Day11 Assignment

By

Narala Praveen

06-JAN-2022

Question1:

Research and write the difference between abstract class and interface in c#?

Abstract class	Interface
1. Abstract class have abstract and non-abstract methods.	Interface can have methods.
2. Abstract class doesn't support multiple inheritance.	Interface supports multiple inheritance.
3.Abstract class can have final, non final, static and non-static variables.	Interface has onlystatic and cfinal variables.
4.Abstract class can provide the implementation of interface.	Interface can't provide the implementation of abstract class.
5. The abstract keyword is used to declare abstract class.	The interface keyword is used to declare interface.
6.Abstract class is a template.	Interface is a contract.
7.Abstract class doesn,t have void in syntax.	Interface can have it.
8.Abstract class have Abstract in syntax.	Interface starts with uppercase letter.

Question2:

Write six points about interface discussed in the class ?

a. Interface is pure abstract class.
b. Interface name should start with I
c. Interface acts like a contract.
d. By default the methods in interface are public and abstract.
e. Interface supports Multiple inheritance.
f. Any class that is implementing interface must override all methods.

Question3:

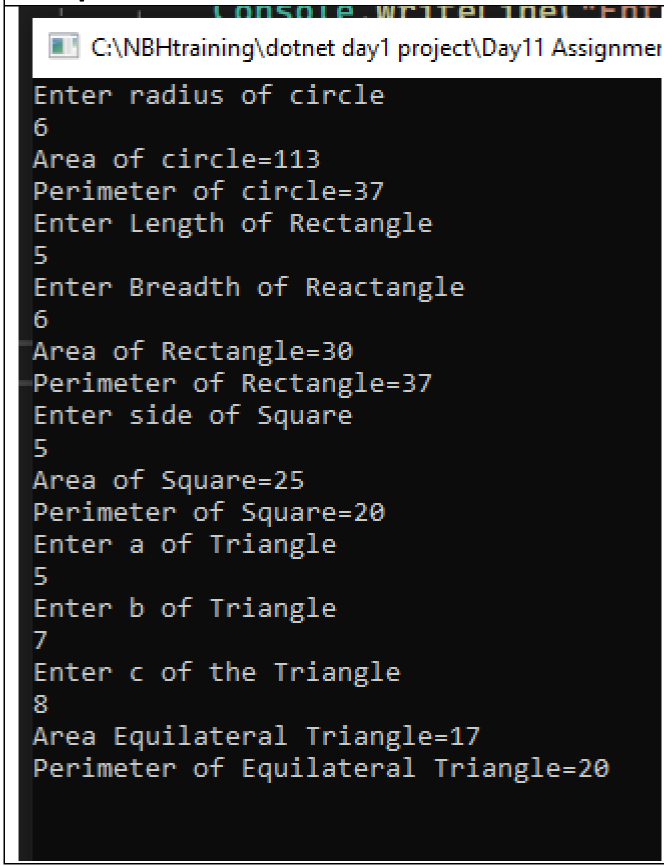
Write example program for interface discussed in the class IShape Include the classes

Circle,Square,Triangle,Rectangle

Code:

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

namespace Day11project1
{    //****************************************************************\\
    //Author:Narala Praveen
    //Purpose:To create a program for Interface
    //****************************************************************\\

    interface IShape
    {
        /// <summary>
        /// This method is for Area
        /// </summary>
        /// <returns></returns>
        int Area();

        /// <summary>
        /// This method is for Perimeter
        /// </summary>
        /// <returns></returns>
        int Perimeter();

    }
    class Circle : IShape
    {
        public int radius;
        public void Readradius()
        {
            Console.WriteLine("Enter radius of circle");
            radius=Convert.ToInt32(Console.ReadLine());
        }
        public int Area()
        {
            return (22 * radius * radius )/ 7;
        }
        public int Perimeter()
        {
            return (2 * 22 * radius) / 7;
        }
    }
    class Reactangle : IShape
    {
        public int Length;
        public int Breadth;
        public void Readdata()
        {
            Console.WriteLine("Enter Length of Rectangle");
            Length=Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Breadth of Reactangle");
            Breadth=Convert.ToInt32(Console.ReadLine());
        }
        public int Area()
        {
            return Length * Breadth;
        }
        public int Perimeter()
        {
            return 2 * (Length + Breadth);
        }

    }
    class Square:IShape
    {
        public int side;
        public void Readdata()
        {
            Console.WriteLine("Enter side of Square");
            side=Convert.ToInt32(Console.ReadLine());
        }
        public int Area()
        {
            return side * side;
        }
        public int Perimeter()
        {
            return 4 * side;
        }
    }
    class Triangle:IShape
    {
        public int a;
        public int b;
        public int c;
        public int s;

        public void Readdata()
        {
            Console.WriteLine("Enter a of Triangle ");
            a = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter b of Triangle ");
            b=Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter c of the Triangle");
            c=Convert.ToInt32(Console.ReadLine());
            s=(a+b+c)/2;
        }
        public int Area()
        {
            return (int)(Math.Sqrt(s * (s - a) * (s - b) * (s - c)));//Standard
formula
        }
        public int Perimeter()
        {
            return a+b+c;
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Circle c = new Circle();
            c.Readradius();
            Console.WriteLine($"Area of circle={c.Area()}");
            Console.WriteLine($"Perimeter of circle={c.Perimeter()}");
            Reactangle r=new Reactangle();
            r.Readdata();
            Console.WriteLine($"Area of Rectangle={r.Area()}");
            Console.WriteLine($"Perimeter of Rectangle={c.Perimeter()}");

            Square s =new Square();
            s.Readdata();
            Console.WriteLine($"Area of Square={s.Area()}");
            Console.WriteLine($"Perimeter of Square={s.Perimeter()}");

            Triangle t = new Triangle();
            t.Readdata();
            Console.WriteLine($"Area Equilateral Triangle={t.Area()}");
            Console.WriteLine($"Perimeter of Equilateral
Triangle={t.Perimeter()}");
            Console.ReadLine();
        }
    }
}
```

Output:



```
C:\NBHtraining\dotnet day1 project\Day11 Assignmer
Enter radius of circle
6
Area of circle=113
Perimeter of circle=37
Enter Length of Rectangle
5
Enter Breadth of Reactangle
6
Area of Rectangle=30
Perimeter of Rectangle=37
Enter side of Square
5
Area of Square=25
Perimeter of Square=20
Enter a of Triangle
5
Enter b of Triangle
7
Enter c of the Triangle
8
Area Equilateral Triangle=17
Perimeter of Equilateral Triangle=20
```

Question4:
Write the 7 points discussed about properties?

Properties:

a. Properties are almost same as class variables with get; & set;.
b. A property with only get;-------is Readonly.
c. A property with only set;____ is Writeonly.
d. A property with both get; and set; is Readable and we can assign too.

History of properties in c#:

1. Properties are introduced are introduced to deal with Private variables.
2. Example of property:

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

  Public int id
   {
     get
       {
        return id;
       }
      Set
        {
         id=value;
        }
      }
   }
```

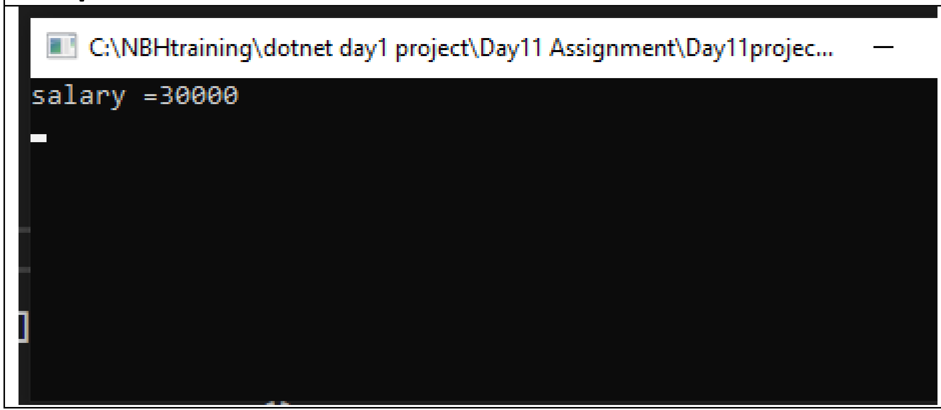
3.Properties names start with uppercase.

Question5:
Write sample code to illustrate properties as discussed in class.
Id
Name
Designation
Salary

Id-get,set
Name-det,set
Designation-set
Salary-get(some function)?

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

namespace Day11project2
{   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;
            }
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Employee employee = new Employee();
            employee.Designation = "s";
            Console.WriteLine($"salary ={employee.Salary} ");
            Console.ReadLine();
        }
    }
}
```

Output:



C:\NBHtraining\dotnet day1 project\Day11 Assignment\Day11projec...

```
salary =30000
```

Question6:
Create a class Employee With only properties?

Code:

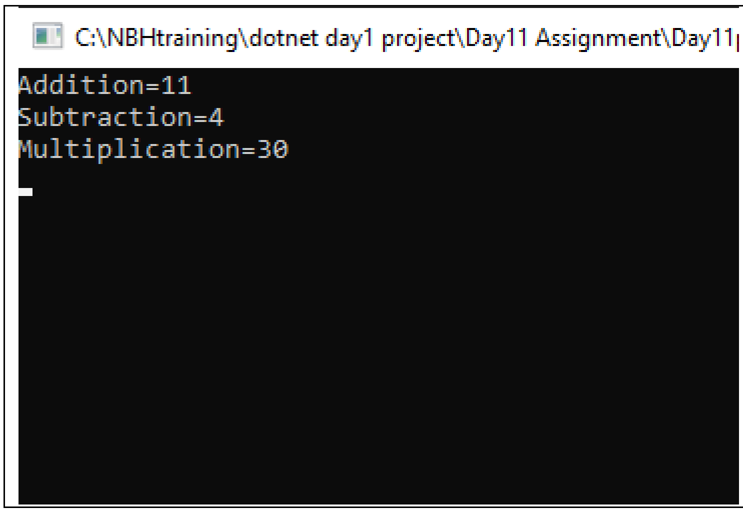
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day11project3
{    //Author:Narala praveen.
    //Purpose:Employee class with only properties.
    class Employee
    {
        public int Id
        {
            get
            {
                return Id;
            }
            set
            {
                Id = 101;
            }
        }
        public string Name
        {
            get { return Name; }
            set { Name = "Praveen"; }
        }
        public string Designation
        {
            set { Designation = value; }
        }



        public int Salary
        {
            get
            {
                Salary = (Designation == "S") ? 30000 : 500000;
                return Salary;
            }
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Employee emp = new Employee();
                emp.Designation = "S";
            Console.WriteLine($"Salary={emp.Salary}");

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

Output:

Question7:
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 Day11project4
{ //Author:Narala Praveen
    //To create Mathematics class with 3 static methods and call in main method.

    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 Multiplication(int a, int b)
        {
            return a*b;
        }

    }
    internal class Program
    {
        static void Main(string[] args)
        {
            MatheMatics math=new MatheMatics();
            Console.WriteLine($"Addition={MatheMatics.Add(5,6)}");
            Console.WriteLine($"Subtraction={MatheMatics.sub(8,4)}");

Console.WriteLine($"Multiplication={MatheMatics.Multiplication(5,6)}");
            Console.ReadLine();

        }
    }
}
```

Output:


C:\NBHtraining\dotnet day1 project\Day11 Assignment\Day11

```
Addition=11
Subtraction=4
Multiplication=30
```

Question8:

Research and write when to use static method?

a. **We should use static method whenever we have a function that does not depend on a particular object of that class.**
b. **When the methods are dealing with static variables.**
c. **If a method deals with class then it is not possible to use static method.**