NB Healthcare Technologies Pvt Ltd

Day 11 Morning Assignment (07 – Feb- 2022)

By

Vamsi Krishna Mandapati

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

Abstract Class :

- Abstract Class Name starts with Normally, It has no Naming Convention.
- Abstract Classes has both normal methods and Abstract methods
- In abstract classes, abstract methods has no body.
- Uses are Code-reusability, and enforce the abstract methods must be overridden in derived class.

Interface:

- Interface Starts with "I", it follows naming conventions.
- Interface is a pure abstract class , it has only abstract methods.
- In Interface, by default the methods are public and abstract

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

Interface :

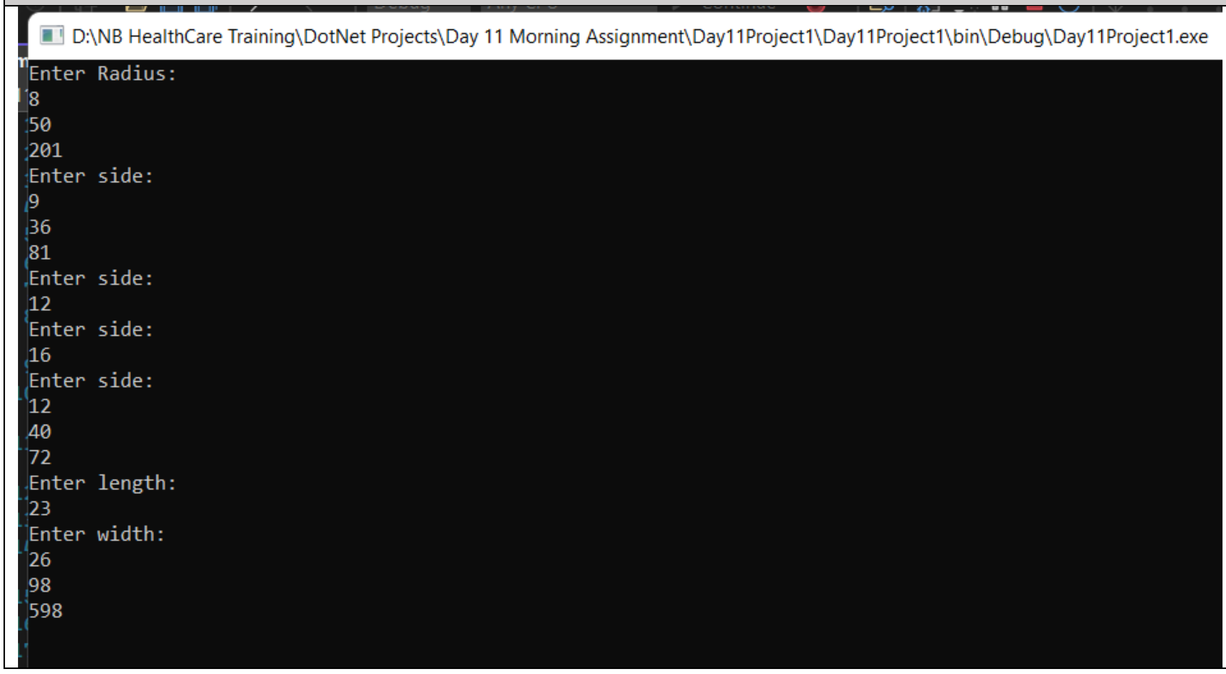
1. Interface is Pure Abstract Class.
2. Interface name should start with I.
3. Interface acts like a contract.
4. By Default, the methods in interface are public and abstract.
5. any class that is implementing interface must override all the methods.
6.Interface support multiple inheritance.

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

Code:

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

namespace Day11Project1
{
    interface IShape
    {
        int CalculatePerimeter();
        int CalculateArea();
    }

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

        public int CalculatePerimeter()
        {
            return 2 * 22 * radius / 7;
         }
    }

    class Square : IShape
    {
        private int side;

        public void ReadSide()
        {
            Console.WriteLine("Enter side:");
            side = Convert.ToInt32(Console.ReadLine());
        }
        public int CalculateArea()
        {
            return side * side;
        }

        public int CalculatePerimeter()
        {
            return 4 * side;
        }
    }
    class Triangle : IShape
    {
        private int side1,side2,side3;
        double semiPerimeter;


        public void ReadSide()
        {
            Console.WriteLine("Enter side:");
            side1 = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter side:");
            side2 = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter side:");
            side3 = Convert.ToInt32(Console.ReadLine());
        }
        public int CalculateArea()
        {
            semiPerimeter = (side1 + side2 + side3) / 2;
            double Area = Math.Sqrt(semiPerimeter * (semiPerimeter - side1)
* (semiPerimeter - side2) * (semiPerimeter - side3));
            return Convert.ToInt32(Area);
        }

        public int CalculatePerimeter()
        {
            return (side1 + side2 + side3);
        }
    }

    class Rectangle : IShape
    {
        private int length,width;

        public void ReadSide()
        {
            Console.WriteLine("Enter length:");
            length = Convert.ToInt32(Console.ReadLine());

            Console.WriteLine("Enter width:");
            width = Convert.ToInt32(Console.ReadLine());
        }
        public int CalculateArea()
        {
            return length * width;
        }

        public int CalculatePerimeter()
        {
            return 2 * (length + width);
        }
    }

    internal class Program
    {
        static void Main(string[] args)
        {
            Circle c = new Circle();
            c.ReadRadius();
            Console.WriteLine(c.CalculatePerimeter());
            Console.WriteLine(c.CalculateArea());

            Square s = new Square();
            s.ReadSide();
            Console.WriteLine(s.CalculatePerimeter());
            Console.WriteLine(s.CalculateArea());

            Triangle t = new Triangle();
            t.ReadSide();
            Console.WriteLine(t.CalculatePerimeter());
            Console.WriteLine(t.CalculateArea());

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

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

OutPut:



D:\NB HealthCare Training\DotNet Projects\Day 11 Morning Assignment\Day11Project1\Day11Project1\bin\Debug\Day11Project1.exe

```
Enter Radius:
8
50
201
Enter side:
9
36
81
Enter side:
12
Enter side:
16
Enter side:
12
40
72
Enter length:
23
Enter width:
26
98
598
```

4. Write the 7 points discussed about properties.

Properties:

1. Properties are almost same as 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 get and set => You can read value and assign value.
5.**History of Properties**:
1.A class will have variables(public,private,static), methods, and Properties. Properties are introduced to deal with private variables.
2.A very 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};
        }
    }
```

3.Property names starts with upper case.

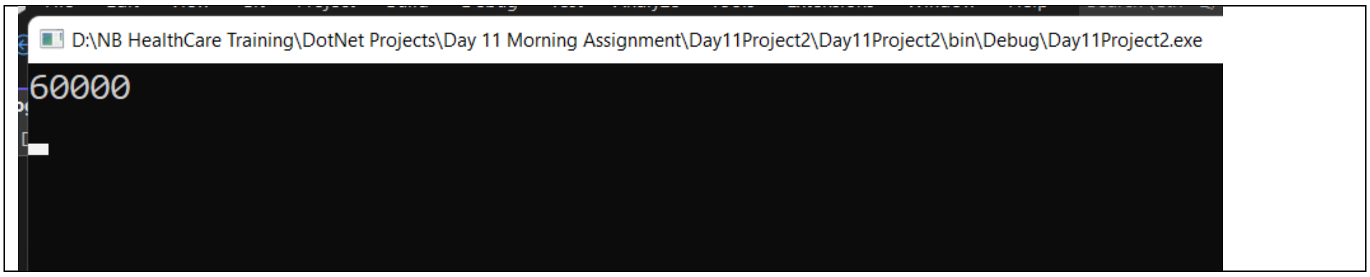
5. Write sample code to illustrate properties as discussed in class.
   id
   name
   designation
   salary

   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;

namespace Day11Project2
{
    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 == "M") ? 30000 : 60000;
                return salary;
            }
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Employee emp = new Employee();
            emp.Designation = "S";
            Console.WriteLine(emp.Salary);

            Console.ReadLine();

        }
    }
}
```

Output:


D:\NB HealthCare Training\DotNet Projects\Day 11 Morning Assignment\Day11Project2\Day11Project2\bin\Debug\Day11Project2.exe

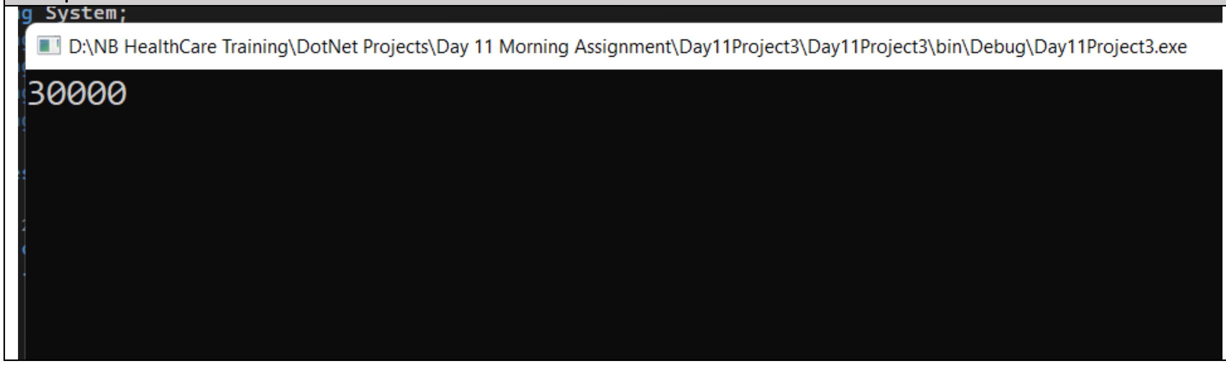
```
60000
```

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;

namespace Day11Project3
{
    class Employee
    {

        public int Id{ get; set; }

        public string Name { get; set; }

        public string Designation { get; set; }

        public int Salary
        {
            get
            {
                if(Designation == "M")
                {
                    return 30000;
                }
                else
                {
                    return 60000;
                }


            }

        }


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



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

Output:



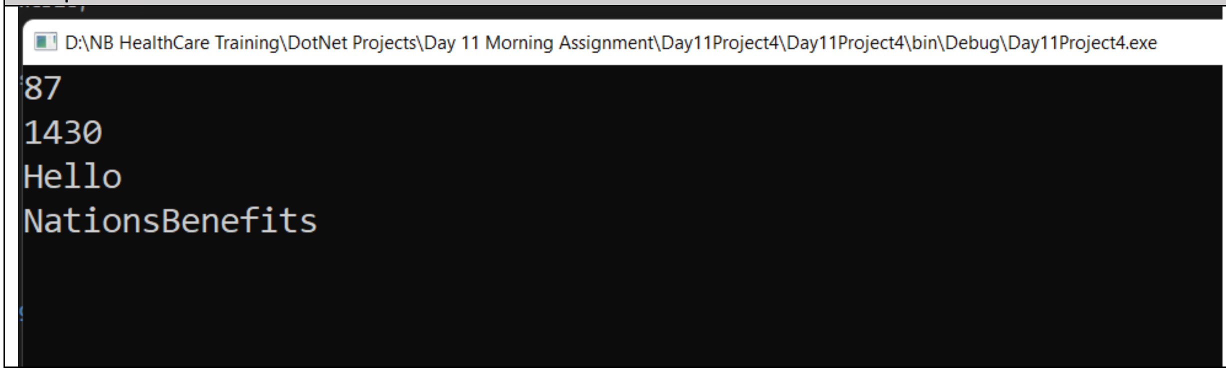
```
D:\NB HealthCare Training\DotNet Projects\Day 11 Morning Assignment\Day11Project3\Day11Project3\bin\Debug\Day11Project3.exe
30000
```

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;

namespace Day11Project4
{
    class Employee
    {
        public int id;
        public string name;
        public static string company = "NationsBenefits";
    }

    class Mathematics : Employee
    {
        public static int Add(int a, int b)
        {
            return a + b;
        }
        public static int Multiplication(int a, int b)
        {
            return a * b;
        }

        public static void PrintCompany()
        {
            Console.WriteLine(company);
        }

        public static void Hello()
        {
            Console.WriteLine("Hello");
        }




    }

    internal class Program
    {

        static void Main(string[] args)
        {
            Console.WriteLine(Mathematics.Add(22,65));
            Console.WriteLine(Mathematics.Multiplication(22, 65));
            Mathematics.Hello();
            Mathematics.PrintCompany();

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

Output:



D:\NB HealthCare Training\DotNet Projects\Day 11 Morning Assignment\Day11Project4\Day11Project4\bin\Debug\Day11Project4.exe

```
87
1430
Hello
NationsBenefits
```

8. Research and understand when to create static methods.

1.If The object is not holding any variables, and if the whole purpose of the object is to only call the methods then we can declare that methods as static.

2. If the Method is not dealing with any class variables , then we can declare that method as static. (if the method is dealing with class variables we can not declare that method as static method)

3.If the method is dealing with static variables alone then we can declare that method as static method.