Day 11 Assignment

By

Nanam Vaishnavi

07-Feb-2022

1) Research and write the difference between abstract class and interface in C#

Abstract Class	Interface
1) An Abstract class doesn't provide full abstraction.	1) An Interface will provide full abstraction
2) We can't achieve multiple inheritance.	2) We can achieve multiple inheritance.
3) We can declare a member field.	3) We can't declare a member field.
4) Abstract class contains access modifiers for the sub functions and properties.	4) Interface doesn't contain access modifiers because in interface everything will be public.
5) A class may inherit only one abstract class.	5) A class may inherit several times.
6) An abstract class can be defined.	6) An interface member can't be defined using keywords like static, public etc.
7) An abstract class can provide complete, default code and just the details that have to be overridden.	7) An interface doesn't provide any code, just the signature.

2)Write the 6 points about interface discussed in the class

1) Interface is a pure abstraction class.
2) Interface name should starts with "I".
3) Interface acts as a **Contractor.**
   Abstract class acts as a **Template.**
4) By default, the methods are public and abstract class.
5) Any class that is implementing interface must override all the methods.
6) Interface supports Multiple Inheritance.

3) Write example program for interfaces discussed in the class IShape include the classes Cricle, Square, Triangle, Rectangle

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
// ***********************************************************************
// Author : Nanam Vaishnavi
// Purpose : class IShape  including Square, Circle,Rectangle,Triangle
// ************************************************************************

namespace Interface
```

```
{
    interface IShape
    {
        int CalculatePerimeter();
        int CalculateArea();
    }
    class Square : IShape
    {
        private int side;
        public void ReadSide()
        {
            Console.WriteLine(" Enter  side: ");
            side = Convert.ToInt32(Console.ReadLine());
        }

        public int CalculatePerimeter()
        {
            return 4 * side;
        }

        public int CalculateArea()
        {
            return side * side;
        }

    }
    class Circle : IShape
    {
        private int radius;

        public void ReadRadius()
        {
            Console.WriteLine(" Enter Radius: ");
            radius = Convert.ToInt32(Console.ReadLine());
        }
        public int CalculatePerimeter()
        {
            return 22 * radius * radius / 7;
        }
        public int CalculateArea()
        {
            return 2 * 22 * radius / 7;
        }
    }
    class Rectangle : IShape
    {
        private int length;
        private int breadth;


        public void ReadLength()
        {
            Console.WriteLine("Enter Length:  ");
            length = Convert.ToInt32(Console.ReadLine());
        }
        public void ReadBreadth()
        {
            Console.WriteLine("Enter Breadth:  ");
            breadth = Convert.ToInt32(Console.ReadLine());
        }

        public int CalculatePerimeter()
        {
```

```
            return 2 * (length + breadth);
        }
        public int CalculateArea()
        {
            return length * breadth;
        }
    }
    class Triangle : IShape
    {

        private int Side1;
        public void ReadSide1()
        {
            Console.WriteLine("Enter side 1");
            Side1 = Convert.ToInt32(Console.ReadLine());
        }
        private int Side2;
        public void ReadSide2()
        {
            Console.WriteLine("Enter side2");
            Side2 = Convert.ToInt32(Console.ReadLine());
        }
        private int Side3;

        public void Readside3()
        {
            Console.WriteLine("Enter side3");
            Side3 = Convert.ToInt32(Console.ReadLine());
        }

        public int CalculateArea()
        {
            float s = (Side1 + Side2 + Side3) / 2;
            int Area = (int)Math.Sqrt((s * (s - Side1) * (s - Side2) * (s -
Side3)));
            return Area;
        }

        public int CalculatePerimeter()
        {
            return Side1 + Side2 + Side3;
        }
        internal class Program
        {
            static void Main(string[] args)
            {
                Square s = new Square();
                s.ReadSide();
                Console.WriteLine(s.CalculatePerimeter());
                Console.WriteLine(s.CalculateArea());

                Circle c = new Circle();
                c.ReadRadius();
                Console.WriteLine(c.CalculatePerimeter());
                Console.WriteLine(c.CalculateArea());

                Rectangle r = new Rectangle();
                r.ReadLength();
                r.ReadBreadth();
                Console.WriteLine(r.CalculatePerimeter());
                Console.WriteLine(r.CalculateArea());

                Triangle tr = new Triangle();
```

```
                tr.ReadSide1();
                tr.ReadSide2();
                tr.Readside3();

                Console.WriteLine(tr.CalculatePerimeter());
                Console.WriteLine(tr.CalculateArea());
                Console.ReadLine();
            }
        }
    }
}
```

OUTPUT

F:\NH\DotNetProjects\Day 11 Assignment\Interface\Interface\b

```
 Enter  side:
5
20
25
 Enter Radius:
6
113
37
Enter Length:
4
Enter Breadth:
3
14
12
Enter side 1
5
Enter side2
6
Enter side3
8
19
10
```

4) Write the 7 points discussed about properties.

1) Properties are similar to class variables with get; & set; methods.

2) A Property with only get; is **Read Only**

3) A Property with only set; is **Write Only**

4) A Property with get; & set; is **we can read values and assign values.**

History of properties

5) Property are introduced to deal with private variables.

6) Property must start with **UpperCase.**

7)Simple example of properties are

```
Class employee
{
    Private int id;
    Private string name;
     Private string designation;
Public int id
  {
     get{return id;}
     Set{id=value;}
   }

}
```

5. Write sample code to illustrate properties as discussed in class.

id
name
designation
salary
id-get, set
name-get,set
designation-set (writeonly)
salary-get (get with some functionality)

CODE

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
// ****************************************************
// Author : Nanam Vaishnavi
// Purpose : class Employee using properties
// ****************************************************
namespace Property_Example
{
    class Employee
    {
        private int id;
        private string name;
        private string designation;
        private int salary;

        public int Id
        {
            get { return id; }
            set { id = value; }
        }
        public string Name
        {
            get { return name; }
            set { name = value; }
```

```
        }
        public string Designation
        {
            set { designation = value; }
        }
        public int Salary
        {
            get
            {
                salary = (designation == "Software Employee") ? 30000 : 60000;
                return salary;
            }

        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Employee emp = new Employee();
            emp.Designation = "M";
            Console.WriteLine(emp.Salary);

            Console.ReadLine();
        }
    }
}
```

OUTPUT

F:\NH\DotNetProjects\Day 11 Assignment\Property Example\Property Example\bin\Debug\Property Examp

```
60000
```

6) Create a class Employee with only properties.

CODE

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
//*****************************************
// Author : Nanam Vaishnavi
// Purpose : Employee with only properties.
//*****************************************

namespace EmployeeProperties
{
    class Employee
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Designation { get; set; }
        public int Salary { get; set; }
    }
    internal class Program
    {
```

```
        static void Main(string[] args)
        {
            Employee emp = new Employee();
            emp.Id = 556;
            Console.WriteLine($"id={emp.Id}");
            emp.Name = "Vaishnavi";
            Console.WriteLine($"name = {emp.Name}");
            emp.Designation = "Programmer";
            Console.WriteLine($"designation = {emp.Designation}");
            emp.Salary = 80000;
            Console.WriteLine($"salary= {emp.Salary}");

            Console.ReadLine();
        }
    }
}
```

OUTPUT

F:\NH\DotNetProjects\Day 11 Assignment\

```
id=556
name = Vaishnavi
designation = Programmer
salary= 80000
```

7) Create Mathematics class and add 3 static methods and call the methods in main method.

CODE

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
// *********************************************************
// Author : Nanam Vaishnavi
// Purpose : Mathematics class and add static methods.
//*********************************************************

namespace StaticMethods
{
    class Mathematics
    {
        public static int Add(int a, int b)
        {
            return a + b;
        }
        public static int Sub(int a, int b)
        {
            return a - b;
        }
        public static int Mul(int a, int b)
        {
            return (a * b);
        }
```

```
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(Mathematics.Add(6,5));
            Console.WriteLine(Mathematics.Sub(9,4));
            Console.WriteLine(Mathematics.Mul(6, 8));

            Console.ReadLine();
        }
    }
}
```

OUTPUT

```
F:\NH\DotNetProjects\Day 11 Assignment\StaticMethod
11
5
48
```

8) Research and understand when to create static methods.

1) We can’t inherit a static class from another class.
2) We use static method whenever the method is independent on creation and is not using any instance variables.
3) Definition of the class should not be changed or overridden.