Day11Assignment (7th Feb 2022)

By

Ram Charan

1.Research and Write difference between Abstract class and Interface in C#.	
Abstract class	Interface
• It can have both abstract and non-abstract methods.	• It can have only abstract methods.
• It doesnot support multiple inheritance.	• It supports multiple inheritance.
• It can provide the implementation of interface.	• It cannot provide the implementation of abstract class.
• Keyword used is “ abstract “	• Keyword used is “ interface “
• It can have access specifiers like public.	• By default, all are public and static.
• EXAMPLE : • public abstract class Shape • { • public abstract void Draw() • { • } • }	• EXAMPLE: • interface Ishape • { • void Draw() • { • } • }

2.Write about Interface 6 points discussed in the class.
• Interface is a pure abstract class.
• Interface supports multiple inheritance.
• The name should starts with "I".
• It acts like a contract.
• By default, methods in interface are public and abstract.
• Any class i.e., implementing interface should override all the methods.

3.Write a program for interfaces discussed in the class
Ishape
include the classes
Circle,Square,Triangle,Rectangle

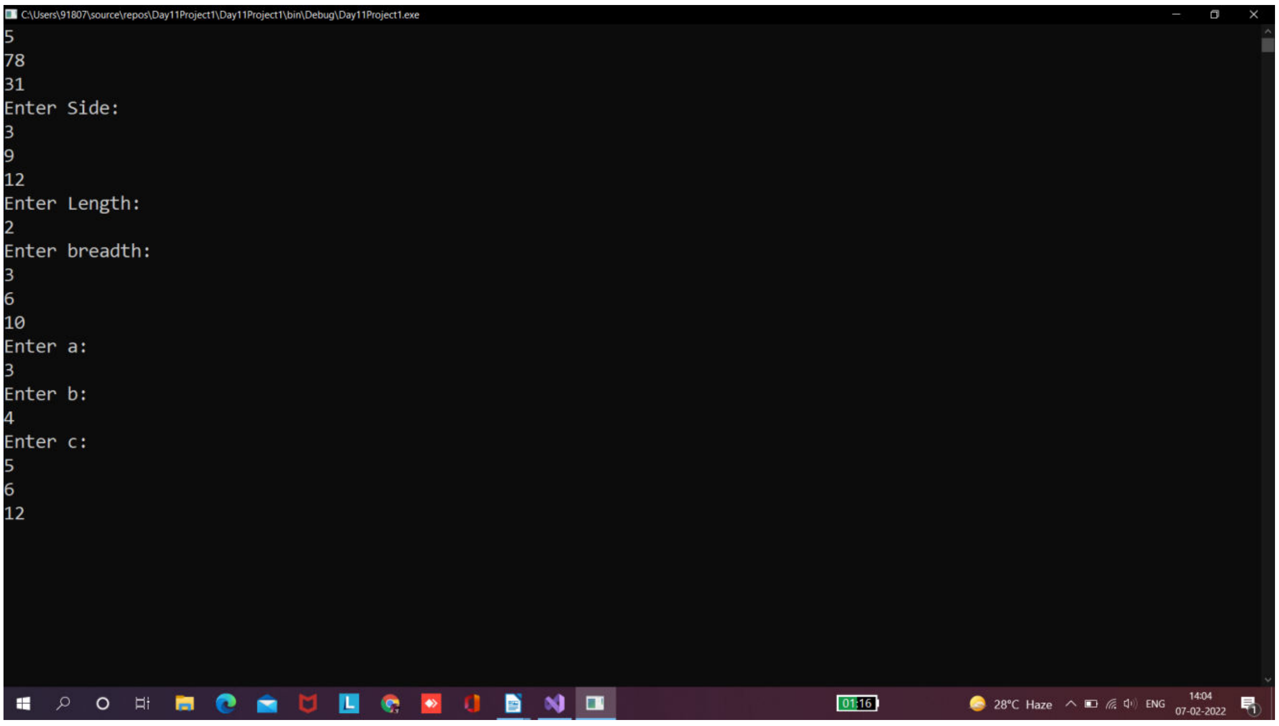
Code:

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

namespace Day11Project1
{
   //Author:Rc
   /*Purpose:Program for interface*/
   /// <summary>
   /// This is interface
   /// </summary>
   interface Ishape
   {
      /// <summary>
      /// This method finds area of given shape
      /// </summary>
      /// <returns>area</returns>
```

```
     int Area();
    /// <summary>
    /// This method find perimeter of given shape
    /// </summary>
    /// <returns>perimeter</returns>
     int Perimeter();
}
//Class declaration
class Circle:Ishape
{
    public int radius;
    /// <summary>
    /// This method is to read data
    /// </summary>
    public void ReadRadius()
    {
        Console.WriteLine("Enter Radius:");
        radius = Convert.ToInt32(Console.ReadLine());
    }
    public int Area()
    {
        return 22 * radius * radius / 7;
    }
    public int Perimeter()
    {
        return 2 * 22 * radius / 7;
    }
}
class Square : Ishape
{
    public int side;
    public void ReadSide()
    {
        Console.WriteLine("Enter Side:");
        side = Convert.ToInt32(Console.ReadLine());
    }
    public int Area()
    {
        return side*side;
    }
    public int Perimeter()
    {
        return 4*side;
    }

}
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();
        }
    }
}
```

Output:


C:\Users\91807\source\repos\Day11Project1\Day11Project1\bin\Debug\Day11Project1.exe
01:16
28°C Haze
ENG
14:04
07-02-2022

```
5
78
31
Enter Side:
3
9
12
Enter Length:
2
Enter breadth:
3
6
10
Enter a:
3
Enter b:
4
Enter c:
5
6
12
```

4. Write 7 points discussed about properties.
• Properties are same as class variables but difference is get; set; access modifiers.
• A property with only get; is ReadOnly.
• A property with only set; is Write only.
• Property acts as a mediator.
• Properties are introduced to deal with private variables.
• A very simple example is,

Class Employee
{
private int id;
private int age;
public int Id
{
get
{
return id;
}
set
{
id=value;
}
}
}

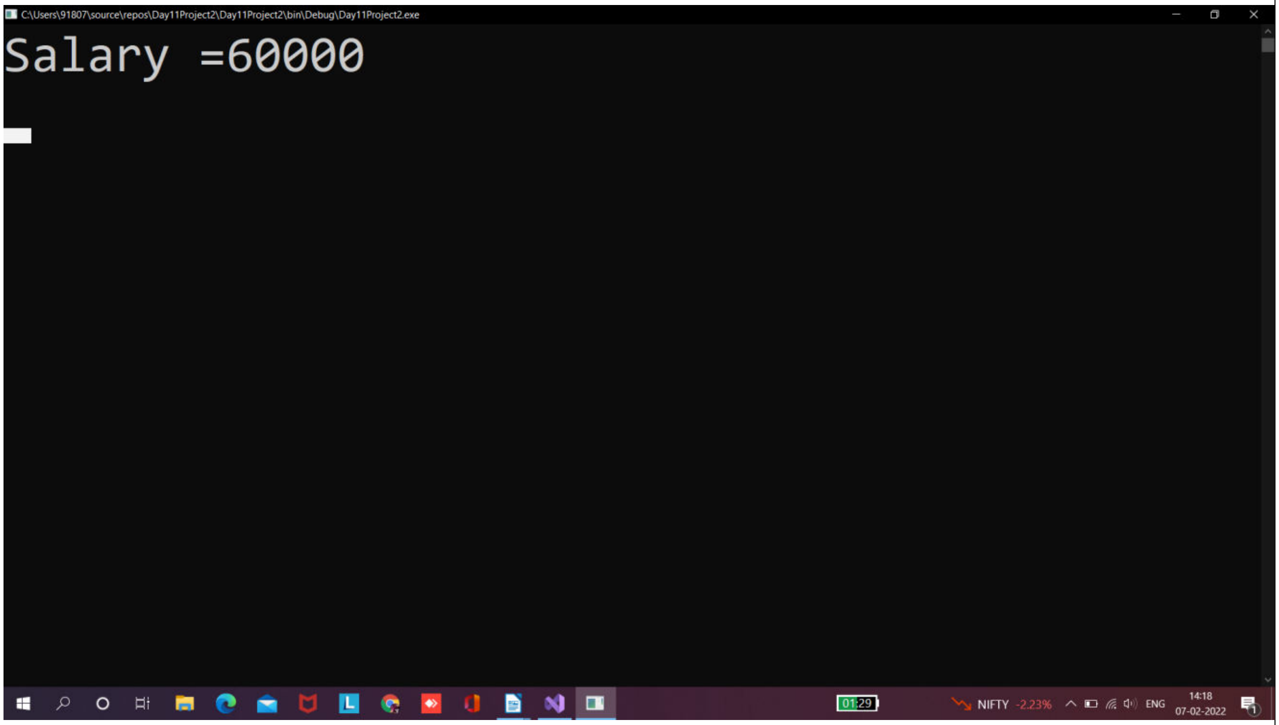
• Properties name should start with uppercase.

5.Write sample code to illustrate properties as discussed in class id -get,set name – get , set designation-set salary-get
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;
            }
            set
            {
                salary = value;
            }
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Employee e = new Employee();
            e.Designation = "M";
            Console.WriteLine($"Salary ={e.Salary}");

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

Output:


C:\Users\91807\source\repos\Day11Project2\Day11Project2\bin\Debug\Day11Project2.exe
01:29
NIFTY -2.23%
ENG
14:18
07-02-2022

```
Salary =60000
```

6.Create Employee class with only properties.

Code:

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

namespace Day11Project3
{
   class Employee
   {
      public int Id
      {

         get
         {
            return Id;
         }
         set
         {
            Id = value;
         }
      }
      public string Name
      {
         get
         {
            return Name;
         }
         set
         {
            Name = value;
         }
      }
      public string Designation
      {
         get
         {
            return Designation;
         }
         set
         {
            Designation = "Automation";
         }
      }
      public int Salary
```

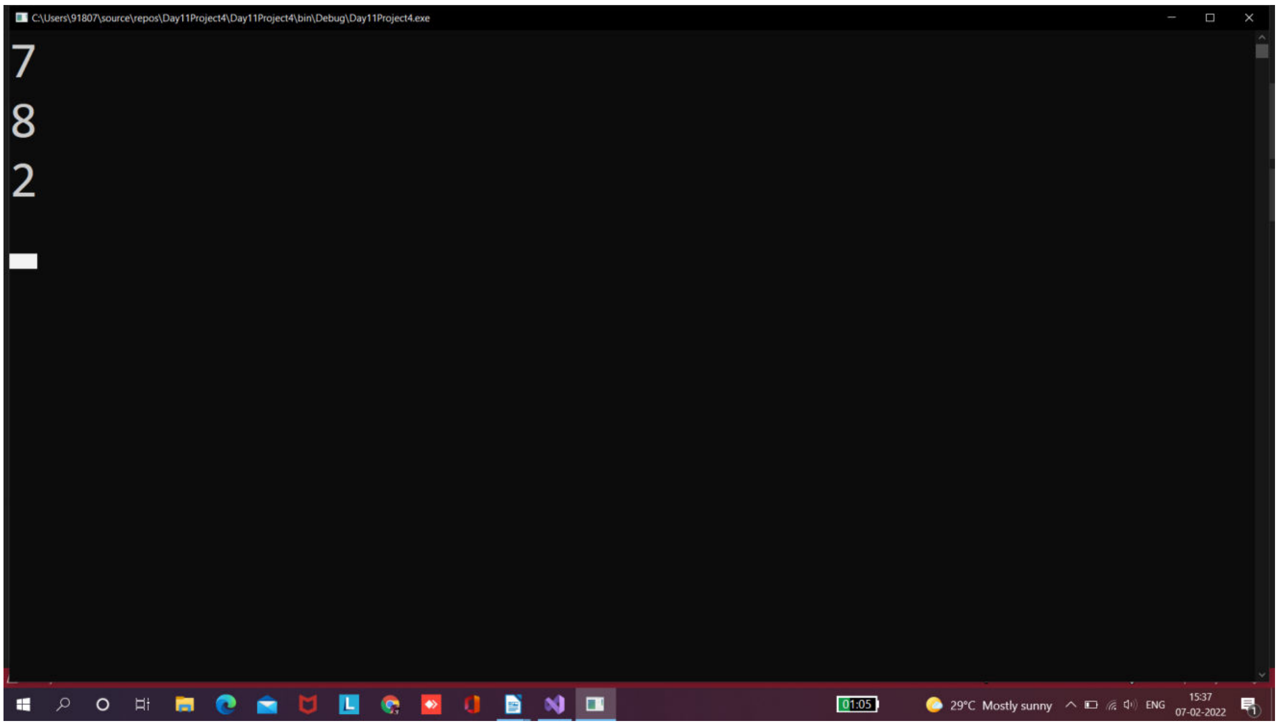
```
        {
            get
            {
                return Salary;
            }
            set
            {
                Salary = 50000;
            }
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {

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

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

Code:

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

namespace Day11Project4
{
    //Author:Rc
    /*Purpose:
     * for maths class create 3 static methods and call in main method*/
    class Maths
    {
        //public static int a = 4;
        //public static int b = 5;
        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) ;
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(Maths.Add(3,4));
            Console.WriteLine(Maths.Mul(4,2));
            Console.WriteLine(Maths.Div(6, 3));

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

Output:



```
C:\Users\91807\source\repos\Day11Project4\Day11Project4\bin\Debug\Day11Project4.exe
7
8
2
```

8.Research and understand when to create Static methods.
• If class is not having class variables then we can initialise static methods. • If a method is dealing with class variables then we cannot implement static methods. • If a method is dealing with static variables then we can use static methods.
• Ex: Math.Pow(), Math.Sqrt()

End of Day11Assignment