| **DAY 11 : Assignment**  **By**  **Vihar D.** |
| --- |

| **Assignment 1** |
| --- |
| **Research and write the difference between abstract class and interface in C#** |
| **Answer :** |
| | **no.** | **ABSTRACT CLASS** | **INTERFACE** | | --- | --- | --- | |  | **Multiple Inheritance cannot be implemented** | **Multiple Inheritance can be implemented** | |  | **It's a combination of normal and abstract methods** | **It only consists of abstract methods, by default** | |  | **It does not provide complete abstraction (declaration and definition)** | **It provides complete abstraction (declaration and definition)** | |  | **It can use different access modifiers** | **It cannot use all access modifiers , except public which is a default** | |  | **It allows to create implementation that subclasses can implement** | **It only allows to define the implementation but cannot implement it** | |  | **It can only extend one abstract class at a time** | **It can extend multiple interfaces at a time** | |  | **It acts as a template** | **It acts as a contract** | |

| **Assignment 2** |
| --- |
| **Write the 6 points about interface discussed in the session** |
| **Answer :** |
| **INTERFACE :**   * **Interface is a pure abstract class.** * **Its name should start with caps ‘ I ‘.** * **It acts as a contract.** * **Methods in interface are public and abstract by default** * **Any class that is implementing an interface must override all the methods.** * **Interfaces support multiple inheritance.** |

| **Assignment 3** |
| --- |
| **Write a C# code for interfaces IShape - include classes Circle, Square, Rectangle & Triangle.** |
| **Answer :** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace interface\_isshapes  {  interface IShape  {  int calcPerimeter();  int calcArea();  }  *//Circle--------------------------------------------------*  class Circle : IShape  {  private int radius;  public void ReadRadius()  {  Console.Write("\nenter radius value : ");  radius = Convert.ToInt32(Console.ReadLine());  }  public int calcArea()  {  return 22 \* radius \* radius / 7;  }  public int calcPerimeter()  {  return 2 \* 22 \* radius / 7;  }  }  *//Square--------------------------------------------------*  class Square : IShape  {  private int side;  public void ReadSide()  {  Console.Write("\nenter side of square : ");  side = Convert.ToInt32(Console.ReadLine());  }  public int calcPerimeter()  {  return 4 \* side;  }  public int calcArea()  {  return side \* side;  }  }  *//Rectangle--------------------------------------------------*  class Rectangle : IShape  {  private int length;  private int width;  public void ReadSide()  {  Console.Write("\nenter length of rectangle : ");  length = Convert.ToInt32(Console.ReadLine());  Console.Write("\nenter width of a rectangle : ");  width = Convert.ToInt32(Console.ReadLine());  }  public int calcPerimeter()  {  return 2 \* (length + width);  }  public int calcArea()  {  return length \* width;  }  }  *//Triangle--------------------------------------------------*  class Triangle : IShape  {  private int a;  private int b;  private int c;  public void ReadSides()  {  Console.WriteLine("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Console.Write("\nenter side a of a Triangle : ");  a = Convert.ToInt32(Console.ReadLine());  Console.Write("\nenter side b of a Triangle : ");  b = Convert.ToInt32(Console.ReadLine());  Console.Write("\nenter side c of a Triangle : ");  c = Convert.ToInt32(Console.ReadLine());  }  public int calcPerimeter()  {  return a + b + c;  }  public int calcArea()  {  double semiperimeter = (a + b + c) / 2;  double Area = Math.Sqrt(semiperimeter \* (semiperimeter - a) \*  (semiperimeter - b) \* (semiperimeter - c));  return Convert.ToInt32(Area);  }  }  internal class Program  {  static void Main(string[] args)  {  Circle circ = new Circle();  circ.ReadRadius();  Console.WriteLine("------------------");  Console.WriteLine($"\nPerimeter of Circle is : {circ.calcPerimeter()}");  Console.WriteLine($"\nArea of Circle is : {circ.calcArea()}");  Console.WriteLine("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Square sq = new Square();  sq.ReadSide();  Console.WriteLine("------------------");  Console.WriteLine($"\nPerimeter of Square is : {sq.calcPerimeter()}");  Console.WriteLine($"\nArea of Square is : {sq.calcArea()}");  Console.WriteLine("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Rectangle rect = new Rectangle();  rect.ReadSide();  Console.WriteLine("------------------");  Console.WriteLine($"\nPerimeter of Rectangle is : {rect.calcPerimeter()}");  Console.WriteLine($"\nArea of Rectangle is : {rect.calcArea()}");  Console.WriteLine("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Triangle tri = new Triangle();  tri.ReadSides();  Console.WriteLine("------------------");  Console.WriteLine($"\nPerimeter of given Triangle is : {tri.calcPerimeter()}");  Console.WriteLine($"\nArea of Triangle is : {tri.calcArea()}");  Console.WriteLine("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Console.ReadLine();  }  }  } |
| **Output :** |
|  |

| **Assignment 4** |
| --- |
| **Write the 7 points about properties as discussed in the session.** |
| **Answer :** |
| **Properties :**   * **Properties are like class variables, with get; and set; access modifiers.** * **Properties are used to deal with private variables.** * **Property names start with uppercase letters.** * **A property with only get; is called Read-only.** * **A property with only set; is called Write-only.** * **A property with both get; and set; is called Read-Write ( reading and assigning values).**   **Property Example Code :**   | class Bikes  {  private string model ;  private string brand ;  private string type ;  private int powercc ;  public string Model  {  get { return id; }  set { id = value; }  }  public int Power  {  get { return name; }  set { name = value; }  }  } | | --- | |

| **Assignment 5** |
| --- |
| **Write a C# code for properties using get; and set; access modifiers.** |
| **Answer :** |
| **using System;**  **using System.Collections.Generic;**  **using System.Linq;**  **using System.Text;**  **using System.Threading.Tasks;**  **namespace get\_set\_accessmods**  **{**  **class Employee**  **{**  **private int emp\_id;**  **private string emp\_name;**  **private string emp\_designation;**  **private int emp\_sal;**  **public int ID**  **{**  **get { return emp\_id; }**  **set { emp\_id = value; }**  **}**  **public string Name**  **{**  **get { return emp\_name; }**  **set { emp\_name = value; }**  **}**  **public string Designation**  **{**  ***//Write only property***  **set { emp\_designation = value; }**  **}**  **public int Salary**  **{**  **get**  **{**  **if (emp\_designation == "M")**  **return 90000;**  **else if (emp\_designation == "HR")**  **return 50000;**  **else if (emp\_designation == "TL")**  **return 75000;**  **else**  **return 30000;**  **}**  **}**  **}**  **internal class Program**  **{**  **static void Main(string[] args)**  **{**  **Console.WriteLine("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* NB Salary Details \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");**  **Employee emp = new Employee();**  **emp.ID = 10;**  **emp.Name = "Mohan N";**  **emp.Designation = "M";**  **Console.WriteLine($"\n {emp.ID}\t {emp.Name}\t\t {emp.Salary}");**  **Employee emp1 = new Employee();**  **emp1.ID = 20;**  **emp1.Name = "JayaKrishna M";**  **emp1.Designation = "TL";**  **Console.WriteLine($"\n {emp1.ID}\t {emp1.Name}\t\t {emp1.Salary}");**  **Employee emp2 = new Employee();**  **emp2.ID = 30;**  **emp2.Name = "Usha S";**  **emp2.Designation = "HR";**  **Console.WriteLine($"\n {emp2.ID}\t {emp2.Name}\t\t {emp2.Salary}");**  **Employee emp3 = new Employee();**  **emp3.ID = 40;**  **emp3.Name = "Vihar D";**  **emp3.Designation = "SD";**  **Console.WriteLine($"\n {emp3.ID}\t {emp3.Name}\t\t {emp3.Salary}");**  **Console.ReadLine();**  **}**  **}**  **}** |
| **Output :** |
|  |

| **Assignment 6** |
| --- |
| **Write a C# code for employee class with only properties.** |
| **Answer :** |
| **using System;**  **using System.Collections.Generic;**  **using System.Linq;**  **using System.Text;**  **using System.Threading.Tasks;**  **namespace only\_property**  **{**  **class Employee**  **{**  **public int ID { get; set; }**  **public string Name { get; set; }**  **public string Designation { get; set; }**  **public int Salary**  **{**  **get**  **{**  **if (Designation == "M")**  **return 90000;**  **else if (Designation == "HR")**  **return 45000;**  **else if (Designation == "TL")**  **return 75000;**  **else**  **return 30000;**  **}**  **}**  **}**  **internal class Program**  **{**  **static void Main(string[] args)**  **{**  **Console.WriteLine("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* NB Salary Details ( using Properties ) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");**  **Employee emp = new Employee();**  **emp.ID = 10;**  **emp.Name = "Mohan N";**  **emp.Designation = "M";**  **Console.WriteLine($"\n {emp.ID}\t {emp.Name}\t\t {emp.Salary}");**  **Employee emp1 = new Employee();**  **emp1.ID = 20;**  **emp1.Name = "JayaKrishna M";**  **emp1.Designation = "TL";**  **Console.WriteLine($"\n {emp1.ID}\t {emp1.Name}\t\t {emp1.Salary}");**  **Employee emp2 = new Employee();**  **emp2.ID = 30;**  **emp2.Name = "Ushaa S";**  **emp2.Designation = "HR";**  **Console.WriteLine($"\n {emp2.ID}\t {emp2.Name}\t\t {emp2.Salary}");**  **Employee emp3 = new Employee();**  **emp3.ID = 40;**  **emp3.Name = "Vihar D";**  **emp3.Designation = "SD";**  **Console.WriteLine($"\n {emp3.ID}\t {emp3.Name}\t\t {emp3.Salary}");**  **Console.ReadLine();**  **}**  **}**  **}** |
| **Output :** |
|  |

| **Assignment 7** |
| --- |
| **Write a C# code for mathematics class and adds 3 static methods & calls them in the main method.** |
| **Answer :** |
| **using System;**  **using System.Collections.Generic;**  **using System.Linq;**  **using System.Text;**  **using System.Threading.Tasks;**  **namespace add\_3\_static**  **{**  **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 Mult(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("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\* Mathematics Class calling in Main Method \*\*\*\*\*\*\*\*\*\*\*\*\*\*");**    ***//Calling static methods using class name***  **Console.WriteLine("\n Addition : {0}", Mathematics.Add(30, 10));**  **Console.WriteLine("\n Subtraction : {0}", Mathematics.Sub(30, 10));**  **Console.WriteLine("\n Multiplication : {0}", Mathematics.Mult(30, 10));**  **Console.WriteLine("\n Division : {0}", Mathematics.Div(30, 10));**    **Console.WriteLine("\n");**  **Console.ReadLine();**  **}**  **}**  **}** |
| **Output :** |
|  |

| **Assignment 8** |
| --- |
| **Research and understand when to create static methods** |
| **Answer :** |
| **Static Methods are used whenever a function is independent of an object of a class.**  **A Static Method is used when the method isn't using any class level variables.**   * **A static method does not require any class object.** * **A static method can be invoked directly from the class level.** * **Any main() method is shared through the entire class scope so it always appears with a static keyword.** |