Day 11 Morning Assignments

By

Praveen Chakravarthi

07-02-2022

NB Health Care

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

Abstract Class	Interface
1. An Abstract Class can be defined by keyword 'Abstract'	1. An interface can be defined using keyword 'Interface'
2. In Abstract class, we cannot achieve multiple inheritance	2. In Interface, we can achieve multiple inheritance
3. It can have both Abstract and Non-Abstract methods	3. It can only have Abstract methods
4. It contains Access Modifiers for functions and properties	4. It cannot have Access Modifiers by default everything is assumed as public
5. Abstract Class contains Constructors	5. Interface doesn't contains Constructors

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

1. Interface is pure Abstract Class
2. Interface Names should start with "I"
3. Interface acts like a contract
4. By default the methods in Interface are public and abstract
5. Interface supports multiple inheritance
6. Any class that is implementing Interface must override all methods

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

Code :

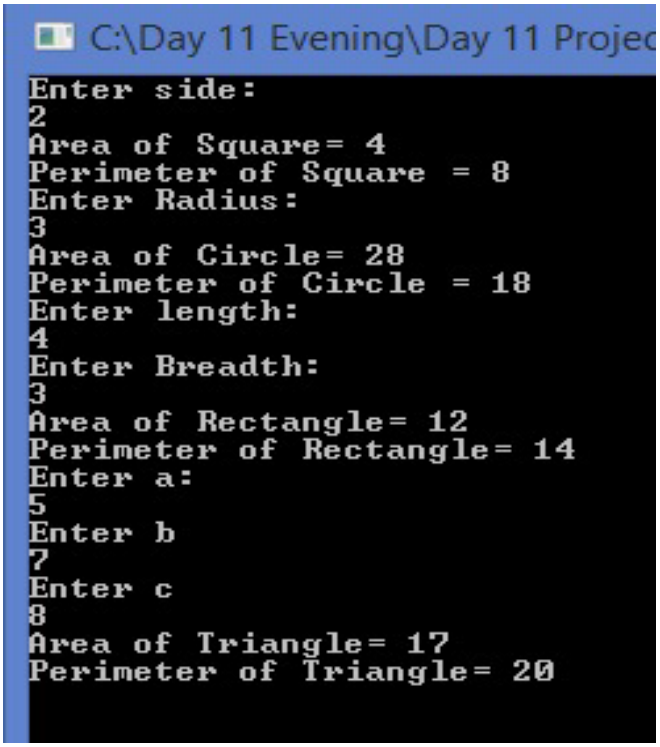
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Day_11_Project_1
{
    // Author : Praveen Chakravarthi
    // Purpose : Interface Class

    interface Ishape
    {
        /// <summary>
        /// This method finds Area of the given Shape
        /// </summary>
        int Area();

        /// <summary>
        /// This Method finds Perimeter of the given Shape
        /// </summary>
        int Perimeter();
    }
    class Square : Ishape
    {
        public int side;

        public void ReadSide()
        {
            Console.WriteLine("Enter side: ");
            side = Convert.ToInt32(Console.ReadLine());
        }
        public int Perimeter()
        {
            return 4 * side;
        }

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

    class Circle : Ishape
    {
        public int radius;

        public void ReadRadius()
        {
            Console.WriteLine("Enter Radius: ");
            radius = Convert.ToInt32(Console.ReadLine());
        }
        public int Perimeter()
        {
            return (2 * 22 * radius) / 7;
        }

        public int Area()
        {
```

```
        return (22 * radius * radius)/7;
    }
}

class Rectangle : Ishape
{
    public int length;
    public int breadth;

    public void ReadLB()
    {
        Console.WriteLine("Enter length: ");
        length = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("Enter Breadth: ");
        breadth = Convert.ToInt32(Console.ReadLine());
    }
    public int Perimeter()
    {
        return 2 * (length+breadth);
    }

    public int Area()
    {
        return length * breadth;
    }
}

class Triangle : Ishape
{
    public int a;
    public int b;
    public int c;
    public int s;

    public void ReadABC()
    {

        Console.WriteLine("Enter a: ");
        a = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("Enter b");
        b = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("Enter c");
        c = Convert.ToInt32(Console.ReadLine());
        s=(a+b+c)/2;
    }
    public int Perimeter()
    {
        return a+b+c;
    }

    public int Area()
    {
        return (int)(Math.Sqrt(s*(s - a)*(s - b)*(s - c))); // Heron's Formula
    }
```

```
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Square sq = new Square();
            sq.ReadSide();
            Console.WriteLine($"Area of Square= {sq.Area()}");
            Console.WriteLine($"Perimeter of Square = {sq.Perimeter()}");

            Circle cr = new Circle();
            cr.ReadRadius();
            Console.WriteLine($"Area of Circle= {cr.Area()}");
            Console.WriteLine($"Perimeter of Circle = {cr.Perimeter()}");

            Rectangle rt = new Rectangle();
            rt.ReadLB();
            Console.WriteLine($"Area of Rectangle= {rt.Area()}");
            Console.WriteLine($"Perimeter of Rectangle= {rt.Perimeter()}");

            Triangle tr = new Triangle();
            tr.ReadABC();
            Console.WriteLine($"Area of Triangle= {tr.Area()}");
            Console.WriteLine($"Perimeter of Triangle= {tr.Perimeter()}");

            Console.ReadLine();

        }
    }
}
```

Output:



4. Write the 7 points discussed about properties.

Properties in C#:
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 both get; and set; is readable and we can assign too

History of C# :
1. Properties are introduced to dea with Private Variables
2. Example of Property :

```
  Class Employee
{
   private int id;
   private string name;

   public int id
   {
     get
      {
      return id ;
       }
      set
      {
      id = value,
      }
   }
}
```

3. Properties names start with Uppercase

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 Day_11_Project_2
{
    // Author : Praveen Chakravarthi
    // Purpose : Sample code for Properties

    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 // Write only
        {
            designation = value;
        }
    }
    public int Salary
    {
        get  // Read only
        {
            salary= (designation == "S")? 30000:50000;
            return salary;
        }
    }

}

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

        Console.WriteLine($"Salary : {emp.Salary}");
        Console.ReadLine();
    }
}
}
```

Output :

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 Day_11_Project_3
{
    // Author : Praveen Chakravarthi
    // Purpose : Employee class with only Properties


    class Employee
    {
        public int Id
        {
            get
            {
                return Id;
            }
            set
            {
                Id = value;
            }
        }
        public string Name
        {
            get
            {
                return Name;
            }
            set
            {
                Name = value;
            }
        }
        public string Designation
        {
            set // Write only
            {
                Designation = value;
            }
        }
        public int Salary
        {
            get
            {
                Salary = (Designation == "S") ? 30000 : 50000;
```

```
            return Salary;
        }

    }
}

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

        Console.WriteLine($"Salary : {emp.Salary}");
        Console.ReadLine();
    }
}
}
```

Output:

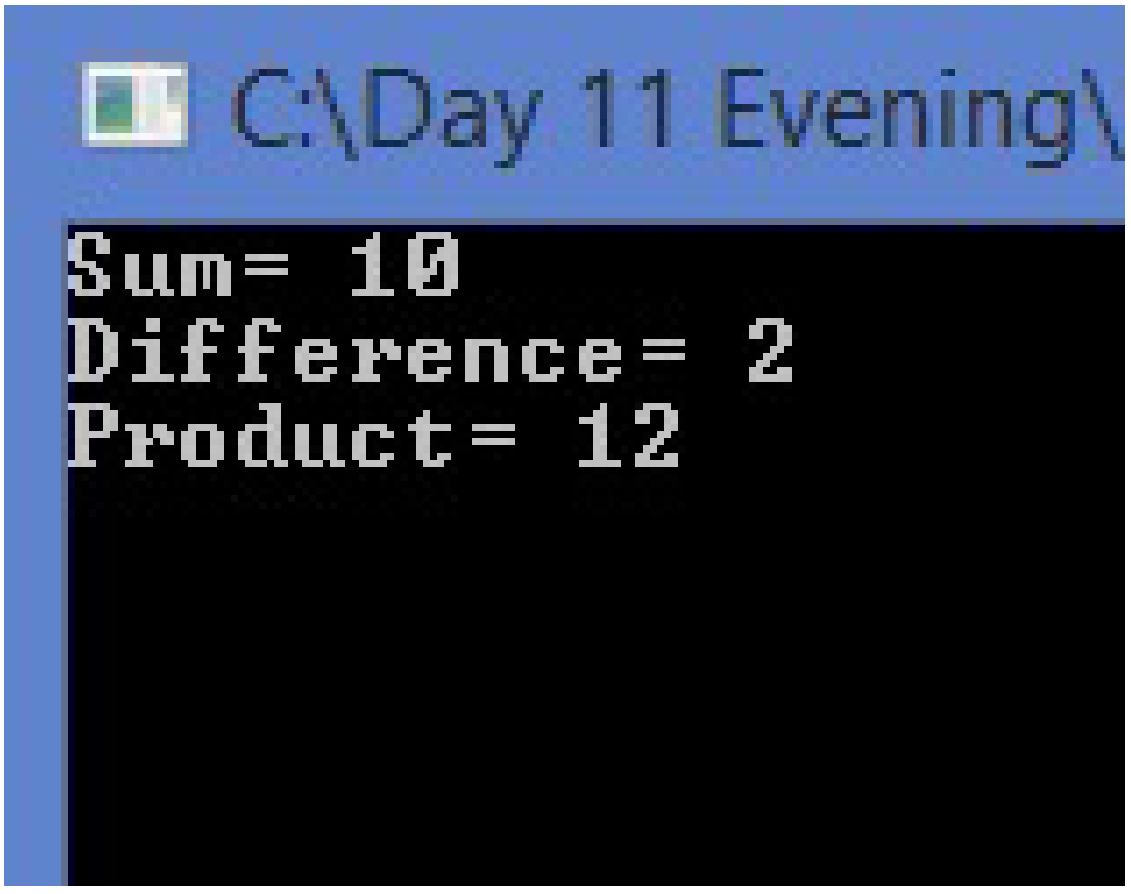
7. Create Mathematics class and add 3 static methods and call the methods in main method.

Code:

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

namespace Day_11_Project_4
{
    // Author : Praveen Chakravarthi
    // Purpose : Mathematics Class with 3 static methods

    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)
        {
            Mathematics mt = new Mathematics();
            Console.WriteLine($"Sum= {Mathematics.Add(6,4)}");
            Console.WriteLine($"Difference= {Mathematics.Sub(5,3)}");
            Console.WriteLine($"Product= {Mathematics.Mul(4,3)}");
            Console.ReadLine();
        }
    }
}
```

Output:



8. Research and understand when to create static methods

1. If a Method is not dealing with any of the class variables then we can create static method
2. If a Method is dealing with static variables then we can use Static Method.
3. If a method is dealing with any of the class variables then we cannot create Static Method