**INHERITANCE IN DOTNET(C#)**

**Inheritance**

Creating a new class from existing class is called as inheritance. 

    //All the car properties can be used by the supercar

    class SuperCar : car

    {

    }

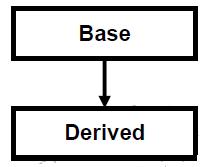
When a new class needs same members as an existing class, then instead of creating those members again in new class, the new class can be created from existing class, which is called as inheritance.  
  
  
**Syntax**

**[Access Modifier] class ClassName : baseclassname**

**{**

**}**

**Types of inheritance**

1. Single Inheritance
2. Hierarchical Inheritance
3. Multi Level Inheritance
4. Hybrid Inheritance
5. Multiple Inheritance
6. **Single Inheritance**  
     
   when a single derived class is created from a single base class then the inheritance is called as single inheritance.  
     
     
   

**//Example program demonstrates singular inheritance**

using System;

namespace ProgramCall

{

    class BaseClass

    {

        //Method to find sum of give  2 numbers

        public int FindSum(int x, int y)

        {

            return (x + y);

        }

        protected void Print(int x, int y)

        {

            Console.WriteLine("First Number: " + x);

            Console.WriteLine("Second Number: " + y);

        }

    }

    class Derivedclass : BaseClass

    {

        public void Print3numbers(int x, int y, int z)

        {

            Print(x, y); //We can directly call baseclass members

            Console.WriteLine("Third Number: " + z);

        }

    }

    class MainClass

    {

        static void Main(string[] args)

        {

            Derivedclass instance = new Derivedclass();

            instance.Print3numbers(30, 40, 50); //Derived class internally calls base class method.

            int sum = instance.FindSum(30, 40); //calling base class method with derived class instance

            Console.WriteLine("Sum : " + sum);

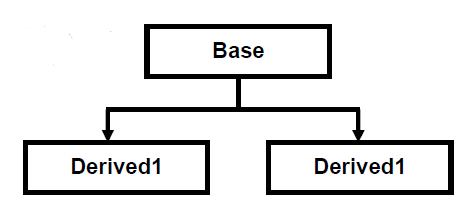
            Console.Read();

        }

    }

}

**Output**  
  
First Number: 30  
Second Number: 40   
Third Number: 50   
Sum : 70

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


**//Example program demonstrates Hierarchical inheritance**

class HeadOffice{

public void HeadOfficeAddress(){

Console.WriteLine("Head Office Address");

}

}

class BranchOffice1 : HeadOffice{

public void BranchOfficeAddress(){

Console.WriteLine("Branch Office Address");

}

}

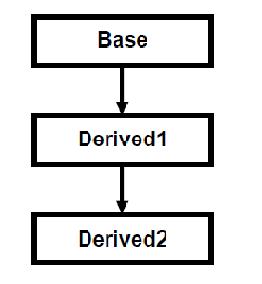
class BranchOffice2 : HeadOffice{

public void BranchOfficeAddress(){

Console.WriteLine("Branch Office Address");

}

}

**3. Multi Level Inheritance**  
when a derived class is created from another derived class, then that inheritance is called as multi level inheritance.  
  


**//Example program demonstrates Multi Level inheritance**

class HeadOffice{

public void HeadOfficeAddress(){

Console.WriteLine("Head Office Address");

}

}

class BranchOffice : HeadOffice{

public void BranchOfficeAddress(){

Console.WriteLine("Branch Office Address");

}

}

class Employee : BranchOffice {

public void NameofEmployee(){

Console.WriteLine("Name of the Employee");

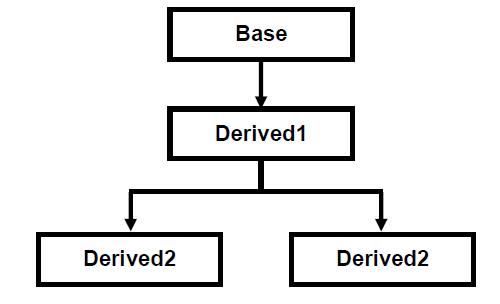
}

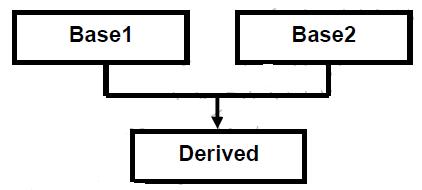
public void Salary(){

Console.WriteLine("Salary of the Employee");

}

}

**4. Hybrid Inheritance**  
Any combination of single, hierarchical and multi level inheritances is called as hybrid inheritance.  
  
  


**5. Multiple Inheritance**  
when a derived class is created from more than one base class then that inheritance is called as multiple inheritance. But multiple inheritance is not supported by .net using classes and can be done using interfaces.  
  


**//Example program demonstrates Multiple inheritance**

using System;

//Example Program for Multiple Inheritance

namespace ProgramCall

{

    //The Icar interface should defines all car properties

    interface Icar

    {

        int WheelsCount();

    }

    //The IPlane interface should defines all plane properties

    interface IPlane

    {

        bool CanFly

        {

            get;

        }

    }

    //The superCar class should implement Icar and Iplane interfaces to become supercar

    class SuperCar : Icar, IPlane

    {

        //The class should implement both intefaces

        public int WheelsCount()

        {

            return 4;

        }

        public bool CanFly

        {

            get

            {

                return true;

            }

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            //Creating instance for SuperCar class

            SuperCar mysupercar = new SuperCar();

            Console.WriteLine("My Super Car has " + mysupercar.WheelsCount() + " Wheels and can fly is " + mysupercar.CanFly);

            Console.ReadLine();

        }

    }

}

//Output

//My Super Car has 4 Wheels and can fly is True

**Output**   
My Super Car has 4 Wheels and can fly is True