|  |
| --- |
| Day 11 Assignment  by  M.Pallavi |

1.research and write the difference between abstract class and interface in c.

|  |  |
| --- | --- |
| **Abstract Class** | **interface** |
| Abstract classes have static members. | Interface does not have static members |
| They have a constructor. | They don’t have a constructor. |
| It includes both a declaration and an explanation. | It includes only a declaration. |
| It includes methods, fields, constants, etc. | It only includes methods |
| Abstract class can be fully, partially or not implemented. | Interfaces can be fully implemented. |

1. Write the 6 points about interface discussed in the class.

* Interface is a Pure Abstract class
* Interface starts with ‘I’
* Interface acts like Contract
* By Default all methods in interface are abstract methods.
* It supports Multiple Inheritance .
* Any class that is implementing interface must override all the methods.

|  |
| --- |
| Program 1:  3 Write example program for interfaces discussed in the class Ishape, include the classes  Circle , Square, Triangle, Rectangle. |
| Code:  using System;  namespace day11prgm1  {  interface Ishape  {  int Calperimeter();  int Calarea();  }  class Circle : Ishape  {  int radius;  public void Readradius()  {  Console.WriteLine("enter radius of circle");  radius = Convert.ToInt32(Console.ReadLine());  }  public int Calarea()  {  return 22 \* radius \* radius / 7;  }  public int Calperimeter()  {  return 2 \* 22 \* radius \* radius / 7;  }  }  class Square : Ishape  {  private int side;  public void Readradius()  {  Console.WriteLine("enter length of side");  side = Convert.ToInt32(Console.ReadLine());  }  public int Calarea()  {  return side \* side;  }  public int Calperimeter()  {  return 4 \* side;  }  }  class Rectangle : Ishape  {  private int length, breadth;  public void Readsides()  {  Console.WriteLine("enter length rectangle");  length = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("enter breadth rectangle");  breadth = Convert.ToInt32(Console.ReadLine());  }  public int Calarea()  {  return length \* breadth;  }  public int Calperimeter()  {  return 2 \* (length + breadth);  }  }  class Triangle : Ishape  {  private int breadth, height;  public void ReadTsides()  {  Console.WriteLine("enter length of triangle");  breadth = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("enter breadth of triangle");  height = Convert.ToInt32(Console.ReadLine());  }  public int Calarea()  {  return breadth \* height / 2;  }  public int Calperimeter()  {  return breadth + height;  }  }  internal class program  {  static void Main(string[] args)  {  Circle c = new Circle();  c.Readradius();  Console.WriteLine(c.Calarea());  Console.WriteLine(c.Calperimeter());  Square s = new Square();  s.Readradius();  Console.WriteLine(s.Calarea());  Console.WriteLine(s.Calperimeter());  Rectangle r = new Rectangle();  r.Readsides();  Console.WriteLine(r.Calarea());  Console.WriteLine(r.Calperimeter());  Triangle t = new Triangle();  t.ReadTsides();  Console.WriteLine(t.Calarea());  Console.WriteLine(t.Calperimeter());  Console.ReadLine();  }  }  } |
|  |

|  |
| --- |
| 4. Write the 7 points discussed about properties. |
| Properties are 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 value is used to read value and assign value. |
| Initially ,properties are introduced to access private variables. |

|  |
| --- |
| Program 5: Write sample code to illustrate properties as discussed in class, id,name,designation,salary. |
| using System;  namespace pgmonPropert  {  class Employee  { /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Author: M.Pallavi  purpose: A sample code to illustrate properties \*/  private int id;  private string name;  private string designation;  private int salary;  public int Id//creating id property  {  get { return id; }  set { id = value; }  }  public string Name//creating name property  {  get { return name; }  set { name = value; }  }  public string Designation //creating designation property  {  get { return designation; }  set { designation = value; }  }  public int Salary  {  get  { salary = (designation == "s") ? 3000:6000;  return salary;  }  set { salary = value; }  }  }  internal class Program  {  static void Main(string[] args)  {  Employee emp=new Employee();  emp.Id = 25;  Console.WriteLine(emp.Id);  emp.Designation = "S";  Console.WriteLine(emp.Salary);    }  }  } |
| Output: |

|  |
| --- |
| 6. Create a class Employee with only properties. |
| Code:  using System;  namespace pgmonOnlyPro  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Author: M.Pallavi  purpose: A sample code using only properties \*/  internal class Employee1  {  public int Id { get; set; }  public string Name { get; set; }  public string Designation  { get; set; }  public int Salary  {  get  {  return (Designation == "S") ? 3000 : 6000;    }  }    static void Main(string[] args)  {    Employee1 emp = new Employee1();  emp.Id = 15;  Console.WriteLine(emp.Id);  emp.Designation = "L";  Console.WriteLine(emp.Salary);  }  }  } |
| Output: |

|  |
| --- |
| Program 7. Create Mathematics class and add 3 static methods and call the methods in main method. |
| Code:  using System;  namespace pgmonStaticMethods  { /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Author : M.Pallavi  purpose : Create Mathematics class and add 3 static methods and call the methods in main method.\*/  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 Mult(int a, int b)  {  return a \* b;  }  internal class Program  {  static void Main(string[] args)  {  Console.WriteLine("addition is "+Mathematics.Add(5, 6));  Console.WriteLine("substraction is "+Mathematics.Sub(9, 1));  Console.WriteLine("multiplication is "+Mult(5, 6));  } }  }  } |
| Output: |

|  |
| --- |
| 8. Research and understand when to create static methods.   * These are the methods , that can be called without creating an object of class. They are referenced by the **class name itself** or reference to the Object of that class. * Static method are associated with the class in which they reside. * They are designed with the aim to be shared among all objects created from the same class. * **Static methods can’t be overridden** because they are associated with class not with the object. * Static methods are normally faster to invoke on the call stack than instance method |