NB Healthcare Technologies Pvt Ltd

Day 14 Morning Assignment (10 – Feb- 2022)

By

Vamsi Krishna Mandapati

1. Research and write what is the use of sealed class. WACP to illustrate sealed class.

Sealed Class: A sealed class is exactly same as normal class, it also can have variables, methods, properties, static methods, etc every thing is there which are present in normal class.

- The only difference is that, the sealed class cannot be used as a parent (or) base class for other classes.
- Sealed class is used **to stop a class to be inherited**. You cannot derive or extend any class from it.
- Classes can be declared as sealed by putting the keyword sealed before the class definition.
- **For example**:

```
 public sealed class D
{
  // Class members here.
}
```

- A sealed class cannot be used as a base class. For this reason, it cannot also be an abstract class.
- Sealed classes prevent derivation. Because they can never be used as a base class, some run-time optimizations can make calling sealed class members slightly faster.

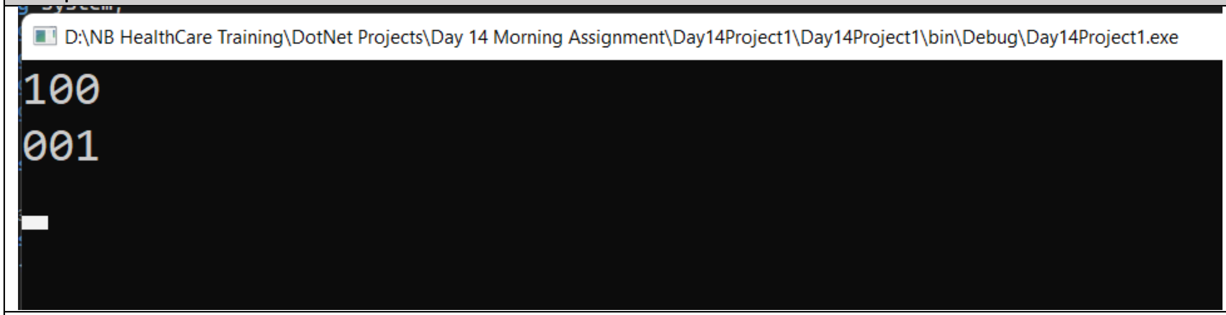
In The Below Code, We print the result with sealed class and using sealed class object but we did not inherited any class from the sealed class (or) base/parent class. We got the output.

Code:

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

namespace Day14Project1
```

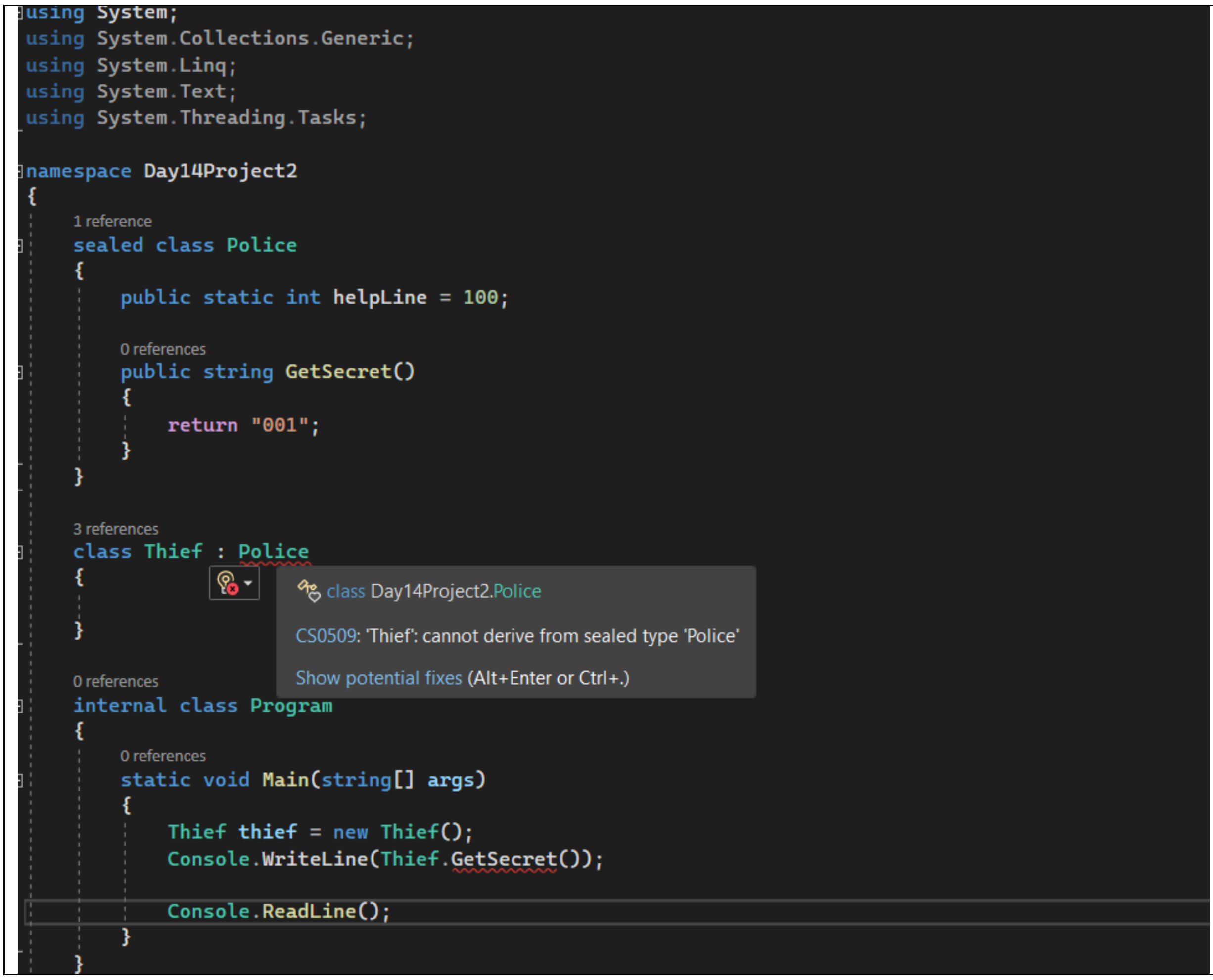
```
{
    sealed class Police
    {
        public static int helpLine = 100;

        public string GetSecret()
        {
            return "001";
        }
    }

    internal class Program
    {
        static void Main(string[] args)
        {
            Police p = new Police();
            Console.WriteLine(Police.helpLine);
            Console.WriteLine(p.GetSecret());

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

Output:



```
D:\NB HealthCare Training\DotNet Projects\Day 14 Morning Assignment\Day14Project1\Day14Project1\bin\Debug\Day14Project1.exe
100
001
```

In The Below Code, We print the result with Theif(derived) class and using derived class object. Here we inherited sealed class (or) base/parent class. We got the Compilation Error..



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

namespace Day14Project2
{
    1 reference
    sealed class Police
    {
        public static int helpLine = 100;

        0 references
        public string GetSecret()
        {
            return "001";
        }
    }

    3 references
    class Thief : Police
    {

    }

    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            Thief thief = new Thief();
            Console.WriteLine(Thief.GetSecret());

            Console.ReadLine();
        }
    }
```

class Day14Project2.Police

CS0509: 'Thief': cannot derive from sealed type 'Police'

Show potential fixes (Alt+Enter or Ctrl+.)

2. Research and write what is the difference between normal properties and auto-implemented properties.
WACP to illustrate normal properties
WACP to illustrate auto-implemented properties

Normal Properties	Auto-Implemented Properties
Properties are mainly introduced to access private variables using set and get accessors.	• Having only set, we can set/write the value and • Having only get, we can get/read the value.
1.Normal Properties access private variables (or) which deals with the other variables 2.Normal properties pointing to private variables	1.Auto-Implemented properties does not Deals with other variables or private variables. 2.Auto-implemented properties are not pointing to private variables

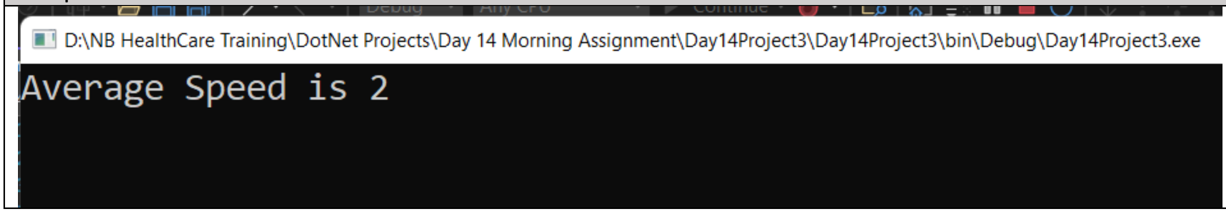
3. Normal Properties will have both set and get accessors	3.Auto-implemented properties also will have both set and get accessors, 90% projects of auto-implemented properties have both set and get accessors. Here set is optional , get is mandatory, if we remove the get then we will get an error.
4.Normal Properties are accessible by using private variables.	4. in auto-implemented properties, Get is accessible and we can run without any private variable.

WACP to illustrate normal properties
WACP to illustrate auto-implemented properties

Code:

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

namespace Day14Project3
{
    internal class Program
    {
        class AverageSpeed
        {
            private int time;
            private int distance;

            public int Distance
            {
                set
                {
                    distance = value;    //Normal Properties
                }
            }

            public int Time
            {
                set
                {
                    time = value;                //Normal Properties
                }
            }

            public int Speed
            {
                get
                {
                    return distance / time;                //Auto-implemented
Properties
                }
            }
        }
        static void Main(string[] args)
        {
            AverageSpeed avg = new AverageSpeed();
            avg.Distance = 70;
            avg.Time = 30;
            Console.WriteLine($"Average Speed is { avg.Speed}");

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

Output:



```
D:\NB HealthCare Training\DotNet Projects\Day 14 Morning Assignment\Day14Project3\Day14Project3\bin\Debug\Day14Project3.exe
Average Speed is 2
```

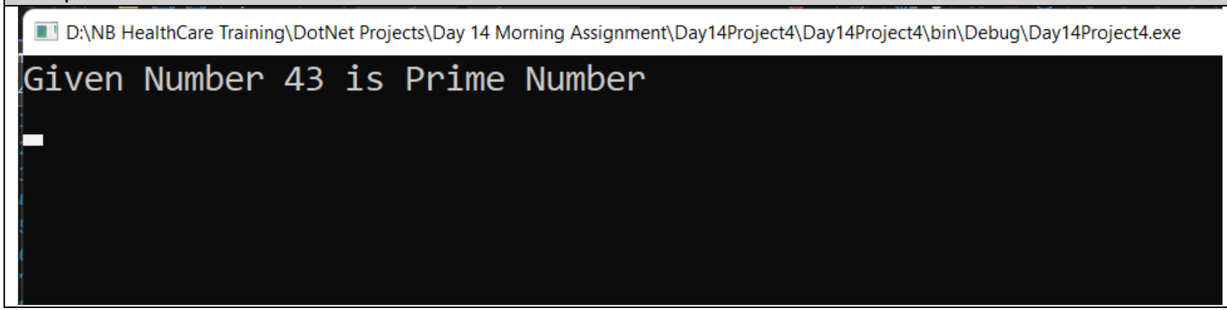
4. WACP to check if the number is prime or not using logic discussed in the class
HINT : use break;

Code:

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

namespace Day14Project4
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int n = 43, i;
            for(i=2;i<n;i++)
            {
                if(n % i == 0)
                {
                    break;
                }
            }
            if(i == n)
            {
```

```
            Console.WriteLine($"Given Number {n} is Prime Number");
        }
        else
        {
            Console.WriteLine($"Given Number {n} is Not a Prime Number");
        }
        Console.ReadLine();
    }
  }
}
```

Output:



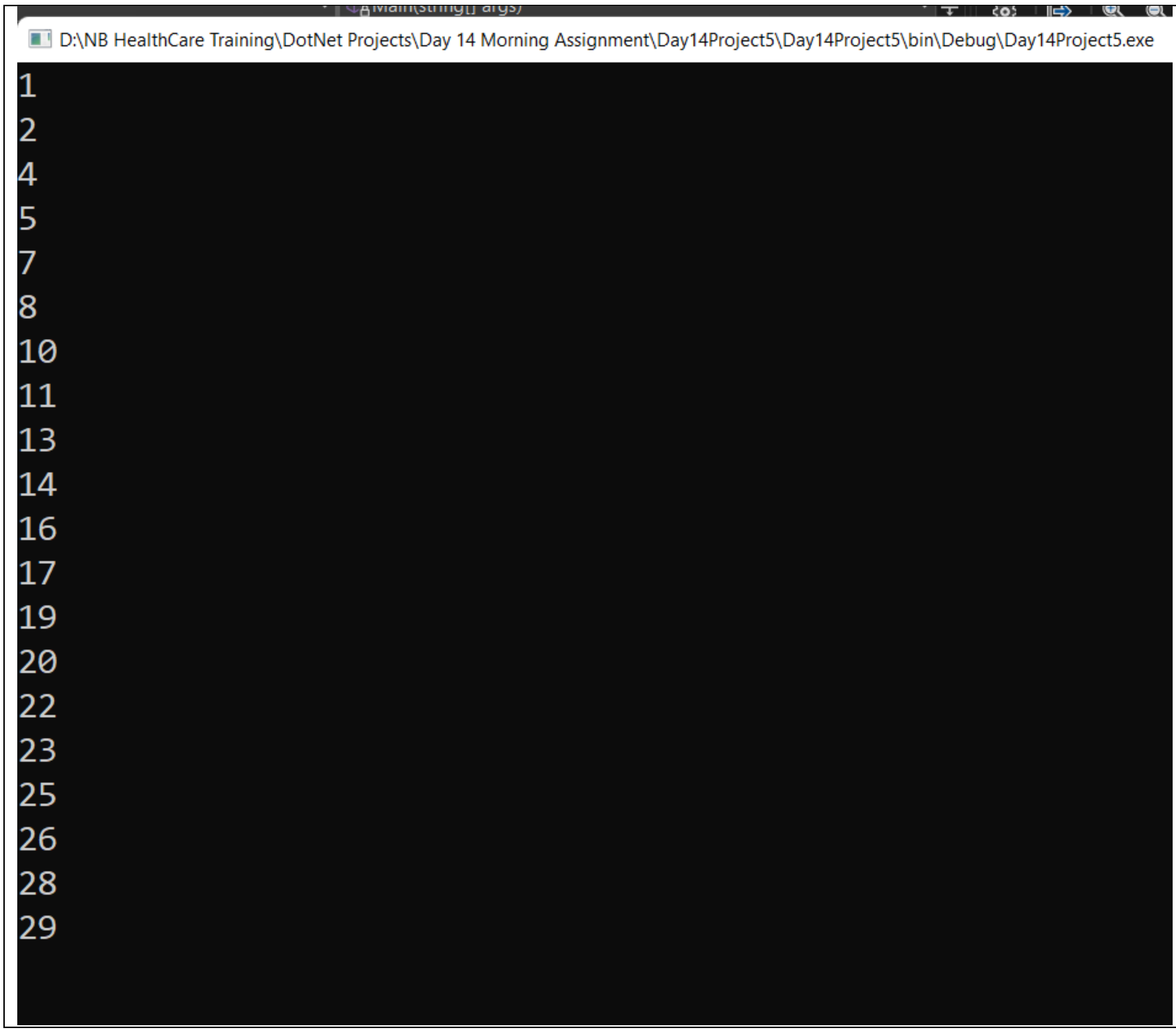
```
D:\NB HealthCare Training\DotNet Projects\Day 14 Morning Assignment\Day14Project4\Day14Project4\bin\Debug\Day14Project4.exe
Given Number 43 is Prime Number
```

5. print numbers from 1 to 30 and skip the numbers divisible by 3
HINT : use continue;

Code:

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

namespace Day14Project5
{
    internal class Program
    {
        static void Main(string[] args)
        {
            for(int i = 1; i <=30; i++)
            {
                if(i%3 == 0)
                {
                    continue;
                }
                Console.WriteLine(i);
            }
            Console.ReadLine();
        }
    }
}
```

Output:


D:\NB HealthCare Training\DotNet Projects\Day 14 Morning Assignment\Day14Project5\Day14Project5\bin\Debug\Day14Project5.exe

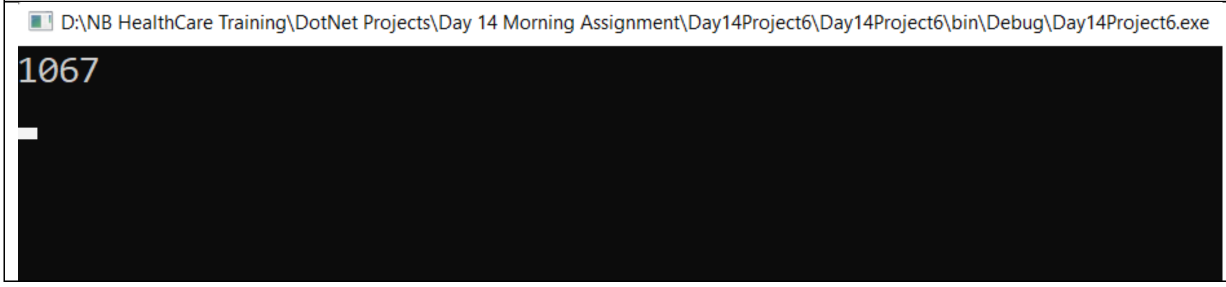
```
1
2
4
5
7
8
10
11
13
14
16
17
19
20
22
23
25
26
28
29
```

6. Find the first number after 1000 which is divisible by 97.
HINT : use for loop and break

Code:

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

namespace Day14Project6
{
    internal class Program
    {
        static void Main(string[] args)
        {
            for(int i = 1000; i <= 1097; i++)
            {
                if(i % 97 == 0)
                {
                    Console.WriteLine(i);
                    break;
                }
            }
            Console.ReadLine();
        }
    }
}
```

Output:



```
D:\NB HealthCare Training\DotNet Projects\Day 14 Morning Assignment\Day14Project6\Day14Project6\bin\Debug\Day14Project6.exe
1067
```

3. Research and fix the below issue:

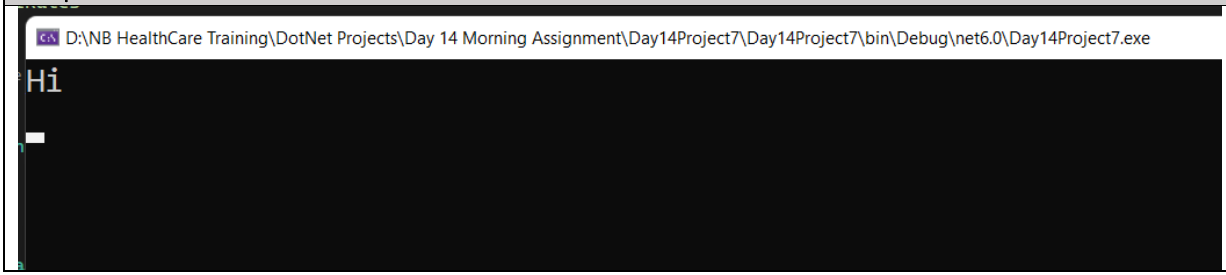
```
interface IRules
  {
      int Age { get; set; }

      int add(int a, int b);

      public void PrintHi()
      {
         Console.WriteLine("Hi");
      }
  }
```

Code:

```
//{ it is done by console app .net core}

using System;

interface IRules
{

    public void PrintHi()
    {
        Console.WriteLine("Hi");
    }
}

class MyClass : IRules
{

}
```

```
class Program
{
    public static void Main(string[] args)
    {
        IRules obj = new MyClass();

        obj.PrintHi();

        Console.ReadLine();
    }
}
```

Output:



```
D:\NB HealthCare Training\DotNet Projects\Day 14 Morning Assignment\Day14Project7\Day14Project7\bin\Debug\net6.0\Day14Project7.exe
Hi
```