|  |
| --- |
| Day11 Assignment  By  J Siva Naga Prasanna  07-02-2022 |

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

|  |  |
| --- | --- |
| **Abstract Class** | **Interface** |
| **1.An abstract class doesn’t provide full abstraction.**  **2.we cannot achieve multiple inheritance while using abstract class**  **3.an abstract class can contain main method**  **4.abstract class can contains declaration and definition parts**  **5.abstract class can contain methods,fields.**  **6.abstract class contain some different access modifiers like public,private,protected** | **1.Interface provides full abstraction.**  **2.by using interface we can achieve multiple inheritance**  **3.interfaces can’t contain main method**  **4.interfaces contains only declaration part**  **5.interfaces contain methods and properties**  **6.interfaces can contain public access modifiers (everything in interface is public)** |

**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 acts 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 supports multiple inheritance.

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

**Ishape**

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

|  |
| --- |
| Program: |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace day11pro1  {  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 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 properties.**

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

2 .A property with only get is read only.

3. A property with only set is write only.

4. A property with get and set => you can read and assign 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;  namespace day11pro2  {  class Employee  {  private int id;  private string name;  private string designation;  private int salary;    public int Id  {  get { return id; }  set { id = 102; }  }    public string Name  {  get { return name; }  set { name = "siva"; }  }    public string 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();    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;  namespace day11pro3  {  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 = 501;  Console.WriteLine($"id = {emp.Id}");  emp.Name = "siva";  Console.WriteLine($"name = {emp.Name}");  emp.Designation = "developer";  Console.WriteLine($"designation = {emp.Designation}");  emp.Salary = 45000;  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;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace day11pro4  {  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(8, 7));  Console.WriteLine(Mathematics.Sub(5, 4));  Console.WriteLine(Mathematics.Mul(3, 9));  Console.ReadLine();  }  }  } |
| **Output:** |
|  |

**8. Research and understand When to create static method**.

1.static method can’t contain any local static variable

2.static methods can be overloaded

3.static method cannot access any non static variables unless they are explicitly passed as parameters, a static method can contain ref and out parameters