Day 14 Assignment (10th Feb 2022)

by

--RamCharan

1.Research and write what is use of sealed class.

- Sealed class is same as normal class.
- It cannot be used as parent class or a base class.
- A sealed is a class that can not be inherited by any class but can be instantiated.
- The design intent of a sealed class is to indicate that the class is specialized and there is no need to extend through inheritance to override its behaviour.
- It can have properties, static variables.

WACP to illustrate sealed class.

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day14Project1
{
    //Author:Rc
    /****Purpose:Sealed class Implementation****/
    sealed class Rc
    {
        private int age;//private variables
        public int id { get; set; } //Property
        public static string name="Ram"; //Static variables
        public void Surya() //Method
        {
            Console.WriteLine("Hi Surya");
            Console.WriteLine("Id is:{0}",id = 2);
```

```
        Console.WriteLine("Age is:{0}",age=22);
      }
    }
    internal class Program
    {
      static void Main(string[] args)
      {
        Rc r=new Rc();
        r.Surya();
        Console.WriteLine("Common friend is : {0}",Rc.name);

        Console.ReadLine();
      }
    }
}
```

Output:

```
C:\Users\91807\source\repos\Day14Project1\Day14Project1\bin\Debug\Day14Project1.exe
Hi Surya
Id is:2
Age is:22
Common friend is : Ram
```

2.Research and write differences between normal and auto properties.

Auto Properties:

- Automatic Property is a property that has backing field generated by compiler.
- It saves developer from writing primitive getters and setters that just return value of backing field or assign to it.
- Ex:

  public int id{get;set;}

Normal Properties:

- Normal properties are the properties with out having any backing field by compiler.
- Instead we have to write primitive getters and setters and we have to assign value of backing fields.

- Ex:

```
class A //normal properties
{
    private int id;
    public int Id
    {
        get
        {
            return id;
        }
        set
        {
            id = value;
        }
    }
```

WACP to illustrate normal and automatic Properties.

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day14Project2
{
    //Author: Rc
    /***Purpose: To implement normal properties and auto-implemented properties****/
    class A //normal properties
    {
        private int id;
        public int Id
        {
            get
            {
                return id;
            }
            set
            {
                id = value;
            }
        }
        public int Name
        {
            set
            {
                Name=value;
            }
        }


    }
    class B //AutoImplemented
    {
        public int Id { get; set; }
        public string Name
        {
            get
            {
                return "Ram";
            }
```

```
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("For normal properties:");
            A a = new A();
            a.Id = 8000;
            Console.WriteLine(a.Id);
            Console.WriteLine("\n");
            Console.WriteLine("For Auto implemented properties");
            //Automated
            B b = new B();
            b.Id = 50;
            Console.WriteLine("Id : {0}",b.Id);
            Console.WriteLine("Name is {0}",b.Name);

            Console.ReadLine();
        }
    }
}
```

Output:

```
C:\Users\91807\source\repos\Day14Project2\Day14Project2\bin\Debug\Day14Project2.exe
For normal properties:
8000


For Auto implemented properties
Id : 50
Name is Ram
```

4.WACP to check if number is Prime or not using break;

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day14Project3
{
    //Author: Rc
    /****Purpose:Print prime numbers using break*****/
    class Prime
    {
        int n,i;
        /// <summary>
        /// This method reads input
        /// </summary>
        public void ReadPrime()
        {
            Console.WriteLine("Enter any num");
            n=Convert.ToInt32(Console.ReadLine());
```

```
        }
        /// <summary>
        /// This method prints the data
        /// </summary>
        public void PrintData()
        {
            for( i = 2; i < n; i++)
            {
                if (n % i == 0)
                    break;
            }
            if (n == i)
                Console.WriteLine("Prime");
            else
                Console.WriteLine("Not a Prime");

        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            //Object for above class
            Prime p = new Prime();
            p.ReadPrime();
            p.PrintData();

            Console.ReadLine();
        }
    }
}
```

Output:

```
C:\Users\91807\source\repos\Day14Project3\Day14Project3\bin\Debug\Day14Project3.exe
Enter any num
7
Prime
```

5.Print numbers from 1-30 and skip the numbers which are divisible by 3.

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day14Project4
{
    //Author:Rc
    /*Purpose:Print numbers from 1-30 and skip numbers divided by 3*/
    class Div
    {
        int i;
        /// <summary>
        /// This method is used to Print data
        /// </summary>
        public void Print()
        {
            for(i=1;i<=30;i++)
            {
                if (i % 3 == 0)
                    continue;
                Console.WriteLine(i);
            }
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            //Object creation for above class
            Div div = new Div();
            div.Print();

            Console.ReadLine();
        }
    }
}
```

Output:

```
C:\Users\91807\source\repos\Day14Project4\Day14Project4\bin\Debug\Day14Project4.exe
1
2
4
5
7
8
10
11
13
14
16
17
19
20
22
23
25
26
28
29
```

6.Find the first number after 1000 which is divisible by 97.

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day14Project5
{
    //Author:Rc
    /*Purpose :find first number after 1000 which is divided by 97*/
    class Find
    {
        int i;
        /// <summary>
        /// This method is used for printing data
        /// </summary>
        public void Print()
        {
            for(i=1000;i<=1097;i++)
            {
                if (i % 97 == 0)
                    break;
            }
            Console.WriteLine(i);
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            //Object creation
            Find find = new Find();
            find.Print();

            Console.ReadLine();
        }
    }
}
```

Output:

```
C:\Users\91807\source\repos\Day14Project5\Day14Project5\bin\Debug\Day14Project5.exe
1067
```