Day 11 Assignment

By

VARUN SAI KUMAR CHEGONI

NB Healthcare and Technology

Date: 07 Feb 2022

**Topics**

**C# Interface**

**C# Properties**

**C# Static Class, Variable and Method**

**Content**

|  |  |  |
| --- | --- | --- |
| S.No | Content | Page No. |
| 1 | Research and write the difference between abstract class and interface in C# |  |
| 2 | Write the 6 points about interface discussed in the class |  |
| 3 | Write example program for interfaces discussed in the class IShape include the classes Cricle, Square, Triangle, Rectangle |  |
| 4 | Write the 7 points discussed about properties. |  |
| 5 | Write sample code to illustrate properties as discussed in class. |  |
| 6 | Create a class Employee with only properties. |  |
| 7 | Create Mathematics class and add 3 static methods and call the methods in main method. |  |
| 8 | Research and understand when to create static methods. |  |

|  |
| --- |
| 1. Research and write the difference between abstract class and interface in C# |
| Answer: |
| |  |  | | --- | --- | | **Abstract Class** | **Interface** | | 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 interface. | | It contains [constructor](https://www.geeksforgeeks.org/c-sharp-constructors/). | It does not contain [constructor](https://www.geeksforgeeks.org/c-sharp-constructors/). | | It can contain static members. | It does not contain static members. | | 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. | | It is used to implement the core identity of class. | It is used to implement peripheral abilities of class. | | A class can only use one abstract class. | A class can use multiple interfaces. | | If many implementations are of the same kind and use common behaviour, then it is superior to use abstract class. | If many implementations only share methods, then it is superior to use Interface. | | Abstract class can contain methods, fields, constants, etc. | Interface can only contain methods, properties, indexers, events. | | It can be fully, partially or not implemented. | It should be fully implemented. | |  |  | |

|  |
| --- |
| 2. Write the 6 points about interface discussed in the class |
| Answer: |
| INTERFACE:   1. Interface is like a 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 abstract methods. 6. 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 IshapeusingInterface  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Author : Varun Sai Kumar Chegoni.  \* Purpose : Write example program for interfaces discussed in the class IShape include the classes Circle, Square, Triangle, Rectangle.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /// <summary>  /// interface containg calculation of perimeter and area method.  /// </summary>  interface IShape  {  int CalculatePerimeter();  int CalculateArea();  }  /// <summary>  /// class Circle inherits interface IShape reads data and calculate perimeter and area.  /// </summary>  class Circle : IShape  {  int radius; // Variable Declaration  public void ReadData()  {  Console.WriteLine("Enter Radius of Circle :");  radius = Convert.ToInt32(Console.ReadLine()); // Read Data from User  }  public int CalculatePerimeter()  {  return 2 \* 22 \* radius/7; // Logic  }  public int CalculateArea()  {  return 22 \* radius \*radius/7; // Logic  }  }  /// <summary>  /// class Square inherits interface IShape reads data and calculate perimeter and area.  /// </summary>  class Square : IShape  {  int side; // Variable Declaration  public void ReadData()  {  Console.WriteLine("Enter Side of Square :");  side = Convert.ToInt32(Console.ReadLine()); // Read Data from User  }  public int CalculatePerimeter()  {  return 4 \* side; // Logic  }  public int CalculateArea()  {  return side \* side; // Logic  }  }  /// <summary>  /// class Triangle inherits interface IShape reads data and calculate perimeter and area.  /// </summary>  class Triangle : IShape  {  int a, b, c;  int height;  int ba; // Variable Declaration  public void ReadData()  {  Console.WriteLine("Enter Side A of Triangle:");  a = Convert.ToInt32(Console.ReadLine()); // Read Data from User  Console.WriteLine("Enter Side B of Triangle:");  b = Convert.ToInt32(Console.ReadLine()); // Read Data from User  Console.WriteLine("Enter Side C of Triangle:");  c = Convert.ToInt32(Console.ReadLine()); // Read Data from User  }  public int CalculatePerimeter()  {  return a + b + c; // Logic  }  public void ReadDataA()  {  Console.WriteLine("Enter Height of Triangle:");  a = Convert.ToInt32(Console.ReadLine()); // Read Data from User  Console.WriteLine("Enter Base of Triangle:");  b = Convert.ToInt32(Console.ReadLine()); // Read Data from User  }  public int CalculateArea()  {  return height \* ba / 2; // Logic  }  }  /// <summary>  /// class Rectangle inherits interface IShape reads data and calculate perimeter and area.  /// </summary>  class Rectangle : IShape  {  int l, b; // Variable Declaration  public void ReadData()  {  Console.WriteLine("Enter Length of Rectangle:");  l = Convert.ToInt32(Console.ReadLine()); // Read Data from User  Console.WriteLine("Enter Breadth of Rectangle:");  b = Convert.ToInt32(Console.ReadLine()); // Read Data from User  }  public int CalculatePerimeter()  {  return 2 \* (l + b); // Logic  }  public int CalculateArea()  {  return l \* b; // Logic  }  }  internal class Program  {  static void Main(string[] args)  {  Circle c = new Circle();  c.ReadData();  Console.WriteLine("Circumference of Circle is " + c.CalculatePerimeter());  Console.WriteLine("Area of Circle is " + c.CalculateArea()); // Print Output  Square s = new Square();  s.ReadData();  Console.WriteLine("Perimeter of Square is " + s.CalculatePerimeter());  Console.WriteLine("Area of Square is " + s.CalculateArea()); // Print Output  Triangle t = new Triangle();  t.ReadData();  Console.WriteLine("Perimeter of Triangle is " + t.CalculatePerimeter());  t.ReadDataA();  Console.WriteLine("Area of Triangle is " + s.CalculateArea()); // Print Output  Rectangle r = new Rectangle();  r.ReadData();  Console.WriteLine("Perimeter of Rectangle is " + r.CalculatePerimeter());  Console.WriteLine("Area of Rectangle is " + r.CalculateArea()); // Print Output  Console.ReadLine();  }  }  } |
| Output : |
|  |
| 4. Write the 7 points discussed about properties. |
| Answer: |
| 1. Properties are almost same as class variables with get; and 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 implies you can read value and assign the value. 5. Properties are introduced to deal with private variables. 6. A very simple example of properties is  class Employee  {  private int id;  private string name;  private string designation;   public int Id  {  get { return id; }  set { id = value; }  }  } 7. Property name 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 functionality) |
| Code : |
|  |
| Output : |
|  |

|  |
| --- |
| 6. Create a class Employee with only properties. |
| Code : |
|  |
| Output : |
|  |

|  |
| --- |
| 7. Create Mathematics class and add 3 static methods and call the methods in main method. |
| Code : |
|  |
| Output : |
|  |

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