|  |
| --- |
| Day-11 Morning Assignment  By  U.Joshna  [7-2-2022] |

|  |
| --- |
| 1.Research and write the difference between abstract class and  Interface in C#? |
| Abstract class:  An [abstract class](https://www.geeksforgeeks.org/c-abstract-classes/) is a way to achieve the abstraction in C#. An Abstract class is never intended to be instantiated directly. This class must contain at least one abstract method, which is marked by the keyword or modifier  abstract in the class definition. The Abstract classes are typically used to define a base class in the class hierarchy.  Interface:  Like a class, [Interface](https://www.geeksforgeeks.org/c-interface/)can have methods, properties, events, and indexers as its members. But interfaces will contain only the declaration of the members. The implementation of interface’s members will be given by the class who implements the interface implicitly or explicitly. |

|  |
| --- |
| 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.Write example program for interfaces discussed in the class  I shape  Include the classes  Circle,Square,Traingle,Rectangle? |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day\_11\_Project\_1  {  interface IShape  {  int CalculatePerimeter();  int CalculateArea();  }  class Circle : IShape  {  private int radius;  public void ReadRadius()  {  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 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("Enter Length of a Rectangle : ");  length = Convert.ToInt32(Console.ReadLine());  Console.Write("Enter 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("Enter Side-1 of a Triangle : ");  side1 = Convert.ToInt32(Console.ReadLine());  Console.Write("Enter side-2 of a Triangle : ");  side2 = Convert.ToInt32(Console.ReadLine());  Console.Write("Enter 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 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.ReadSide();  Console.WriteLine(r.CalculatePerimeter());  Console.WriteLine(r.CalculateArea());  Triangle t = new Triangle();  t.ReadSides();  Console.WriteLine(t.CalculatePerimeter());  Console.WriteLine(t.CalculateArea());  Console.ReadLine();  }  }  }  Output: |
|  |
|  |
|  |
|  |
|  |

|  |
| --- |
| 4.Write the 7 points discussed about properties?  1.properties are almost some as class variables with get; and set;  2.A property with only get is readonly  3.A property with only set is writeonly  4.A property with get and set =>You can read value and assign the value  History of properties:  1.properties are introduced to deal with private variables  2.A very simple example of 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\_Project\_5  {    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  {  set { designation = value; }  }  public int Salary  {  get  {  if (designation == "Manager")  return 100000;  else if (designation == "HR")  return 70000;  else if (designation == "Team Leader")  return 50000;  else  return 25000;  }  }  }  internal class Program  {  static void Main(string[] args)  {  Console.WriteLine("\n NB Salary Details \n");  Employee emp = new Employee();  emp.Id = 101;  emp.Name = "Sagar";  emp.Designation = "Manager";    Console.WriteLine($"\n {emp.Id}\t {emp.Name}\t {emp.Salary}");  Employee emp1 = new Employee();  emp1.Id = 102;  emp1.Name = "Surya";  emp1.Designation = "Team Leader";  Console.WriteLine($"\n {emp1.Id}\t {emp1.Name}\t\t {emp1.Salary}");  Employee emp2 = new Employee();  emp2.Id = 103;  emp2.Name = "Sachin";  emp2.Designation = "HR";  Console.WriteLine($"\n {emp2.Id}\t {emp2.Name}\t {emp2.Salary}");  Employee emp3 = new Employee();  emp3.Id = 104;  emp3.Name = "Sunil";  emp3.Designation = "S";  Console.WriteLine($"\n {emp3.Id}\t {emp3.Name}\t\t {emp3.Salary}");  Console.ReadLine();  }  }  } |
| Output: |
|  |
|  |
|  |
|  |

|  |
| --- |
| 6.create a class Employee with only properties. |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Properties  {  class Employee  {  public int Id { get; set; }  public string Name { get; set; }  public int salary  {  get  {  return (Designation == "J") ? 40000 : 70000;  }  }  public string Designation { get; set; }  }  internal class Program  {  static void Main(string[] args)  {  Employee emp = new Employee();  emp.Id = 3;  emp.Name = "joshna";  emp.Designation = "N";  Console.WriteLine("Enter Id:{0}",emp.Id);  Console.WriteLine("Enter Name:{0}",emp.Name);  Console.WriteLine("Enter salary:{0}", emp.salary);  Console.WriteLine("Enter Designation:{0}",emp.Designation);  Console.ReadLine();  }  }  } |
| Output: |
|  |
|  |
|  |
|  |

|  |
| --- |
| 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\_project\_7  {  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");    Console.WriteLine("\n Doing Addition using static Method : { 0} ", Mathematics.Add(20, 5));  Console.WriteLine("\n Doing Subtraction using static Method : { 0} ", Mathematics.Subtract(20, 5));  Console.WriteLine("\n Doing Multiplication using static Method: { 0}", Mathematics.Multiplication(20, 5));  Console.WriteLine("\n Doing Division using static Method : { 0} ", Mathematics.Division(20, 5));  Console.ReadLine();  }  }  } |
| Output: |
|  |
|  |
|  |
|  |

|  |
| --- |
| 8.Research and understand when to create static methods?  .You should use static methods whenever you have a function that does not depend on a particular object of that class.There is no harm in adding the static keyword: it will not break any of the code that reffered to it |