

WZORCE PROJEKTOWE

SPRAWOZDANIE

ZADANIE BUILDER

Patryk Figas
Informatyka, programowanie
Grupa 34_Inf_P_NW_6

1. Cel

Celem dokumentu jest przedstawienie rozwiązania ćwiczenia polegającego na zastosowaniu wzorca projektowego **Builder**. Dokument opisuje strukturę zaprojektowanego systemu, zastosowany wzorec oraz wykonane działania.

Zaprojektowano i zaimplementowano klasy służące do tworzenia obiektu **Pizza** krok po kroku, wykorzystano interfejs budowniczy, klasę konkretnego budowniczego **ConcretePizzaBuilder**, klasę zarządzającą **Director** oraz klasę klienta **Program**.

W ramach ćwiczenia zaprojektowano klasę za pomocą „pseudokodu”, diagramu UML i implementacji klasy do programu oraz użycie jej w programie **Main**.

W rozwiązaniu zastosowano wzorec projektowy **Builder**, który umożliwia tworzenie obiektów poprzez wywoływanie kolejnych metod budujących poszczególne części obiektu.

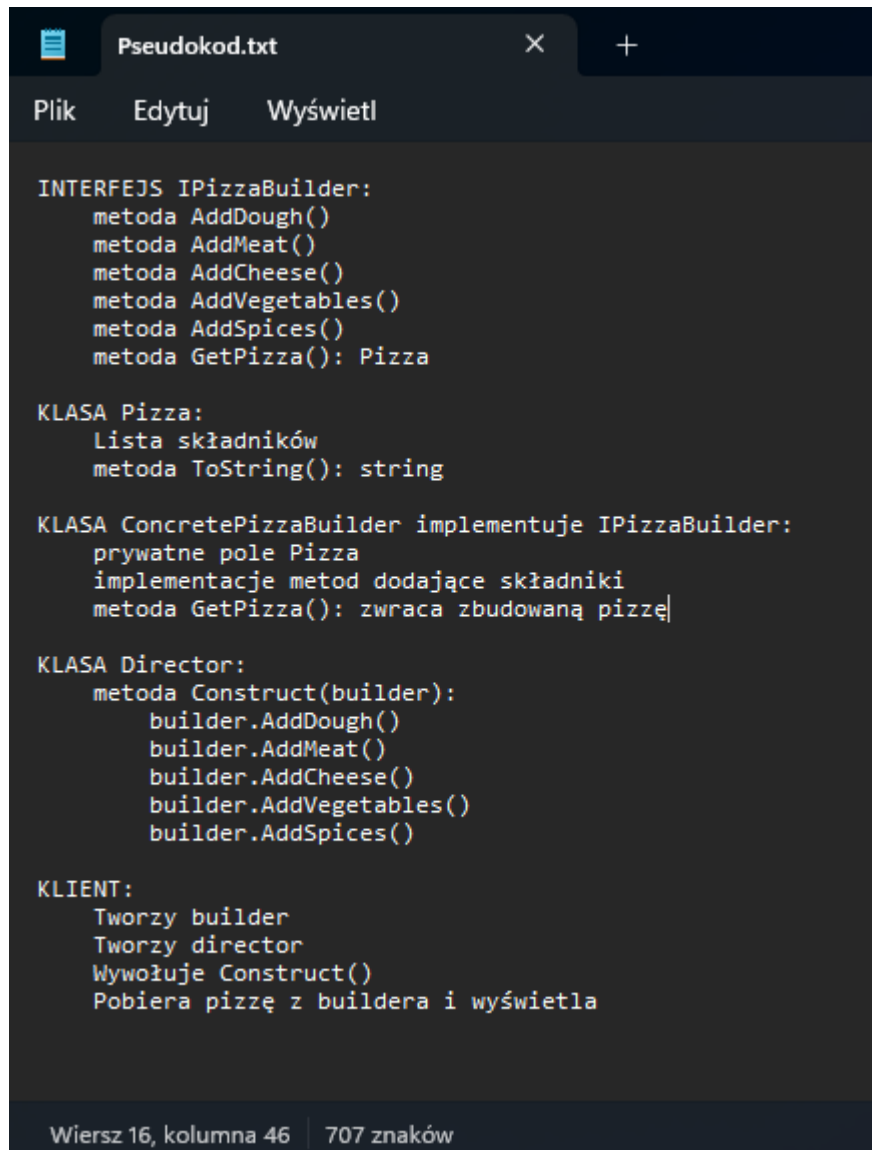
2. Opis rozwiązania

Zadanie polegało na zaimplementowaniu procesu zamawiania pizzy z wykorzystaniem wzorca projektowego **Builder**. Wzorec ten pozwala tworzyć obiekt krok po kroku, bez konieczności znajomości szczegółów jego budowy przez klienta. W naszym przypadku stworzono interfejs **IPizzaBuilder**, który definiuje metody budujące poszczególne elementy pizzy (ciasto, mięso, ser, warzywa, przyprawy). Klasa **ConcretePizzaBuilder** implementuje ten interfejs i rzeczywiście buduje pizzę, a klasa **Director** steruje procesem budowy. Na końcu klient (klasa **Program**) otrzymuje gotową pizzę.

- diagram klas programu



- pseudokod klas



```
INTERFEJS IPizzaBuilder:
    metoda AddDough()
    metoda AddMeat()
    metoda AddCheese()
    metoda AddVegetables()
    metoda AddSpices()
    metoda GetPizza(): Pizza

KLASA Pizza:
    Lista składników
    metoda ToString(): string

KLASA ConcretePizzaBuilder implementuje IPizzaBuilder:
    prywatne pole Pizza
    implementacje metod dodające składniki
    metoda GetPizza(): zwraca zbudowaną pizzę

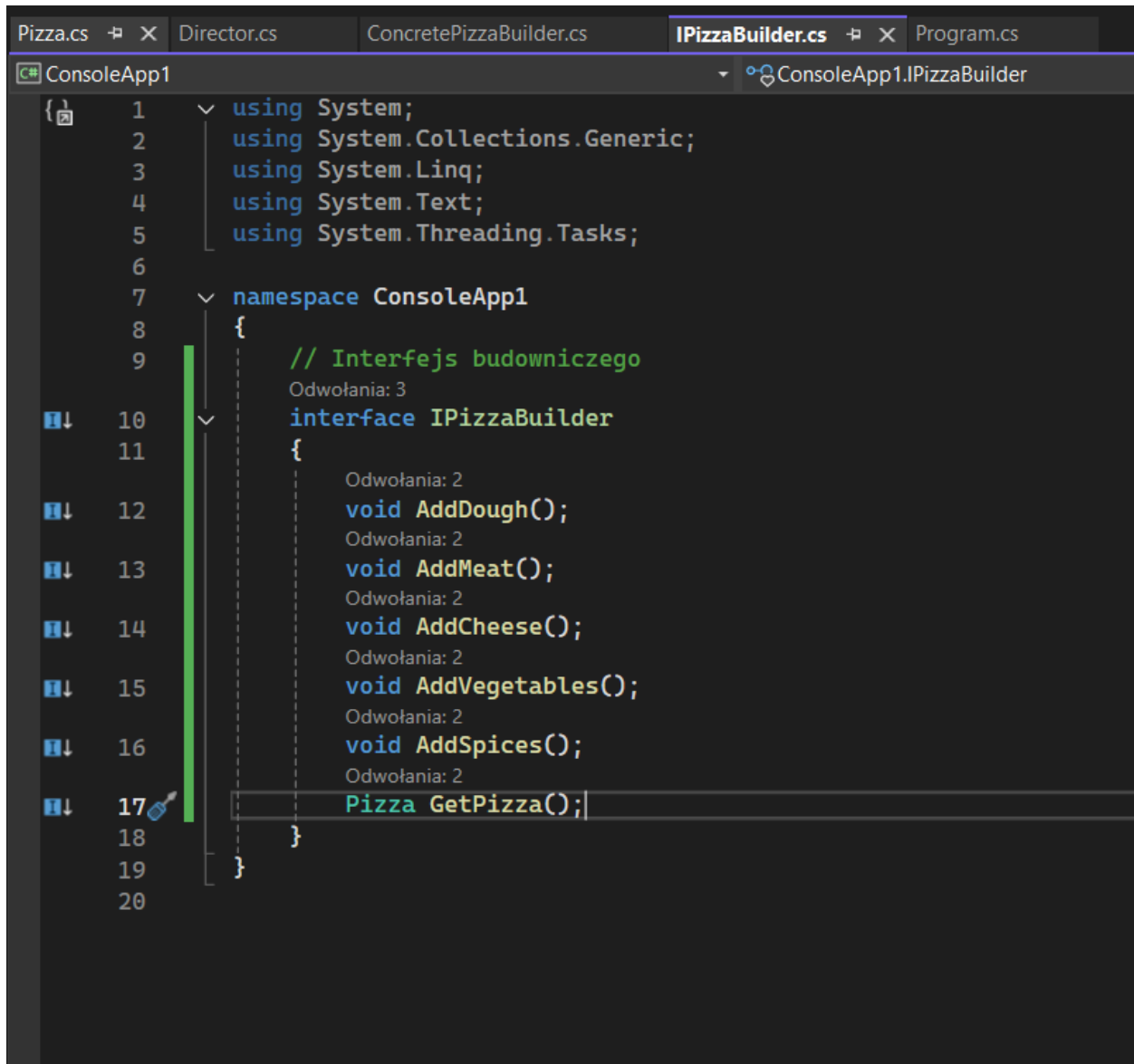
KLASA Director:
    metoda Construct(builder):
        builder.AddDough()
        builder.AddMeat()
        builder.AddCheese()
        builder.AddVegetables()
        builder.AddSpices()

KLIENT:
    Tworzy builder
    Tworzy director
    Wywołuje Construct()
    Pobiera pizzę z buildera i wyświetla
```

Wiersz 16, kolumna 46 | 707 znaków

3. Implementacja

- kod interfejsu **IPizzaBuilder.cs**

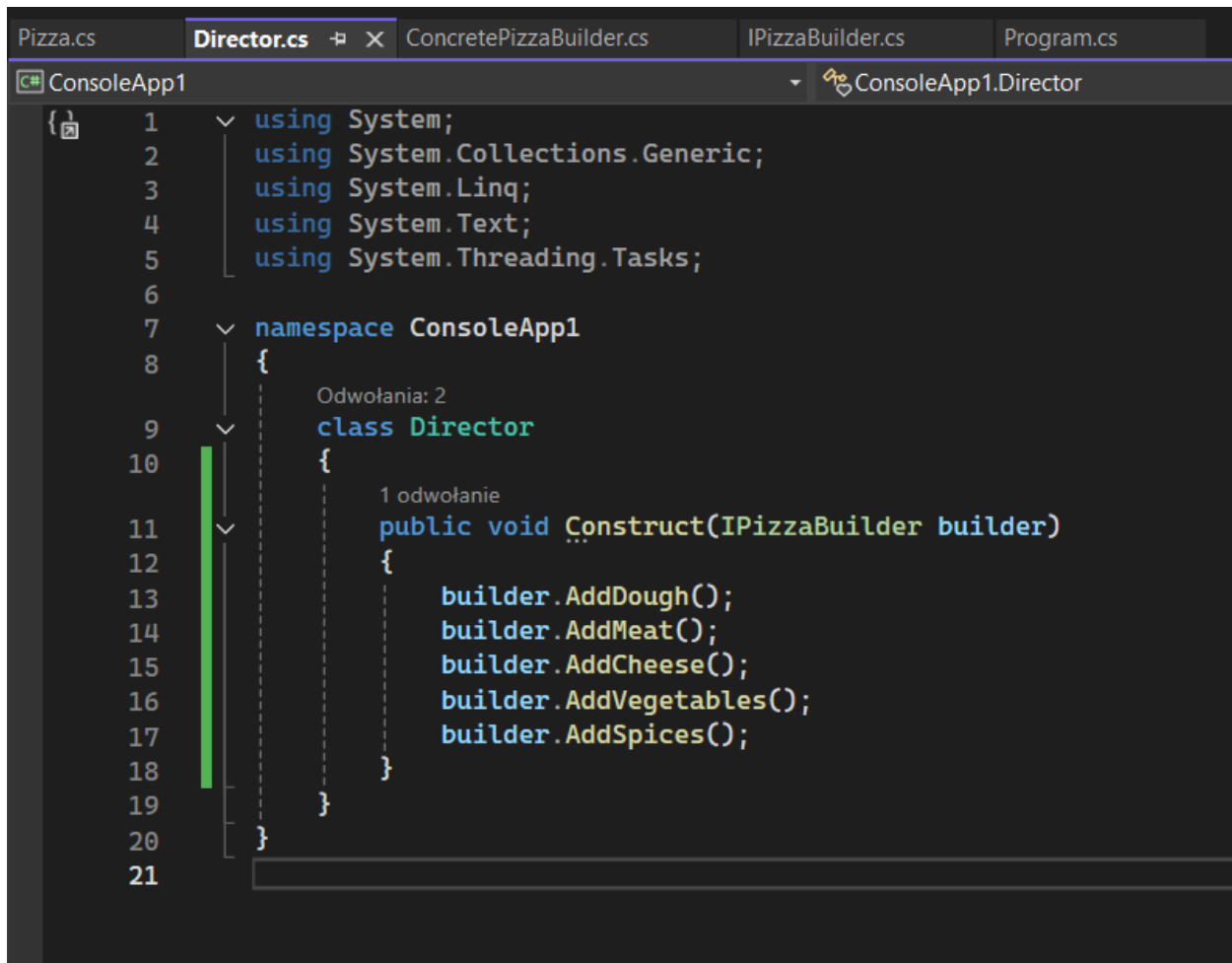


```
Pizza.cs  Director.cs  ConcretePizzaBuilder.cs  IPizzaBuilder.cs  Program.cs
ConsoleApp1
{
  1  using System;
  2  using System.Collections.Generic;
  3  using System.Linq;
  4  using System.Text;
  5  using System.Threading.Tasks;
  6
  7  namespace ConsoleApp1
  8  {
  9      // Interfejs budowniczego
 10      // Odwołania: 3
 11      interface IPizzaBuilder
 12      {
 13          // Odwołania: 2
 14          void AddDough();
 15          // Odwołania: 2
 16          void AddMeat();
 17          // Odwołania: 2
 18          void AddCheese();
 19          // Odwołania: 2
 20          void AddVegetables();
 21          // Odwołania: 2
 22          void AddSpices();
 23          // Odwołania: 2
 24          Pizza GetPizza();
 25      }
 26  }
```

- kod klasy **ConcretePizzaBuilder.cs**

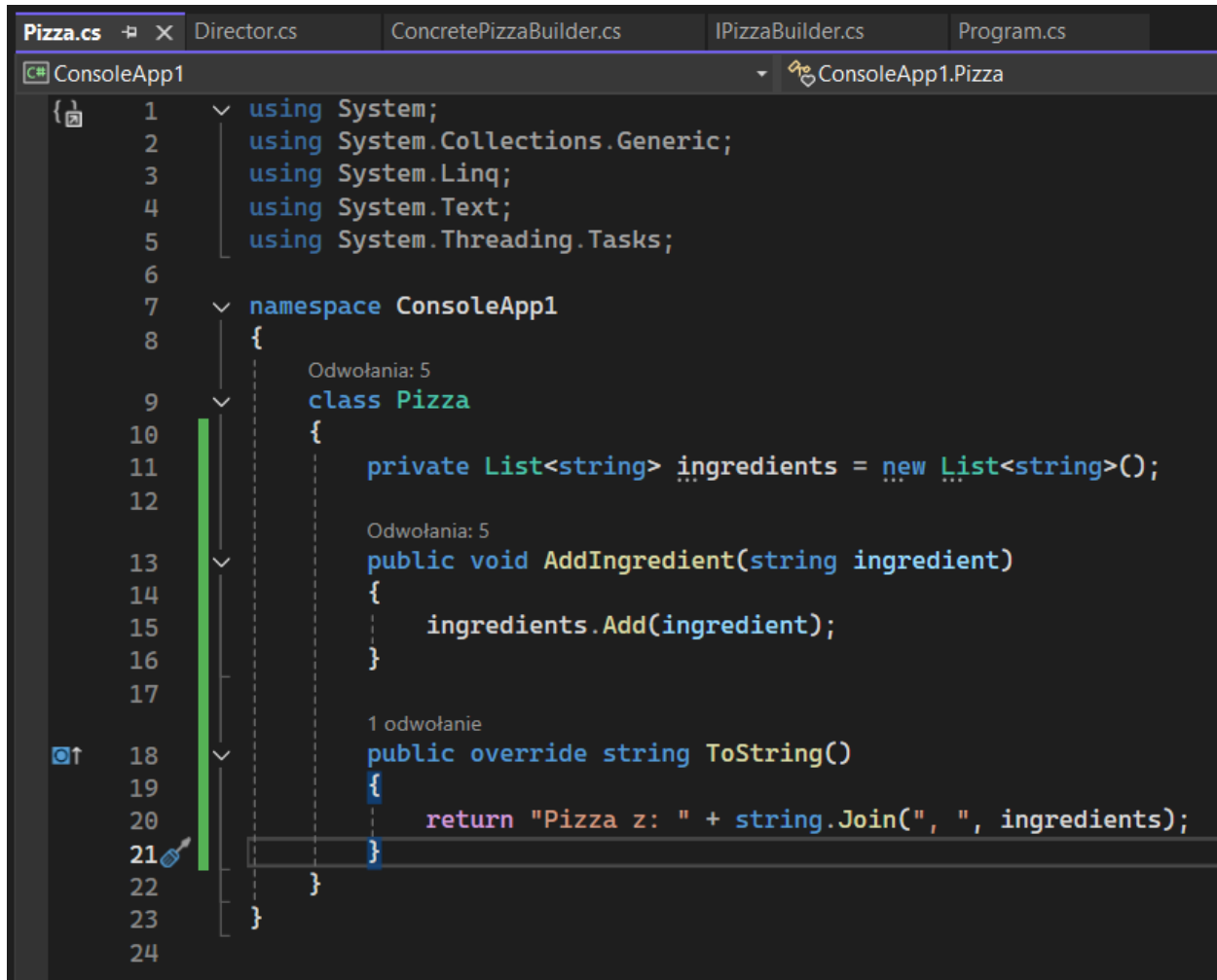
```
Pizza.cs    Director.cs    ConcretePizzaBuilder.cs    IPizzaBuilder.cs    Program.cs
C# ConsoleApp1
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp1
8  {
9      1 odwołanie
10     class ConcretePizzaBuilder : IPizzaBuilder
11     {
12         private Pizza pizza = new Pizza();
13
14         Odwołania: 2
15         public void AddDough()
16         {
17             pizza.AddIngredient("ciasto");
18         }
19
20         Odwołania: 2
21         public void AddMeat()
22         {
23             pizza.AddIngredient("salami");
24         }
25
26         Odwołania: 2
27         public void AddCheese()
28         {
29             pizza.AddIngredient("ser mozzarella");
30         }
31
32         Odwołania: 2
33         public void AddVegetables()
34         {
35             pizza.AddIngredient("papryka");
36         }
37
38         Odwołania: 2
39         public void AddSpices()
40         {
41             pizza.AddIngredient("oregano");
42         }
43
44         Odwołania: 2
45         public Pizza GetPizza()
46         {
47             return pizza;
48         }
49     }
50 }
```

- kod klasy **Director.cs**



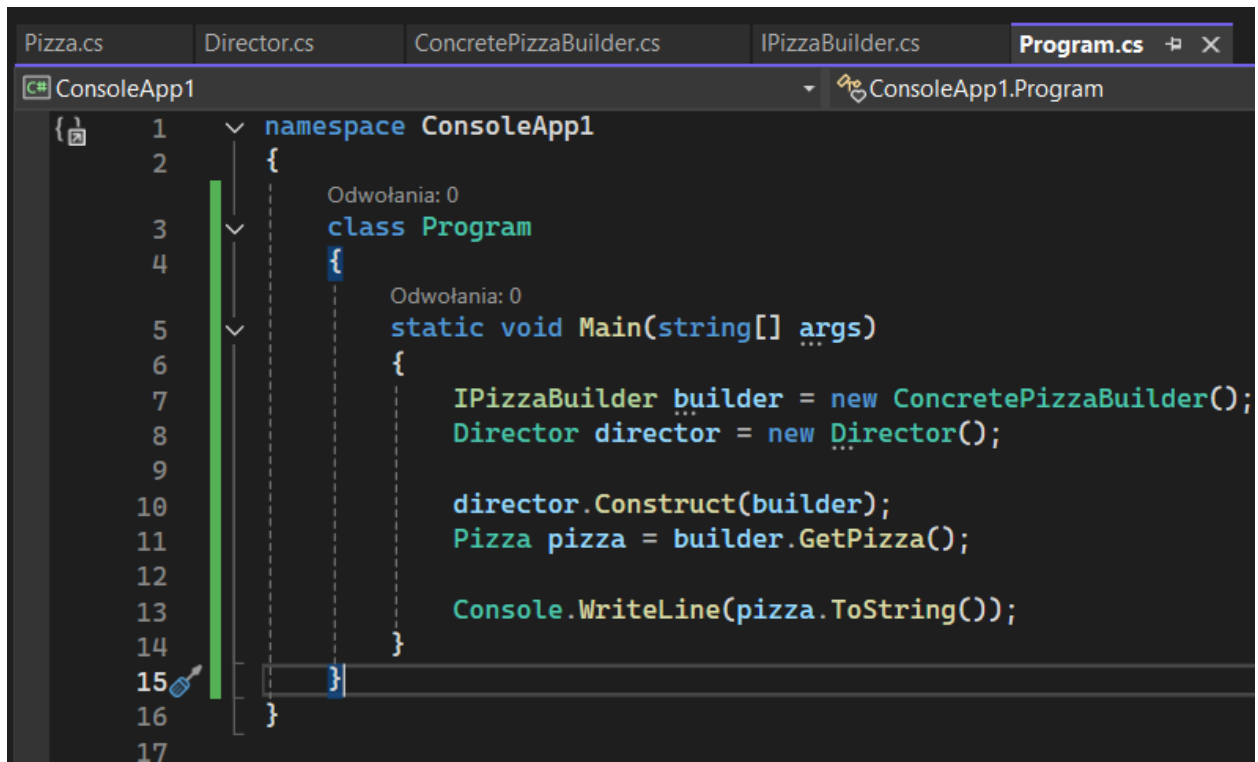
```
Pizza.cs | Director.cs | ConcretePizzaBuilder.cs | IPizzaBuilder.cs | Program.cs
C# ConsoleApp1 ConsoleApp1.Director
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp1
8  {
9      class Director
10     {
11         public void Construct(IPizzaBuilder builder)
12         {
13             builder.AddDough();
14             builder.AddMeat();
15             builder.AddCheese();
16             builder.AddVegetables();
17             builder.AddSpices();
18         }
19     }
20 }
21
```

- kod klasy **Pizza.cs**



```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp1
8  {
9      class Pizza
10     {
11         private List<string> ingredients = new List<string>();
12
13         public void AddIngredient(string ingredient)
14         {
15             ingredients.Add(ingredient);
16         }
17
18         public override string ToString()
19         {
20             return "Pizza z: " + string.Join(", ", ingredients);
21         }
22     }
23 }
24
```


- kod klasy **Program.cs**



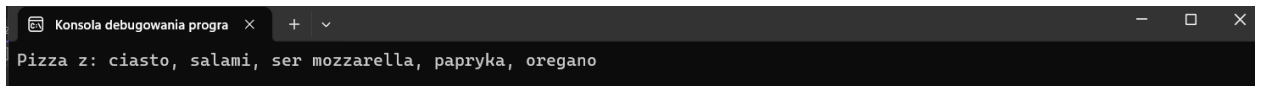
```
1 namespace ConsoleApp1
2 {
3     class Program
4     {
5         static void Main(string[] args)
6         {
7             IPizzaBuilder builder = new ConcretePizzaBuilder();
8             Director director = new Director();
9
10            director.Construct(builder);
11            Pizza pizza = builder.GetPizza();
12
13            Console.WriteLine(pizza.ToString());
14        }
15    }
16 }
17
```

4. Podsumowanie

Do rozwiązania zadania został użyty wzorec projektowy **Builder**, ponieważ idealnie pasuje on do sytuacji, w której obiekt (pizza) składa się z wielu części, a sposób jego tworzenia powinien być oddzielony od reprezentacji końcowego produktu. Builder pozwala budować różne warianty obiektu bez zmieniania kodu klienta.

Kod został podzielony na odpowiednie klasy i interfejsy. Działający program w poprawny sposób buduje pizzę krok po kroku i prezentuje rezultat w konsoli. W tym przypadku wzorec **Builder** jest najbardziej odpowiedni, ponieważ:

- pozwala oddzielić tworzenie pizzy od jej składników,
- umożliwia tworzenie różnych wariantów bez modyfikacji klas klienta i dyrektora.

A screenshot of a debug console window with a dark background. The title bar at the top reads "Konsola debugowania progra" followed by a close button (X) and a dropdown menu with a plus sign and a downward arrow. The main area of the console displays the text "Pizza z: ciasto, salami, ser mozzarella, papryka, oregano" in a light-colored monospace font.

Alternatywnie można rozważyć wzorec **Factory Method**, który również służy do tworzenia obiektów. Jednak **Factory Method** zwraca od razu **gotowy produkt** i **nie daje tak dużej kontroli** nad procesem **tworzenia krok po kroku**. Dlatego **Builder** w tej sytuacji jest **najlepszym wyborem**.

Lista załączników

Repozytorium GITHUB z projektem:

<https://github.com/PatrykFigas/Wzorce-projektowe.git>