CLASS : BCCA - III (SEMESTER-VI)

SUBJECT : C#.NET

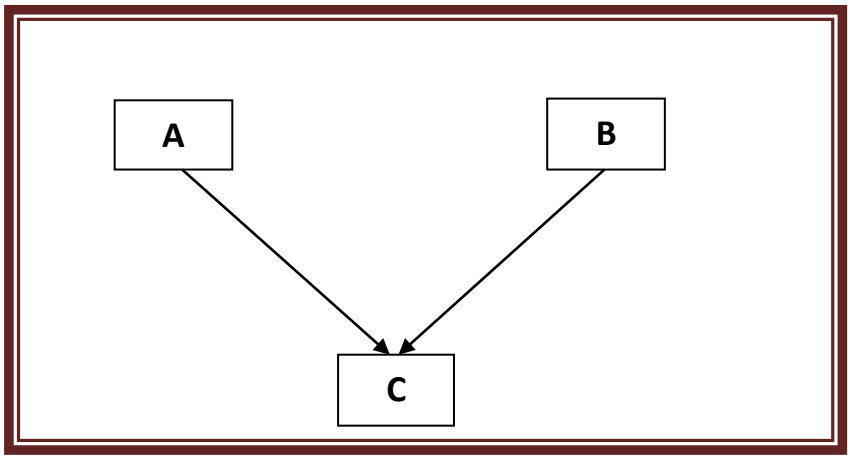
TOPIC : INTERFACE

UNIT-IV

MULTIPLE INHERITANCE

- In Multiple Inheritance there are several base classes. The derived class inherits the properties of these several base classes.

- In C# implementation of Multiple Inheritance proves difficult and adds complexity to the language. C# provides alternate approach known as Interface to support the concept of multiple inheritance.



Multiple Inheritance

MULTIPLE INHERITANCE USING INTERFACE

- In C#, an interface can be defined using the **interface** keyword. Interfaces can contain methods, properties, indexers, and events as members.

- Interfaces will contain only the declaration of the members. The implementation of the interface's members or to define the code for implementation of these members will be given by class who implements the interface.

- In C#, we cannot use any access modifier for any member of an interface. By default all the members of Interface are public and abstract

INTERFACE

An Interface can contain one or more methods, properties, indexers and events but none of them are implemented in the interface itself. It is the responsibility of the class that implements the interface to define the code for implementation of these members.

Syntax for Interface Declaration

```
Interface interfacename
{
  Member declarations;
}
```

IMPLEMENTING INTERFACES

Interfaces are used as 'superclasses' whose properties are inherited by classes. It is therefore necessary to create a class that inherits the given interface.

Syntax for Implementing Interface

```
class classname : interfacename
{
        Body of classname
}
```

Syntax for Implementing Multiple Interfaces

```
class classname : interface1, interface2,. . . .
{
        Body of classname
}
```

- **C# code illustrating the example of Implementation of Multiple Interfaces**

```
interface IA    //interface1
{
   string setImgs(string a);
}
interface IB     //interface2
{
   int getAmount(int Amt);
}
class ICar : IA, IB   //implementation
{
   public int getAmount(int Amt)
   {
      return 100;
   }
   public string setImgs(string a)
   {
      return "this is the car";
   }
}
```

In the above coding the ICar class inherits the features of the two interfaces hence this type of inheritance is called Multiple Inheritance.

- **Program demonstrating how multiple inheritance can be achieved in C# using Interface Concept.**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;


namespace MultipleInheritApplication
{
   interface calc1
   {
      int add(int a, int b);
   }
   interface calc2
   {
      int sub(int x, int y);
   }
   interface calc3
   {
      int mul(int r, int s);
   }
   interface calc4
   {
```

```
    int div(int c, int d);
}
class Calculation : calc1, calc2, calc3, calc4
{
    public int result1;
    public int add(int a, int b)
    {
        return result1 = a + b;
    }
    public int result2;
    public int sub(int x, int y)
    {
        return result2 = x - y;
    }
    public int result3;
    public int mul(int r, int s)
    {
        return result3 = r * s;
    }
    public int result4;
    public int div(int c, int d)
    {
        return result4 = c / d;
    }
```

```
    class Program
    {
        static void Main(string[] args)
        {
            Calculation c = new Calculation();
            c.add(8, 2);
            c.sub(20, 10);
            c.mul(5, 2);
            c.div(20, 10);
            Console.WriteLine("Multiple Inheritance concept Using Interfaces :\n ");
            Console.WriteLine("Addition : " + c.result1);
            Console.WriteLine("Substraction : " + c.result2);
            Console.WriteLine("Multiplication :" + c.result3);
            Console.WriteLine("Division : " + c.result4);
            Console.ReadKey();
        }
    }
  }
}
```

Output

Multiple Inheritance concept Using Interfaces :
Addition : 10
Substraction : 10
Multiplication : 10
Division : 2