Inheritance

-by Praveen Choudhary

What is Inheritance?

Inheritance is the ability of one class to inherit the properties of another class. A new class can be created from an existing class. The existing class is called the Base class or Super class and the new class is called the Derived class or Sub-class.

e.g:

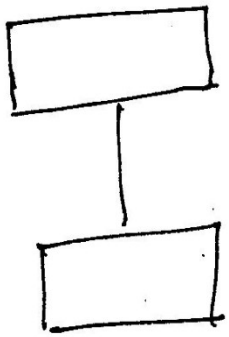
Car inherits from another class auto-mobile.
Science student inherits from class student

Advantages of Inheritance:

1. Reusability of code
2. Size of the code is reduced.
3. Transitivity:
   If B is derived from A and C is derived from B then C is also derived from A.

1) SINGLE INHERITANCE:

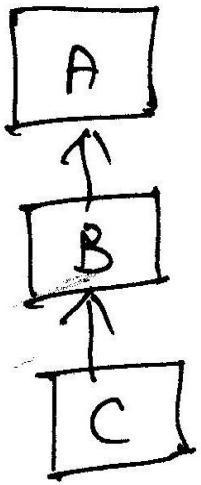
When a subclass inherits from one base class



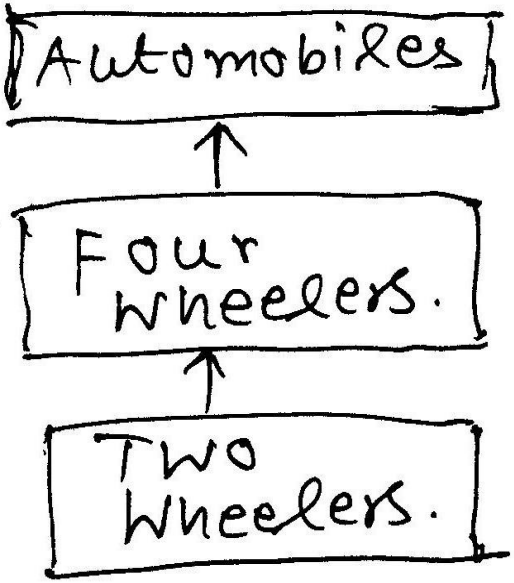
Person - Base Class

Student - Derived class

2) MULTI-LEVEL INHERITANCE:



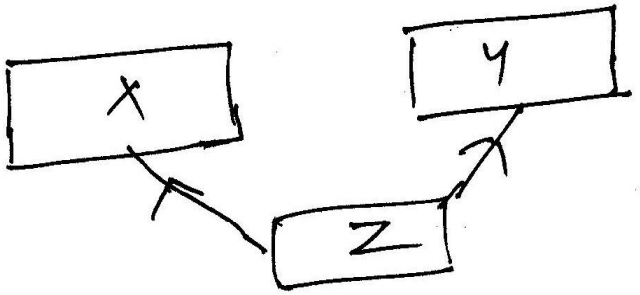
e.g.



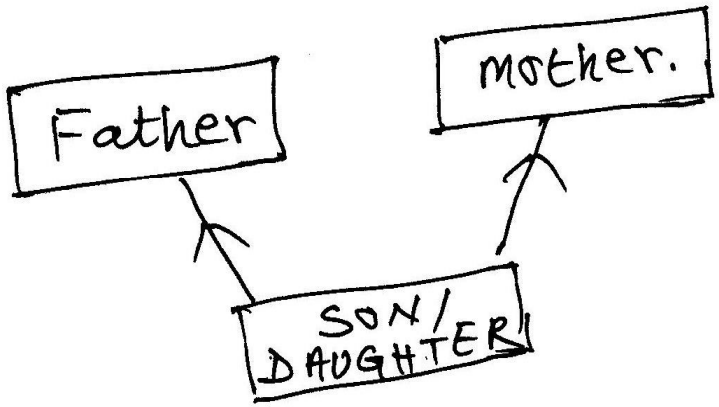
When a sub class inherits from a class that itself inherits from another class

MULTIPLE INHERITANCE.

When a Sub class inherit from multiple base classes.

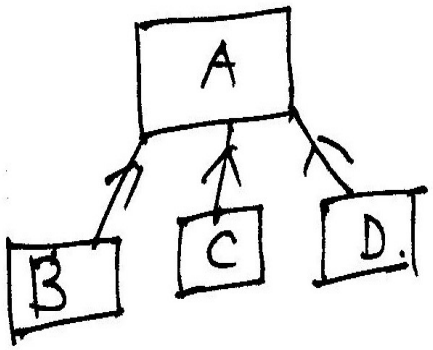


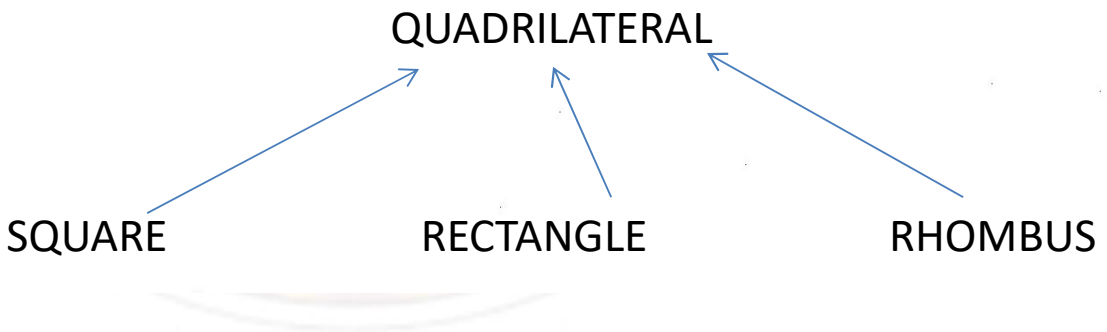
e.g



Hierarchical Inheritance:

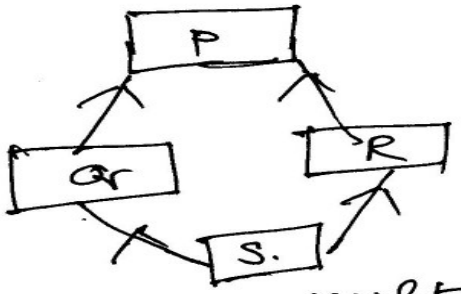
When many sub-classes inherit from a single Base class.





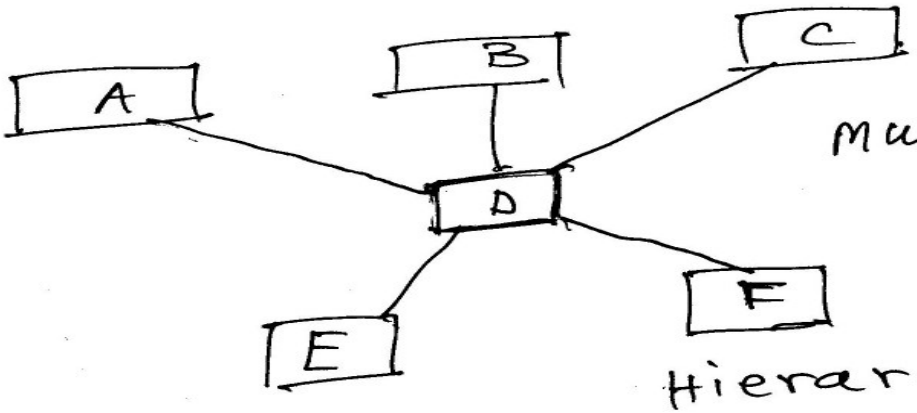
Hybrid Inheritance:

Contains two or more forms of inheritance.



Hierachical

multiple.



multiple

Hierarchical

```
#include <iostream.h>
class person
{ char name[20];
  int age;
  public:
    void get()
      { cout<<"enter name and age"<<endl;
        cin >> name >>age;
      }
    void show()
    { cout<< name <<"is"<< age <<endl;
    }
};
class student : public person
{
    char group[10];
    public:
     void get1()
      { get(); "enter science or commerce group"<<endl;
        cout<<
        cin>> group;
      }
    void show1()
      { show();
        cout<<" group " << group << endl;
      };

int main()
{ student ob1;
  ob1.get1();
  ob1.show1();
  return 0;
}
```

Identify the type of inheritance:

```
class FacetoFace
{
     char CenterCode[10];
      public:
     void Input();
      void Output()
  };

class Online
{
      char website[50];
     public:
     void SiteIn();
     void SiteOut();
 };
```

```
class Training : public  FacetoFace, private Online
{

            long Tcode;
            float Charge;
            int Period;
    public:
        void Register();
        void Show();
};
```

Base Classes:

FacetoFace **Online**

Derived Class:

Training

Multiple base classes so multiple inheritance

```
Class Dolls
{
        char Dcode[5];

    protected:
        float price;
        void CalcPrice(float);

    public:
        Dolls();
        void Dinput();
        void Dshow();
};
```

```
class  SoftDolls: public Dolls
{
                        char SDName[20];
                        float Weight;
            public:
                         SoftDolls();
                        void SDInput();
                        void SDShow();
};

class ElectronicDolls:  public Dolls
{
                        char EDName[20];
                        char BatteryType[10];
                         int Batteries;
            public:
                        ElectronicDolls();
                        void EDInput();
                        void EDShow();
};
```

Hierarchical Inheritance

```
class furniture
{
                    char Type;
                    char Model[10];
          public:
                    furniture();
                    void Read_fur_Details();
                    void Disp_fur_Details();
  };

class Sofa : public furniture
{
                    int no_of_seats;
                    float cost_of_sofa;

          public:
                    void Read_sofa_details();
                    void Disp_sofa_details();
};
```

```
class office : private  Sofa
{
                        int no_of_pieces;
                        char Delivery_date[10];

        public:
                        void Read_office_details();
                        void Disp_office_details();
};

void main()
{
        office MyFurniture;
}
```

Sofa is derived from furniture
Office is derived from sofa.

Multi-level Inheritance

Visibility Modes

It can be public, private or protected.

The private data of base class cannot be inherited.

(i) If inheritance is done in public mode, public members of the base class become the public members of derived class and protected members of base class become the protected members of derived class.

(ii) If inheritance is done in a private mode, public and protected members of base class become the private members of derived class.

(iii) If inheritance is done in a protected mode, public and protected members of base class become the protected members of derived class.

Accessibility of Base Class members:

Access	public	protected	private
members of the same class	yes	yes	yes
members of derived classes	yes	yes	no
not members	yes	no	no

```
#include<iostream.h>
class one
{
        int a;   // only for class members
        protected:
        int b;    // for class members and derived classes
        public:
        int c;   // for class members, derived classes, main
        one()
        {
                a=3;
                b=5;
                c=10;
                }
         void show()
         {
                cout<<a<<":"<<b<<":"<<c<<endl;
                }
                };
```

```
class two :public one
{
        int p;
        public:
        two()
        {
                p=25;
                }
        void show1()
        {
                cout<<a<<endl;  \\ error.  Not accessible
                cout<<b<<endl; \\o.k.
                cout<<c<<endl; \\o.k.
                }
                };
```

```
class three : public two
        {
                int x;
                public :
                three()
                {
                        x=100;
                        }
                        void show2()
                        {
                                cout<<x<<endl; \\o.k.
                                cout<<p<<endl; \\error. Not accessible
                                cout<<b<<endl; \\o.k.
                                cout<<c<<endl; \\o.k.
                                }
                                };
```

```
int main()
        {
                three ob;
                cout<<ob.c<<endl; \\o.k. public member
                cout<<ob.b<<endl; \\ error.  Not available
                ob.show();
                ob.show1();
                ob.show2();
                return 0;
        }
```

```
#include<iostream.h>
class one
{
        int a;   // only for class members
        protected:
        int b;    // for class members and derived classes
        public:
        int c;   // for class members, derived classes,main
        one()
        {
                a=3;
                b=5;
                c=10;
                }
         void show()
         {
                cout<<a<<":"<<b<<":"<<c<<endl;
                }
                };
```

```
class two :protected one
        {
                int p;
                public:
                two()
                {
                        p=25;
                        }
                void show1()
                {
                        cout<<a<<endl;  // error.  Not accessible
                        cout<<b<<endl; // o.k. protected
                        cout<<c<<endl; // o.k. becomes protected
                        }
                        };
```

```
class three : protected  two
{
        int x;
        public :
        three()
        {
                x=100;
        }
        void show2()
        {
                cout<<x<<endl; // o.k. its own member
                cout<<p<<endl; // error. Not accessible
                cout<<b<<endl; // o.k. protected
                cout<<c<<endl; // o.k. has become protected

        }
};
```

```
int main()
{
        three ob;
        cout<<ob.c<<endl; // error has become protected not available
        cout<<ob.b<<endl; // error.  Not available
        ob.show();   // error.  Has become protected not available
        ob.show1();  // error.  Has become protected not available
        ob.show2(); // O.K.
        return 0;
}
```

```
#include<iostream.h>
class one
{
        int a;   // only for class members
        protected:
        int b;    // for class members and derived classes
        public:
        int c;   // for class members, derived classes, main
        one()
        {
                a=3;
                b=5;
                c=10;
                }
         void show()
         {
                cout<<a<<":"<<b<<":"<<c<<endl;
                }
                };
```

```
class two :private  one
{
        int p;
        public:
        two()
        {
                p=25;
                }
        void show1()
        {
                cout<<p<<endl; // o.k. its own member
                cout<<a<<endl;  // error.  Not accessible
                cout<<b<<endl; // error.  has become private .

                cout<<c<<endl; // error .  has become private
                                }
                };
```

```
class three : private   two
        {
                int x;
                public :
                three()
                {
                x=100;
                }
        void show2()
                {
cout<<x<<endl; // o.k. its own member
cout<<p<<endl; // error. Not accessible
cout<<b<<endl; // error.  not available
cout<<c<<endl; // error. not available
                }
                };
```

```
int main()
{
        three ob;
        cout<<ob.c<<endl; // error not available
        cout<<ob.b<<endl; // error.  Not available
        ob.show();   // error.   not available
        ob.show1();  // error . not available
        ob.show2(); // o.k. its own member
        return 0;
}
```