Day 11 - Assignment

By Manoj Karnatapu - NBHealthCareTechnologies

Assignment 1

Research and write the difference between abstract class and interface in C#

Answer

Abstract Class	Interface
1). Using abstract, we cannot achieve **Multiple Inheritance**.	1). Using an interface, we can achieve **Multiple Inheritance**.
2). It acts like a template.	2). It acts like a Contract.
3). It is a combination of both Normal and Abstract Methods, in an Abstract Class.	3). It consists only Abstract Methods. By default, any method in Interface is Abstract methods only.
4). An Abstract class doesn't provide full abstraction. i.e., both declaration and definition are not given in an abstract class.	4). An Interface does provide full abstraction. i.e., both declaration and definition are given in an interface.
5). It has different types of access modifiers.	5). We cannot use any access modifier i.e., public, private, protected, internal etc. because within an interface by default everything is public.
6). An abstract class allows you to create functionality that subclasses can implement or override.	6). An interface only allows you to define functionality, not implement it.
7). whereas a class can extend only one abstract class	7). Whereas a class can extend multiple interfaces.

.

Assignment 2

Write the 6 points about interface discussed in the class

Answer

Interface:

1. Interface is pure abstract class.
2. Interface name should start with Capital '**I**'.
3. Interface acts like a contract.
4. By default, the methods in interface are public and abstract.
5. Any class that is implementing interface must override all the methods.
6. Interface support multiple inheritance.

Assignment 3

Write C# Code for interfaces IShape -include classes Circle, Square, Triangle & Rectangle.

Code

```
using System;

// Author : Manoj.Karnatapu
// purpose : Write example program for interfaces discussed in the class IShape include the classes Cricle, Square, Triangle, Rectangle

// For Reference, check Day11Project1 in the same repository.

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

    class Circle : IShape
    {
        private int radius;
        public void ReadRadius()
        {
            Console.Write("\nEnter 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("\nEnter 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 length;
        private int width;
        public void ReadSide()
        {
            Console.Write("\nEnter Length of a Rectangle : ");
            length = Convert.ToInt32(Console.ReadLine());
            Console.Write("\nEnter width of a Rectangle : ");
            width = Convert.ToInt32(Console.ReadLine());
        }
        public int CalculatePerimeter()
        {
            return 2 * (length + width);
        }
        public int CalculateArea()
        {
            return length * width;
        }
    }
    class Triangle : IShape
    {
        private int side1;
        private int side2;
        private int side3;
        public void ReadSides()
        {
            Console.Write("\nEnter Side-1 of a Triangle : ");
            side1 = Convert.ToInt32(Console.ReadLine());
            Console.Write("\nEnter side-2 of a Triangle : ");
            side2 = Convert.ToInt32(Console.ReadLine());
            Console.Write("\nEnter side-3 of a Triangle : ");
            side3 = Convert.ToInt32(Console.ReadLine());
        }
        public int CalculatePerimeter()
        {
            return side1 + side2 + side3;
        }
        public int CalculateArea()
        {
            double semiperimeter = (side1 + side2 + side3) / 2;
            double Area = Math.Sqrt(semiperimeter * (semiperimeter - side1) *(semiperimeter
- side2) * (semiperimeter - side3));
            return Convert.ToInt32(Area);
        }
    }
    internal class Program
```

```
    {
        static void Main(string[] args)
        {
            Circle circle = new Circle();
            circle.ReadRadius();
            Console.WriteLine("------------------");
            Console.WriteLine($"\nThe Perimeter of Circle is :
{circle.CalculatePerimeter()}");
            Console.WriteLine($"\nThe Area of Circle is :  {circle.CalculateArea()}");
            Console.WriteLine("\n ------------------------------------------------------------
-- \n");

            Square square = new Square();
            square.ReadSide();
            Console.WriteLine("------------------");
            Console.WriteLine($"\nThe Perimeter of Square is :
{square.CalculatePerimeter()}");
            Console.WriteLine($"\nThe Area of Square is : {square.CalculateArea()}");
            Console.WriteLine("\n ------------------------------------------------------------
-- \n");

            Rectangle rectangle = new Rectangle();
            rectangle.ReadSide();
            Console.WriteLine("------------------");
            Console.WriteLine($"\nThe Perimeter of a Rectangle is :
{rectangle.CalculatePerimeter()}");
            Console.WriteLine($"\nThe Area of a Rectangle is :
{rectangle.CalculateArea()}");
            Console.WriteLine("\n ------------------------------------------------------------
-- \n");

            Triangle tri = new Triangle();
            tri.ReadSides();
            Console.WriteLine("------------------");
            Console.WriteLine($"\nThe Perimeter of a given Triangle is :
{tri.CalculatePerimeter()}");
            Console.WriteLine($"\nThe Area of a Triangle is : {tri.CalculateArea()}");
            Console.WriteLine("\n ------------------------------------------------------------
-- \n");

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

Output


C:\Windows\system32\
Enter Radius Value : 7
The Perimeter of Circle is : 44
The Area of Circle is : 154
Enter Side of a Square : 5
The Perimeter of Square is : 20
The Area of Square is : 25
Enter Length of a Rectangle : 10
Enter width of a Rectangle : 5
The Perimeter of a Rectangle is : 30
The Area of a Rectangle is : 50
Enter Side-1 of a Triangle : 4
Enter side-2 of a Triangle : 13
Enter side-3 of a Triangle : 15
The Perimeter of a given Triangle is : 32
The Area of a Triangle is : 24
Press any key to continue . . .

Assignment 4

Write the 7 points discussed about properties

Answer

Properties:

- Properties are similar to class variables, with get; & set; access modifiers.
- A Property with only **get**; → is called as Read Only.
- A Property with only **set**; → is called as Write Only.
- A Property with **get**; & **set**; → is Both Reading Values & Assigning Values.
- Properties are introduced to deal with private variables.
- Properties names start with **Upper Case** Letters.

Sample Property Example Code:

```
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; }
    }
 }
```

Assignment 5

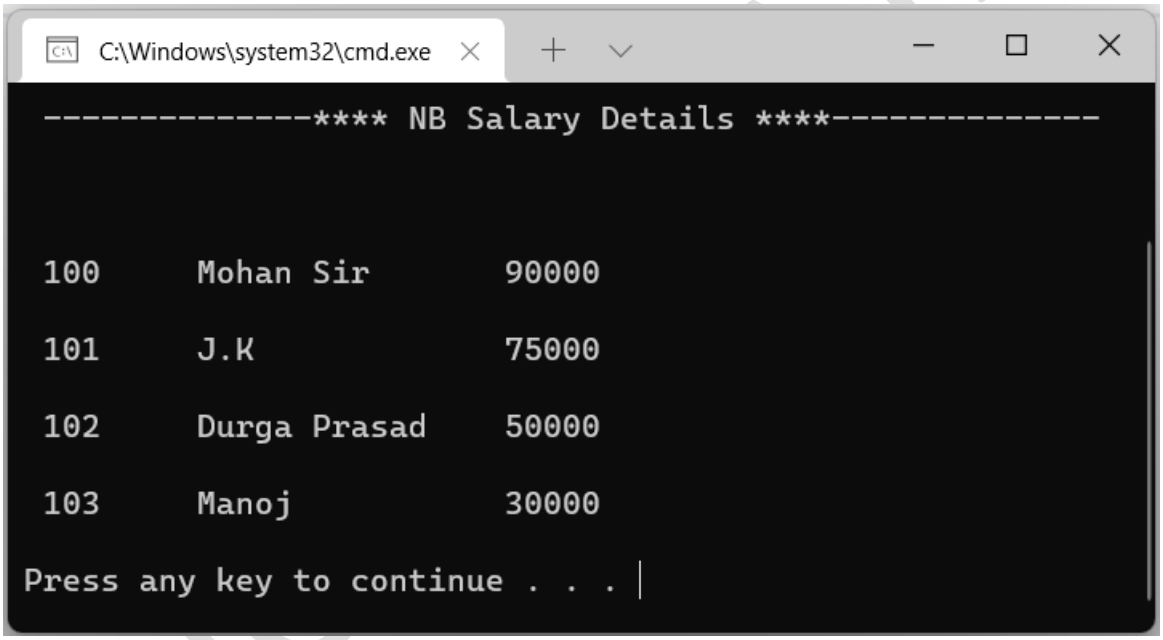
Write C# Code for properties using get; and set; access Modifiers.

Code

```
using System;

// Author : Manoj.Karnatapu
// Purpose : C# Code to illustrate properties using only get , only set & both set and get
Methods.

// for Reference, check Day11Project2 in the same repository.

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
        {
            // Setting only Write Only Property
            set { designation = value; }
        }
        public int Salary
        {
            get
            {
                if (designation == "M")
                    return 90000;
                else if (designation == "HR")
                    return 50000;
                else if (designation == "TL")
                    return 75000;
                else
                    return 30000;
            }
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("\n --------------**** NB Salary Details ****--------------
\n");
            Employee emp = new Employee();
            emp.Id = 100;
            emp.Name = "Mohan Sir";
            emp.Designation = "M";
            Console.WriteLine($"\n {emp.Id}\t {emp.Name}\t {emp.Salary}");

            Employee emp1 = new Employee();
            emp1.Id = 101;
```

```
            emp1.Name = "J.K";
            emp1.Designation = "TL";
            Console.WriteLine($"\n {emp1.Id}\t {emp1.Name}\t\t {emp1.Salary}");

            Employee emp2 = new Employee();
            emp2.Id = 102;
            emp2.Name = "Durga Prasad";
            emp2.Designation = "HR";
            Console.WriteLine($"\n {emp2.Id}\t {emp2.Name}\t {emp2.Salary}");

            Employee emp3 = new Employee();
            emp3.Id = 103;
            emp3.Name = "Manoj";
            emp3.Designation = "S";
            Console.WriteLine($"\n {emp3.Id}\t {emp3.Name}\t\t {emp3.Salary}");
            Console.ReadLine();
        }
    }
}
```

Output



Assignment 6

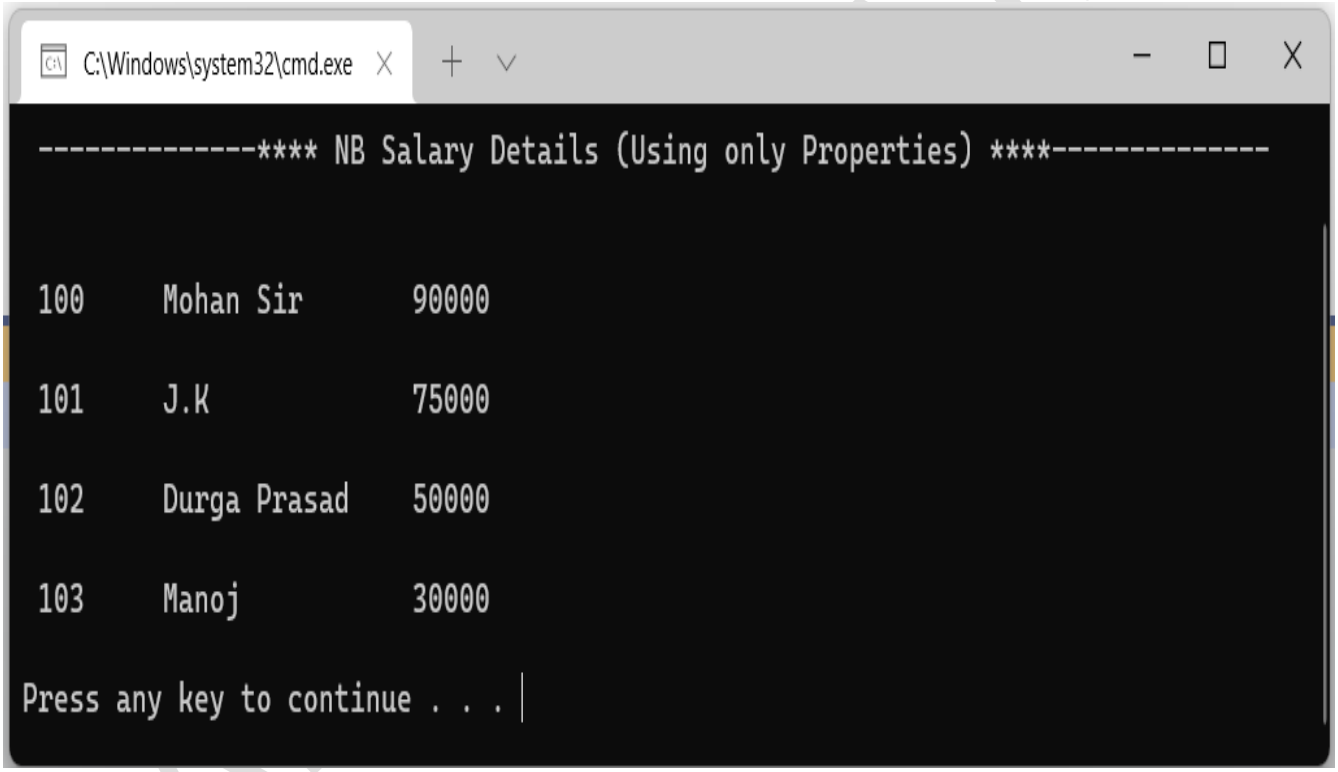
Write C# Code, for Employee /class with only Properties.

Code

```
using System;

// Author : Manoj.Karnatapu
// Purpose : C# Code to illustrate using only Properties (No private or public classlevel
Variables).

// for Reference, check Day11Project3 in the same repository.
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
        {
            // Setting only Write Only Property
            set { designation = value; }
        }
        public int Salary
        {
            get
            {
                if (designation == "M")
                    return 90000;
                else if (designation == "HR")
                    return 50000;
                else if (designation == "TL")
                    return 75000;
                else
                    return 30000;
            }
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("\n --------------**** NB Salary Details (Using only
Properties) ****--------------\n");
            Employee emp = new Employee();
            emp.Id = 100;
            emp.Name = "Mohan Sir";
            emp.Designation = "M";
            Console.WriteLine($"\n {emp.Id}\t {emp.Name}\t {emp.Salary}");

            Employee emp1 = new Employee();
            emp1.Id = 101;
            emp1.Name = "J.K";
```

```
            emp1.Designation = "TL";
            Console.WriteLine($"\n {emp1.Id}\t {emp1.Name}\t\t {emp1.Salary}");

            Employee emp2 = new Employee();
            emp2.Id = 102;
            emp2.Name = "Durga Prasad";
            emp2.Designation = "HR";
            Console.WriteLine($"\n {emp2.Id}\t {emp2.Name}\t {emp2.Salary}");

            Employee emp3 = new Employee();
            emp3.Id = 103;
            emp3.Name = "Manoj";
            emp3.Designation = "S";
            Console.WriteLine($"\n {emp3.Id}\t {emp3.Name}\t\t {emp3.Salary}");

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

Output



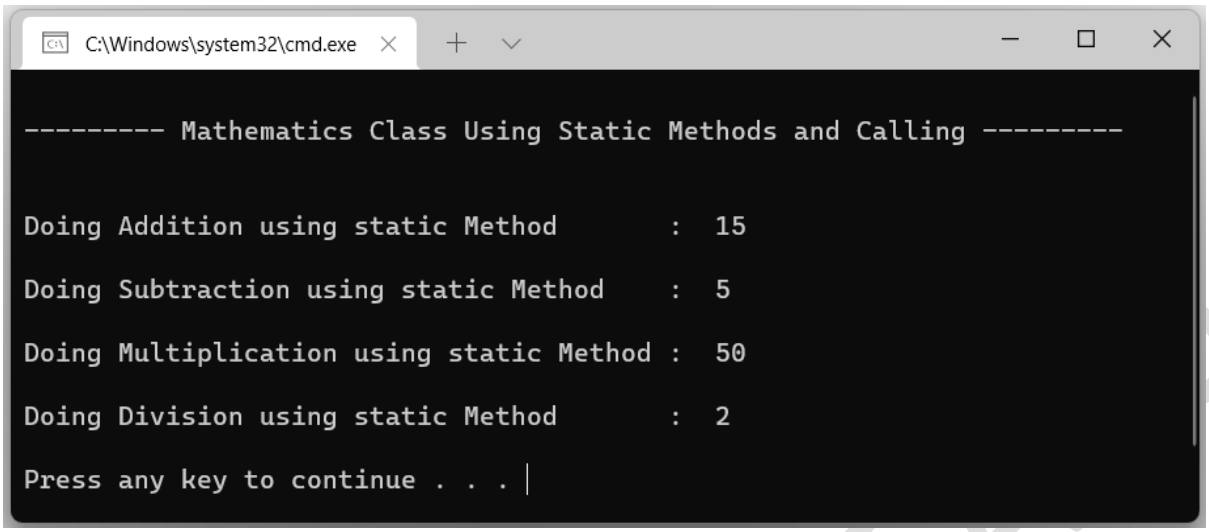
Assignment 7

Write C# Code, for Mathematics class, add 3 Static Methods – Call them in Main Method.

Code

```
using System;

// Author : Manoj.Karnatapu
// Purpose : Create Mathematics class and add 3 static methods and call
the methods in main method.

// for Reference, Check Day11Projct4 in the Same Repository.

namespace Day11Project4
{
    internal class Program
    {
        class Mathematics
        {
            public static int Add(int a, int b)
                { return a + b;}

            public static int Subtract(int a, int b)
                { return a - b;}

            public static int Multiplication(int a, int b)
                { return a * b;}
            public static int Division(int a, int b)
                { return a / b;}
        }
        static void Main(string[] args)
        {
            Console.WriteLine("\n--------- Mathematics Class Using
Static Methods and Calling ---------\n");
            // Calling Static Methods Using its Class Name
            Console.WriteLine("\nDoing Addition using static Method
:  {0} ",Mathematics.Add(10,5));
            Console.WriteLine("\nDoing Subtraction using static Method
:  {0} ",Mathematics.Subtract(10,5));
            Console.WriteLine("\nDoing Multiplication using static
Method :  {0} ",Mathematics.Multiplication(10, 5));
            Console.WriteLine("\nDoing Division using static Method
:  {0} ",Mathematics.Division(10, 5));

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

Output



Assignment 8

Research and understand when to create static methods

Answer

You should use static methods whenever you have a function that does not depend on a particular object of that class.

---------- OR ----------

If the method is not using Class level Variables, we can use static Method.

- ✓ A static method can be invoked directly from the class level
- ✓ A static method does not require any class object
- ✓ Any main() method is shared through the entire class scope so it always appears with static keyword.

~~~~~~~~~~~ THE END ~~~~~~~~~~~
~~~~~~~~~~~