Day 11 Assignment

By K.SANJAY

|  |
| --- |
| 1. Research and write the difference between  abstract class and interface in C# |
| |  |  | | --- | --- | | Abstract Class | Interface | | Abstract class does not support multiple Inheritance | Interface supports multiple Inheritance | | The keyword of Abstract class is “abstract” | The keyword of Interface is “interface” | | Abstract class conatins Constructors | Interface does’nt contains Constructors | | An abstract classes can have non-abstract methods as well as abstarct method. | An interface can contain only deceleration of methods. | | Abstract implemented using extend. | Interface implemented using implement. | |

|  |
| --- |
| 2. Write the 6 points about interface discussed in the class |
| * It is also like a pure abstract class. * It acts like a contract. * The name stands with “I” (capital). * By default the methods in interface are public and abstract. * Any class that is implementing interface must be override. * Interface supports multiple inheritance. |

|  |
| --- |
| 3. 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\_assignment  {  interface IShape  {  int CalculatePerimeter();  int CalculateArea();  }  class Circle : IShape  {  private 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;  }  }  class Square : IShape  {  private int side;  public void ReadSide()  {  Console.WriteLine("Enter side:");  side = Convert.ToInt32(Console.ReadLine());  }  public int CalculateArea()  {  return side \* side;  }  public int CalculatePerimeter()  {  return 4 \* side;  }  }  class Rectangle : IShape  {  private int length;  public void ReadLength()  {  Console.WriteLine("Enter length:");  length = Convert.ToInt32(Console.ReadLine());  }  private int width;  public void ReadWidth()  {  Console.WriteLine("Enter width:");  width = Convert.ToInt32(Console.ReadLine());  }  public int CalculateArea()  {  return length \* width;  }  public int CalculatePerimeter()  {  return 2 \* (length + width);  }  }  class Triangle : IShape  {  private int Firstside;  public void ReadFirstside()  {  Console.WriteLine("Enter First side");  Firstside = Convert.ToInt32(Console.ReadLine());  }  private int Secondside;  public void ReadSecondside()  {  Console.WriteLine("Enter Second side");  Secondside = Convert.ToInt32(Console.ReadLine());  }  private int Lastside;  public void ReadLastside()  {  Console.WriteLine("Enter Last side");  Lastside = Convert.ToInt32(Console.ReadLine());  }  public int CalculateArea()  {  float s = (Firstside + Secondside + Secondside) / 2;  int Area = (int)Math.Sqrt((s \* (s - Firstside) \* (s - Secondside) \* (s - Lastside)));  return Area;  }  public int CalculatePerimeter()  {  return Firstside + Secondside + Lastside;  }  }  internal class Program  {  static void Main(string[] args)  {  Circle c = new Circle();  c.ReadRadius();  Console.WriteLine(c.CalculatePerimeter());  Console.WriteLine(c.CalculateArea());  Square s = new Square();  s.ReadSide();  Console.WriteLine(s.CalculatePerimeter());  Console.WriteLine(s.CalculateArea());  Rectangle r = new Rectangle();  r.ReadLength();  r.ReadWidth();  Console.WriteLine(r.CalculatePerimeter());  Console.WriteLine(r.CalculateArea());  Triangle t = new Triangle();  t.ReadFirstside();  t.ReadSecondside();  t.ReadLastside();  Console.WriteLine(t.CalculatePerimeter());  Console.WriteLine(t.CalculateArea());  Console.ReadLine();  }  }  } |
| Output |
|  |

|  |
| --- |
| 4. Write the 7 points discussed about properties. |
| * Properties are same as like class variables with get; and set; * A property we use get. Is only for read the value. * A property we use set. Is only for write the value or Assign the value. * A property with get and set => you can read the value and assign the value.   History of Properties   * Properties are introduced to deal with private variables. * Property name starts with uppercase. * Simple example for Properties are :   class Employee  {  private int id;  private string name;  private string designation;  public int Id  {  get { return id; }  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 Day\_11\_Assignment\_project\_2  {  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") ? 30000 : 60000;  return salary;  }    }  }  internal class Program  {  static void Main(string[] args)  {  Employee emp = new Employee();  emp.Designation = "m";  Console.WriteLine(emp.Salary);  Console.ReadLine();  }  }  } |

|  |
| --- |
| 6. Create a class Employee with only properties. |
| class Employee  {  private int id;  private string name;  private string designation;  private int salary;  public int Id { get; set; }    public string Name { get; set;}    public string Designation { get; set; }    public int Salary { get; set; }    }  internal class Program  {  static void Main(string[] args)  {  Employee emp = new Employee();  emp.Id = 11001;  Console.WriteLine($"id = {emp.Id}");  emp.Name = "kiran";  Console.WriteLine($"name = {emp.Name}");  emp.Designation = "developer";  Console.WriteLine($"designation = {emp.Designation}");  emp.Salary = 80000;  Console.WriteLine(emp.Salary);  Console.ReadLine();  }  }  } |

|  |
| --- |
| 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 Day\_11\_Assignment\_project\_2  {  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;  }  }  internal class Program  {  static void Main(string[] args)  {  Console.WriteLine(Mathematics.Add(10,15));  Console.WriteLine(Mathematics.sub(100, 50));  Console.WriteLine(Mathematics.mul(25, 2));  Console.ReadLine();  }  }  } |
| Output |
|  |

|  |
| --- |
| 8. Research and understand when to create static methods. |
| * The code in the method is not dependent on instance creation and is not using any instance variable. * A particular piece of code is to be shared by all the instance methods. * The definition of the method should not be changed or overridden. |