CLASS : BCCA - III (SEMESTER-VI)

SUBJECT : C#.NET

TOPIC : INHERITANCE

UNIT-III

INHERITANCE

- Inheritance is a fundamental concept in Object-Oriented programming. Many objects exist that share a lot of commonality between them. By using inheritance, subclasses "inherit" the public variables and methods of the super class, in addition to their own variables and methods

- A relationship between a more general class (called the super class or base class) and a more specialized class (called the subclass or derived class).For example, a Cat is a specific type of Animal. Therefore the Animal class could be the super class/base class, and the Cat class could be a subclass of Animal.

- Acquiring (taking) the properties of one class into another class is called inheritance.

ADVANTAGES OF INHERITANCE

- Inheritance helps to improve code re-use and minimize duplicate code among related classes.
- Inheritance provides reusability by allowing us to extend an existing class.
- Code reusability (save time and money).
- Increasing program reliability.
- Better problem-solving and program design.
- Supporting polymorphism.

Note : Any private method or variables from the base class/ super class/parent class cannot be inherited by the derived class/sub class/child class.

DEFINING A SUBCLASS

A subclass is defined as follows :

Class subclass-name : superclass-name

{

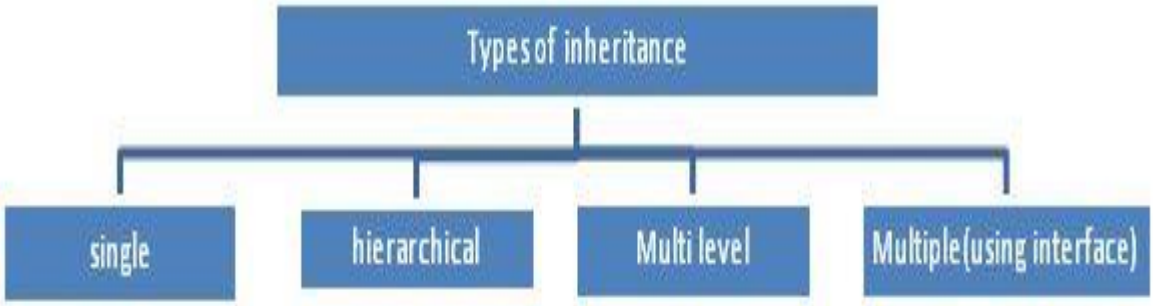
Variables declaration;

Methods declaration;

}

TYPES OF INHERITANCE

The following are the types of inheritance in C#:

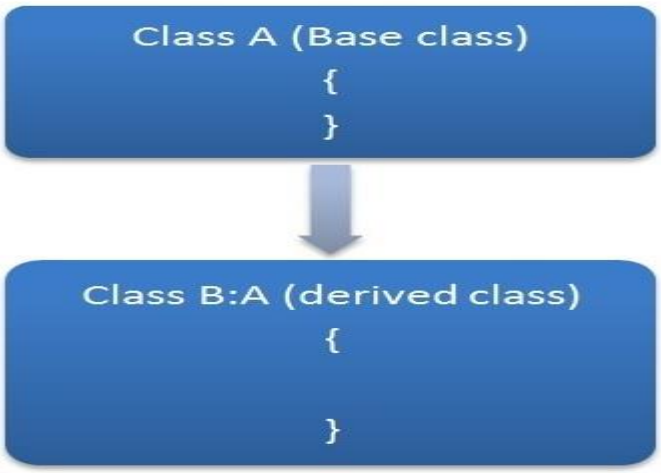


The inheritance concept is based on a base class and derived class.

- Base class/Parent class/ Super class - is the class from which its features are to be inherited into another class.
- Derived class/ Child class / Sub class - it is the class in which the base class features are inherited and also it can have its own features.

SINGLE INHERITANCE

It is the type of inheritance in which there is one base class and one derived class.



Consider the following example illustrating the concept of Simple Inheritance

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Demo
{
    class Item      // base class
    {
        public void Company ( )
        {
                Console.WriteLine (" Item Code = XXX");
        }
    }

    class Fan : Item                //derived class
    {
```

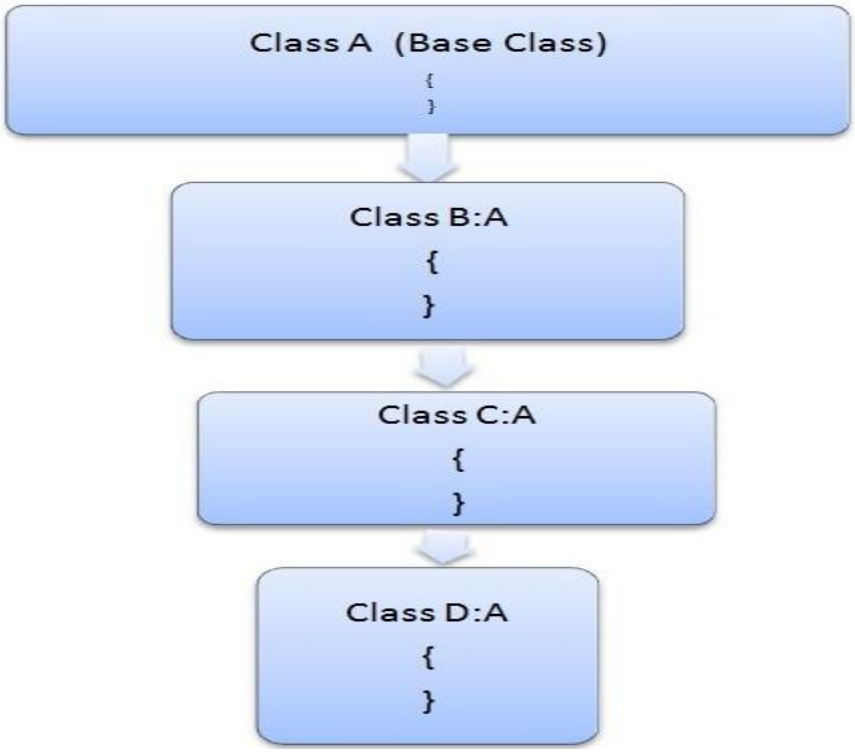
```
    public void Model ( )
    {
        Console.WriteLine ("Fan Model = Classic");
    }
}
class SimpleInheritance
{
    public static void Main(string[] args)
    {
        Item i1 = new Item( );
        Fan f1 = new Fan( );
        i1.Company( );
        f1.Company( );
        f1.Model( );
        Console.Read();
    }
}
}
```

When the above code is compiled and executed, it produces the following result:

```
 Item Code = XXX
 Item Code = XXX
Fan Model = Classic
```

HIERARCHICAL INHERITANCE

This is the type of inheritance in which there are multiple classes derived from one base class. This type of inheritance is used when there is a requirement of one class feature that is needed in multiple classes.



Consider the following example illustrating the concept of Hierarchical Inheritance

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Demo
{
    class A  //base class
    {
       public void msg( )
       {
           Console.WriteLine ( "this is A class Method");
       }
    }
```

```
class B : A   //derived class
{
   public void info( )
   {
      msg( );
      Console.WriteLine ("this is B class Method");
   }
}
   class C : A    //derived class
   {
      public void getinfo( )
      {
         msg( );
         Console.WriteLine ("this is C class Method");
      }
   }
class HierarchicalInheritance
{
         public static void Main(string[] args)
    {
               A a = new A( );
               B b = new B( );
               C c = new C( );
               a.Msg( );
               b.Info( );
```

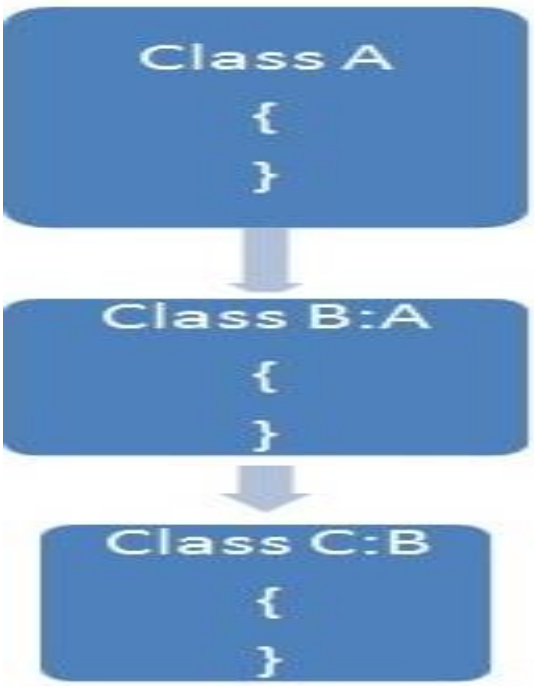
```
            c.getInfo( );
        Console.Read();
    }
  }
}
```

When the above code is compiled and executed, it produces the following result:

this is A class Method
this is A class Method
this is B class Method
this is A class Method
this is C class Method

MULTILEVEL INHERITANCE

When one class is derived from another derived class then this type of inheritance is called multilevel inheritance.



Consider the following example illustrating the concept of Multilevel inheritance

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Demo
{

class Grandfather   //base class
{
        public void Display()
      {
              Console.WriteLine("Grandfather...");
       }
  }
 class Father : Grandfather    //derived class of base class- Grandfather
{
       public void DisplayOne()
    {
           Console.WriteLine("Father...");
     }
  }

class Son : Father    //derived class of base class - Father
{
       public void DisplayTwo()
    {
         Console.WriteLine("Son.. ");
     }
}
```

```
    static void Main(string[] args)
{
        Son s = new Son();
        s.Display();
        s.DisplayOne();
        s.DisplayTwo();
        Console.Read();
    }
 }
```

When the above code is compiled and executed, it produces the following result:

```
Grandfather...
Father...
Son..
```