Page No:
Date: 21/5/16

Inheritance

→ Reusability is a very most important feature of OOP.

→ old class is refer as base class or parent class.

→ New class is refer as derived class or child class.

class [A] Parent class / Base class

↓

class [B] child class / Derived class.

→ Inheritance means reusable, reusable mean something is already exist in base class and we use or derived in another class that, which is already exist in base class.

→ There Derived class :-

it is defined by specifing its relationship with the base class in additional its own details.

Inheritance Accesible operator

Syntax :-

class derived clsnm : visibility mode baseclsnm

{

==

};

Mode
Public
Protected
Private

Single Inheritance

```
class a
{
    int i; // private member
public:
        a()
        {
            i=5;
        }
        void show()
        {
            cout<<"i="<<i;
        }
};

class b : public a
{
    int j;
public:
```

o/p

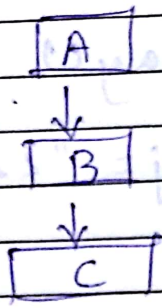
```
b()
{
  j=10;
}
void display()
{
  cout<<"j= "<<j;
  show();
}
};
void main()
{
  classr()
  {
    b b1;
    b1.display();
  }
}
```

o/p
j=10
i=5

→ Private member of derived class ni direct Access sathi dilela nahi.

→ Thats why we using show function. Here we can see that without creating b a class object we use its member function from b class object.

Multilevel Inheritance



Example

```
#include<iostream.h>
#include<conio.h>
class a
{
    int a1,a2;
    public:
        int sum(int n1, int n2)
        {
            a1=n1;
            a2=n2;
            int Sum1=0;
            Sum1 = a1+a2;
            return Sum1;
        }
        void show()
        {
            cout<<"In a class";
        }
};
```

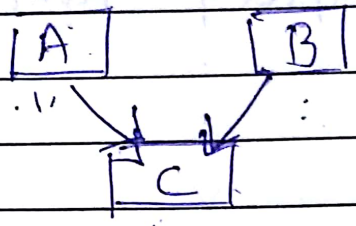
```
class b : public a
{
    public:
    void display()
    {
        cout<<"display function in bclass";
    }
};

class c : public b
{
    public:
        void output()
        {
            cout<<" In c class :";
            int s = sum(10,5);
            cout<<"sum = "<<s;
        }
};

void main()
{
    c c1;
    c1.show();
    c1.display();
    c1.output();
}
```

O/P -

```
in a class
in b class
in c class
sum = 15
```

Multiple Inheritance

More than one parents of child

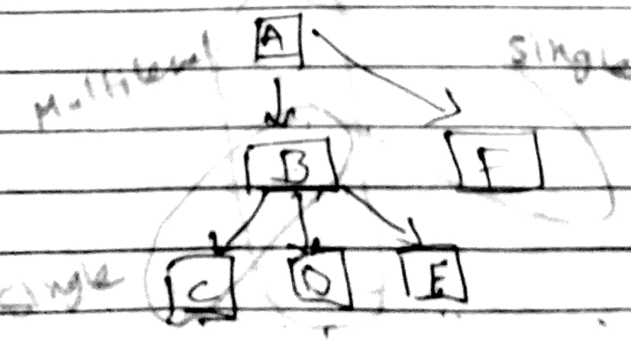



```
#include<iostream.h>
#include<conio.h>
class A
{
   public:
          void display()
          {
             cout<<"in a class";
          }
};

class B
{
   public:
          void show()
          {
            cout<<"in b class";
          }
};
```

```
class c : public A , public B
{
    public:
        void output()
        {
            cout<<" In c class ";
        }
};

void main()
{   clrscr();
    c c1;
    c1.display();
    c1.show();
    c1.output();
    getch();
}
```

O/P:
In a class
In b class
In c class

Hierarchical Inheritance



→ Combination of Single & Multilevel Inheritance

* Different Between Function Overloading and Function Overriding.

(Compile Time)
Function Overloading

→ Inheritance is not required.

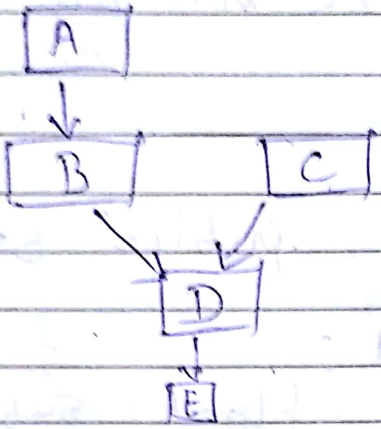
→ Ex. class a
{
void show()
{
==
}
void show(int a)
{
==
}
void show(int a, int b)
{
==
}
};

(Run Time)
Function Overriding

→ Inheritance is required.

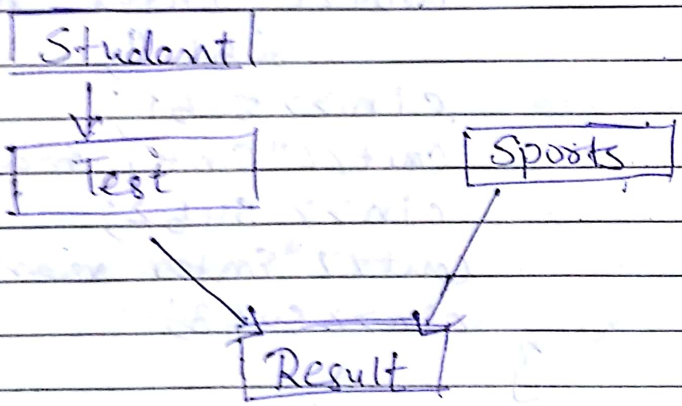
→ class a
{
void show()
{
==
}
};
class b : public a
{
void show()
{
==
}
};

Hybrid Inheritance



→ Combination of single, Multiple and Multilevel Inheritance

Example




```
#include<iostream.h>
#include<conio.h>
class Student
{
    public:
        int rno;
        Student() // Constructor
        {
            cout<<"Enter roll no:";
        cin>>rno;
```

```
    void outputrno()
    {
        cout<<" roll no = "<<rno;
    }
};

class Test : public student
{
    protected:
            float sub1, sub2, sub3;
    public:
            Test()  // constructor
            {
                cout<<"Enter marks for
                    sub1:";
                cin>>sub1;
                cout<<"Enter marks for sub2";
                cin>>sub2;
                cout<<"Enter marks for sub3";
                cin>>sub3;
            }

            void outputmarks()
            {
                cout<<" Marks of subject 1 "<<sub
                cout<<" Marks of subject 2 "<<sub2
                cout<<" Marks of subject 3 "<<sub3
            }
};
```

```
class sports
{
    public:
        float score;
        Sports() //constructor
        {
            cout<<"Enter Sports score:";
            cin>>score;
        }
        void outputscore()
        {
            cout<<" Score = "<<score;
        }
};

class result : public Test, public Sports
{
    public:
        float total;
        void outputresult();
};

void result :: outputresult()
{
    total = sub1 +sub2 +sub3 +score;
    cout<<"      result      \n";
    outputroll();
    outputmarks();
    outputscore();
    cout<<" Total Score =" <<total;
}
```

```
Void main()
{
    clrscr();
    result r1; // default Constructor
    r1.outputresult();
    getche();
}
```

o/p -

```
Enter roll no - 101
Enter sub1 marks - 66
Enter sub2 marks - 77
Enter sub3 marks - 88
Enter sport score - 99

    result
roll no - 101
sub1 marks - 66
sub2 marks - 77
sub3 marks - 88
Score - 99
Total score - 9
```

Type Casting / Polymorphism

→ Convert to Base class pointer to Derived class pointer.

(Derived *(bptr))

Derived class name — Base class pointer name

Ans

```
#include<iostream.h>
#include<conio.h>
class Base
{
    public:
        void show()
        {
            cout<<" In base class";
        }
};

class Derived : public Base
{
    Public:
        Void show()
        {
            cout<<" In child class";
        }
};
```

dptr = &b; } Error
dptr → show() } Solu → Can not convert base → to derived →

Void main()
{
clrscr();
Base b1; Base *bptr;
Derived d1; Derived *dptr;

Base class fun call
bptr = &b1; Base pointer
bptr => show(); ↓ Base object fun call

bptr = &d1; Base pointer
bptr => show(); Derived ↓ Base object fun call

Derived class fun call
dptr = &d1; Derived pointer
dptr -> show(); ↓ Derived object fun call

((Derived *) bptr); Derived pointer
((Derived *)) -> show(); base object fun call

}

Explicitly call

Ex. int i
float j;

(int i) float) → conver float to int

→ જે class ના pointer હોય તે class ની function call થશે.

Page No.:
Date: 7/5/12

Polymorphism / Function Overriding

```
#include <iostream.h>
#include <conio.h>
class Base1
{
    public:
        void display()
        {
        cout<<"\n base class 1";
        }
};

class Base2
{
    public:
        void display()
        {
        cout<<"\n base class 2";
        }
};

class Derived : public Base1, public Base2
{
    public:
        void display()
        {
        cout<<"\n derived class";
        Base1::display();
        }
};
```

```
int main()
{
    clrscr();
    Derived d;
    Derived *d1;

    d.Base1::display();   // base1 fun call
    d.display();          // Derived fun call

    ((Base2 *)&d)->display();   // base2 fun call
    ((Base2 *)d1)->display();
    getch();
    return 0;
}
```

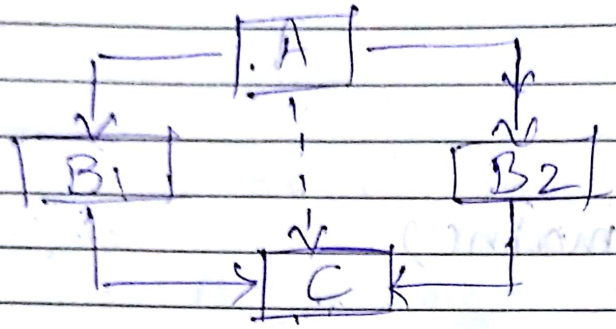
⇒ Function Call using classname in function:

Ex. void display()
{
    base1::display();
}

⇒ We can create main function return type int, float etc but we have to write return 0.

Ambiguity (Virtual Base Class)

Virtual key words.



→ यहाँ class A का data B1 & B2 में derived कर रहे है. तो class A का Data 2 time copy होगा. जिस memory को without use waste करेगा. तो इसको Virtual key words का use करते है, class A का only one ही copy करेगा.

Ex.

```
class A
{
    public:
        ==
};
class B1 : virtual public A
{
    ==
};
class B2 : public virtual A
{
    ==
};
```

class C : public B1, public B2
{
==
};

void main()
{
——
——
}

→ When base class derived more than one class this time situation create is Ambiguity.

IMP Initialization List in Constructor
or
Calling Parameterized Constructor

Ex.
class a
{
int n;
public:
a(int a)
{
n = a;
}

void display()
{
cout<<"n = "<<n;
};

```
public
class b: public a
{
    int no1, no2;
public:
    b(int n1, int n2) : a(n1, n2)
    {
        // a(n1, n2)
        no1 = n1;
        no2 = n2;
        cout<<" Derived class Constructor";
    }
    Void show()
    {
        cout<<" no1 = "<<no1;
        cout<<" no2 = "<<no2;
    }
};
void main()
{
    int one, two;
    cout<<" Enter two number :";
    cin>> one >> two;
    b b1(one, two);
    b1.display(); b1.show();
    getch();
```

Calling parameterized Constructor

Simple base class constructor call using this.

O/P - Enter two nums - 4, 2

base class constructor - 8

Derived class constructor - 4, 2

IMP => જ્યારે Default constructor હોય અને ત્યારે Derived class નો Object create કરીએ તો તે Automaticaly first Base class નું default constructor Call કરશે. પરંતુ parameterized constructor હોય તો તેનો ઉપયોગ કરવા derived class ના constructor માં Base class ના constructor ને Call કરાવવું પડે / call કરવું પડે.

Virtual Function

```
#include<iostream.h>
#include<conio.h>
class a
{
    public:
        virtual void show()
        {
            cout<<"In base class";
        }
};
```

```
class b : public a
{
  public:
        void show()
        {
          cout<<" In derived class ";
        }
};

void main()
{
    a a1;
    a *aptr;

    aptr = &a1;
    aptr -> show();
    b b1;
    aptr = &b1;
    aptr -> show();

    aptr = &b1;
    aptr -> show();

    //  (aptr*).show();

}
```

If i write virtual key ↓ Derived class fun call

it i not write ex it create fun create ↓ Base class fun call

O/P - In base class
In derived class
In base class

this keyword

Example

```
#include<iostream.h>
#include<conio.h>
class a
{
        int n;
        public:
                void input(int i)
                {
                    n=i;
                }
                void output()
                {
                    cout<<"maximum number-"<<n;
                }
                a max(a ob)
                {
                    if(ob.n>n)
                    {      20 > 10
                        return ob;
                    }
                    else
                        return *this;
                }
};
void main()
{
    clrscr();
    a a1,a2,a3;
    int i
```

this keyword जो है function object जो fun call [illegible] value return [illegible]

Page No.:
Date: / /

```
a1.input(10);
a2.input(20);

a aa = a1.max(a2);
aa.output();
getch();
}
```

> જ્યારે object return કરાવવો હોય ત્યારે fun return type classname એવું પડશે.