|  |
| --- |
| Day11 morning assignment  By  Paluru Mounika  07-02-2022 |

**1.Write the difference between abstract class and interface.**

|  |  |
| --- | --- |
| **Abstract Class** | **Interface** |
| **It contains both declaration and definition part.**  **Multiple inheritance is not achieved by abstract class.**  **It contains constructor**  **It can contain static members.**  **It can contain different types of access modifiers like public, private, protected etc.**  **Abstract class can contain methods, fields, constants, etc.**  **A class can only use one abstract class.**  **Abstract class can contain methods, fields, constants, etc.**  **If many implementations are of the same kind and use common behavior, then it is superior to use abstract class.**  **It can be fully, partially or not implemented.** | **It contains only a declaration part.**  **Multiple inheritance is achieved by interface.**  **It deos not contain constructor**  **It does not contain static members.**  **It only contains public access modifier because everything in the interface is public.**  **Interface can only contains methods, properties, indexers, events.**  **A class can use multiple interface.**  **Interface can only contains methods, properties, indexers, events.**  **If many implementations only share methods, then it is superior to use Interface.**  **It should be fully implemented.** |

**2.Write the 6 points about interface discussed in the class.**

1.Interface is pure abstract class.

2.Interface name should start with I.

3.Interface act 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.

**3.write a example program for interface discussed in the class**

**Ishape**

**Include the classes circle,square,triangle,rectangle.**

|  |
| --- |
| Program: example program for interface |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //AUTHOR:PALURU MOUNIKA  //PURPOSE:EXAMPLE PROGRAM FOR INTERFACE  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  namespace day11project1  {  interface IShape  {  int CalculatePerimeter();  int CalculateArea();  }  /// <summary>  /// circle  /// </summary>  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;  }  }  /// <summary>  /// square  /// </summary>  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;  }  }  /// <summary>  /// rectangle  /// </summary>  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);  }  }  /// <summary>  /// triangle  /// </summary>  class Triangle : IShape  {  private int side;  public void ReadSide()  {  Console.WriteLine("Enter side:");  side = Convert.ToInt32(Console.ReadLine());  }  private int height;  public void ReadHeight()  {  Console.WriteLine("Enter height:");  height = Convert.ToInt32(Console.ReadLine());  }  private int breadth;  public void ReadBreadth()  {  Console.WriteLine("Enter breadth:");  breadth = Convert.ToInt32(Console.ReadLine());  }  public int CalculateArea()  {  return (breadth \* height) / 2;  }  public int CalculatePerimeter()  {  return side + side + side;  }  }  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.ReadSide();  t.ReadHeight();  t.ReadBreadth();  Console.WriteLine(t.CalculatePerimeter());  Console.WriteLine(t.CalculateArea());  Console.ReadLine();  }  }  } |
| Output: |
|  |

**4.write 7 points discussed about proparties.**

1.proparties are almost same class variable with get; set;

2.A property with only get is readonly.

3.A property with only get is write only.

4.A property with get and set => you can read and assine value.

5.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;}

}

}

3.property names start with upper case letters.

**5.write sample code to illustrate properties as discussed in the class**

**id,name, designation,salary**

|  |
| --- |
| **Program:**sample code to illustrate properties |
| **Code:** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //AUTHOR:PALURU MOUNIKA  //PURPOSE:SAMPLE CODE TO ILLUSTRATE PROPERTY  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  namespace Property\_Example  {  class Employee  {  private int id;  private string name;  private string designation;  private int salary;  /// <summary>  /// Id  /// </summary>  public int Id  {  get { return id; }  set { id = 101; }  }  /// <summary>  /// name  /// </summary>  public string Name  {  get { return name; }  set { name = "pavani"; }  }  /// <summary>  /// designation  /// </summary>  public string Designation  {  set { designation = value; }  }  /// <summary>  /// salary  /// </summary>  public int Salary  {  get  {  salary = (designation == "s") ? 30000 : 60000;  return salary;  }  }  }  internal class Program  {  static void Main(string[] args)  {  Employee emp = new Employee();  emp.Id = 101;  Console.WriteLine(emp.Id);  emp.Name = "pavani";  Console.WriteLine(emp.Name);  emp.Designation = "s";  Console.WriteLine(emp.Salary);  Console.ReadLine();  }  }  } |
| **Output:** |
|  |

**6.creat a class employee with only properties.**

|  |
| --- |
| **Program:**class employee with only properties |
| **Code:** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //AUTHOR:PALURU MOUNIKA  //PURPOSE:TO PRINT MATH OPARATIONS  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  namespace day11project3  {  class Employee  {  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 = 104;  Console.WriteLine($"id = {emp.Id}");  emp.Name = "madhavi";  Console.WriteLine($"name = {emp.Name}");  emp.Designation = "scrummaster";  Console.WriteLine($"designation = {emp.Designation}");  emp.Salary = 60000;  Console.WriteLine(emp.Salary);  Console.ReadLine();  }  }  } |
| **Output:** |
|  |

**7.Creat mathematics class and add 3 static methods and call the methods in main method.**

|  |
| --- |
| **Program:** |
| **Code:** |
| using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //AUTHOR:PALURU MOUNIKA  //PUPOSE:TO CREAT A MATHMATICS CLASS AND ADD 3 STATIC METHODES  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  namespace day11project4  {  class Mathematics  {  /// <summary>  /// Addition  /// </summary>  /// <param name="a"></param>  /// <param name="b"></param>  /// <returns></returns>  public static int Add(int a, int b)  {  return a + b;  }  /// <summary>  /// subtraction  /// </summary>  /// <param name="a"></param>  /// <param name="b"></param>  /// <returns></returns>  public static int Sub(int a, int b)  {  return a - b;  }  /// <summary>  /// multiplication  /// </summary>  /// <param name="a"></param>  /// <param name="b"></param>  /// <returns></returns>  public static int Mul(int a, int b)  {  return (a \* b);  }  }  internal class Program  {  static void Main(string[] args)  {  Console.WriteLine(Mathematics.Add(3, 6));  Console.WriteLine(Mathematics.Sub(7, 5));  Console.WriteLine(Mathematics.Mul(4, 2));  Console.ReadLine();  }  }  } |
| **Output:** |
|  |

**8.When to create static method.**

1.The definition of the method shoud not be changed or override.

2.A particular piece of code is to be shared by all the instance methods.

3.The code in the method is not dependent on instance creation and is not using any instance variable.