DAY11 ASSIGNMENT
BY
ANDE MANOHAR
7th FEB 2022

Q1. Research and find difference between Abstract class and interface in c#

Abstract class	Interface
1.Abstract class contain constructor. 2.It can contain different types of access modifiers like public, private, etc 3.A Class can only use one abstract class.	1.Interface does no contain constructor 2.It only contains public access modifier because everything in the public. 3.A class can use multiple interface.

Q2.Write 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 tha is implementing Interface must override all methods.

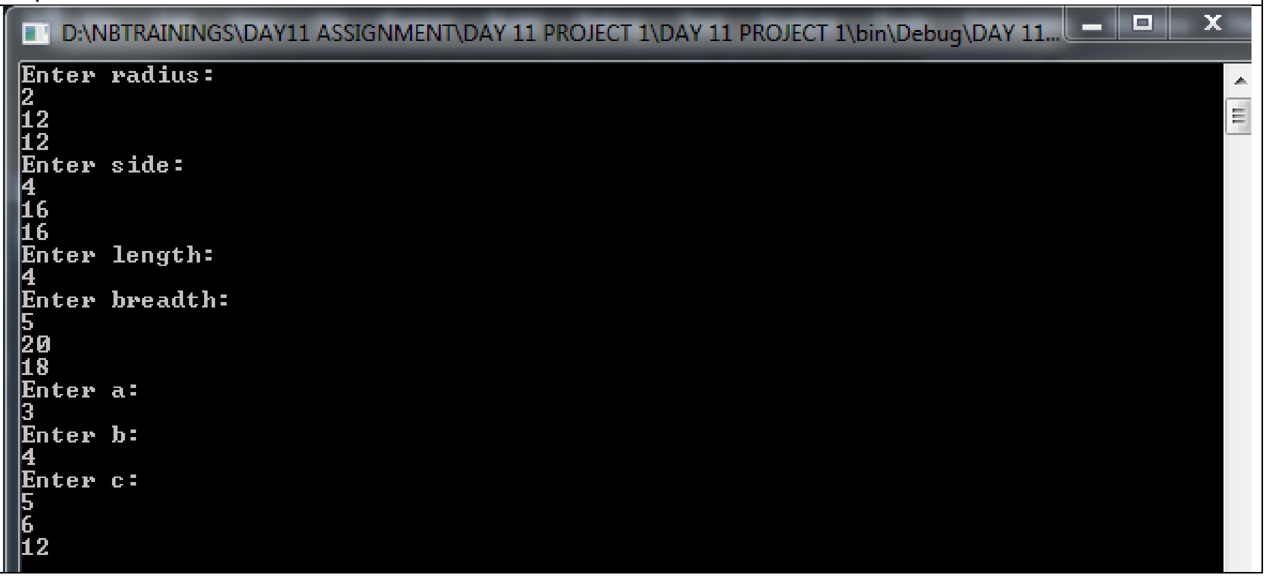
Q3.Write example program for interface discussed in the class
Ishape include the classes
Circle, square, Triangle, Rectangle

Code:

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

namespace DAY_11_PROJECT_1
{
    interface Ishape
    {
        int Area();
        int Perimeter();
    }
    class Circle :Ishape
    {
        public int radius;

        public int Area()
        {
            return 22 * radius * radius / 7;
        }
    public int Perimeter()
    {
        return 2 * 22 * radius / 7;
    }

    public void ReadRadius()
    {
        Console.WriteLine("Enter radius:");
        radius = Convert.ToInt32(Console.ReadLine());
    }

}
class Square : Ishape
{
    public int Side;

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

    public int Perimeter()
    {
        return 4 * Side;
    }

    public void ReadSide()
    {
        Console.WriteLine("Enter side:");
        Side = Convert.ToInt32(Console.ReadLine());
    }
}
class Rectangle : Ishape
{
    public int l;
    public int b;
    public  void ReadData()
    {
        Console.WriteLine("Enter length:");
        l = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("Enter breadth:");
        b = Convert.ToInt32(Console.ReadLine());
    }

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

    public int Perimeter()
```

```
        {
            return 2 * (l + b);
        }
    }
    class Triangle : Ishape
    {
        public int s, a, b, c;
        public void ReadSide()
        {
            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 Area()
        {
            return (int)Math.Sqrt(s * (s - a) * (s - b) * (s - c));
        }

        public int Perimeter()
        {
            return 2 * s;
        }
        internal class Program
        {
            static void Main(string[] args)
            {
                Circle c = new Circle();
                c.ReadRadius();
                Console.WriteLine(c.Area());
                Console.WriteLine(c.Perimeter());

                Square s = new Square();
                s.ReadSide();
                Console.WriteLine(s.Area());
                Console.WriteLine(s.Perimeter());

                Rectangle r = new Rectangle();
                r.ReadData();
                Console.WriteLine(r.Area());
                Console.WriteLine(r.Perimeter());

                Triangle t = new Triangle();
                t.ReadSide();
                Console.WriteLine(t.Area());
                Console.WriteLine(t.Perimeter());
                Console.ReadLine();
            }
        }

    }

}
```

Ouput:



D:\NBTRAININGS\DAY11 ASSIGNMENT\DAY 11 PROJECT 1\DAY 11 PROJECT 1\bin\Debug\DAY 11...

```
Enter radius:
2
12
12
Enter side:
4
16
16
Enter length:
4
Enter breadth:
5
20
18
Enter a:
3
Enter b:
4
Enter c:
5
6
12
```

Q4.Write 7 points discussed about properties

1. Properties are same as class variables but difference is 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.
5. Properties are introduced to deal with private variables.
6.Example code:

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

Public int id
{
Get
{
Reurn id;
}
Set
{
id = value;
}
}
}
```

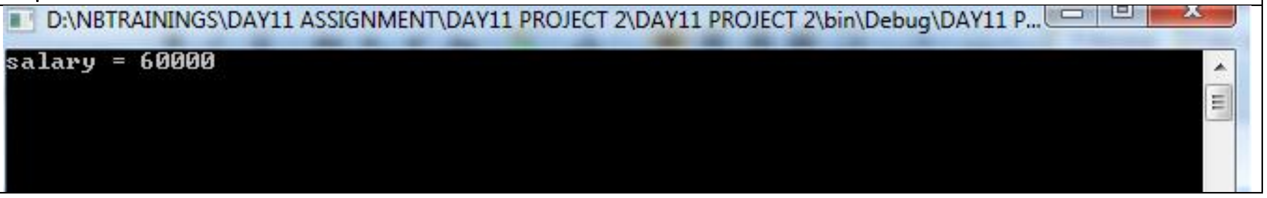
7.Property names start with uppercase.

Q5. Write sample code to illustrate properties as discussed in class.ID Name,desgination,salary

Code:

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

namespace DAY11_PROJECT_2
{
    class Employees
    {
        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;
            }
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Employees e = new Employees();
            e.Designation = "M";
            Console.WriteLine($"salary = {e.Salary}");
            Console.ReadLine();
        }
    }
}
```

Ouput:



D:\NBTRAININGS\DAY11 ASSIGNMENT\DAY11 PROJECT 2\DAY11 PROJECT 2\bin\Debug\DAY11 P...

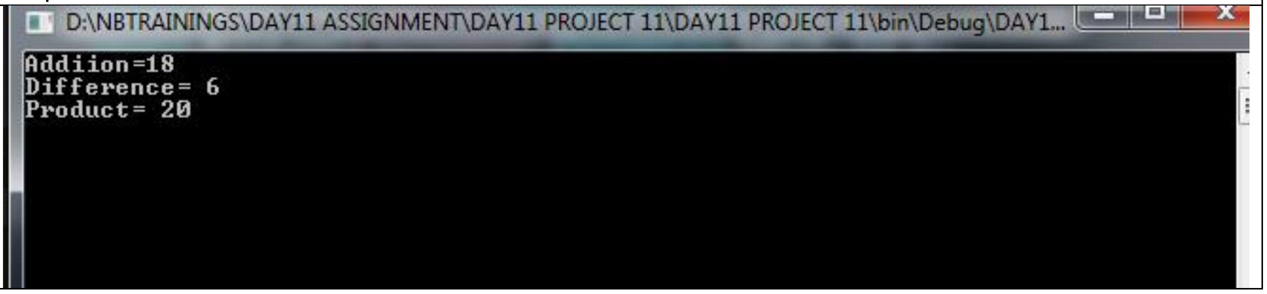
```
salary = 60000
```

Q7.Create Mathematics class and add 3 static methods and call the methods in main method

Code:

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

namespace DAY11_PROJECT_11
{
    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 Product(int a, int b)
        {
            return a * b;
        }

    }

    internal class Program
    {
        static void Main(string[] args)
        {
            Mathematics math = new Mathematics();
            Console.WriteLine($"Addiion={Mathematics.Add(12,6)}");
            Console.WriteLine($"Difference= {Mathematics.Sub(10, 4)}");
            Console.WriteLine($"Product= {Mathematics.Product(5, 4)}");
            Console.ReadLine();


        }
    }
}
```

Output:



D:\NBTRAININGS\DAY11 ASSIGNMENT\DAY11 PROJECT 11\DAY11 PROJECT 11\bin\Debug\DAY1...

```
Addiion=18
Difference= 6
Product= 20
```

Q8.Reasearch and understand when to create static methods.
1.Static method is used whenever we have function that does npt depend on a particular object of that class. 2.There is no harm in adding the static keyword: it will not break any of the code that referred to it.