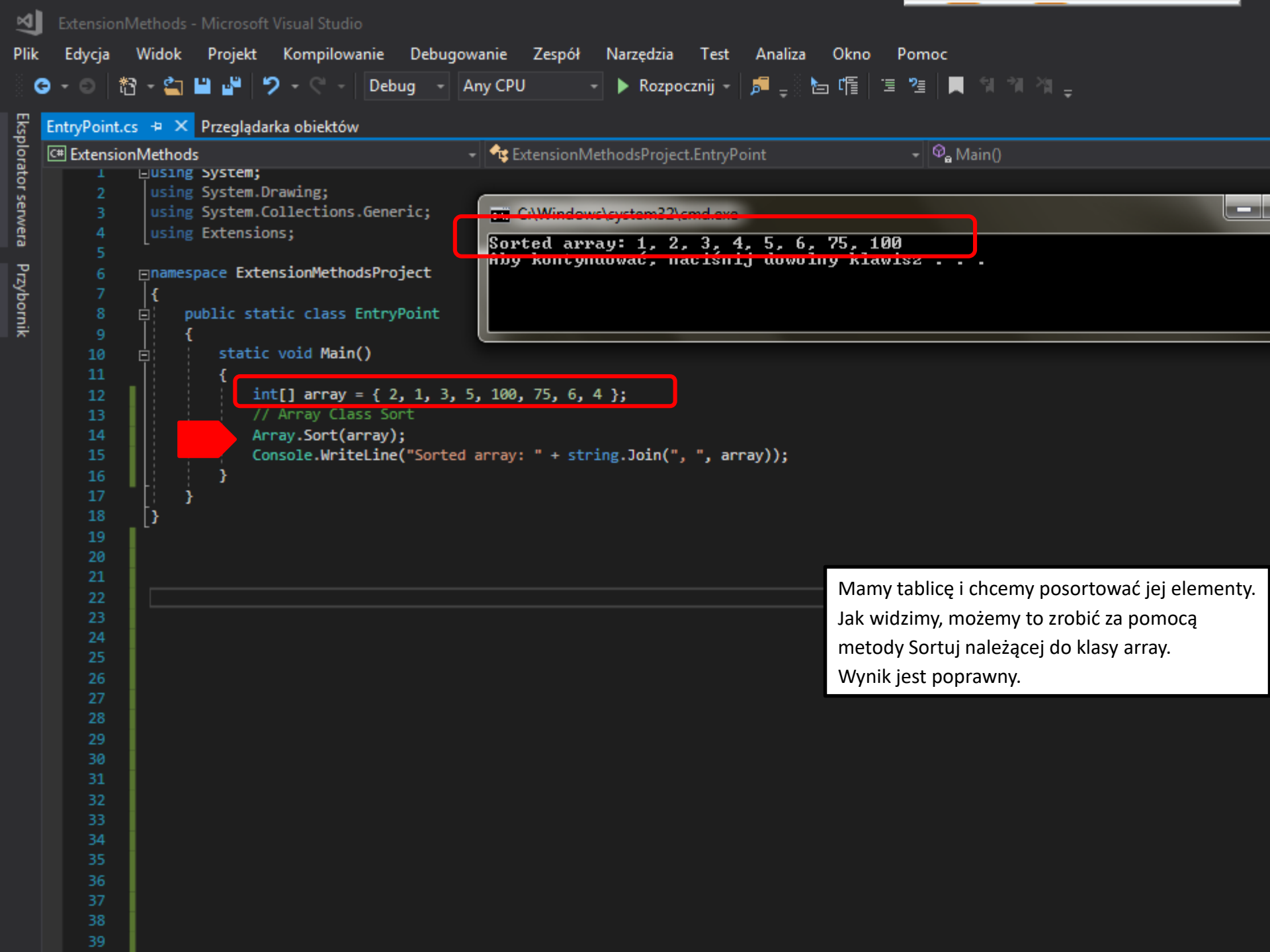
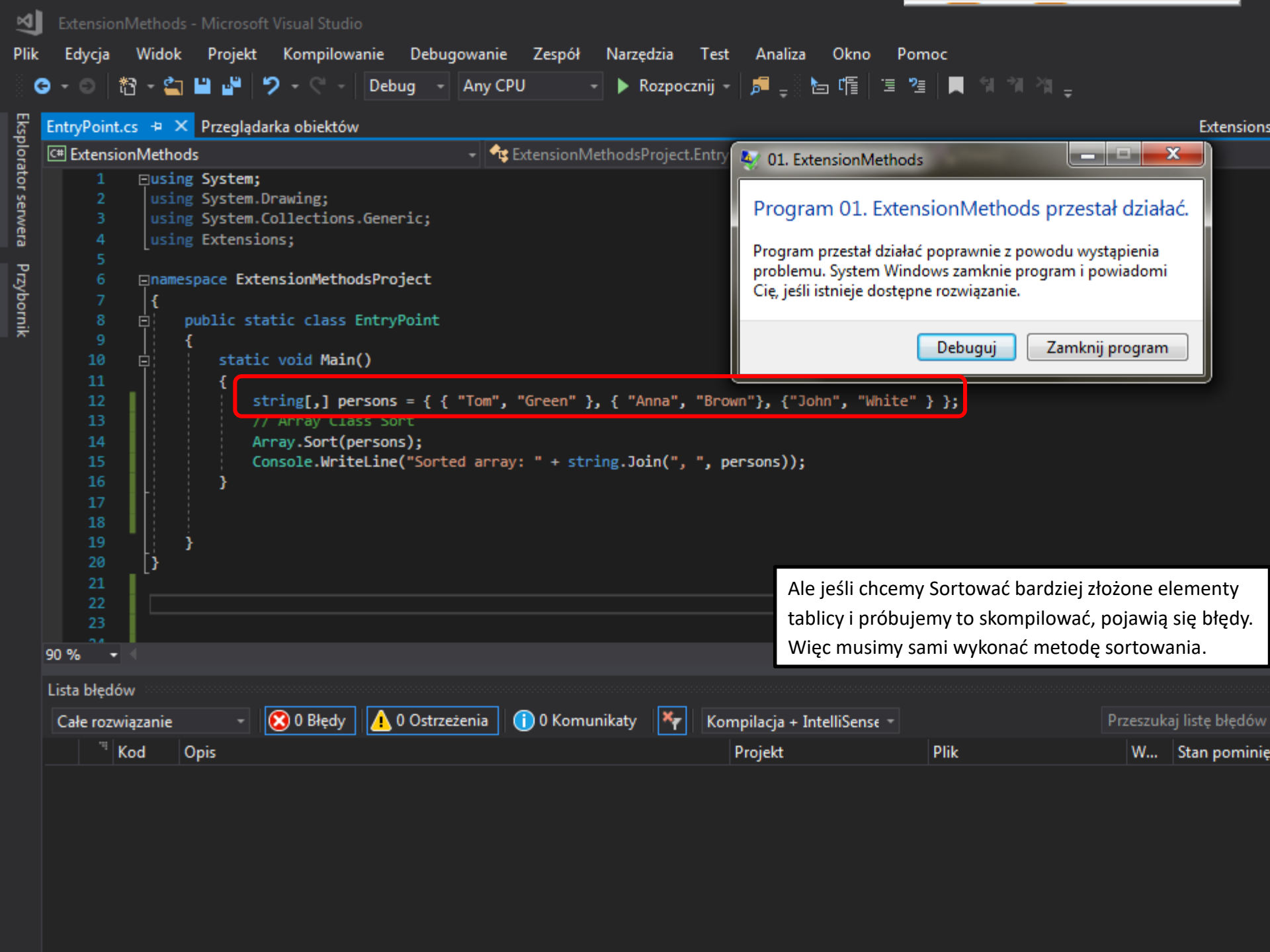
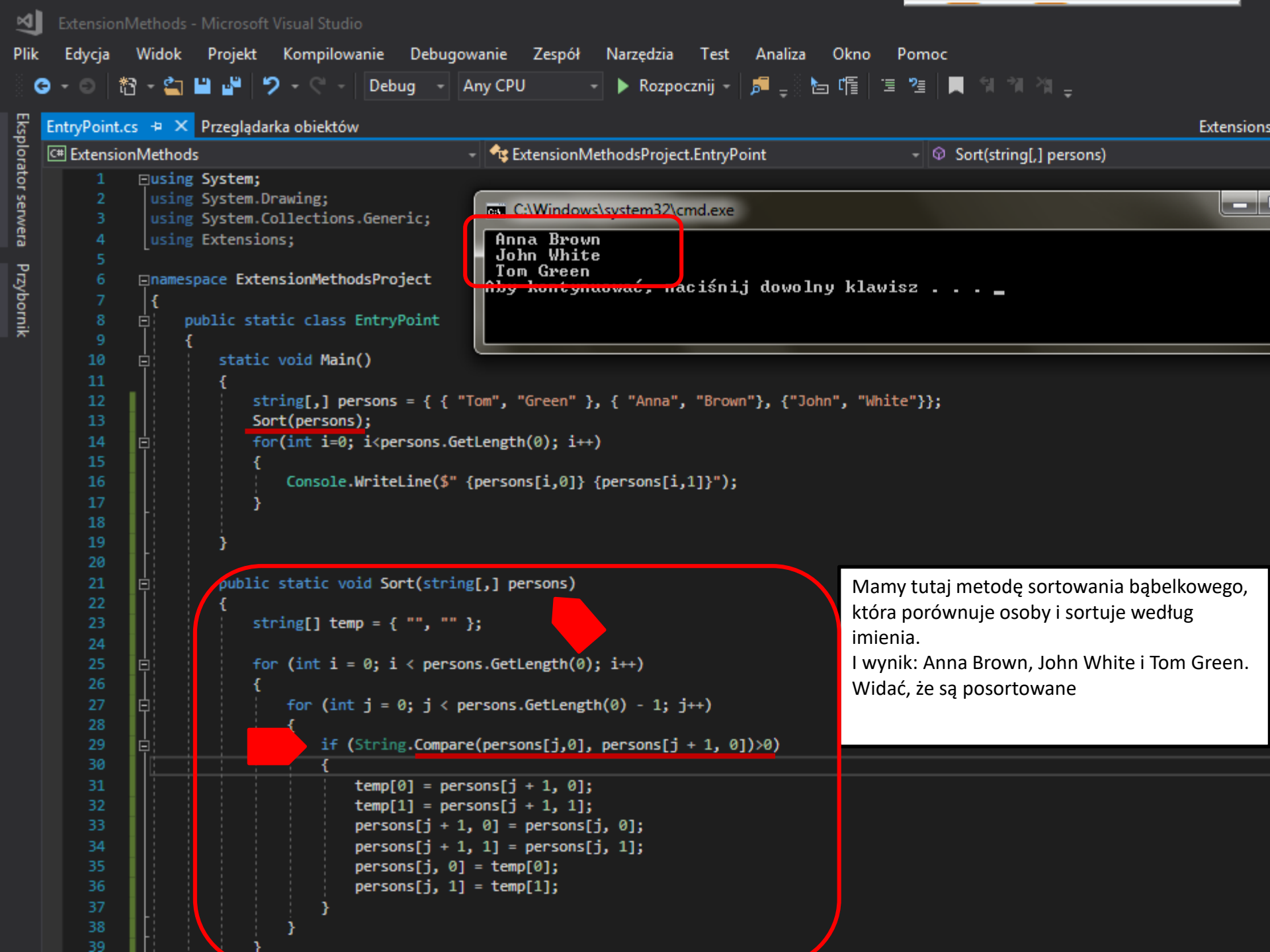
Extension methods

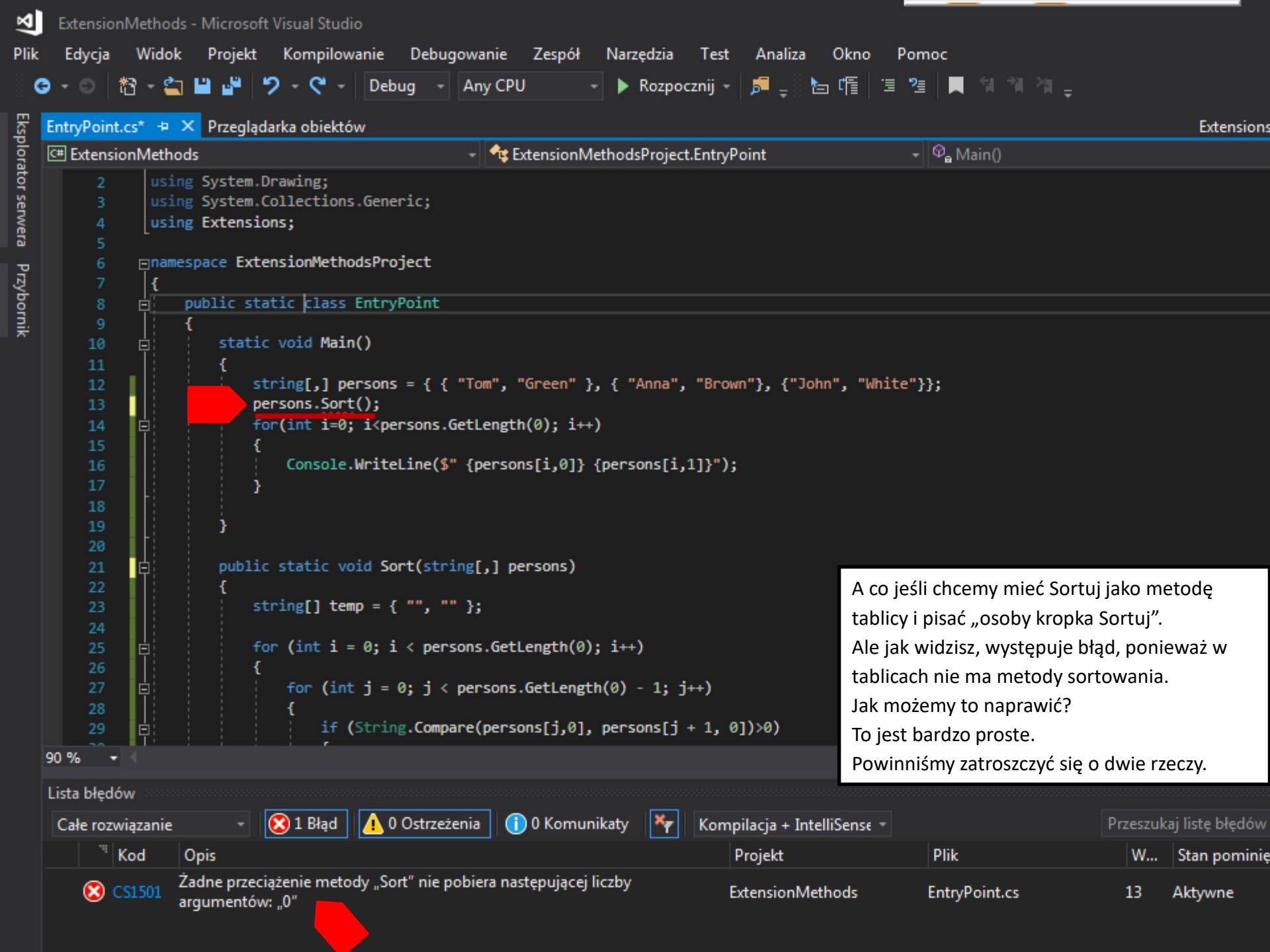


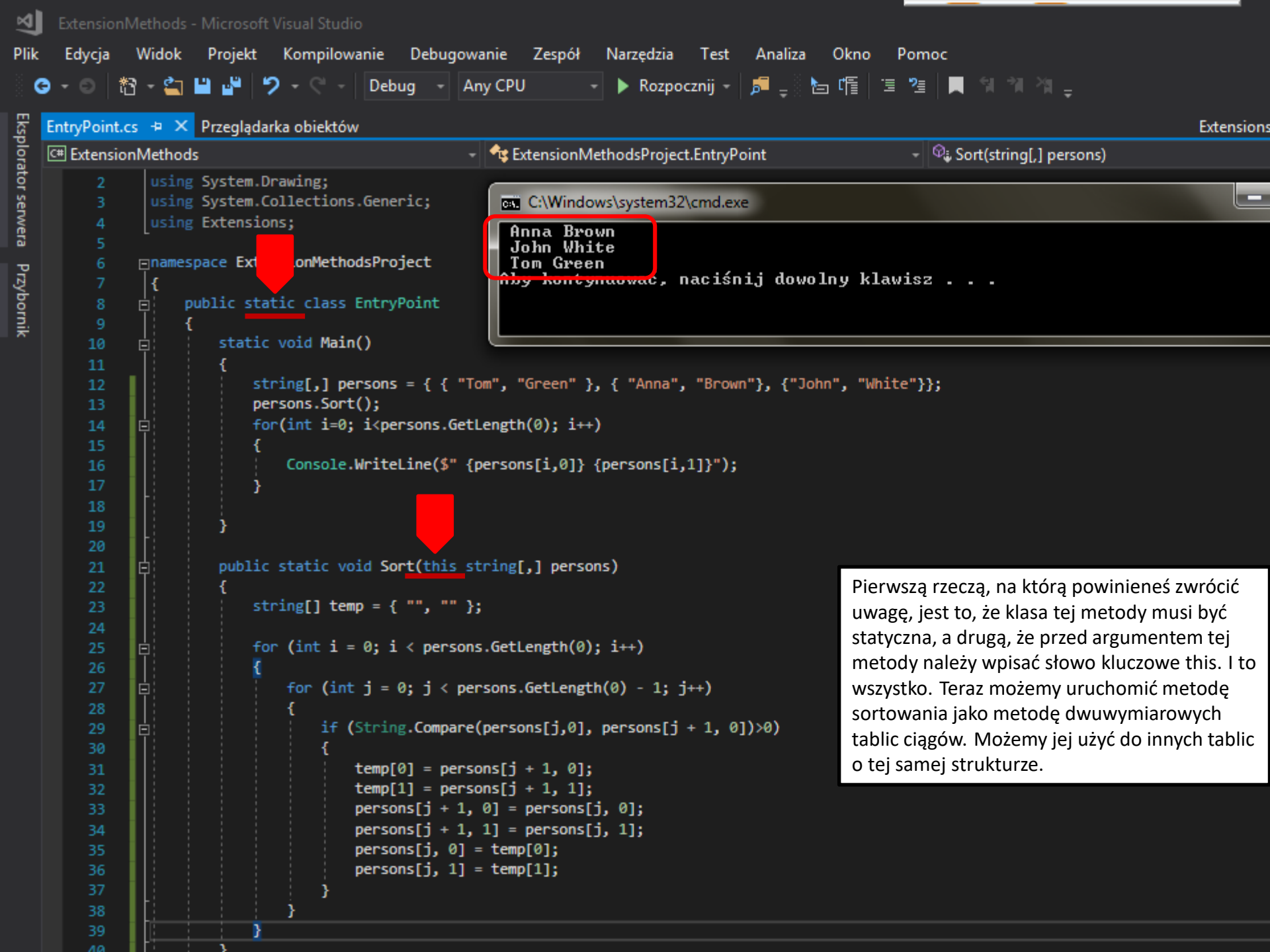
Mamy tablicę i chcemy posortować jej elementy. Jak widzimy, możemy to zrobić za pomocą metody Sortuj należącej do klasy array. Wynik jest poprawny.


ExtensionMethods - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
Rozpocznij
EntryPoint.cs
Przeglądarka obiektów
ExtensionMethods
ExtensionMethodsProject.Entry
Eksplorator serwera
Przybornik
1 using System;
2 using System.Drawing;
3 using System.Collections.Generic;
4 using Extensions;
5
6 namespace ExtensionMethodsProject
7 {
8 public static class EntryPoint
9 {
10 static void Main()
11 {
12 string[,] persons = { { "Tom", "Green" }, { "Anna", "Brown"}, {"John", "White" } };
13 // Array Class Sort
14 Array.Sort(persons);
15 Console.WriteLine("Sorted array: " + string.Join(", ", persons));
16 }
17
18
19 }
20 }
21
22
23
01. ExtensionMethods
Program 01. ExtensionMethods przestał działać.
Program przestał działać poprawnie z powodu wystąpienia problemu. System Windows zamknie program i powiadomi Cię, jeśli istnieje dostępne rozwiązanie.
Debuguj
Zamknij program
90 %
Ale jeśli chcemy Sortować bardziej złożone elementy tablicy i próbujemy to skompilować, pojawią się błędy. Więc musimy sami wykonać metodę sortowania.
Lista błędów
Całe rozwiązanie
0 Błędy
0 Ostrzeżenia
0 Komunikaty
Kompilacja + IntelliSense
Przeszukaj listę błędów
Kod
Opis
Projekt
Plik
W...
Stan pominię

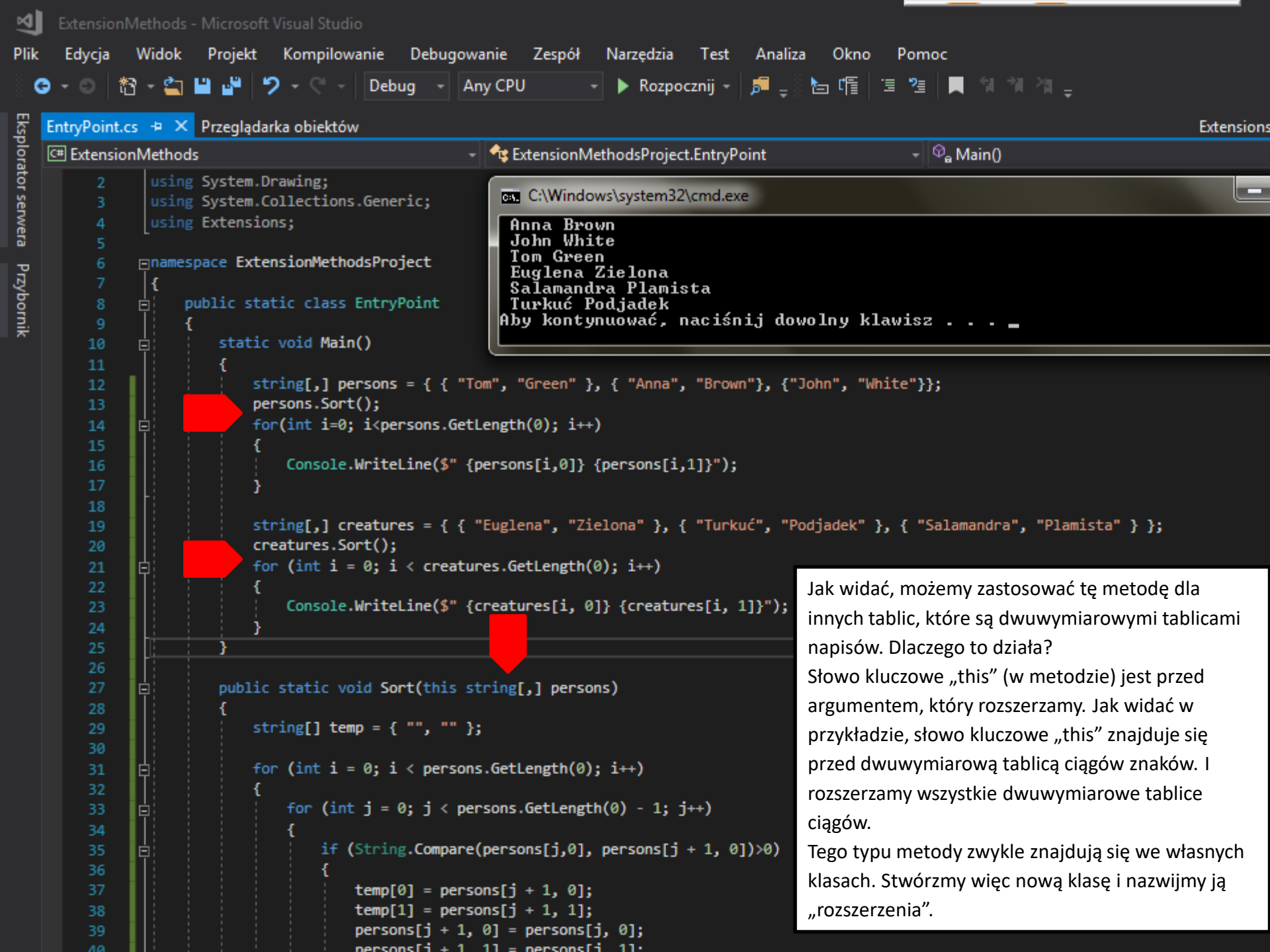


Mamy tutaj metodę sortowania bąbelkowego, która porównuje osoby i sortuje według imienia.
I wynik: Anna Brown, John White i Tom Green.
Widać, że są posortowane


ExtensionMethods - Microsoft Visual Studio
Plik Edycja Widok Projekt Kompilowanie Debugowanie Zespół Narzędzia Test Analiza Okno Pomoc
Debug
Any CPU
Rozpocznij
EntryPoint.cs*
Przeglądarka obiektów
ExtensionMethods
ExtensionMethodsProject.EntryPoint
Main()
using System.Drawing;
using System.Collections.Generic;
using Extensions;
namespace ExtensionMethodsProject
{
public static class EntryPoint
{
static void Main()
{
string[,] persons = { { "Tom", "Green" }, { "Anna", "Brown"}, {"John", "White"}};
persons.Sort();
for(int i=0; i<persons.GetLength(0); i++)
{
Console.WriteLine($" {persons[i,0]} {persons[i,1]}");
}
}
public static void Sort(string[,] persons)
{
string[] temp = { "", "" };
for (int i = 0; i < persons.GetLength(0); i++)
{
for (int j = 0; j < persons.GetLength(0) - 1; j++)
{
if (String.Compare(persons[j,0], persons[j + 1, 0])>0)
90 %
A co jeśli chcemy mieć Sortuj jako metodę tablicy i pisać „osoby kropka Sortuj”.
Ale jak widzisz, występuje błąd, ponieważ w tablicach nie ma metody sortowania.
Jak możemy to naprawić?
To jest bardzo proste.
Powinniśmy zatroszczyć się o dwie rzeczy.
Lista błędów
Całe rozwiązanie
1 Błąd
0 Ostrzeżenia
0 Komunikaty
Kompilacja + IntelliSense
Przeszukaj listę błędów
Kod
Opis
Projekt
Plik
W...
Stan pominię
CS1501
Żadne przeciążenie metody „Sort” nie pobiera następującej liczby argumentów: „0”
ExtensionMethods
EntryPoint.cs
13
Aktywne



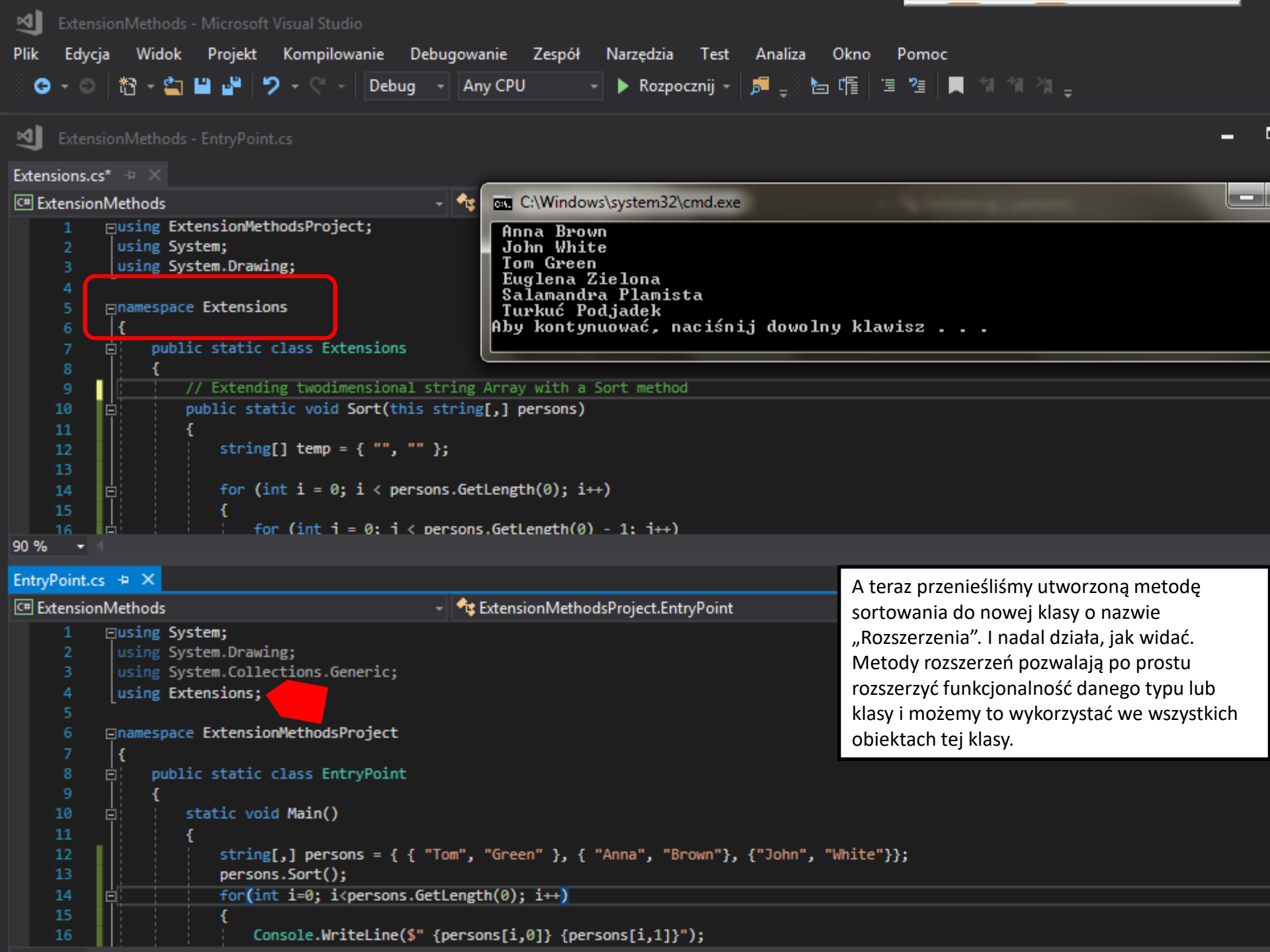
Pierwszą rzeczą, na którą powinieneś zwrócić uwagę, jest to, że klasa tej metody musi być statyczna, a drugą, że przed argumentem tej metody należy wpisać słowo kluczowe this. I to wszystko. Teraz możemy uruchomić metodę sortowania jako metodę dwuwymiarowych tablic ciągów. Możemy jej użyć do innych tablic o tej samej strukturze.



Jak widać, możemy zastosować tę metodę dla innych tablic, które są dwuwymiarowymi tablicami napisów. Dlaczego to działa?

Słowo kluczowe „this” (w metodzie) jest przed argumentem, który rozszerzamy. Jak widać w przykładzie, słowo kluczowe „this” znajduje się przed dwuwymiarową tablicą ciągów znaków. I rozszerzamy wszystkie dwuwymiarowe tablice ciągów.

Tego typu metody zwykle znajdują się we własnych klasach. Stwórzmy więc nową klasę i nazwijmy ją „rozszerzenia”.



A teraz przenieśliśmy utworzoną metodę sortowania do nowej klasy o nazwie „Rozszerzenia”. I nadal działa, jak widać. Metody rozszerzeń pozwalają po prostu rozszerzyć funkcjonalność danego typu lub klasy i możemy to wykorzystać we wszystkich obiektach tej klasy.

Multiple argument extension methods

Ok. A jeśli chcemy posortować tablicę osób, ale według nazwiska. Możemy użyć drugiego argumentu metody sortowania, aby ustawić te informacje.

ExtensionMethods - EntryPoint.cs

Extensions.cs

ExtensionMethods | Extensions.Extensions | Sort(s

```
1   using ExtensionMethodsProject;
2   using System;
3   using System.Drawing;
4
5   namespace Extensions
6   {
7       public static class Extensions
8       {
9           // Extending twodimensional string Array with a Sort method
10          public static void Sort(this string[,] persons, int index)
11          {
12              string[] temp = { "", "" };
13
14              for (int i = 0; i < persons.GetLength(0); i++)
15              {
16                  for (int j = 0; j < persons.GetLength(0) - 1; j++)
17                  {
18                      if (String.Compare(persons[j, index], persons[j + 1, index]) > 0)
19                      {
```

```
C:\Windows\system32\cmd.exe
Anna Brown
John White
Tom Green
Salamandra Plamista
Turkuć Podjadek
Euglena Zielona
Aby kontynuować, naciśnij dowolny klawisz . . .
```

90 %

EntryPoint.cs

ExtensionMethods | ExtensionMethodsProject.EntryPoint | Main

```
11          {
12              string[,] persons = { { "Tom", "Green" }, { "Anna", "Brown"}, {"John", "White"}};
13              persons.Sort(0);
14              for(int i=0; i<persons.GetLength(0); i++)
15              {
16                  Console.WriteLine($" {persons[i,0]} {persons[i,1]}");
17              }
18
19              string[,] creatures = { { "Euglena", "Zielona" }, { "Turkuć", "Podjadek" }, { "Salamandra", "Plamista" } };
20              creatures.Sort(1);
21              for (int i = 0; i < creatures.GetLength(0); i++)
22              {
23                  Console.WriteLine($" {creatures[i, 0]} {creatures[i, 1]}");
24              }
25          }
26
27
28
```

To nie jest duży problem. Powinniśmy zmodyfikować metodę sortowania bąbelkowego w trzech miejscach i możemy użyć metody sortowania rozszerzenia z argumentem.

Używamy indeksu typu int jako drugiego parametru metody i zmieniamy warunek porównania w dwóch punktach.
Jak widać w przykładzie, pierwsza tablica jest sortowana według pierwszego napisu w każdym elemencie, a druga tablica jest sortowana według drugiego napisu (nazwisko)

ExtensionMethods - Extensions.cs

Extensions.cs

ExtensionMethods | Extensions.Extensions | Sort(

```
1   using ExtensionMethodsProject;
2   using System;
3   using System.Drawing;
4
5   namespace Extensions
6   {
7       public static class Extensions
8       {
9           // Extending twodimensional string Array with a Sort method
10          public static void Sort(this string[,] persons, int index = 0)
11          {
12              string[] temp = { "", "" };
13
14              for (int i = 0; i < persons.GetLength(0); i++)
15              {
16                  for (int j = 0; j < persons.GetLength(0) - 1; j++)
17                  {
18                      if (String.Compare(persons[j, index], persons[j + 1, index]) > 0)
19                      {
```

C:\Windows\system32\cmd.exe

```
Anna Brown
John White
Tom Green
Euglena Zielona
Salamandra Plamista
Turkuć Podjadek
Aby kontynuować, naciśnij dowolny klawisz . . .
```

90 %

EntryPoint.cs

ExtensionMethods | ExtensionMethodsProject.EntryPoint | Main

```
11          {
12              string[,] persons = { { "Tom", "Green" }, { "Anna", "Brown"}, {"John", "White"}};
13              persons.Sort();
14              for(int i=0; i<persons.GetLength(0); i++)
15              {
16                  Console.WriteLine($" {persons[i,0]} {persons[i,1]}");
17              }
18
19              string[,] creatures = { { "Euglena", "Zielona" }, { "Turkuć", "Podjadek" }, { "Salamandra", "Plamista" } };
20              creatures.Sort(0);
21              for (int i = 0; i < creatures.GetLength(0); i++)
22              {
23                  Console.WriteLine($" {creatures[i, 0]} {creatures[i, 1]}");
24              }
25          }
26
27
28
```

Jeśli chcemy, aby drugi argument był opcjonalny, powinniśmy zapisać dla niego wartość domyślną. I jest to wartość 0. A teraz możemy spróbować uruchomić sortowanie z argumentem 0 i bez argumentów. Jak widać, wynikiem jest tablica posortowana według imion w obu przypadkach.

ExtensionMethods - EntryPoint.cs

Extensions.cs

ExtensionMethods | Extensions.Extensions | Sort(i

```
37          }
38      }
39
40      // Extending twodimensional string Array with Reverse method
41      public static void Reverse(this string[,] array)
42      {
43          string[] temp = { "", "" };
44          for (int i = 0, j = array.GetLength(0)-1; i < j; i++, j--)
45          {
46              temp[0] = array[i, 0];
47              temp[1] = array[i, 1];
48              array[i, 0] = array[j, 0];
49              array[i, 1] = array[j, 1];
50              array[j, 0] = temp[0];
51              array[j, 1] = temp[1];
52          }
53      }
54
55
```

90 %

Możemy stworzyć inną metodę, która odwróci kolejność osób. Zmienia pierwszy element z ostatnim, drugi z przedostatnim, i tak dalej. Wykonujemy metodę „reverse” i widzimy, że działa - wszystkie elementy są w odwrotnej kolejności.
Ok. Ale co powinniśmy zrobić, jeśli chcemy sortować tablicę i opcjonalnie odwracać jej kolejność.

C:\Windows\System32\cmd.exe

```
John White
Anna Brown
Tom Green
Salamandra Plamista
Turkuć Podjadek
Euglena Zielona
Aby kontynuować, naciśnij dowolny klawisz . . .
```

EntryPoint.cs

ExtensionMethods

```
9       {
10          static void Main()
11          {
12              string[,] persons = { { "Tom", "Green" }, { "Anna", "Brown"}, {"John", "White"}};
13              persons.Reverse();
14              for(int i=0; i<persons.GetLength(0); i++)
15              {
16                  Console.WriteLine($" {persons[i, 0]} {persons[i, 1]}");
17              }
18
19              string[,] creatures = { { "Euglena", "Zielona" }, { "Turkuć", "Podjadek" }, { "Salamandra", "Plamista" } };
20              creatures.Reverse();
21              for (int i = 0; i < creatures.GetLength(0); i++)
22              {
23                  Console.WriteLine($" {creatures[i, 0]} {creatures[i, 1]}");
24              }
25          }
26
```

ExtensionMethods - EntryPoint.cs

Extensions.cs

ExtensionMethods | Extensions.Extensions | Sort(

```
7   public static class Extensions
8   {
9       // Extending twodimensional string Array with a Sort method
10      public static void Sort(this string[,] persons, int index = 0, bool reverse = false)
11      {
12          persons.Sort(index);
13          if (reverse)
14          {
15              persons.Reverse();
16          }
17      }
18
19      public static void Sort(this string[,] persons, int index = 0)
20      {
21          string[] temp = { "", "" };
22
23          for (int i = 0; i < persons.GetLength(0); i++)
24          {
25              for (int j = 0; j < persons.GetLength(0) - 1; j++)
```

Możemy przeładować metodę sortowania. Jak widać, istnieją dwie metody sortowania. Różnią się, ponieważ mają różne argumenty.
Po wywołaniu metody, zostanie wykonana ta, która ma dokładnie ten sam typ argumentów.

```
C:\Windows\system32\cmd.exe
Tom Green
John White
Anna Brown
Salamandra Plamista
Turkuć Podjadek
Euglena Zielona
Aby kontynuować, naciśnij dowolny klawisz . . .
```

90 %

EntryPoint.cs

ExtensionMethods | ExtensionMethodsProject.EntryPoint | Main

```
4   using Extensions;
5
6   namespace ExtensionMethodsProject
7   {
8       public static class EntryPoint
9       {
10          static void Main()
11          {
12              string[,] persons = { { "Tom", "Green" }, { "Anna", "Brown"}, {"John", "White"}};
13              persons.Sort(0, true);
14              for(int i=0; i<persons.GetLength(0); i++)
15              {
16                  Console.WriteLine($" {persons[i, 0]} {persons[i, 1]}");
17              }
18
19              string[,] creatures = { { "Euglena", "Zielona" }, { "Turkuć", "Podjadek" }, { "Salamandra", "Plamista" } };
20              creatures.Sort(1);
21              for (int i = 0; i < creatures.GetLength(0); i++)
```

Teraz najpierw wywołujemy metodę Sortuj dla imienia i malejącej kolejności (pierwsza implementacja metody).
I po drugie dla drugiego imienia i bez argumentów na temat kolejności (druga implementacja metody).
Możesz przekazać wiele argumentów i przeciążyć te metody.

Extending classes that we can't modify

Kolejny przykład - rozszerzamy klasę.
Możemy zastosować to samo podejście do rozszerzenia dowolnych klas; czasem ma to sens, a czasem nie.
Czasami po prostu lepiej i łatwiej jest po prostu stworzyć metodę w tej klasie, ale nie zawsze jest do niej dostęp aby rozszerzyć.
Zwłaszcza jeśli używamy frameworków lub bibliotek innych osób.

Weźmy na przykład klasę Point.
Jest to wbudowana klasa C-sharp, której możemy używać, ale nie mamy do niej dostępu, ponieważ potrzebujemy specjalnego odniesienia do niej: „System.Drawing”
Point to po prostu klasa, która pozwala nam tworzyć punkty w przestrzeni.
Point to po prostu klasa, która pozwala nam zapisać współrzędne x i y.
Utworzyliśmy dwa punkty - pierwszy o współrzędnych 20, 20 i drugi o współrzędnych 50 i 60.

A teraz rozszerzmy klasę Punkt i nadajmy jej metodę, która mówi nam, jaka jest odległość między dwoma punktami.
W tym celu została utworzona nowa klasa zwana odległością.
Po prostu przechowuje dwie wartości odległość x i odległość y.
Stworzono więc nową metodę o nazwie DistanceTo. Ta metoda zwróci odległość - obiekt klasy Odległość..

I w tym obiekcie mamy dwie właściwości Xdistance i Ydistance
Stworzyliśmy też metodę DistanceTo. Ma dwa punkty jako argumenty. Przed pierwszym argumentem piszemy słowo kluczowe „this”- w celu rozszerzenia klasy Point.
Najpierw wydrukujemy to na konsoli.
Następnie obliczamy odległość między punktem 1 a punktem 2. Xdistance jest równy „punkt2.x - punkt1.x”, analogicznie Ydistance.

Jest to po prostu wiadomość, którą dostaniesz na konsoli.

Następnie zwracamy nową odległość i podajemy wartość odległości x i odległości y.
W porządku. Wykonujemy metodę piszącą DistanceTo jako metodę klasy punktowej. Metoda działa tak, jak to widać.

End of extension methods

I to jest koniec metod rozszerzeń. Czasami bardzo przydatne jest stworzenie metod rozszerzeń. I to wszystko.