Day 14 - Assignment

By **Manoj Karnatapu** - **NBHealthCareTechnologies**

Assignment 1

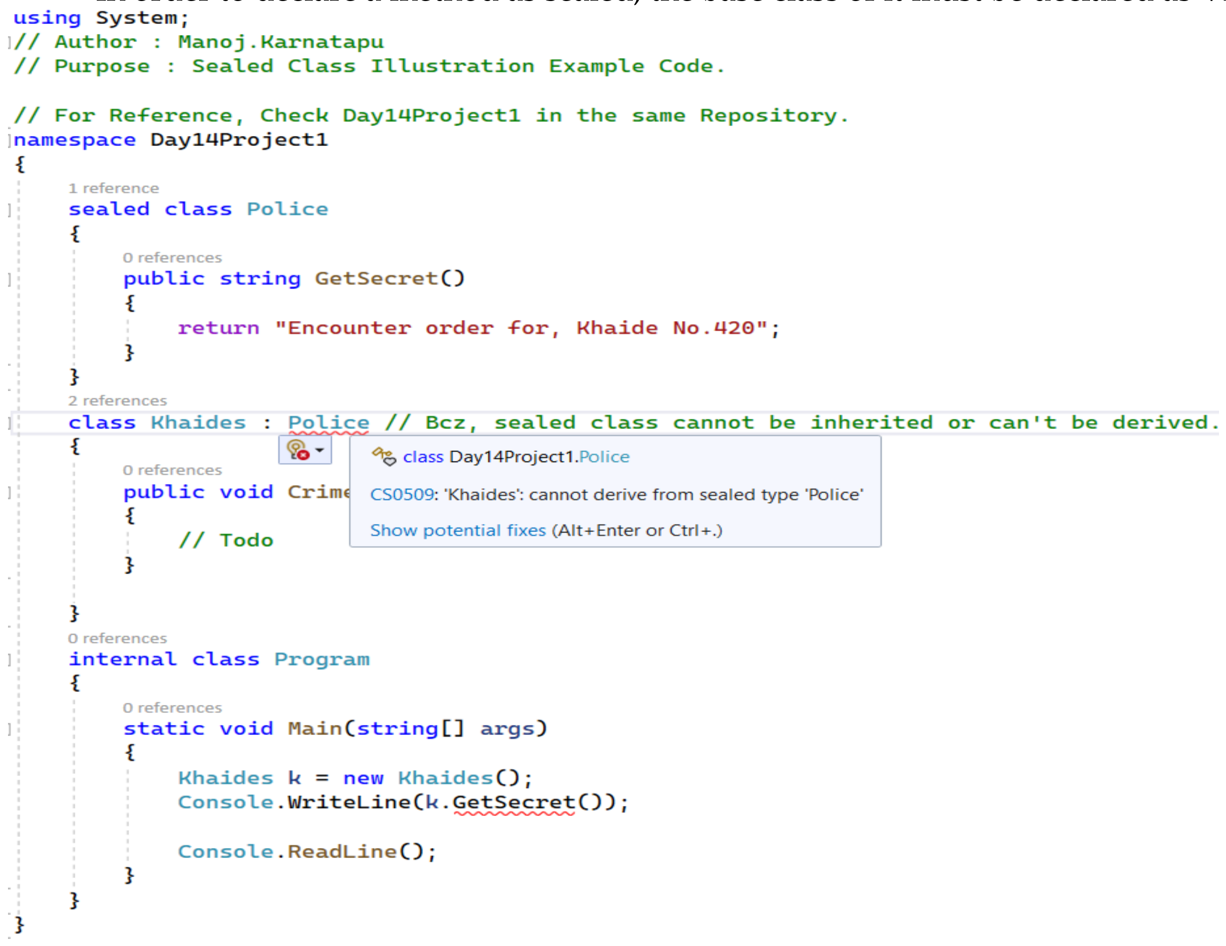
Research & write, what is the use of Sealed class with sample code.

Answer

- Sealed class is used to stop a class to be inherited. You cannot derive or extend any class from it.

------------ OR ------------

- Sealed classes are primarily used to prevent derivation. Because they can never be used as a base class.
- Sealed method is implemented so that no other class can overthrow it and implement its own method.
- some run-time optimizations can make calling sealed class members slightly faster.
- In order to declare a method as sealed, the base class of it must be declared as Virtual.



```
using System;
// Author : Manoj.Karnatapu
// Purpose : Sealed Class Illustration Example Code.

// For Reference, Check Day14Project1 in the same Repository.
namespace Day14Project1
{
    1 reference
    sealed class Police
    {
        0 references
        public string GetSecret()
        {
            return "Encounter order for, Khaide No.420";
        }
    }
    2 references
    class Khaides : Police // Bcz, sealed class cannot be inherited or can't be derived.
    {
        0 references
        public void Crime
        {
            // Todo
        }

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

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

```
class Day14Project1.Police
CS0509: 'Khaides': cannot derive from sealed type 'Police'
Show potential fixes (Alt+Enter or Ctrl+.)
```

Assignment 2

Research & write, what is the difference between Normal properties & Auto-implemented properties with sample codes.

Answer

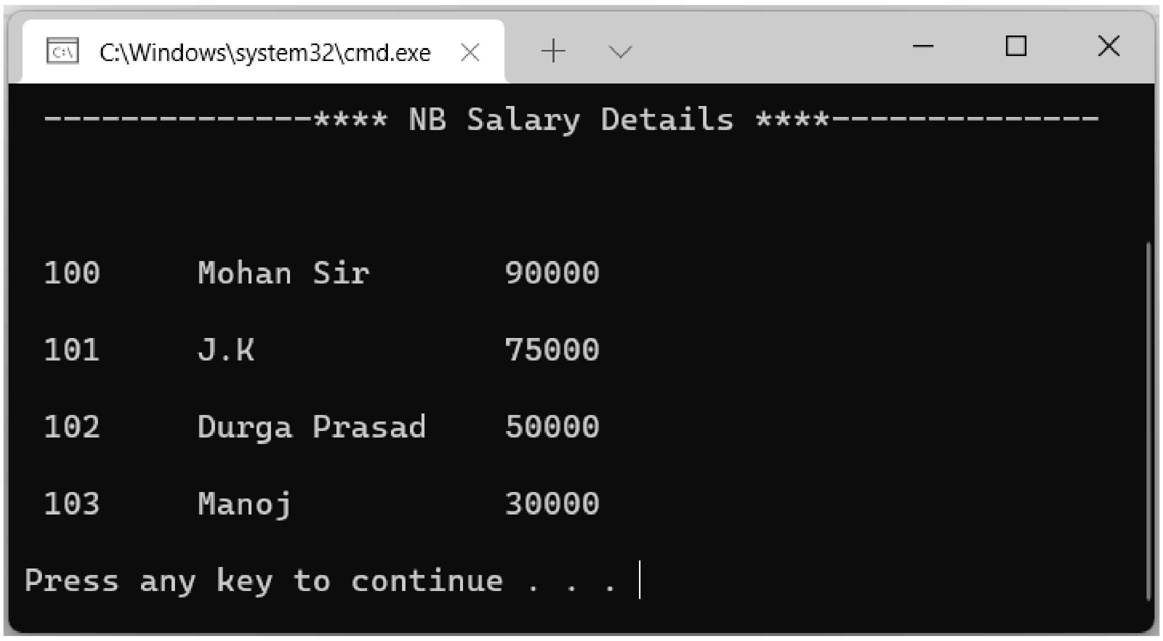
Normal Properties:

Normal Properties, deals and depends on Class level variables. To use normal properties, we have to declare class level variables.

Code : Normal Properties

```
using System;

// Author : Manoj.Karnatapu
// Purpose : C# Code to illustrate Normal properties i.e.,{using only get , only set &
both set and get Methods.}

// for Reference, check Day14Project2 in the same repository.

namespace Day14Project2
{
    class Employee
    {
        private int id;
        private string name;
        private string designation;
        public int Id
        {
            get { return id; }
            set { id = value; }
        }
        public string Name
        {
            get { return name; }
            set { name = value; }
        }
        public string Designation
        {
            // Setting only Write Only Property
            set { designation = value; }
        }
        public int Salary
        {
            get
            {
                if (designation == "M")
                    return 90000;
                else if (designation == "HR")
                    return 50000;
                else if (designation == "TL")
                    return 75000;
                else
                    return 30000;
            }
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("\n --------------**** NB Salary Details ****--------------
\n");
            Employee emp = new Employee();
            emp.Id = 100;
            emp.Name = "Mohan Sir";
            emp.Designation = "M";
            Console.WriteLine($"\n {emp.Id}\t {emp.Name}\t {emp.Salary}");

            Employee emp1 = new Employee();
            emp1.Id = 101;
            emp1.Name = "J.K";
            emp1.Designation = "TL";
            Console.WriteLine($"\n {emp1.Id}\t {emp1.Name}\t\t {emp1.Salary}");

            Employee emp2 = new Employee();
            emp2.Id = 102;
            emp2.Name = "Durga Prasad";
            emp2.Designation = "HR";
            Console.WriteLine($"\n {emp2.Id}\t {emp2.Name}\t {emp2.Salary}");

            Employee emp3 = new Employee();
            emp3.Id = 103;
            emp3.Name = "Manoj";
            emp3.Designation = "S";
            Console.WriteLine($"\n {emp3.Id}\t {emp3.Name}\t\t {emp3.Salary}");
            Console.ReadLine();
        }
    }
}
```

Output



```
 --------------**** NB Salary Details ****--------------



 100     Mohan Sir       90000

 101     J.K             75000

 102     Durga Prasad    50000

 103     Manoj           30000

Press any key to continue . . .
```

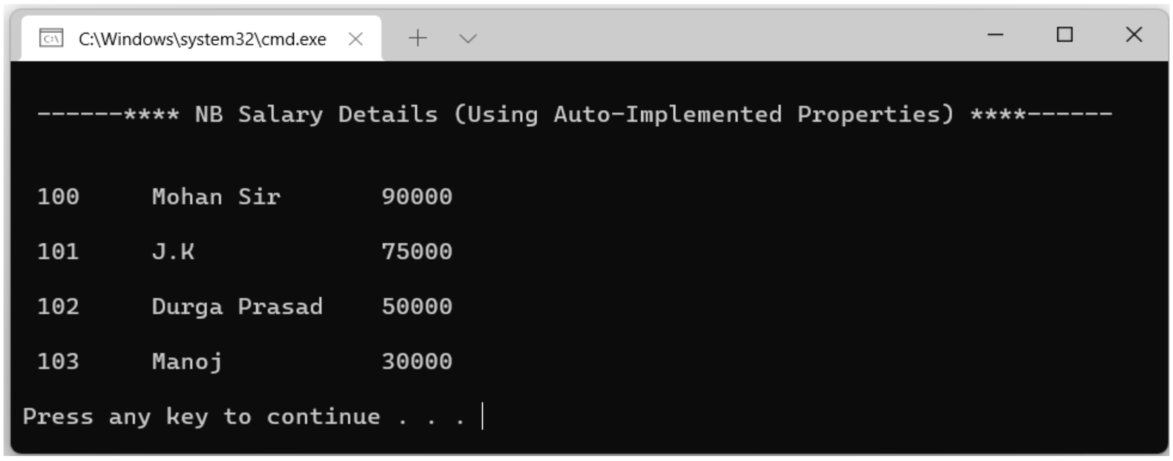
Auto Implemented Properties:

The Properties, doesn't depend or don't deal with class level variables are called Auto-Implemented Properties. The C# Compiler By default creates variables which are required.

Code : Auto Implemented Properties

```
using System;

// Author : Manoj.Karnatapu
// Purpose : C# Code to illustrate using Auto-Implemented Properties (No private or
public classlevel Variables).

// for Reference, check Day14Project3 in the same repository.
namespace Day14Project3
{
    class Employee
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Designation { get; set; }

        public int Salary
        {
            get
            {
                if (Designation == "M")
                    return 90000;
                else if (Designation == "HR")
                    return 50000;
                else if (Designation == "TL")
                    return 75000;
                else
                    return 30000;
            }
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("\n ------**** NB Salary Details (Using Auto-Implemented
Properties) ****------\n");
            Employee emp = new Employee();
            emp.Id = 100;
            emp.Name = "Mohan Sir";
            emp.Designation = "M";
            //emp.Salary = (emp.Designation == "M") ? 90000 : 60000;

            Console.WriteLine($"\n {emp.Id}\t {emp.Name}\t {emp.Salary}");

            Employee emp1 = new Employee();
            emp1.Id = 101;
            emp1.Name = "J.K";
            emp1.Designation = "TL";

            Console.WriteLine($"\n {emp1.Id}\t {emp1.Name}\t\t {emp1.Salary}");

            Employee emp2 = new Employee();
            emp2.Id = 102;
            emp2.Name = "Durga Prasad";
            emp2.Designation = "HR";

            Console.WriteLine($"\n {emp2.Id}\t {emp2.Name}\t {emp2.Salary}");

            Employee emp3 = new Employee();
            emp3.Id = 103;
            emp3.Name = "Manoj";
            emp3.Designation = "S";

            Console.WriteLine($"\n {emp3.Id}\t {emp3.Name}\t\t {emp3.Salary}");

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

Output



```
------**** NB Salary Details (Using Auto-Implemented Properties) ****------

 100     Mohan Sir       90000

 101     J.K             75000

 102     Durga Prasad    50000

 103     Manoj           30000

Press any key to continue . . .
```

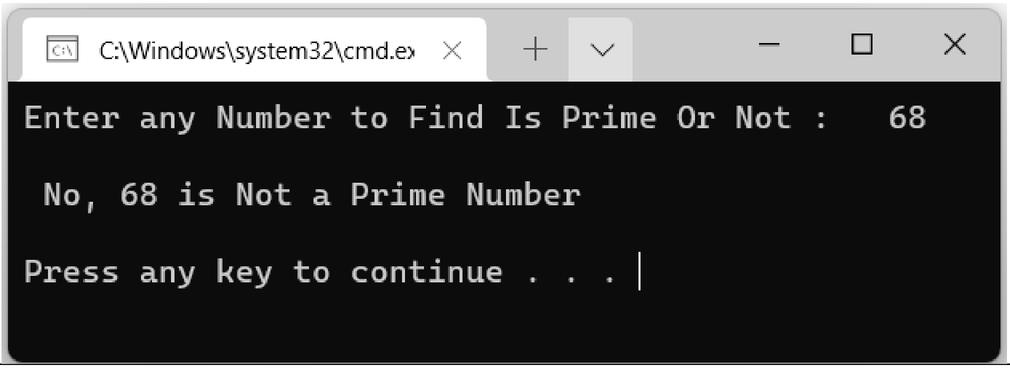
Assignment 4

Write a C# Program to check, if the number is prime or not using break statement.

Code

```
using System;
// Author : Manoj.Karnatapu
// Purpose : Write a C# Code for, Prime Or Not using Break statement.
// for Reference, check Day14Project4 in the same Repository.
namespace Day14Project4
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int n, i;
            Console.Write("Enter any Number to Find Is Prime Or Not :   ");
            n = int.Parse(Console.ReadLine());

            for(i = 2; i < n; i++)
            {
                if (n % i == 0)
                    break;
            }
            if(i == n)
                Console.WriteLine("\n Yes, {0} is a Prime Number", n);
            else
                Console.WriteLine("\n No, {0} is Not a Prime Number",n);
            Console.ReadLine();
        }
    }
}
```

Output



```
Enter any Number to Find Is Prime Or Not :   68

 No, 68 is Not a Prime Number

Press any key to continue . . .
```

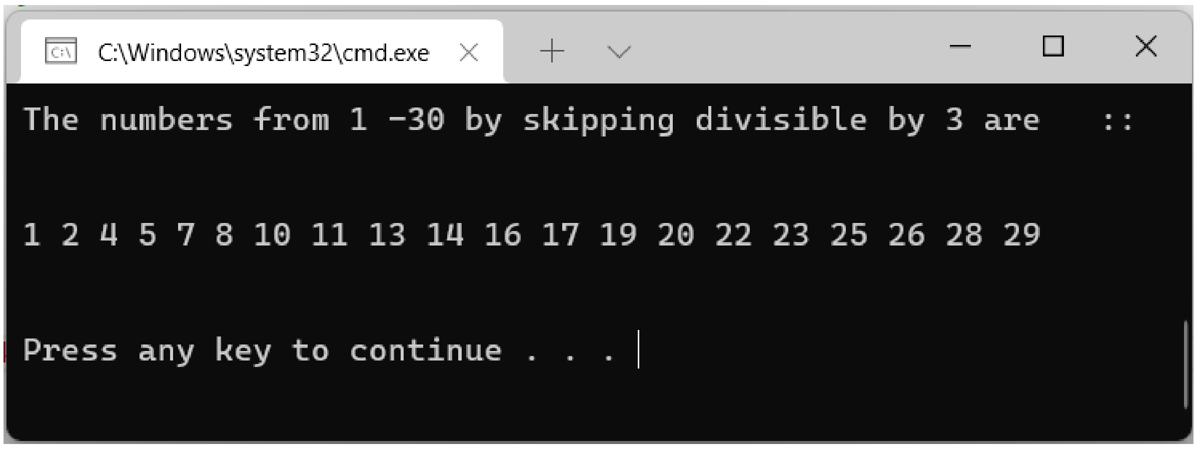
Assignment 5

Write a C# Program to print numbers from 1 to 30, skip the numbers divisible by 3

Code

```
using System;

// Author : Manoj.Karnatapu
// Purpose : Printing Numbers from 1 to 30 by skipping, the numbers divisible by 3.

// for Reference, Check Day14Project5 in the same repository.

namespace Day14Project5
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("The numbers from 1 -30 by skipping divisible by 3 are   ::
\n");
            for (int i = 0;i<=30;i++)
            {
                if (i % 3 == 0)
                    continue;

                Console.Write(i + " ");
            }
            Console.WriteLine("\n");
            Console.ReadLine();
        }
    }
}
```

Output



```
The numbers from 1 -30 by skipping divisible by 3 are   ::


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


Press any key to continue . . .
```

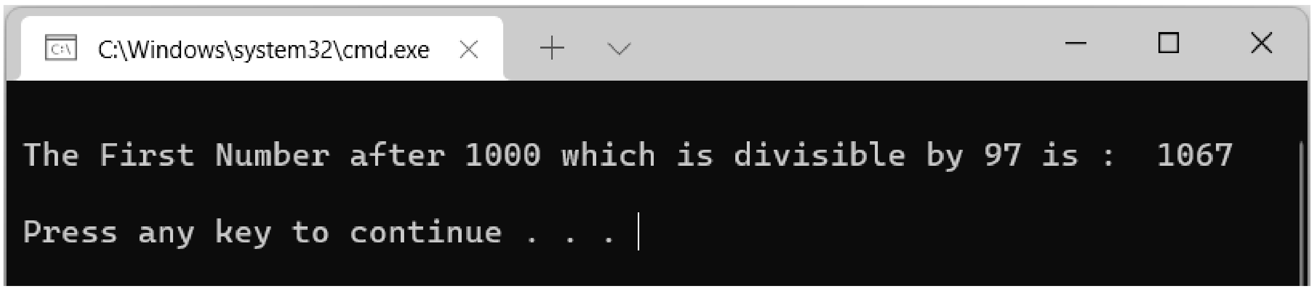
Assignment 6

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

Code

```
using System;
// Author : Manoj.Karnatapu
// Purpose : Find the first number after 1000 which is divisible by 97

// for reference, Check Day14Project6 in the same Repository.
namespace Day14Project6
{
    internal class Program
    {
        static void Main(string[] args)
        {
            for (int i = 1000; i <= 1097; i++)
            {
                if ( i % 97 == 0 )
                {
                    Console.WriteLine("\nThe First Number after 1000 which is divisible by 
97 is :  {0}",i);
                    break;
                }
            }

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

Output



```

The First Number after 1000 which is divisible by 97 is :  1067

Press any key to continue . . .
```

Assignment 3

Research & find , how to declare normal interface & assign a normal method.

Answer