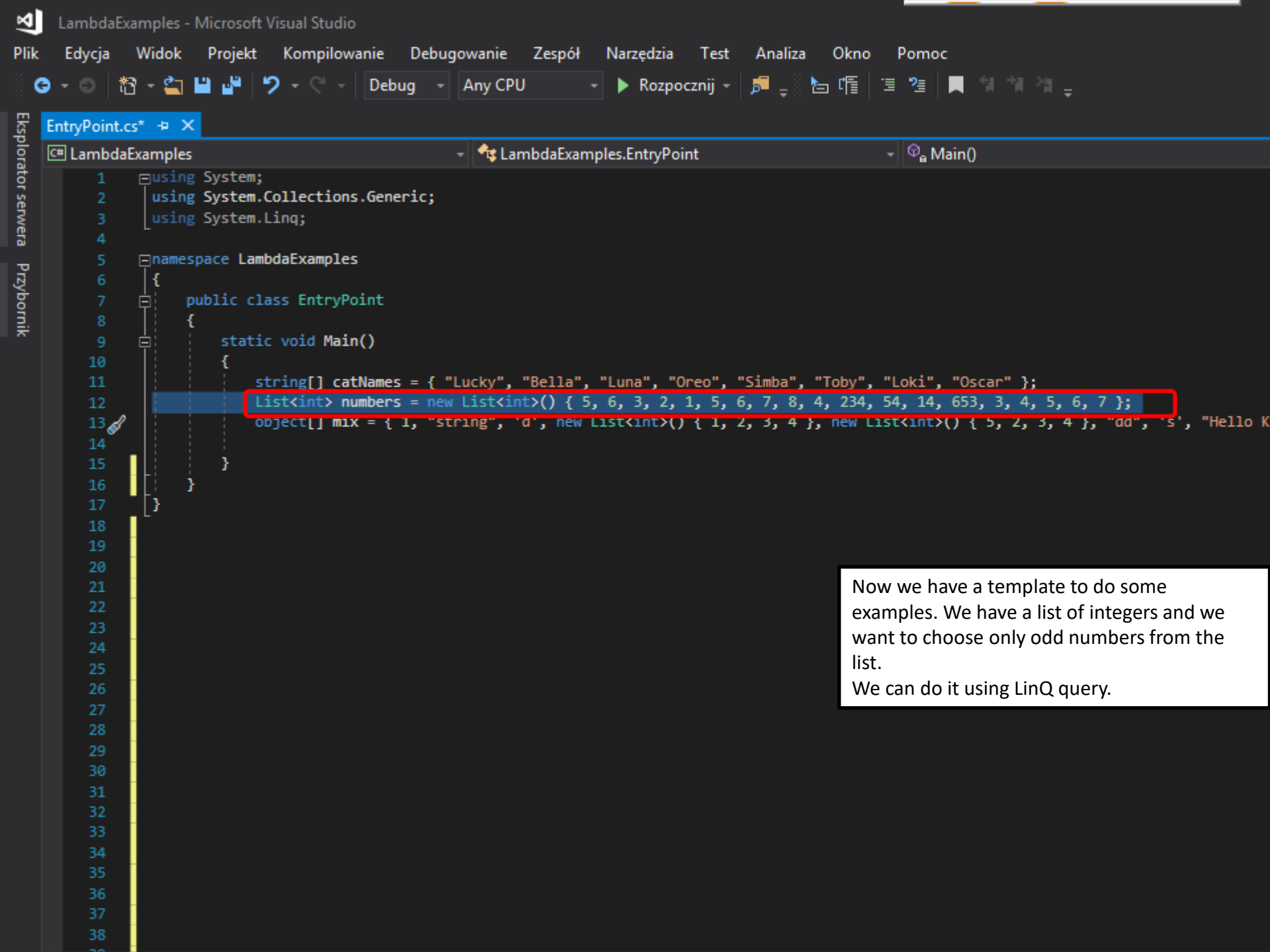
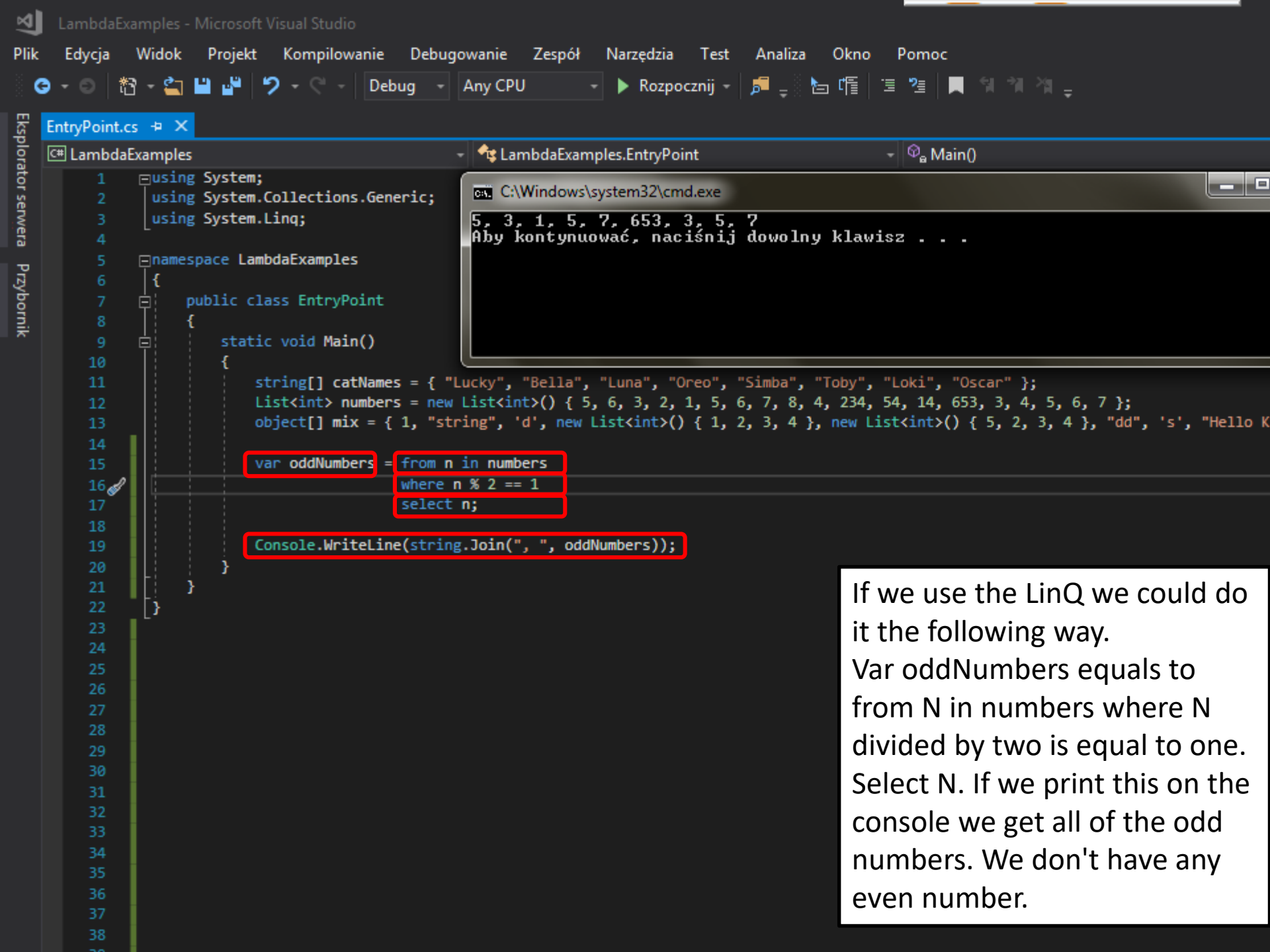
Lambda expressions

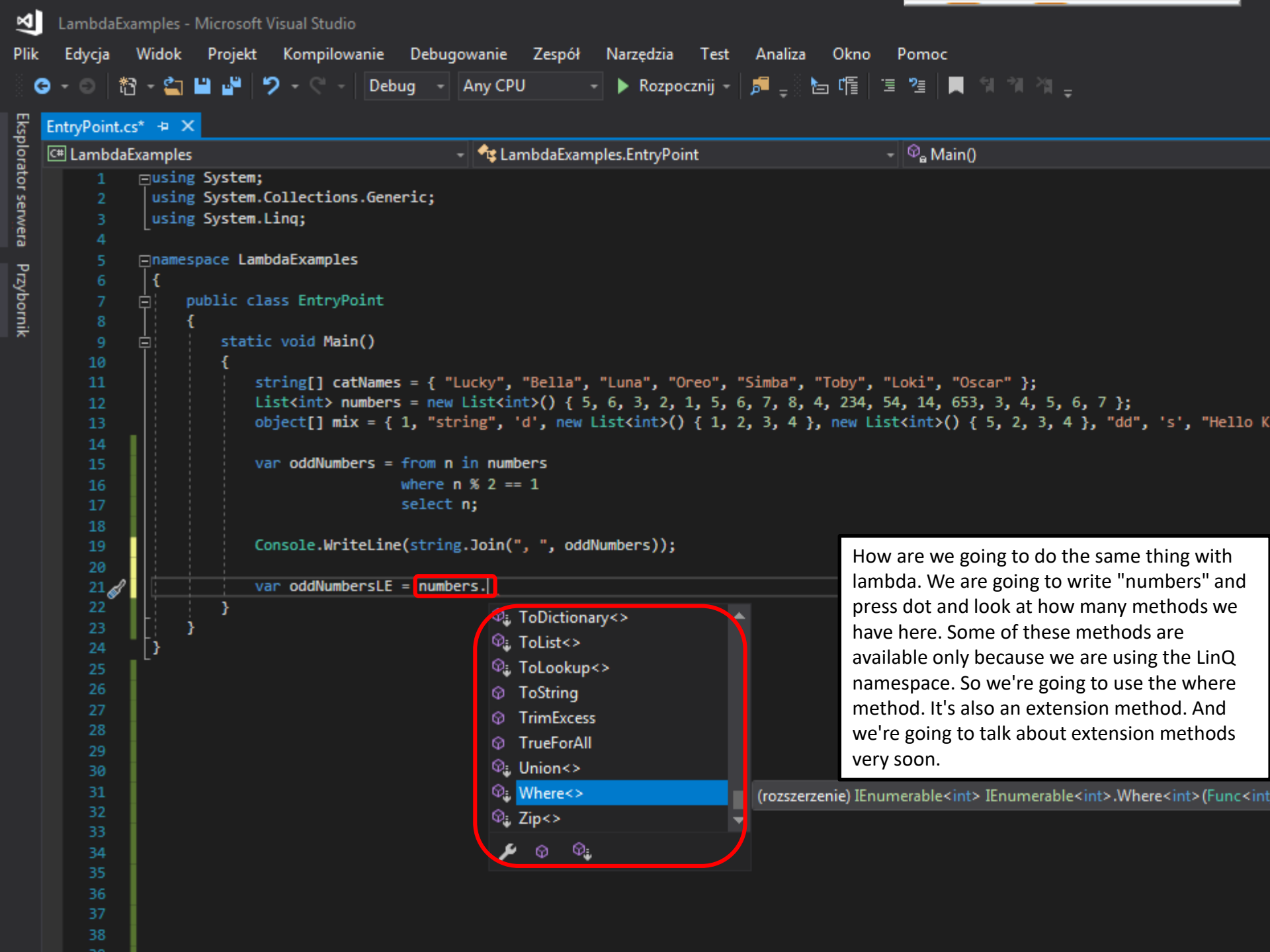
- Denoted by the lambda operator =>
- Input => (work on the input);
- N => ((N%2) == 1);

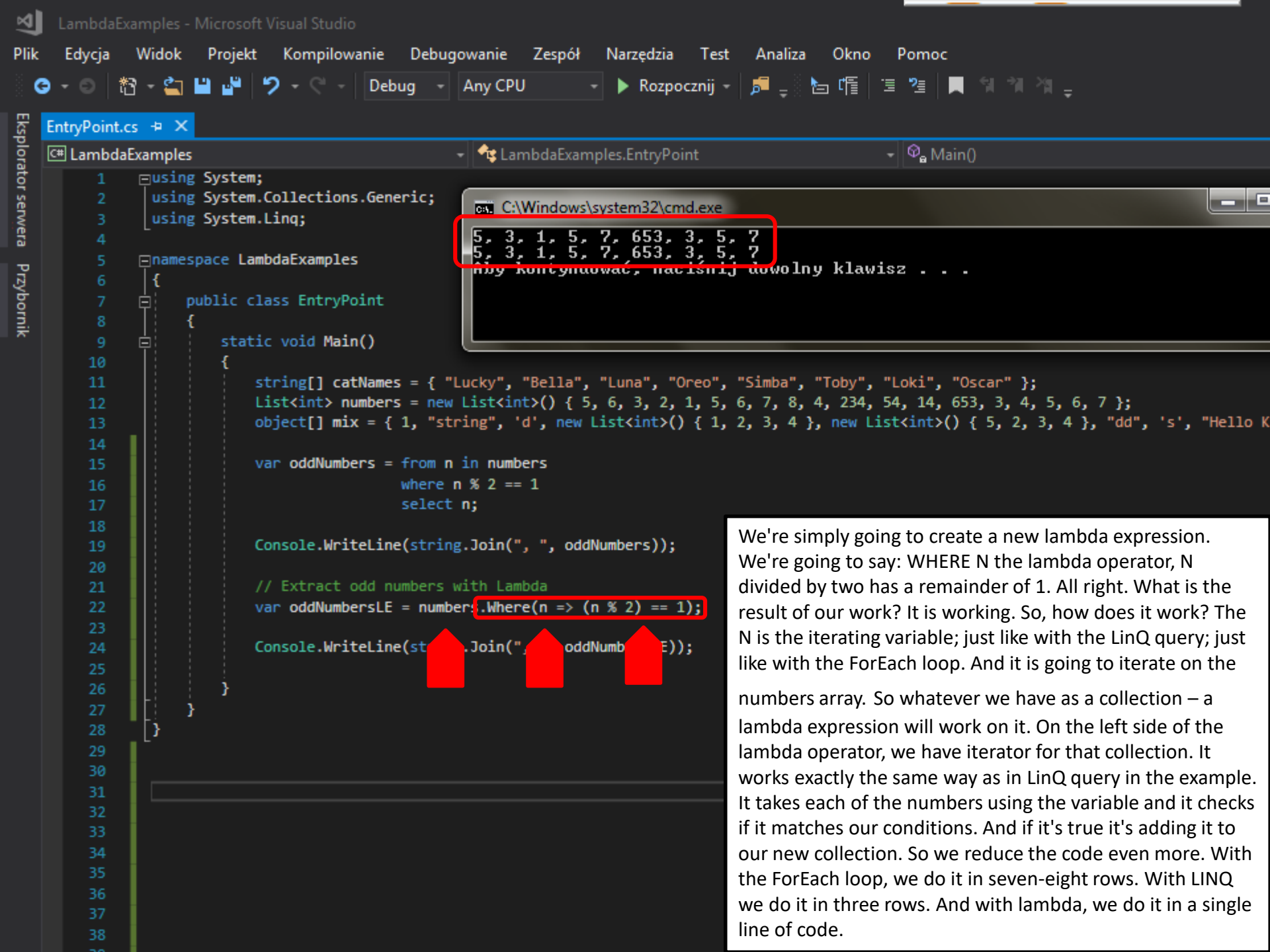
The lambda operator which is an equal sign and bigger than arrow.
It's used to separate the input values on the left side of the arrow, and the body of the code on the right side.
So basically we have an input on the left side and some code that is going to work on this input.
It’s simply a method without a decoration.
It lets you create your method in the same place where you're going to use if that needs to be used only once.
It saves you the effort of writing a separate method.
And there is the simple example to do with lambda expression.
We have one input which is just N variable. Let's say that this is a number an instance for number.
We're checking if N divided by two has remainder of 1 and then we're going to return some value to the variable that lambda expression.
When we check if a given number divides by two with a remainder or without remainder.
If we get a 0 it's an even number.
If we get a 1 it's an odd number.
So basically we're checking if it is an odd number.


LambdaExamples - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
Rozpocznij
EntryPoint.cs*
LambdaExamples
LambdaExamples.EntryPoint
Main()
Eksplorator serwera
Przybornik
using System;
using System.Collections.Generic;
using System.Linq;
namespace LambdaExamples
{
public class EntryPoint
{
static void Main()
{
string[] catNames = { "Lucky", "Bella", "Luna", "Oreo", "Simba", "Toby", "Loki", "Oscar" };
List<int> numbers = new List<int>() { 5, 6, 3, 2, 1, 5, 6, 7, 8, 4, 234, 54, 14, 653, 3, 4, 5, 6, 7 };
object[] mix = { 1, "string", 'd', new List<int>() { 1, 2, 3, 4 }, new List<int>() { 5, 2, 3, 4 }, "dd", 's', "Hello K
}
}
}
Now we have a template to do some examples. We have a list of integers and we want to choose only odd numbers from the list.
We can do it using LinQ query.

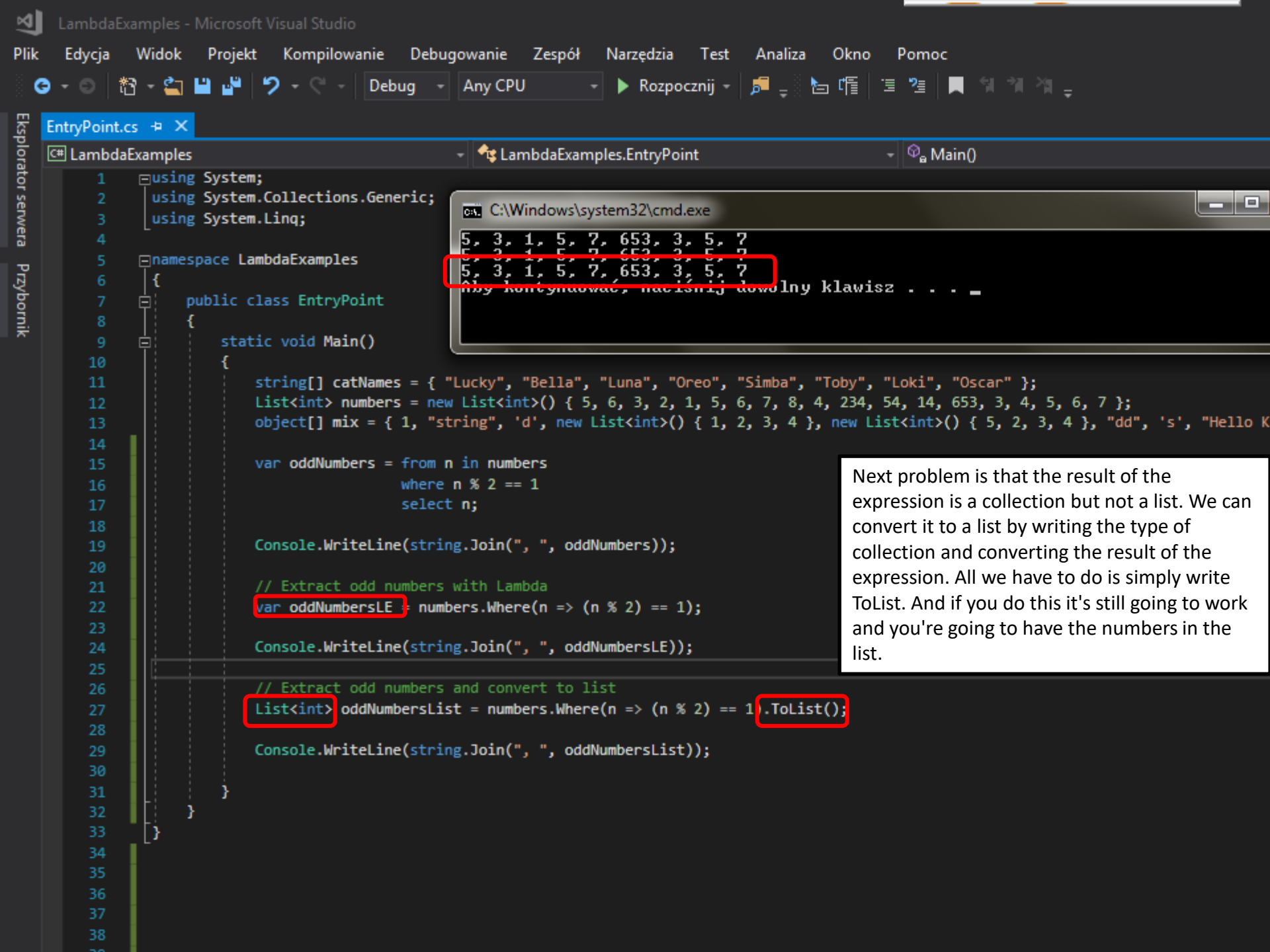


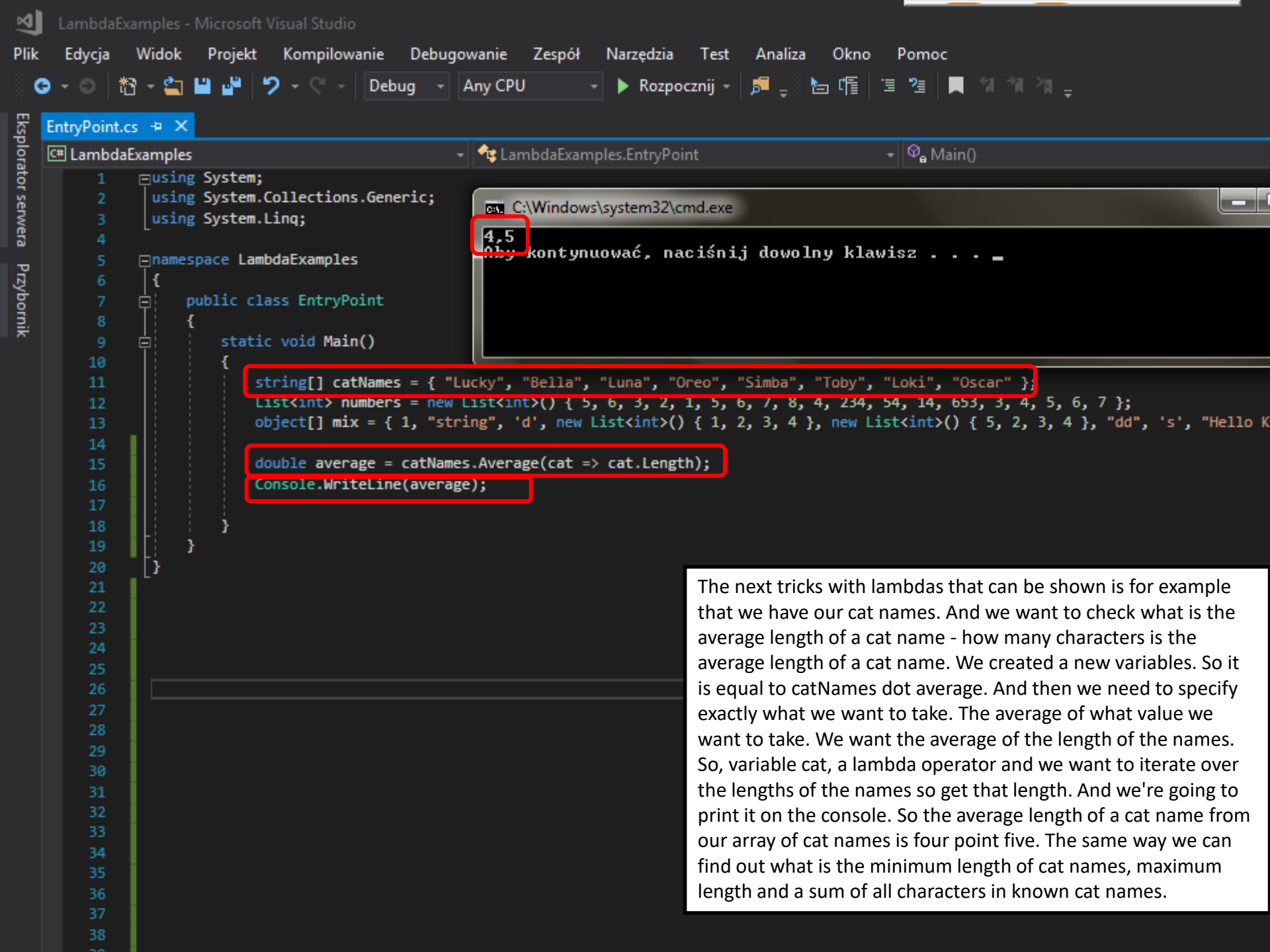
If we use the LinQ we could do it the following way.
Var oddNumbers equals to from N in numbers where N divided by two is equal to one. Select N. If we print this on the console we get all of the odd numbers. We don't have any even number.


LambdaExamples - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
Rozpocznij
EntryPoint.cs*
LambdaExamples
LambdaExamples.EntryPoint
Main()
Eksplorator serwera
Przybornik
using System;
using System.Collections.Generic;
using System.Linq;
namespace LambdaExamples
{
public class EntryPoint
{
static void Main()
{
string[] catNames = { "Lucky", "Bella", "Luna", "Oreo", "Simba", "Toby", "Loki", "Oscar" };
List<int> numbers = new List<int>() { 5, 6, 3, 2, 1, 5, 6, 7, 8, 4, 234, 54, 14, 653, 3, 4, 5, 6, 7 };
object[] mix = { 1, "string", 'd', new List<int>() { 1, 2, 3, 4 }, new List<int>() { 5, 2, 3, 4 }, "dd", 's', "Hello K
var oddNumbers = from n in numbers
where n % 2 == 1
select n;
Console.WriteLine(string.Join(", ", oddNumbers));
var oddNumbersLE = numbers.
ToDictionary<>
ToList<>
ToLookup<>
ToString
TrimExcess
TrueForAll
Union<>
Where<>
Zip<>
(rozszerzenie) IEnumerable<int> IEnumerable<int>.Where<int>(Func<int
How are we going to do the same thing with lambda. We are going to write "numbers" and press dot and look at how many methods we have here. Some of these methods are available only because we are using the LinQ namespace. So we're going to use the where method. It's also an extension method. And we're going to talk about extension methods very soon.



We're simply going to create a new lambda expression. We're going to say: WHERE N the lambda operator, N divided by two has a remainder of 1. All right. What is the result of our work? It is working. So, how does it work? The N is the iterating variable; just like with the LinQ query; just like with the ForEach loop. And it is going to iterate on the numbers array. So whatever we have as a collection – a lambda expression will work on it. On the left side of the lambda operator, we have iterator for that collection. It works exactly the same way as in LinQ query in the example. It takes each of the numbers using the variable and it checks if it matches our conditions. And if it's true it's adding it to our new collection. So we reduce the code even more. With the ForEach loop, we do it in seven-eight rows. With LINQ we do it in three rows. And with lambda, we do it in a single line of code.


LambdaExamples - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
Rozpocznij
EntryPoint.cs
LambdaExamples
LambdaExamples.EntryPoint
Main()
Eksplorator serwera
Przybornik
using System;
using System.Collections.Generic;
using System.Linq;
namespace LambdaExamples
{
public class EntryPoint
{
static void Main()
{
string[] catNames = { "Lucky", "Bella", "Luna", "Oreo", "Simba", "Toby", "Loki", "Oscar" };
List<int> numbers = new List<int>() { 5, 6, 3, 2, 1, 5, 6, 7, 8, 4, 234, 54, 14, 653, 3, 4, 5, 6, 7 };
object[] mix = { 1, "string", 'd', new List<int>() { 1, 2, 3, 4 }, new List<int>() { 5, 2, 3, 4 }, "dd", 's', "Hello K
var oddNumbers = from n in numbers
where n % 2 == 1
select n;
Console.WriteLine(string.Join(", ", oddNumbers));
// Extract odd numbers with Lambda
var oddNumbersLE = numbers.Where(n => (n % 2) == 1);
Console.WriteLine(string.Join(", ", oddNumbersLE));
// Extract odd numbers and convert to list
List<int> oddNumbersList = numbers.Where(n => (n % 2) == 1).ToList();
Console.WriteLine(string.Join(", ", oddNumbersList));
}
}
}
C:\Windows\system32\cmd.exe
5, 3, 1, 5, 7, 653, 3, 5, 7
5, 3, 1, 5, 7, 653, 3, 5, 7
Next problem is that the result of the expression is a collection but not a list. We can convert it to a list by writing the type of collection and converting the result of the expression. All we have to do is simply write ToList. And if you do this it's still going to work and you're going to have the numbers in the list.



The next tricks with lambdas that can be shown is for example that we have our cat names. And we want to check what is the average length of a cat name - how many characters is the average length of a cat name. We created a new variables. So it is equal to catNames dot average. And then we need to specify exactly what we want to take. The average of what value we want to take. We want the average of the length of the names. So, variable cat, a lambda operator and we want to iterate over the lengths of the names so get that length. And we're going to print it on the console. So the average length of a cat name from our array of cat names is four point five. The same way we can find out what is the minimum length of cat names, maximum length and a sum of all characters in known cat names.

LambdaExamples - Microsoft Visual Studio

Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc

Debug | Any CPU | Rozpocznij

EntryPoint.cs

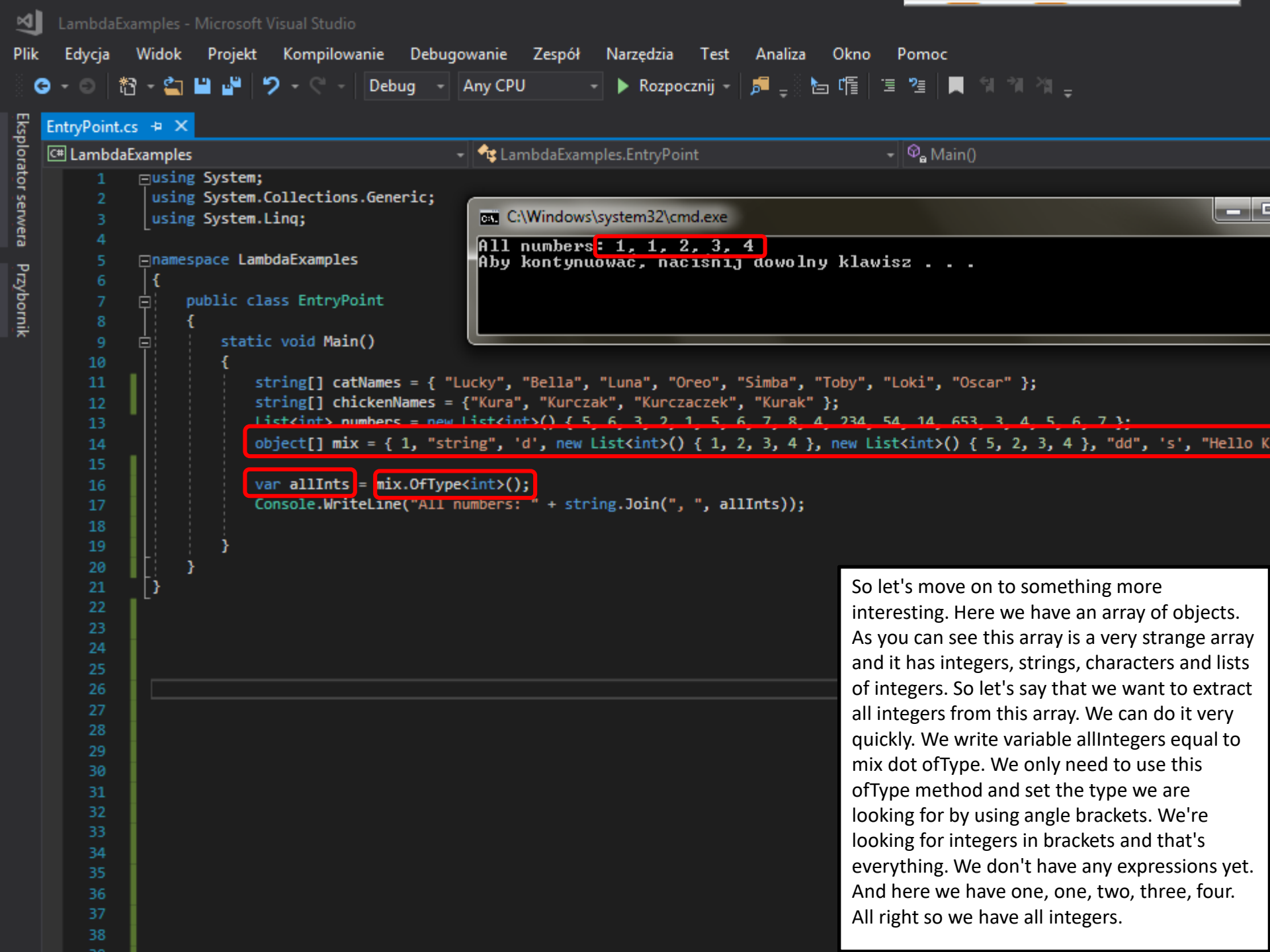
LambdaExamples | LambdaExamples.EntryPoint | Main()

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

namespace LambdaExamples
{
    public class EntryPoint
    {
        static void Main()
        {
            string[] catNames = { "Lucky", "Bella", "Luna", "Oreo", "Simba", "Toby", "Loki", "Oscar" };
            List<int> numbers = new List<int>() { 5, 6, 3, 2, 1, 5, 6, 7, 8, 4, 234, 54, 14, 653, 3, 4, 5, 6, 7 };
            object[] mix = { 1, "string", 'd', new List<int>() { 1, 2, 3, 4 }, new List<int>() { 5, 2, 3, 4 }, "dd", 's', "Hello K

            double average = catNames.Average(cat => cat.Length);
            Console.WriteLine("Average " + average);

            double minCatNameLength = catNames.Min(cat => cat.Length);
            Console.WriteLine("Min     " + minCatNameLength);

            double maxCatNameLength = catNames.Max(cat => cat.Length);
            Console.WriteLine("Max     " + maxCatNameLength);

            double sumCatNameLength = catNames.Sum(cat => cat.Length);
            Console.WriteLine("Sum     " + sumCatNameLength);

        }
    }
}
```

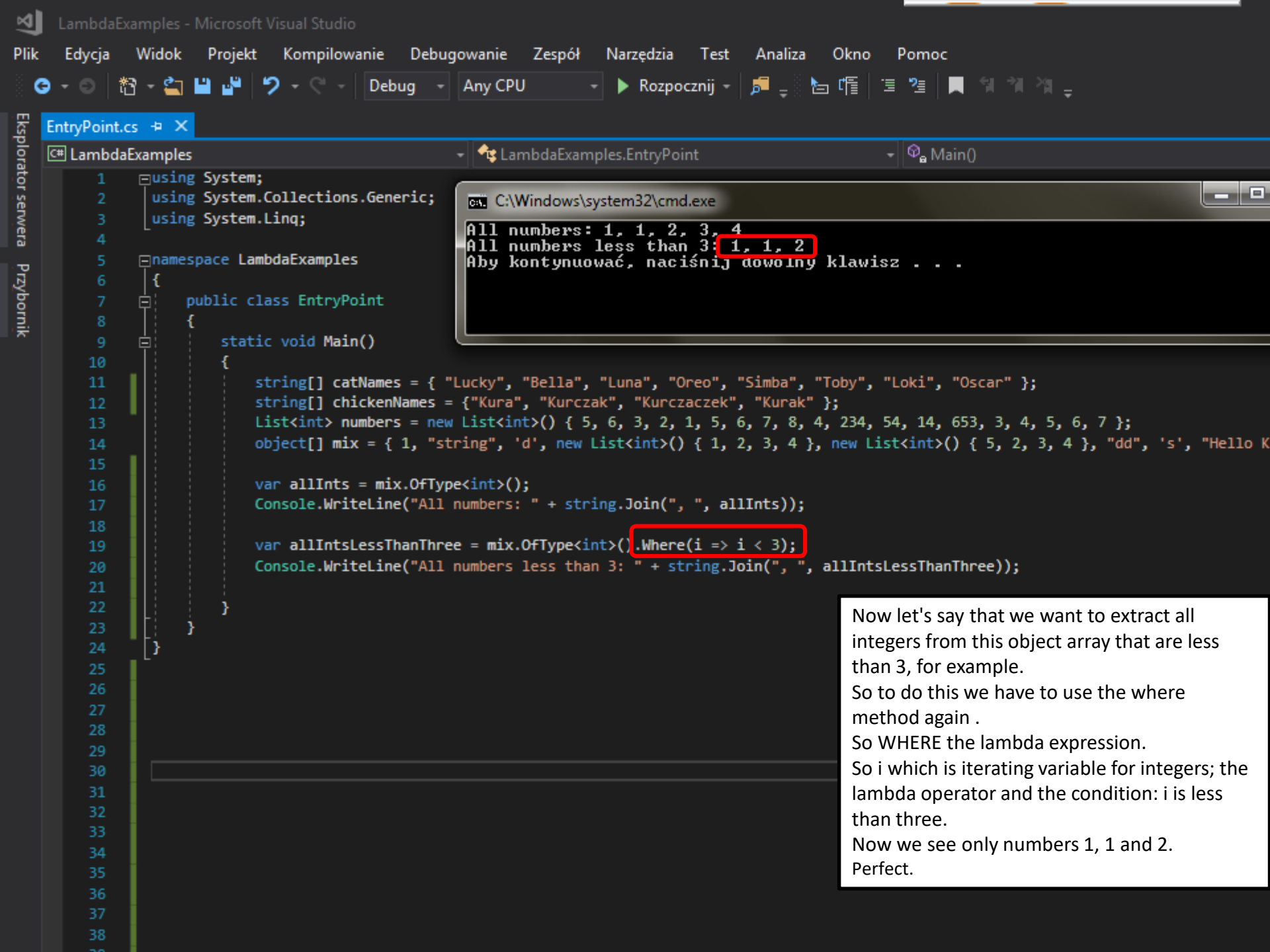
C:\Windows\system32\cmd.exe

```
Average 4.5
Min     4
Max     5
Sum     36
Aby kontynuować, naciśnij dowolny klawisz . . .
```

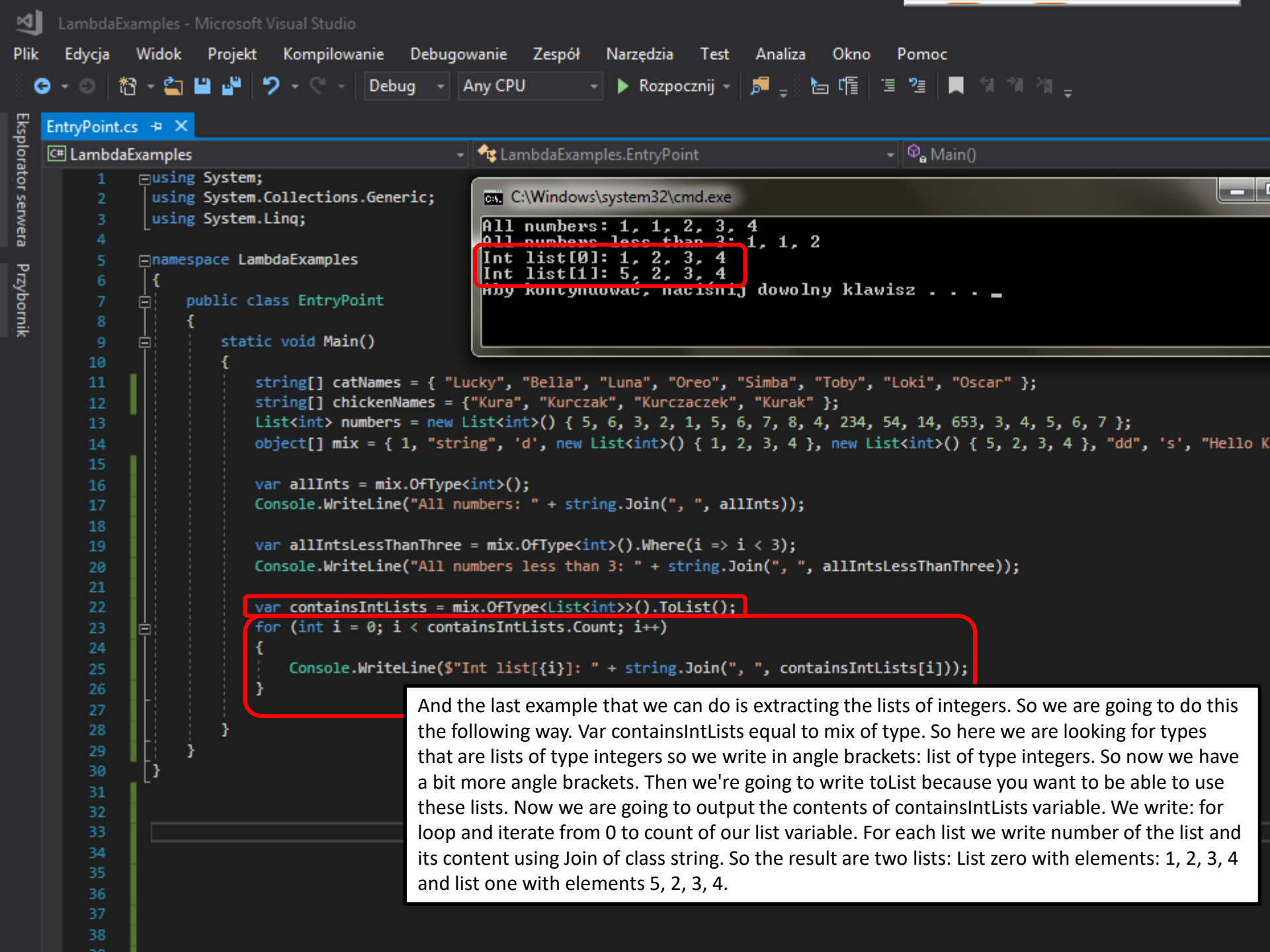
So minCatLength equals catNames dot min open bracket and lambda expression: cat lambda operator and lambda dot length – because we still working with the lengths of names.
And so maxCatLength and sum of the cat names.
And we are writing it on the Console, and it is the result: min is 4 letters, max is 5 letters and sum of every cat names is 36 characters.
Of course here you can use some more complex formulas if you need too.



So let's move on to something more interesting. Here we have an array of objects. As you can see this array is a very strange array and it has integers, strings, characters and lists of integers. So let's say that we want to extract all integers from this array. We can do it very quickly. We write variable allIntegers equal to mix dot ofType. We only need to use this ofType method and set the type we are looking for by using angle brackets. We're looking for integers in brackets and that's everything. We don't have any expressions yet. And here we have one, one, two, three, four. All right so we have all integers.



Now let's say that we want to extract all integers from this object array that are less than 3, for example.
So to do this we have to use the where method again .
So WHERE the lambda expression.
So i which is iterating variable for integers; the lambda operator and the condition: i is less than three.
Now we see only numbers 1, 1 and 2.
Perfect.

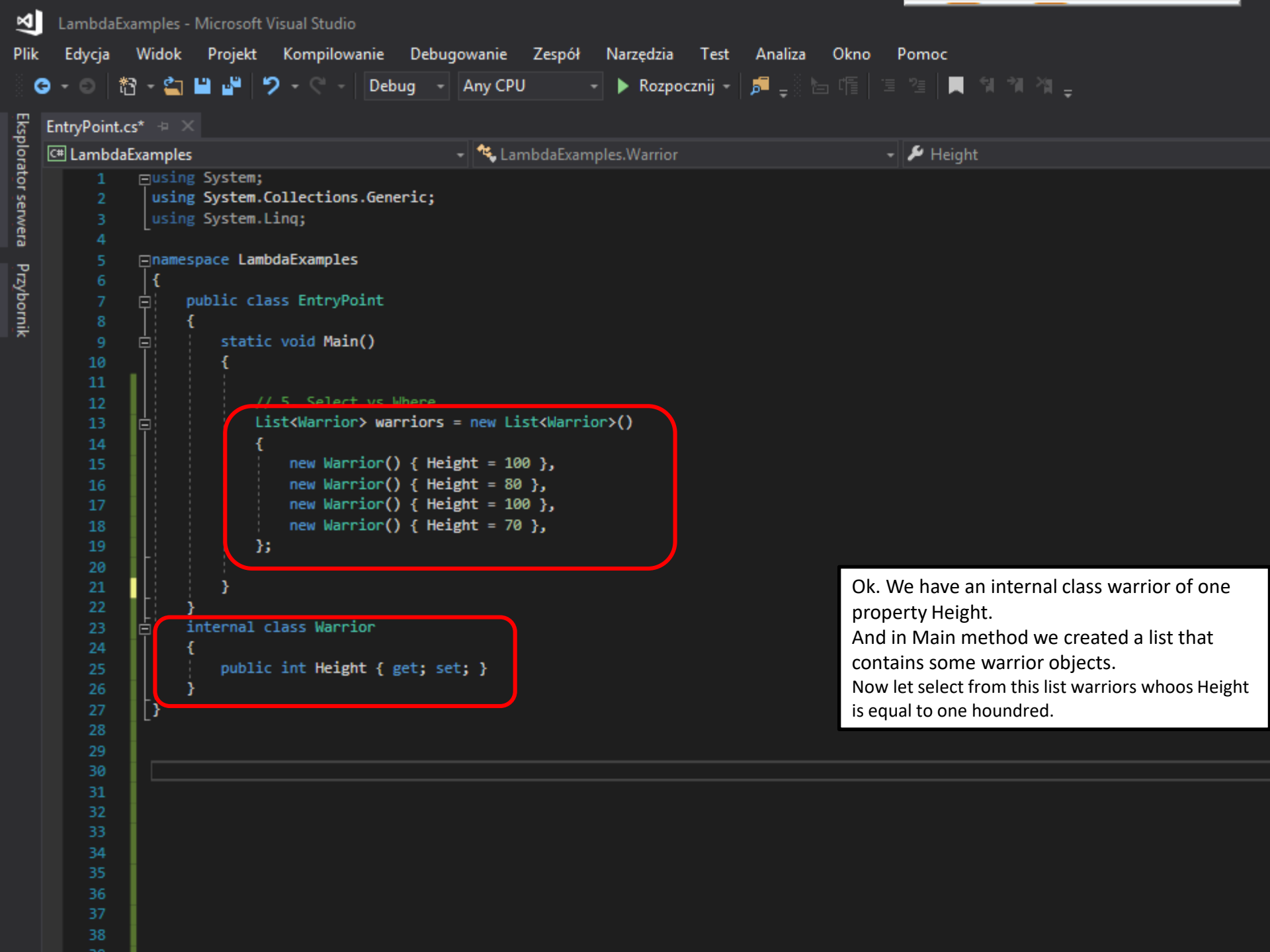


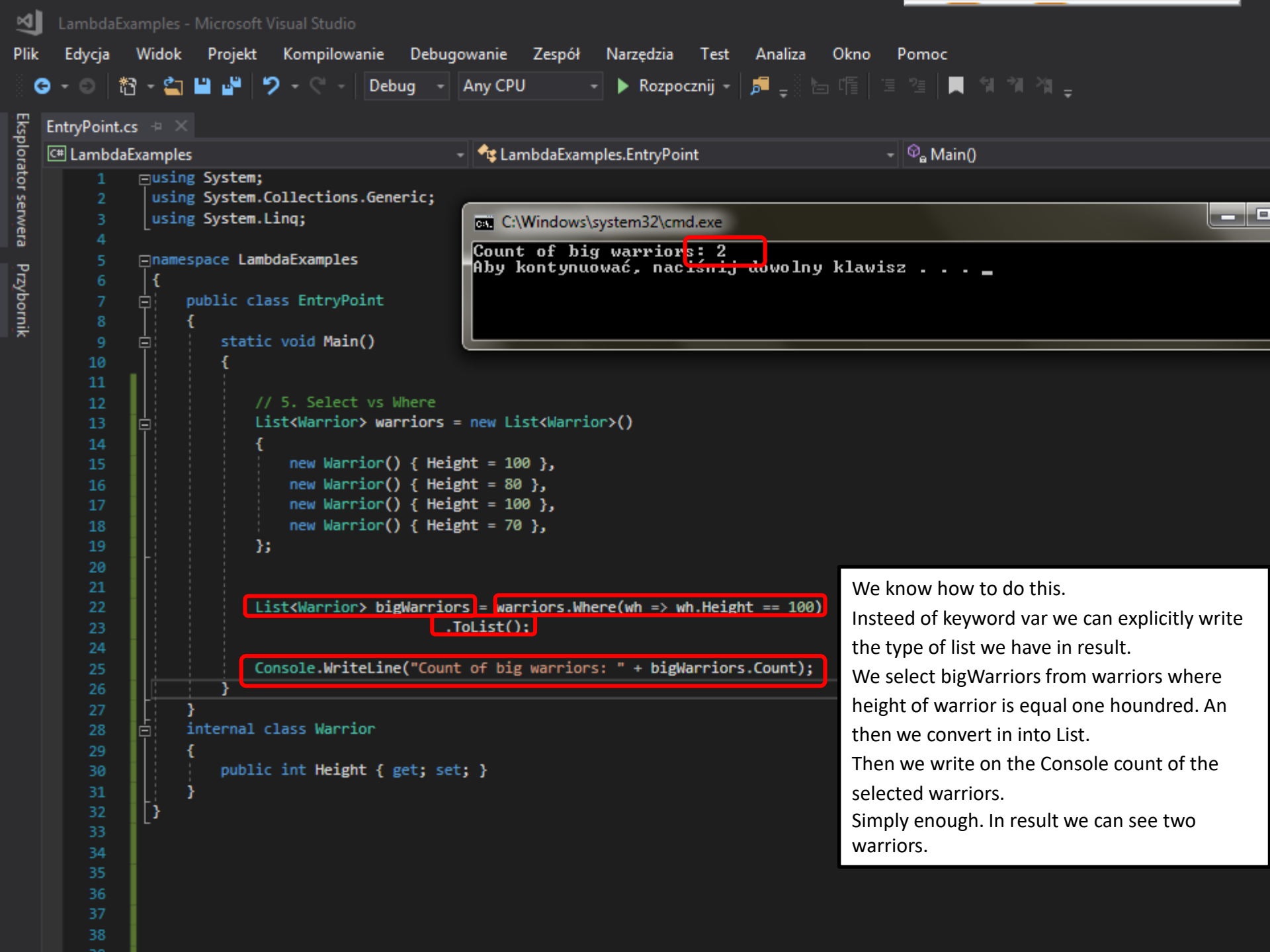
And the last example that we can do is extracting the lists of integers. So we are going to do this the following way. Var containsIntLists equal to mix of type. So here we are looking for types that are lists of type integers so we write in angle brackets: list of type integers. So now we have a bit more angle brackets. Then we're going to write toList because you want to be able to use these lists. Now we are going to output the contents of containsIntLists variable. We write: for loop and iterate from 0 to count of our list variable. For each list we write number of the list and its content using Join of class string. So the result are two lists: List zero with elements: 1, 2, 3, 4 and list one with elements 5, 2, 3, 4.

Select method

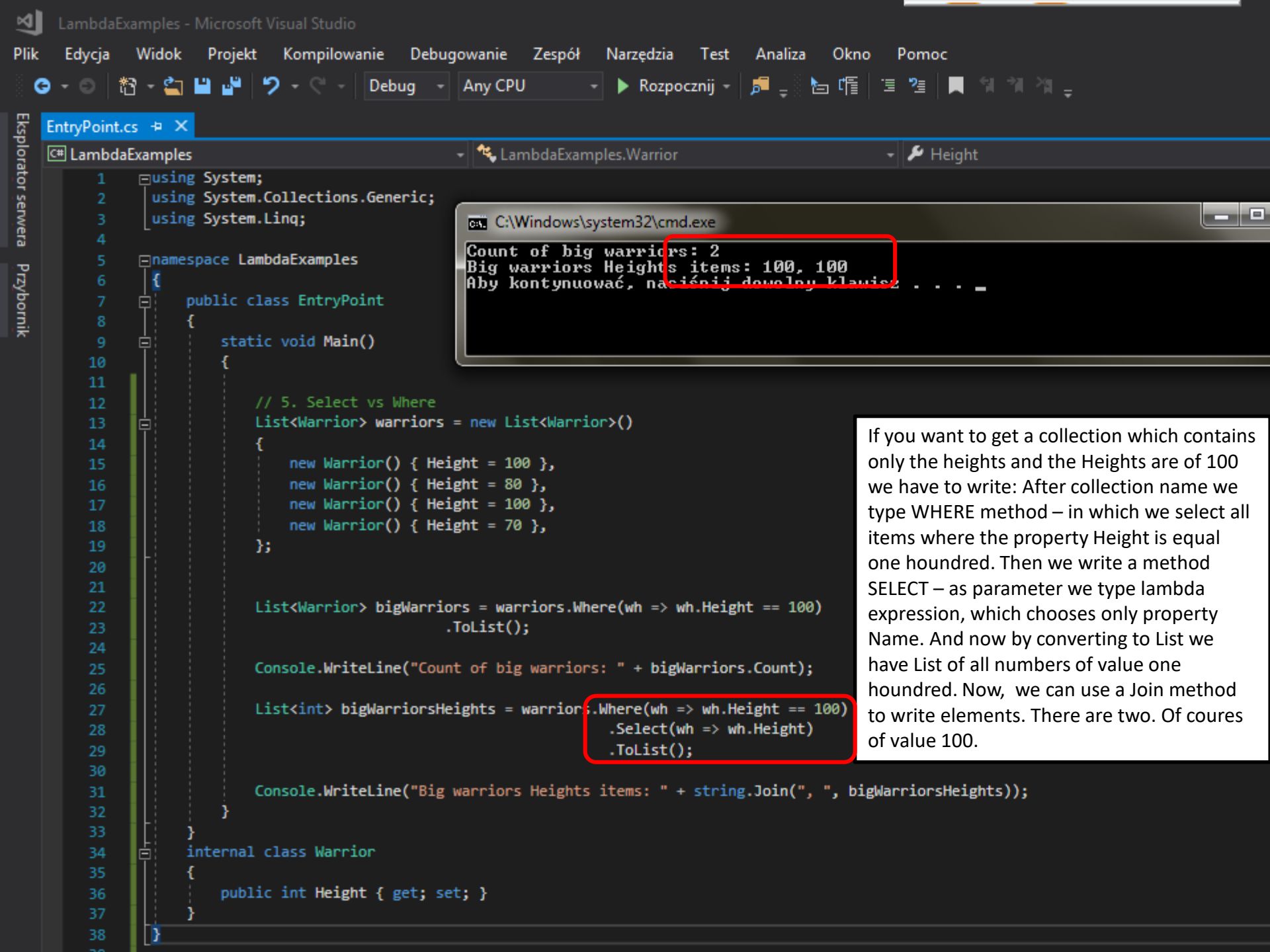
Difference between *where* and *select* methods

Now we're going to talk about the difference between select and where methods.
So lets show the difference with a simple example.


LambdaExamples - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
Rozpocznij
EntryPoint.cs*
LambdaExamples
LambdaExamples.Warrior
Height
Eksplorator serwera
Przybornik
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4
5 namespace LambdaExamples
6 {
7 public class EntryPoint
8 {
9 static void Main()
10 {
11
12 // 5. Select vs Where
13 List<Warrior> warriors = new List<Warrior>()
14 {
15 new Warrior() { Height = 100 },
16 new Warrior() { Height = 80 },
17 new Warrior() { Height = 100 },
18 new Warrior() { Height = 70 },
19 };
20
21 }
22 }
23 internal class Warrior
24 {
25 public int Height { get; set; }
26 }
27 }
28
29
30
31
32
33
34
35
36
37
38
Ok. We have an internal class warrior of one property Height.
And in Main method we created a list that contains some warrior objects.
Now let select from this list warriors whoos Height is equal to one houndred.

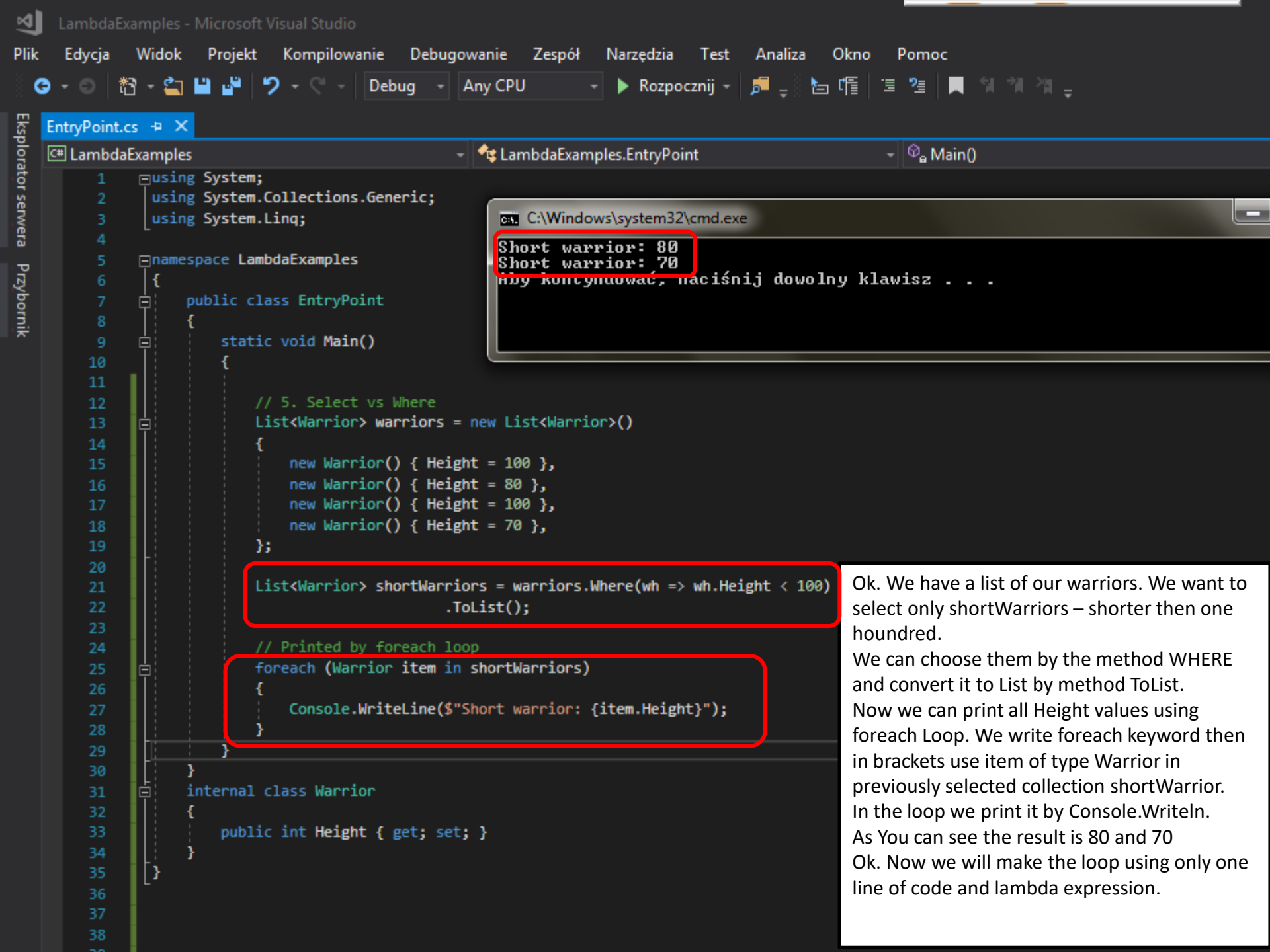


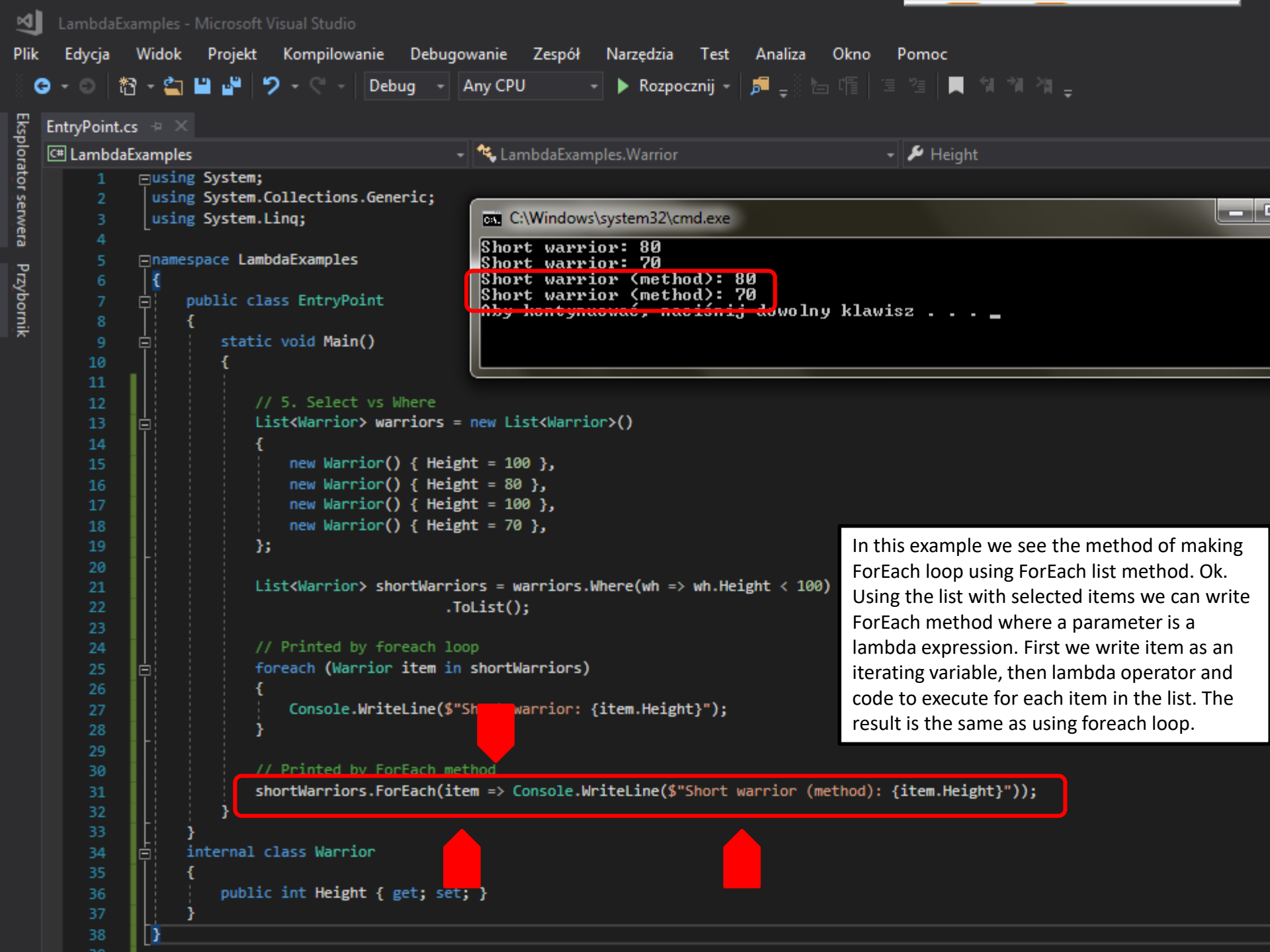
We know how to do this.
Insteed of keyword var we can explicitly write the type of list we have in result.
We select bigWarriors from warriors where height of warrior is equal one houndred. An then we convert in into List.
Then we write on the Console count of the selected warriors.
Simply enough. In result we can see two warriors.

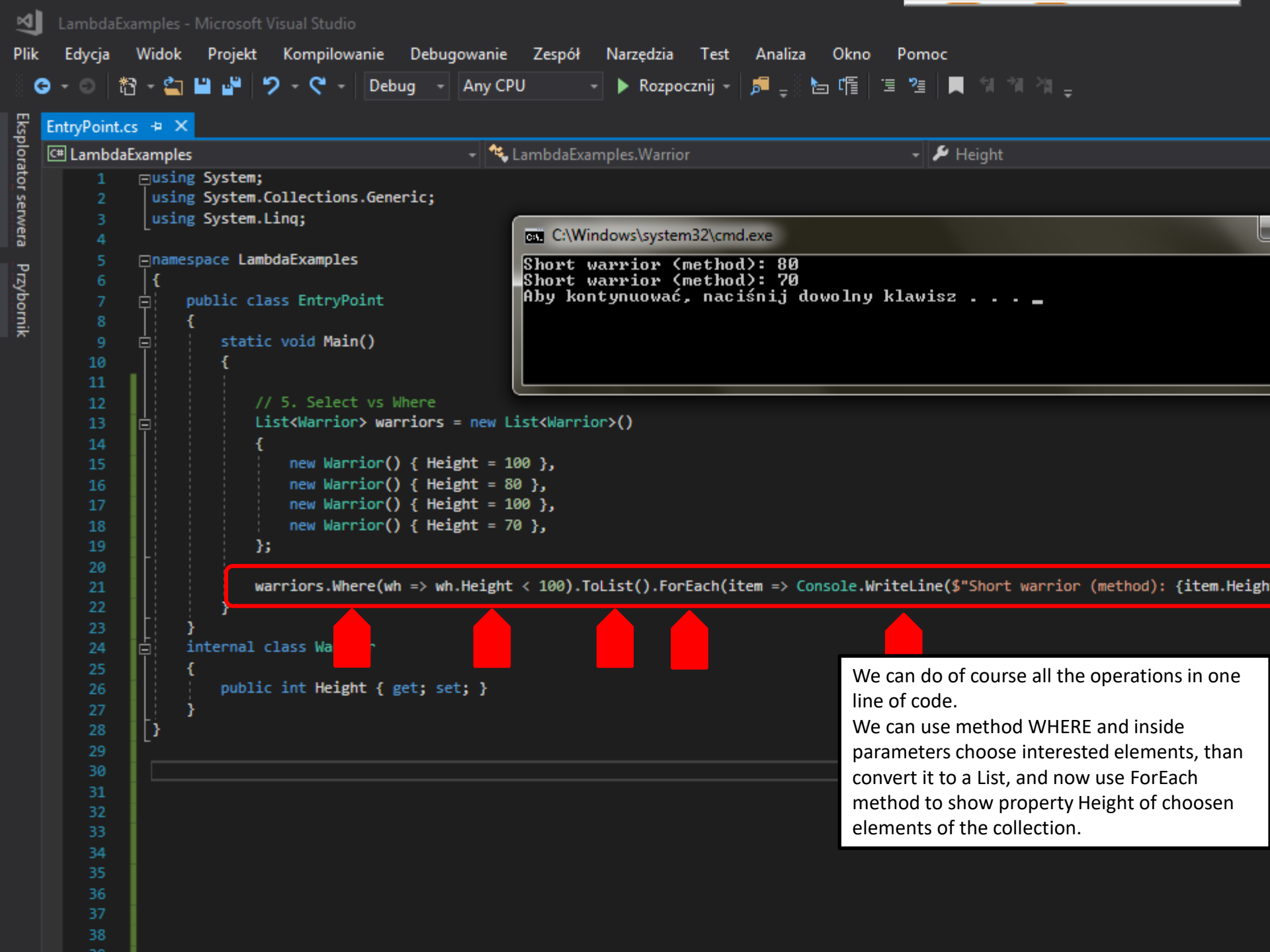

LambdaExamples - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
Rozpocznij
EntryPoint.cs
LambdaExamples
LambdaExamples.Warrior
Height
using System;
using System.Collections.Generic;
using System.Linq;
namespace LambdaExamples
{
public class EntryPoint
{
static void Main()
{
// 5. Select vs Where
List<Warrior> warriors = new List<Warrior>()
{
new Warrior() { Height = 100 },
new Warrior() { Height = 80 },
new Warrior() { Height = 100 },
new Warrior() { Height = 70 },
};
List<Warrior> bigWarriors = warriors.Where(wh => wh.Height == 100)
.ToList();
Console.WriteLine("Count of big warriors: " + bigWarriors.Count);
List<int> bigWarriorsHeights = warriors.Where(wh => wh.Height == 100)
.Select(wh => wh.Height)
.ToList();
Console.WriteLine("Big warriors Heights items: " + string.Join(", ", bigWarriorsHeights));
}
}
internal class Warrior
{
public int Height { get; set; }
}
}
C:\Windows\system32\cmd.exe
Count of big warriors: 2
Big warriors Heights items: 100, 100
Aby kontynuować, naciśnij dowolny klawisz . . .
If you want to get a collection which contains only the heights and the Heights are of 100 we have to write: After collection name we type WHERE method – in which we select all items where the property Height is equal one houndred. Then we write a method SELECT – as parameter we type lambda expression, which chooses only property Name. And now by converting to List we have List of all numbers of value one houndred. Now, we can use a Join method to write elements. There are two. Of coures of value 100.

ForEach method

Now it will be shown how we can use ForEach method inside collections instead of traditional foreach loop.


LambdaExamples - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
Rozpocznij
EntryPoint.cs
LambdaExamples
LambdaExamples.EntryPoint
Main()
Eksplorator serwera
Przybornik
using System;
using System.Collections.Generic;
using System.Linq;
namespace LambdaExamples
{
public class EntryPoint
{
static void Main()
{
// 5. Select vs Where
List<Warrior> warriors = new List<Warrior>()
{
new Warrior() { Height = 100 },
new Warrior() { Height = 80 },
new Warrior() { Height = 100 },
new Warrior() { Height = 70 },
};
List<Warrior> shortWarriors = warriors.Where(wh => wh.Height < 100)
.ToList();
// Printed by foreach loop
foreach (Warrior item in shortWarriors)
{
Console.WriteLine($"Short warrior: {item.Height}");
}
}
}
internal class Warrior
{
public int Height { get; set; }
}
}
C:\Windows\system32\cmd.exe
Short warrior: 80
Short warrior: 70
Aby kontynuować, naciśnij dowolny klawisz . . .
Ok. We have a list of our warriors. We want to select only shortWarriors – shorter then one houndred.
We can choose them by the method WHERE and convert it to List by method ToList.
Now we can print all Height values using foreach Loop. We write foreach keyword then in brackets use item of type Warrior in previously selected collection shortWarrior.
In the loop we print it by Console.Writeln.
As You can see the result is 80 and 70
Ok. Now we will make the loop using only one line of code and lambda expression.


LambdaExamples - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
Rozpocznij
EntryPoint.cs
LambdaExamples
LambdaExamples.Warrior
Height
Eksplorator serwera
Przybornik
using System;
using System.Collections.Generic;
using System.Linq;
namespace LambdaExamples
{
public class EntryPoint
{
static void Main()
{
// 5. Select vs Where
List<Warrior> warriors = new List<Warrior>()
{
new Warrior() { Height = 100 },
new Warrior() { Height = 80 },
new Warrior() { Height = 100 },
new Warrior() { Height = 70 },
};
List<Warrior> shortWarriors = warriors.Where(wh => wh.Height < 100)
.ToList();
// Printed by foreach loop
foreach (Warrior item in shortWarriors)
{
Console.WriteLine($"Short warrior: {item.Height}");
}
// Printed by ForEach method
shortWarriors.ForEach(item => Console.WriteLine($"Short warrior (method): {item.Height}"));
}
}
internal class Warrior
{
public int Height { get; set; }
}
}
C:\Windows\system32\cmd.exe
Short warrior: 80
Short warrior: 70
Short warrior (method): 80
Short warrior (method): 70
Aby kontynuować, naciśnij dowolny klawisz . . .
In this example we see the method of making ForEach loop using ForEach list method. Ok. Using the list with selected items we can write ForEach method where a parameter is a lambda expression. First we write item as an iterating variable, then lambda operator and code to execute for each item in the list. The result is the same as using foreach loop.


LambdaExamples - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
Rozpocznij
EntryPoint.cs
LambdaExamples
LambdaExamples.Warrior
Height
Eksplorator serwera
Przybornik
using System;
using System.Collections.Generic;
using System.Linq;
namespace LambdaExamples
{
public class EntryPoint
{
static void Main()
{
// 5. Select vs Where
List<Warrior> warriors = new List<Warrior>()
{
new Warrior() { Height = 100 },
new Warrior() { Height = 80 },
new Warrior() { Height = 100 },
new Warrior() { Height = 70 },
};
warriors.Where(wh => wh.Height < 100).ToList().ForEach(item => Console.WriteLine($"Short warrior (method): {item.Heigh
}
}
internal class Wa
{
public int Height { get; set; }
}
}
C:\Windows\system32\cmd.exe
Short warrior (method): 80
Short warrior (method): 70
Aby kontynuować, naciśnij dowolny klawisz . . .
We can do of course all the operations in one line of code.
We can use method WHERE and inside parameters choose interested elements, than convert it to a List, and now use ForEach method to show property Height of choosen elements of the collection.