Feb 10th Assignment

By

Surya Teja Chandolu

|  |
| --- |
| 1. Research and write what is the use of sealed class. WACP to illustrate sealed class. |
| * The sealed classes prevents users from inheriting the class. The sealed keyword can be used to seal a class. The keyword indicates to the compiler that the class is closed and so cannot be extended. * If you want to declare a method as sealed, its base class must declare it as virtual. |
| Code: |
| using System;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Author: Surya Teja  \* Purpose: Sealed Class  \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  namespace SealedClass  {  sealed class Customer  {  private string userName;  private string password;  public void ReadData()  {  Console.Write("Enter User Name: ");  userName = Console.ReadLine();  Console.Write("Enter Password: ");  password = Console.ReadLine();  }  public void PrintData()  {  Console.WriteLine($"Username is {userName}.");  Console.WriteLine($"Password is {password}");  }  }  internal class Program  {  static void Main(string[] args)  {  Customer cu = new Customer();  cu.ReadData();  cu.PrintData();  Console.ReadLine();  }  }  } |
| Output: |
|  |

|  |
| --- |
| 1. Research and write what is the difference between normal properties and auto-implemented properties.  * WACP to illustrate normal properties. * WACP to issustrate auto-implemented properties |
| Properties: A property is a member that provides a flexible mechanism to read, write, or compute the value of a private field. Properties can be used as if they are public data members, but they are actually special methods called accessors. This enables data to be accessed easily and still helps promote the safety and flexibility of methods.  Auto-Implemented Properties: In C# 3.0 and later, auto-implemented properties make property-declaration more concise when no additional logic is required in the property accessors. They also enable client code to create objects. |
| Code: |
| using System;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Author: Surya Teja  \* Purpose: Property  \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  namespace AutoProperty  {  class CalculateSpeed  {  private int time;  private int distance;  //Normal Properties  public int Time { set { time = value; } }  public int Distance { set { distance = value; } }  public int Speed { get { return distance / time; } }  //Auto - Implemented Properties  public int Velocity { get { return distance / time; } set {} }  }  internal class Program  {  static void Main(string[] args)  {  CalculateSpeed calc = new CalculateSpeed();  calc.Time = 20;  calc.Distance = 60;  Console.WriteLine($"Speed is {calc.Speed}");  Console.ReadLine();  }  }  } |
| Output: |
|  |

|  |
| --- |
| 1. Research and fix the below issue:   interface IRules  {  int Age { get; set; }  int add(int a, int b);  public void PrintHi()  {  Console.WriteLine("Hi");  }  } |
|  |

|  |
| --- |
| 1. WACP to check if the number is prime or not using logic discussed in the class (HINT: use break;) |
| Code: |
| using System;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Author: Surya Teja  \* Purpose: Prime  \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  namespace Prime  {  internal class Program  {  static void Main(string[] args)  {  int input, i;  Console.Write("Enter number to calculate prime: ");  input = Convert.ToInt32(Console.ReadLine());  for (i = 2; i < input; i++)  {  if (input % i == 0)  break;  }  if (i == input)  Console.WriteLine($"{input} is prime number.");  else  Console.WriteLine($"{input} is not prime number.");  Console.ReadLine();  }  }  } |
| Output: |
|  |

|  |
| --- |
| 1. print numbers from 1 to 30 and skip the numbers divisible by 3 (HINT : use continue;) |
| Code: |
| using System;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Author: Surya Teja  \* Purpose: ContinueAndBreak  \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  namespace ContinueAndBreak  {  internal class Program  {  static void Main(string[] args)  {  Console.Write("Numbers divisible by 3: ");  for(int i = 1; i <= 30; i++)  {  if (i % 3 == 0)  continue;  Console.Write($"{i} ");  }  Console.ReadLine();  }  }  } |
| Output: |
|  |

|  |
| --- |
| 1. Find the first number after 1000 which is divisible by 97. (HINT : use for loop and break) |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Author: Surya Teja  \* Purpose: ContinueAndBreak  \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  namespace ContinueAndBreak  {  internal class Program  {  static void Main(string[] args)  {  int i;  for(i = 1000; i <= 2000; i++)  {  if (i % 97 == 0)  break;  }  Console.WriteLine($"First number after 1000 divisible by 97 is {i}");  Console.ReadLine();  }  }  } |
| Output: |
|  |