Paradygmaty Programowania

Marcin Bączyk

Wykład 5

21 marca 2021

Treść dzisiejszego wykładu

- wstęp
- przykład
- mechanizmy obiektowości
- relacje między obiektami

Literatura

- Robert C. Martin: “Czysta architektura. Struktura i design oprogramowania. Przewodnik dla profesjonalistów”
- Bartosz Walter: “Zaawansowane projektowanie obiektowe”
- Grady Booch, James Rumbaugh, Ivar Jacobson: “UML przewodnik użytkownika”

Programowanie obiektowe

Programowanie obiektowe jest sposobem wyrażenia programu komputerowego w postaci zbioru obiektów komunikujących się między sobą w celu wykonania zadań.

wykorzystywane mechanizmy / techniki :

- abstrakcja
- hermetyzacja
- dziedziczenie
- polimorfizm

Programowanie obiektowe

Programowanie obiektowe jest sposobem wyrażenia programu komputerowego w postaci zbioru obiektów komunikujących się między sobą w celu wykonania zadań.

wykorzystywane mechanizmy / techniki :

- abstrakcja
- hermetyzacja
- dziedziczenie
- polimorfizm
- **odpowiedzialność**

Konsekwencją stosowania podejścia czysto obiektowego jest przypisanie każdej operacji w systemie konkretnemu obiektowy i przekazanie mu odpowiedzialności za tę operację.

Odpowiedzialność

1. Rodzina funkcji, które służą jednemu konkretnemu aktorowi.
2. Zakres wykonywanych czynność.
3. Zobowiązanie albo kontrakt danego typu lub klasy.

Aktor

Spójny zbiór ról odgrywanych przez użytkowników klasy lub systemu. Obiekt, niebędący częścią danego systemu, który wchodzi z nim w interakcję.

Klient

Obiekt na rzecz, którego świadczona jest usługa.

Programowanie obiektowe

Programowanie obiektowe jest odpowiedzią na potrzebę modelowania świata rzeczywistego w możliwie wierny sposób (z dokładnością do poziomu abstrakcji). Podstawowe mechanizmy programowania obiektowego ułatwiają tworzenie oprogramowania dla konkretnej dziedziny.

Żaden z podstawowych mechanizmów programowania obiektowego nie musi być wspierany przez dany język, aby móc napisać w nim program obiektowy.

Systemy projektowane obiektowo składają się z dużej liczby "małych" obiektów współpracujących ze sobą. Interfejs publiczny, oznaczający usługi jakie obiekt może świadczyć swoim klientów informuje jedynie o tym co może być zrobione. Klienci poszczególnych obiektów nie ingerują w to w jaki sposób dana usługa jest świadczona.

Systemy obiektowe projektowane są w taki sposób aby obiekt odpowiedzialny za daną funkcjonalność mógł być zastąpiony innym bez zmiany kodu klienta.

Obiekt

Obiekt istniejący w w programie reprezentuje konkretny obiekt w świecie rzeczywistym.

Każdy obiekt posiada:

- tożsamość (np. adres w pamięci)
- stan (wartości atrybutów)
- zachowanie (metody, czyli usługi które mogą świadczyć klientom)

Obiekt jest elementem modelowanej dziedziny (pewnego obszaru rzeczywistości), odpowiadającym za wybrany jej fragment.

Od obiektu zależy w jaki sposób realizowana jest odpowiedzialność.

Przykład

Przykład pokazujący różnice w dekompozycji obiektowej i funkcjonalnej

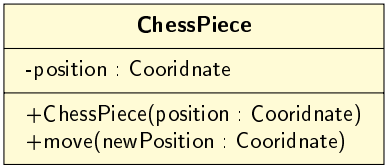
Mechanizmy obiektowości

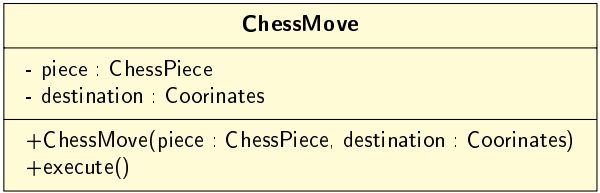
Klasa - typ

W językach obiektowych, w których występuje pojęcie klasy oznacza ono sposób definiowania obiektów.

Klasa opisuje atrybuty jakie posiadają obiekty danego typu oraz sposób ich zachowania. Klasa może (i powinna) definiować dopuszczalny stan obiektów danego typu.

Obiekty utworzone na podstawie danej klasy nazywane są jej instancjami. Do tworzenia obiektów służy specjalna metoda nawyzna konstruktorem.





Mechanizmy obiektowości

Abstrakcja (z łac. *abstractio* - oderwanie

Abstrakcja, w kontekście projektowania obiektowego, jest zdolnością do pomijania niektórych, nieistotnych aspektów modelowanego fragmentu rzeczywistości.

Abstrahowaniu podlegają zarówno dane jakie przechowują obiekty jak również ich zachowanie.

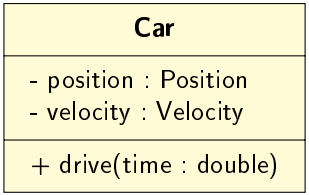
Abstrakcję można rozumieć na dwa sposoby:

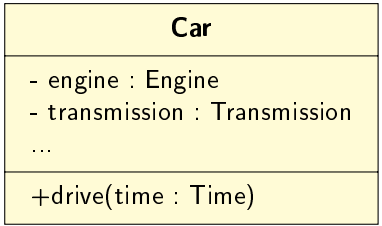
1. jako zbiór istotnych w kontekście rozwiązywanego problemu parametrów i cech zachowania danego obiektu, oraz
2. jako zestaw cech wspólnych większej ilości obiektów.

Abstrakcja

W zależności od budowanego systemu, poszczególne fragmenty będą modelowane na różnym poziomie szczegółowości.

Inaczej będzie modelowany samochód na potrzeby prostej gry zręcznościowej, a inaczej złożony symulator do nauki jazdy samochodem.





Abstrakcja

- Abstrakcja pozwala na odsunięcie w czasie konieczności podejmowania decyzji projektowych.
- Dzięki abstrakcji projektant może się skupić na istotnych cechach modelowanego problemu odsuwając (być może na zawsze) konieczność zajmowania się nieistotnymi jego aspektami.
- Pozwala na stosowanie warstw pośredniczących.
- Pozwala korzystnie wpływać na koszt utrzymania oprogramowania.
- System zbudowany z abstrakcyjnych komponentów może być łatwo rozszerzany.

Mechanizmy obiektowości

Polimorfizm (z gr. *poly* - wiele, *morph* - postać

Polimorfizm jest mechanizmem umożliwiającym korzystanie z różnych obiektów niezależnie od ich konkretnego typu przez tego samego klienta.

Dzięki polimorfizmowi wybór konkretnej metody wywoływanej przez klienta nie zależy od niego samego (czyt. brak instrukcji if / then / else / etc.), a od obiektu świadczącego usługi.

Polimorfizm charakteryzuje się istnieniem metod o takiej samej sygnaturze przez obiekty różnych typów.

Rodzaje polimorfizmu

- dynamiczny - czasu wykonania
- statyczny - czasu kompilacji (jeżeli kod jest kompilowany)

Mechanizmy obiektowości

Polimorfizm dynamiczny

Decyzja o tym dokąd jest przekazywane sterowanie w programie zależy od obiektu i wyznaczana jest w czasie działania programu.

W językach kompilowanych (np. Java, C++) do polimorficznego wywoływania metod wykorzystywane jest mechanizm dziedziczenia lub implementacji interfejsu.

W językach interpretowanych (np. Matlab) do polimorficznego wywołania metody mechanizm dziedziczenia oraz implementacji interfejsu nie jest niezbędny

Polimorfizm statyczny

Decyzja o przekazaniu sterowania zależy od typu obiektu i wyznaczana jest w trakcie kompilacji programu.
W językach z rozbudowanym programowaniem generycznym (C++) ten rodzaj polimorfizmu wykorzystywany jest do tworzenia rodziny klas o podobnej strukturze.

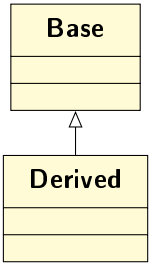
Dziedziczenie

Dziedziczenie jest mechanizmem współdzielenia funkcjonalności między klasami.

Dzięki dziedziczeniu klasa oprócz swoich własnych atrybutów oraz zachowań uzyskuje także dostęp do atrybutów i zachowań klasy, po której dziedziczy.

Klasa, z której następuje dziedziczenie nazywana jest klasą bazową.

Klasa dziedzicząca po klasie bazowej nazywana jest klasą pochodną.

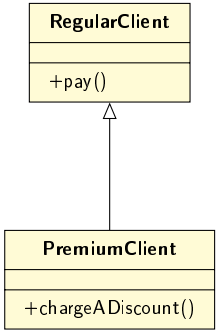


Dziedziczenie a implementacja interfejsu

Dziedziczenie powoduje przeniesienie z klasy bazowej do klasy pochodnej typu oraz implementacji.

Implementacja interfejsu powoduje przeniesienie z klasy bazowej do klasy pochodnej jedynie typu.

Korzystanie z interfejsów skutkuje powstawaniem słabszych zależności pomiędzy klasami.

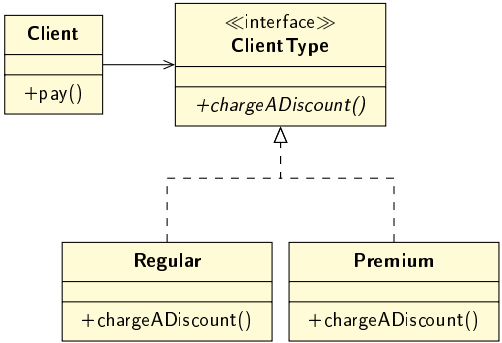


Dziedziczenie a implementacja interfejsu

Dziedziczenie powoduje przeniesienie z klasy bazowej do klasy pochodnej typu oraz implementacji.

Implementacja interfejsu powoduje przeniesienie z klasy bazowej do klasy pochodnej jedynie typu.

Korzystanie z interfejsów skutkuje powstawaniem słabszych zależności pomiędzy klasami.



Interfejs

W programowaniu obiektowym można wyróżnić dwa znaczenia słowa interfejs. Można je rozumieć jako:

1. Zestaw publicznych atrybutów i metod obiektu, które są dostępne dla jego klientów.

   Niekiedy interfejsem publicznym określa się również wszystkie (publiczne i prywatne) atrybuty i metody klasy.

2. Abstrakcyjną klasę bazową definiującą zestaw metod, które klasy pochodne muszą zaimplementować.

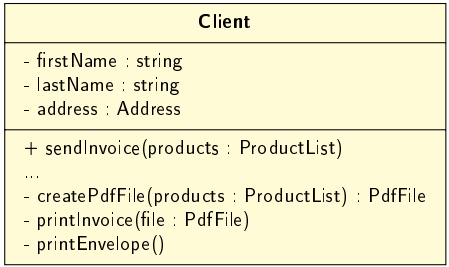
   Klasą abstrakcyjną nazywa się klasę, która nie ma swoich instancji.

Mechanizmy obiektowości

Hermetyzacja

Ukrywanie przed klientami szczegółów implementacji obiektu.

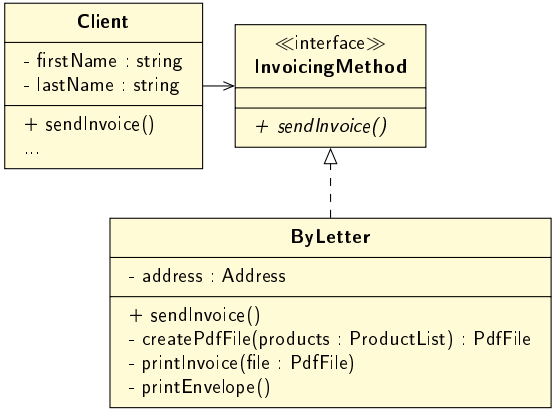
Ukrywanie szczegółów implementacji podnosi stopień abstrakcji.



Hermetyzacja

Ukrywanie szczegółów decyzji projektowych.

W szczególności należy hermetyzować decyzje, które na późniejszym etapie projektu mogą ulec zmianie, np. zmiana zachowania metody lub sposobu przechowywania danych.



Rodzaje relacji pomiędzy obiektami

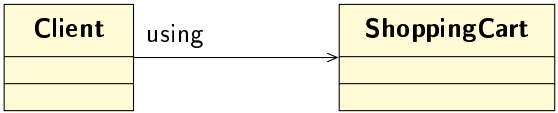
Asocjacja

Asocjacja polega na umieszczeniu w danym obiekcie referencji (odnośnika) do innego obiektu. Poprzez referencje obiekty mogą się odwoływać do metod innych obiektów.

Asocjacja reprezentuje relację pomiędzy dwoma obiektami, w której oba obiekty istnieją niezależnie od siebie.

Ustanowiona relacja nie musi być trwała. Obiekty nie są ze sobą związane na stałe i mogą się zmieniać na inne.

Asocjacja może być jedno lub dwukierunkowa.



Rodzaje relacji pomiędzy obiektami

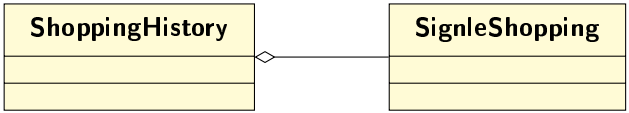
Agregacja

Agregacja jest rodzajem relacji niesymetrycznej. Obiekt agregujący przechowuje referencje do obiektów agregowanych oraz zarządza nimi poprzez:

- dodawanie i usuwanie oraz
- przeszukiwanie w odpowiedzi na żądania swoich klientów.

Agregacja reprezentuje relację pomiędzy dwoma obiektami, w której oba obiekty mogą istnieć niezależnie od siebie.

Ustanowiona relacja nie musi być wyłączna. Obiekty agregowane mogą należeć do wielu obiektów agregujących.



Rodzaje relacji pomiędzy obiektami

Kompozycja

Kompozycja jest rodzajem znacznie silniejszej, niż w przypadku agregacji, relacji niesymetrycznej. Komponowany obiekt składa się (jest właścicielem) ze swoich obiektów zależnych

Kompozycja reprezentuje relację pomiędzy dwoma obiektami, w której obiekt zależnie nie może istnieć niezależnie obiektu komponującego.

Ustanowiona relacja jest wyłączna. Komponowane obiekty mogą należeć do tylko jednego obiektu reprezentującego całość.

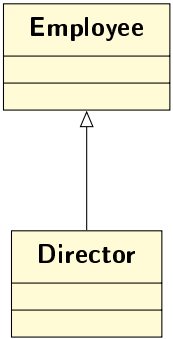
Rodzaje relacji pomiędzy obiektami

Dziedziczenie

Dziedziczenie jest najsilniejszym rodzajem relacji, w której jedna klasa jest tworzona na podstawie innej.

Silna relacja pomiędzy klasami jaką jest dziedziczenie może łamać hermetyzację.

Silna zależność pomiędzy dwoma typami powoduje duży stopnień sprzężenia, utrudniającego rozwój oprogramowania. Tam gdzie to możliwe dziedziczenie powinno być zastępowane stosowaniem interfejsów.

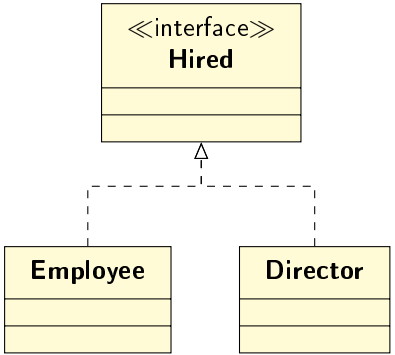


Rodzaje relacji pomiędzy obiektami

Realizacja (implementacja interfejsu)

Realizacja jest rodzajem relacji nieco mniej silnym niż dziedziczenie. Tworzona klasa jedynie implementuje zachowanie narzucone przez interfejs, nie dziedziczy zaś żadnych atrybutów i zachowania po klasie bazowej.

Zależność jaka powstaje przy realizacji pozwala na izolowanie implementacji obiektu od jego późniejszego wykorzystania. Stosowanie implementacji interfejsów pozwala na skuteczną hermetyzację modułów.



Dziedziczenie a kompozycja

Dziedziczenie

- relacja silna mogąca łamać hermetyzację
- relacja ustalana w czasie kompilacji
- klasa pochodna dziedziczy typ oraz atrybuty i zachowanie klasy bazowej

Kompozycja

- relacja relatywnie słaba
- ustalana w trakcie wykonywania programu
- wiąże obiekty jedynie przez typ agregowanych obiektów

Tam gdzie to możliwe należy preferować kompozycję nad dziedziczenie.

Podsumowanie

W programowaniu obiektowym:

- Podstawą systemu są współpracujące ze sobą obiekty.
- Obiekty mogą być powiązane ze sobą za pomocą kilku rodzajów relacji.
- Obiekty przyjmują na siebie odpowiedzialność za wybrany obszar implementowanego modelu.
- Mechanizmy obiektowe pozwalają w prawidłowy sposób projektować systemu i przydzielać odpowiedzialność obiektom.
- Klienci danego obiektu nie wiedzą w jaki sposób realizowana jest jego odpowiedzialność.
- Obiekty mogą być przekazywane pomiędzy modułami bez wprowadzania dodatkowych (zbędnych) zależności.
- Rozwój (dodawanie nowych) funkcjonalności odbywa się bez zmiany wysokopoziomowych reguł biznesowych (ograniczeń wynikających z logiki i specyfiki danej dziedziny).