***DAY 11 ASSIGNMENT PRESENTED***

***BY***

***POTUKANUMA JEEVITHA***

***07-02-2022***

|  |  |
| --- | --- |
| **1.Reasearch and write the difference between abstract class and interface in C#?** | |
| **Ans:-** | |
| **ABSTRACT CLASS** | **INTERFACE CLASS** |
| * It contains both declaration and definition part. | * It contains only a declaration part. |
| * Multiple inheritance is not achieved by abstract class. | * Multiple inheritance is achieved by inheritance. |
| * It contain constructor. | * It does not contain constructor. |
| * It can contain static members. | * It does not contain static members. |
| * It can contain different types of access modifiers like public, private, protected etc. | * It only contains public access modifier because everything in the interface is public. |
| * A class can only use one abstract class. | * A class can use multiple interface. |
| * It can be fully , partially or not implemented. | * It should be fully implemented. |
| * The performance of an abstract class is fast. | * The performance of interface is slow because it requires time to search actual method in the corresponding class. |

|  |
| --- |
| **2.write the 6 points about interface discussed in the class?** |
| **Ans:-** |
| * Interface is pure abstract class. * Interface name should start with “I”. * Interface acts like a contract. * By default the methods in interface. * Any class that is implementing interface must override all the abstract method. * Interface support multiple inheritance. |

|  |
| --- |
| **3.Write example program for interfaces discussed in the class**  **IShape**  **Include the classes**  **Circle,square,triangle,rectangle.** |
| **Code:-** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace interfaces\_circle\_square\_rectangle\_triangle  {    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 the 7 points discussed about properties?** |
| **Ans:-** |
| * Properties are almost same as class variables with “get; and set;”. * A property with only get is “readonly”. * A property with only set is “writeonly”. * A property with get and set => you can read value and assign the value. * HISTORY OF PROPERTIES: * Properties are introduced to deal with private variable. * A very simple example of properties are: * Class Employee   {  Private int ID;  Private string name;  Private string designation;  Public int ID  {  get { return id; }  get { id = value;}  }  }   * Property names start with upper case. |

|  |
| --- |
| **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 functonality)** |
| **Code:-** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace set\_get\_id\_name\_designation\_salary  {  class Employee  {  private int id;  private string name;  private string designation;  private int salary;  public int Id  {  get { return id; }  set { id = 123; }  }  public string Name  {  get { return name; }  set { name = "JOHN"; }  }  public string Designation  {    set { designation = value; }  }  public int Salary  {  get  {  salary = (designation == "S") ? 25000 : 50000;  return salary;  }    }  }  internal class Program  {  static void Main(string[] args)  {  Employee emp = new Employee();  emp.Id = 123;  Console.WriteLine(emp.Id);  emp.Name = "JOHN";  Console.WriteLine(emp.Name);  emp.Designation = "M";  Console.WriteLine(emp.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 create\_emp\_with\_only\_properties  {  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 = 123;  Console.WriteLine($"id = {emp.Id}");  emp.Name = "JOHN";  Console.WriteLine($"name = {emp.Name}");  emp.Designation = "developer";  Console.WriteLine($"designation = {emp.Designation}");  emp.Salary = 50000;  Console.WriteLine(emp.Salary);  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 math\_class  {  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(30, 14));  Console.WriteLine(Mathematics.Mul(14, 7));  Console.ReadLine();  }  }  } |
| **Output:-** |
|  |

|  |
| --- |
| **8.Research and understand when to create static methods.** |
| **Ans:-** |
| * 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. |

|  |
| --- |
| **Write example program for interfaces discussed in the class**  **Scalene triangle** |
| **Code:-** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace interfaces\_scalene\_triangle  {  interface IShape  {  int CalculatePerimeter();  int CalculateArea();  }  class Scalene : IShape  {  private int side1;  private int side2;  private int side3;  public void ReadSide()  {  Console.WriteLine("enter side1:");  side1 = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("enter side2:");  side2 = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("enter side3:");  side3 = Convert.ToInt32(Console.ReadLine());  }  public int CalculateArea()  {  int s = (side1 + side2 + side3) / 2;  int area = (int)Math.Sqrt((s \* (s - side1) + (s - side2) + (s - side3)));  return area;  }  public int CalculatePerimeter()  {  return side1+side2 + side3;  }  }  internal class Program  {  static void Main(string[] args)  {  Scalene s= new Scalene();  s.ReadSide();    Console.WriteLine(s.CalculatePerimeter());  Console.WriteLine(s.CalculateArea());  Console.ReadLine();  }  }  } |
| **Output:-** |
|  |