INHERITANCE

- The process of Inheriting the properties of one class into another class is called Inheritance.
- The process of Inheritance can be either simple or complex.

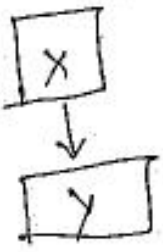
(i) No base classes - The program can use one or more base class and a new class can be derived from it.

(ii) Nested derivation - The derive class can be used as a base class and a new class can be derived from it

This based on the above points inheritance can be classified as

(a) single Inheritance
(b) multiple "
(c) hierarchical "
(d) multilevel "
(e) Hybride "
(f) multipath "

SINGLE INHERITANCE



X is a base class
Y is a derived class. This type of involves one base and one derived class. Further no class is derived from Y.

When one class is derived from a single base class, such a derivation of class is known as single inheritance. Further the derived class is not used as a base class. This type of Inheritance uses one base class and one derived class.

* write a program to show single Inheritance between two classes.

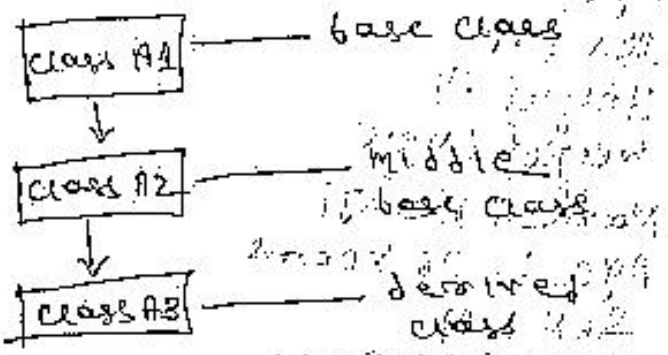
```
# Include <iostream.h>
# include <conio.h>
class ABC
{
protected:
 char name[15];
  int age;
 };
class abc : public ABC
 {
float height;
 float weight;
 public:
 void ~~show~~ getdata()
 {
cout << "\n Enter Name and Age:";
 cin >> name >> age;
cout << "\n Enter Height and weight :";
 cin >> height >> weight;
 }
void Show()
 {
cout << "\n Name:" << Name << "\n Age:" < age;
          << "years";
cout << "\n Height :" << height << "feets" << "\n
weight:" << weight << "kg.";
}
};
int main()
{
clrscr();
abc x;
x.getdata(); // reads data through keyboard
x.show(); // display data on screen.
return 0;
}
```

output

```
Enter Name and Age:
santosh 24
Enter Height and weight:
4.5 50
Name : santosh
Age: 24 years
Height : 4.5 feets
weight : 50 kg.
```

Multilevel Inheritance

The procedure of deriving a class from a derived class is called Multilevel inheritance



Write a program to create multilevel inheritance. create classes A1, A2, and A3

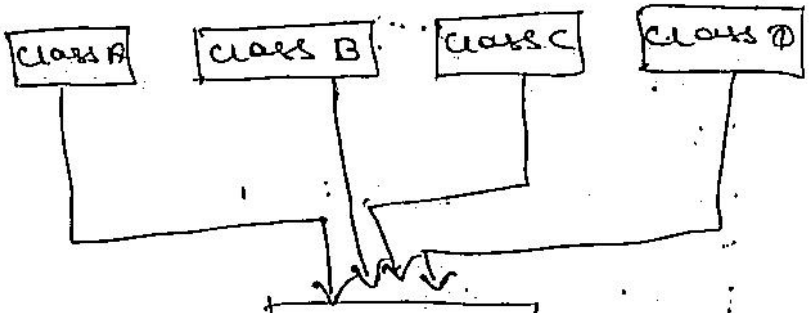
```
#include<iostream.h>
#include<conio.h>
class A1
{
protected:
 char name[15];
 int age;
};
class A2 : public A1
{
protected:
 float height;
 float weight;
};
class A3 : public A2
{
protected:
 char sex;
 public:
 void get()
 {
cout<<"Name:";
 cin>>name;
cout<<"Sex:";
 cin>>sex;
cout<<"Age:";
 cin>>age;
cout<<"Height:";
 cin>>height;
 cout<<"Weight:";
 cin>>weight;
 }
 void show()
 {
cout<<"\nName:"<<name;
cout<<"\nAge:"<<age;
cout<<"\nSex:"<<sex;
cout<<"\nHeight:"<<height
 <<"feets";
cout<<"\nWeight:"<<weight
 <<"Kg.";
 }
};
int main()
{
clrscr();
A3 x;
x.get();
x.show();
return 0;
}
```

Output

```
Name : Balaji
Age : 26
Sex : M
Height : 4
Weight : 49.5
Name : Balaji
Age : 26 years
Sex : M
Height : 4 feets
Weight : 49.5 Kg
```

Multiple Inheritance

When a class is derived from more than one class, this type of Inheritance is called Multiple Inheritance.



Write a program to derive a class from multiple base class

```
#include<iostream.h>
#include<conio.h>
class A
{
protected:
int a;
};
class B
{
protected:
int b;
};
```

```
class C
{
protected:
 int c;
};
class D
{
protected:
 int d;
};
class E : public A, B, C, D
{
 int e;
public:
 void getdata()
{
cout<<"\n Enter values of a, b, C and d and e:";
  cin>>a>>b>>c>>d>>e;
}
void Showdata()
{
cout<<"\n a="<<a<<"b="<<b<<"C="<<c<<"d="<<d<<
 "e="<<e;
}};
int main()
 {
  class c);
  E x;
 x.getdata(); //reads data
 x.Showdata(); // displays data.
  return 0;
 }
```

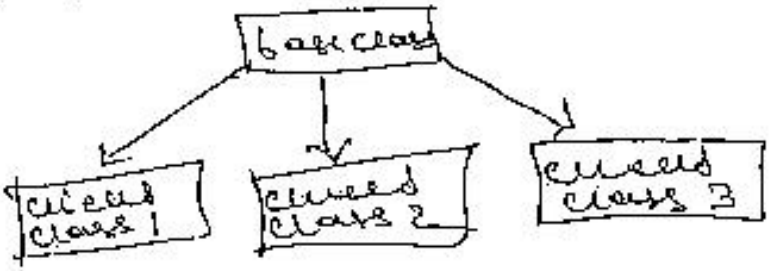
Output

Enter values a,b,c and d and e : 1 2 3 4 5

a=1 b=2 c=3 d=4 e=5

Hierarchical Inheritance

When more than one classes are derived from a single base class is known as hierarchical Inheritance

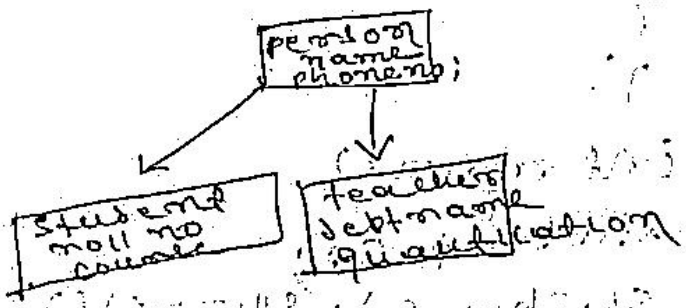



```
class base-class name
 {
 properties;
 methods;
 };
class derived class name 1 : visibility mode base-class
 name
 {
  properties;
  methods;
 };
class derived class name2 : visibility mode base-class
 name
 {
  properties;
  methods;
 };
 - - -
 - - -
class derived_class N : visibility mode base class
                         name.
 {
 properties;
  methods;
 }
```

Write a program to demonstrate hierarchical inheritance.

```
#include<iostream.h>
#include<conio.h>
  class person
   {
    protected:
```

```
char name[20];
long int phno; // int phno[12];
public:
void read()
{
cout<<"\n Enter name";
cin>>name;
cout<<"\n Enter phno";
cin>>phno;
}
void show()
{
cout<<"\n Name="<<name;
cout<<"\n phno="<<phno;
}
};
class student : public person
{
protected:
int rollno;
char course[20];
public:
void read()
{
cout<<"\n Enter Roll No";
cin>>rollno;
cout<<"\n Enter Course";
cin>>course;
}
void show()
{
cout<<"\n ROLLNO="<<rollno;
cout<<"\n course="<<course;
}
};
```

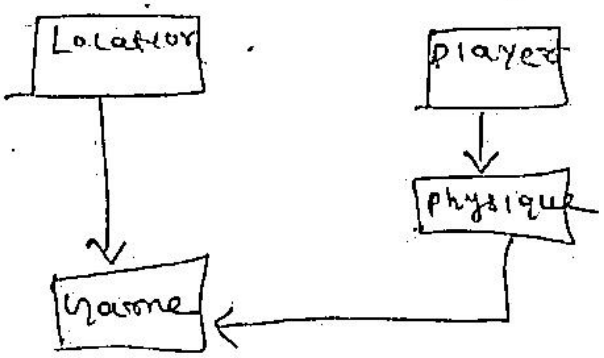


```
class teacher : public person
{
protected:
char dept_name[10];
char qual[10];
public:
void read()
{
cout<<" Enter dept_name and qualification"
cin>>dept_name>>qual;
}
void show()
{
cout<<" In Department ="<<dept_name;
cout<<" In Qualification ="<<qual;
}
};
int main()
{
student s1;
s1.show(); s1.read(); // enter student information
                      // displaying student information
teacher t1;
t1.read(); // enter teacher information
t1.show(); // displaying teacher information
return 0;
}
```

Hybride Inheritance

A combination of one or more type of inheritance is known as hybrid Inheritance

Write a program to create a derived class from base class




```
#include<iostream.h>
#include<conio.h>
class PLAYER
{
protected:
char name[15];
char gender;
int age;
};
class PHYSIQUE : public PLAYER
{
protected:
float height;
float weight;
};
class LOCATION
{
protected:
char city[10];
char pin[7]; // int pin;
};
class NAME : public PHYSIQUE, LOCATION
{
protected:
char game[15];
public:
void getdata()
{
cout<<"Enter following Information \n";
cout<<"Name:"; cin>>name;
cout<<"Gender:"; cin>>gender;
cout<<"Age:"; cin>>age;
cout<<"Height:"; cin>>height;
```

```
cout << "Weight:"; cin >> weight;
cout << "city:"; cin >> city;
cout << "pincode:"; cin >> pin;
cout << "name:"; cin >> name;
}
void show()
{
cout << "\n Entered Information";
cout << "\n Name:"; cout << name;
cout << "\n Gender:" << gender;
cout << "\n Age:" << age;
cout << "\n Height:" << height;
cout << "\n Weight:" << weight;
cout << "\n city:" << city;
cout << "\n pincode:" << pin;
cout << "\n name:" << name;
}
};
int main()
{
clrscr();
NAME n;
n.getdata();
n.show();
return 0;
}
```

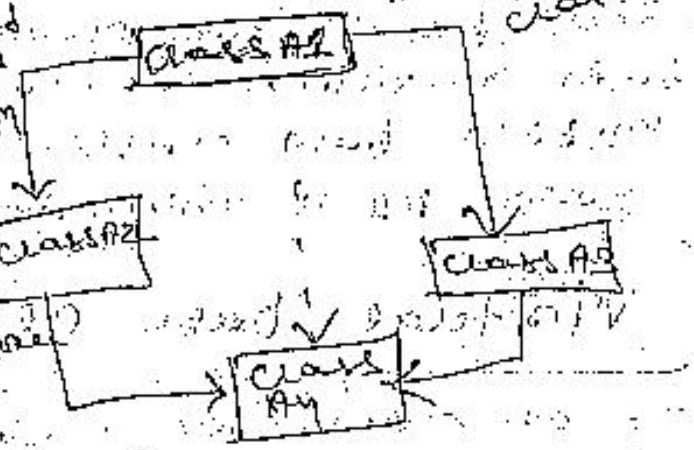
Output

```
Enter the Following Information
Name: Mathew
Gender: M
Age: 25
Height: 4.9
Weight: 55
city: cuttack
pincode: 768018
name: cricket

Entered Information
Name: Mathew
Gender: M
Age: 25
Height: 4.9
Weight: 55
city: cuttack
pincode: 768018
name: cricket
```

Multipath Inheritance

When a class is derived from two or more classes and these classes are derived from the same base class such a type of inheritance is known as multipath Inheritance



Ambiguity in classes

The multipath inheritance also consists of many types of inheritance, such as multiple multilevel and hierarchical

Ex:-
```
class A1
{
protected:
 int a1;
};
class A2 : public A1
{
protected:
 int a2;
};
class A3 : public A1
{
protected
 int a3;
};
class A4 : public A2, A3
{
int a4;
};
```

class A2 and A3 are derived from class A1 that is their base class is similar to class A2 (Hierarchical inheritance). Both classes A2 and A3 can access the variables of class A1 the class A4 is derived from classes A2 and A3 by multiple inheritance

if we try to access the variable a1 of class A1. the compiler shows error message.
- here the derived class A4 has two sets of data members of class A1 through the middle base classes A2 and A3. The class A1 is inherited twice

Virtual base class

To overcome the ambiguity occurring due to multipath inheritance C++ provides a keyword virtual
- The keyword declares the specified class virtual. that is only one copy of the base class A1 will come to derived class A4 through A2 and A3

```
class A1
{
protected:
  int a1;
};
class A2 : virtual public A1 // virtual class declaration
{
protected:
  int a2;
};
class A3 : virtual public A1 // virtual class declaration
{
protected:
  int a3;
};
class A4 : public A2, A3
{
  int a4;
};
```

When classes are declared as virtual, the compiler takes the essential caution to avoid the

duplication of member variables

8% Write a program to declare virtual base classes. Define a class using two virtual classes.

```
#include<iostream.h>
#include<conio.h>
class A1
{
protected:
int a1;
};
class A2: public virtual A1 // virtual declaration
{
protected:
int a2;
};
class A3: public virtual A1 // virtual declaration
{
protected:
int a3;
};
class A4: public A2, A3
{
int a4;
public:
void get()
{
cout<<"Enter value for a1, a2, a3, a4:";
cin>>a1>>a2>>a3>>a4;
}
void put()
{
cout<<"a1="<<a1<<"a2="<<a2<<"a3="<<a3<<
"a4="<<a4;
}
};
```

```
int main()
{
clrscr();
A y;
a.get(); // reads data
a.out(); // displays data
}
```

output

Enter values for a1, a2, a3 and a4 : 5 8 7 3

a1=5 a2=8 a3=7 a4=3

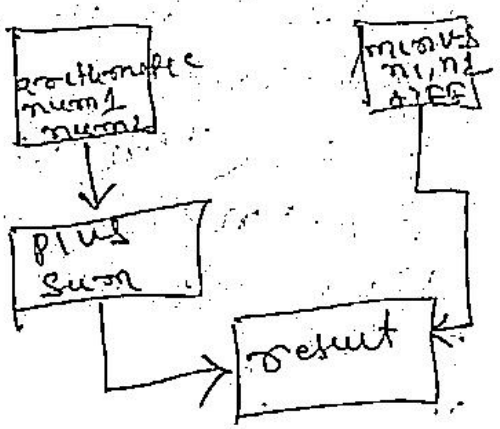
Advantages of Inheritance

- It provides code reusability. The existing classes remains unchanged. By reusability the development time of software is reduced.
- The derived classes extend the properties of base classes to generate more desired objects.
- The same base class can be used by number of derived classes in class hierarchy.
- When a class can be used by number of derived classes in class hierarchy.
- When a class is derived from more than one class, all the derived classes have similar properties to those of base class.

Disadvantages of Inheritance

- Inappropriate use of inheritance makes programs more complicated
- Invoking member functions using objects creates more complex overhead

hybrid inheritance example




```
#include<iostream.h>
#include<conio.h>
class arithmetic
{
protected:
int num1, num2;
public:
void getdata()
{
cout<<"For Addition:";
cout<<"\n Enter the first number:";
cin>>num1;
cout<<"\n Enter the second number:";
cin>>num2;
}
};
class plus : public arithmetic
{
protected:
int sum;
public:
void add()
{
sum=num1+num2;
}
};
class minus
{
protected:
int n1, n2, diff;
public:
void sub()
}
```

```
cout<<"In for Subtraction:";
cout<<"In Enter the first number:";
cin>>n1;
cout<<"In Enter the second number:";
cin>>n2;
diff = n1-n2;
}
};
class result: public plus, public minus
{
public:
void display()
{
cout<<"In Sum of"<<num1<<"and"<<num2
<<"="<<sum;
cout<<"In Difference of"<<n1<<"and"<<
n2<<"="<<diff;
}
};
int main()
{
result r;
r.getdata();
r.add();
r.sub();
r.display();
return 0;
}
```

output

```
for Addition
Enter the first number:10
Enter the second number:5
for Subtraction
Enter the first number:6
Enter the second number:7
Sum of 10 and 5 =15
Difference of 6 and 7=-1
```