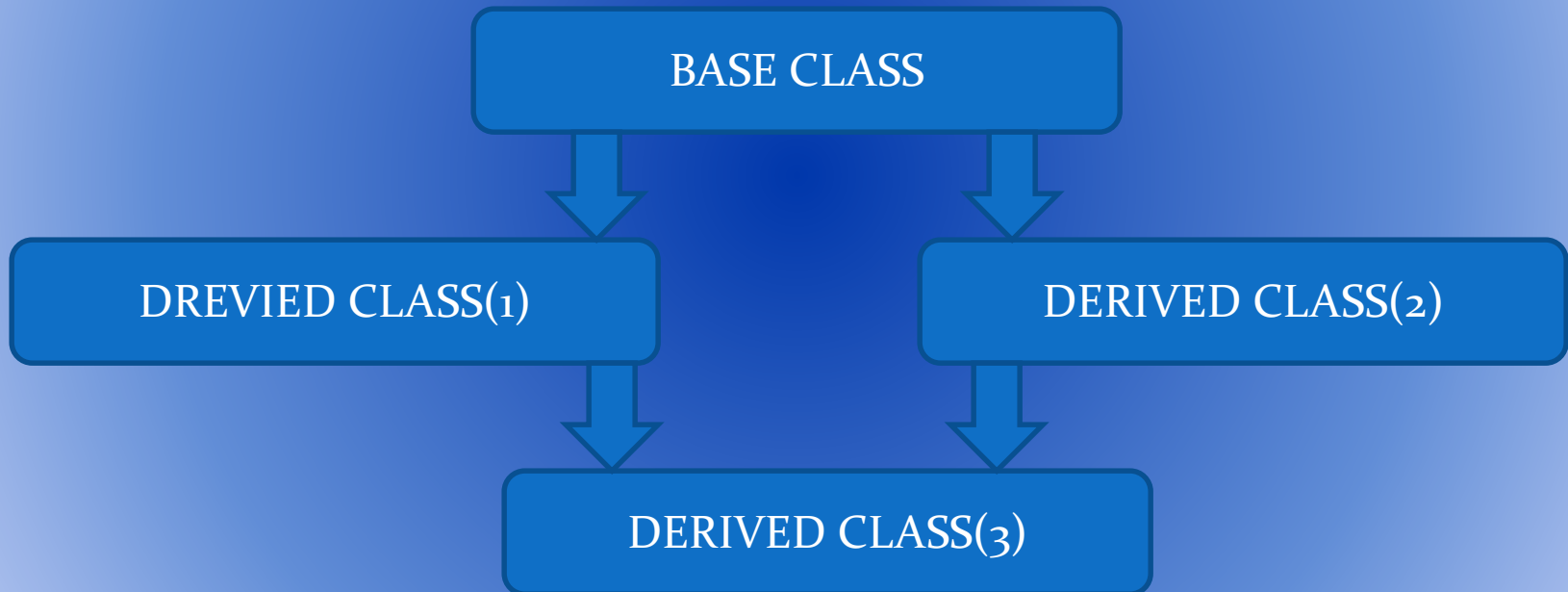
SYNTAX OF MULTILEVEL INHERITANCE

```
Class a
{
Public: all member data;
Public: all member
      function;
};
Class b:public a
{
Public: all member data;
```

```
Public: all member
      function;
};
Class c:public c
{
Private: all member data;
Private: all member
      function;
};
```

Hybrid Inheritance

Hybrid inheritance is combination of two or more inheritances such as single, multiple, multilevel or Hierarchical inheritances.



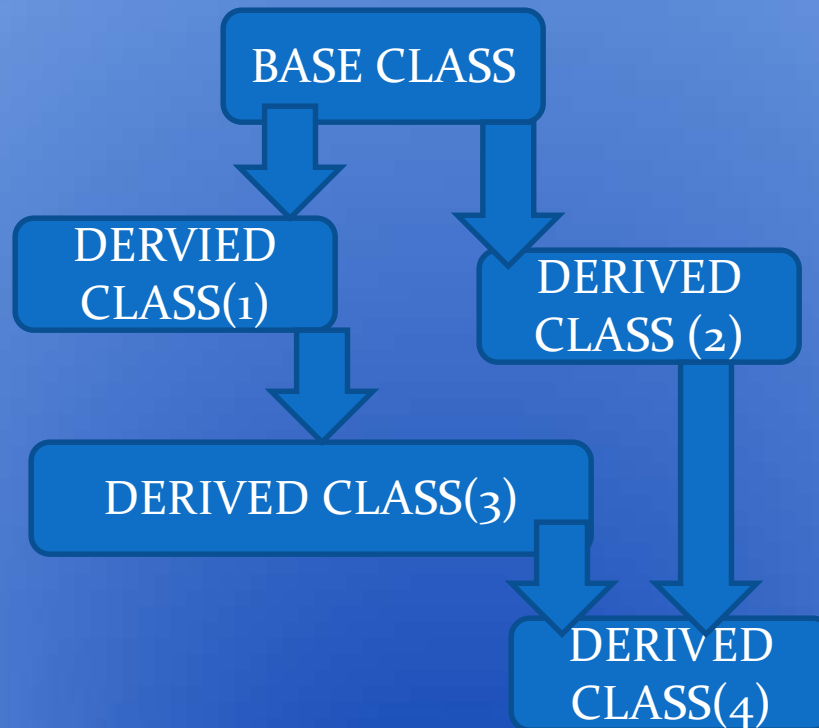
SYNTAX OF HYBRID INHERITANCE

```
Class a
{
Public:all member data;
Public:all member function;
};
Class b:public a
{
Public:all member data;
Public:all member function;
};
```

```
Class c:public c
{
Public:all member data;
Public:all member function;
};
Class d:public d
{
Public:all member data;
Public:all member function;
};
```

Multiple Inheritance

Deriving directly from more than one class is usually called multiple inheritance.



SYNTAX OF MULTIPLE INHERITANCE

```
Class a
{
Public:all memberdata;
Public:all member function;
};
Class b:public a
{
Public:all memberdata;
Public:all member function;
};
```

```
Class c:public a
{
Public:all memberdata;
Public:all member function;
};
Class d:public b
{
Public:all memberdata;
Public:all member function;
};
```

Class e:public c,public d
{
Private:all member data;
Public:all member
function;
};www.cpd-india.com