3. Procedural Programming

Well procedural programming is a programming paradigm based on procedure calls. By procedure - mean function method routine subroutine. These are all the same thing.

So far all the code you have written in your exercises was in the main method but that's not how we build real world applications because a real world application includes quite a number of functionalitie and writing all that code inside the main method is going to be a disaster.

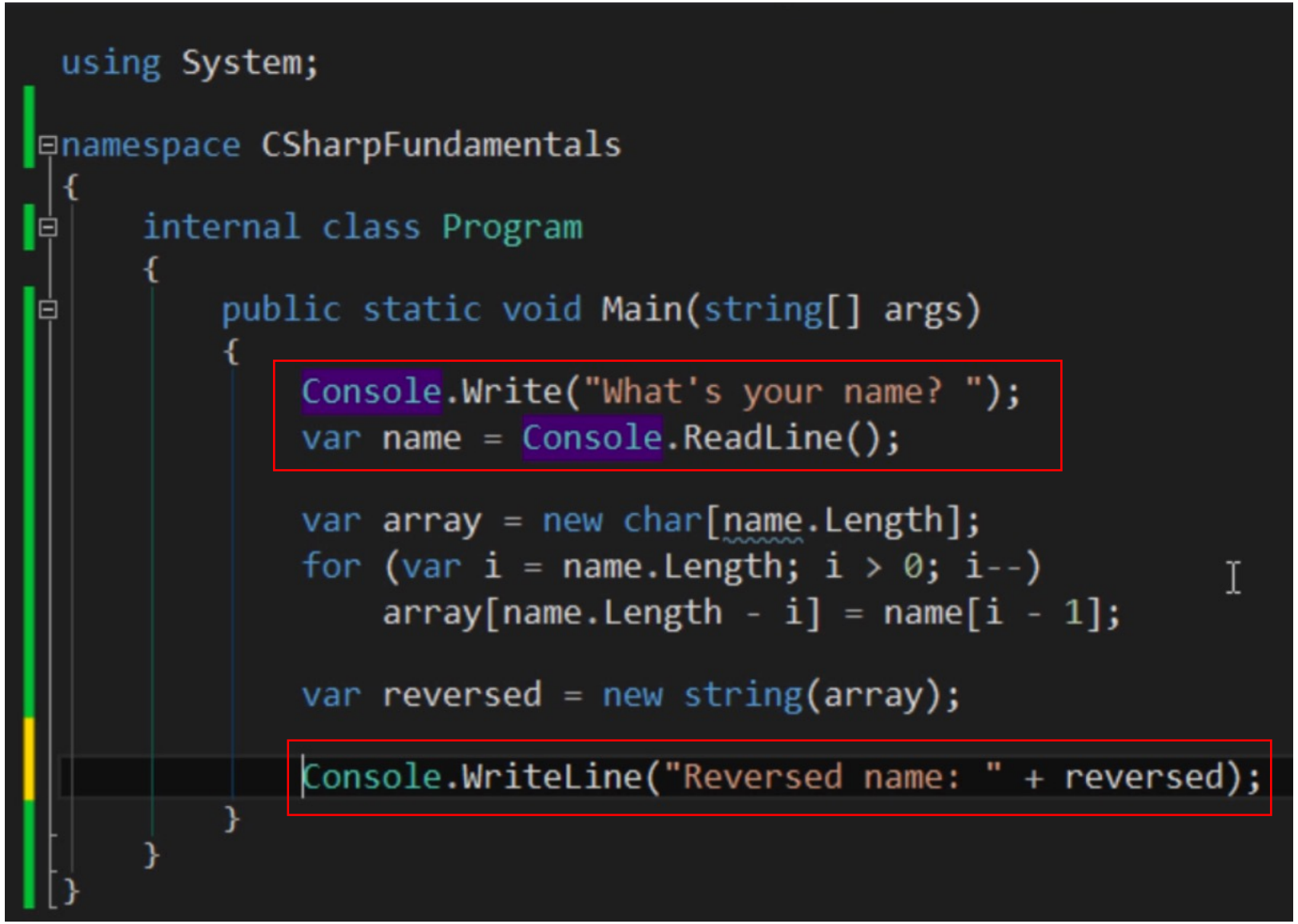
So we need to break down our code into a number of methods or functions each responsible for one thing And this is what we call procedural programming.

Object-oriented Programming	A programming paradigm based on objects.

We also have object oriented programming which is another programming paradigm which is based on objects
So instead of thinking in methods and functions we think in objects

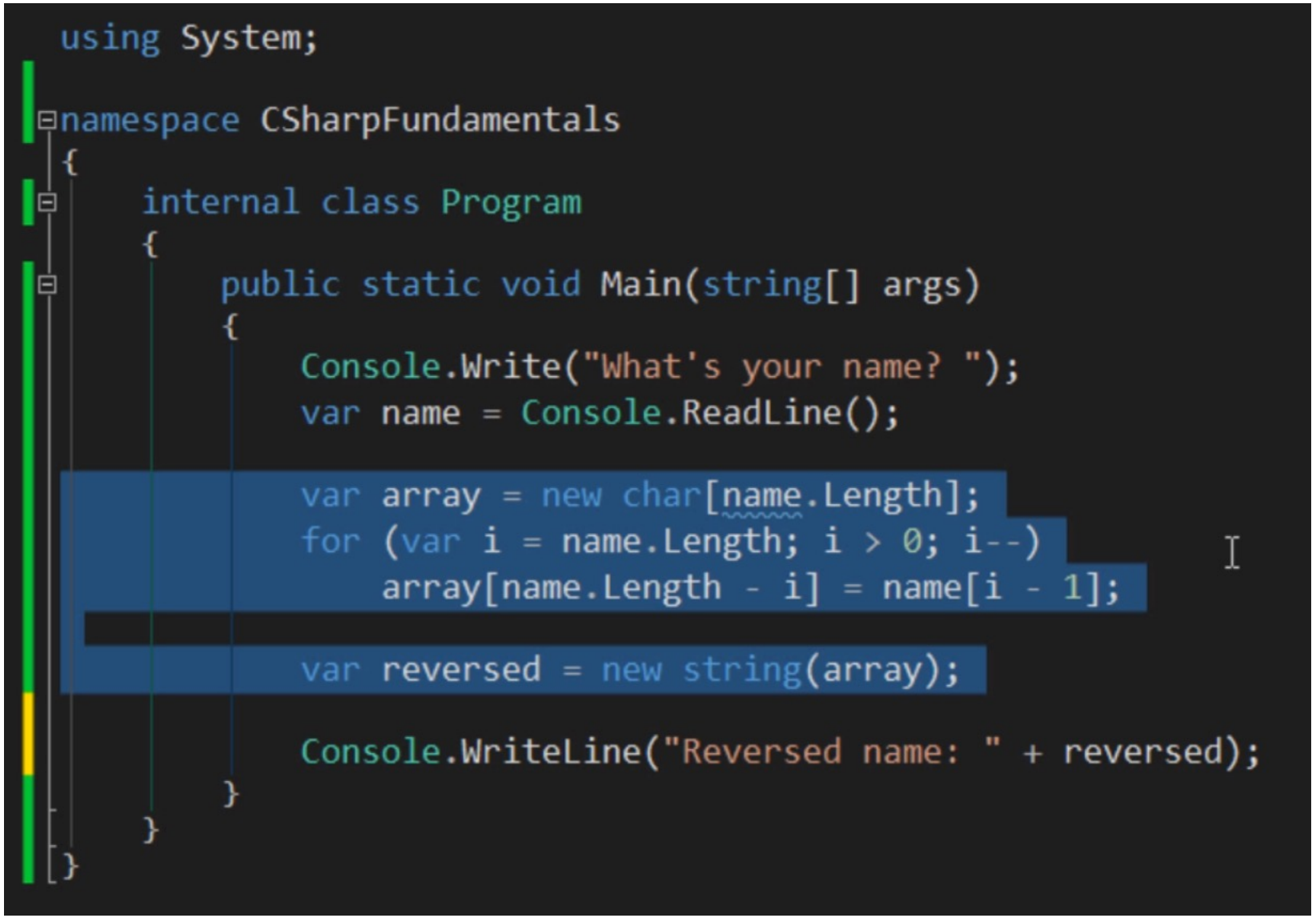
```
using System;

namespace CSharpFundamentals
{
    internal class Program
    {
        public static void Main(string[] args)
        {
            Console.Write("What's your name? ");
            var name = Console.ReadLine();

            var array = new char[name.Length];
            for (var i = name.Length; i > 0; i--)
                array[name.Length - i] = name[i - 1];

            var reversed = new string(array);
            Console.WriteLine("Reversed name: " + reversed);
        }
    }
}
```

The first rule of thumb you should always separate the code that works with the console from the code that implements some logic.



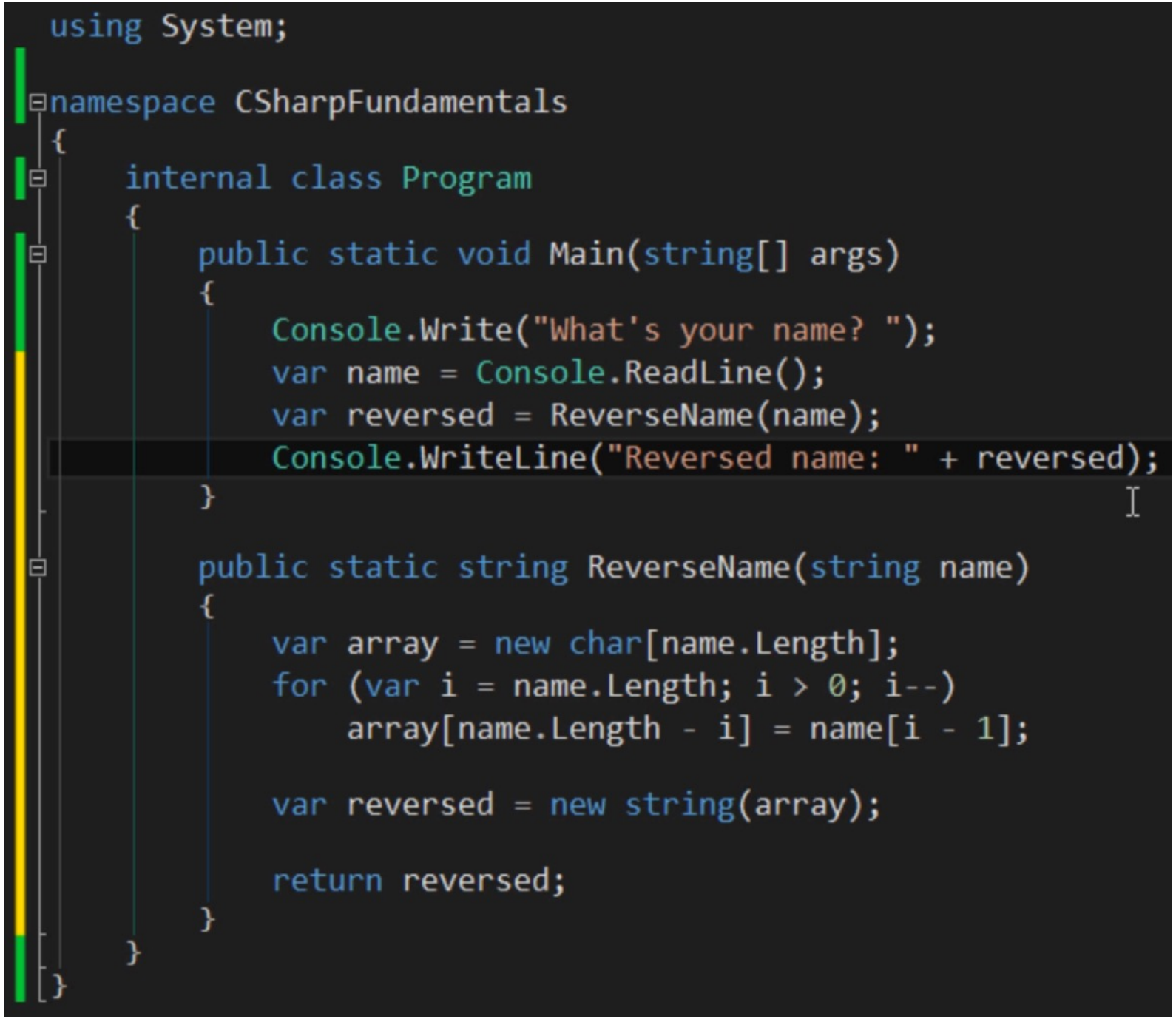
```
using System;

namespace CSharpFundamentals
{
    internal class Program
    {
        public static void Main(string[] args)
        {
            Console.Write("What's your name? ");
            var name = Console.ReadLine();

            var array = new char[name.Length];
            for (var i = name.Length; i > 0; i--)
                array[name.Length - i] = name[i - 1];

            var reversed = new string(array);

            Console.WriteLine("Reversed name: " + reversed);
        }
    }
}
```

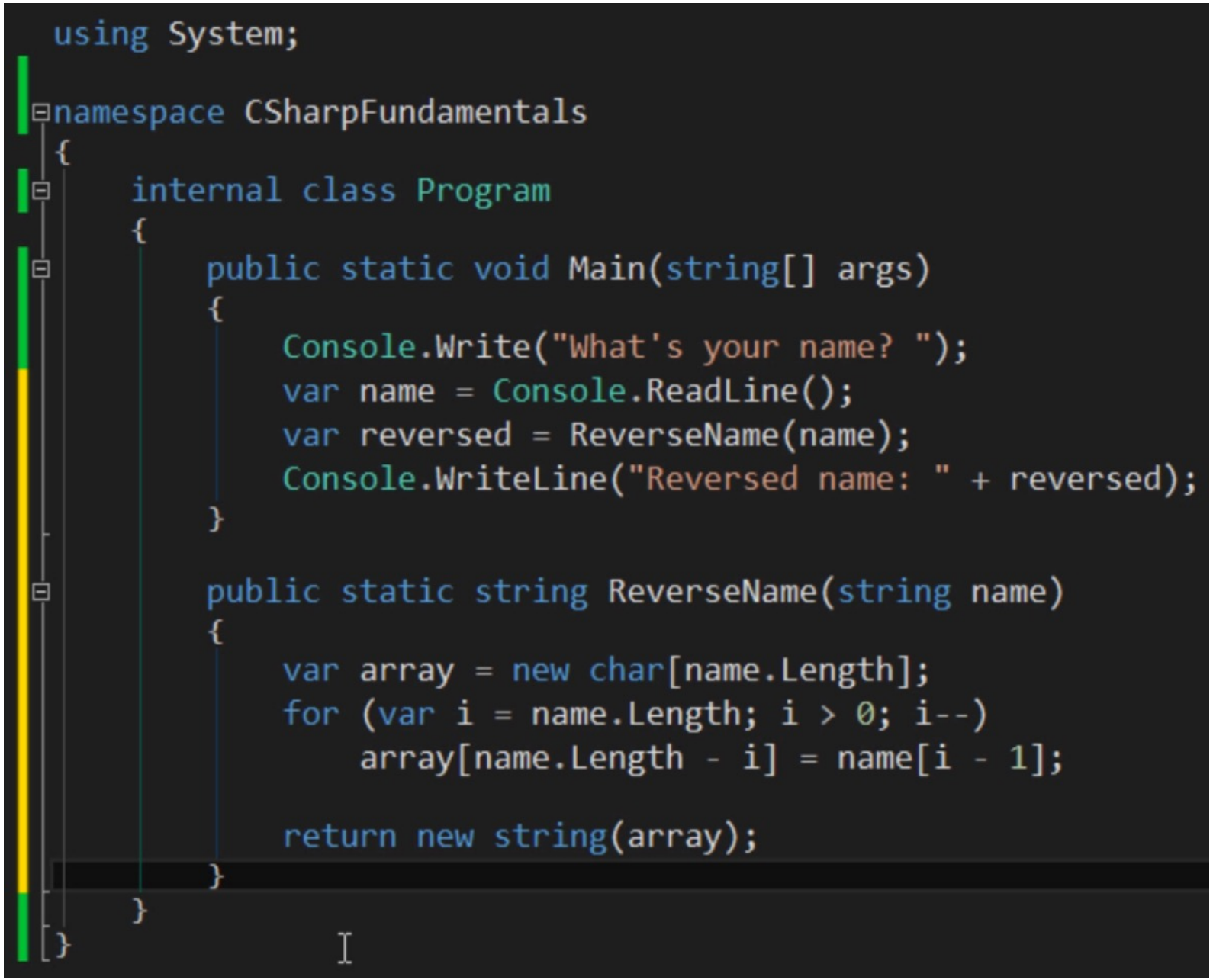
Because in the real world we have different presentation technologies. We have desktop application, web applications, mobile apps and there we don't have console.



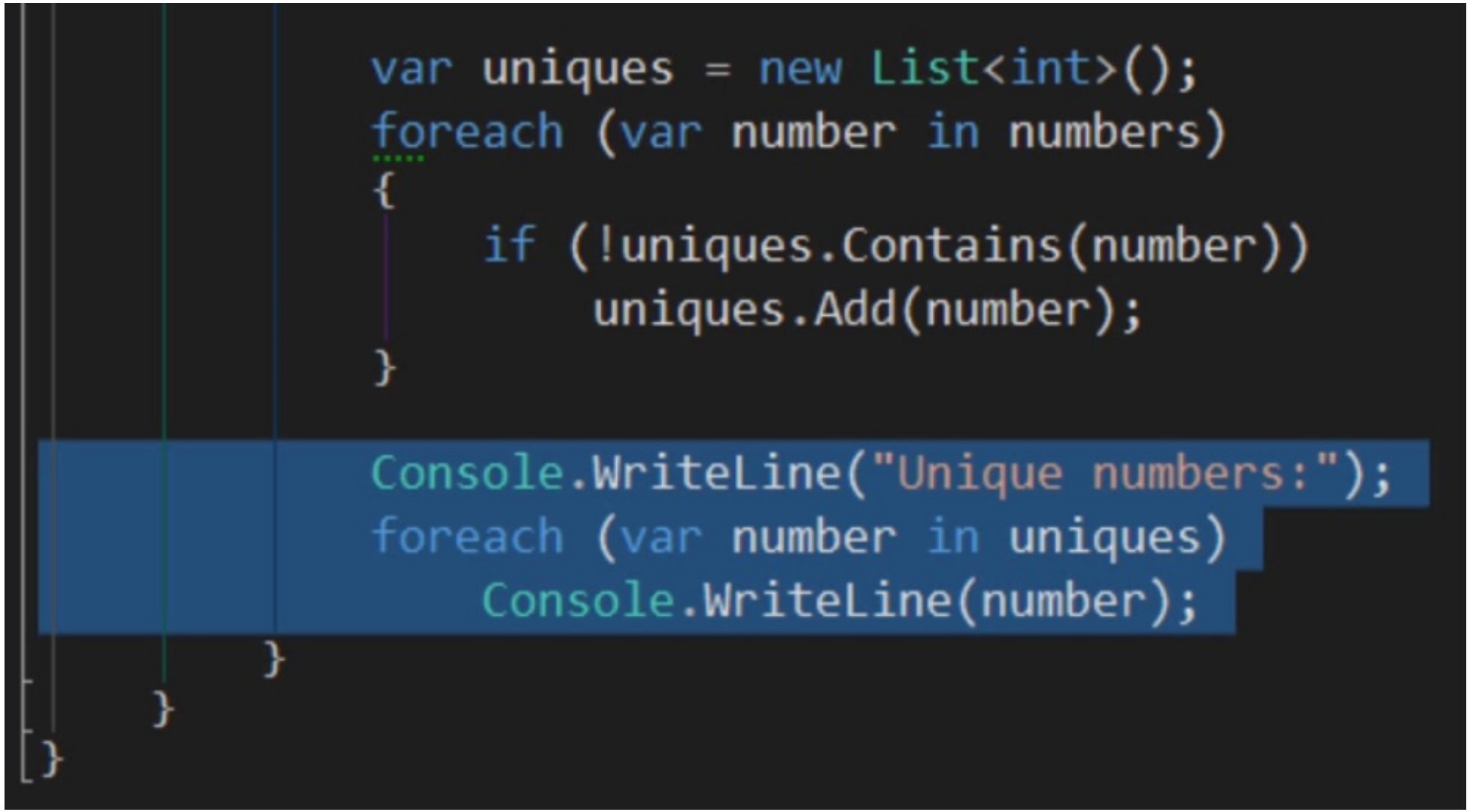
```
using System;

namespace CSharpFundamentals
{
    internal class Program
    {
        public static void Main(string[] args)
        {
            Console.Write("What's your name? ");
            var name = Console.ReadLine();

            var array = new char[name.Length];
            for (var i = name.Length; i > 0; i--)
                array[name.Length - i] = name[i - 1];

            var reversed = new string(array);

            Console.WriteLine("Reversed name: " + reversed);
        }
    }
}
```

So by extracting this logic here that is highlighted in to a separate method we can reuse that method in any application console is only relevant in console applications.

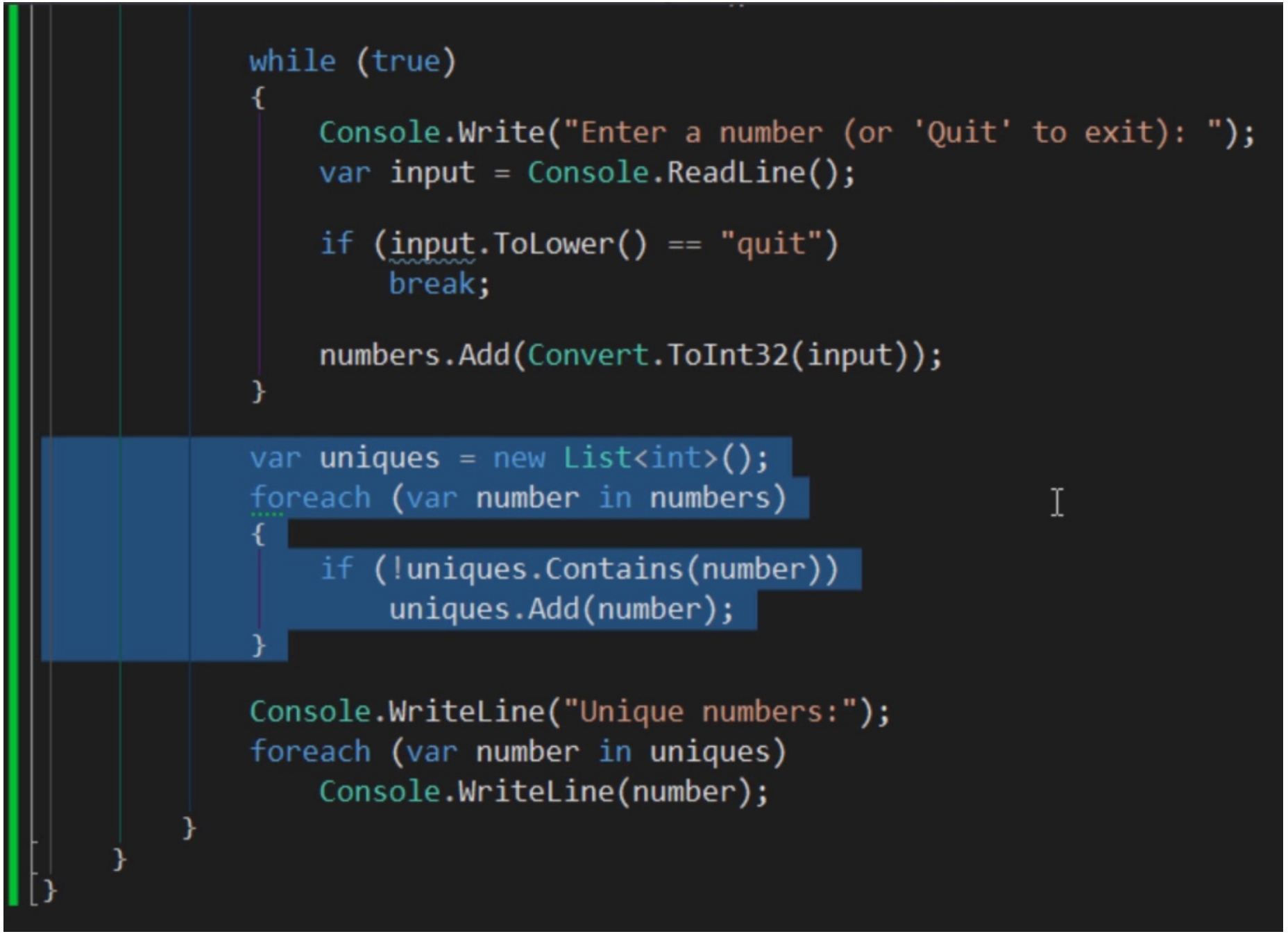

using System;
namespace CSharpFundamentals
{
internal class Program
{
public static void Main(string[] args)
{
Console.Write("What's your name? ");
var name = Console.ReadLine();
var reversed = ReverseName(name);
Console.WriteLine("Reversed name: " + reversed);
}
public static string ReverseName(string name)
{
var array = new char[name.Length];
for (var i = name.Length; i > 0; i--)
array[name.Length - i] = name[i - 1];
var reversed = new string(array);
return reversed;
}
}
}


using System;
namespace CSharpFundamentals
{
internal class Program
{
public static void Main(string[] args)
{
Console.Write("What's your name? ");
var name = Console.ReadLine();
var reversed = ReverseName(name);
Console.WriteLine("Reversed name: " + reversed);
}
public static string ReverseName(string name)
{
var array = new char[name.Length];
for (var i = name.Length; i > 0; i--)
array[name.Length - i] = name[i - 1];
return new string(array);
}
}
}

```
using System;
using System.Collections.Generic;

namespace CSharpFundamentals
{
    internal class Program
    {
        public static void Main(string[] args)
        {
            var numbers = new List<int>();

            while (true)
            {
                Console.Write("Enter a number (or 'Quit' to exit): ");
                var input = Console.ReadLine();

                if (input.ToLower() == "quit")
                    break;

                numbers.Add(Convert.ToInt32(input));
            }
```



```
            var uniques = new List<int>();
            foreach (var number in numbers)
            {
                if (!uniques.Contains(number))
                    uniques.Add(number);
            }

            Console.WriteLine("Unique numbers:");
            foreach (var number in uniques)
                Console.WriteLine(number);
        }
    }
}
```



```
        while (true)
        {
            Console.Write("Enter a number (or 'Quit' to exit): ");
            var input = Console.ReadLine();

            if (input.ToLower() == "quit")
                break;

            numbers.Add(Convert.ToInt32(input));
        }

        var uniques = new List<int>();
        foreach (var number in numbers)
        {
            if (!uniques.Contains(number))
                uniques.Add(number);
        }

        Console.WriteLine("Unique numbers:");
        foreach (var number in uniques)
            Console.WriteLine(number);
    }
}
}
```

```
            Console.Write("Enter a number (or 'Quit' to exit): ");
            var input = Console.ReadLine();

            if (input.ToLower() == "quit")
                break;

            numbers.Add(Convert.ToInt32(input));
        }


        Console.WriteLine("Unique numbers:");
        foreach (var number in uniques)
            Console.WriteLine(number);
    }

    public static List<int> GetUniqueNumbers(List<int> numbers)
    {
        var uniques = new List<int>();
        foreach (var number in numbers)
        {
            if (!uniques.Contains(number))
                uniques.Add(number);
        }

        return uniques;

    }
}
```

```
        while (true)
        {
            Console.Write("Enter a number (or 'Quit' to exit): ");
            var input = Console.ReadLine();

            if (input.ToLower() == "quit")
                break;

            numbers.Add(Convert.ToInt32(input));
        }

        Console.WriteLine("Unique numbers:");
        foreach (var number in GetUniqueNumbers(numbers))
            Console.WriteLine(number);
    }

    public static List<int> GetUniqueNumbers(List<int> numbers)
    {
        var uniques = new List<int>();
        foreach (var number in numbers)
        {
            if (!uniques.Contains(number))
                uniques.Add(number);
        }

        return uniques;
    }
```

Exercises

Go over the exercises in the string section the ones you did just before the lecture and extract methods or functions or procedures from your code.