|  |
| --- |
| **Day11 Morning Assignments**  **By**  **Manoj Yekolla**  **07-Feb-2022** |

|  |
| --- |
| **1. Research and write the difference between**  **abstract class and interface in C#** |
| **Abstract Class :**   * Abstract class does not support multiple inheritance. * Abstract class contains Constructors. * Abstract class mainly use code re-usability. * Abstract class can contain Access modifiers for Subs, Functions and Properties. |
| **Interface :**   * Interface Support Multiple Inheritance. * Interface does not contain Multiple inheritance. * Member of interface can not be static. * An Interface can not have access modifiers by defalut everything is assumed as public. |

|  |
| --- |
| **2. Write the 6 points about interface discussed in the class ?** |
| * Interface is pure Abstract Class. * Interface name should start with I . * Interface acts like a Contract. * By default the methods in interface are public and abstract. * Any class that is implementing interface must override all the methods. * Interface support multiple inheritance. |

|  |
| --- |
| **3(a) . Write example program for interfaces discussed in the class IShape - ( Circle ) ?** |
| Code : |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day11\_Project1  {  interface Ishape  {  int CalculatePerimeter();  int CalculateArea();  }  class circle : Ishape  {  private int radius;  // Read Radius  public void ReadRadius()  {  Console.WriteLine("Enter a 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)  {  circle c = new circle();  c.ReadRadius();  Console.WriteLine("-------------------------------------------------------");  Console.WriteLine("Enter circle perimeter is :{0}",c.CalculatePerimeter());  Console.WriteLine("Enter a circle Area is :{0}",c.CalculateArea());  Console.ReadLine();    }  }  } |
| Output :  Screenshot (170) |

|  |
| --- |
| **3(b) Write example program for interfaces discussed in the class ishape - (Square) ?** |
| Code : |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day11\_Project1\_b\_  {  interface Ishape  {  int CalPerimeter();  int CalArea();  }  class square : Ishape  {  private int side;  public void ReadSide()  {  Console.WriteLine("Enter a Square Side :");  side = Convert.ToInt32(Console.ReadLine());  }  public int CalArea()  {  return side \* side;  }  public int CalPerimeter()  {  return 4 \* side;  }  }  internal class Program  {  static void Main(string[] args)  {  square obj =new square();  obj.ReadSide();  Console.WriteLine("square Area value is :{0}",obj.CalArea());  Console.WriteLine("square Perimeter value is :{0}",obj.CalPerimeter());  Console.ReadLine();  }  }  } |
| Output :  Screenshot (172) |

|  |
| --- |
| **3(c). Write example program for interfaces discussed in the class IShape**  **( Rectangle ) ?** |
| Code : |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day11\_Project1\_c\_  {  interface Ishape  {  int CalPerimeter();  int CalArea();  }  class Rectangle : Ishape  {  private int length;  private int width;  public void ReadValues()  {  Console.WriteLine("Enter Rectangle length :");  length = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("Enter Rectangle Width :");  width = Convert.ToInt32(Console.ReadLine());  }  public int CalArea()  {  return length \* width;  }  public int CalPerimeter()  {  return 2\*(length + width);  }  }  internal class Program  {  static void Main(string[] args)  {  Rectangle rect = new Rectangle();  rect.ReadValues();  Console.WriteLine("rectangle Area value is :{0}",rect.CalArea());  Console.WriteLine("rectangle Perimeter value is :{0}",rect.CalPerimeter());  Console.ReadLine();  }  }  } |
| Output :  Screenshot (174) |

|  |
| --- |
| **3(d) Write example program for interfaces discussed in the class ishape**  **( Triangle) ?** |
| Code : |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day11\_Project\_1\_d\_  {  interface Ishape  {  int CalPerimeter();  int CalArea();  }  class Triangle : Ishape  {  private int side1;  private int side2;  private int side3;  public void ReadTriangle()  {  Console.WriteLine("Enter a Side 1 triangle value :");  side1 = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("Enter a Side 2 triangle value :");  side2 = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("Enter a Side 3 triangle value :");  side3 = Convert.ToInt32(Console.ReadLine());  }  public int CalArea()  {  double semiperimeter = (side1 + side2 + side3) / 2;  double Area = Math.Sqrt(semiperimeter \* (semiperimeter - side1) \* (semiperimeter - side2) \* (semiperimeter - side3));  return Convert.ToInt32( Area);  }  public int CalPerimeter()  {  return side1 + side2 + side3;  }  }  internal class Program  {  static void Main(string[] args)  {  Triangle tri = new Triangle();  tri.ReadTriangle();  Console.WriteLine("Enter triangle Perimeter Value is :{0}",tri.CalPerimeter());  Console.WriteLine("Enter triangle Area Value is :{0}",tri.CalArea());  Console.ReadLine();  }  }  } |
| Output :  Screenshot (176) |

|  |
| --- |
| **4. Write the 7 points discussed about properties.** |
| * Properties are almost same as class variables with get; and set; * A Property with only Get is : Read only. * A Property with only Set is : Write only. * A Property with Get and Set : we can Read value and Assign Value. * Properties are introducing to deal with private variables. * Property name start with Upper Case. * A very simple example of properties :     Class Employee{  private int id;  private string name; //normal variables  private string designation;    Public int Id{  get{return id;} //property variables  set{id=value:}  }  } |

|  |
| --- |
| **5. Write sample code to illustrate properties as discussed in class. Id,name,designation,salary**    **id-get, set**  **name-get,set**  **designation-set (writeonly)**  **salary-get (get with some functionality)** |
| Code : |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day11\_Project2  {  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  {  //get { return designation; }  set { designation = value; }  }  public int Salary  {  get {  salary = (designation == "S") ? 20000 : 60000;  return salary;  }    }  }  internal class Program  {  static void Main(string[] args)  {  Employee employee = new Employee();  employee.Designation = "S";  Console.WriteLine(employee.Salary);  Console.ReadLine();  }  }  }  Output : |
| Screenshot (178) |

|  |
| --- |
| **6. Create a class Employee with only properties.** |
| Code :  class Employee  {  public int Id { get; set; }  public string Name { get; set; }  public int Salary { get; set; }  public string Designation { get; set; }  } |

|  |
| --- |
| **7. 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 Day11\_Project3  {  class Maths  {  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;  }  }  internal class Program  {  static void Main(string[] args)  {  Console.WriteLine(Maths.Add(3,3));  Console.ReadLine();  }  }  } |
| Output :  Screenshot (180) |

|  |
| --- |
| **8. Research and understand when to create static methods.** |
| * If a class unique variable values is there , that time static methods to use. * A static method does not require any class object. |