Day 11 Assignment
By
Triveni Anumolu

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

Abstract class	Interface
1.Abstract class contains both declaration and definition part.	1.It contains only declaration part.
2.It contains constructor.	2.It does not contain constructor.
3.Multiple inheritance is not achieved by using abstract class.	3.Multiple inheritance is achieved by using interface.
4.A class can only use one abstract class.	4.A class can use multiple interfaces.
5.It can contain static members.	5. It does not contain static members.

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

- Interface is like a pure abstract class.
- Interface name should start with "I".
- Interface acts like a contract.
- In interface, by default the methods are public and abstract.
- Any class that is implementing interface must override all the methods.
- Interface supports multiple inheritance.

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

Code:

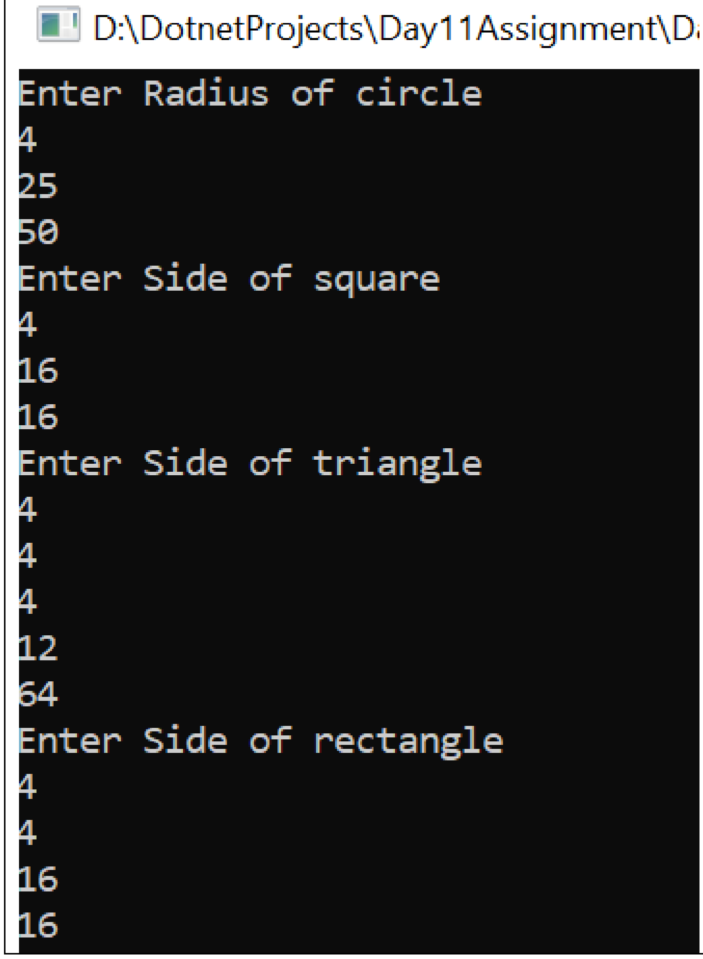
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day11Project2
{
    class Program
    {
```

```
/*****************************************************************
Author: Triveni Anumolu
//Purpose: Creating two classes and methods using Interface
*******************************************************************/
interface IShape
{
   int CalculatePerimeter();
   int CalculateArea();
}

class Circle : IShape
{
   int radius;

   public void ReadRadius()
   {
      Console.WriteLine("Enter Radius of circle");
      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 Readdata()
   {
      Console.WriteLine("Enter Side of square");
      side = Convert.ToInt32(Console.ReadLine());
   }

   public int CalculateArea()
   {

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

```
            return 4 * side;
        }
    }

    class Triangle : IShape
    {
        private int x;
        private int y;
        private int z;

        public void ReadSide()
        {
            Console.WriteLine("Enter Side of triangle");
            x = Convert.ToInt32(Console.ReadLine());
            y = Convert.ToInt32(Console.ReadLine());
            z = Convert.ToInt32(Console.ReadLine());
        }

        public int CalculateArea()
        {

            return x * y * z;
        }
        public int CalculatePerimeter()
        {
            return x + y + z;
        }
    }
    class Rectangle : IShape
    {
        private int length;
        private int breadth;


        public void ReadSide()
        {
            Console.WriteLine("Enter Side of rectangle");
            length = Convert.ToInt32(Console.ReadLine());
            breadth = Convert.ToInt32(Console.ReadLine());
        }

        public int CalculateArea()
        {

            return length * breadth;
        }
```

```
            public int CalculatePerimeter()
            {
                return 2 * (length + breadth);
            }
        }
        internal class program
        {
            static void Main(String[] args)
            {
                Circle c1 = new Circle();
                c1.ReadRadius();
                Console.WriteLine(c1.CalculatePerimeter());
                Console.WriteLine(c1.CalculateArea());
                Square s1 = new Square();
                s1.Readdata();
                Console.WriteLine(s1.CalculatePerimeter());
                Console.WriteLine(s1.CalculateArea());
                Triangle t1 = new Triangle();
                t1.ReadSide();
                Console.WriteLine(t1.CalculatePerimeter());
                Console.WriteLine(t1.CalculateArea());
                Rectangle r1 = new Rectangle();
                r1.ReadSide();
                Console.WriteLine(r1.CalculatePerimeter());
                Console.WriteLine(r1.CalculateArea());

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

Result:



D:\DotnetProjects\Day11Assignment\D

```
Enter Radius of circle
4
25
50
Enter Side of square
4
16
16
Enter Side of triangle
4
4
4
12
64
Enter Side of rectangle
4
4
16
16
```

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 get and set => you can read value and assign the value.
History of properties:
5.Properties are introduced to deal with private variables.

6.A sample example of properties are:
```
class Employee
{
    private int id;
    private string name;
    private string designation;
    public int Id
    {
        get { return id; }
```

```
        set { id = value;}
    }
```

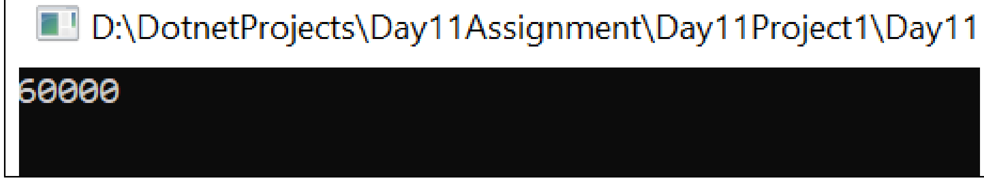
7.Property name starts with uppercase.

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 Day11Project1
{
    /*****************************************************************
     Author: Triveni Anumolu
     Purpose:Code to illustrate properties of employee like id,name,salary, designation
   *****************************************************************/

    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 == "s") ? 30000 : 60000;
                return salary;

            }
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Employee e1 = new Employee();
            e1.Designation = "v";
            Console.WriteLine(e1.Salary);
            Console.ReadLine();
        }
    }
}
```

Result:



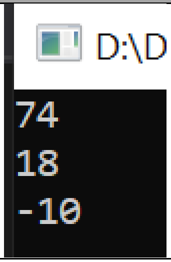
```
D:\DotnetProjects\Day11Assignment\Day11Project1\Day11
60000
```

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 Day11project3
{
    class mathematics
    {
        //Triveni Anumolu
        //Purpose: creating a class with three static methods
        public static int add(int a, int b)
        {
            return a + b;
        }
        public static int mul(int c, int d)
        {
            return c * d;
        }
        public static int sub(int e, int f)
        {
            return e - f;
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(mathematics.add(7, 9));
            Console.WriteLine(mathematics.mul(9, 7));
            Console.WriteLine(mathematics.sub(4, 2));
            Console.ReadLine();
        }
    }
}
```

Result:



```
74
18
-10
```

8.Research and understand when to use static methods.

- If a method is not dealing with any variables of a class we can make it static.
- If a method is dealing with static variables of a class then we can make it static.