Kompozyt

Kamil Niewiara

Wydział Matematyki, Fizyki i Informatyki
Politechnika Krakowska im. Tadeusza Kościuszki

5 lipca 2019

Wprowadzenie

Kompozyt to strukturalny wzorzec projektowy, którego zadaniem jest:

- łączenie obiektów w ten sposób, aby klient widział wiele z nich jako pojedynczy obiekt
- umożliwienie klientom jednolitego traktowania pojedynczych obiektów, a także ich złożeń
- składanie obiektów w struktury drzewiaste, przedstawiające zależność typu całość-część

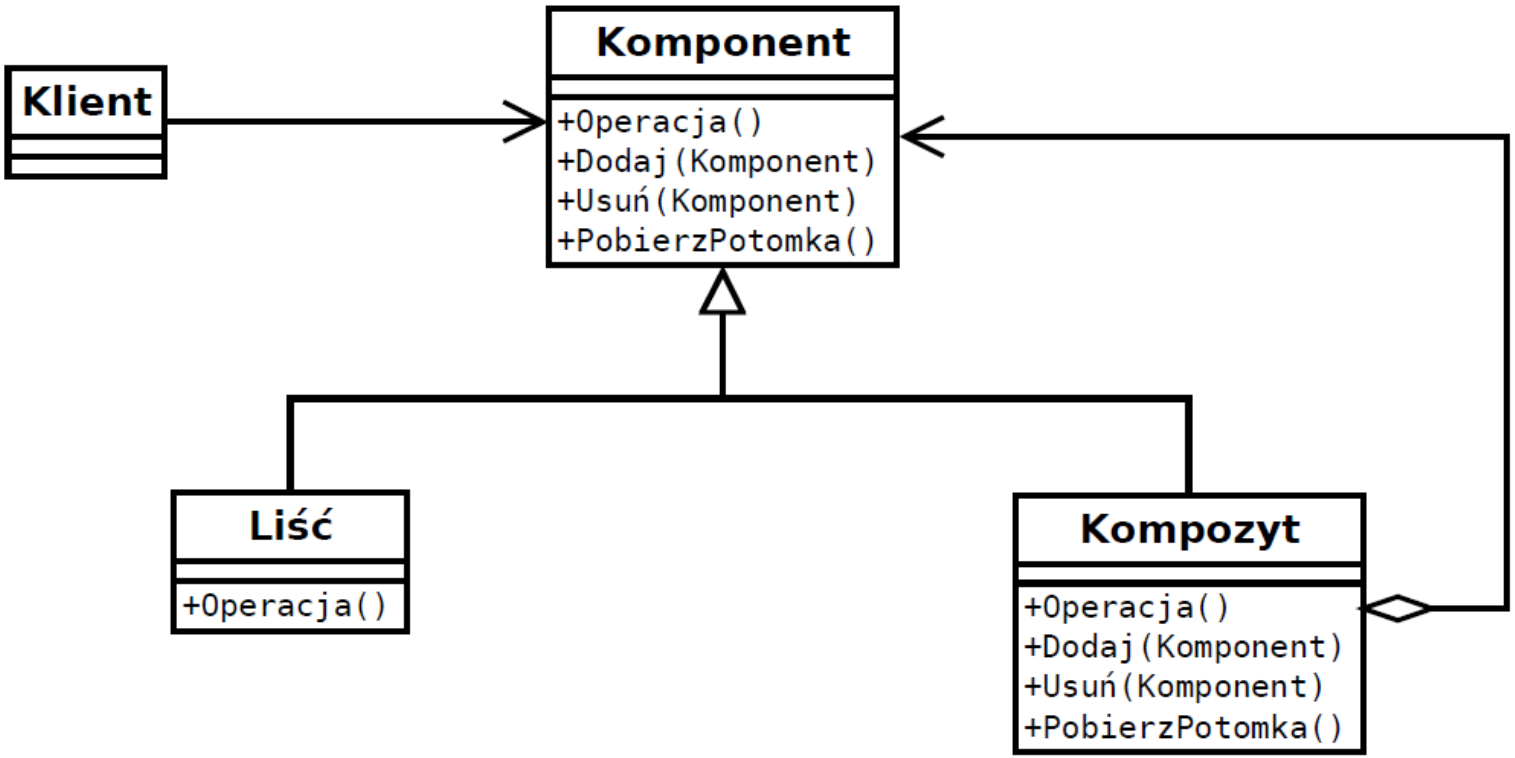
Podstawą wzorca jest klasa abstrakcyjna, która reprezentuje zarówno pojemniki jak i elementy pierwotne.

Zastosowanie

Zadaniem wzorca jest przedstawienie hierarchi obiektów typu całość-część, która ma wspólną bazową klasę abstrakcyjną. Klienci powinni ignorować różnicę pomiędzy złożeniamni obiektów, a pojedynczymi obiektami - powinni traktować jednakowo wszystkie obiekty w strukturze.

Elementy składowe

- Komponent
  - deklaruje interfejs składanych obiektów
  - na potrzeby wspólnego interfejsu dla wszystkich klas, implementuje wspólne zachowanie
  - deklaruje interfejs umożliwiający dostęp do komponentów podrzędnych
- Liść
  - reprezentuje obiekt bez elementów podrzędnych
  - określa jak zachowuje się obiekt prosty
- Kompozyt
  - przechowuje komponenty podrzędne
  - określa jak zachowują się komponenty mające podrzędne elementy
  - implementuje operacje z interfejsu klasy komponent
- Klient
  - za pomocą interfejsku komponentu, manipuluje obiektami



Rysunek: Diagram klas kompozytu.

Właściwości wzorca

- Uproszczenie interfejsu polegające na tym, że nadrzędny element reprezentuje wszystkie elementy podrzędne.
- Ułatwia dodawanie nowych komponentów. Nowe podklasy kompozytu i liścia współdziałają z klientem, bez konieczności jego modyfikacji.
- Wadą może być brak ograniczeń w dodawaniu komponentów, jeśli programista chce by kompozyt składał się tylko z komponentów określonego typu.

Przykładowy program

kompozyt.py

```
class Zamowienie:
    """To jest komponent."""
    def __init__(self, cena):
        self.cena = cena
    def oblicz_cene(self):
        return self.cena

class Towar(Zamowienie):
    """To jest jednostkowy obiekt."""
    def __init__(self, cena):
        super().__init__(cena)

    def oblicz_cene(self):
        return self.cena

class Paczka(Zamowienie):
    """To jest kompozyt."""
    def __init__(self):
        self.lista = []

    def dodaj_towar(self, towar):
        self.lista.append(towar)

    def usun_towar(self, towar):
        self.lista.remove(towar)

    def oblicz_cene(self):
        suma = 0
        for x in self.lista:
            suma += x.oblicz_cene()
        return suma
```

Przykładowy program

kompozyt.py

```
37
38
39  paczka1 = Paczka()
40  towar1 = Towar(1)
41  paczka1.dodaj_towar(towar1)
42  print(f'Wartość paczki1: {paczka1.oblicz_cene()}')
43
44  paczka2 = Paczka()
45  towar2 = Towar(2)
46  paczka2.dodaj_towar(towar2)
47  towar3 = Towar(3)
48  paczka2.dodaj_towar(towar3)
49  towar4 = Towar(4)
50  paczka2.dodaj_towar(towar4)
51  paczka2.dodaj_towar(paczka1)
52  print(f'Wartość paczki2: {paczka2.oblicz_cene()}')
53  paczka2.usun_towar(towar4)
54  print(f'Teraz wartość paczki2 wynosi: {paczka2.oblicz_cene()}')
```

Paczka › oblicz_cene() › for x in self.lista

Przykładowy program - wynik

```
Wartość paczki1: 1
Wartość paczki2: 10
Teraz wartość paczki2 wynosi: 6

Process finished with exit code 0
```

Źródła

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
Wzorce projektowe. Elementy oprogramowania obiektowego wielokrotnego użytku,
Helion, 2010

[2] http://devman.pl/pl/techniki/wzorce-projektowe-9-kompozytcomposite/