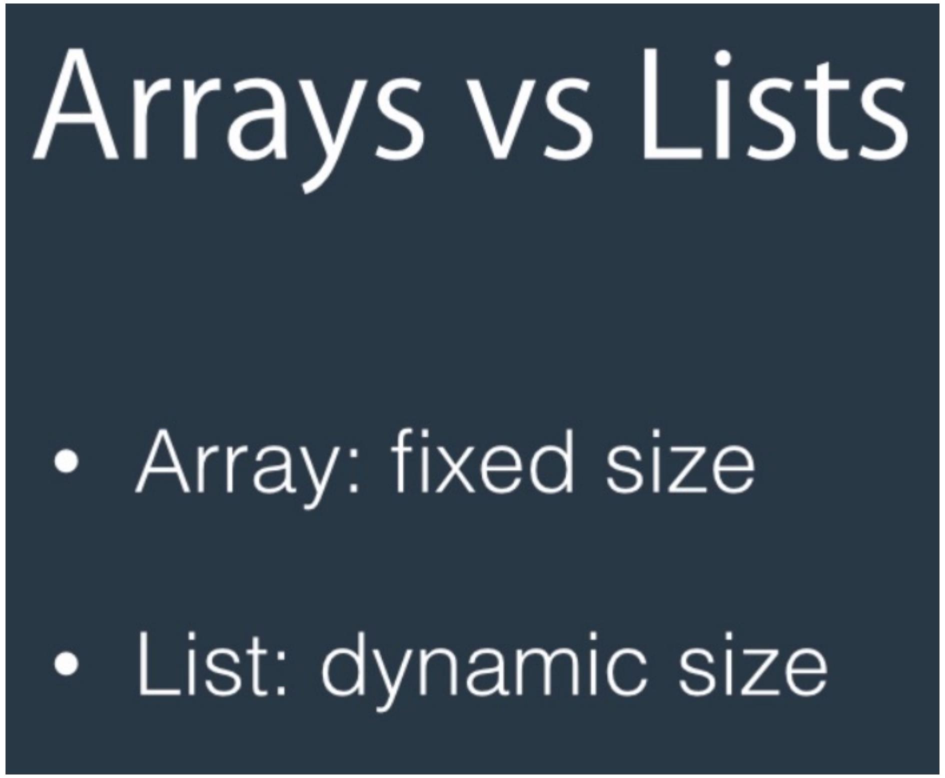
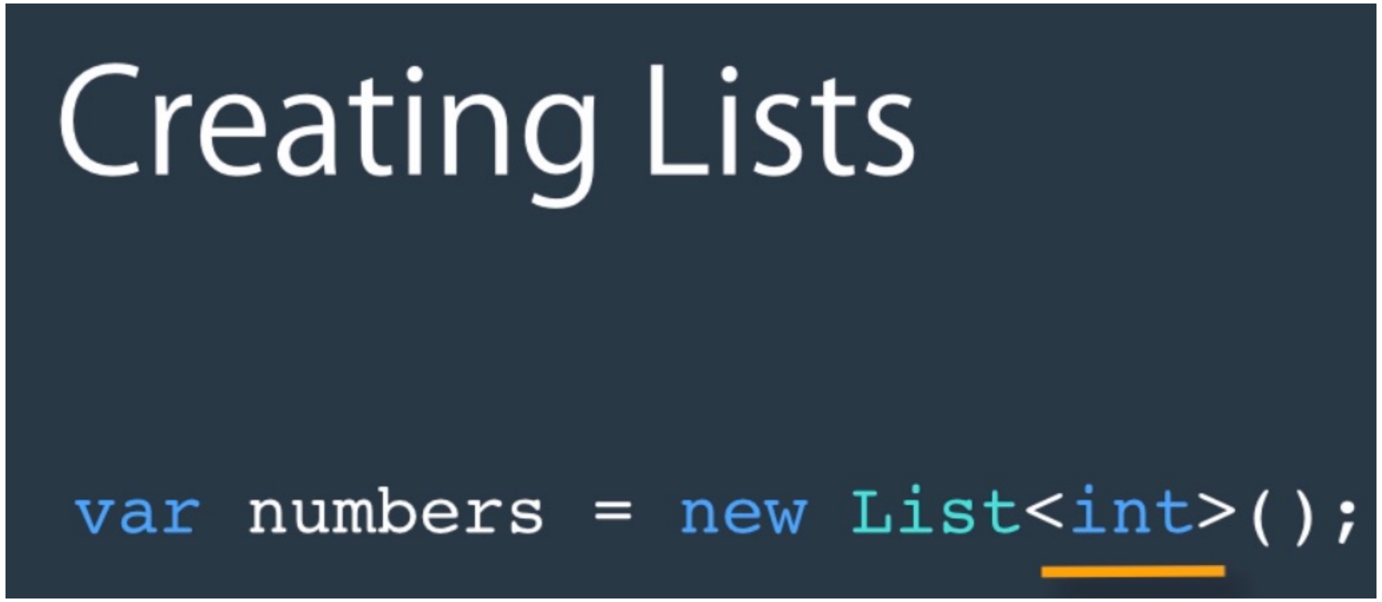
2. Lists



Arrays vs Lists

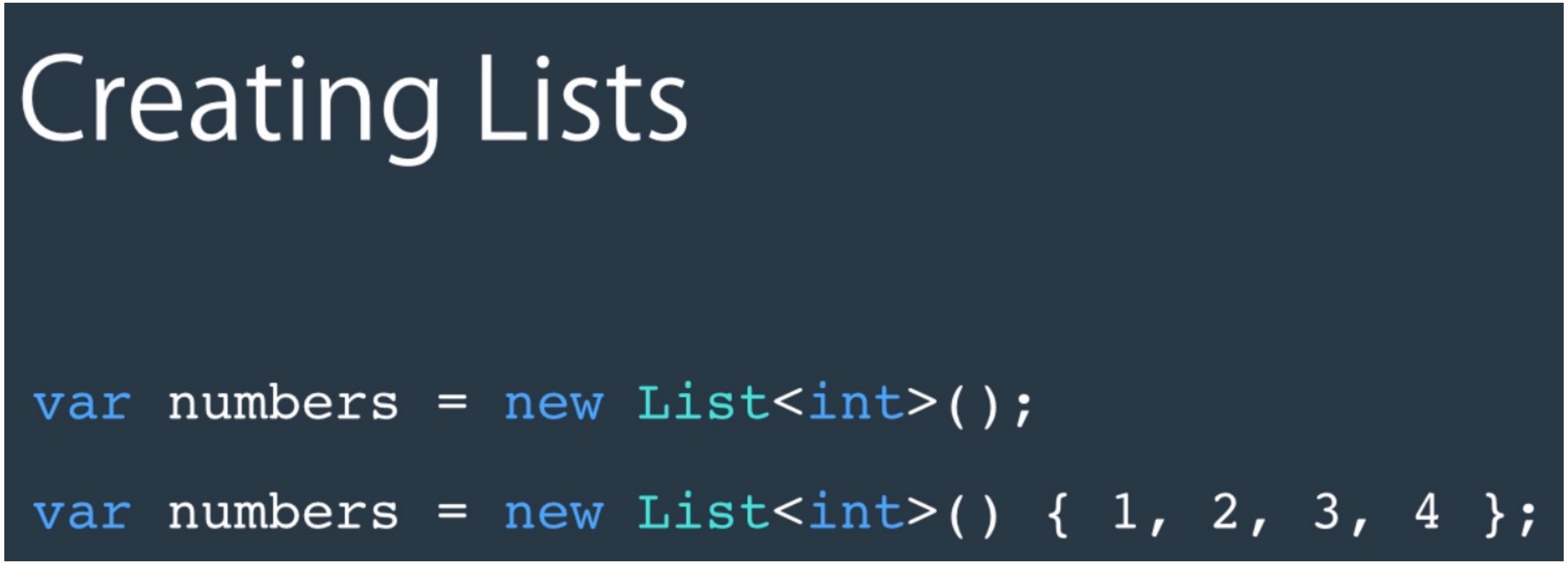
- Array: fixed size
- List: dynamic size



Creating Lists

```
var numbers = new List<int>();
```

List is a genaric type and is shown by <>

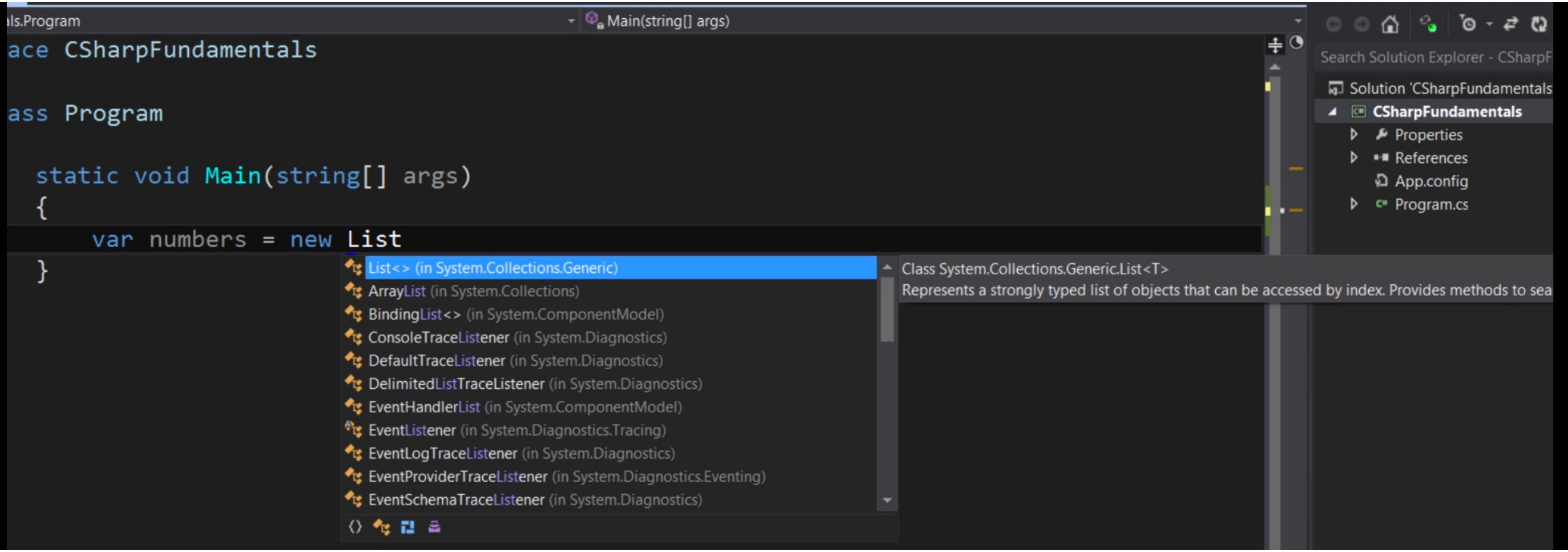


Creating Lists

```
var numbers = new List<int>();

var numbers = new List<int>() { 1, 2, 3, 4 };
```

Useful Methods

- Add()
- AddRange()
- Remove()
- RemoveAt()
- IndexOf()
- Contains()
- Count



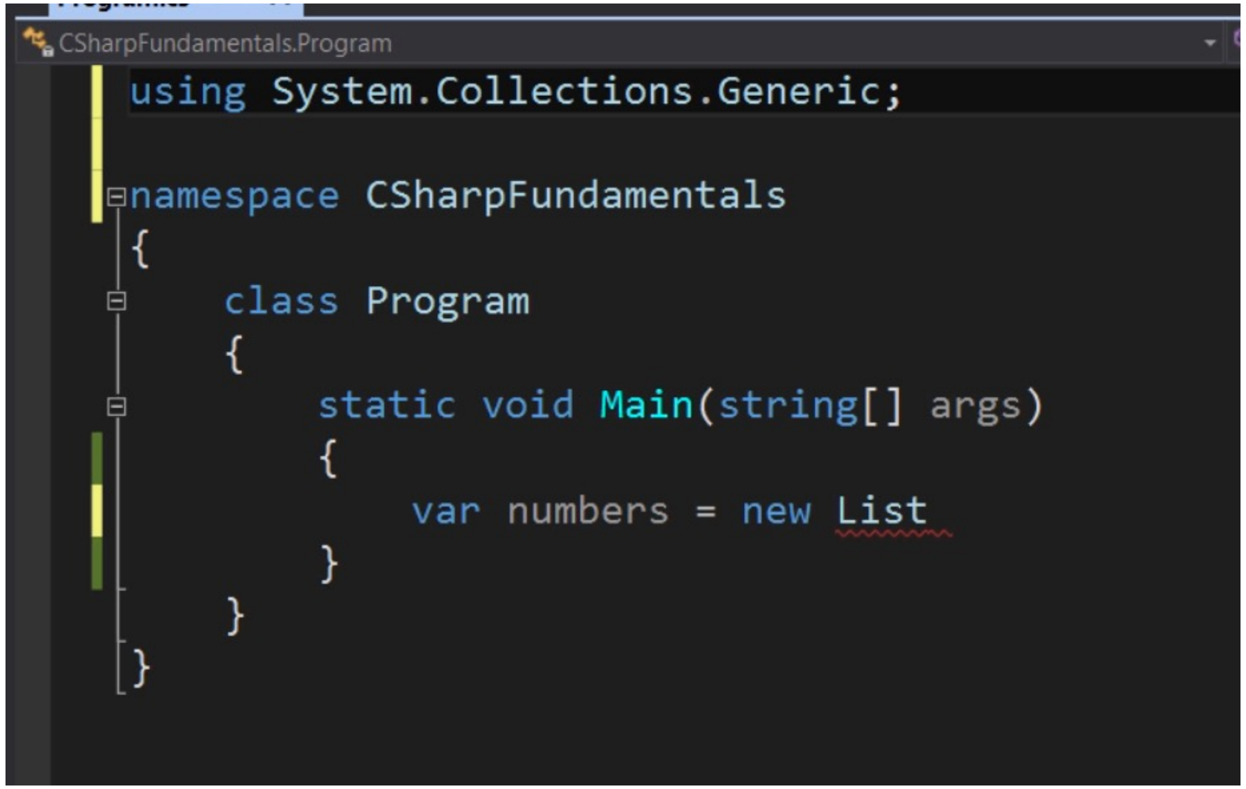
ls.Program | Main(string[] args)

```
ace CSharpFundamentals

ass Program

  static void Main(string[] args)
  {
      var numbers = new List
  }
```

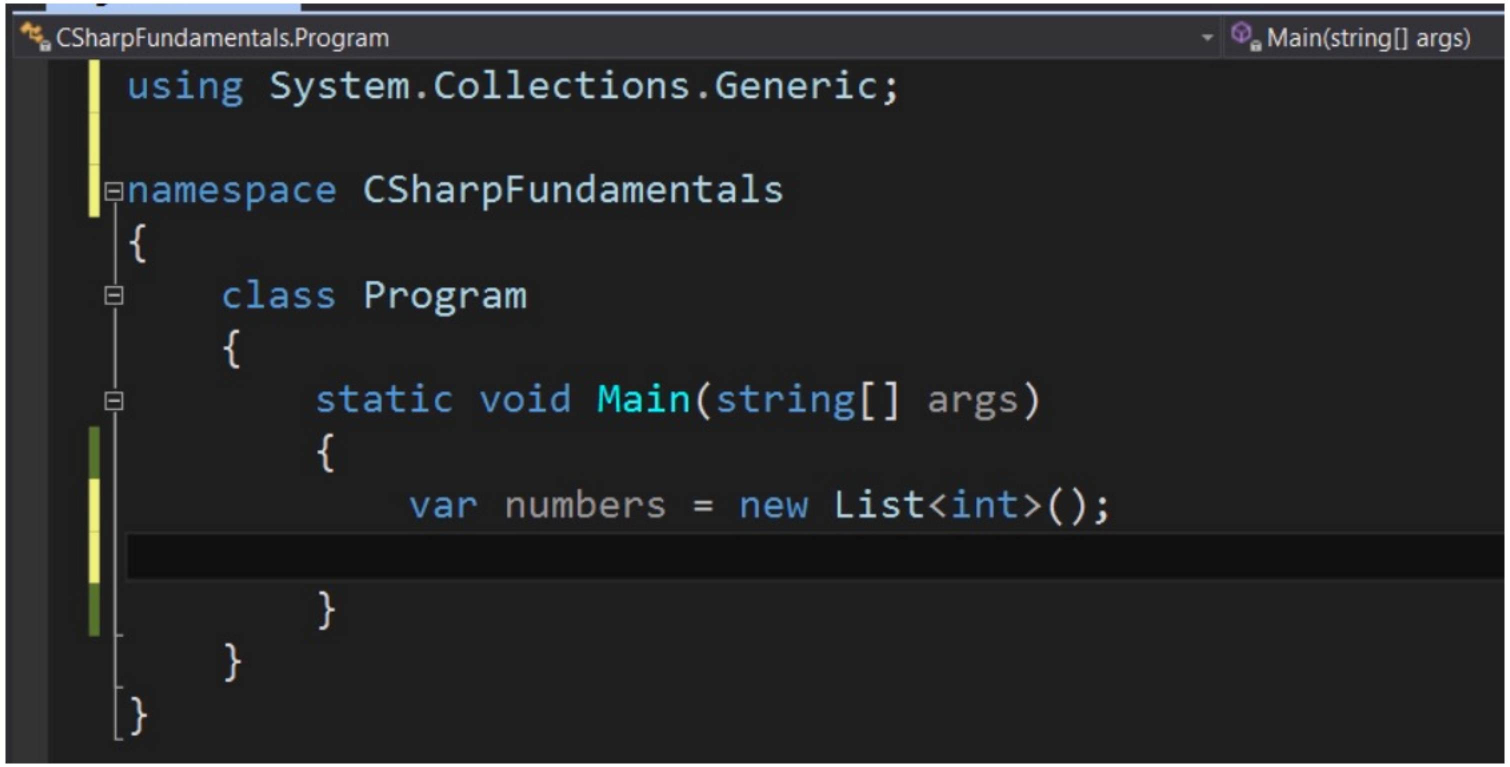
List<> (in System.Collections.Generic)
ArrayList (in System.Collections)
BindingList<> (in System.ComponentModel)
ConsoleTraceListener (in System.Diagnostics)
DefaultTraceListener (in System.Diagnostics)
DelimitedListTraceListener (in System.Diagnostics)
EventHandlerList (in System.ComponentModel)
EventListener (in System.Diagnostics.Tracing)
EventLogTraceListener (in System.Diagnostics)
EventProviderTraceListener (in System.Diagnostics.Eventing)
EventSchemaTraceListener (in System.Diagnostics)

Class System.Collections.Generic.List<T>
Represents a strongly typed list of objects that can be accessed by index. Provides methods to sea

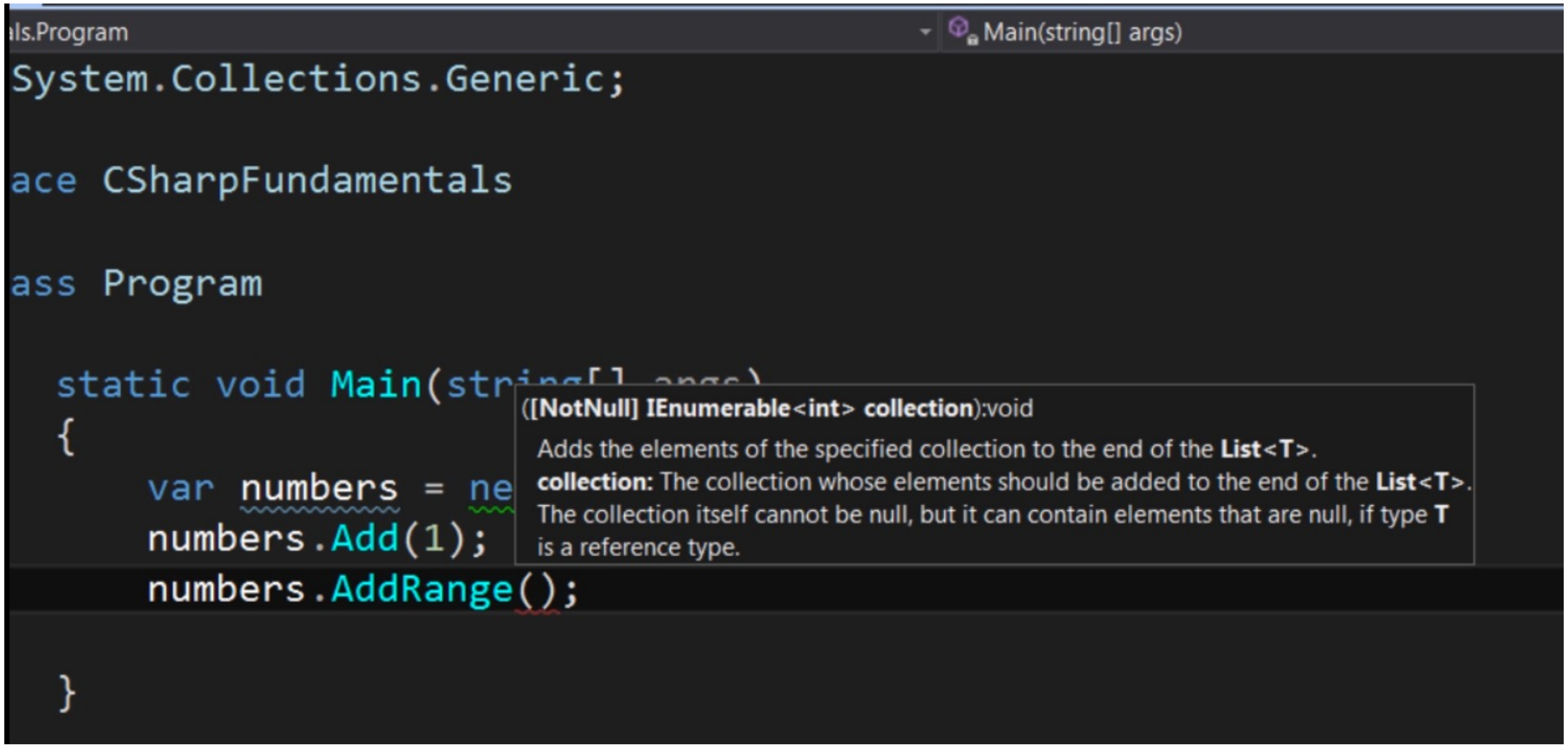
Search Solution Explorer - CSharpF
Solution 'CSharpFundamentals
CSharpFundamentals
Properties
References
App.config
Program.cs



```
using System.Collections.Generic;

namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var numbers = new List
        }
    }
}
```



```
using System.Collections.Generic;

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

        }
    }
}
```

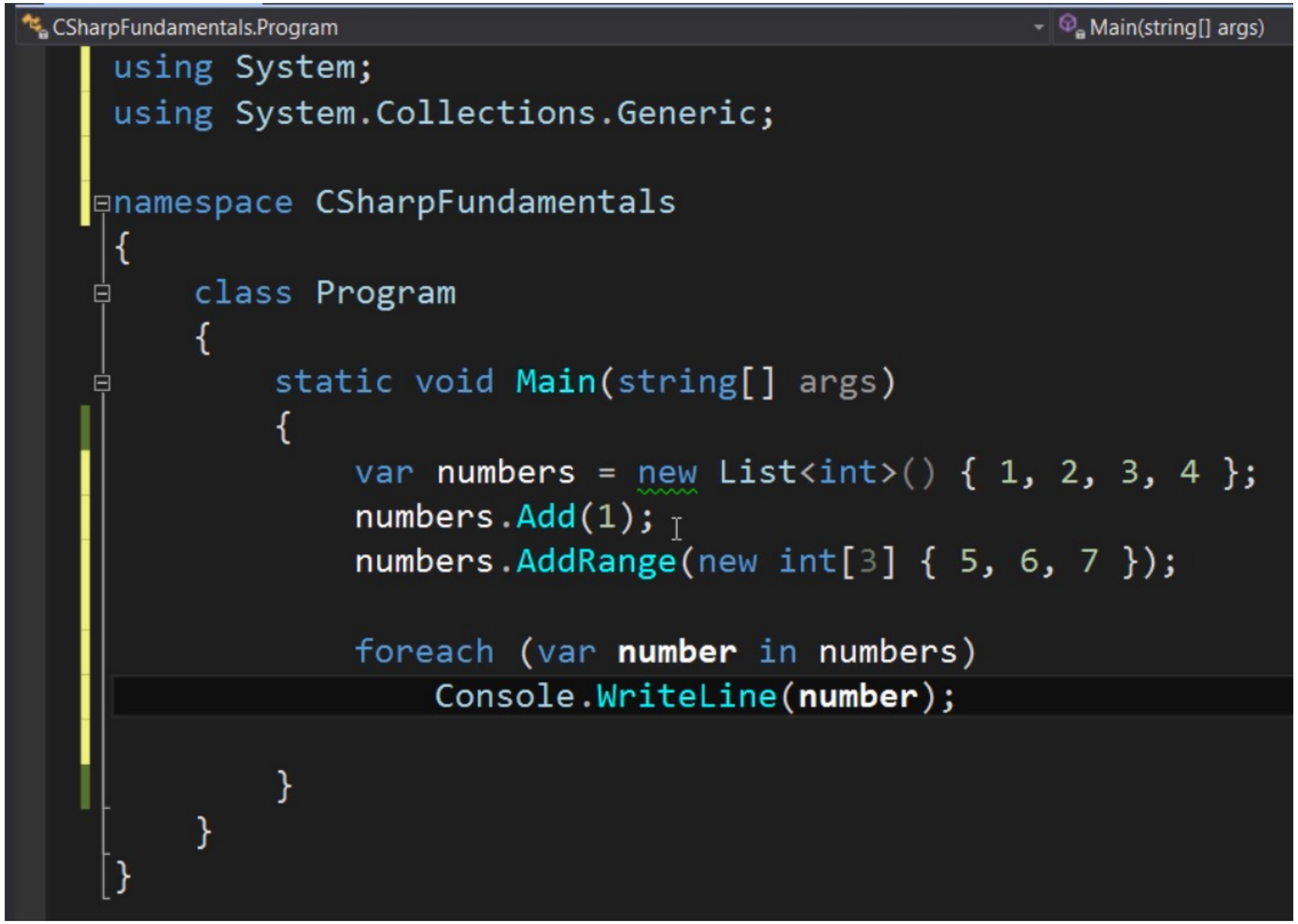


```
System.Collections.Generic;

ace CSharpFundamentals

ass Program

  static void Main(string[] args)
  {
      var numbers = ne
      numbers.Add(1);
      numbers.AddRange();

  }
```

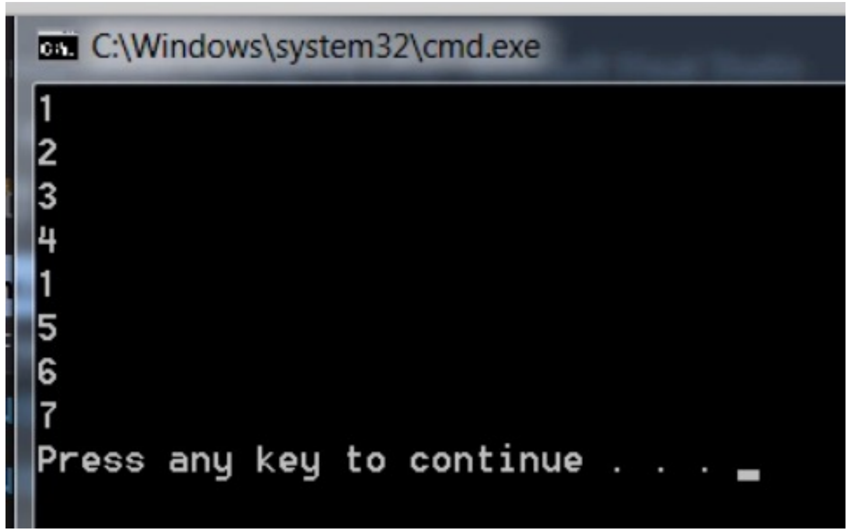
([NotNull] IEnumerable<int> collection):void

Adds the elements of the specified collection to the end of the **List<T>**.

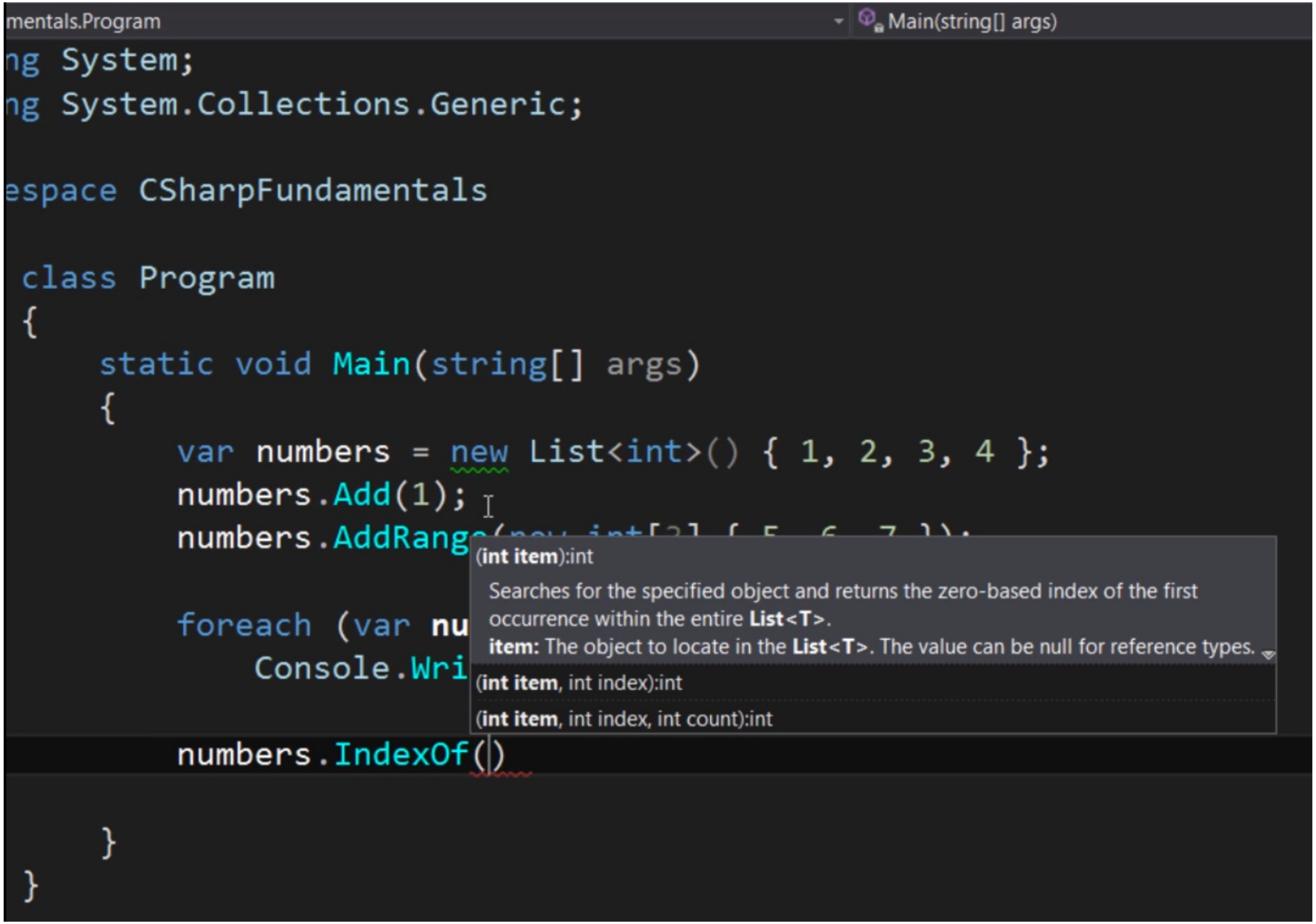
collection: The collection whose elements should be added to the end of the **List<T>**. The collection itself cannot be null, but it can contain elements that are null, if type **T** is a reference type.


CSharpFundamentals.Program   Main(string[] args)

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

namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var numbers = new List<int>() { 1, 2, 3, 4 };
            numbers.Add(1);
            numbers.AddRange(new int[3] { 5, 6, 7 });

            foreach (var number in numbers)
                Console.WriteLine(number);

        }
    }
}
```


C:\Windows\system32\cmd.exe

```
1
2
3
4
1
5
6
7
Press any key to continue . . .
```



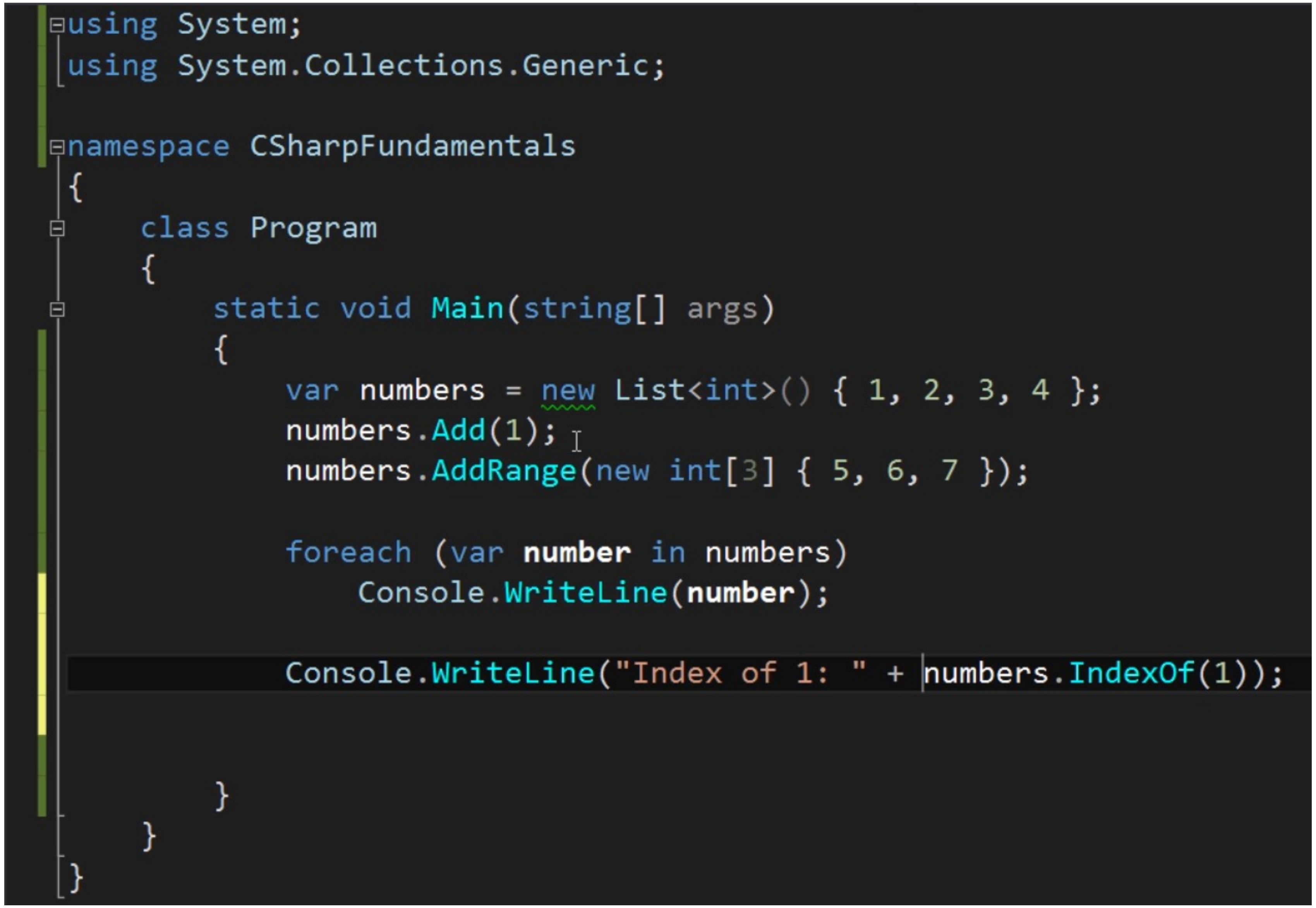
mentals.Program
Main(string[] args)

```
ng System;
ng System.Collections.Generic;

espace CSharpFundamentals

 class Program
 {
     static void Main(string[] args)
     {
         var numbers = new List<int>() { 1, 2, 3, 4 };
         numbers.Add(1);
         numbers.AddRang

         foreach (var nu
             Console.Wri

         numbers.IndexOf()

     }
 }
```

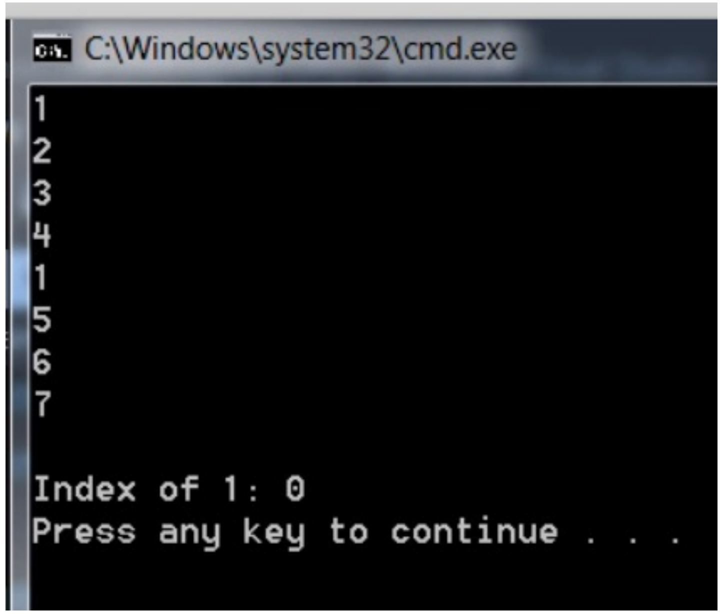
(int item):int
Searches for the specified object and returns the zero-based index of the first occurrence within the entire **List<T>**.
item: The object to locate in the **List<T>**. The value can be null for reference types.

(int item, int index):int

(int item, int index, int count):int

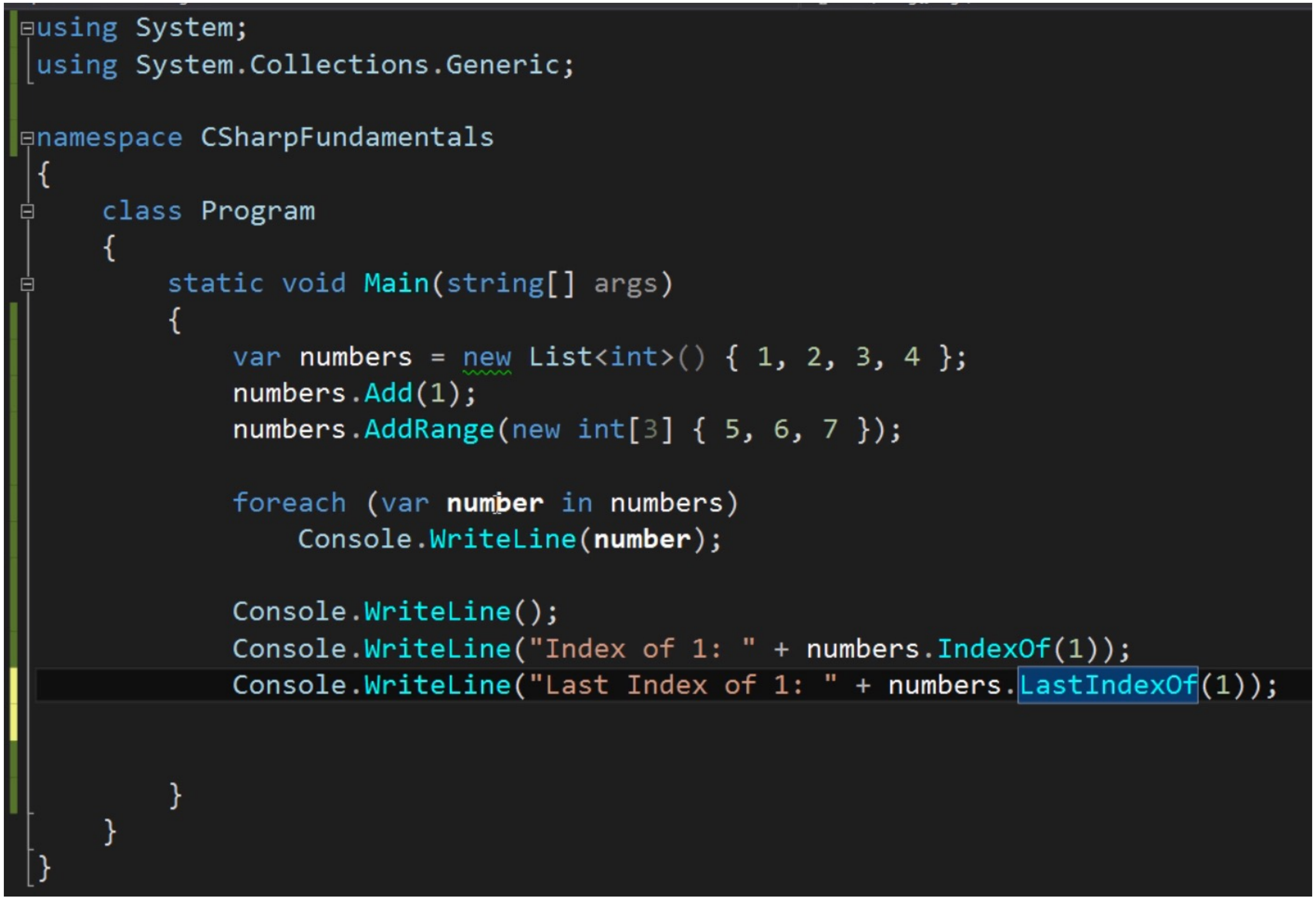


```
using System;
using System.Collections.Generic;

namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var numbers = new List<int>() { 1, 2, 3, 4 };
            numbers.Add(1);
            numbers.AddRange(new int[3] { 5, 6, 7 });

            foreach (var number in numbers)
                Console.WriteLine(number);

            Console.WriteLine("Index of 1: " + numbers.IndexOf(1));


        }
    }
}
```

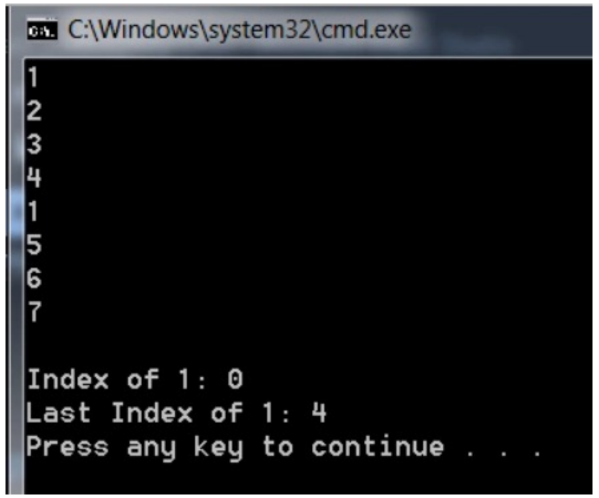


C:\Windows\system32\cmd.exe

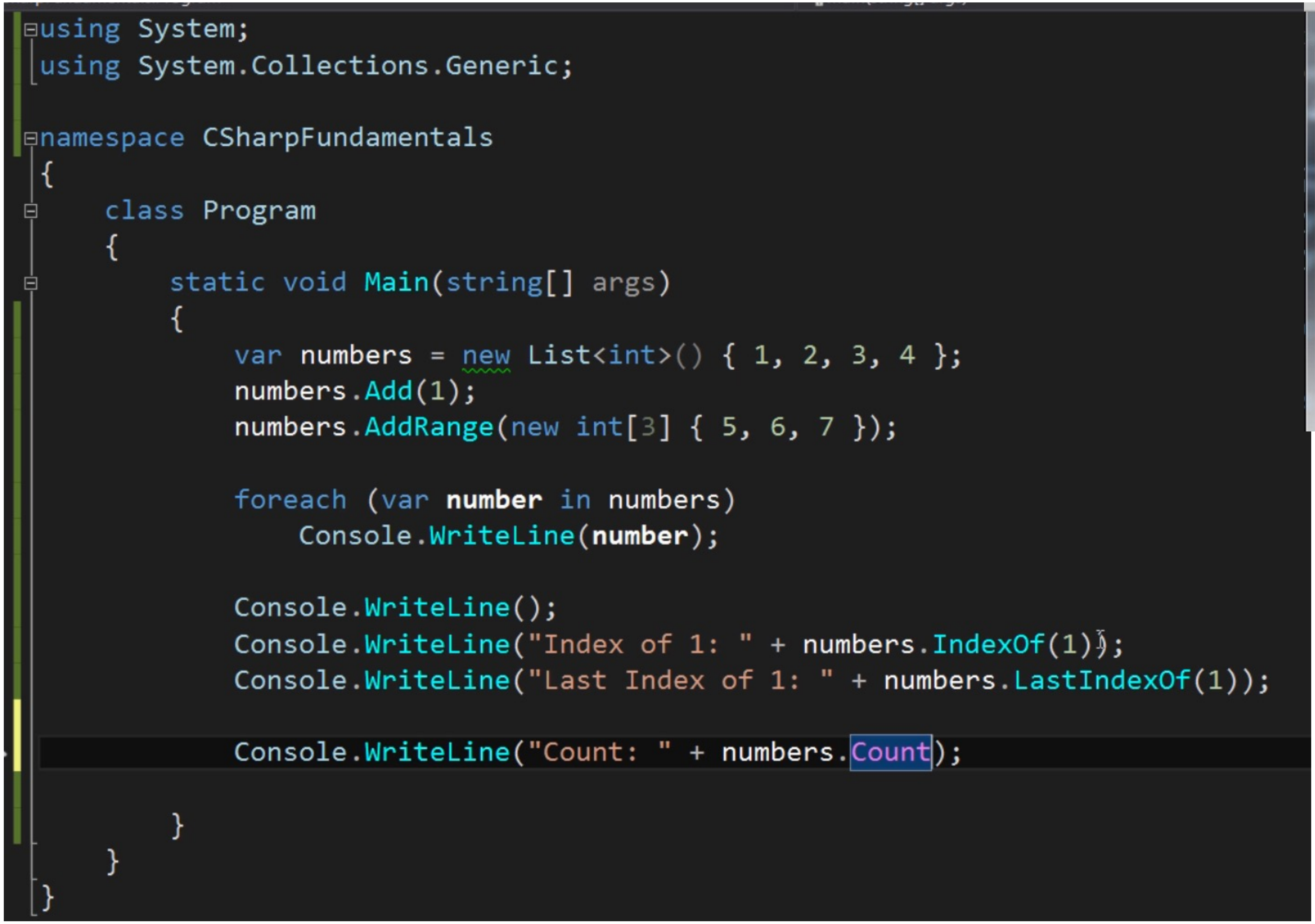
```
1
2
3
4
1
5
6
7

Index of 1: 0
Press any key to continue . . .
```



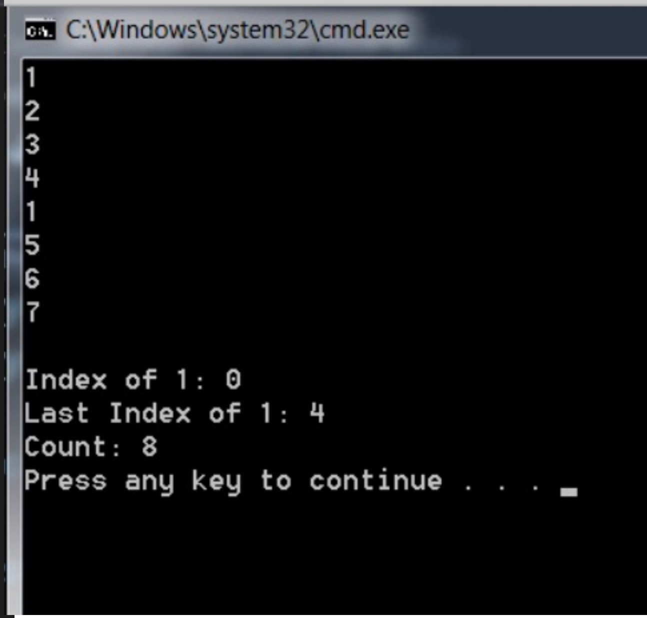
```csharp
using System;
using System.Collections.Generic;

namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var numbers = new List<int>() { 1, 2, 3, 4 };
            numbers.Add(1);
            numbers.AddRange(new int[3] { 5, 6, 7 });

            foreach (var number in numbers)
                Console.WriteLine(number);

            Console.WriteLine();
            Console.WriteLine("Index of 1: " + numbers.IndexOf(1));
            Console.WriteLine("Last Index of 1: " + numbers.LastIndexOf(1));


        }
    }
}
```



```
1
2
3
4
1
5
6
7

Index of 1: 0
Last Index of 1: 4
Press any key to continue . . .
```

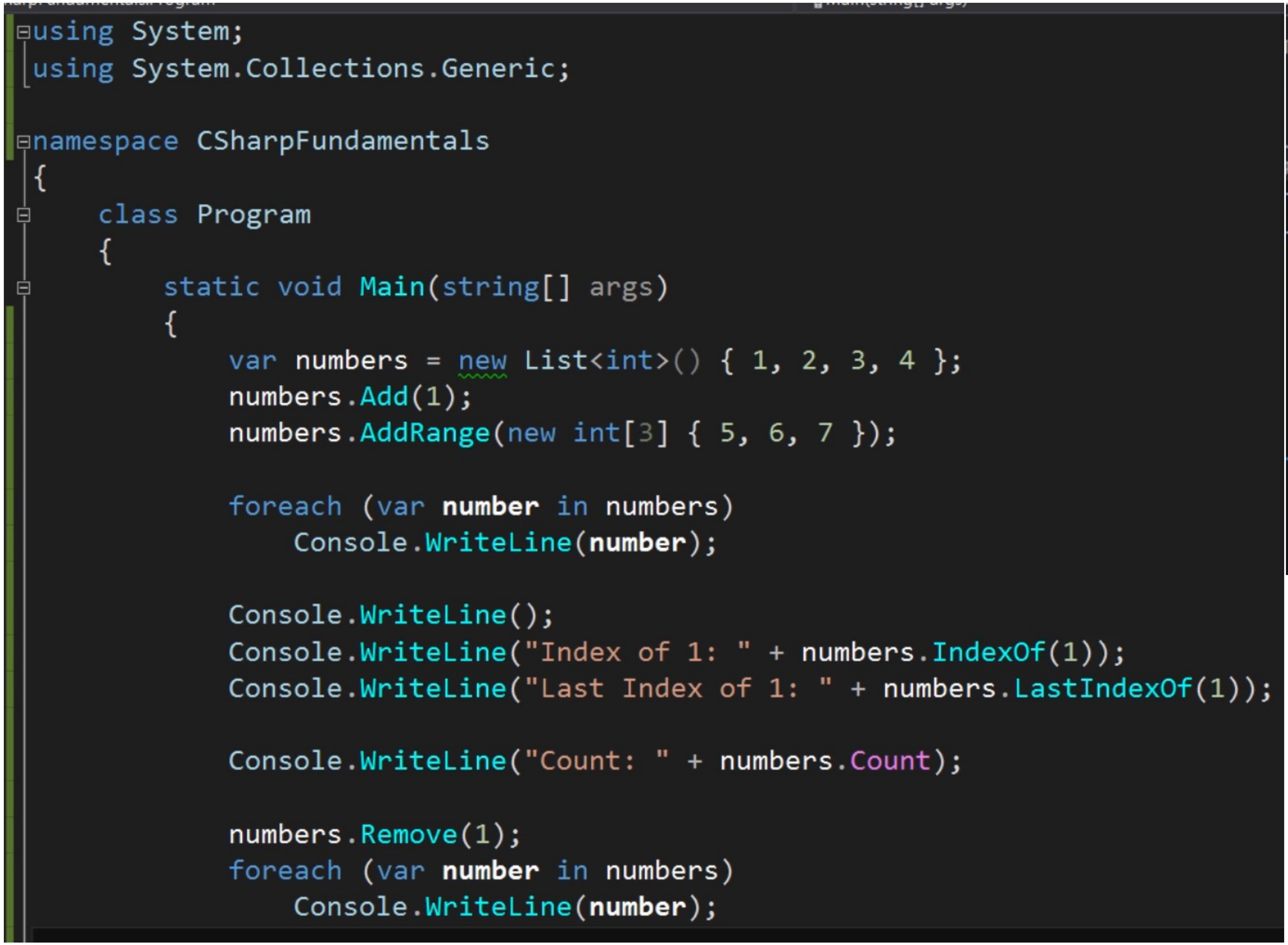


```
using System;
using System.Collections.Generic;

namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var numbers = new List<int>() { 1, 2, 3, 4 };
            numbers.Add(1);
            numbers.AddRange(new int[3] { 5, 6, 7 });

            foreach (var number in numbers)
                Console.WriteLine(number);

            Console.WriteLine();
            Console.WriteLine("Index of 1: " + numbers.IndexOf(1));
            Console.WriteLine("Last Index of 1: " + numbers.LastIndexOf(1));

            Console.WriteLine("Count: " + numbers.Count);

        }
    }
}
```

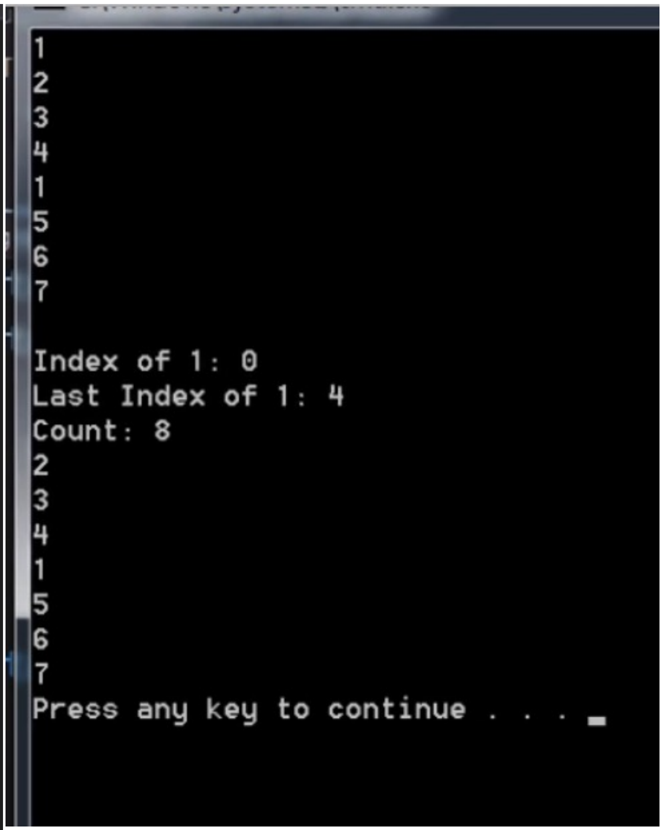


C:\Windows\system32\cmd.exe

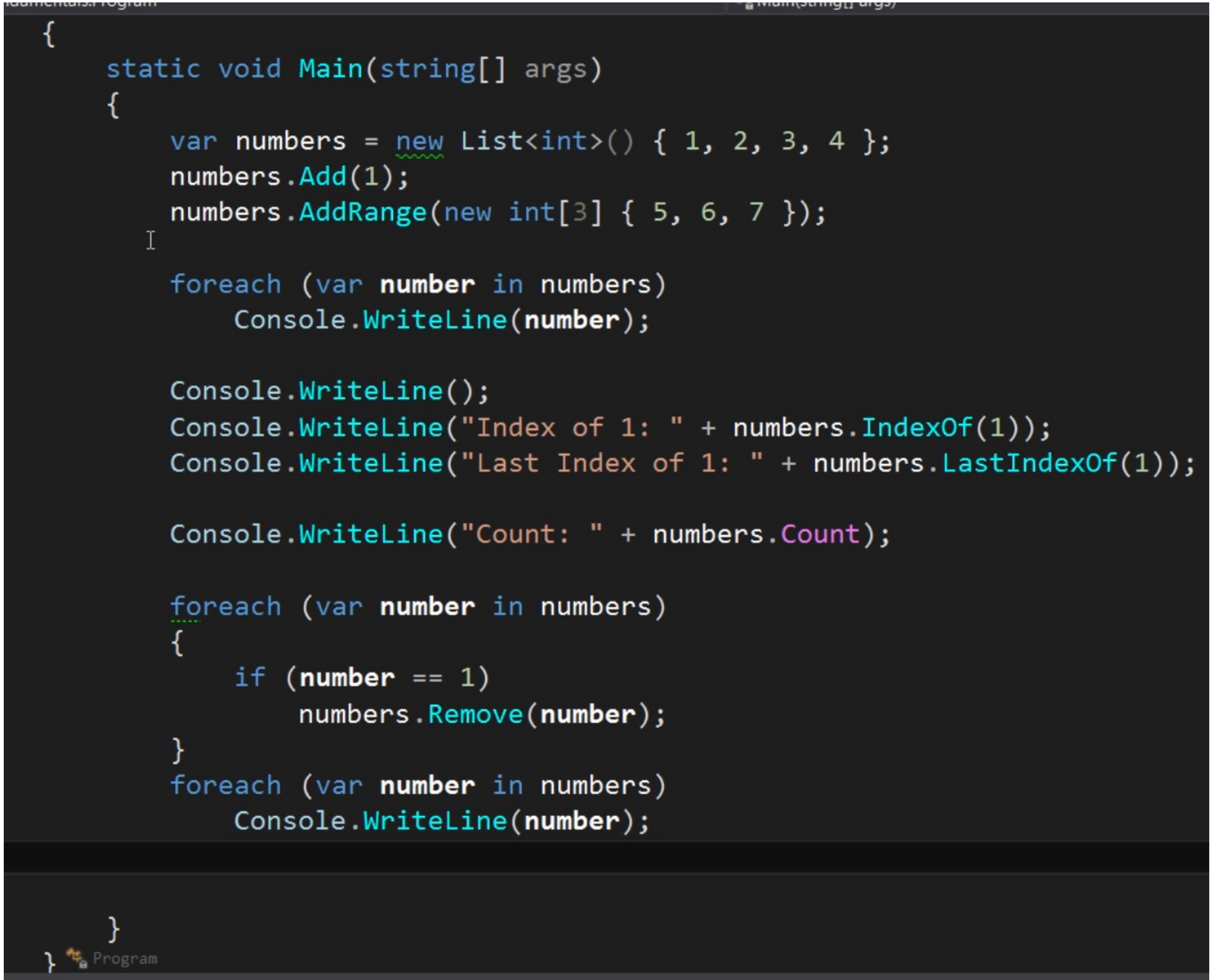
```
1
2
3
4
1
5
6
7

Index of 1: 0
Last Index of 1: 4
Count: 8
Press any key to continue . . .
```



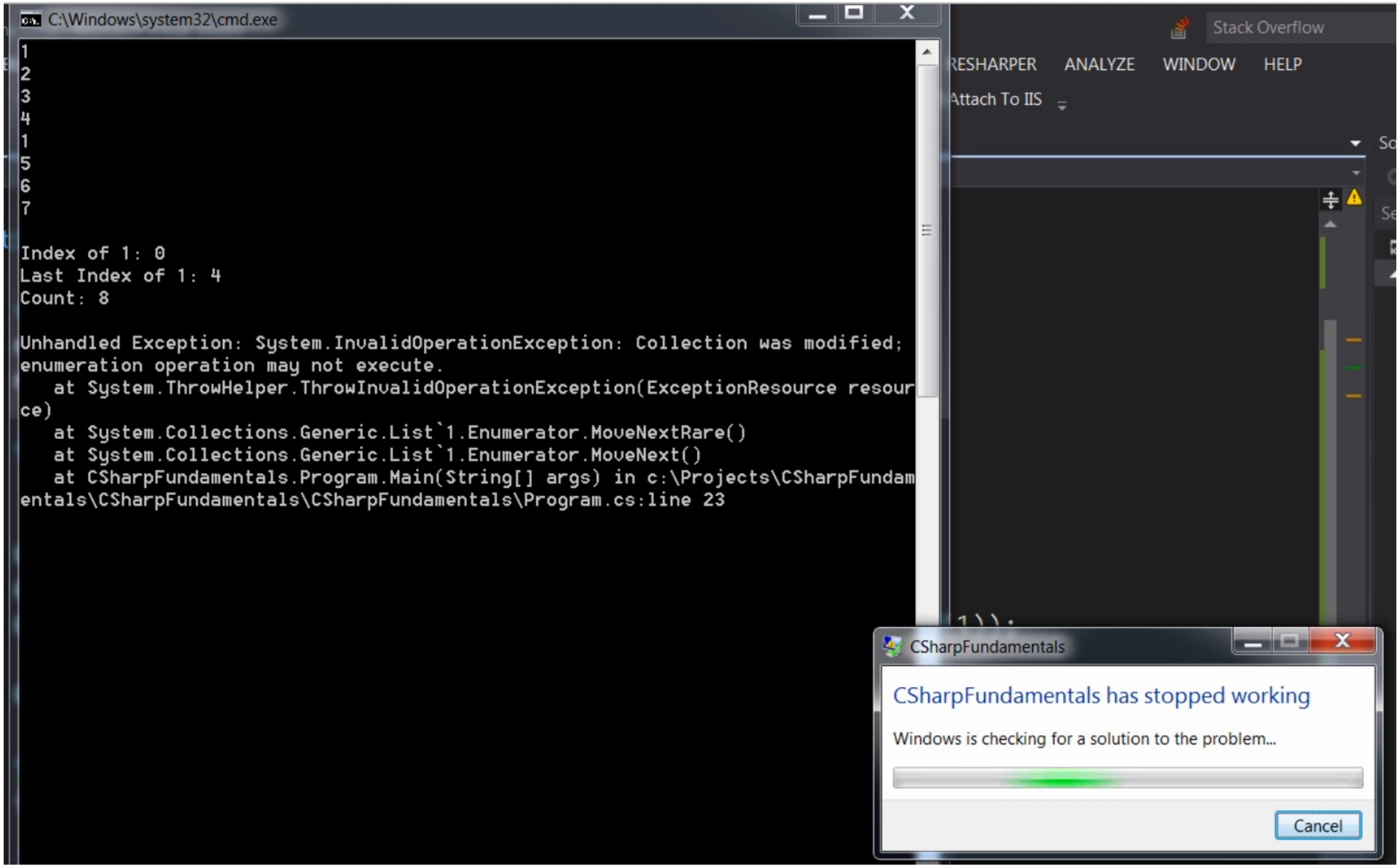
```csharp
using System;
using System.Collections.Generic;

namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var numbers = new List<int>() { 1, 2, 3, 4 };
            numbers.Add(1);
            numbers.AddRange(new int[3] { 5, 6, 7 });

            foreach (var number in numbers)
                Console.WriteLine(number);

            Console.WriteLine();
            Console.WriteLine("Index of 1: " + numbers.IndexOf(1));
            Console.WriteLine("Last Index of 1: " + numbers.LastIndexOf(1));

            Console.WriteLine("Count: " + numbers.Count);

            numbers.Remove(1);
            foreach (var number in numbers)
                Console.WriteLine(number);
```



```
1
2
3
4
1
5
6
7

Index of 1: 0
Last Index of 1: 4
Count: 8
2
3
4
1
5
6
7
Press any key to continue . . .
```



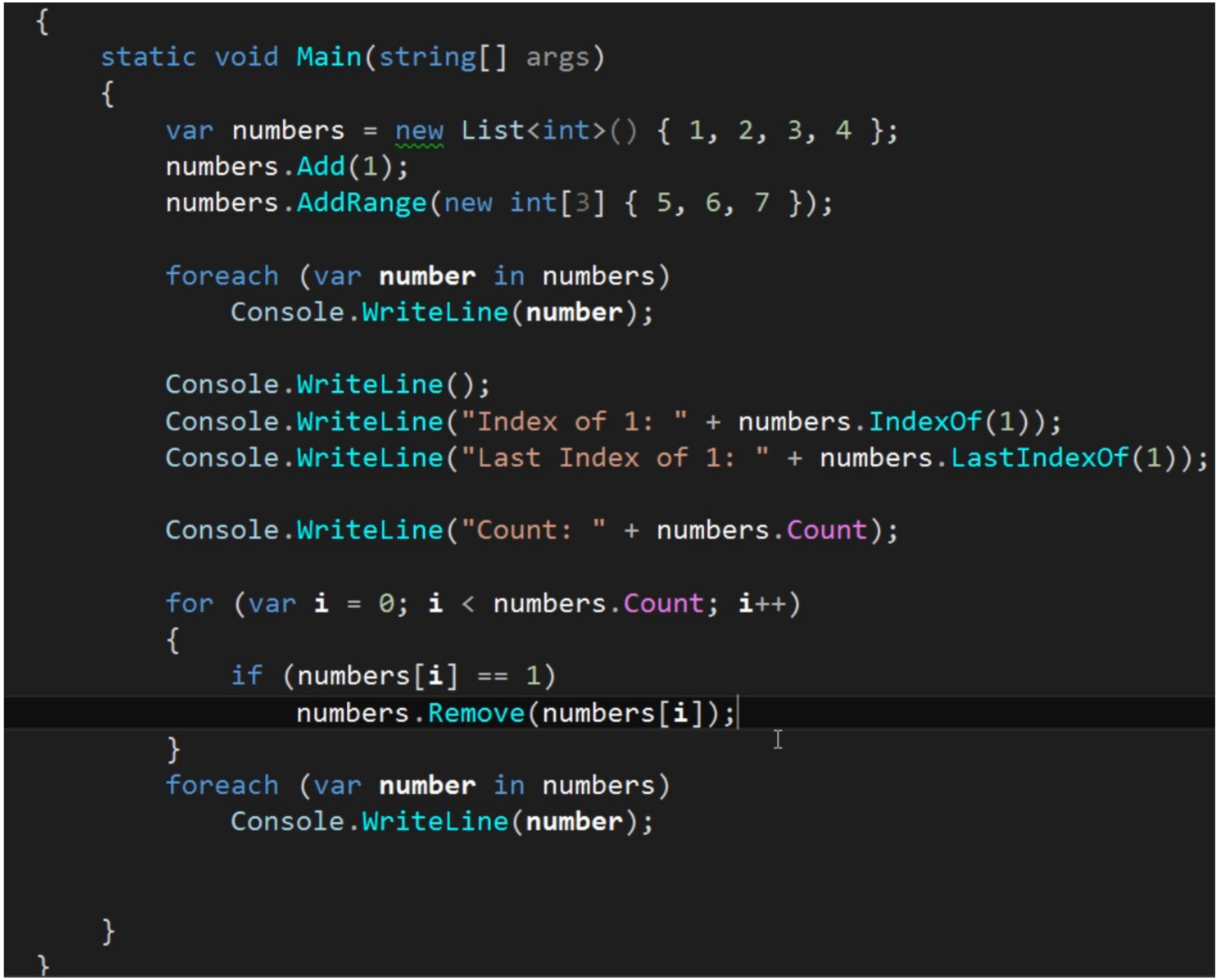
```
{
    static void Main(string[] args)
    {
        var numbers = new List<int>() { 1, 2, 3, 4 };
        numbers.Add(1);
        numbers.AddRange(new int[3] { 5, 6, 7 });

        foreach (var number in numbers)
            Console.WriteLine(number);

        Console.WriteLine();
        Console.WriteLine("Index of 1: " + numbers.IndexOf(1));
        Console.WriteLine("Last Index of 1: " + numbers.LastIndexOf(1));

        Console.WriteLine("Count: " + numbers.Count);

        foreach (var number in numbers)
        {
            if (number == 1)
                numbers.Remove(number);
        }
        foreach (var number in numbers)
            Console.WriteLine(number);


    }
}
```



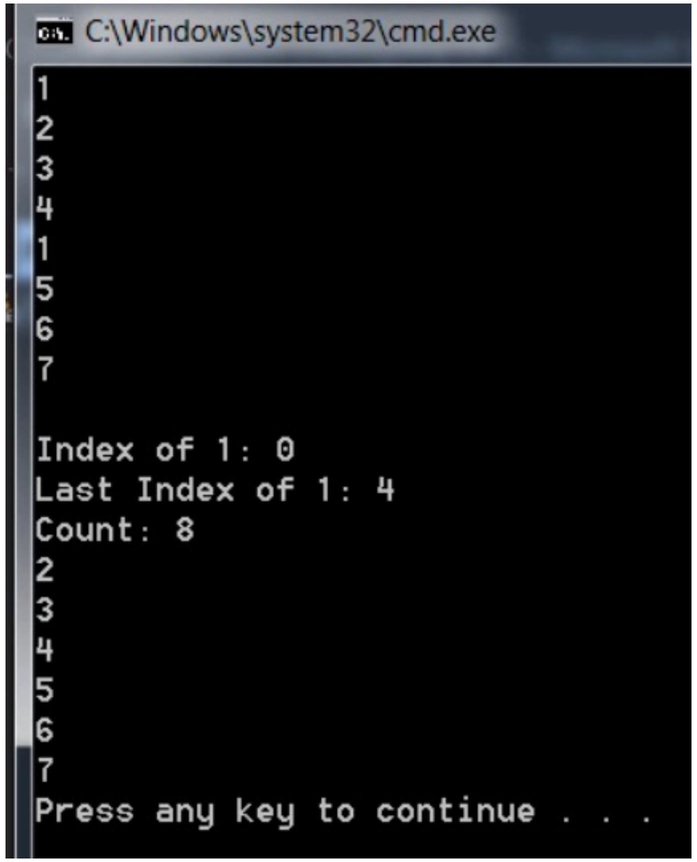
```
1
2
3
4
1
5
6
7

Index of 1: 0
Last Index of 1: 4
Count: 8

Unhandled Exception: System.InvalidOperationException: Collection was modified;
enumeration operation may not execute.
   at System.ThrowHelper.ThrowInvalidOperationException(ExceptionResource resour
ce)
   at System.Collections.Generic.List`1.Enumerator.MoveNextRare()
   at System.Collections.Generic.List`1.Enumerator.MoveNext()
   at CSharpFundamentals.Program.Main(String[] args) in c:\Projects\CSharpFundam
entals\CSharpFundamentals\CSharpFundamentals\Program.cs:line 23
```

CSharpFundamentals has stopped working

Windows is checking for a solution to the problem...

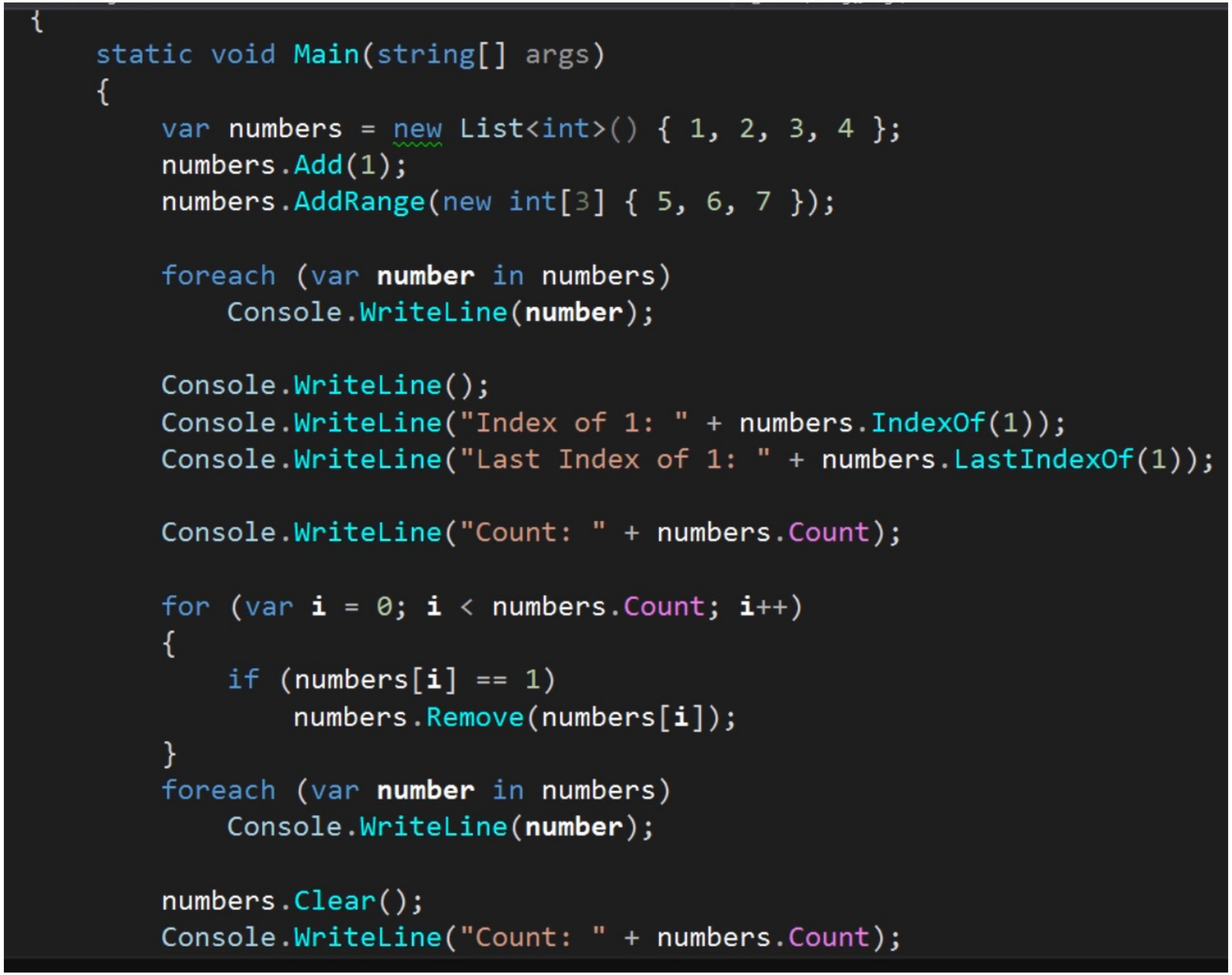


```
{
    static void Main(string[] args)
    {
        var numbers = new List<int>() { 1, 2, 3, 4 };
        numbers.Add(1);
        numbers.AddRange(new int[3] { 5, 6, 7 });

        foreach (var number in numbers)
            Console.WriteLine(number);

        Console.WriteLine();
        Console.WriteLine("Index of 1: " + numbers.IndexOf(1));
        Console.WriteLine("Last Index of 1: " + numbers.LastIndexOf(1));

        Console.WriteLine("Count: " + numbers.Count);

        for (var i = 0; i < numbers.Count; i++)
        {
            if (numbers[i] == 1)
                numbers.Remove(numbers[i]);
        }
        foreach (var number in numbers)
            Console.WriteLine(number);


    }
}
```

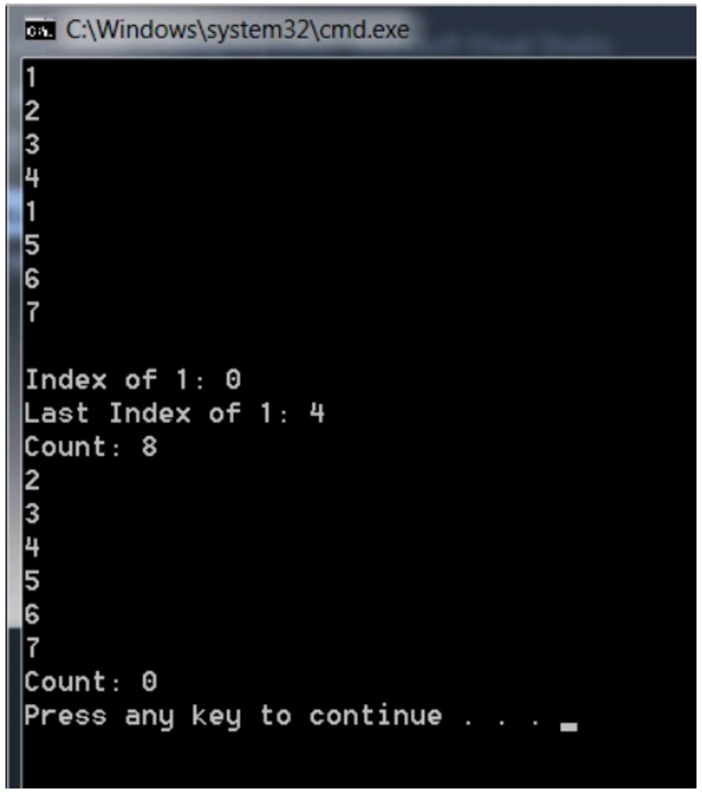


C:\Windows\system32\cmd.exe

```
1
2
3
4
1
5
6
7

Index of 1: 0
Last Index of 1: 4
Count: 8
2
3
4
5
6
7
Press any key to continue . . .
```



```
{
    static void Main(string[] args)
    {
        var numbers = new List<int>() { 1, 2, 3, 4 };
        numbers.Add(1);
        numbers.AddRange(new int[3] { 5, 6, 7 });

        foreach (var number in numbers)
            Console.WriteLine(number);

        Console.WriteLine();
        Console.WriteLine("Index of 1: " + numbers.IndexOf(1));
        Console.WriteLine("Last Index of 1: " + numbers.LastIndexOf(1));

        Console.WriteLine("Count: " + numbers.Count);

        for (var i = 0; i < numbers.Count; i++)
        {
            if (numbers[i] == 1)
                numbers.Remove(numbers[i]);
        }
        foreach (var number in numbers)
            Console.WriteLine(number);

        numbers.Clear();
        Console.WriteLine("Count: " + numbers.Count);
```



```
1
2
3
4
1
5
6
7

Index of 1: 0
Last Index of 1: 4
Count: 8
2
3
4
5
6
7
Count: 0
Press any key to continue . . .
```

Excersise

Note: For any of these exercises, ignore input validation unless otherwise directed. Assume the user enters values in the format that the program expects.

1- When you post a message on Facebook, depending on the number of people who like your post, Facebook displays different information.

- If no one likes your post, it doesn't display anything.
- If only one person likes your post, it displays: [Friend's Name] likes your post.
- If two people like your post, it displays: [Friend 1] and [Friend 2] like your post.
- If more than two people like your post, it displays: [Friend 1], [Friend 2] and [Number of Other People] others like your post.

Write a program and continuously ask the user to enter different names, until the user presses Enter (without supplying a name). Depending on the number of names provided, display a message based on the above pattern.

2- Write a program and ask the user to enter their name. Use an array to reverse the name and then store the result in a new string. Display the reversed name on the console.

3- Write a program and ask the user to enter 5 numbers. If a number has been previously entered, display an error message and ask the user to re-try. Once the user successfully enters 5 unique numbers, sort them and display the result on the console.

4- Write a program and ask the user to continuously enter a number or type "Quit" to exit. The list of numbers may include duplicates. Display the unique numbers that the user has entered.

5- Write a program and ask the user to supply a list of comma separated numbers (e.g 5, 1, 9, 2, 10). If the list is empty or includes less than 5 numbers, display "Invalid List" and ask the user to re
try; otherwise, display the 3 smallest numbers in the list.