|  |
| --- |
| **Day 11(7 feb) Assignment by**  **P RamaKrishna** |

|  |
| --- |
| **1.Write the 6 points about interface discussed in the class**  Interface:   * Interface is a pure Abstract Class. * Interface name should start with I. * Interface acts like a contract. * By default the method in interface are public and abstract. * Any Class that is implementing interface must override all the methods . * Interface Support multiple inheritance. |

|  |
| --- |
| **2 .Write the 7 points discussed about properties.** |
| * Properties are almost same as class variable with get; & set; . * A property with only get is read only. .. * A property with only set is written only. * A property with get and set is you can read value and assign value.   History of properties :   * Properties are introduced to deal with private variables . * A very simple syntax of properties are:   Class employee  {  Private int id;  Private string name;  Private string designation;  Public int Id  {  Get{return id;}  Set{id =value;}  }  }   * Property name start with upper case. |

|  |
| --- |
| 3 .Research and write the difference between abstract class and interface in C# |
| Abstract class :   * Abstract class allow you to create functionality that subclass can implement Or override * Class can extend only one abstract class.   Interface:   * Interface allow you to define functionality , not implement it. * It can take advantage of multiple interface. |

|  |
| --- |
| **4 .Research and understand when to create Static methods.** |
| * When the function does not make use of any member variable. * When using factory methods to create objects. * When you are Controlling or otherwise keep track of number of instantiations of class * When declaring constants. |

|  |
| --- |
| 5 .Create Mathematics class and add 3 static methods and call the  methods in main method. |
| Code: 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(20,30));  Console.WriteLine(Mathematics.sub(55, 28));  Console.WriteLine(Mathematics.mul(5,8));  Console.ReadLine();  }  }  } |
| Output: |

|  |
| --- |
| 6 .Write example program for interfaces discussed in the class  IShape  include the classes  Cricle, Square, Triangle, Rectangle |
| Code: using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Interface  {  interface Ishape  {  int Calculateperimeter();  int Calculatearea();  }  class Circle : Ishape  {  int radius;  public void Readdradius()  {  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  {  int sides;  public void Readsides()  {  Console.WriteLine("enter sides value");  sides = Convert.ToInt32(Console.ReadLine());  }  public int Calculatearea()  {  return 4 \* sides;  }  public int Calculateperimeter()  {  return sides \* sides;  }  }    class Rectangle : Ishape  {  int length, breadth;  public void Readlength()  {  Console.WriteLine("enter length");  length = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("enter breadth");  breadth = Convert.ToInt32(Console.ReadLine());  }  public int Calculatearea()  {  return 2\*(length + breadth);  }  public int Calculateperimeter()  {  return length \* breadth;  }  }  internal class Program  {  static void Main(string[] args)  {  Circle c = new Circle();  c.Readdradius();  Console.WriteLine(c.Calculatearea());  Console.WriteLine(c.Calculateperimeter());  Square s = new Square();  s.Readsides();  Console.WriteLine(s.Calculatearea());  Console.WriteLine(s.Calculateperimeter());      Rectangle r = new Rectangle();  r.Readlength();  Console.WriteLine(r.Calculatearea());  Console.WriteLine(r.Calculateperimeter());  Console.ReadLine();  }  }  } |
| Output : |

|  |
| --- |
| 7 .Write sample code to illustrate properties as discussed in class.  id  name  designation  salary |
| Code: using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace emloyee\_class  {  class employee  {  private int id;  private string name;  private string desigination;  private int salary;  public int ID  {  get { return id; }  set { id = value; }  }  public string Name  {  get { return name; }  set { name = value; }  }  public string Desigination  {  get { return desigination; }  set { desigination = value; }  }  public int Salary  {  get  {  salary=(desigination=="s") ? 3000 : 6000;  return salary;  }  }  }  internal class Program  {  static void Main(string[] args)  {  employee emp =new employee();  emp.Desigination="m";  Console.WriteLine(emp.Salary);  emp.Desigination="s";  Console.WriteLine(emp.Salary);  Console.ReadLine();  }  }  } |
| Output: |

|  |
| --- |
| Create a class Employee with only properties. |
| Code: Code: using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace emloyee\_class  {  class employee  {  private int id;  private string name;  private string desigination;  private int salary;  public int ID  {  get { return id; }  set { id = value; }  }  public string Name  {  get { return name; }  set { name = value; }  }  public string Desigination  {  get { return desigination; }  set { desigination = value; }  }  public int Salary  {  get { return salary;}  }  }  internal class Program  {  static void Main(string[] args)  {  employee emp =new employee();  Console.ReadLine();  }  }  } |