Day 11 Assignment
By
M Mary Margarette
On 07-02-2022

Difference Between Abstract class and Interface

Abstract Class	Interface
→ Class can only use one abstract class.	→ Class can use multiple interfaces.
→ Multiple inheritance is not achieved by abstract class.	→ Multiple inheritance can be achieved by Interface.
→ An Abstract member should define using the keyword abstract.	→ An Interface member cannot be defined using the keyword static, virtual, abstract.

Write 6 points about Interface

➢ Class name starts with I in Interface.
➢ Interface is pure Abstract class.
➢ Interface acts like a contract.
➢ Methods in Interface can be defined as Abstract method and Public Methods.
➢ Any class implementing interface must override all the methods.
➢ Interface supports Multiple-Inheritance.

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

Code:

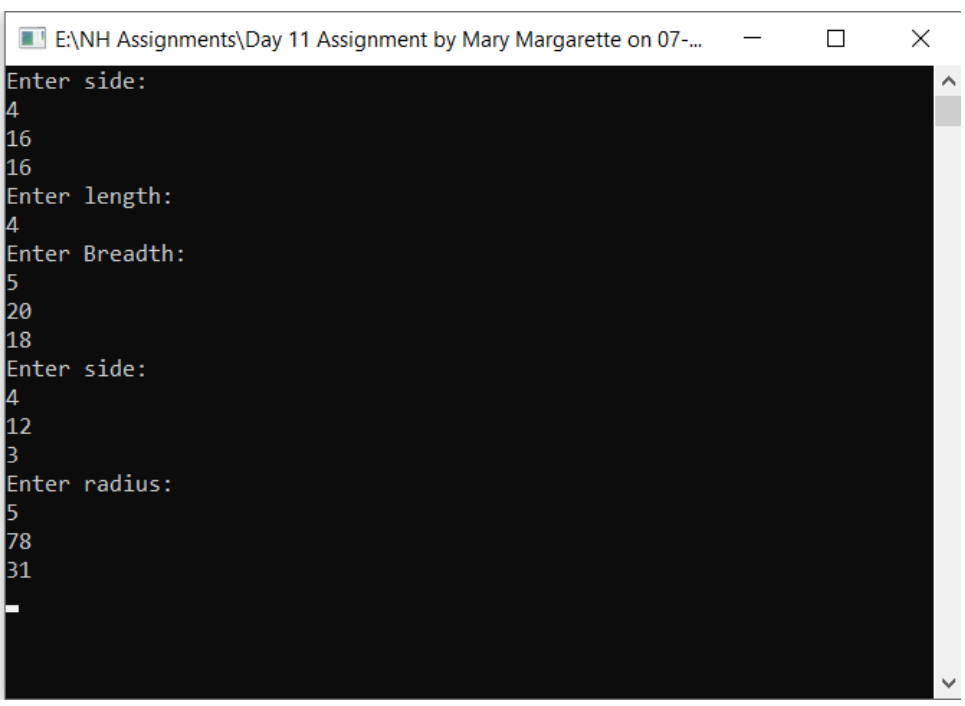
```
using System;
using System.Collections.Generic;
```

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

namespace Day_11_Project_1
{
   //Author: Mary Margaret
   //Program for interfaces class IShape including the classes Cricle, Square, Triangle, Rectangle.

   interface IShape
   {
      int CalculateArea();
      int CalculatePerimeter();

   }
   class Square : IShape
   {
      int s;
      public void Readside()
      {
         Console.WriteLine("Enter side:");
         s = Convert.ToInt32(Console.ReadLine());
      }
      public int CalculateArea()
      {
         return s * s;
      }

      public int CalculatePerimeter()
      {
         return 4 * s;
      }
   }
   class Rectangle : IShape
   {
      int l;
      int b;
      public void Readsides()
      {
         Console.WriteLine("Enter length:");
         l = Convert.ToInt32(Console.ReadLine());
         Console.WriteLine("Enter Breadth:");
         b = Convert.ToInt32(Console.ReadLine());
      }
      public int CalculateArea()
      {
         return l*b;
      }
```

```
    public int CalculatePerimeter()
    {
        return 2*(l+b);
    }
}
class Triangle : IShape
{
    int s;
    public void Readside()
    {
        Console.WriteLine("Enter side:");
        s = Convert.ToInt32(Console.ReadLine());
    }
    public int CalculateArea()
    {
        return 3 * s;
    }

    public int CalculatePerimeter()
    {
        return (3)^1/2*s*s/4;
    }
}
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;
    }
}

internal class Program
{
    static void Main(string[] args)
    {
        Square s = new Square();
```

```
        s.Readside();
        Console.WriteLine(s.CalculateArea());
        Console.WriteLine(s.CalculatePerimeter());

        Rectangle r = new Rectangle();
        r.Readsides();
        Console.WriteLine(r.CalculateArea());
        Console.WriteLine(r.CalculatePerimeter());

        Triangle t = new Triangle();
        t.Readside();
        Console.WriteLine(t.CalculateArea());
        Console.WriteLine(t.CalculatePerimeter());

        Circle c = new Circle();
        c.Readradius();
        Console.WriteLine(c.CalculateArea());
        Console.WriteLine(c.CalculatePerimeter());

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

Output:



Points about Properties

- Properties are the same as a class variable.
- A property with only get is – Read Only
- A property with only set is – Read Only
- Property with get and set >= read and assign values.
- Properties are meant to deal with Private Variables.
- Name of the property starts with Upper case.
- Simple program for property:

```
class Employee
{
private int id;
private string name;
private string designation;
private int salary;

public int Id
{
get {return id;}
set{id=value;}
}
}
```

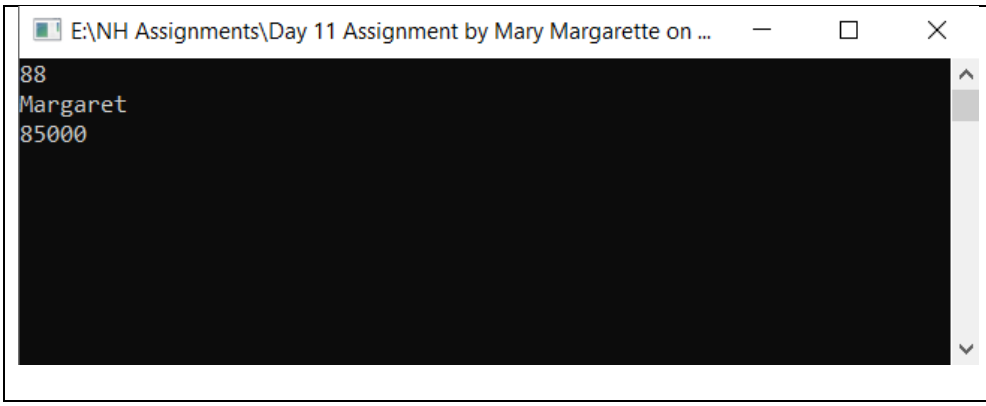
Write sample code to illustrate properties. id name designation salary

Code:

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

namespace Day_11_Project_2
{
   //Author: Mary Margaret
   //Write sample code to illustrate properties id name designation salary
   internal class Program
   {

      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 == "M") ? 85000 : 30000;
                return salary;
            }

        }
    }

    static void Main(string[] args)
    {
        Employee emp = new Employee();
        emp.Id = 88;
        Console.WriteLine(emp.Id);
        emp.Name = "Margaret";
        Console.WriteLine(emp.Name);
        emp.Designation = "M";
        Console.WriteLine(emp.Salary);

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

Output:



Create a class Employee with only properties

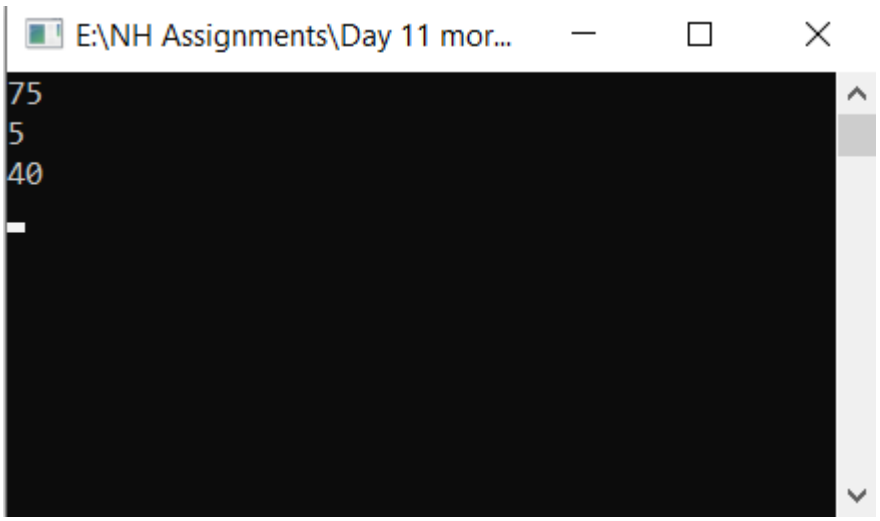
Code:

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

namespace Day_11_Project_4
{
    //Author: Mary Margaret
    //Create a class Employee with only properties
    internal class Program
    {
        class Employee
        {
            public int Id { get; set; }
            public string Name { get; set; }
            public string Designation { get; set; }
            public int Birth { get; set; }

        }
        static void Main(string[] args)
        {
        }
    }
}
```

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 Day_11_Project_3
{
   //Author: Mary Margaret
   //Create Mathematics class and add 3 static methods and call the   methods in main method.
   internal class Program
   {
      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 Mul(int a, int b)
         {
            return a * b;
         }
      }
      static void Main(string[] args)
      {
         Console.WriteLine(Mathematics.Add(55,20));
         Console.WriteLine(Mathematics.Sub(85,80));
         Console.WriteLine(Mathematics.Mul(8,5));
         Console.ReadLine();


      }
   }
}
```

Output:



Research and understand when to create static methods.
→ whenever you have a function that does not depend on a particular object of that class. → Creating a static class is therefore basically the same as creating a class that contains only static members and a private constructor.