Text

Description automatically generated with medium confidence

wzorce projektowe

Sprawozdanie

ZADANIE decorator kawa

Patryk Figas

Informatyka, programowanie

Grupa 34\_Inf\_P\_NW\_6

Rok akademicki 2024/25

# Cel

Celem niniejszego dokumentu jest przedstawienie rozwiązania zadania polegającego na zaprojektowaniu elastycznego systemu zamawiania kawy, który umożliwia dynamiczne dodawanie dodatków do napoju (takich jak mleko, cukier czy czekolada), bez potrzeby tworzenia wielu wariantów klas.

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

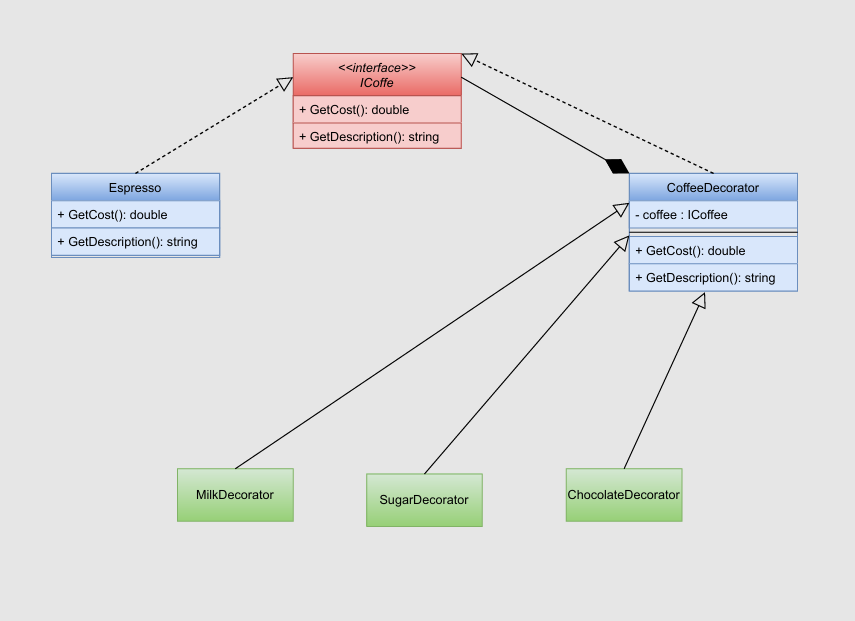
Zaimplementowano system oparty na wzorcu projektowym **Decorator**, który pozwala na rozszerzanie funkcjonalności obiektów w sposób zgodny z zasadą **Open/Closed**.

# Opis rozwiązania

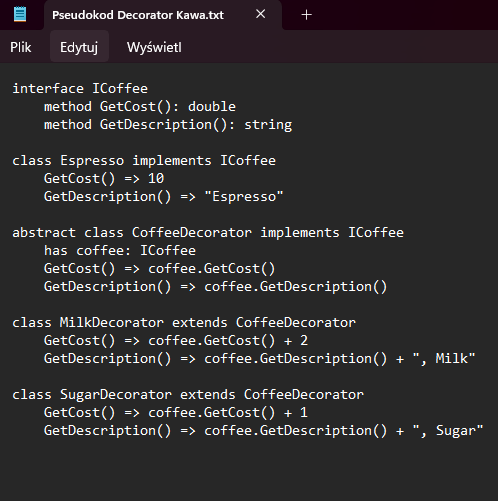
Rozwiązanie opiera się na **interfejsie ICoffee**, który definiuje podstawowe metody: **GetCost()** oraz **GetDescription()**. Klasa Espresso implementuje ten interfejs i reprezentuje bazowy napój.  
 Następnie wprowadzono abstrakcyjną klasę **CoffeeDecorator**, która również implementuje **ICoffee** i przechowuje referencję do innego obiektu typu **ICoffee**. Konkretne **klasy dekoratorów** **(MilkDecorator, SugarDecorator, ChocolateDecorator**) rozszerzają **CoffeeDecorator** i modyfikują koszt oraz opis kawy, dodając odpowiednie składniki.

Dzięki temu możliwe jest dynamiczne tworzenie napoju z dowolną liczbą dodatków, bez mnożenia klas i łamania zasad SOLID.

* Propozycja diagramu klas

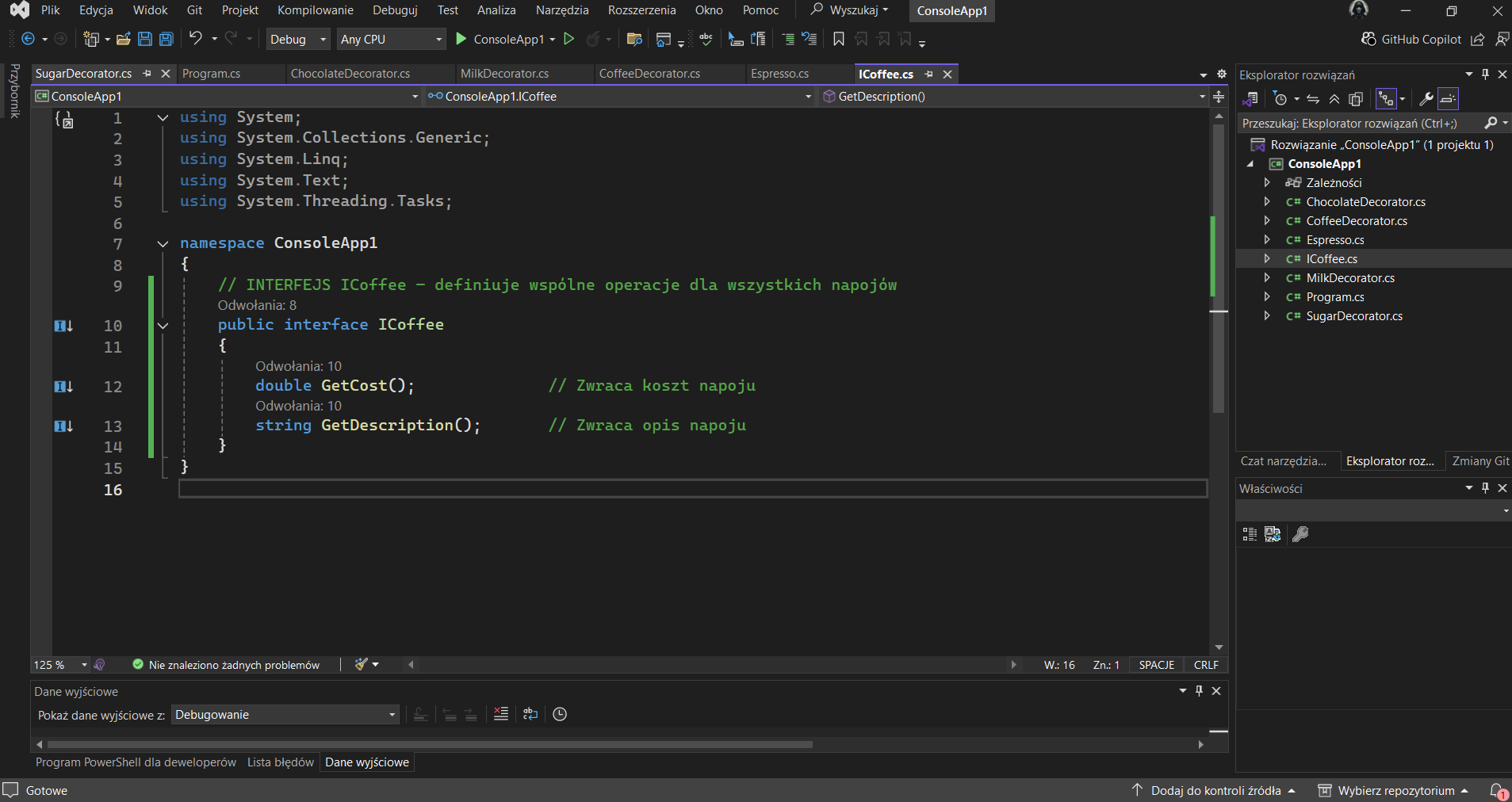


* Pseudokod

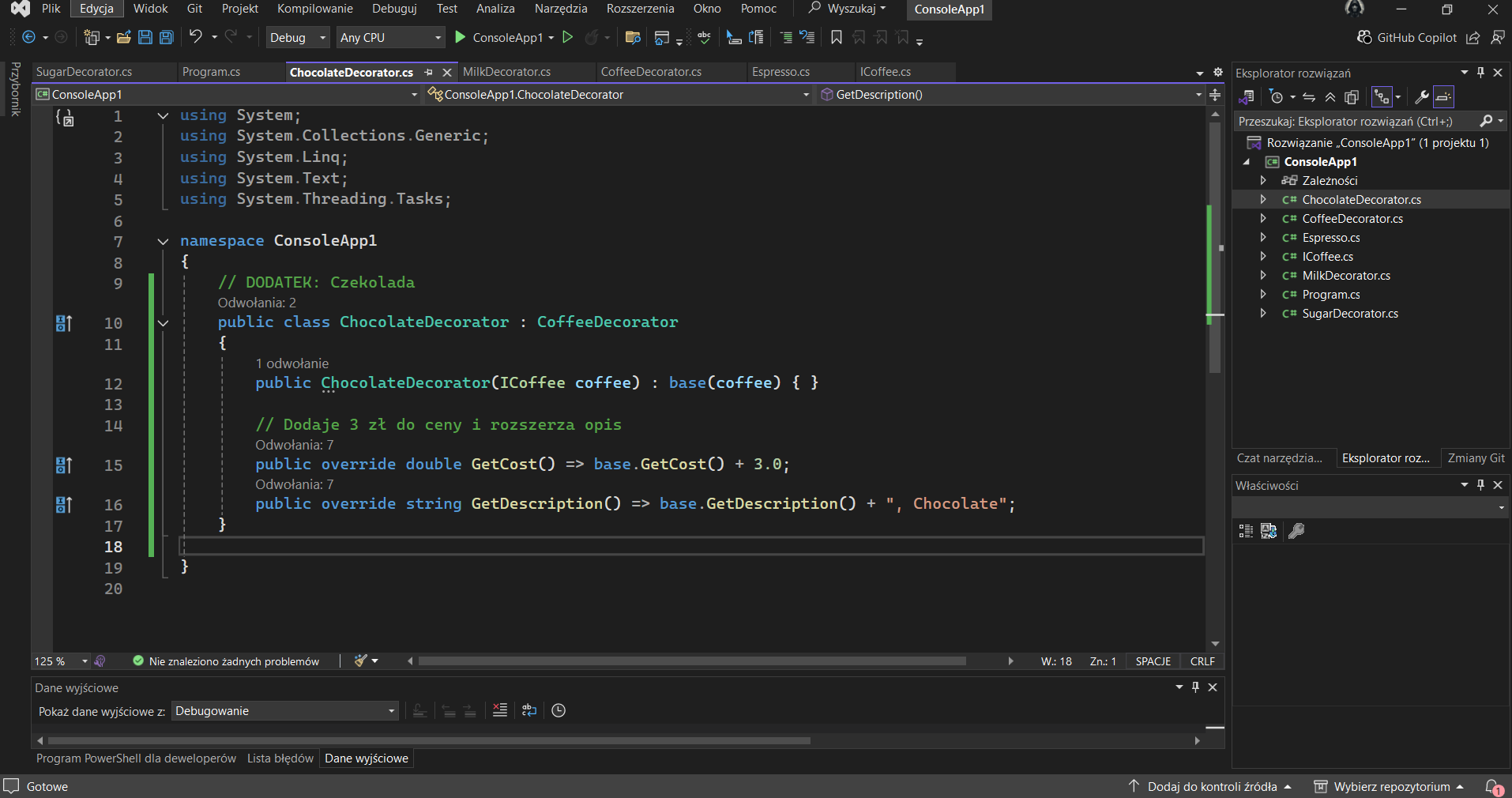


# Implementacja

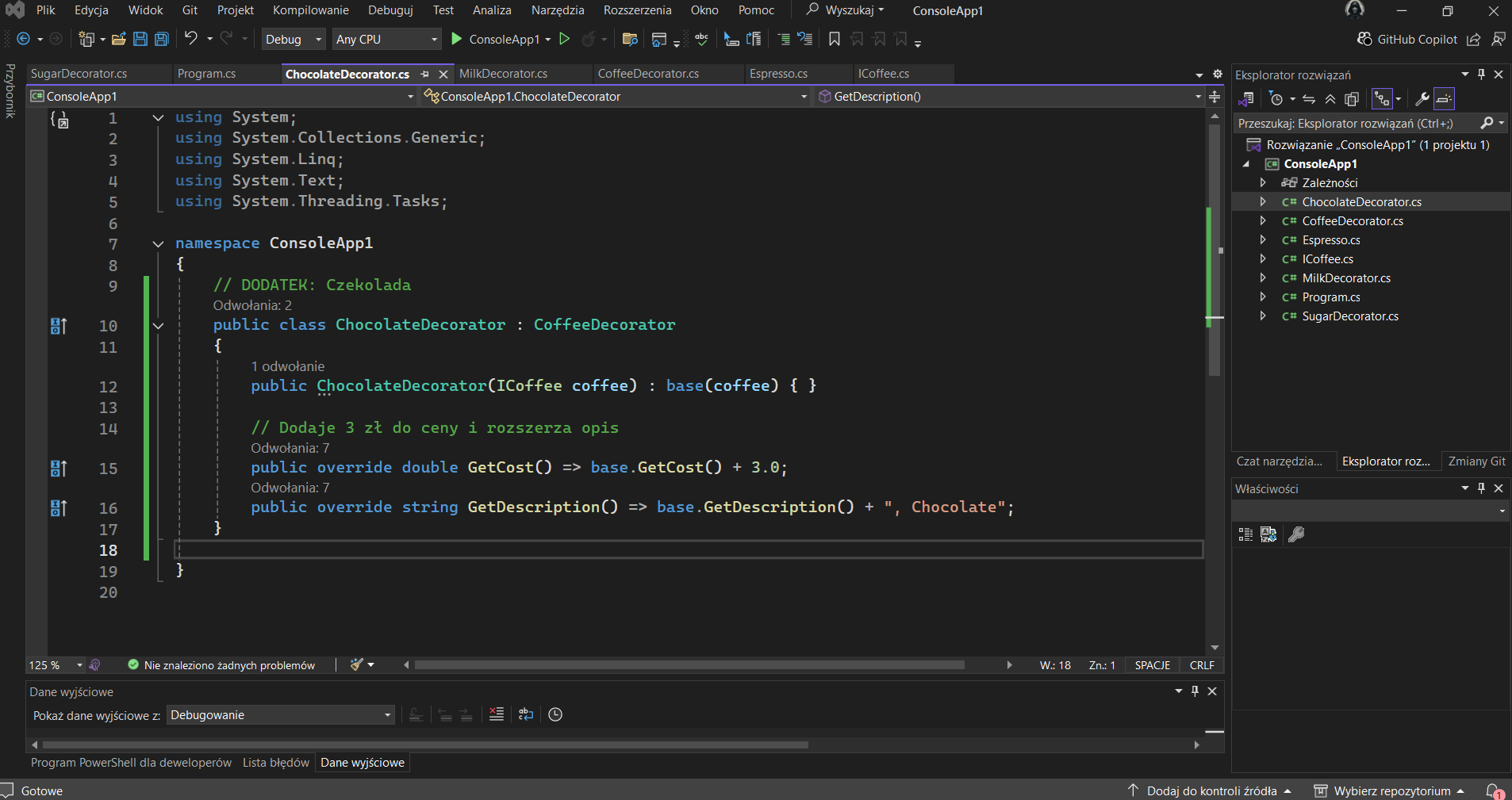
* Kod **interfejsu ICoffee.cs**



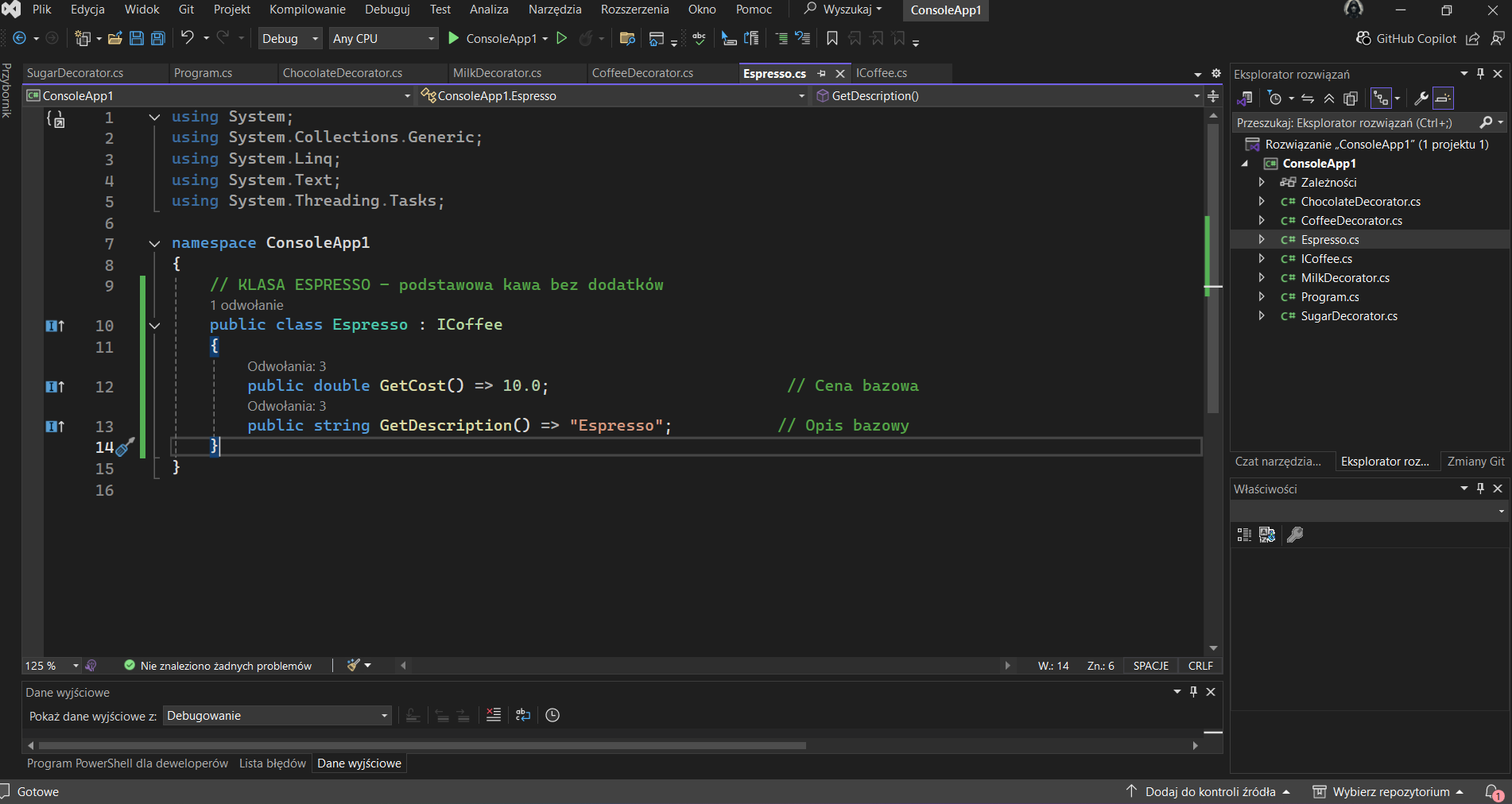
* Kod **klasy ChocolateDecorator.cs**



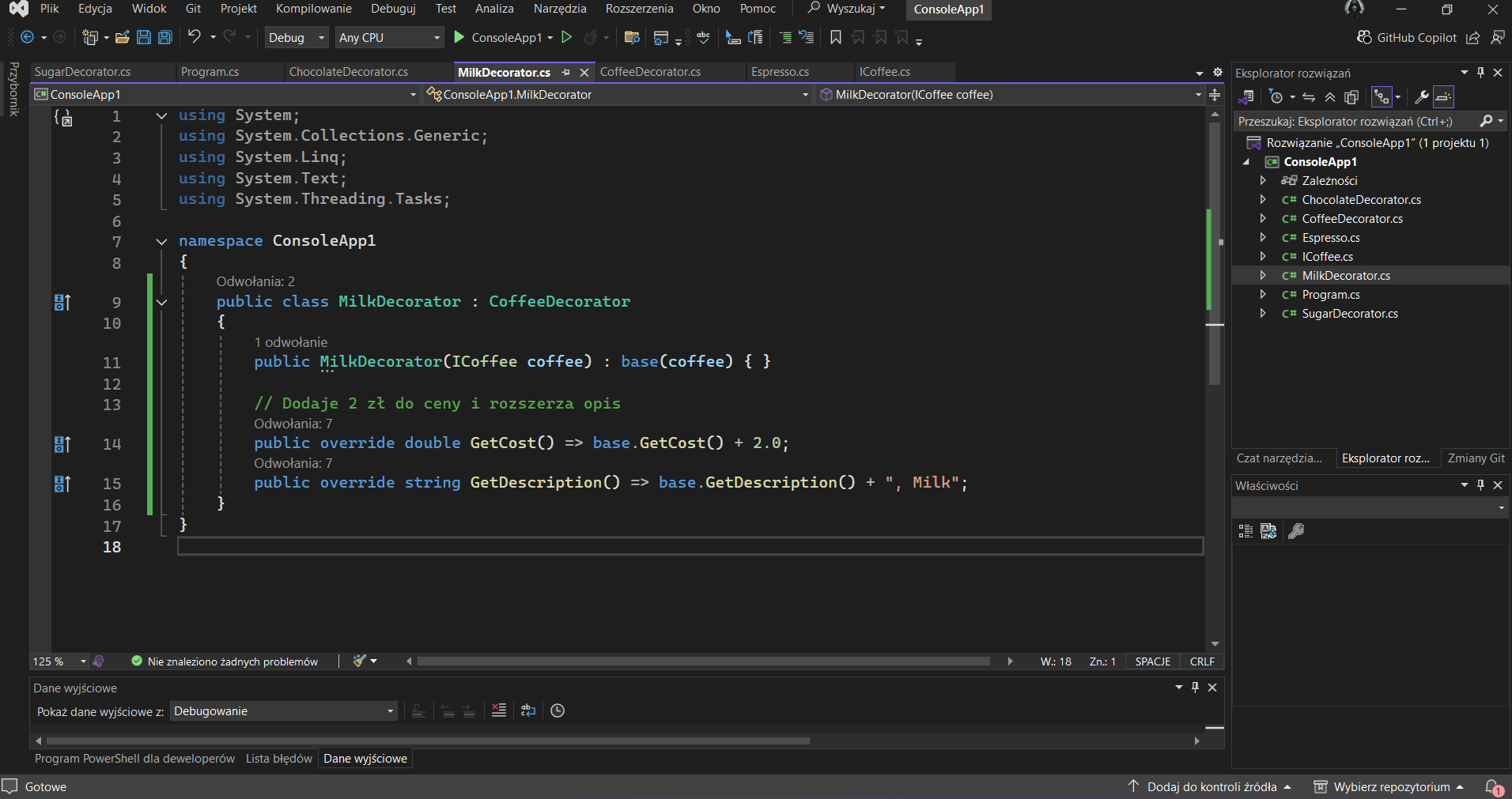
* Kod **klasy CoffeeDecorator.cs**



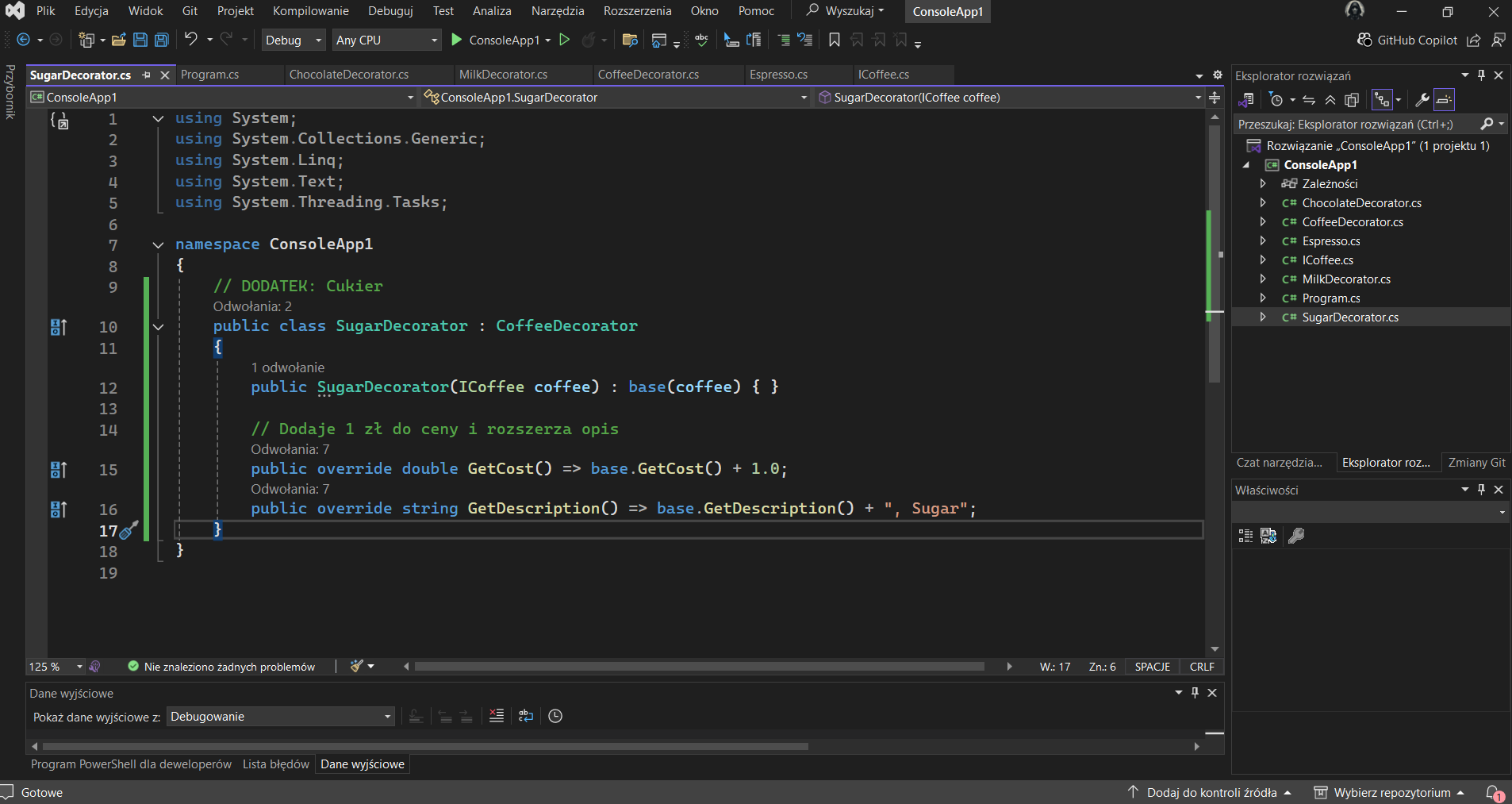
* Kod **klasy Espresso.cs**



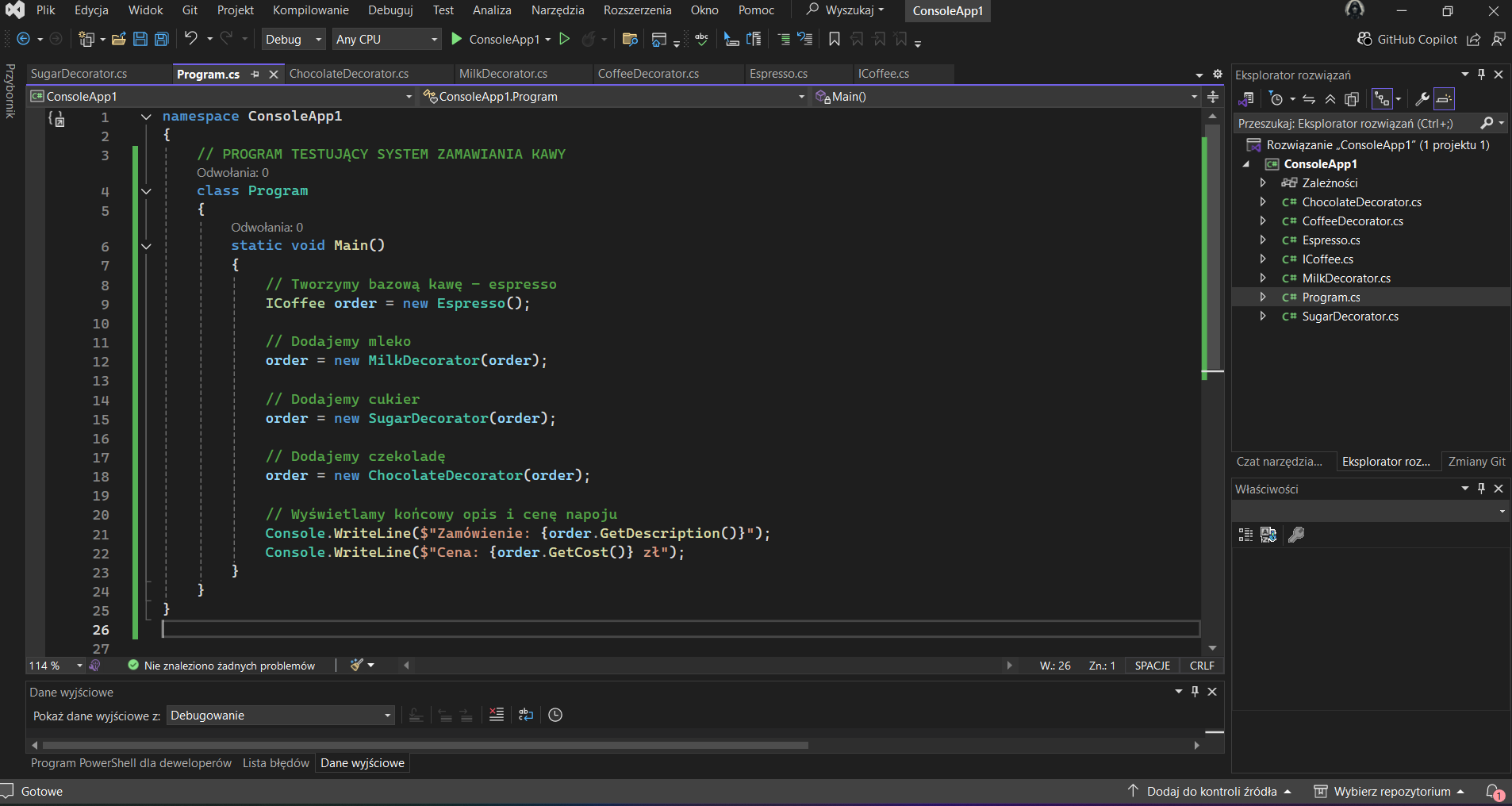
* Kod **klasy MilkDecorator.cs**



* Kod **klasy SugarDecorator.cs**



* Kod **klasy** **Program.cs**

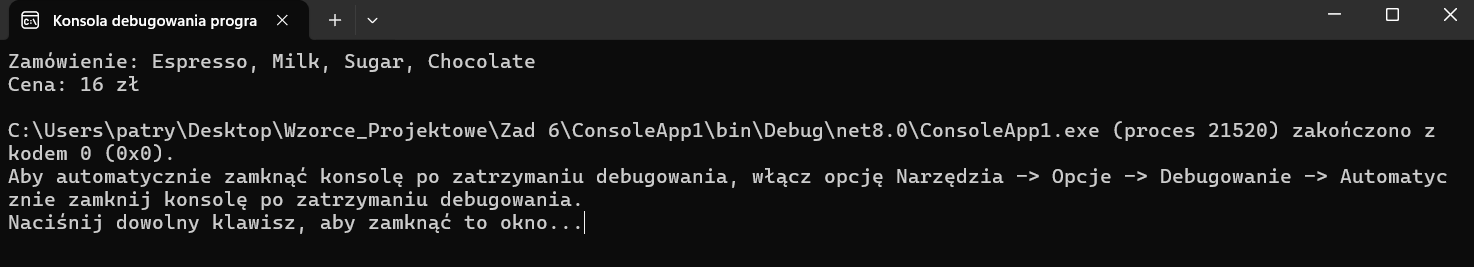


# Podsumowanie

W zadaniu zastosowano wzorzec projektowy **Decorator**, który pozwala na dynamiczne rozszerzanie funkcjonalności obiektów – w tym przypadku: dodawanie składników do kawy. Dzięki niemu nie trzeba tworzyć wielu klas dla każdej możliwej kombinacji dodatków.

**Implementacja powiodła się**, ponieważ:

* system działa zgodnie z założeniami,
* spełnia zasadę **Open/Closed** (rozszerzanie bez modyfikowania),
* umożliwia łatwe dodawanie nowych dekoratorów.



**Inny wzorzec projektowy** – np. **Builder** – mógłby się sprawdzić przy bardziej złożonych zamówieniach, np. z konfiguracją przez użytkownika. Jednak dla potrzeb prostego rozszerzania funkcji obiektu **Decorator jest najlepszym wyborem**.

# Lista załączników

Repozytorium GITHUB z projektem:

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