Day 11Morning Assignment
By Vinay Kudali
07-02-22

nations benefits

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

1. Interface is Pure Abstract Class
2. Interface name should start with I.
3. Interface acts like a contract.
4. In the interface, we must override all the methods.
5. Interface supports Multiple Inheritance.
6. The interface can have only Abstract methods

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

Code:

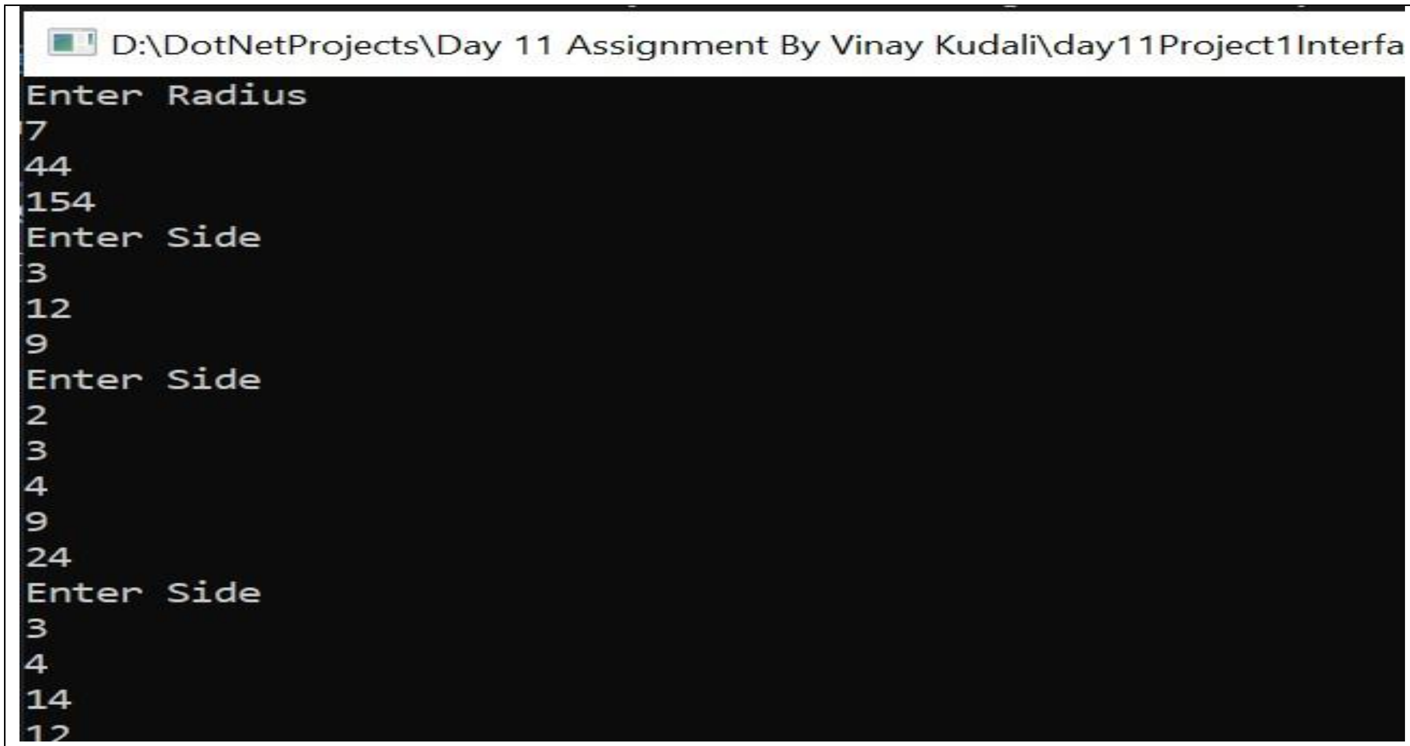
```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace day11Project1Interface
{
   //Author: Vinay Kudali
   //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");
         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");
        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");
        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");
            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 c = new Circle();
            c.ReadRadius();
            Console.WriteLine(c.CalculatePerimeter
            ());
            Console.WriteLine(c.CalculateArea());
            Square s = new Square();
            s.Readdata();
            Console.WriteLine(s.CalculatePerimeter
            ());
            Console.WriteLine(s.CalculateArea());
            Triangle t = new Triangle();
            t.ReadSide();
            Console.WriteLine(t.CalculatePerimete
            r());
            Console.WriteLine(t.CalculateArea());
            Rectangle r = new Rectangle();
            r.ReadSide();
            Console.WriteLine(r.CalculatePerimete
            r());
            Console.WriteLine(r.CalculateArea());

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

Output:



```
D:\DotNetProjects\Day 11 Assignment By Vinay Kudali\day11Project1Interfa
Enter Radius
7
44
154
Enter Side
3
12
9
Enter Side
2
3
4
9
24
Enter Side
3
4
14
12
```

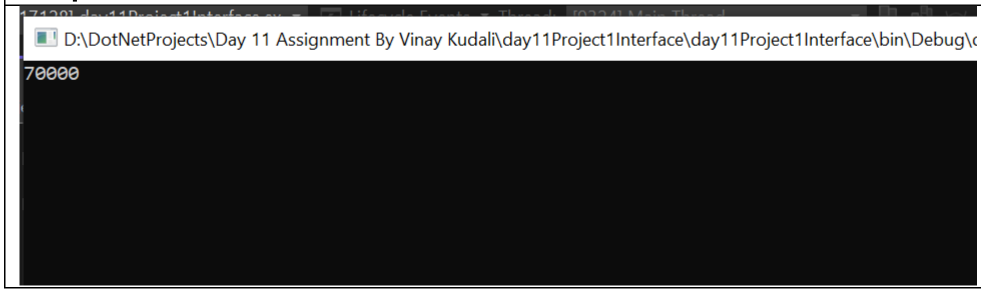
4. Write the 7 points discussed about properties.
Properties are almost same as class variables with get and set. Properties gives access to private variables in class. Properties are case sensitive. Get property –> To read Only Setter property → To write only. Properties with get and set we can do read and write.

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 Day11Project5
{
   //Author: Vinay Kudali
   //Purpose: id-get, set,name-get,set,designation-set(writeonly),salary-get(get with some
   functionality) 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 == "A") ? 40000 : 70000;
                return salary;

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

Output:



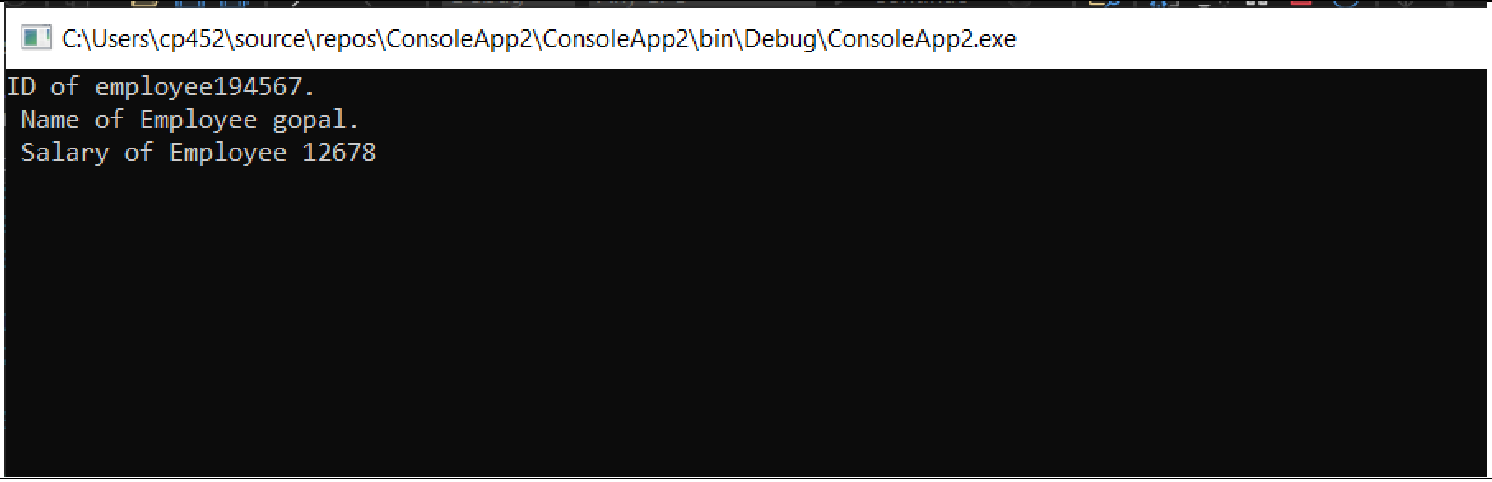
```
D:\DotNetProjects\Day 11 Assignment By Vinay Kudali\day11Project1Interface\day11Project1Interface\bin\Debug\
70000
```

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 Employee_Prop
{
    //Author: vinay kudali
    //purpouse: create class with only properties
    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)
        {
            Employee emp = new Employee();
            emp.Id = 194567;
            emp.Name = "gopal";
            emp.Salary = 12678;
            Console.WriteLine($"ID of employee{emp.Id}.\n Name of Employee {emp.Name}.\n Salary of Employee
{emp.Salary}");
            Console.ReadLine();
        }
    }
}
```

Output:



C:\Users\cp452\source\repos\ConsoleApp2\ConsoleApp2\bin\Debug\ConsoleApp2.exe

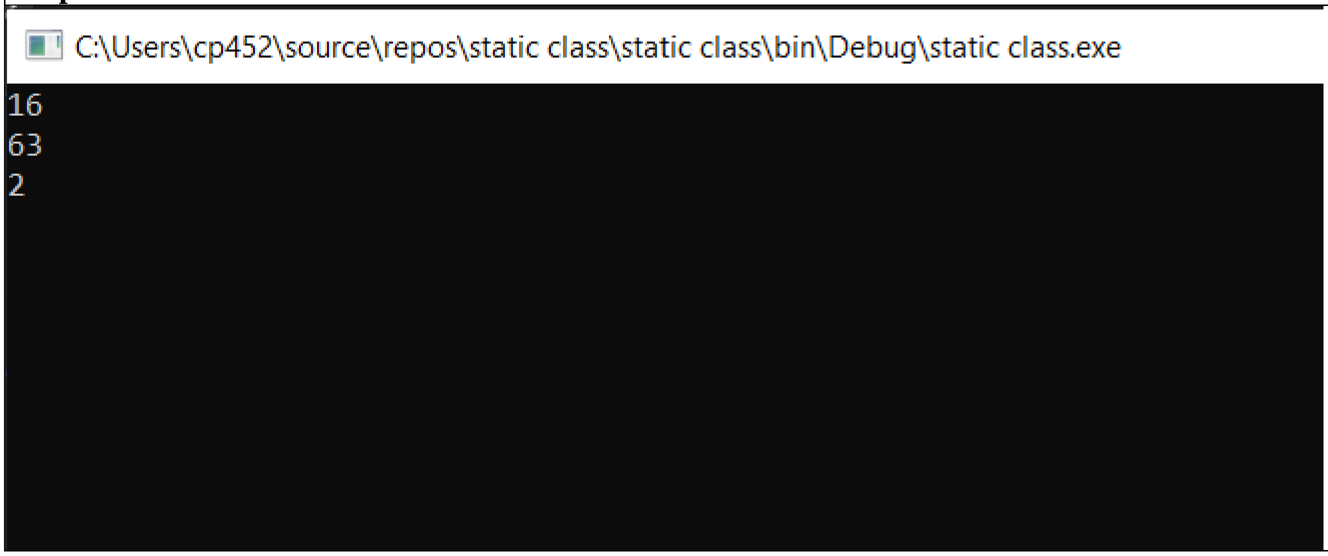
```
ID of employee194567.
 Name of Employee gopal.
 Salary of Employee 12678
```

7. Create mathematics class and afdd 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 _11_th_day_project3
{
    class mathematics
    {
        //Author : Vinay Kudali
        //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();
        }
    }
}
```

Output:



```
C:\Users\cp452\source\repos\static class\static class\bin\Debug\static class.exe
16
63
2
```

8. Research and understand when to create static methods.

1. If a Method Is dealing with Normal variables of a class, we can’t make it static.
2. If a method is not dealing with any variables it is dealing with static variables.

1. Research and write the difference between abstract class and interface in C# ?	
Abstrac t	**Interfac e**
Multiple Inheritance is not achieved by abstract class.	Multiple inheritance is achieved by interface.
It contains constructor.	It does not contain Constructor
It contains both declaration and definition part.	It contains Only Declaration part
It can contain static members.	It does not contain static members.
Abstract class is used for code re-usability.	Interface is used to achieve Multiple Inheritance.
Abstracts Acts like a Template.	Interface Acts like a Contractor.