Strategia

1. Czym różni się wzorzec Strategia od zwykłej implementacji interfejsu? Jakie są wady i zalety tego wzorca?

We wzorcu strategia implementujemy wspólny interfejs, ale algorytmy oraz implementacje funkcji mogą być różne w zależności od potrzeb.

Zalety:

- pozwala na implementowanie rozszerzalnych algorytmów za pomocą interfejsu
- bazuje na kompozycji, nie na dziedziczeniu
- eliminuje instrukcje warunkowe (if else hell)
- daje możliwość wyboru implementacji funkcji, algorytmów

daje możliwość testowania pojedynczych klas strategii

Wady:

- dodatkowy koszt komunikacji pomiędzy klientem, a strategią
- zwiększenie liczby obiektów

2. Narysuj diagram klas wzorca Strategia dla Symulatora Kaczki.

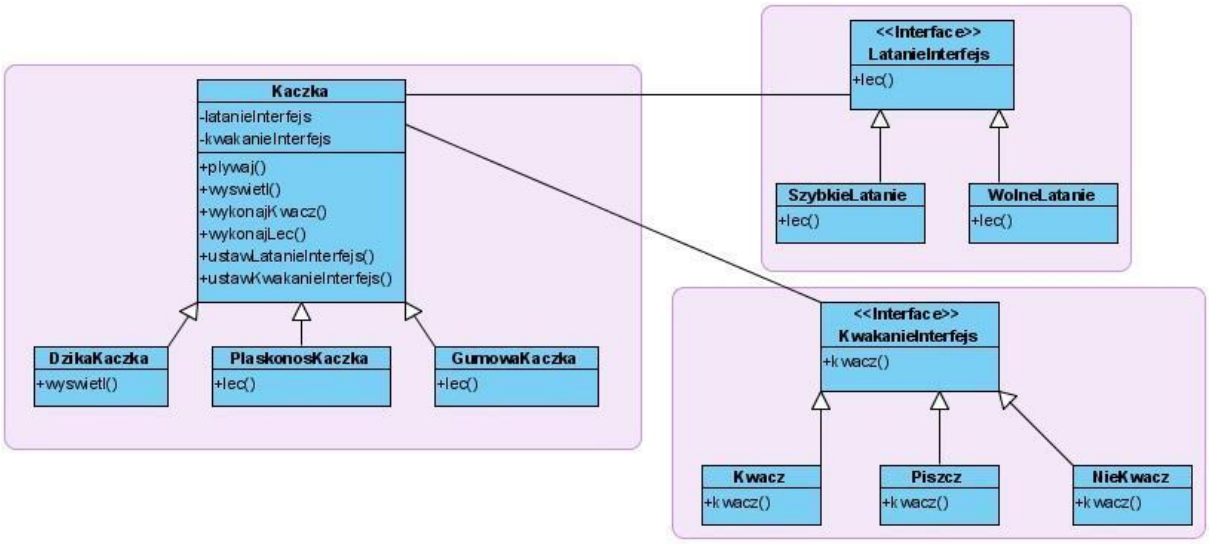


Diagram 1 Wzorce projektowe. dr Jarosław Skaruz

Jak widać na powyższym diagramie zaimplementowanie wzorca Strategia polega na stworzeniu wspólnej klasy (Kaczka) dla każdego przypadku, może ona implementować różne interfejsy (IKwakanie, ILatanie). Klasy poszczególnych kaczek korzystają z kompozycji.

Obserwator

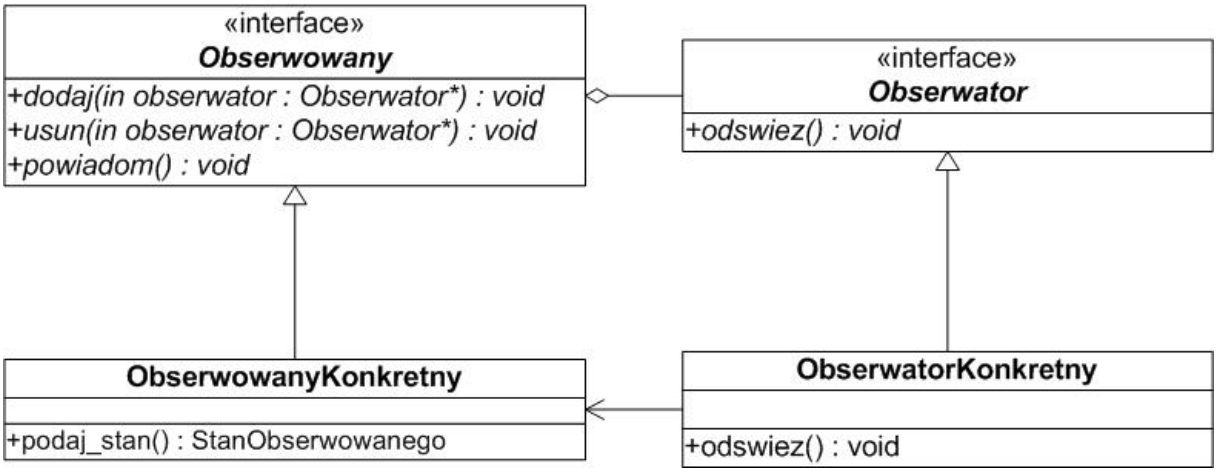
1. Jakie jest najczęstsze wykorzystanie wzorca Obserwator?

Wzorzec obserwator wykorzystujemy wszędzie tam gdzie musimy poinformować obiekty o zmianie stanu innego obiektu.

Przykłady:

- subskrybcje YouTube – użytkownik subskrybuje kanał na YT, gdy właściciel kanału doda nowy film wszyscy subskrybenci są powiadamiani
- system alertowania – np. obserwowanie cen akcji na giełdzie i informowanie użytkownika o spadkach i wzrostach wartości

2. Narysować diagram klas wzorca Obserwator.

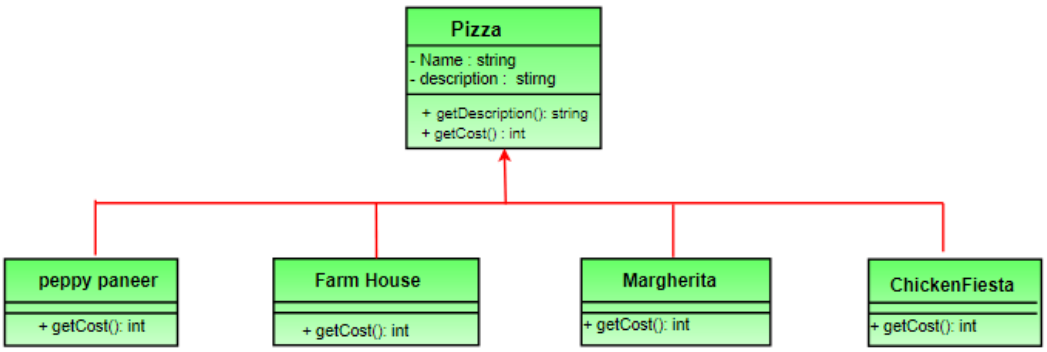


Jak widać na powyższym diagramie zaimplementowanie wzorca Obserwator polega na stworzeniu dwóch interfejsów (IObserwowany – Subject, IObserwator – Observer). Klasa implementująca IObserwowany odpowiedzialna jest za dodawanie, usuwanie oraz powiadamianie obiektów z listy o zmianach.

Dekorator

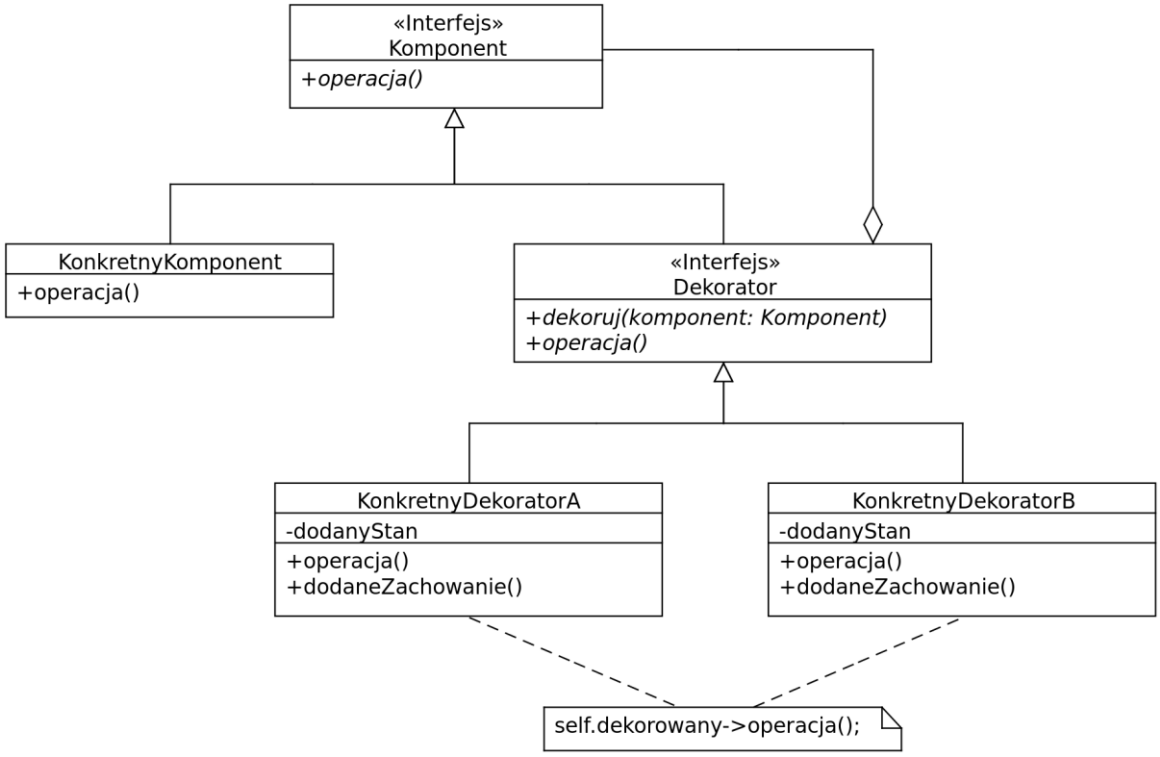
1. Jakie jest najczęstsze wykorzystanie wzorca Dekorator?

Wzorzec dekorator jest wykorzystywany wszędzie tam gdzie posiadamy jeden obiekt w kilku wariantach. Mówimy wtedy, że obiekt zostaje „udekorowany” różnymi właściwościami. Przykład z wykładu: Pizza – restauracja serwuje różne rodzaje pizz (z serem, z szynką, z [wybierz składnik]…).



Mamy więc podstawową wersję pizzy oraz „dekorujemy” ją różnymi składnikami zwiększając przy tym cenę.

2. Narysować diagram klas wzorca Dekorator.



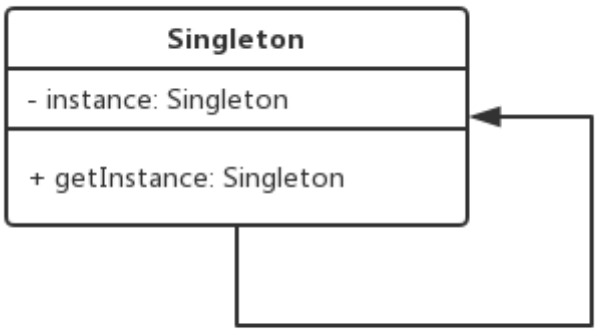
Jak widać na powyższym diagramie zaimplementowanie wzorca Dekorator polega na stworzeniu interfejsu, który zawiera konkretne operacje wspólne dla wszystkich klas. Koncepcja polega na opakowaniu oryginalnej klasy w nową klasę dekorującą.

Singleton

1. Jakie jest najczęstsze wykorzystanie wzorca Singleton?

Wzorzec singleton jest wykorzystywany tam gdzie potrzebujemy mieć globalny dostęp do stanu. Ograniczamy instancje danej klasy do jednego obiektu. Przykład dziennika: mamy klasę dziennik, która zapisuje wpisy, poszczególne komponenty mogą przekazać treść wpisu, ale to dziennik decyduje gdzie go zapisać. Każdy z komponentów może uzyskać dostęp w dowolnym momencie trwania programu.

2. Narysować diagram klas wzorca Singleton.



Singleton posiada metodę statyczną sprawdzającą czy jego instancja już istnieje, jeżeli nie - tworzy ją. Obiekt ten jest zwracany przez referencję.

4. W jaki sposób zaimplementować wzorzec Singleton tak, aby można go było wykorzystać programach współbieżnych? Podać przykład takiej implementacji.

Wykorzystanie wzorca Singleton przez program wielowątkowy stwarza problem jednoczesnego utworzenia instancji klasy co powoduje dwie lub więcej obiektów Singleton. Aby temu zapobiec należy zaimplementować dodatkowy mechanizm zapobiegający takiemu działaniu.

```java
public static Singleton getInstance(ArrayList<String[]> value) {
    if (instance == null) {
        synchronized (Singleton .class) {
            if(instance == null) {
                instance = new Singleton(value);
            }
        }
    }
    return instance;
}
```

Frag. kodu 1 Moja implementacja z double check'iem