

DAY 14 ASSIGNMENT

BHANU PRAKASH REDDY

10-02-2022



An IT division of NationsBenefits®, LLC. USA



CREATED BY BHANU

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

- Sealed class is used to stop a class to be inherited.
- Sealed classes are primarily used to prevent derivation, because they can never be used as a base class.
- If a class is derived from a sealed class, compiler throws an error.

Code:

```
//Author: Bhanu Prakash Reddy
//WACP using the Sealed Class
sealed class Player
{
    private string playername;
    private int jerseynumber;

    /// <summary>
    /// Reading Player Information
    /// </summary>
    public void ReadPlayer()
    {
        Console.WriteLine("Enter Player Name : ");
        playername = Console.ReadLine();

        jerseynumber = Convert.ToInt32(Console.ReadLine());
    }

    /// <summary>
    /// Printing Player information
    /// </summary>
    public void PrintPlayer()
    {
        Console.WriteLine($"Name of the Player is {playername}");
        Console.WriteLine($"Jersey Number of the Player is {jerseynumber} ");
    }
}
internal class Program
{
    static void Main(string[] args)
    {
        Player p = new Player();
        p.ReadPlayer();
        p.PrintPlayer();

        Console.ReadLine();
    }
}
```

Output:

D:\assignments\Sealed class\Sealed class\bin\Debug\Sealed class.exe

```
Enter Player Name :  
Lebron James  
Enter Jersey Number :  
6  
Name of the Player is Lebron James  
Jersey Number of the Player is 6
```

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.

- **Properties:** Properties are the special type of class members 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 help us to promote the flexibility and safety of methods.
- **Auto Implemented Properties:** 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. You can't declare auto implemented properties in interfaces.

Code:

```
// Author: Bhanu Prakash Reddy
//WACP using property and auto implemented property
class Speed
{
    private int distance;
    private int time;

    //Normal Properties
    public int Distance
    {
        set
        {
            distance = value;
        }
    }
    public int Time
    {
        set
        {
            time = value;
        }
    }
    public int GetSpeed
    {
        get
        {
            return distance / time;
        }
    }
    //Auto implemented Property
    public int Velocity
    {
        get
        {
            return distance / time;
        }
    }
}
internal class Program
{
    static void Main(string[] args)
    {
        Speed s = new Speed();
        s.Distance = 90;
        s.Time = 10;
        Console.WriteLine($"Speed is {s.GetSpeed}");

        Console.ReadLine();
    }
}
```

D:\assignments\Property and Auto implemented Property\Property and Auto implemented Property\bin\Debug\Property and Auto implemented Property.exe

Speed is 9

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");
    }
}
```

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

Code:


```
//Author: Bhanu Prakash Reddy
//WACP for prime number using break;
internal class Program
{
    static void Main(string[] args)
    {
        int n, i;

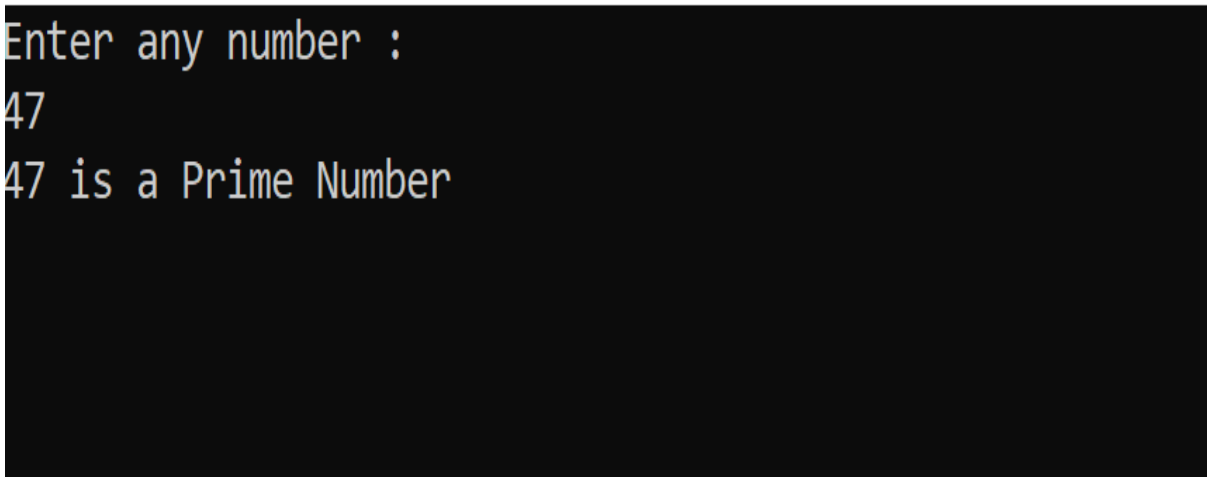
        Console.WriteLine("Enter any number : ");
        n = Convert.ToInt32(Console.ReadLine());

        for (i = 2; i <= n; i++)
        {
            if (n % i == 0)
                break;
        }
        if (i == n)
            Console.WriteLine($"{n} is a Prime Number");
        else
            Console.WriteLine($"{n} is not a prime Number");

        Console.ReadLine();
    }
}
```

Output:

 D:\assignments\Prime Number using Break\Prime Number using Break\bin\Debug\Prime Number using Break.exe



```
Enter any number :
47
47 is a Prime Number
```

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

Code:

```
//Author: Bhanu Prakash reddy
//WACP using continue to print numbers
internal class Program
{
    static void Main(string[] args)
    {
        int i;
        Console.WriteLine("Skipping numbers divisible by 3 : ");

        for(i=1;i<=30;i++)
        {
            if (i % 3 == 0)
                continue;
            Console.Write($"{i} ");
        }
        Console.ReadLine();
    }
}
```

Output:

D:\assignments\Using Continue To print 1 to 30\Using Continue To print 1 to 30\bin\Debug\Using Continue To print 1 to 30.exe

Skipping numbers divisible by 3 :

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:

```
//Author: Bhanu Prakash Reddy
//WACP for print first number after 1000 divided by 97
internal class Program
{
    static void Main(string[] args)
    {
        int i;

        for (i = 1000; i<=1097; i++)
        {
            if (i % 97 == 0)
                break;
        }
        Console.WriteLine($"First number after 1000 divided by 97 is {i}");
        Console.ReadLine();
    }
}
```

Output:

 D:\assignments\First number after 1000 divided by 97\First number after 1000 divided by 97\bin\Debug\First number after 1000 divided by 97.exe



```
First number after 1000 divided by 97 is 1067
```