DAY 11 ASSIGNMENT
BY
CH. PAVAN KUMAR REDDY (07-02-2022)

Q1). Write the difference between abstract class and interface in C#	
ABSTRACT CLASS	**INTERFACE CLASS**
1). It consists of both abstract class and normal class variables.	**1).** It purely consists of Abstract class.
2). We cannot perform Multiple Inheritance using abstract class.	**2)**. We can perform Multiple inheritance using Interface.
3). It acts like a Template.	**3)**. It acts like a contract.
4). Abstract class can provide the implementation of Interface	**4).** Interface can't provide the implementation of Abstract class.
5). A class can only use one abstract class	**5)**. A class can use multiple interfaces.
6). It contains different types of access modifiers like **public, private, protected.**	**6).** It only contains **public access** modifiers because everything in the interface is publi**c**

Q2). Write the 6 points about interface discussed in the class.
1). Interface is a pure Abstract class.
2). By default, the methods in interface are public and abstract.
3). Any class that is implementing interface must override all the methods.
4). Interface acts like a contractor. Its name should start with "**I**".
5). Interface supports Multiple Inheritance.
6). Interface can contain properties and methods, but not fields.
7). To access Interface methods, the interface must be implemented by another class.

Q4). Write the 7 points discussed about properties.
1) Properties introduced to access values or which deals with private variables.
2) A property with **GET** is to Read the data. A property with **SET** is to Write the data.
3) Property name starts with uppercase.
4). Properties are almost same as variables with **{get}** and **{set}** access modifiers.
5). A simple example of property: **class Employee**

```
{
private int Id;
private string Name;
private string designation;

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

Q8) Research and understand when to create static methods

- The main purpose of using static classes in C# is to provide blueprints of its inherited classes. Static classes are created using the static keyword in c#.
- We should use static methods whenever a function does not depend on a particular object of that class.
- When declaring constants.
- A static method doesn't require a class.

PROJECT:1

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

CODE:

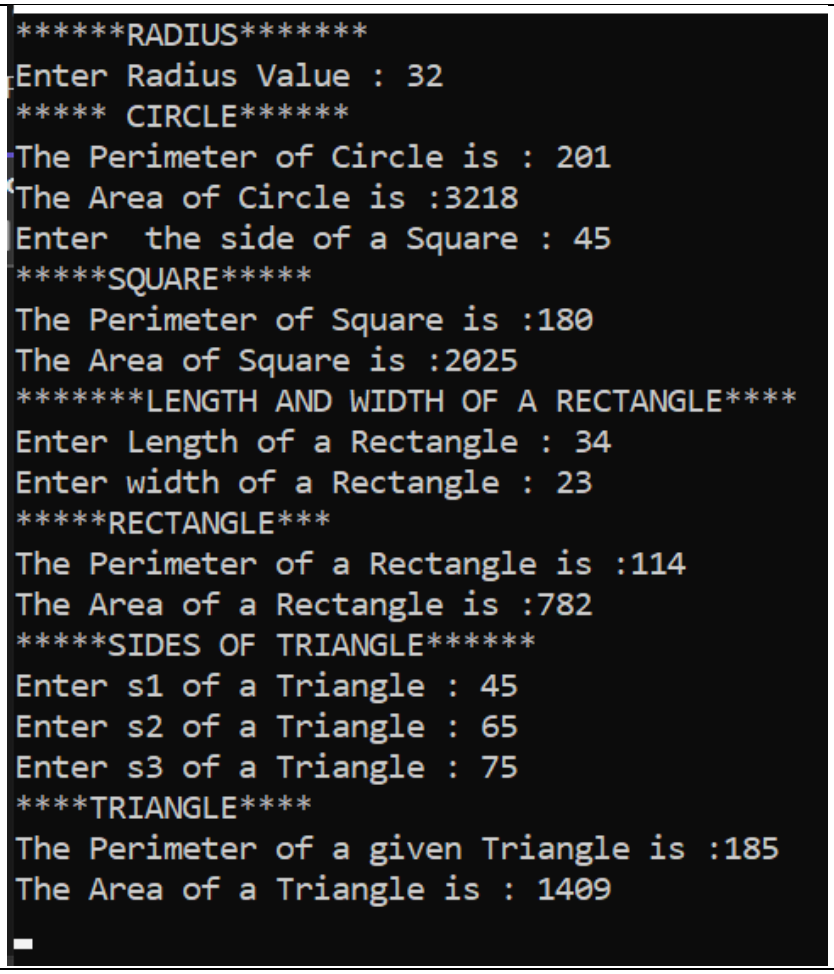
```
using System;
using System.Collections;

/// <summary>
/// DONE BY: PAVAN
/// PURPOSE: CALCULATING THE AREA AND PERIMETER OF CIRCLE, SQUARE, TRIANGLE, RECTANGLE:///
/// </summary>

namespace Day11Project1
{
   interface IShape
   {
      int CalculatePerimeter();
      int CalculateArea();
   }
```

```
class Circle: IShape
{
   private int radius;
   public void ReadRadius()
   {
      Console.WriteLine("******RADIUS*******");
      Console.Write("Enter Radius Value: ");
      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 ReadSide()
   {
      Console.Write("Enter the side of a Square: ");
      side = Convert.ToInt32(Console.ReadLine());
   }
   public int CalculatePerimeter()
   {
      return 4 * side;
   }
   public int CalculateArea()
   {
      return side * side;
   }
}
class Rectangle: IShape
{
   private int lgt;
   private int wdt;
```

```
    public void ReadSide()
    {
      Console.WriteLine("*******LENGTH AND WIDTH OF A RECTANGLE****");
      Console.Write("Enter Length of a Rectangle: ");
      lgt = Convert.ToInt32(Console.ReadLine());
      Console.Write("Enter width of a Rectangle: ");
      wdt = Convert.ToInt32(Console.ReadLine());
    }
    public int CalculatePerimeter()
    {
      return 2 * (lgt + wdt);
    }
    public int CalculateArea()
    {
      return lgt * wdt;
    }
  }
  class Triangle: IShape
  {
    private int s1;
    private int s2;
    private int s3;
    public void ReadSides()
    {
      Console.WriteLine("*****SIDES OF TRIANGLE******");
      Console.Write("Enter s1 of a Triangle: ");
      s1 = Convert.ToInt32(Console.ReadLine());
      Console.Write("Enter s2 of a Triangle: ");
      s2 = Convert.ToInt32(Console.ReadLine());
      Console.Write("Enter s3 of a Triangle: ");
      s3 = Convert.ToInt32(Console.ReadLine());
    }
    public int CalculatePerimeter()
    {
      return s1 + s2 + s3;
    }
    public int CalculateArea()
    {
      double semiperimeter = (s1 + s2 + s3) / 2;
      double Area = Math.Sqrt(semiperimeter * (semiperimeter - s1) * (semiperimeter - s2)
* (semiperimeter - s3));
      return Convert.ToInt32(Area);
```

```
    }
  }
  internal class Program
  {
    static void Main(string[] args)
    {
      Circle circle = new Circle();
      circle.ReadRadius();
      Console.WriteLine("***** CIRCLE******");
      Console.WriteLine($"The Perimeter of Circle is: {circle.CalculatePerimeter()}");
      Console.WriteLine($"The Area of Circle is:{circle.CalculateArea()} ");

      Square squ = new Square();
      squ.ReadSide();
      Console.WriteLine("*****SQUARE*****");
      Console.WriteLine($"The Perimeter of Square is:{square.CalculatePerimeter()}");
      Console.WriteLine($"The Area of Square is:{square.CalculateArea()}");

      Rectangle rect = new Rectangle();
      rect.ReadSide();
      Console.WriteLine("*****RECTANGLE***");
      Console.WriteLine($"The Perimeter of a Rectangle
is:{rectangle.CalculatePerimeter()}");
      Console.WriteLine($"The Area of a Rectangle is:{rectangle.CalculateArea()}");

      Triangle tri = new Triangle();
      tri.ReadSides();
      Console.WriteLine("****TRIANGLE****");
      Console.WriteLine($"The Perimeter of a given Triangle is:{tri.CalculatePerimeter()}");
      Console.WriteLine($"The Area of a Triangle is: {tri.CalculateArea()}");

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

OUTPUT:



PROJECT: 2

Write sample code to illustrate properties as discussed in class. id name designation salary id-get, set name-get, set designation-set (write only) salary-get (get with some functionality)

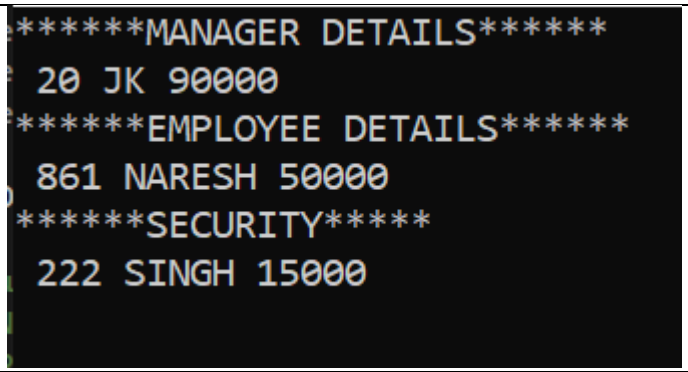
CODE:

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

namespace Day11Project2
{
   /// <summary>
   /// DONY BY: PAVAN
   /// PURPOSE: CREATING sample code to illustrate properties as discussed in class.
  // id
   //name
   //designation
```

```
// salary
 /// </summary>
 class Employee
 {
    private int id;
    private string name;
    private string designation;

    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
       {
          if (designation == "M")
             return 90000;
          else if (designation == "EMP")
             return 50000;
          else if (designation == "SECURITY")
             return 15000;
          else
             return 30000;
       }
    }
 }
 internal class Program
 {
    static void Main(string[] args)
    {
       Employee e = new Employee();
```

```
        Console.WriteLine("******MANAGER DETAILS******");
        e.Id = 20;
        e.Name = "JK";
        e.Designation = "M";
        Console.WriteLine($" {e.Id}, {e.Name},{e.Salary}");

        Employee emp1 = new Employee();
        Console.WriteLine("******EMPLOYEE DETAILS******");
        emp1.Id = 861;
        emp1.Name = "NARESH";
        emp1.Designation = "EMP";
        Console.WriteLine($" {emp1.Id}, {emp1.Name}, {emp1.Salary}");

        Employee emp2 = new Employee();
        Console.WriteLine("******SECURITY*****");
        emp2.Id = 222;
        emp2.Name = "SINGH";
        emp2.Designation = "SECURITY";
        Console.WriteLine($" {emp2.Id}, {emp2.Name}, {emp2.Salary}");


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

OUTPUT:



PROJECT: 3

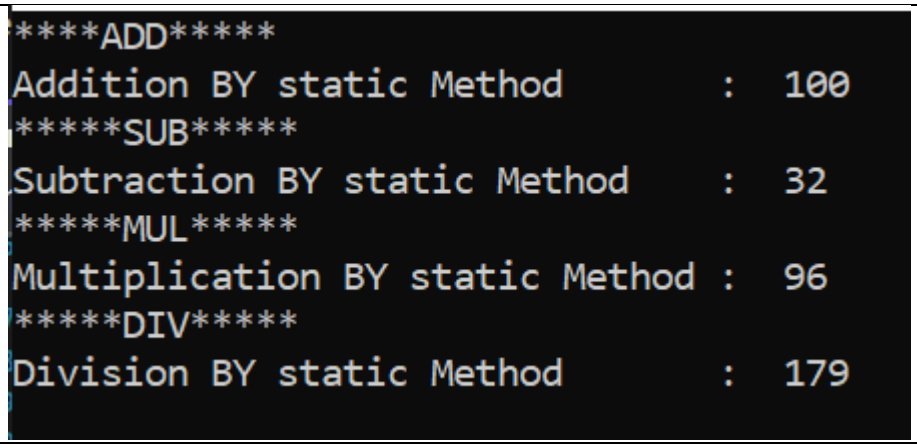
Create Mathematics class and add 3 static methods and call the methods in main method.

CODE:

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


namespace Day11Project4
{
   /// <summary>
   /// DONE BY: PAVAN
   /// PURPOSE: Create Mathematics class and add 3 static methods and call the methods in
main method
   /// </summary>
   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;
         }
         public static int Div(int a, int b)
         {
            return a / b;
         }
      }
      static void Main(string[] args)
```

```
    {
        Console.WriteLine("****ADD*****");
        Console.WriteLine("Addition BY static Method:  {0} ", Mathematics.Add(45, 55));

        Console.WriteLine("*****SUB*****");
        Console.WriteLine("Subtraction BY static Method:{0} ", Mathematics.Sub(56, 24));

        Console.WriteLine("*****MUL*****");
        Console.WriteLine("Multiplication BY static Method:{0} ", Mathematics.Mul(24, 4));

        Console.WriteLine("*****DIV*****");
        Console.WriteLine("Division BY static Method:  {0} ", Mathematics.Div(898, 5));

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

OUTPUT:



PROJECT: 4

Create a class Employee with only properties.

CODE:

```
using System;

/// <summary>
/// DONE BY: PAVAN
/// PURPOSE: Create a class Employee with only properties.
/// </summary>
namespace Day11Project3
{
  class Employee
```

```
{
    private int id;
    private string name;
    private string designation;
    public int Id
    {
        get {return id;}
        set {id = value;}
    }
    public string Name
    {
        get => name;
        set {name = value;}
    }
    public string Designation
    {
        set {designation = value;}
    }
    public int Salary
    {
        get
        {
            if (designation == "M")
                return 90000;
            else if (designation == "EMP")
                return 50000;
            else if (designation == "SECURITY")
                return 75000;
            else
                return 30000;
        }
    }
}
internal class Program
{
    static void Main(string[] args)
    {

        Employee e = new Employee();
        e.Id = 20;
        e.Name = "JK";
        e.Designation = "TL";
```

```
        Console.WriteLine($"{e.Id},{e.Name},{e.Salary}");

        Employee emp1 = new Employee();
        emp1.Id = 876;
        emp1.Name = "NARESH";
        emp1.Designation = "EMP";
        Console.WriteLine($"{emp1.Id}, {emp1.Name}, {emp1.Salary}");

        Employee emp2 = new Employee();
        emp2.Id = 222;
        emp2.Name = "SINGH";
        emp2.Designation = "SECURITY";
        Console.WriteLine($"{emp2.Id}, {emp2.Name}, {emp2.Salary}");

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

OUTPUT: