DAY 11 ASSIGNMENT

-- BY SARATH KASIMSETTY

1) Write the difference between abstract class and interface in C#

ABSTRACT CLASS	INTERFACT CLASS
1). It consists of both abstract class and normal class variables.	1). It purely consists of Abstract class.
2). We cannot perform Multiple Inheritance using abstract class.	2). We can perform Multiple inheritance using Interface.
3). It acts like a Template.	3). It acts like a contract.
4). Abstract class can provide the implementation of Interface	4). Interface can't provide the implementation of Abstract class.
5). A class can only use one abstract class	5). A class can use multiple interfaces.
6). It contains different types of access modifiers like public, private, protected.	6). It only contains public access modifiers because everything in the interface is public

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

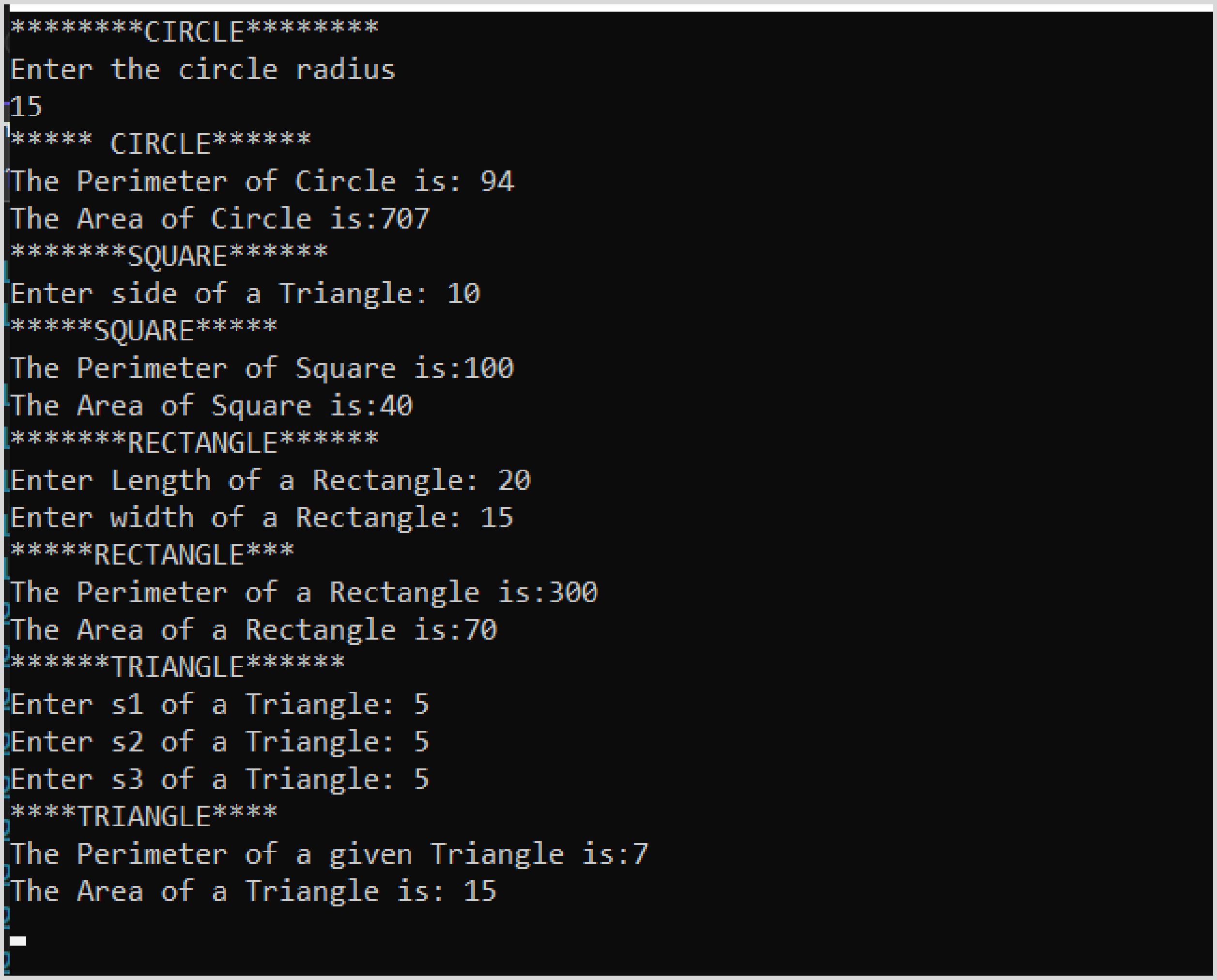
1. Interface is a pure Abstract class.
2. By default, the methods in interface are public and abstract.
3. Any class that is implementing interface must override all the methods.
4. Interface acts like a contractor. Its name should start with "I".
5. Interface supports Multiple Inheritance.
6. Interface can contain properties and methods, but not fields.
7. To access Interface methods, the interface must be implemented by another class

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;
//sarath kasimsetty
//Write example program for interfaces discussed in the class IShape
//include the classes Cricle, Square, Triangle, Rectangle
namespace Day11Project3
{
    /// <summary>
    /// Interface class id shape in Abstarct methods to must override in derived
all classes
    /// </summary>
    interface Ishape
    {
        int CalculatePerimeter();
        int CalculateArea();
    }

    class Circle:Ishape
    {
        private int radius;
        /// <summary>
        /// Read value from user
        /// </summary>
        public void ReadRadius()
        {
            Console.WriteLine("********CIRCLE********");
            Console.WriteLine("Enter the circle 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;
        /// <summary>
        /// Read value from user
        /// </summary>
        public void ReadSides()
        {
            Console.WriteLine("*******SQUARE******");
            Console.Write("Enter side of a Square: ");
            side = Convert.ToInt32(Console.ReadLine());

        }
        public int CalculateArea()
        {
            return 4 * side;
        }

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

    class Rectangle : Ishape
    {
        private int l;
        private int b;

        /// <summary>
        /// Read value from user
        /// </summary>
        public void ReadSide()
        {
            Console.WriteLine("*******RECTANGLE******");
            Console.Write("Enter Length of a Rectangle: ");
            l = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter width of a Rectangle: ");
            b = Convert.ToInt32(Console.ReadLine());
        }
        public int CalculateArea()
        {
            return 2*(l+b);
        }

        public int CalculatePerimeter()
        {
            return l*b;
        }
    }

    class Triangle : Ishape
    {
        private int s1;
        private int s2;
        private int s3;
        /// <summary>
        /// Read value from user
        /// </summary>
        public void ReadSides()
        {
            Console.WriteLine("******TRIANGLE******");
            Console.Write("Enter s1 of a Triangle: ");
            s1 = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter s2 of a Triangle: ");
            s2 = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter s3 of a Triangle: ");
            s3 = Convert.ToInt32(Console.ReadLine());
        }
        public int CalculateArea()
        {
            return s1 + s2 + s3;
        }

        public int CalculatePerimeter()
        {
            double semiperimeter = (s1 + s2 + s3) / 2;
            double Area = Math.Sqrt(semiperimeter * (semiperimeter - s1) *
(semiperimeter - s2)* (semiperimeter - s3));
            return Convert.ToInt32(Area);

        }
    }

    internal class Program
    {
        static void Main(string[] args)
        {
            Circle circle = new Circle();
            circle.ReadRadius();
            Console.WriteLine("***** CIRCLE******");
            Console.WriteLine($"The Perimeter of Circle is:
{circle.CalculatePerimeter()}");
            Console.WriteLine($"The Area of Circle is:{circle.CalculateArea()}");

            Square squ = new Square();
            squ.ReadSides();
            Console.WriteLine("*****SQUARE*****");
            Console.WriteLine($"The Perimeter of Square
is:{squ.CalculatePerimeter()}");
            Console.WriteLine($"The Area of Square is:{squ.CalculateArea()}");

            Rectangle rectangle = new Rectangle();
            rectangle.ReadSide();
            Console.WriteLine("*****RECTANGLE***");
            Console.WriteLine($"The Perimeter of a Rectangle is:{
rectangle.CalculatePerimeter()}");
            Console.WriteLine($"The Area of a Rectangle
is:{rectangle.CalculateArea()}");

            Triangle triangle = new Triangle();
            triangle.ReadSides();
            Console.WriteLine("****TRIANGLE****");
            Console.WriteLine($"The Perimeter of a given Triangle
is:{triangle.CalculatePerimeter()}");
            Console.WriteLine($"The Area of a Triangle is:
{triangle.CalculateArea()}");

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

OUTPUT:



```
********CIRCLE********
Enter the circle radius
15
***** CIRCLE******
The Perimeter of Circle is: 94
The Area of Circle is:707
*******SQUARE******
Enter side of a Triangle: 10
*****SQUARE*****
The Perimeter of Square is:100
The Area of Square is:40
*******RECTANGLE******
Enter Length of a Rectangle: 20
Enter width of a Rectangle: 15
*****RECTANGLE***
The Perimeter of a Rectangle is:300
The Area of a Rectangle is:70
******TRIANGLE******
Enter s1 of a Triangle: 5
Enter s2 of a Triangle: 5
Enter s3 of a Triangle: 5
****TRIANGLE****
The Perimeter of a given Triangle is:7
The Area of a Triangle is: 15
```

4) Write the 7 points discussed about properties.

1) Properties introduced to access values or which deals with private variables.
2) A property with GET is to Read the data. A property with SET is to Write the data.
3) Property name starts with uppercase.
4). Properties are almost same as variables with {get} and {set} access modifiers.
5). A simple example of property:

```
class Employee
    {
        private int Id;
        private string Name;
        private string designation;
        public int Id
        {
            get { return Id; }
            set { id = value; }
        }
    }
```

5) Write sample code to illustrate properties as discussed in class.

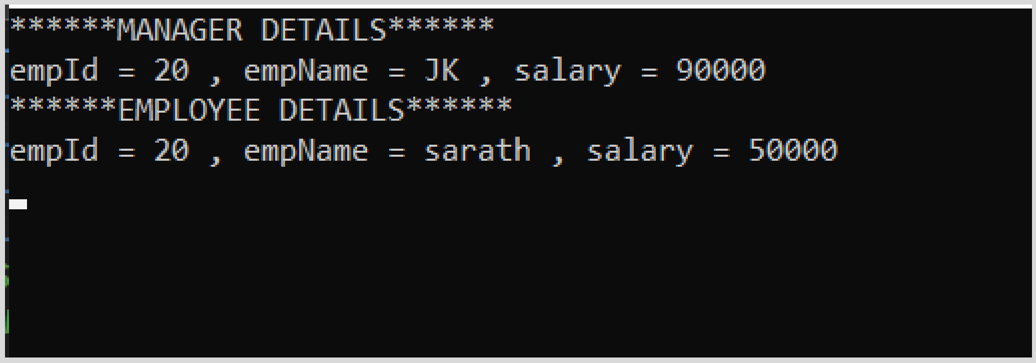
Id, name ,designation, salary
designation-set (writeonly) ,salary-get (get with some functionality)

CODE:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
//SARATH KASIMSETTY
//Write sample code to illustrate properties as discussed in class.
//id,name,designation,salary
//designation-set (writeonly)
//salary - get(get with some functionality)

namespace Day11Project5
```

```
{
    /// <summary>
    /// properties class
    /// </summary>
    class Employee
    {
        private int id;
        private string name;
        private string designation;


        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 {
                if (designation == "M")
                    return 90000;
                else if (designation == "EMP")
                    return 50000;
                else
                 return 15000;
            }
        }
    }

    internal class Program
    {
        static void Main(string[] args)
        {
            Employee emp1 = new Employee();

            Console.WriteLine("******MANAGER DETAILS******");
            emp1.Id = 20;
            emp1.Name = "JK";
            emp1.Designation = "M";
            Console.WriteLine("empId = {0} , empName = {1} , salary = {2}",emp1.Id,
emp1.Name, emp1.Salary);

            Employee emp2 = new Employee();

            Console.WriteLine("******EMPLOYEE DETAILS******");
            emp2.Id = 20;
            emp2.Name = "sarath";
            emp2.Designation = "EMP";
            Console.WriteLine("empId = {0} , empName = {1} , salary = {2}", emp2.Id,
emp2.Name, emp2.Salary);

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

OUTPUT:



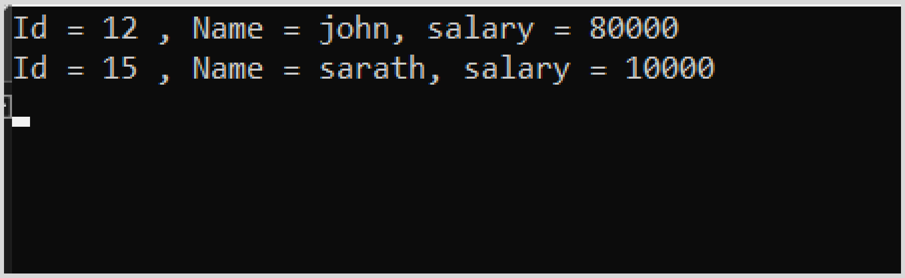
```
******MANAGER DETAILS******
empId = 20 , empName = JK , salary = 90000
******EMPLOYEE DETAILS******
empId = 20 , empName = sarath , salary = 50000
```

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;
//sarath kasimsetty
//Create a class Employee with only properties.

namespace Day11Project6
{
    /// <summary>
    /// properties of private variables
    /// </summary>
    internal class Employee
    {
        private int id;
        private string name;
        private int salary;


        public int Id
        {
            get { return id; } //read the value
            set { id = value; }//print  the value
        }
        public string Name
        {
            get { return name; }
            set { name = value; }
        }

        public int Salary
        {
            get { return salary;}//read the value
            set { salary = value; }//print the  value
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Employee emp1 = new Employee();

            emp1.Id = 12;
            emp1.Name = "john";
            emp1.Salary = 80000;

            Employee emp2 = new Employee();
            emp2.Id = 15;
            emp2.Name = "sarath";
            emp2.Salary = 10000;


            Console.WriteLine("Id = {0} , Name = {1}, salary = 
{2}",emp1.Id,emp1.Name,emp1.Salary);
            Console.WriteLine("Id = {0} , Name = {1}, salary = {2}", emp2.Id, emp2.Name, 
emp2.Salary);

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

OUTPUT:



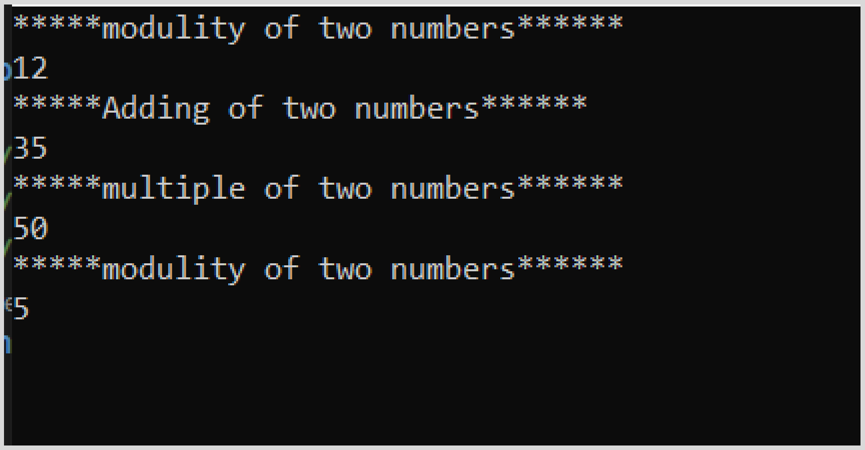
```
Id = 12 , Name = john, salary = 80000
Id = 15 , Name = sarath, salary = 10000
```

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;
//sarath kasimsetty
//Create Mathematics class and add 3 static methods and call the methods in
main method.

namespace Day11Project7
{
    /// <summary>
    /// Static methods
    /// </summary>
    internal class Mathematics
    {
        public static int Add(int a , int b)
        { return a + b; }

        public static int Mul(int a,int b)
        { return a * b; }

        public static int Div(int a, int b)
        { return a / b; }

        public static int Mod(int a ,int b)
        { return a % b; }

    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("*****modulity of two numbers******");
            Console.WriteLine(Mathematics.Mod(12, 24));

            Console.WriteLine("*****Adding of two numbers******");
            Console.WriteLine(Mathematics.Add(10, 25));

            Console.WriteLine("*****multiple of two numbers******");
            Console.WriteLine(Mathematics.Mul(10, 5));

            Console.WriteLine("*****modulity of two numbers******");
            Console.WriteLine(Mathematics.Mod(25, 10));

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

OUTPUT:



```
*****modulity of two numbers******
12
*****Adding of two numbers******
35
*****multiple of two numbers******
50
*****modulity of two numbers******
5
```

8) Research and understand when to create static methods.

- The main purpose of using static classes in C# is to provide blueprints of its inherited classes. Static classes are created using the static keyword in c#.
- We should use static methods whenever a function does not depend on a particular object of that class.
- When declaring constants.
- A static method doesn’t require a class