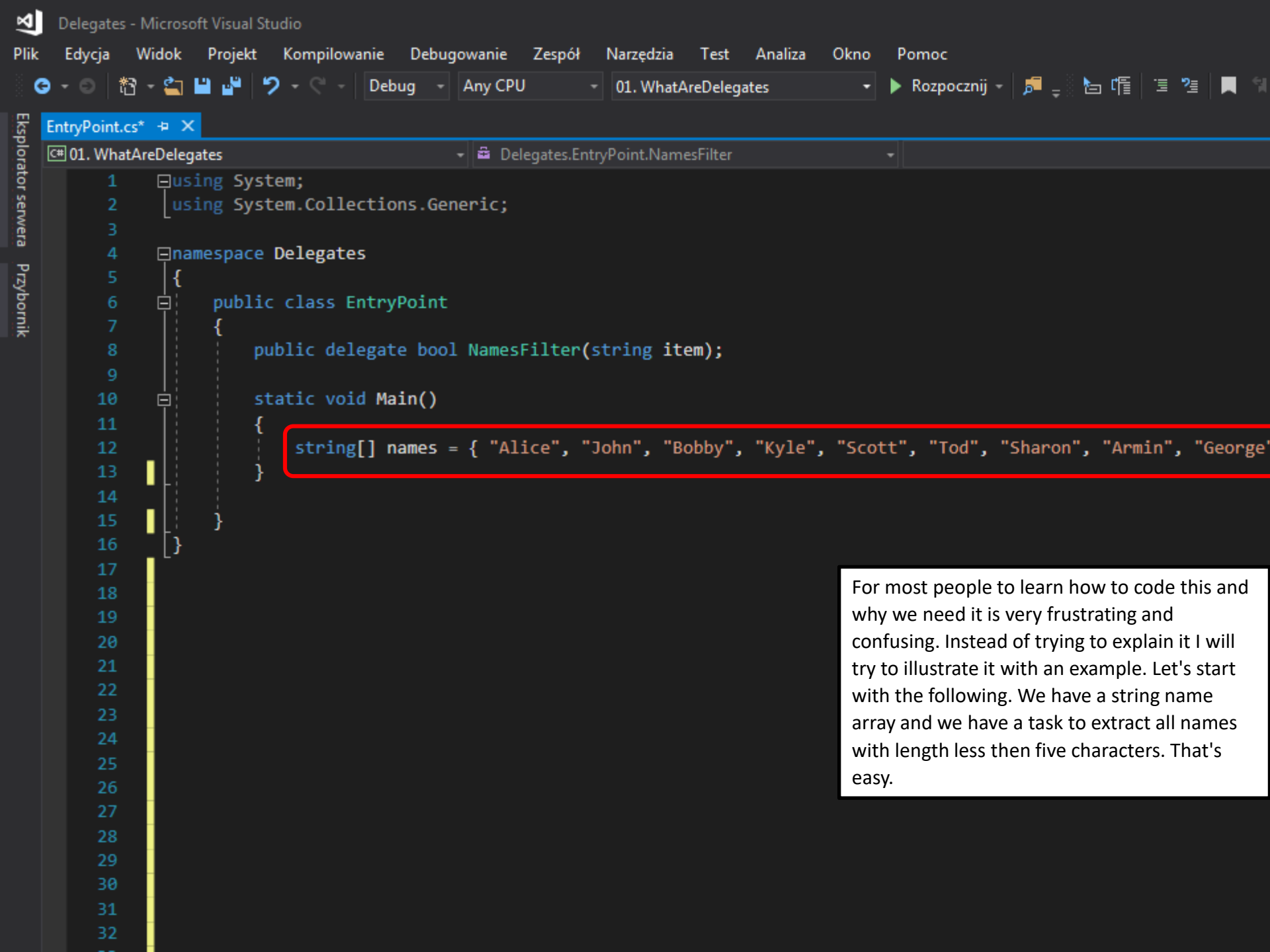
Delegates – methods as variables


Delegates - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
01. WhatAreDelegates
Rozpocznij
EntryPoint.cs*
01. WhatAreDelegates
Delegates.EntryPoint.NamesFilter
Eksplorator serwera
Przybornik
1 using System;
2 using System.Collections.Generic;
3
4 namespace Delegates
5 {
6 public class EntryPoint
7 {
8 public delegate bool NamesFilter(string item);
9
10 static void Main()
11 {
12 string[] names = { "Alice", "John", "Bobby", "Kyle", "Scott", "Tod", "Sharon", "Armin", "George"
13 }
14
15 }
16 }
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
For most people to learn how to code this and why we need it is very frustrating and confusing. Instead of trying to explain it I will try to illustrate it with an example. Let's start with the following. We have a string name array and we have a task to extract all names with length less then five characters. That's easy.

Delegates - Microsoft Visual Studio

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

Debug | Any CPU | 01. WhatAreDelegates | Rozpocznij

EntryPoint.cs

01. WhatAreDelegates | Delegates.EntryPoint | Main()

```
1  using System;
2  using System.Collections.Generic;
3
4  namespace Delegates
5  {
6      public class EntryPoint
7      {
8
9          static void Main()
10         {
11             string[] names = { "Alice", "John", "Bobby", "Kyle", "Scott", "Tod", "Sharon", "Armin", "George"
12
13             List<string> lessThenFiveChars = new List<string>();
14
15             foreach (var item in names)
16             {
17                 if (item.Length < 5)
18                 {
19                     lessThenFiveChars.Add(item);
20                 }
21             }
22
23             Console.WriteLine("Names: " + string.Join(", ", lessThenFiveChars));
24         }
25
26     }
27 }
28
29
30
31
32
```

C:\Windows\system32\cmd.exe

```
Names: John, Kyle, Tod
Aby kontynuować, naciśnij dowolny klawisz . . .
```

All we have to do is create a loop with an IF condition in it and extract them in a new collection. So let's do it. Let's write: list of string we're going to use, a list of string to store these new names less than 5 chars equals to new list of strings. And we're going to create a for loop for each.

And we are going to check if that length is less than 5. And if it is we are going to add them to the lessThenFiveChars list. Simple enough.
Let's try it. So we get: John, Kyle and Tod.

Delegates - Microsoft Visual Studio

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

Debug | Any CPU | 01. WhatAreDelegates | Rozpocznij

EntryPoint.cs

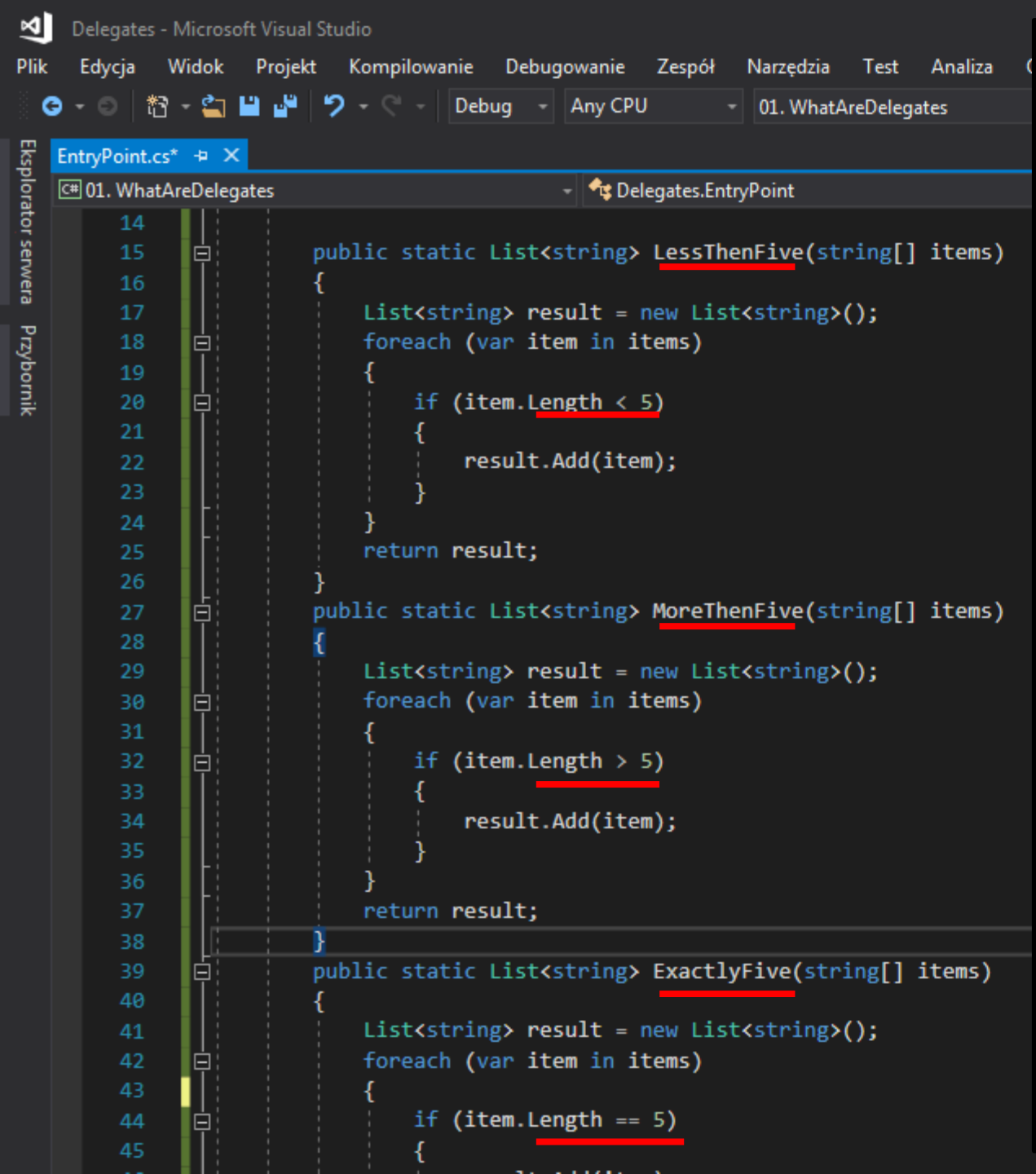
01. WhatAreDelegates | Delegates.EntryPoint | Main()

```
1   using System;
2   using System.Collections.Generic;
3
4   namespace Delegates
5   {
6       public class EntryPoint
7       {
8           static void Main()
9           {
10              string[] names = { "Alice", "John", "Bobby", "Kyle", "Scott", "Tod", "Sharon", "Armin", "George"
11              List<string> lessThenFiveChars = LessThenFive(names);
12              Console.WriteLine("Names: " + string.Join(", ", lessThenFiveChars));
13          }
14
15          public static List<string> LessThenFive(string[] items)
16          {
17              List<string> result = new List<string>();
18              foreach (var item in items)
19              {
20                  if (item.Length < 5)
21                  {
22                      result.Add(item);
23                  }
24              }
25              return result;
26          }
27      }
28  }
29
30
31
32
```

C:\Windows\system32\cmd.exe

```
Names: John, Kyle, Tod
Aby kontynuować, naciśnij dowolny klawisz . . .
```

And here we have a LessThenFive method. So that takes a string array at the input and we are going to return a new list of strings.
So, new list. This is a result and we are going to return them. And we're going to do this less than five. It works the same way.
Now what if we want to do the same thing but with all names that are more than five characters.



Ok that's it. We can create another method to do it.

But what if we want only those that are exactly five characters. That's it.

What if all of those are 10 characters long or longer.

We can see how quickly we start to get many different cases and we can't create infinite number of methods to solve them all.

This is where a delegate comes in.

You can take a piece of code and make it varying. So let's show what we mean by this.

Let's take a look at our method that we created.

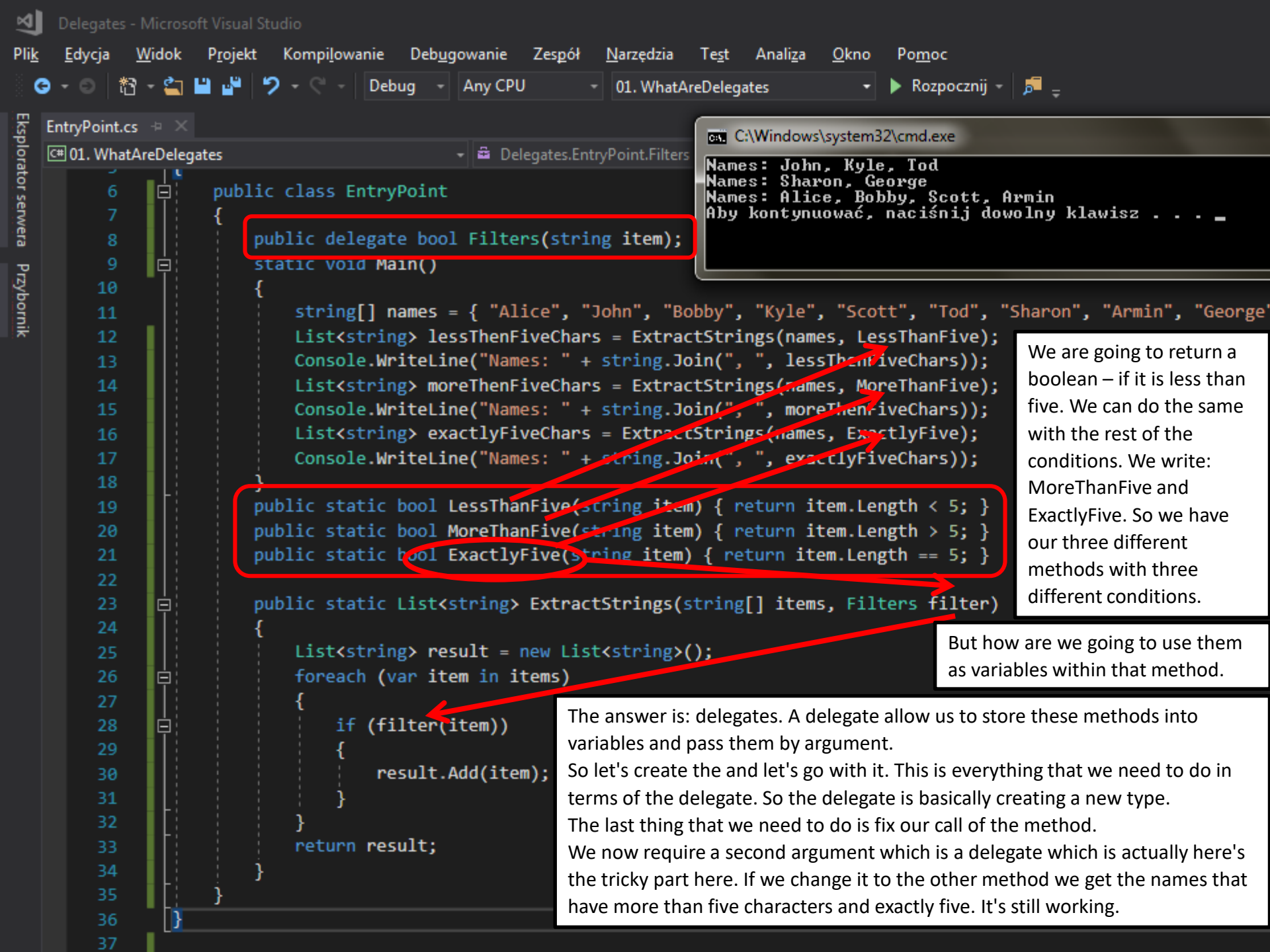
We have only one part of the code that is varying. And this is the condition that filters the names.

If we want to get the names that are more than five characters we simply have to change the sign.

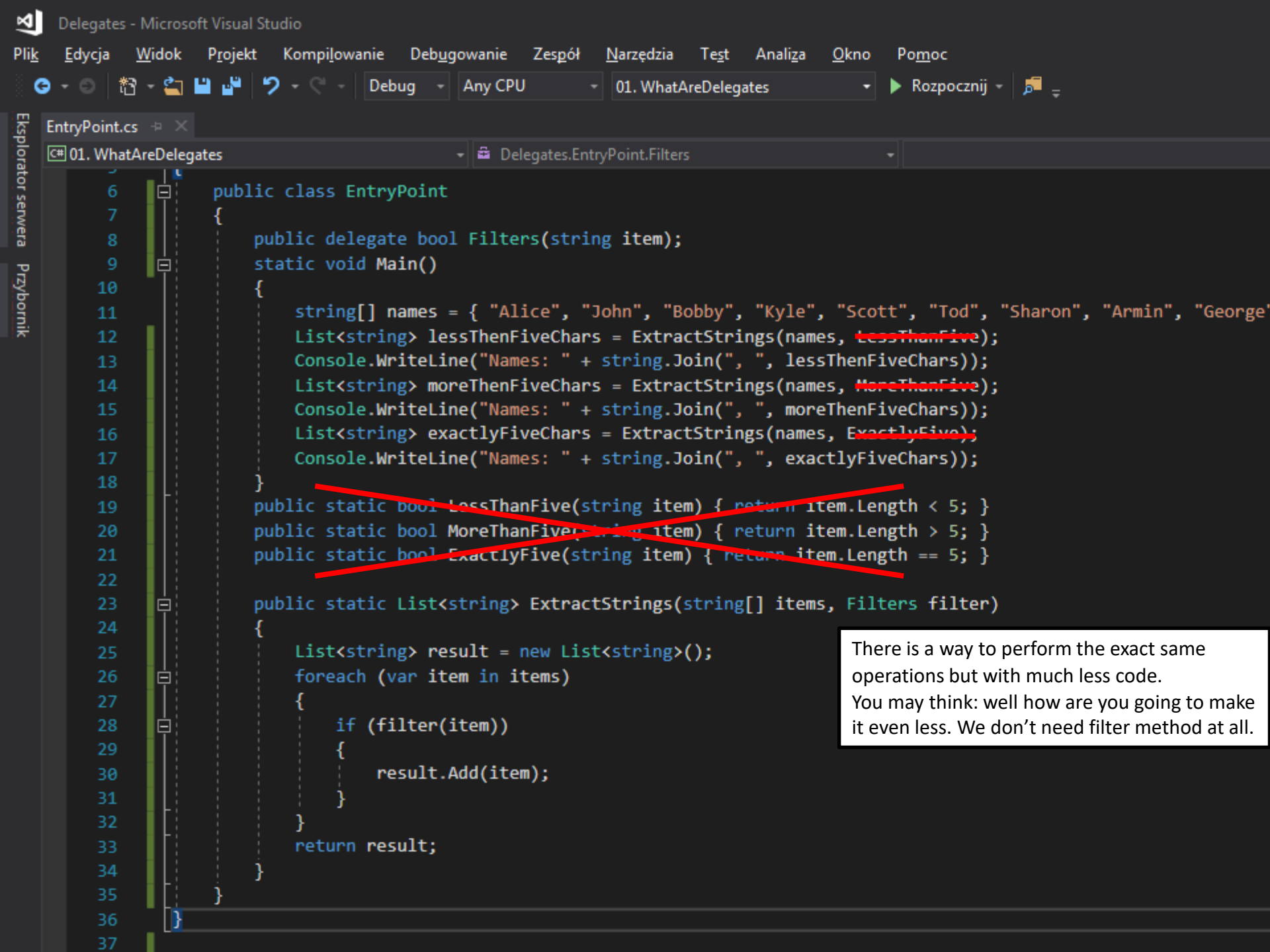
This is the place where we are going to change this piece of code. In order to extract this piece of code we need to think a little. What does it need as an input and what output is it going to return. (Something like when you create a method).

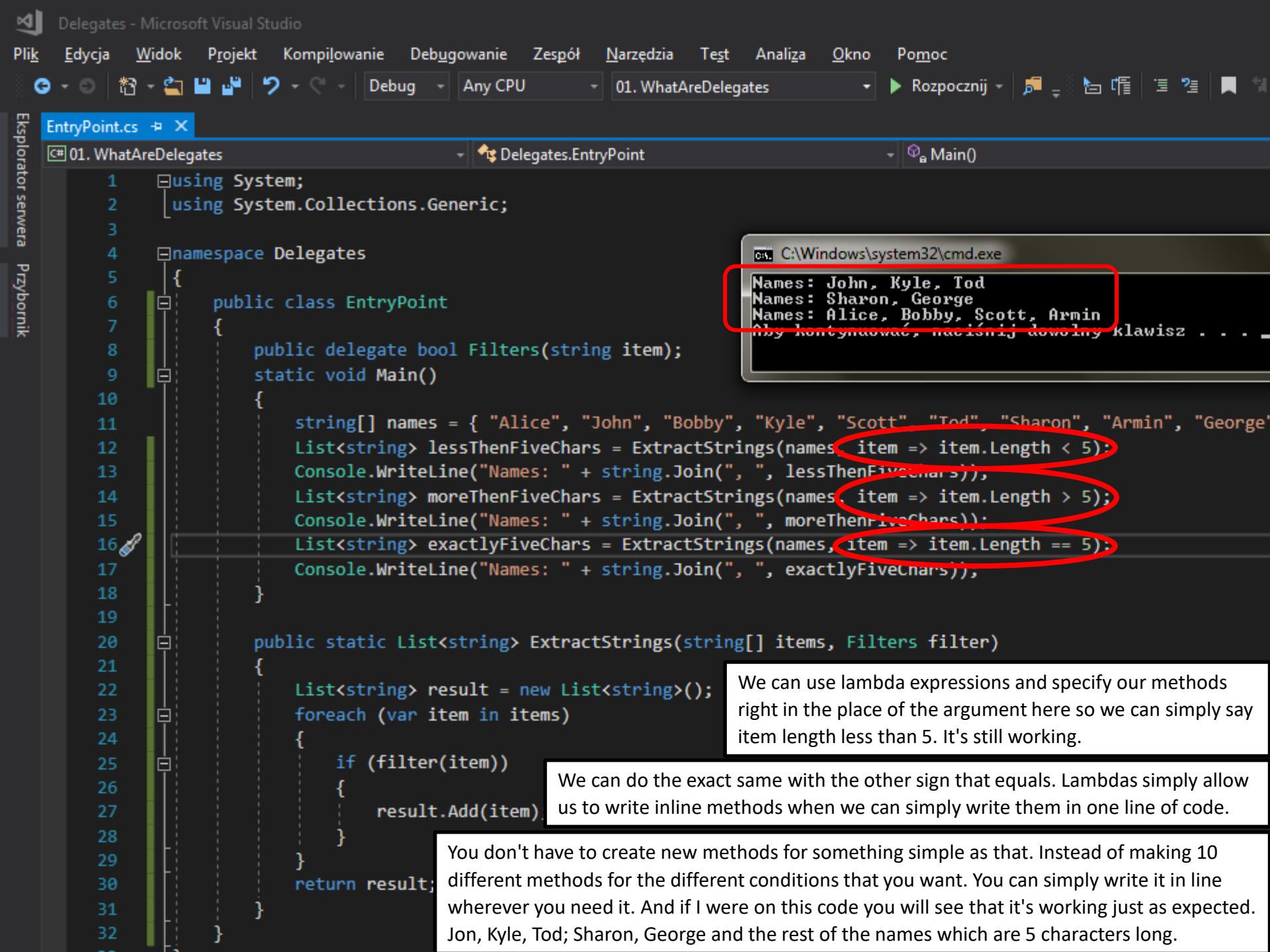
Well, we need a string in the input, and we are going to return a boolean: true or false which will indicate whether the item has exceeds five characters or not.

So we will simply extract our conditions into new methods that only contain the conditions and nothing else.


Delegates - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
01. WhatAreDelegates
Rozpocznij
EntryPoint.cs
01. WhatAreDelegates
Delegates.EntryPoint.Filters
C:\Windows\system32\cmd.exe
Names: John, Kyle, Tod
Names: Sharon, George
Names: Alice, Bobby, Scott, Armin
Aby kontynuować, naciśnij dowolny klawisz . . .
public class EntryPoint
{
public delegate bool Filters(string item);
static void Main()
{
string[] names = { "Alice", "John", "Bobby", "Kyle", "Scott", "Tod", "Sharon", "Armin", "George"
List<string> lessThenFiveChars = ExtractStrings(names, LessThanFive);
Console.WriteLine("Names: " + string.Join(", ", lessThenFiveChars));
List<string> moreThenFiveChars = ExtractStrings(names, MoreThanFive);
Console.WriteLine("Names: " + string.Join(", ", moreThenFiveChars));
List<string> exactlyFiveChars = ExtractStrings(names, ExactlyFive);
Console.WriteLine("Names: " + string.Join(", ", exactlyFiveChars));
}
public static bool LessThanFive(string item) { return item.Length < 5; }
public static bool MoreThanFive(string item) { return item.Length > 5; }
public static bool ExactlyFive(string item) { return item.Length == 5; }
public static List<string> ExtractStrings(string[] items, Filters filter)
{
List<string> result = new List<string>();
foreach (var item in items)
{
if (filter(item))
{
result.Add(item);
}
}
return result;
}
}
We are going to return a boolean – if it is less than five. We can do the same with the rest of the conditions. We write: MoreThanFive and ExactlyFive. So we have our three different methods with three different conditions.
But how are we going to use them as variables within that method.
The answer is: delegates. A delegate allow us to store these methods into variables and pass them by argument.
So let's create the and let's go with it. This is everything that we need to do in terms of the delegate. So the delegate is basically creating a new type.
The last thing that we need to do is fix our call of the method.
We now require a second argument which is a delegate which is actually here's the tricky part here. If we change it to the other method we get the names that have more than five characters and exactly five. It's still working.

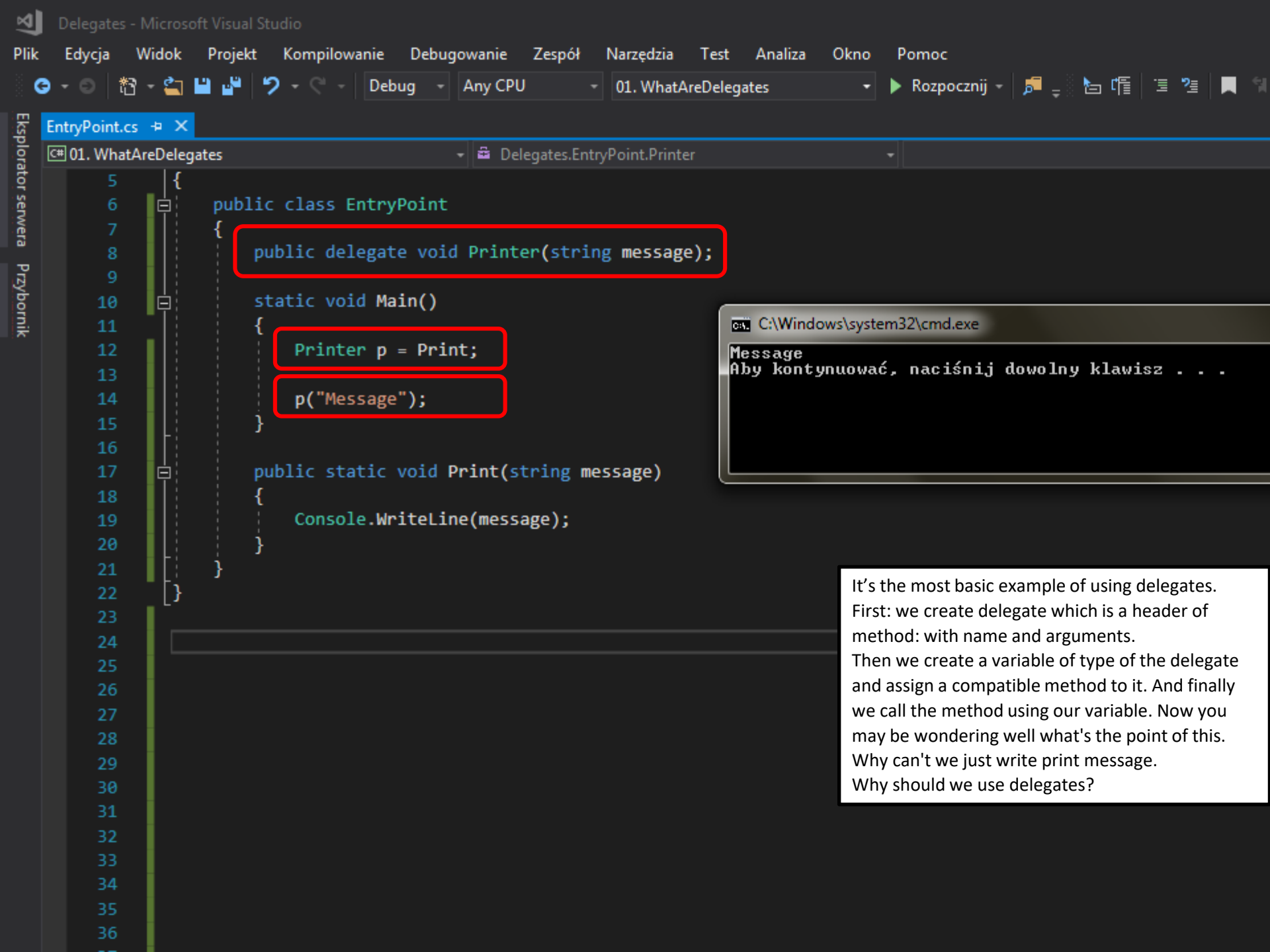
Delegates and lambda expressions

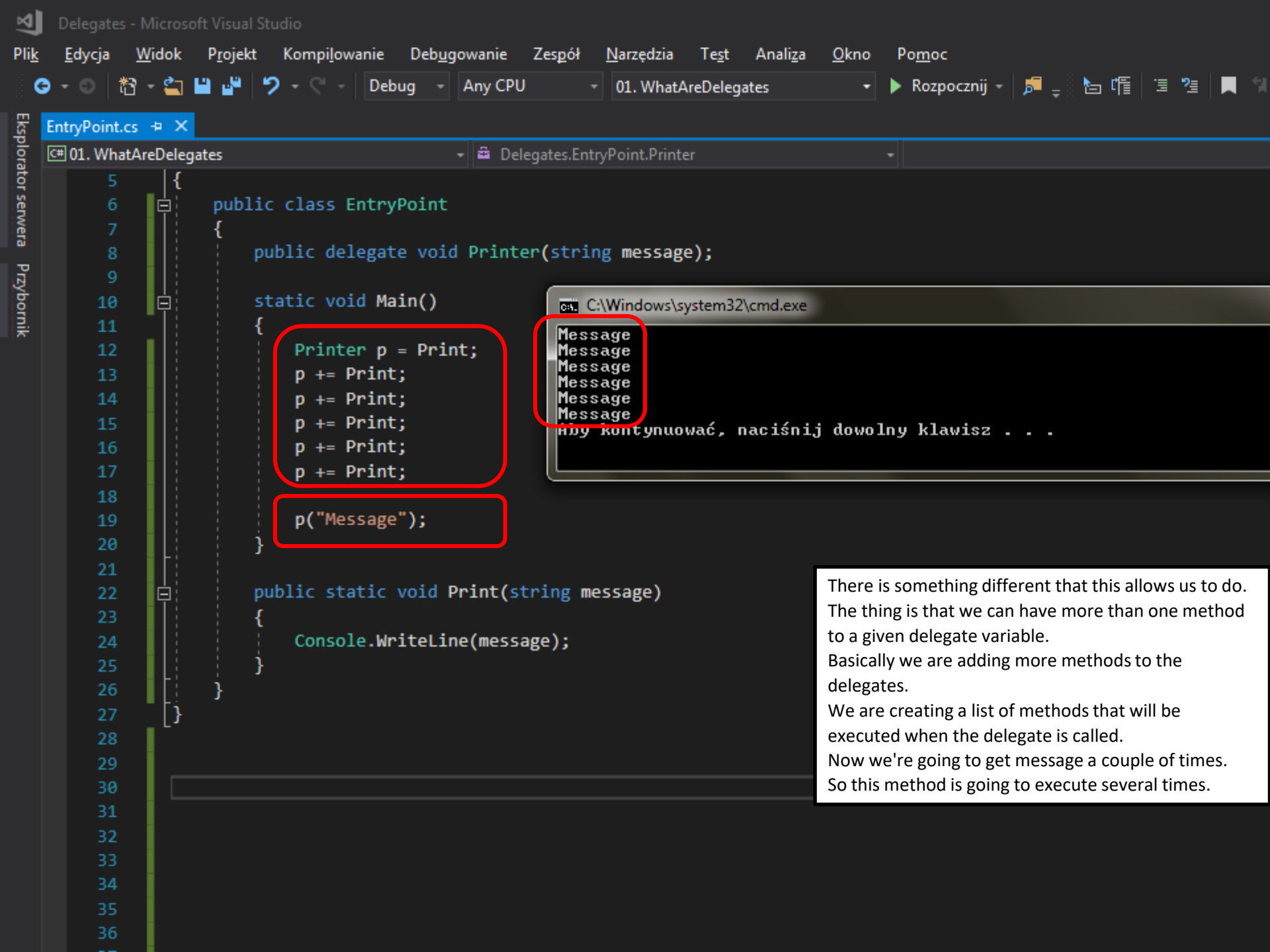

Delegates - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
01. WhatAreDelegates
Rozpocznij
EntryPoint.cs
01. WhatAreDelegates
Delegates.EntryPoint.Filters
Eksplorator serwera
Przybornik
public class EntryPoint
{
public delegate bool Filters(string item);
static void Main()
{
string[] names = { "Alice", "John", "Bobby", "Kyle", "Scott", "Tod", "Sharon", "Armin", "George"
List<string> lessThenFiveChars = ExtractStrings(names,
Console.WriteLine("Names: " + string.Join(", ", lessThenFiveChars));
List<string> moreThenFiveChars = ExtractStrings(names,
Console.WriteLine("Names: " + string.Join(", ", moreThenFiveChars));
List<string> exactlyFiveChars = ExtractStrings(names,
Console.WriteLine("Names: " + string.Join(", ", exactlyFiveChars));
}
public static bool LessThanFive(string item) { return item.Length < 5; }
public static bool MoreThanFive(string item) { return item.Length > 5; }
public static bool ExactlyFive(string item) { return item.Length == 5; }
public static List<string> ExtractStrings(string[] items, Filters filter)
{
List<string> result = new List<string>();
foreach (var item in items)
{
if (filter(item))
{
result.Add(item);
}
}
return result;
}
}
There is a way to perform the exact same operations but with much less code.
You may think: well how are you going to make it even less. We don’t need filter method at all.


Delegates - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
01. WhatAreDelegates
Rozpocznij
EntryPoint.cs
01. WhatAreDelegates
Delegates.EntryPoint
Main()
Eksplorator serwera
Przybornik
using System;
using System.Collections.Generic;
namespace Delegates
{
public class EntryPoint
{
public delegate bool Filters(string item);
static void Main()
{
string[] names = { "Alice", "John", "Bobby", "Kyle", "Scott", "Tod", "Sharon", "Armin", "George"
List<string> lessThenFiveChars = ExtractStrings(names, item => item.Length < 5);
Console.WriteLine("Names: " + string.Join(", ", lessThenFiveChars));
List<string> moreThenFiveChars = ExtractStrings(names, item => item.Length > 5);
Console.WriteLine("Names: " + string.Join(", ", moreThenFiveChars));
List<string> exactlyFiveChars = ExtractStrings(names, item => item.Length == 5);
Console.WriteLine("Names: " + string.Join(", ", exactlyFiveChars));
}
public static List<string> ExtractStrings(string[] items, Filters filter)
{
List<string> result = new List<string>();
foreach (var item in items)
{
if (filter(item))
{
result.Add(item);
}
}
return result;
}
}
C:\Windows\system32\cmd.exe
Names: John, Kyle, Tod
Names: Sharon, George
Names: Alice, Bobby, Scott, Armin
We can use lambda expressions and specify our methods right in the place of the argument here so we can simply say item length less than 5. It's still working.
We can do the exact same with the other sign that equals. Lambdas simply allow us to write inline methods when we can simply write them in one line of code.
You don't have to create new methods for something simple as that. Instead of making 10 different methods for the different conditions that you want. You can simply write it in line wherever you need it. And if I were on this code you will see that it's working just as expected. Jon, Kyle, Tod; Sharon, George and the rest of the names which are 5 characters long.

Delegates chaining with many methods

Let's create one very simple method that is simply going to print something on the console.


Delegates - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
01. WhatAreDelegates
Rozpocznij
EntryPoint.cs
01. WhatAreDelegates
Delegates.EntryPoint.Printer
Eksplorator serwera
Przybornik
{
public class EntryPoint
{
public delegate void Printer(string message);
static void Main()
{
Printer p = Print;
p("Message");
}
public static void Print(string message)
{
Console.WriteLine(message);
}
}
}
C:\Windows\system32\cmd.exe
Message
Aby kontynuować, naciśnij dowolny klawisz . . .
It’s the most basic example of using delegates.
First: we create delegate which is a header of method: with name and arguments.
Then we create a variable of type of the delegate and assign a compatible method to it. And finally we call the method using our variable. Now you may be wondering well what's the point of this.
Why can't we just write print message.
Why should we use delegates?


Delegates - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
01. WhatAreDelegates
Rozpocznij
EntryPoint.cs
01. WhatAreDelegates
Delegates.EntryPoint.Printer
Eksplorator serwera
Przybornik
5 {
6 public class EntryPoint
7 {
8 public delegate void Printer(string message);
9
10 static void Main()
11 {
12 Printer p = Print;
13 p += Print;
14 p += Print;
15 p += Print;
16 p += Print;
17 p += Print;
18
19 p("Message");
20 }
21
22 public static void Print(string message)
23 {
24 Console.WriteLine(message);
25 }
26 }
27 }
28
29
30
31
32
33
34
35
36
C:\Windows\system32\cmd.exe
Message
Message
Message
Message
Message
Message
Aby kontynuować, naciśnij dowolny klawisz . . .
There is something different that this allows us to do.
The thing is that we can have more than one method to a given delegate variable.
Basically we are adding more methods to the delegates.
We are creating a list of methods that will be executed when the delegate is called.
Now we're going to get message a couple of times.
So this method is going to execute several times.

Delegates - Microsoft Visual Studio

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

Debug | Any CPU | 01. WhatAreDelegates | Rozpocznij

EntryPoint.cs

01. WhatAreDelegates | Delegates.EntryPoint.Printer

```
1   using System;
2   using System.Collections.Generic;
3
4   namespace Delegates
5   {
6       public class EntryPoint
7       {
8           public delegate void Printer(string message);
9
10          static void Main()
11          {
12              Printer p = Print;
13              p += Print;
14              p += PrintTwice;
15              p += Print;
16              p += PrintTwice;
17              p += Print;
18
19              p("Message");
20          }
21
22          public static void PrintTwice(string message)
23          {
24              Console.WriteLine(message + " " + 1);
25              Console.WriteLine(message + " " + 2);
26          }
27
28          public static void Print(string message)
29          {
30              Console.WriteLine(message);
31          }
32      }
```

C:\Windows\system32\cmd.exe

```
Message
Message
Message 1
Message 2
Message
Message 1
Message 2
Message
Aby kontynuować, naciśnij dowolny klawisz . . .
```

Of course we can call different methods as well. So if we create a second method to print twice the message we're going to do the exact same thing.

Delegates - Microsoft Visual Studio

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

Debug | Any CPU | 01. WhatAreDelegates | Rozpocznij

EntryPoint.cs

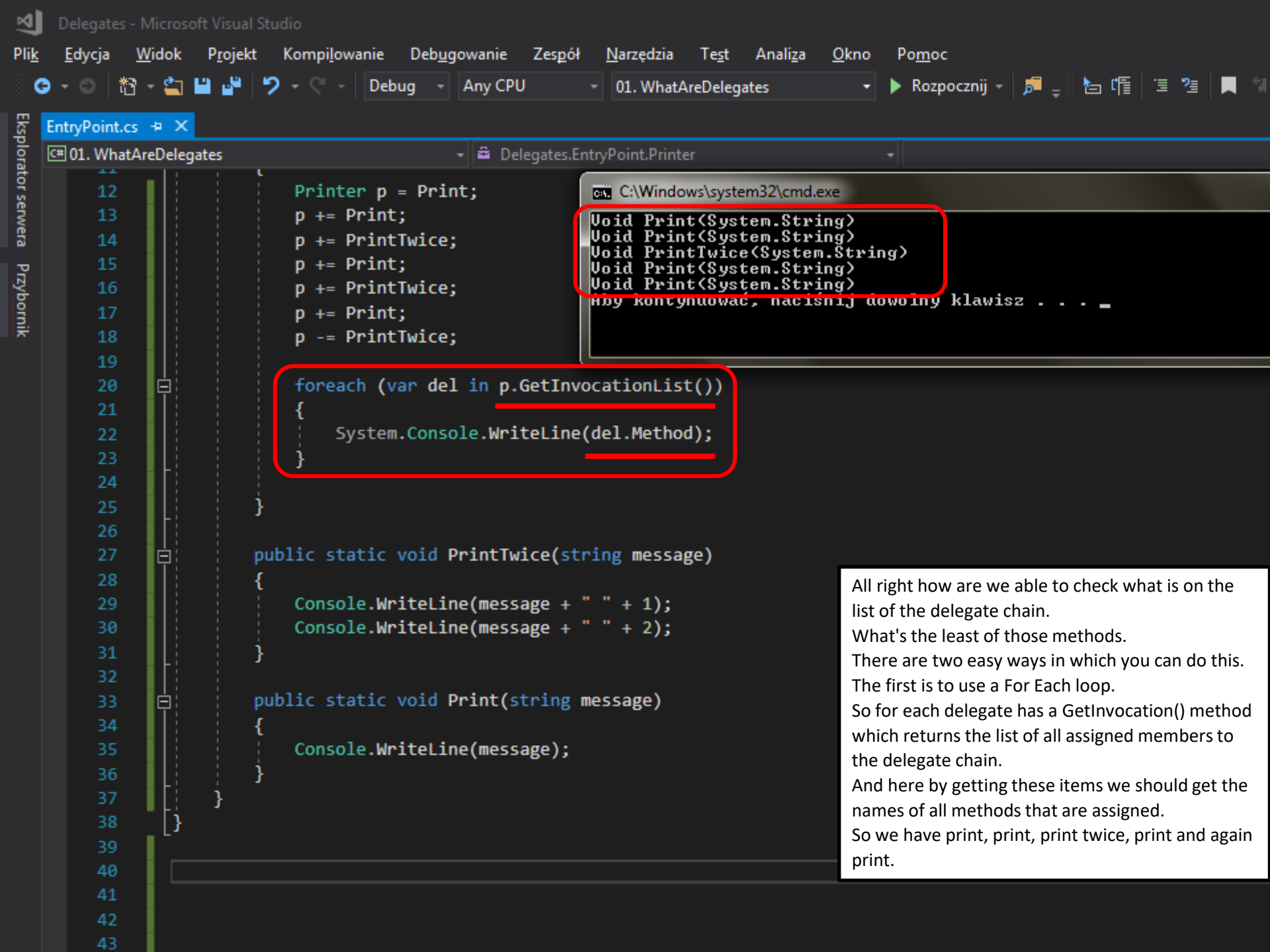
01. WhatAreDelegates | Delegates.EntryPoint | Main()

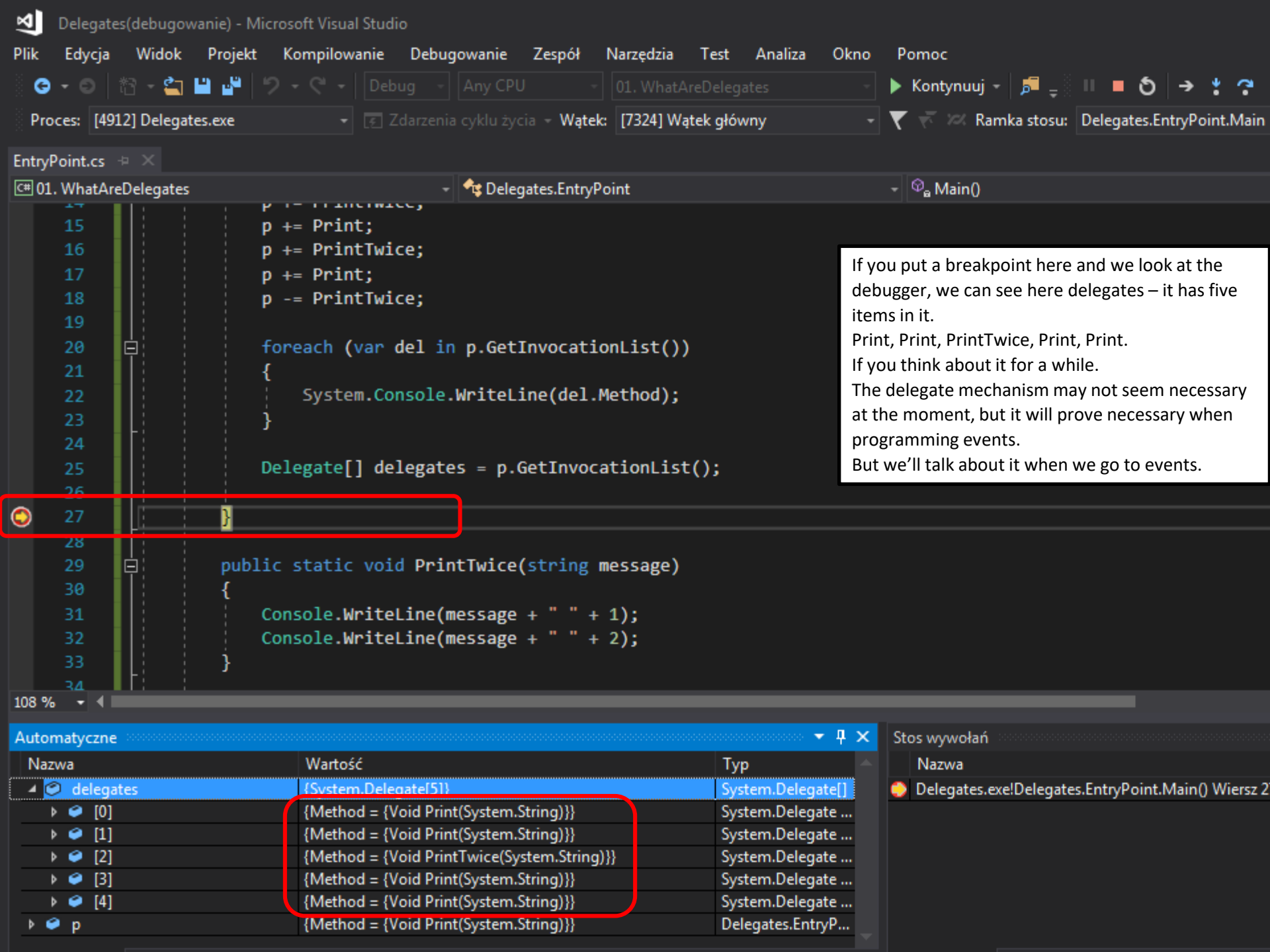
```
1   using System;
2   using System.Collections.Generic;
3
4   namespace Delegates
5   {
6       public class EntryPoint
7       {
8           public delegate void Printer(string message);
9
10          static void Main()
11          {
12              Printer p = Print;
13              p += Print;
14              p += PrintTwice;
15              p += Print;
16              p += PrintTwice;
17              p += Print;
18              p -= PrintTwice;
19
20              p("Message");
21          }
22
23          public static void PrintTwice(string message)
24          {
25              Console.WriteLine(message + " " + 1);
26              Console.WriteLine(message + " " + 2);
27          }
28
29          public static void Print(string message)
30          {
31              Console.WriteLine(message);
32          }
```

C:\Windows\system32\cmd.exe

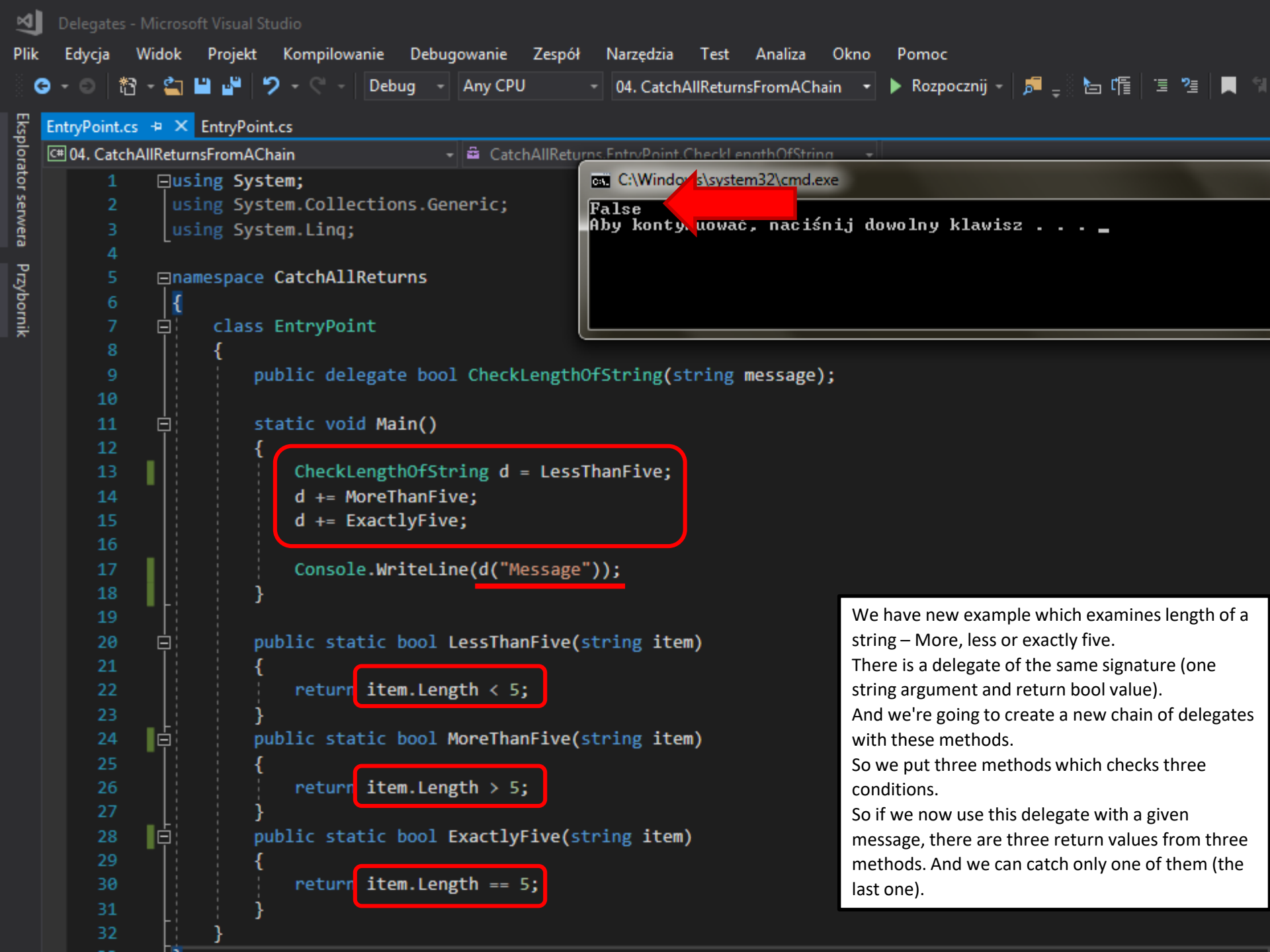
```
Message
Message
Message 1
Message 2
Message
Message
Aby kontynuować, naciśnij dowolny klawisz . . .
```

We can also remove methods from the delegate chain so we can say P minus equals PrintTwice. And if we call p again then you'll get PrintTwice only once.
Now when we subtract methods from the delegate chain we are removing the last occurrence of the delegate.


Delegates - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
01. WhatAreDelegates
Rozpocznij
EntryPoint.cs
01. WhatAreDelegates
Delegates.EntryPoint.Printer
Eksplorator serwera
Przybornik
Printer p = Print;
p += Print;
p += PrintTwice;
p += Print;
p += PrintTwice;
p += Print;
p -= PrintTwice;
foreach (var del in p.GetInvocationList())
{
System.Console.WriteLine(del.Method);
}
}
public static void PrintTwice(string message)
{
Console.WriteLine(message + " " + 1);
Console.WriteLine(message + " " + 2);
}
public static void Print(string message)
{
Console.WriteLine(message);
}
}
}
C:\Windows\system32\cmd.exe
Void Print(System.String)
Void Print(System.String)
Void PrintTwice(System.String)
Void Print(System.String)
Void Print(System.String)
Aby kontynuować, naciśnij dowolny klawisz . . .
All right how are we able to check what is on the list of the delegate chain.
What's the least of those methods.
There are two easy ways in which you can do this.
The first is to use a For Each loop.
So for each delegate has a GetInvocation() method which returns the list of all assigned members to the delegate chain.
And here by getting these items we should get the names of all methods that are assigned.
So we have print, print, print twice, print and again print.


Delegates(debugowanie) - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
01. WhatAreDelegates
Kontynuuj
Proces: [4912] Delegates.exe
Zdarzenia cyklu życia
Wątek: [7324] Wątek główny
Ramka stosu: Delegates.EntryPoint.Main
EntryPoint.cs
01. WhatAreDelegates
Delegates.EntryPoint
Main()
p += Print;
p += PrintTwice;
p += Print;
p -= PrintTwice;
foreach (var del in p.GetInvocationList())
{
System.Console.WriteLine(del.Method);
}
Delegate[] delegates = p.GetInvocationList();
}
public static void PrintTwice(string message)
{
Console.WriteLine(message + " " + 1);
Console.WriteLine(message + " " + 2);
}
If you put a breakpoint here and we look at the debugger, we can see here delegates – it has five items in it.
Print, Print, PrintTwice, Print, Print.
If you think about it for a while.
The delegate mechanism may not seem necessary at the moment, but it will prove necessary when programming events.
But we'll talk about it when we go to events.
108 %
Automatyczne
Nazwa
Wartość
Typ
delegates
{System.Delegate[5]}
System.Delegate[]
[0]
{Method = {Void Print(System.String)}}
System.Delegate ...
[1]
{Method = {Void Print(System.String)}}
System.Delegate ...
[2]
{Method = {Void PrintTwice(System.String)}}
System.Delegate ...
[3]
{Method = {Void Print(System.String)}}
System.Delegate ...
[4]
{Method = {Void Print(System.String)}}
System.Delegate ...
p
{Method = {Void Print(System.String)}}
Delegates.EntryP...
Stos wywołań
Nazwa
Delegates.exe!Delegates.EntryPoint.Main() Wiersz 2

Delegates chains with returning methods – catching all returns


Delegates - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
04. CatchAllReturnsFromAChain
Rozpocznij
EntryPoint.cs
EntryPoint.cs
04. CatchAllReturnsFromAChain
CatchAllReturns.EntryPoint.CheckLengthOfString
C:\Windows\system32\cmd.exe
False
Aby kontynuować, naciśnij dowolny klawisz . . .
Eksplorator serwera
Przybornik
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4
5 namespace CatchAllReturns
6 {
7 class EntryPoint
8 {
9 public delegate bool CheckLengthOfString(string message);
10
11 static void Main()
12 {
13 CheckLengthOfString d = LessThanFive;
14 d += MoreThanFive;
15 d += ExactlyFive;
16
17 Console.WriteLine(d("Message"));
18 }
19
20 public static bool LessThanFive(string item)
21 {
22 return item.Length < 5;
23 }
24 public static bool MoreThanFive(string item)
25 {
26 return item.Length > 5;
27 }
28 public static bool ExactlyFive(string item)
29 {
30 return item.Length == 5;
31 }
32 }
We have new example which examines length of a string – More, less or exactly five.
There is a delegate of the same signature (one string argument and return bool value).
And we're going to create a new chain of delegates with these methods.
So we put three methods which checks three conditions.
So if we now use this delegate with a given message, there are three return values from three methods. And we can catch only one of them (the last one).

Delegates - Microsoft Visual Studio

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

Debug | Any CPU | 04. CatchAllReturnsFromAChain | Rozpocznij

EntryPoint.cs | EntryPoint.cs

04. CatchAllReturnsFromAChain | CatchAllReturns.EntryPoint.CheckLengthOfString

```
4
5   namespace CatchAllReturns
6   {
7       class EntryPoint
8       {
9           public delegate bool CheckLengthOfString(string message);
10
11          static void Main()
12          {
13              CheckLengthOfString d = LessThanFive;
14              d += MoreThanFive;
15              d += ExactlyFive;
16
17              List<bool> results = new List<bool>();
18              foreach (var del in d.GetInvocationList())
19              {
20                  results.Add((bool)del.DynamicInvoke("Message"));
21              }
22
23              Console.WriteLine("Foreach loop results: " + string.Join(", ", results));
24          }
25
26          public static bool LessThanFive(string item)
27          {
28              return item.Length < 5;
29          }
30          public static bool MoreThanFive(string item)
31          {
32              return item.Length > 5;
33          }
34          public static bool ExactlyFive(string item)
35          {
```

C:\Windows\system32\cmd.exe

```
Foreach loop results: False, True, False
Aby kontynuować, naciśnij dowolny klawisz . . .
```

So how do we get all three values. The way to do this is to individually execute each of the items in this delegate and create a list of results.
We are going to use the invocation list and DynamicInvoke. So in this example we return Booleans to the returns list. The DynamicInvoke returns object value. It which basically means that we have to manually type cast it into a boolean.
So let's also print the results from this operation and the results: we get all three returns from these methods. We get false true and false. We can do this in another way, an easier way.

Delegates - Microsoft Visual Studio

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

Debug | Any CPU | 04. CatchAllReturnsFromAChain | Rozpocznij

EntryPoint.cs | EntryPoint.cs

04. CatchAllReturnsFromAChain | CatchAllReturns.EntryPoint | MoreThanFive(string item)

```
4
5   namespace CatchAllReturns
6   {
7       class EntryPoint
8       {
9           public delegate bool CheckLengthOfString(string message);
10
11          static void Main()
12          {
13              CheckLengthOfString d = LessThanFive;
14              d += MoreThanFive;
15              d += ExactlyFive;
16
17              List<bool> results = new List<bool>();
18
19              results = d.GetInvocationList().Select(del => (bool)del.DynamicInvoke("Message")).ToList();
20              Console.WriteLine("Lambda expression: " + string.Join(", ", results));
21          }
22
23          public static bool LessThanFive(string item)
24          {
25              return item.Length < 5;
26          }
27          public static bool MoreThanFive(string item)
28          {
29              return item.Length > 5;
30          }
31          public static bool ExactlyFive(string item)
32          {
33              return item.Length == 5;
34          }
35      }
```

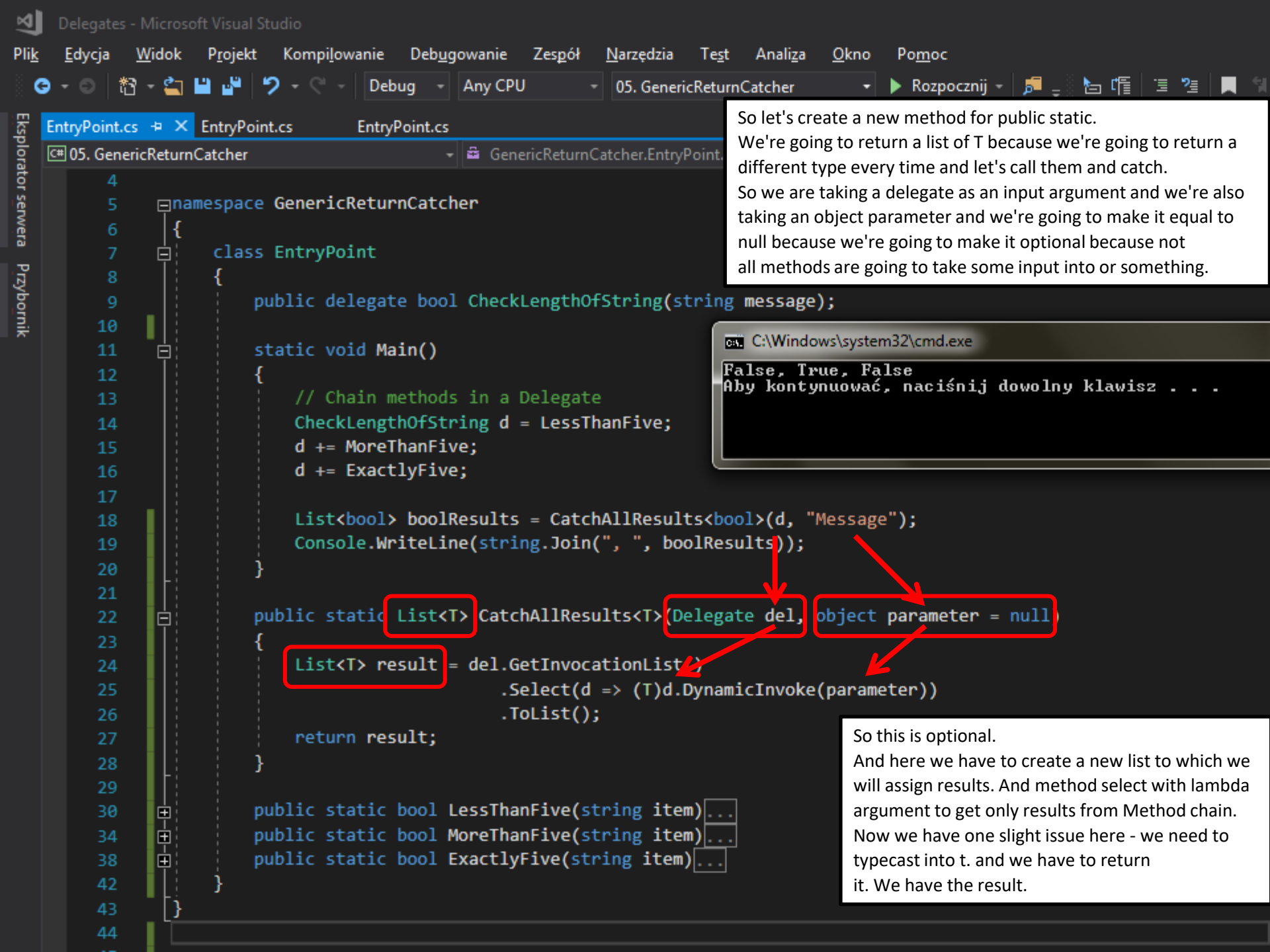
C:\Windows\system32\cmd.exe

```
Lambda expression: False, True, False
Aby kontynuować, naciśnij dowolny klawisz . . .
```

We can do it in an easier way by using Lambda and select method.
We call select for invocation list, and results of DynamicInvoke calls convert to the list of bools. As you can see result is the same but code is placing in just one line of code.
You can even create a generic method that would work for any delegate that you have and you will be able to catch all of the return types.
And this is what we will do next.

Generic methods to catch all returns

We can catch all of the results that are returned by a delegate chain.
But why do we have to rewrite the code every time; we can extract it in a generic method.


Delegates - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
05. GenericReturnCatcher
Rozpocznij
EntryPoint.cs
EntryPoint.cs
EntryPoint.cs
05. GenericReturnCatcher
GenericReturnCatcher.EntryPoint
Eksplorator serwera
Przybornik
So let's create a new method for public static.
We're going to return a list of T because we're going to return a different type every time and let's call them and catch.
So we are taking a delegate as an input argument and we're also taking an object parameter and we're going to make it equal to null because we're going to make it optional because not all methods are going to take some input into or something.
C:\Windows\system32\cmd.exe
False, True, False
Aby kontynuować, naciśnij dowolny klawisz . . .
namespace GenericReturnCatcher
{
class EntryPoint
{
public delegate bool CheckLengthOfString(string message);
static void Main()
{
// Chain methods in a Delegate
CheckLengthOfString d = LessThanFive;
d += MoreThanFive;
d += ExactlyFive;
List<bool> boolResults = CatchAllResults<bool>(d, "Message");
Console.WriteLine(string.Join(", ", boolResults));
}
public static List<T> CatchAllResults<T>(Delegate del, object parameter = null)
{
List<T> result = del.GetInvocationList()
.Select(d => (T)d.DynamicInvoke(parameter))
.ToList();
return result;
}
public static bool LessThanFive(string item)...
public static bool MoreThanFive(string item)...
public static bool ExactlyFive(string item)...
}
}
So this is optional.
And here we have to create a new list to which we will assign results. And method select with lambda argument to get only results from Method chain.
Now we have one slight issue here - we need to typecast into t. and we have to return it. We have the result.

Delegates - Microsoft Visual Studio

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

Debug | Any CPU | 05. GenericReturnCatcher | Rozpocznij

EntryPoint.cs* | EntryPoint.cs | EntryPoint.cs

05. GenericReturnCatcher | GenericReturnCatcher.EntryPoint | LessThanFive(string item)

```
9   public delegate bool CheckLengthOfString(string message);
10  public delegate int GetLength(string message);
11
12  static void Main()
13  {
14      // Chain methods in a Delegate
15      CheckLengthOfString d = LessThanFive;
16      d += MoreThanFive;
17      d += ExactlyFive;
18
19      List<bool> boolResults = CatchAllResults<bool>(d, "Message");
20      Console.WriteLine(string.Join(", ", boolResults));
21
22      // Second Medthod Chain
23      GetLength p = x => x.Length;
24      p += x => x.Length + 1;
25      p += x => x.Length + 2;
26
27      List<int> lengths = CatchAllResults<int>(p, "asd");
28      Console.WriteLine(string.Join(", ", lengths));
29  }
30
31  public static List<T> CatchAllResults<T>(Delegate del, object parameter = null)
32  {
33      List<T> result = del.GetInvocationList()
34                          .Select(d => (T)d.DynamicInvoke(parameter))
35                          .ToList();
36      return result;
37  }
38
39  public static bool LessThanFive(string item)
43  public static bool MoreThanFive(string item)
```

C:\Windows\system32\cmd.exe

```
False, True, False
3, 4, 5
Aby kontynuować, naciśnij dowolny klawisz . . .
```

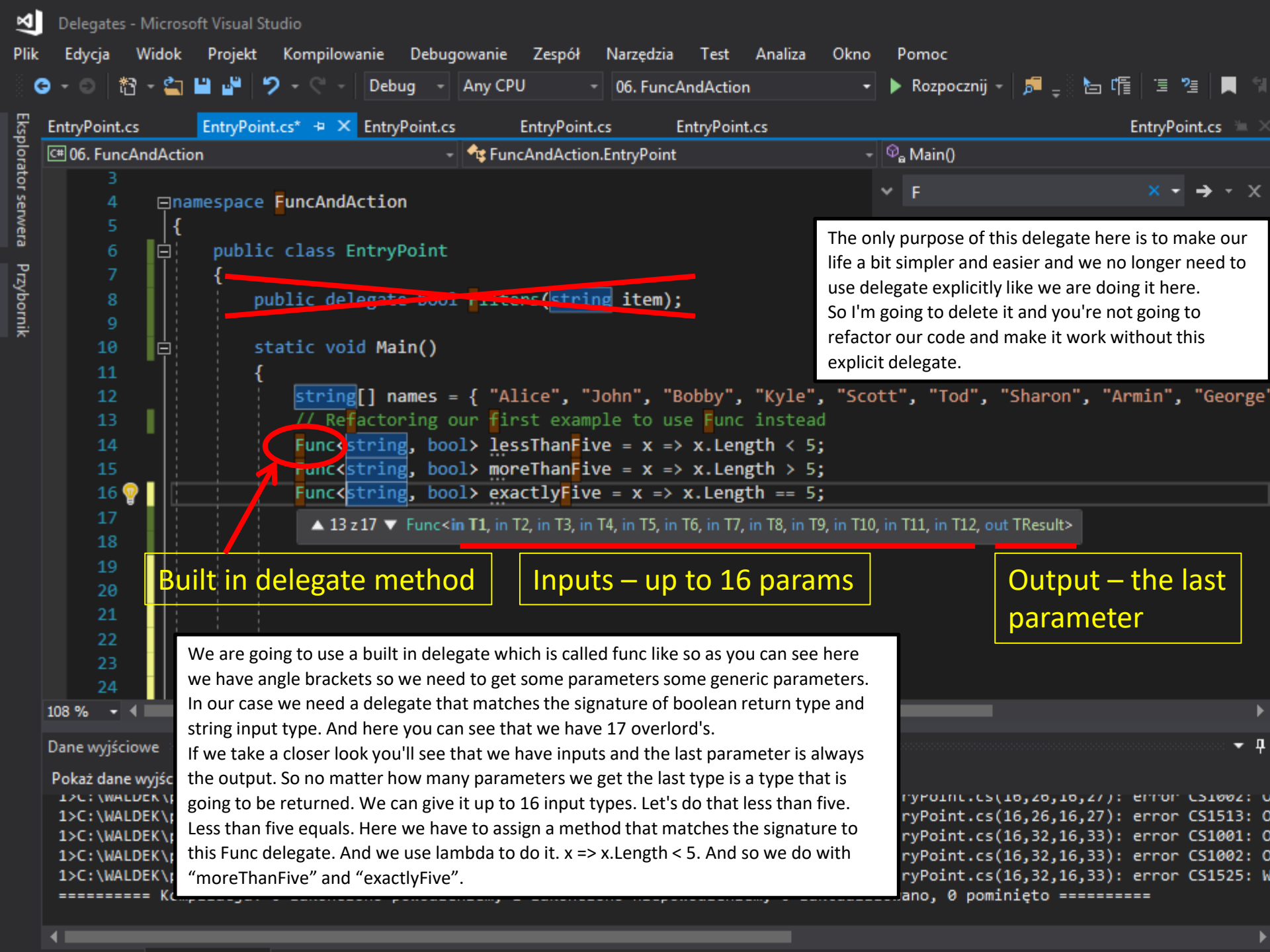
Now just to confirm that it's working for different types let's create a new delegate. It get lengths. So we are turning results to integers.
We're going to use lambda expressions this time because it's easier and faster.
We assign three methods to the chain: length, length+1, length+2.

So we have two methods that are going to take the length of the input argument which is the message. In first delegate chain we return boolean values and in second integers.

So in each cases we catch all the results and we can show results on the console. So we managed to generalize this method as well.

Func and *Action* delegates

All right I'm going to show you one new type.
It's not exactly the type of delegate but is to delegate.


Delegates - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
06. FuncAndAction
Rozpocznij
EntryPoint.cs
EntryPoint.cs*
EntryPoint.cs
EntryPoint.cs
EntryPoint.cs
EntryPoint.cs
06. FuncAndAction
FuncAndAction.EntryPoint
Main()
namespace FuncAndAction
{
public class EntryPoint
{
public delegate bool Filters(string item);
static void Main()
{
string[] names = { "Alice", "John", "Bobby", "Kyle", "Scott", "Tod", "Sharon", "Armin", "George"
// Refactoring our first example to use Func instead
Func<string, bool> lessThanFive = x => x.Length < 5;
Func<string, bool> moreThanFive = x => x.Length > 5;
Func<string, bool> exactlyFive = x => x.Length == 5;
▲ 13 z 17 ▼ Func<in T1, in T2, in T3, in T4, in T5, in T6, in T7, in T8, in T9, in T10, in T11, in T12, out TResult>
Built in delegate method
Inputs – up to 16 params
Output – the last parameter
The only purpose of this delegate here is to make our life a bit simpler and easier and we no longer need to use delegate explicitly like we are doing it here.
So I'm going to delete it and you're not going to refactor our code and make it work without this explicit delegate.
We are going to use a built in delegate which is called func like so as you can see here we have angle brackets so we need to get some parameters some generic parameters. In our case we need a delegate that matches the signature of boolean return type and string input type. And here you can see that we have 17 overlord's.
If we take a closer look you'll see that we have inputs and the last parameter is always the output. So no matter how many parameters we get the last type is a type that is going to be returned. We can give it up to 16 input types. Let's do that less than five. Less than five equals. Here we have to assign a method that matches the signature to this Func delegate. And we use lambda to do it. x => x.Length < 5. And so we do with “moreThanFive” and “exactlyFive”.
108 %
Dane wyjściowe
Pokaż dane wyjśc
1>C:\WALDEK\p
1>C:\WALDEK\p
1>C:\WALDEK\p
1>C:\WALDEK\p
1>C:\WALDEK\p
ryPoint.cs(16,26,16,27): error CS1513:
ryPoint.cs(16,32,16,33): error CS1001:
ryPoint.cs(16,32,16,33): error CS1002:
ryPoint.cs(16,32,16,33): error CS1525:
ano, 0 pominięto ==========

Even without us creating a new delegate is still going to work simply because these methods here match the signature of this func delegate. Let's use the different approaches. Instead of giving it the methods we already saw that they are working we are going to use the expression.

```
static void Main()
{
    string[] names = { "Alice", "John", "Bobby", "Kyle", "Scott", "Tod", "Sharon", "Armin", "George"
    // Refactoring our first example to use Func instead
    Func<string, bool> lessThanFive = x => x.Length < 5;
    Func<string, bool> moreThanFive = x => x.Length > 5;
    Func<string, bool> exactlyFive = x => x.Length == 5;

    List<string> namesLessThanFiveChars = ExtractStrings(names, lessThanFive);
    List<string> namesMoreThanFiveChars = ExtractStrings(names, moreThanFive);
    List<string> namesExactlyFiveChars = ExtractStrings(names, exactlyFive);

    Console.WriteLine("Names less than five chars: " + string.Join(", ", namesLessThanFiveChars));
    Console.WriteLine("Names more than five chars: " + string.Join(", ", namesMoreThanFiveChars));
    Console.WriteLine("Names exactly five chars: " + string.Join(", ", namesExactlyFiveChars));
}

public static List<string> ExtractStrings(string[] array, Func<string, bool> filter)
{
    List<string> result = new List<string>();
    for (int i = 0; i < array.Length; i++)
    {
        if (filter(array[i]))
        {
            result.Add(array[i]);
        }
    }
    return result;
```

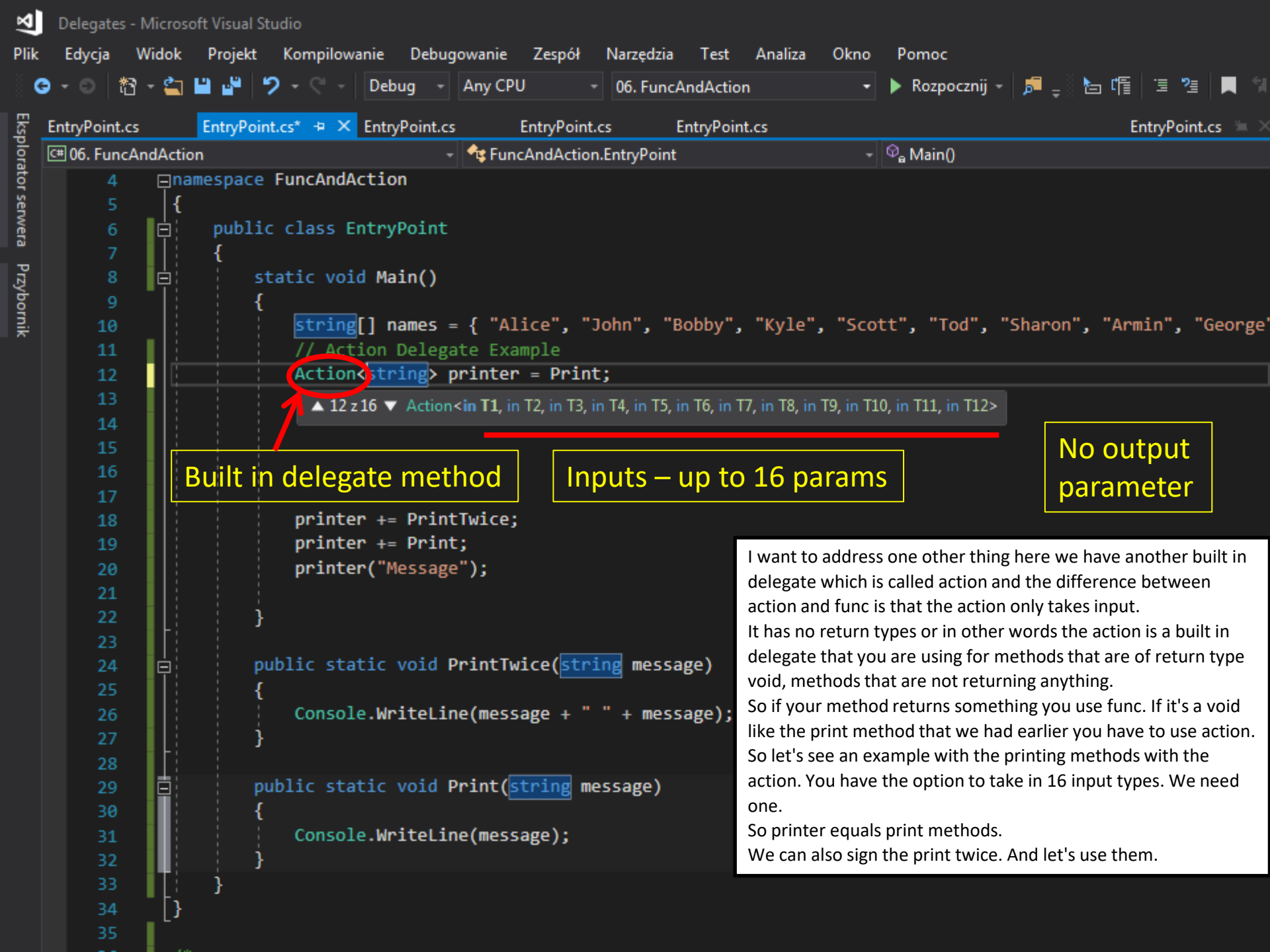
Type - delegate

Variable – method name

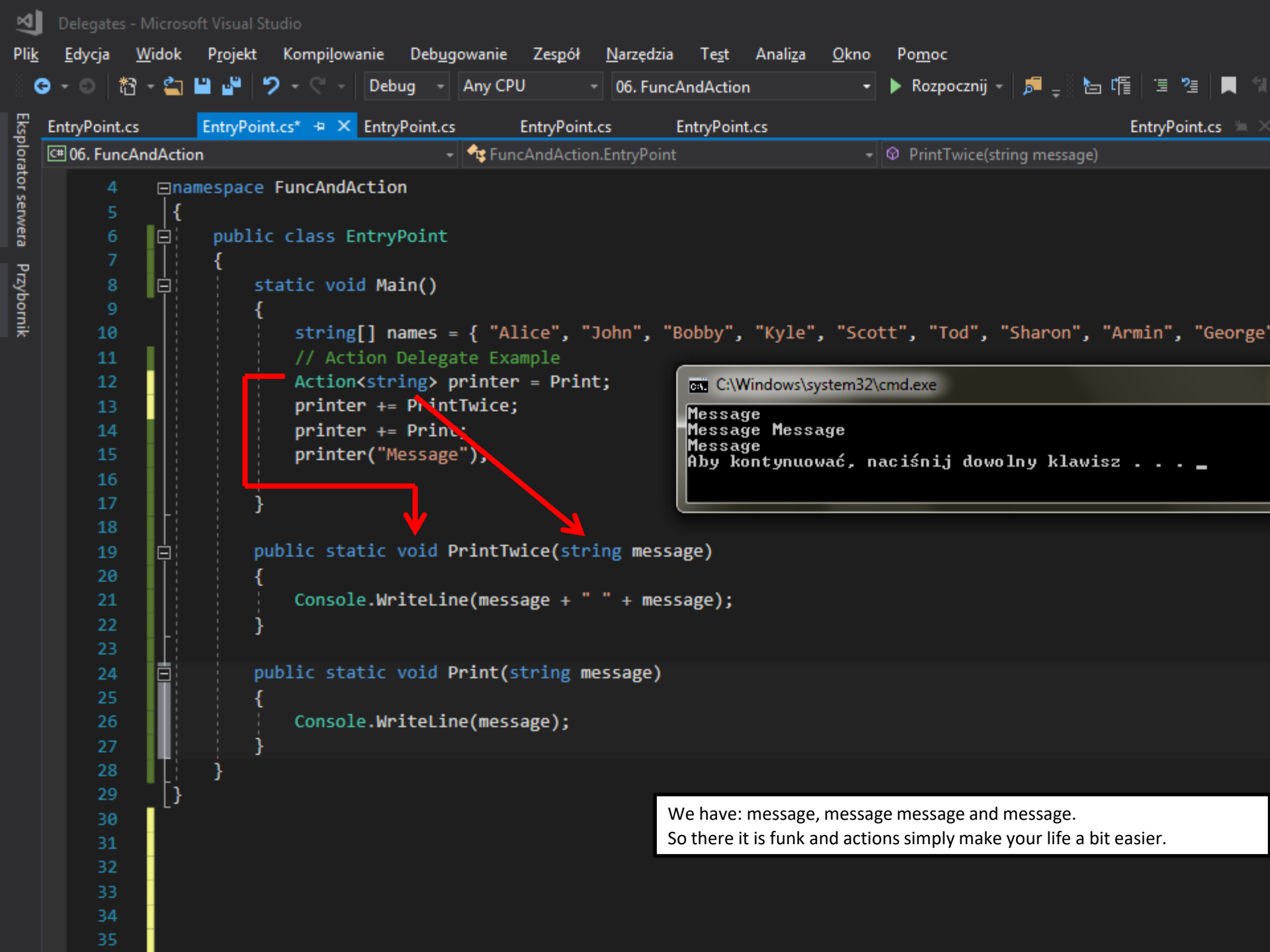
Value – body of method

```
C:\Windows\system32\cmd.exe
Names less than five chars: John, Kyle, Tod
Names more than five chars: Sharon, George
Names exactly five chars: Alice, Bobby, Scott, Arm
Aby kontynuować, naciśnij dowolny klawisz . . .
```

So X is the input X of length less than 5 and instead of using this method we're going to use this delegate that we created here. It's still working. We can do the same thing for more than five. Exactly five but we have seen this enough times already.



I want to address one other thing here we have another built in delegate which is called action and the difference between action and func is that the action only takes input.
It has no return types or in other words the action is a built in delegate that you are using for methods that are of return type void, methods that are not returning anything.
So if your method returns something you use func. If it's a void like the print method that we had earlier you have to use action.
So let's see an example with the printing methods with the action. You have the option to take in 16 input types. We need one.
So printer equals print methods.
We can also sign the print twice. And let's use them.


Delegates - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
06. FuncAndAction
Rozpocznij
EntryPoint.cs
EntryPoint.cs*
EntryPoint.cs
EntryPoint.cs
EntryPoint.cs
EntryPoint.cs
06. FuncAndAction
FuncAndAction.EntryPoint
PrintTwice(string message)
Eksplorator serwera
Przybornik
4 namespace FuncAndAction
5 {
6 public class EntryPoint
7 {
8 static void Main()
9 {
10 string[] names = { "Alice", "John", "Bobby", "Kyle", "Scott", "Tod", "Sharon", "Armin", "George"
11 // Action Delegate Example
12 Action<string> printer = Print;
13 printer += PrintTwice;
14 printer += Print;
15 printer("Message");
16
17 }
18
19 public static void PrintTwice(string message)
20 {
21 Console.WriteLine(message + " " + message);
22 }
23
24 public static void Print(string message)
25 {
26 Console.WriteLine(message);
27 }
28 }
29 }
30
31
32
33
34
35
C:\Windows\system32\cmd.exe
Message
Message Message
Message
Aby kontynuować, naciśnij dowolny klawisz . . .
We have: message, message message and message.
So there it is funk and actions simply make your life a bit easier.

Anonymous methods and lambda expressions

So you might have heard the term anonymous methods. This is what we're talking about.
And this can be used with func and action delegates. So let's see how this works out.

Delegates - Microsoft Visual Studio

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

Debug | Any CPU | 07. AnonymousMethodAndLambc | Rozpocznij

EntryPoint.cs | EntryPoint.cs | EntryPoint.cs | EntryPoint.cs | EntryPoint.cs | EntryPoint.cs | EntryPoint.cs

07. AnonymousMethodAndLambda | AnonymousMethodAndLambda.Ent

```
1   using System;
2   using System.Collections.Generic;
3
4   namespace AnonymousMethodAndLambda
5   {
6       public class EntryPoint
7       {
8           static void Main()
9           {
10
11
12
13              Func<int, bool> checkIntegers = i => i < 8;
14
15
16
17
18
19
20
21              Console.WriteLine(checkIntegers(5));
22          }
23
24      }
25  }
26
27
28
29
30
31
32
```

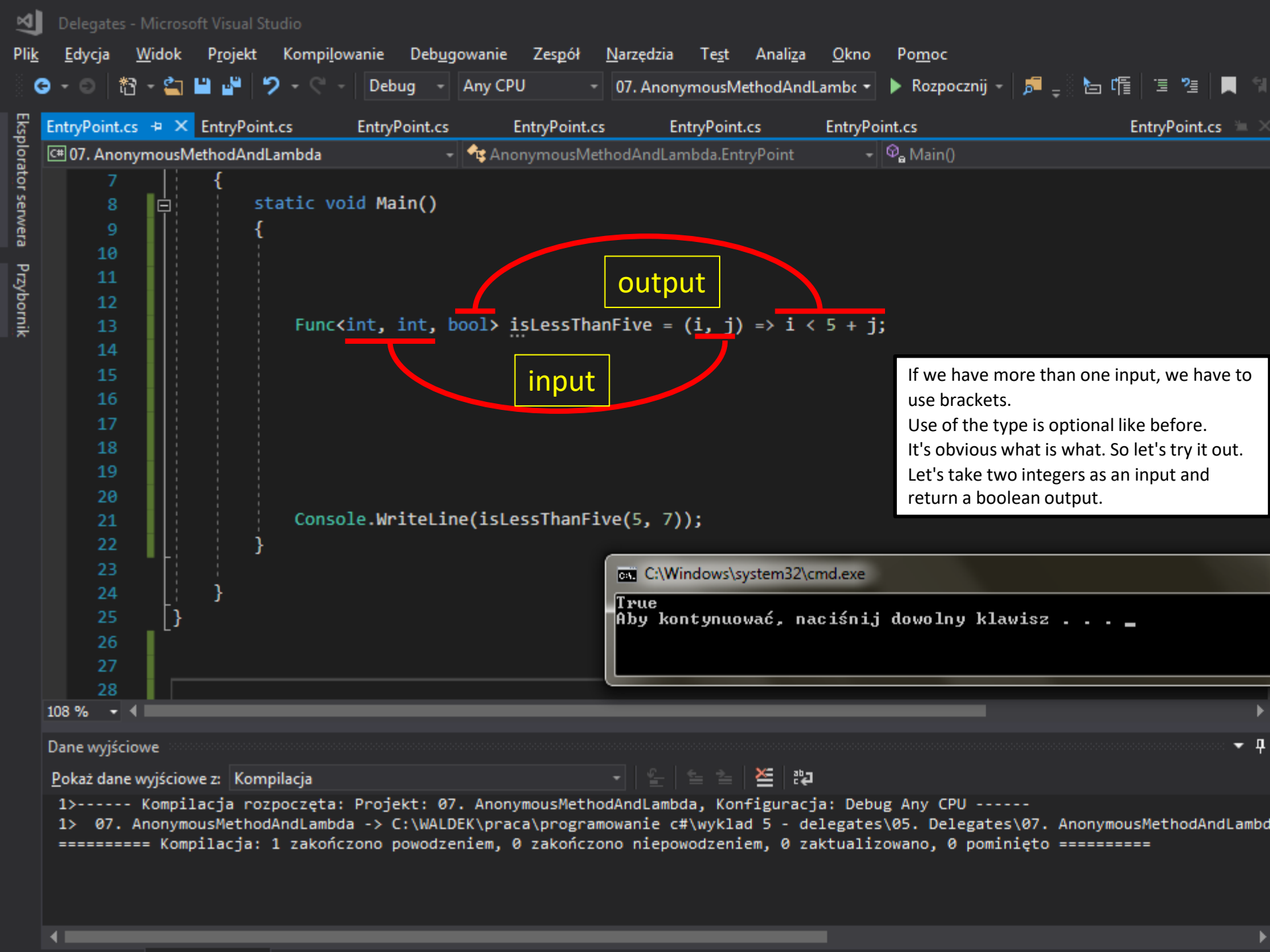
output

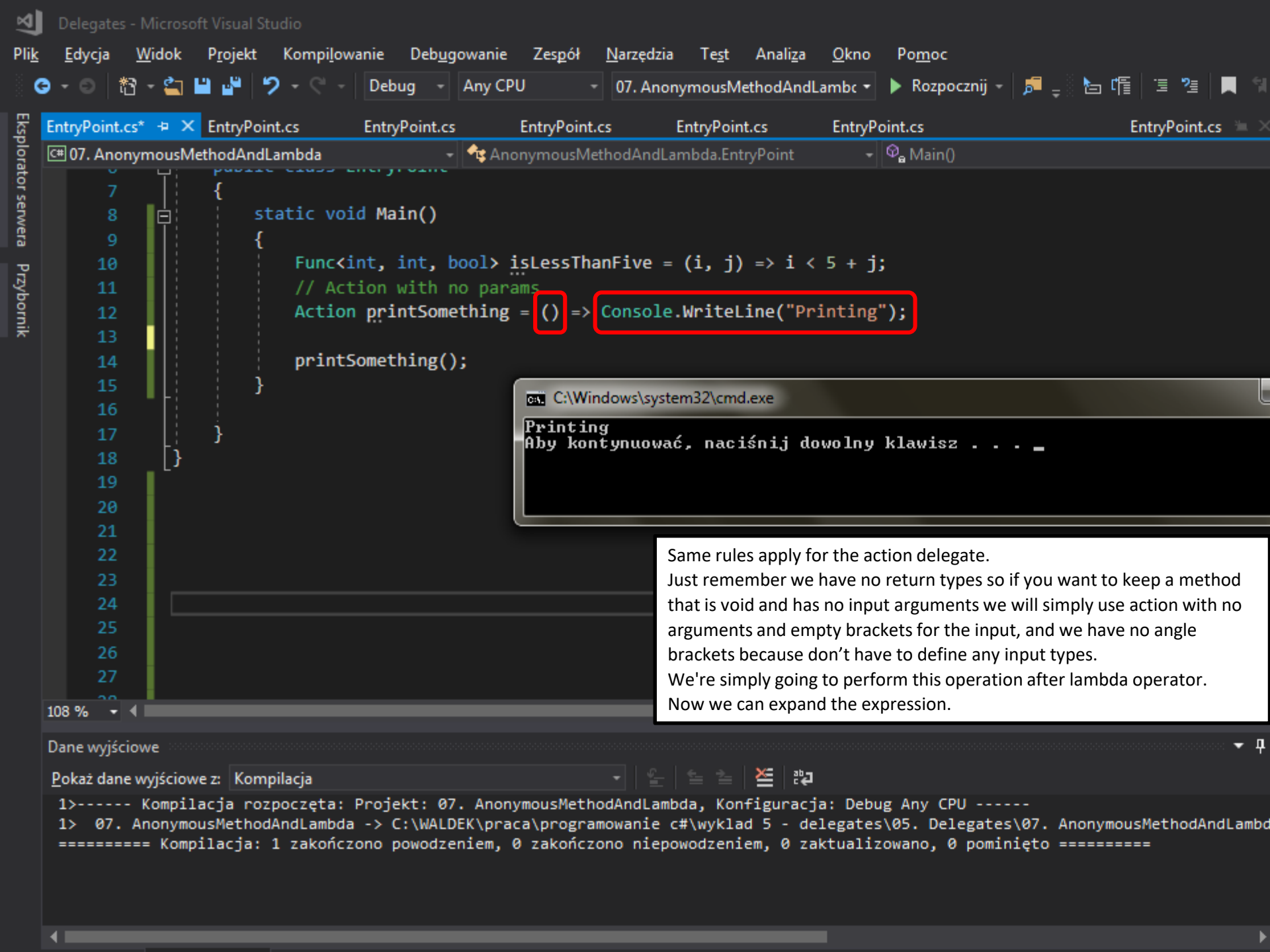
input

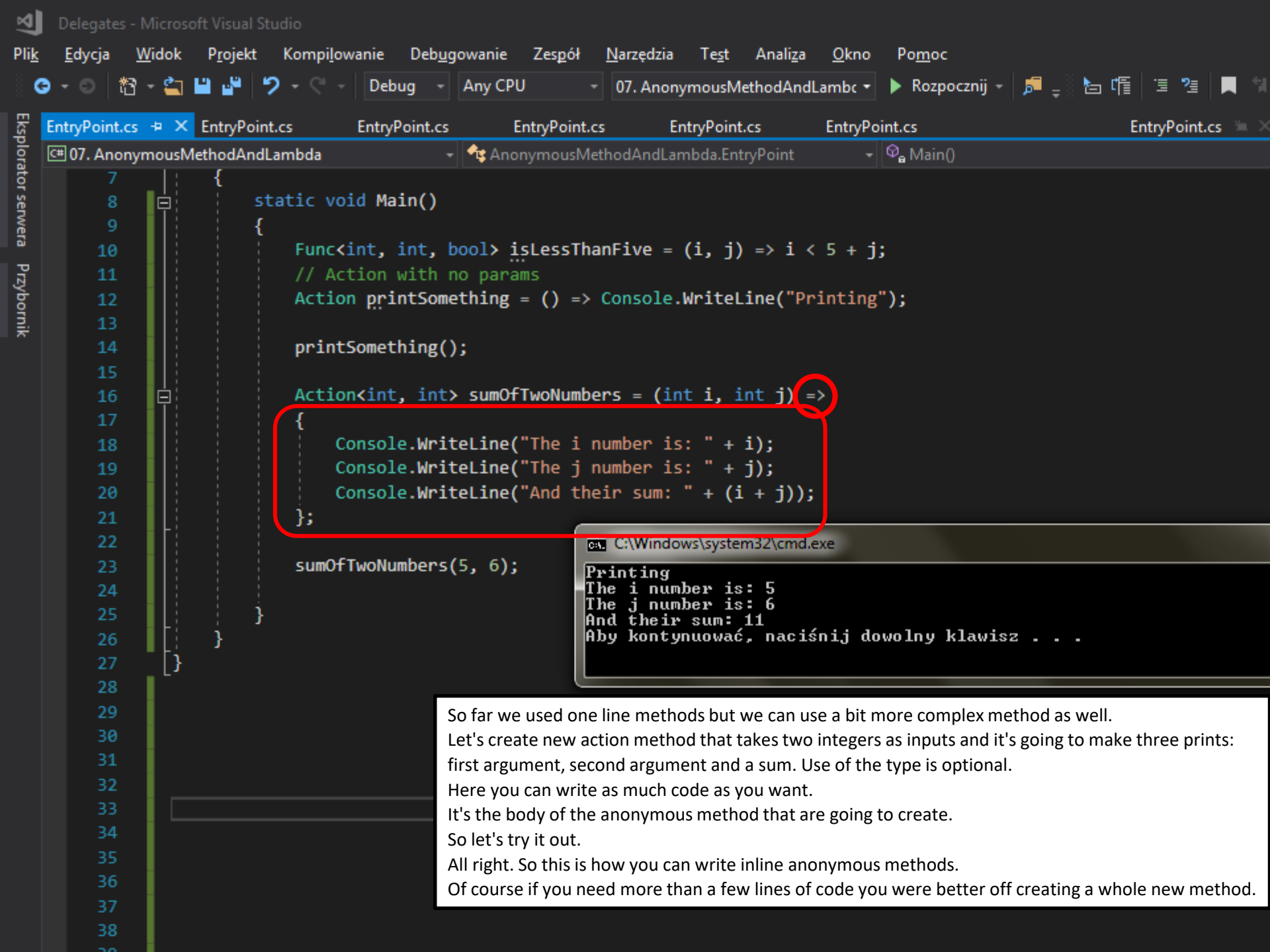
A simple example is to create a delegate that takes one integer input and use a boolean return type. One integer input and check if this integer input is less than 8.
And we can check integer 5. Of course, it is less than 8.
All right. Easy enough.
Something that you may have wondered is why we don't have types when we use variables.
This is a very good question.
But we can have them we can have them but we don't need them. Why.
C-Sharp knows what is a type that we are addressing.
We already defined the types in the func types.
So here we say that we are taking one integer input and we are returning a boolean.
So it's obvious what types we are going to need. This is our input. This is our output.
It's obvious that it's going to be integer. And this code here will give us our return value.
We're performing a comparison and the comparison returns a boolean.
So this is why we are not using any types when we write our code with lambdas.

C:\Windows\system32\cmd.exe

```
True
Aby kontynuować, naciśnij dowolny klawisz . . .
```


Delegates - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
07. AnonymousMethodAndLambc
Rozpocznij
EntryPoint.cs
07. AnonymousMethodAndLambda
AnonymousMethodAndLambda.EntryPoint
Main()
{
static void Main()
{
Func<int, int, bool> isLessThanFive = (i, j) => i < 5 + j;
output
input
Console.WriteLine(isLessThanFive(5, 7));
}
}
}
If we have more than one input, we have to use brackets.
Use of the type is optional like before.
It's obvious what is what. So let's try it out.
Let's take two integers as an input and return a boolean output.
C:\Windows\system32\cmd.exe
True
Aby kontynuować, naciśnij dowolny klawisz . . .
108 %
Dane wyjściowe
Pokaż dane wyjściowe z: Kompilacja
1>------ Kompilacja rozpoczęta: Projekt: 07. AnonymousMethodAndLambda, Konfiguracja: Debug Any CPU ------
1> 07. AnonymousMethodAndLambda -> C:\WALDEK\praca\programowanie c#\wyklad 5 - delegates\05. Delegates\07. AnonymousMethodAndLambd
========== Kompilacja: 1 zakończono powodzeniem, 0 zakończono niepowodzeniem, 0 zaktualizowano, 0 pominięto ==========


Delegates - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
07. AnonymousMethodAndLambc
Rozpocznij
EntryPoint.cs*
EntryPoint.cs
07. AnonymousMethodAndLambda
AnonymousMethodAndLambda.EntryPoint
Main()
{
static void Main()
{
Func<int, int, bool> isLessThanFive = (i, j) => i < 5 + j;
// Action with no params
Action printSomething = () => Console.WriteLine("Printing");
printSomething();
}
}
C:\Windows\system32\cmd.exe
Printing
Aby kontynuować, naciśnij dowolny klawisz . . .
Same rules apply for the action delegate.
Just remember we have no return types so if you want to keep a method that is void and has no input arguments we will simply use action with no arguments and empty brackets for the input, and we have no angle brackets because don't have to define any input types.
We're simply going to perform this operation after lambda operator.
Now we can expand the expression.
108 %
Dane wyjściowe
Pokaż dane wyjściowe z: Kompilacja
1>------ Kompilacja rozpoczęta: Projekt: 07. AnonymousMethodAndLambda, Konfiguracja: Debug Any CPU ------
1> 07. AnonymousMethodAndLambda -> C:\WALDEK\praca\programowanie c#\wyklad 5 - delegates\05. Delegates\07. AnonymousMethodAndLambd
========== Kompilacja: 1 zakończono powodzeniem, 0 zakończono niepowodzeniem, 0 zaktualizowano, 0 pominięto ==========


Delegates - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
07. AnonymousMethodAndLambc
Rozpocznij
EntryPoint.cs
07. AnonymousMethodAndLambda
AnonymousMethodAndLambda.EntryPoint
Main()
Eksplorator serwera
Przybornik
{
static void Main()
{
Func<int, int, bool> isLessThanFive = (i, j) => i < 5 + j;
// Action with no params
Action printSomething = () => Console.WriteLine("Printing");
printSomething();
Action<int, int> sumOfTwoNumbers = (int i, int j) =>
{
Console.WriteLine("The i number is: " + i);
Console.WriteLine("The j number is: " + j);
Console.WriteLine("And their sum: " + (i + j));
};
sumOfTwoNumbers(5, 6);
}
}
}
C:\Windows\system32\cmd.exe
Printing
The i number is: 5
The j number is: 6
And their sum: 11
Aby kontynuować, naciśnij dowolny klawisz . . .
So far we used one line methods but we can use a bit more complex method as well.
Let's create new action method that takes two integers as inputs and it's going to make three prints:
first argument, second argument and a sum. Use of the type is optional.
Here you can write as much code as you want.
It's the body of the anonymous method that are going to create.
So let's try it out.
All right. So this is how you can write inline anonymous methods.
Of course if you need more than a few lines of code you were better off creating a whole new method.

More complex anonymous methods

All right so we are going to work on a bit more complex anonymous method. Example we're going to do is to perform the filtering with an anonymous method.

Delegates - Microsoft Visual Studio

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

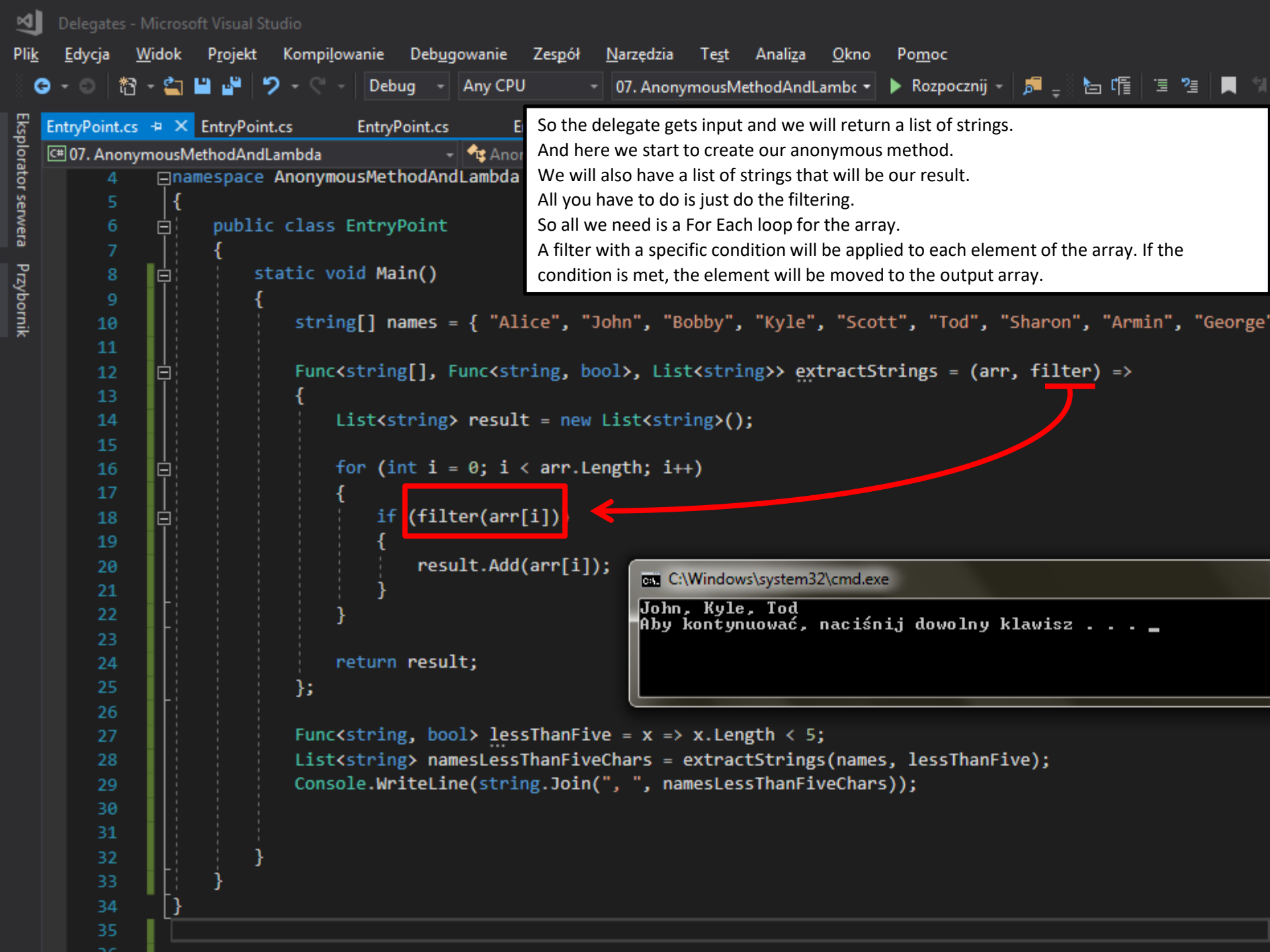
Debug | Any CPU | 07. AnonymousMethodAndLambc | Rozpocznij

EntryPoint.cs* | EntryPoint.cs | EntryPoint.cs | EntryPoint.cs | EntryPoint.cs | EntryPoint.cs | EntryPoint.cs

07. AnonymousMethodAndLambda | AnonymousMethodAndLambda.EntryPoint | Main()

```
7       {
8           static void Main()
9           {
10              string[] names = { "Alice", "John", "Bobby", "Kyle", "Scott", "Tod", "Sharon", "Armin", "George"
11
12
13
14
15
16
17              Func<string[], Func<string, bool>, List<string>> extractStrings = (arr, filter) =>
18              {
19                  List<string> result = new List<string>();
20
21                  // condition
22
23                  return result;
24              };
25          }
26      }
27  }
28
29
30
31
32
33
34
35
36
37
38
```

So we're going to create func delegate.
We are taking a string array as a first input and a delegate (which will be our filter) as a second. Third parameter (string array) is an output. The second parameter that is delegate will check a particular string and return a Boolean value.
So, we have a type input which is another func of string and bool


Delegates - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
07. AnonymousMethodAndLambc
Rozpocznij
EntryPoint.cs
EntryPoint.cs
EntryPoint.cs
07. AnonymousMethodAndLambda
So the delegate gets input and we will return a list of strings.
And here we start to create our anonymous method.
We will also have a list of strings that will be our result.
All you have to do is just do the filtering.
So all we need is a For Each loop for the array.
A filter with a specific condition will be applied to each element of the array. If the condition is met, the element will be moved to the output array.
namespace AnonymousMethodAndLambda
{
public class EntryPoint
{
static void Main()
{
string[] names = { "Alice", "John", "Bobby", "Kyle", "Scott", "Tod", "Sharon", "Armin", "George"
Func<string[], Func<string, bool>, List<string>> extractStrings = (arr, filter) =>
{
List<string> result = new List<string>();
for (int i = 0; i < arr.Length; i++)
{
if (filter(arr[i]))
{
result.Add(arr[i]);
}
}
return result;
};
Func<string, bool> lessThanFive = x => x.Length < 5;
List<string> namesLessThanFiveChars = extractStrings(names, lessThanFive);
Console.WriteLine(string.Join(", ", namesLessThanFiveChars));
}
}
}
C:\Windows\system32\cmd.exe
John, Kyle, Tod
Aby kontynuować, naciśnij dowolny klawisz . . .

Delegates - Microsoft Visual Studio

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

Debug | Any CPU | 07. AnonymousMethodAndLambc | Rozpocznij

EntryPoint.cs

07. AnonymousMethodAndLambda | AnonymousMethodAndLambda.EntryPoint | Main()

```
4   namespace AnonymousMethodAndLambda
5   {
6       public class EntryPoint
7       {
8           static void Main()
9           {
10              string[] names = { "Alice", "John", "Bobby", "Kyle", "Scott", "Tod", "Sharon", "Armin", "George"
11
12              Func<string[], Func<string, bool>, List<string>> extractStrings = (arr, filter) =>...
26
27  //              Func<string, bool> lessThanFive = x => x.Length < 5;
28  //              List<string> namesLessThanFiveChars = extractStrings(names, x => x.Length < 5);
29              Console.WriteLine(string.Join(", ", extractStrings(names, x => x.Length < 5)));
30              Console.WriteLine(string.Join(", ", extractStrings(names, x => x[1] == 'o')));
31              Console.WriteLine(string.Join(", ", extractStrings(names, x => x.CompareTo("Kyle") < 0)));
32              Console.WriteLine(string.Join(", ", extractStrings(names, z => z.ToUpper().Equals("SCOTT"))));
33
34
35          }
36      }
37  }
38
39
40
41
42
43
44
45
46
47
48
```

C:\Windows\system32\cmd.exe

```
John, Kyle, Tod
John, Bobby, Tod
Alice, John, Bobby, Armin, George
Scott
Aby kontynuować, naciśnij dowolny klawisz . . .
```

Alright, let's take advantage of this. We will create a list of strings that will be our output. We also need a filter, which we will give as the second input parameter. Select strings whose lengths are less than 5. We save the filter as the second parameter using the lambda expression. X is the filter input, which is a string. The filter returns a logical value, which is our output. Fine. As you can see it works. Because our filter expression is a delegate, we can pass different pieces of code through it. In the second example, we check whether the letter 'o' is at the specified position of the string. In the following examples, we call methods that compare and convert strings. We can also build more advanced filters. However, if we have to do something really complicated, it is better to create a separate method.