Day 11(07-02-2022) Assignment
By
Sudha Kumari Sugasani

Q1.Research and write the difference between Abstract class and Interface in C#

Abstract Class	Interface
1.Multiple Inheritance is not achieved by Abstract Class.	1.Multiple Inheritance is achieved by Interface.
2.It contains Constructors.	2.It doesn't contain Constructors.
3.It contain static members.	3.It doesn't contain static members.
4.It contain different types of access modifiers like public,private,protected,etc..	4.It contain public access modifier only because by default it is public in Interface.
5.The performance of an Abstract class is fast	5.The performance of an Interface is slow because it require time to search actual method in corresponding class.
6.It can be fully , partially,not implemented.	6.It can be fully implemented.

Q2.Write the six points about Interfaces discussed in the class.

1.Interface is pure Abstract class.
2.Iterface name must starts with 'I'.
3.Interface acts like contract where Abstract class acts like a template.
4.By default the methods in the Interfaces are public and abstract.
5.Any class that is implementing the Interface must override all the methods(abstract).
6.Interfaces support multiple Inheritance.

Q3.Write the example program for Inheritance discussed in the class.
Ishape include the classes
Circle,Square,Triangle,Rectangle

Code:

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

namespace Day10project1
{

/*****************************************************************************
     * Authhor:Sudha Kumari Sugasani
     * Purpose:Creating an Interface name with IShape which imclude the classes
     *          Circle,Square,Traingle,Rectangle
     *
*****************************************************************************/
    interface IShape
    {
        int CaluculatePerimeter();
        int CaluculateArea();

    }
    class Circle:IShape
```

```
    {

        private int radius;
        /// <summary>
        /// This method will read the input from the user.
        /// </summary>
        public void ReadRadius()
        {
            Console.WriteLine("Enter radius");
            radius =Convert.ToInt32(Console.ReadLine());
        }
        /// <summary>
        /// This method will caluculate the Perimeter of the Circle
        /// </summary>
        /// <returns>Perimeter of the Circle</returns>
        public int CaluculatePerimeter()
        {
            return 2*22*radius/7;
        }
        /// <summary>
        /// This method will caluculate the Area of the Circle
        /// </summary>
        /// <returns>Area of the Circle</returns>
        public int CaluculateArea()
        {
            return 22*radius*radius/7;
        }

    }
    class Square : IShape
    {

        private int side;
        /// <summary>
        /// This method will read the input from the user.
        /// </summary>
        public void ReadSide()
        {
            Console.WriteLine("Enter side");
            side = Convert.ToInt32(Console.ReadLine());
        }
        /// <summary>
        /// This method will caluculate the Perimeter of the Square
        /// </summary>
        /// <returns>Perimeter of the Square</returns>
        public int CaluculatePerimeter()
        {
            return 4*side;
        }
        /// <summary>
        /// This method will caluculate the Area of the Square
        /// </summary>
        /// <returns>Area of the Square</returns>
        public int CaluculateArea()
        {
            return side*side;
        }

    }
    class Rectangle : IShape
    {

        private int length;
```

```
        public int breadth;
        /// <summary>
        /// This method will read the input from the user.
        /// </summary>
        public void ReadLengthandBreadth()
        {
            Console.WriteLine("Enter Length");
            length= Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Breadth");
            breadth= Convert.ToInt32(Console.ReadLine());
        }
        /// <summary>
        /// This method will caluculate the Perimeter of the Rectangle
        /// </summary>
        /// <returns>Perimeter of the Rectangle</returns>
        public int CaluculatePerimeter()
        {

            return (length+breadth)*2;
        }
        /// <summary>
        /// This method will caluculate the Area of the Rectangle
        /// </summary>
        /// <returns>Area of the Circle</returns>
        public int CaluculateArea()
        {
            return (length*breadth);
        }

    }
    class Triangle : IShape
    {

        private int side1;
        private int side2;
        private int side3;
        public int semiparameter;


        /// <summary>
        /// This method will read the input from the user.
        /// </summary>
        public void ReadSides()
        {
            Console.WriteLine("Enter Side1");
            side1 = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Side2");
            side2= Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Side3");
            side3= Convert.ToInt32(Console.ReadLine());
            semiparameter = (side1 + side2 + side3) / 2;


        }


        /// <summary>
        /// This method will caluculate the Perimeter of the Triangle
        /// </summary>
        /// <returns>Perimeter of the Traingle</returns>
        public int CaluculatePerimeter()
        {
            int perimeter;
```

```
            perimeter=(side1+side2+ side3);
            return perimeter;
        }
        /// <summary>
        /// This method will caluculate the Area of the Triangle
        /// </summary>
        /// <returns>Area of the Traingle</returns>
        public int CaluculateArea()
        {
            int Area;
            int s1,s2,s3;
            s1 = (semiparameter - side1);
            s2= (semiparameter - side2);
            s3= (semiparameter - side3);
            Area = (int)Math.Sqrt((semiparameter)* (s1*s2*s3));
            //Area = (int)Math.Sqrt((semiparameter) * ((semiparameter - side1)
* (semiparameter - side2) * (semiparameter - side3)));
            return Area;
        }

    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Circle c1=new Circle();
            c1.ReadRadius();
            Console.WriteLine($"Area of Circle is {c1.CaluculateArea()}");
            Console.WriteLine($"Perimeter of Circle is
{c1.CaluculatePerimeter()}");
            Square s1=new Square();
            s1.ReadSide();
            Console.WriteLine($"Area of Square is {s1.CaluculateArea()}");
            Console.WriteLine($"Perimeter of Square is
{s1.CaluculatePerimeter()}");
            Rectangle r1=new Rectangle();
            r1.ReadLengthandBreadth();
            Console.WriteLine($"Area of Rectangle is {r1.CaluculateArea()}");
            Console.WriteLine($"Perimeter of Rectangle is
{r1.CaluculatePerimeter()}");
            Triangle t1=new Triangle();
            t1.ReadSides();
            Console.WriteLine($"Area of Triangle is {t1.CaluculateArea()}");
            Console.WriteLine($"Perimeter of Triangle is
{t1.CaluculatePerimeter()}");
            Console.ReadLine();
        }
    }
}
```

Output:

C:\NH\.NET Projects\Day10project1\Day10project1\bin\Debug\

```
Enter radius
2
Area of Circle is12
Perimeter of Circle is 12
Enter side
4
Area of Square is 16
Perimeter of Square is 16
Enter Length
2
Enter Breadth
3
Area of Rectangle is 6
Perimeter of Rectangle is 10
Enter Side1
2
Enter Side2
3
Enter Side3
4
Area of Triangle is 0
Perimeter of Triangle is 9
```

Q4.Write the seven points discussed about the properties.

1.Properties are almost like class variables with get; and set;
2.A property with only get is readonly.
3.A property with only set is writeonly.
4.A property with both get; and set; will read the value and assign the value.
5.Properties are introduced to deal with private variables.
6.Properties names starts with uppercase.
7.Example of properties:

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

namespace Day11project2
{
    internal class Program
    {

/*****************************************************************************
         * Author:Sudha Kumari Sugasani
         * Purpose:Example program for properties
         *
*****************************************************************************/
        class Circle
        {
            public int Radius { get; set; }
            public int Area
            {
                get
                {
                    return 22 * Radius * Radius / 7;
                }
            }
```

```
            public int Perimeter
            {
                get
                {
                    return 2 * 22 * Radius / 7;
                }
            }

        }
            static void Main(string[] args)
            {
                Circle c1 = new Circle();
            c1.Radius = 5;
            Console.WriteLine($"Radius of Circle is {c1.Radius}");
            Console.WriteLine($"Area of Circle is {c1.Area}");
            Console.WriteLine($"perimeter of Circle is {c1.Perimeter}");
            Console.ReadLine();
            }
        }
    }
```

Output:

```
C:\NH\.NET Projects\Day11project2\Day1
Radius of Circle is 5
Area of Circle is 78
perimeter of Circle is 31
```

Q5.Write sample code to illustrate properties as discussed in the class.

Id;
Name;
Designation;
Salary;
Id-get,set
Name-get,set
Designation-set(writeonly);
Salary-get(get with some functionality)

Code:

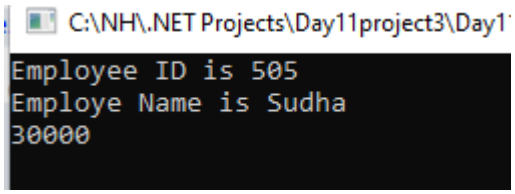
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day11project3
{
    /*****************************************************************
     * Author:Sudha Kumari Sugasani
     * Purpose:Creating a class with properties to access private
     *         Variables.
     *****************************************************************/
    class Employe
    {
        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
        {
            get { return designation; }
            set { designation = value; }
        }
        public int Salary
        {
            get
            {
                salary = (designation == "S") ? 30000 : 60000;
                return salary;
            }
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Employe emp = new Employe();
            emp.Id = 505;
            Console.WriteLine($"Employee ID is {emp.Id}");
            emp.Name = "Sudha";
            Console.WriteLine($"Employe Name is {emp.Name}");

            emp.Designation = "S";
            Console.WriteLine(emp.Salary);
            Console.ReadLine();
        }
    }
}
```

Output:



Q6.Create a class Employee with only properties.

Code:

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

```
using System.Threading.Tasks;

namespace Day11project4
{

    /*****************************************
     * Author:Sudha Kumari Sugasani
     * Purpose:Creating a class with properties
     *
     *****************************************/
    class Employe
    {

        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)
            {
                Employe emp = new Employe();
                emp.Id = 505;
                Console.WriteLine($"Employee ID is {emp.Id}");
                emp.Name = "Sudha";
                Console.WriteLine($"Employe Name is {emp.Name}");
                emp.Designation = "Developer";
                Console.WriteLine($"Employee Designation is
{emp.Designation}");
                emp.Salary = 30000;
                Console.WriteLine($"Saalry of Employee is {emp.Salary}");
                Console.ReadLine();
            }
        }
    }
```

Output:

C:\NH\.NET Projects\Day11project4\Day11project4\bi

```
Employee ID is 505
Employe Name is Sudha
Employee Designation is Developer
30000
```

Q7.Create Mathematics class and add three 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;

namespace Day11project5
{

    /*************************************************************************
     * Author:Sudha Kumari Sugasani
     * Purpose:Creating a Mathematics class with three static methods and call
```

```
     *         them in main method.
     ***********************************************************************/
    class Mathematics
    {
        /// <summary>
        /// This is a static method
        /// It will return Sum of two numbers
        /// </summary>
        /// <param name="a">a</param>
        /// <param name="b">b</param>
        /// <returns>Sum</returns>
        public static int Add(int a, int b)
        {
            return a + b;
        }
        /// <summary>
        /// This is a static method
        /// It will return difference of two numbers
        /// </summary>
        /// <param name="a">a</param>
        /// <param name="b">b</param>
        /// <returns>Diifference</returns>
        public static int Sub(int a,int b)
        {
            return a - b;
        }
        /// <summary>
        /// This is a static method
        /// It will return product of two numbers
        /// </summary>
        /// <param name="a">a</param>
        /// <param name="b">b</param>
        /// <returns>Product</returns>
        public static int Mul(int a,int b)
        {
            return a * b;
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine($"The sum of two numbers is
{Mathematics.Add(2,3)}");
            Console.WriteLine($"The difference of two numbers is
{Mathematics.Sub(15, 10)}");
            Console.WriteLine($"The product of two numbers is
{Mathematics.Mul(2, 3)}");
            Console.ReadLine();
        }
    }
}
```

Output:

C:\NH\.NET Projects\Day11project5\Day11project5\bin

```
The sum of two numbers is 5
The difference of two numbers is 5
The product of two numbers is 6
```

Q8.Research and understand when to create static methods.

- If a method is not holding any class variables then we can make the method as static.
- If a method is only dealing with static variables then we can make the method as static.
- Whenever you have a function that does not depend on a particular object of that class,we can create static method there.