Text

Description automatically generated with medium confidence

wzorce projektowe

Sprawozdanie

Zadanie builder

Patryk Figas

Informatyka, programowanie

Grupa 34\_Inf\_P\_NW\_6

Rok akademicki 2023/24

# Cel

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

Zaprojektowano i zaimplementowano klasy służące do tworzenia obiektu **Pizza** krok po kroku,

wykorzystano interfejs budowniczego, klasę konkretnego budowniczego **ConcretePizzaBuilder**, klasę zarządzającą **Director** oraz klasę klienta **Program.**

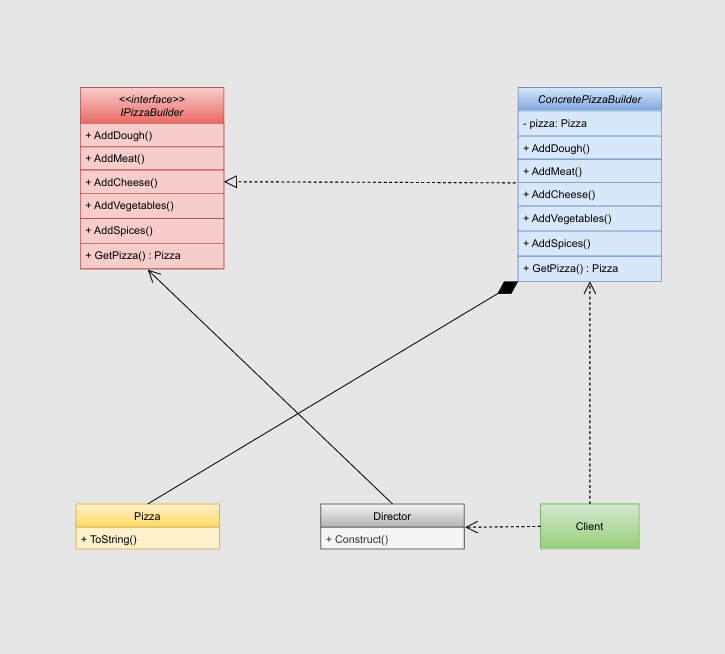
W ramach ćwiczenia zaprojektowano klasę za pomocą „pseudokodu”, diagramu UML i implementacji klasy do programu oraz użycie jej w programie **Main**.

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

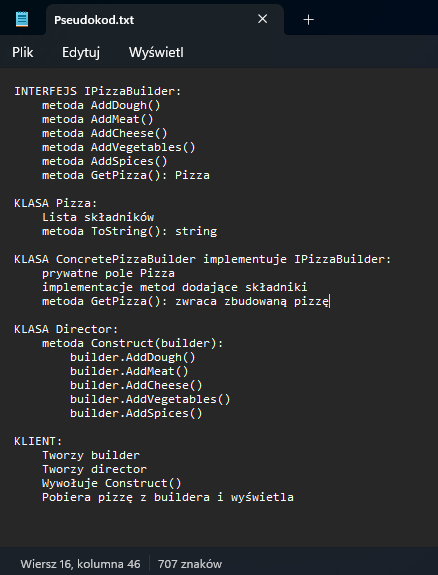
# Opis rozwiązania

Zadanie polegało na zaimplementowaniu procesu zamawiania pizzy z wykorzystaniem wzorca projektowego **Builder**. Wzorzec 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

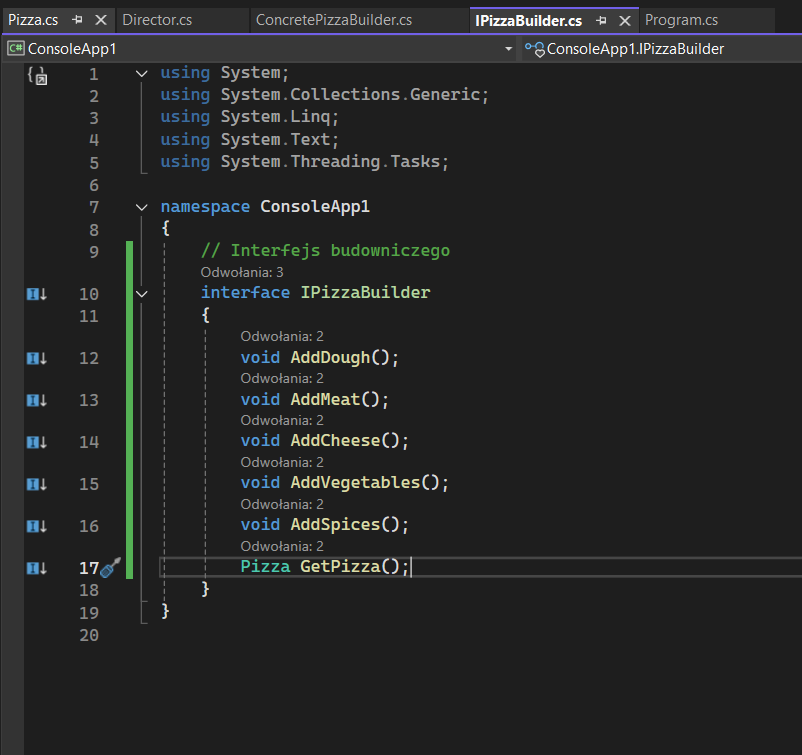


* pseudokod klas

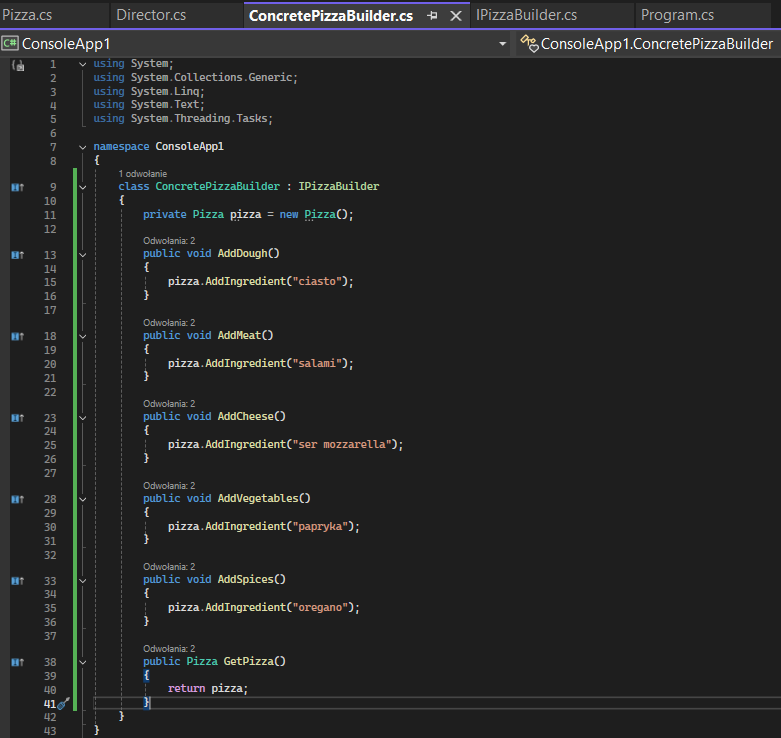


# Implementacja

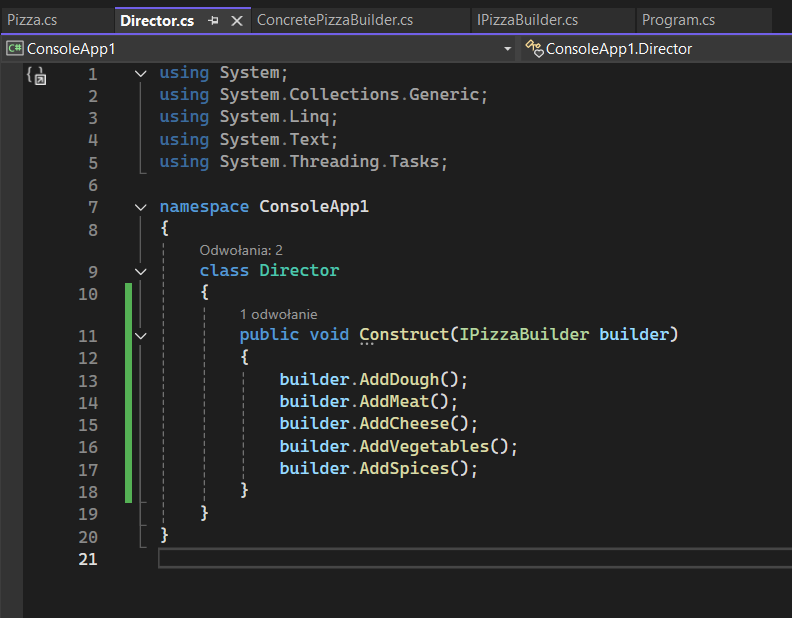
* kod interfejsu **IPizzaBuilder.cs**



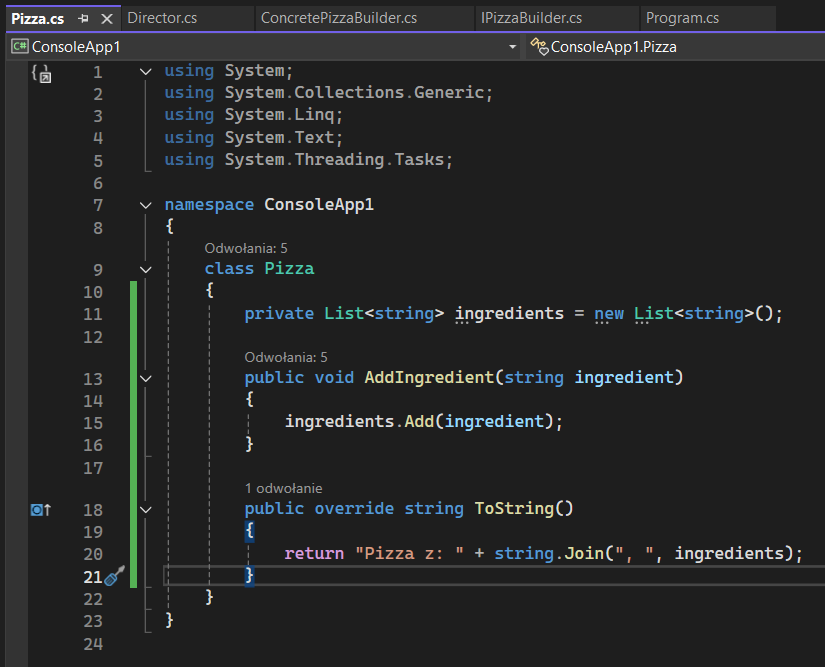
* kod klasy **ConcretePizzaBuilder.cs**



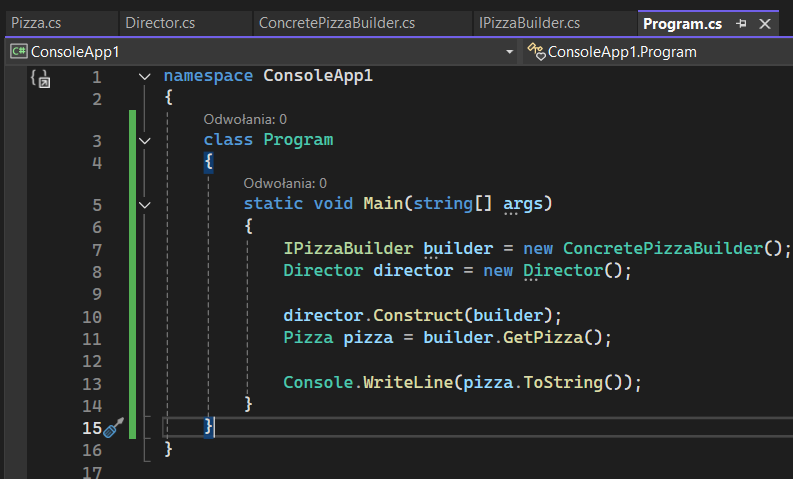
* kod klasy **Director.cs**



* kod klasy **Pizza.cs**



* kod klasy **Program.cs**



# Podsumowanie

Do rozwiązania zadania został użyty wzorzec 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 wzorzec **Builder** jest najbardziej odpowiedni, ponieważ:

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

Alternatywnie można rozważyć wzorzec **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

W tej części proszę podać listę załączników z kodem lub link do zdalnego repozytorium.