07-02-2022, MONDAY
C# OOP'S CONCEPTS ABOUT INTERFACE AND PROPERTIES

Nalli Prudhvi

NB_HEALTH_CARE TECH

==========

Q. Difference between interface and abstract.

Interface	Abstract
1. Interface consists only abstract methods.	1. Abstract class Contains both abstract and non abstract methods.
2. Interface have only static and final variables.	2. Abstract class can have only static , non static , final and non-final variables.
3. Interface it supports multiple inheritance	3. Abstract doesn't support multiple inheritance
4. it has class members like private and protect	4. it has class members public by default
5. The interface can be declared with the interface keyword.	5. To declare abstract class abstract keyword is used

Q. Write 6 points about interface discuss in the class

A.

- **Interface is a pure abstract class.**
- **Interface name should be started with I.**
- **Interface act like a contract.**
- **Any class implementing interface must overwrite all methods**
- **Interface supports multiple inheritance.**

Q. Write the 7 points discussed about properties.

A.

- **Properties are almost same as class variables with getter and setter.**
- **Properties avails access to private variables in class.**
- **Properties needs camel casing or Pascal casing.**
- **Properties with Getter** → To read only.
- **Properties with Setter** → To write only.
- **Properties with getter and setter we can read and write.**

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

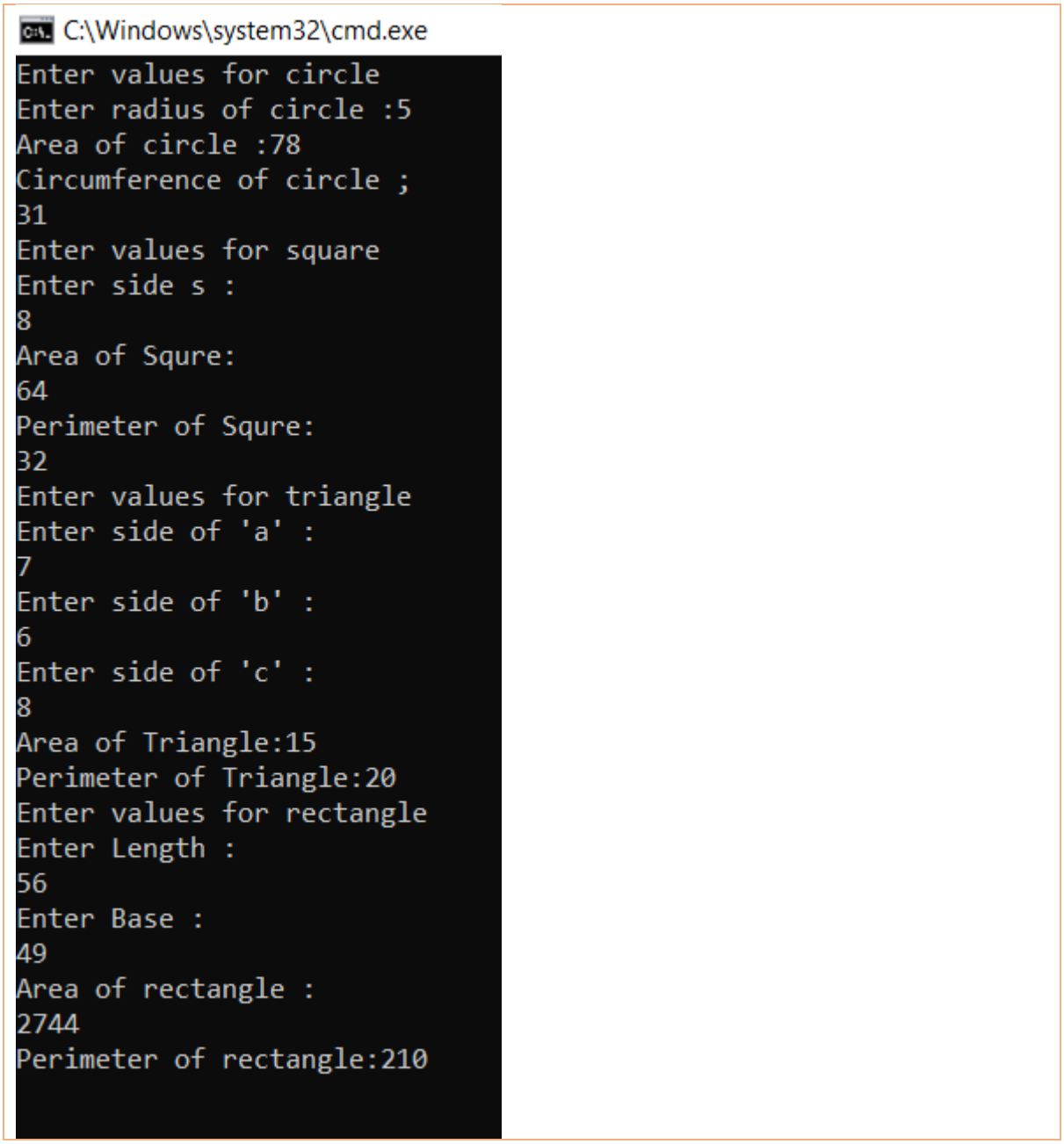
CODE

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

namespace Ishape
{
    /// <summary>
    /// iNTERFACE FOR SHAPES
    /// </summary>
    interface IShapePrg
    {
        int GET_Area();
        int Get_Perimeter();
    }
    /// <summary>
    /// RETURNS AREA AND CIRCUMFERENCE OF CIRCLE
    /// </summary>
    class ShapePrg_circle : IShapePrg
    {
        public int Radius;
        public void GetRadius()
        {
            Console.Write("Enter radius of circle :");
            Radius = Convert.ToInt32(Console.ReadLine());

        }

        public int GET_Area()
        {
            Console.Write("Area of circle :");
            return 22 * Radius * Radius/7;
        }

        public int Get_Perimeter()
        {
            Console.WriteLine("Circumference of circle ;");
            return 2*22*Radius/7;
        }
    }
    /// <summary>
    /// RETURNS AREA AND PERIMETER OF SQUARE
    /// </summary>
    class ShapePrg_square : IShapePrg
    {
        public int Sides;
        public void GetSides()
        {
            Console.WriteLine("Enter side s :");
            Sides = Convert.ToInt32(Console.ReadLine());

        }

        public int GET_Area()
        {
            Console.WriteLine("Area of Squre:");
            return Sides*Sides;
        }
```

```
        public int Get_Perimeter()
        {
            Console.WriteLine("Perimeter of Squre:");
            return 4*Sides;
        }
    }
    /// <summary>
    /// RETURNS AREA AND PERIMETR OF TRIANGLE
    /// </summary>
    class ShapePrg_Triangle : IShapePrg
    {
        public int A;
        public int B;
        public int C;

        public int side;
        public void Get_Sides()
        {
            Console.WriteLine("Enter side of 'a' :");
            A = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter side of 'b' :");
            B = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter side of 'c' :");
            C = Convert.ToInt32(Console.ReadLine());
            side = (A+B+C) / 2;

        }

        public int GET_Area()
        {
            Console.Write("Area of Triangle:");
            double val = side * (side-A) * (side-B) * (side-C);
            return (int)(double)Math.Sqrt(val);
        }

        public int Get_Perimeter()
        {
            Console.Write("Perimeter of Triangle:");
            return 2 * side;
        }
    }
    /// <summary>
    /// RETURNS AREA AND PERIMETER OF TRIANGLE
    /// </summary>
    class ShapePrg_Rectaangle : IShapePrg
    {
        public int Length;
        public int Base;
        public void Getside()
        {
            Console.WriteLine("Enter Length :");
            Length = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Base :");
            Base = Convert.ToInt32(Console.ReadLine());

        }

        public int GET_Area()
        {
            Console.WriteLine("Area of rectangle :");
            return Length*Base;
        }
```

```
        public int Get_Perimeter()
        {
            Console.Write("Perimeter of rectangle:");
            return 2 * (Length+Base);
        }
    }

    internal class Program
    {
        static void Main(string[] args)
        {

/*****************************************************************************
 * AUTHOR : PRUDHVI.                                                         *
 * PURPOSE : Write example program for interfaces discussed in the class     *
 * IShape include the classes Cricle, Square, Triangle, Rectangle.           *
*****************************************************************************
***/
            var circle = new ShapePrg_circle();
            var squre = new ShapePrg_square();
            var triangle = new ShapePrg_Triangle();
            var rectangle = new ShapePrg_Rectaangle();
            circle.GetRadius();
            Console.WriteLine(circle.GET_Area());
            Console.WriteLine(circle.Get_Perimeter());
            squre.GetSides();
            Console.WriteLine(squre.GET_Area());
            Console.WriteLine(squre.Get_Perimeter());
            triangle.Get_Sides();
            Console.WriteLine(triangle.GET_Area());
            Console.WriteLine(triangle.Get_Perimeter());
            rectangle.Getside();
            Console.WriteLine(rectangle.GET_Area());
            Console.WriteLine(rectangle.Get_Perimeter());
            Console.ReadLine();
        }
    }
}
```

OUTPUT



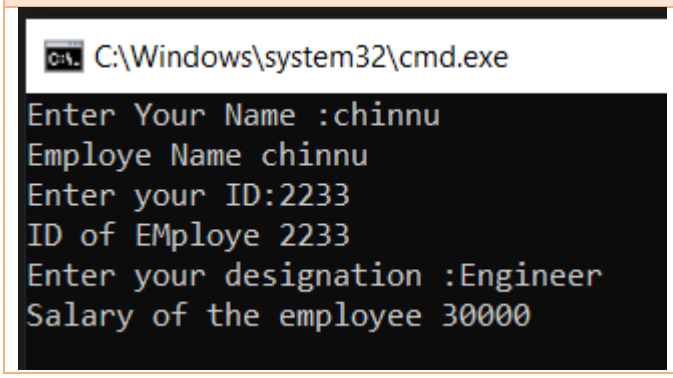
Q. Write sample code to illustrate properties as discussed in class.

CODE

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

namespace D11Properties_Illustration
{
```

```
    internal class Program
    {
        static void Main(string[] args)
        {
/*****************************************************************************
 * AUTHOR : PRUDHVI.                                                         *
 * PURPOSE : Write sample code to illustrate properties as discussed in class .                 *
*****************************************************************************
***/

            var emp = new Employee();
            Console.Write("Enter Your Name :");
            emp.Name = Console.ReadLine();
            Console.WriteLine($"Employe Name {emp.Name}");
            Console.Write("Enter your ID:");
            emp.Id = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine($"ID of EMploye {emp.Id}");
            Console.Write("Enter your designation :");
            emp.Designation = Console.ReadLine();
            Console.WriteLine($"Salary of the employee {emp.Salary}");
            Console.ReadLine();
        }
    }
    class Employee
    {
        private int salary;
        private string designation;

        public int Id { get; set; }
        public string Name { get; set; }

        public string Designation { set { designation = value; } }

        public int Salary
        {
            get
            {
                salary = (designation == "Engineer") ? 30000 : 50000;
                return salary;
            }
        }


    }
}
```

OUTPUT

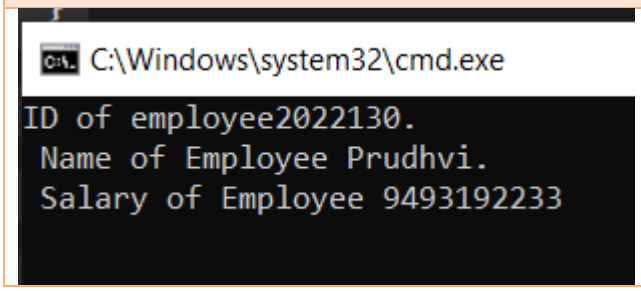


Q. Create a class Employee with only properties.

CODE

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

namespace Employee_Prop
{
    public class Employee
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public long Salary { get; set; }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
/*****************************************************************************
 * AUTHOR : PRUDHVI.                                                         *
 * PURPOSE : Write sample code to illustrate properties as discussed in class .                *
*****************************************************************************
***/

            Employee employees = new Employee();
            employees.Id = 2022130;
            employees.Name = "Prudhvi";
            employees.Salary = 9493192233;
            Console.WriteLine($"ID of employee{employees.Id}.\n Name of
Employee {employees.Name}.\n Salary of Employee {employees.Salary}");
            Console.ReadLine();
        }
    }
}
```

OUTPUT

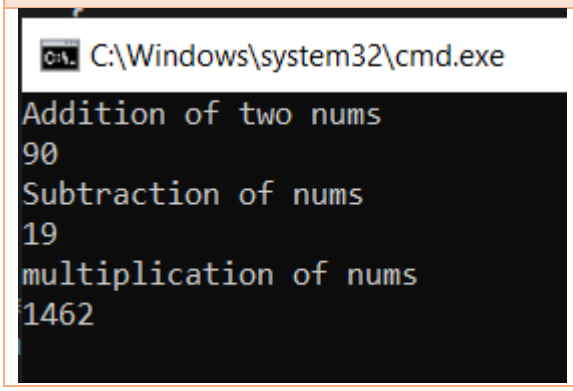


Q. Create Mathematics class and add 3 static methods and call the methods in main method.

CODE

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

namespace day11_properties
{
    public class Mathematics
    {

        public static int Add(int a, int b)
        {
            Console.WriteLine("Addition of two nums");
            return a + b;
        }
        public static int Sub(int c, int d)
        {
            Console.WriteLine("Subtraction of nums");
            return c - d;
        }
        public static int Mul(int e, int f)
        {
            Console.WriteLine("multiplication of nums");
            return e * f;
        }

    }
    internal class Program
    {
        static void Main(string[] args)
        {

            Console.WriteLine(Mathematics.Add(34,56));
            Console.WriteLine(Mathematics.Sub(75,56));
            Console.WriteLine(Mathematics.Mul(34,43));
            Console.ReadLine();
        }
    }

}
```

OUTPUT



-Thank you