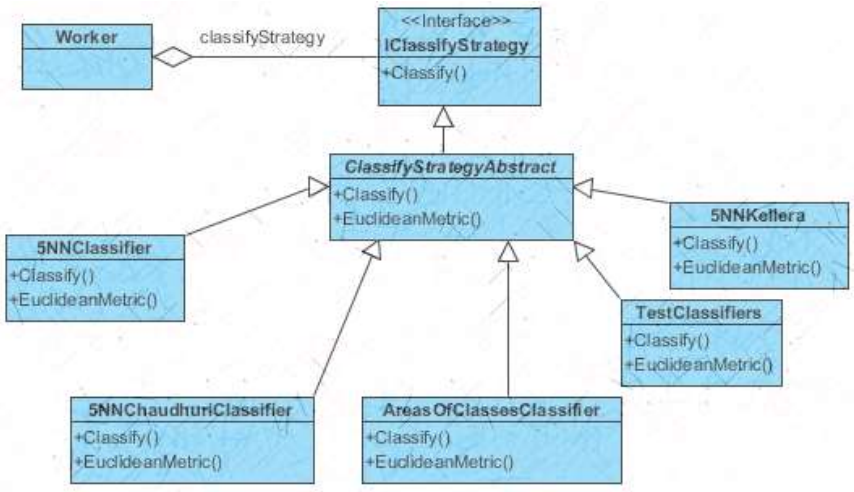
PODSTAWY INŻYNIERII OPROGRAMOWANIA

Diagramy Klas - Wprowadzenie



Przygotował: dr inż. Radosław Adamus[1]
Aktualizacja: dr inż. Tomasz Kowalski
Aktualizacja: dr inż. Joanna Jojczyk

Instytut Informatyki Stosowanej, 2020

[1] Na podstawie: Subieta K., Język UML, V Konferencja PLOUG, Zakopane, 1999.

Wstęp

Diagram klas (zwany poprzednio także diagramem asocjacji klas lub modelem obiektowym) jest pojęciem centralnym we wszystkich znanych metodykach obiektowych. Koncentruje się on wokół pojęcia klasy wraz z przypisanymi jej atrybutami i metodami. Oprócz tego zasadniczego elementu pojawiają się w diagramach klas różnorodne oznaczenia o charakterze pomocniczym. Diagram klas pokazuje klasy w postaci pewnych oznaczeń graficzno-językowych powiązanych w sieć zależnościami należącymi do trzech kategorii:

- **Dziedziczenie** (ang. inheritance), czyli ustalenie związku generalizacji/specjalizacji pomiędzy klasami.
- **Asocjacja** (ang. association), czyli dowolny związek pomiędzy obiektami dziedziny przedmiotowej, który ma znaczenie dla modelowania.
- **Agregacja** (ang. aggregation), czyli szczególny przypadek asocjacji, odwzorowujący stosunek całość-część pomiędzy obiektami z modelowanej dziedziny przedmiotowej.

Diagram klas w identycznej wersji jest stosowany zarówno do zapisu wyników analizy jak i do specyfikowania założeń projektowych. Diagramy klas są podstawą dowolnej analizy i dowolnego projektu, oraz sednem metody określanej jako „obiektowa”.

Klasy

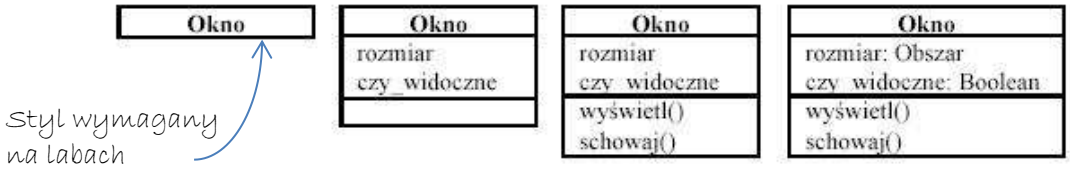
Celem klas jest uchwycenie zarówno statycznych własności obiektów (ich struktury), jak i własności dynamicznych, w tym operacji, które można wykonywać na obiektach. Poniżej przedstawiona jest ogólna definicja klasy:

Klasa jest miejscem przechowywania tych informacji dotyczących obiektów, które są dla nich niezmienne, wspólne lub dotyczą całej ich populacji. Takie informacje są nazywane *inwariantami* klasy.

Inwarianty dotyczące jednego obiektu mogą być przechowywane w wielu klasach tworzących hierarchię lub inną strukturę dziedziczenia.

Obiekt przypisany do klasy zawierającej jego inwarianty jest nazywany wystąpieniem (instancją) tej klasy

W języku UML oznaczeniem klasy jest prostokąt podzielony na trzy części. Pierwsza część zawiera nazwę klasy, druga specyfikację atrybutów, jakie będą posiadały obiekty tej klasy, a trzecia specyfikację operacji, czyli usług, jakie udostępniają obiekty tej klasy. Rysunek 1 pokazuje różne możliwe sposoby oznaczania klasy w zależności od wymaganego stopnia szczegółowości.



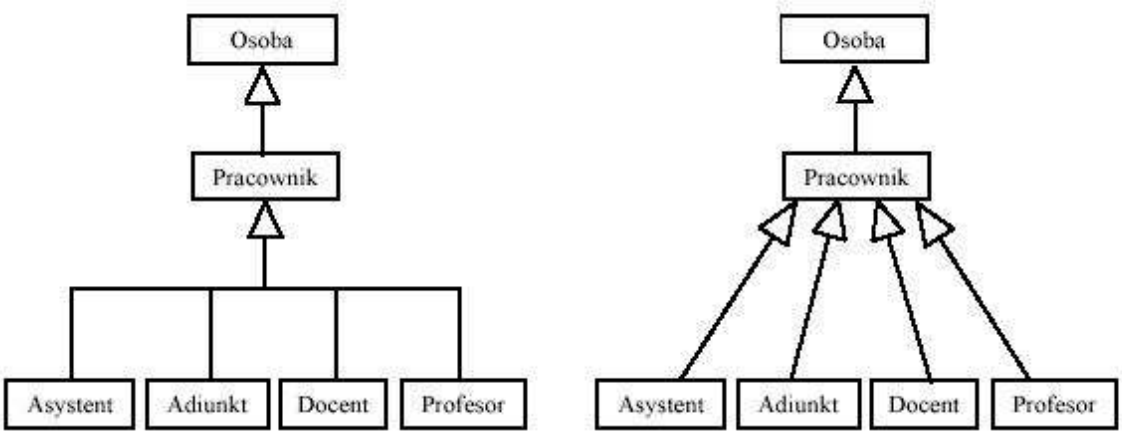
Rys 1. Najprostsze warianty specyfikacji klasy w UML

W wielu wypadkach **(*np. laboratoriów PIO*)** wystarczającą informacją jest nazwa klasy, stąd możliwość sprowadzenia specyfikacji klasy w UML jedynie do prostokąta z wpisaną nazwą klasy. Możemy elementy należące do klasy wyspecyfikować abstrahując od konkretnego języka programowania lub w sposób bardzo zbliżony do specyfikacji w konkretnym języku, np. dla języka Java **+**, #, **-** oznaczają odpowiednio public, protected, private, podkreślenie oznacza atrybut lub metodę klasy (static). Oznaczenia specyficzne dla Java czy C++ nie są jednak składową UML; nie ma przeszkód, aby UML zastosować do dowolnego języka programowania.

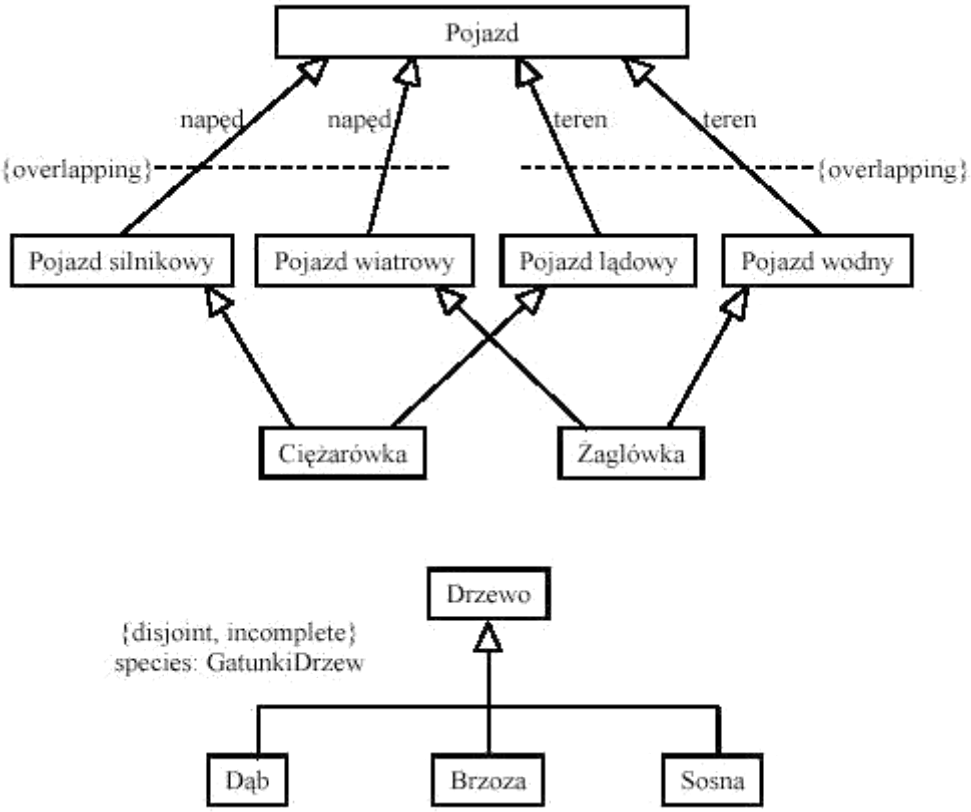
Zależności pomiędzy klasami:

Dziedziczenie, czyli związek generalizacji - specjalizacji

Strzałka (z białym trójkątnym grotem) prowadzi od pod-klasy do jej bezpośredniej nadklasy, (Rys.2). Zakłada się, że obiekt pod-klasy automatycznie dziedziczy wszystkie atrybuty, metody, asocjacje i agregacje z wszystkich jej nadklas. Projektant może explicite zadeklarować aspekt, według którego specjalizuje się dany obiekt (napęd lub teren na Rys.3), oraz określić fakt niepustego przecięcia zakresów znaczeniowych - zbiorów obiektów (overlapping). Klasy Ciężarówka i Żaglówka zostały zdefiniowane z użyciem wielokrotnego dziedziczenia. Na Rys.3 zilustrowano również możliwość określania faktu, że podklasy są rozłączne (disjoint) i nie przykrywają całego zakresu znaczeniowego ich nadklasy (incomplete). Są to przykłady oznaczeń UML o charakterze pomocniczym i rzadko występują na diagramach. Niemniej jednak, doświadczeni analitycy i programiści konstruując model obiektowy (w postaci diagramu lub kodu) są świadomi powyższych aspektów relacji dziedziczenia, tak aby zbudowany model był spójny, jednoznaczny a zarazem elastyczny.



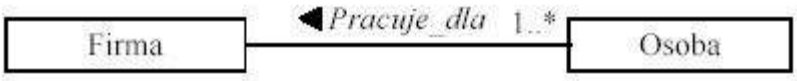
Rys. 2. Dziedziczenie w UML i jego równoważne sposoby zapisu (ze względu na czytelność, na zajęciach preferowane jest użycie sposobu po prawej)



Rys. 3 Dodatkowe elementy specyfikacji dziedziczenia

Asocjacje

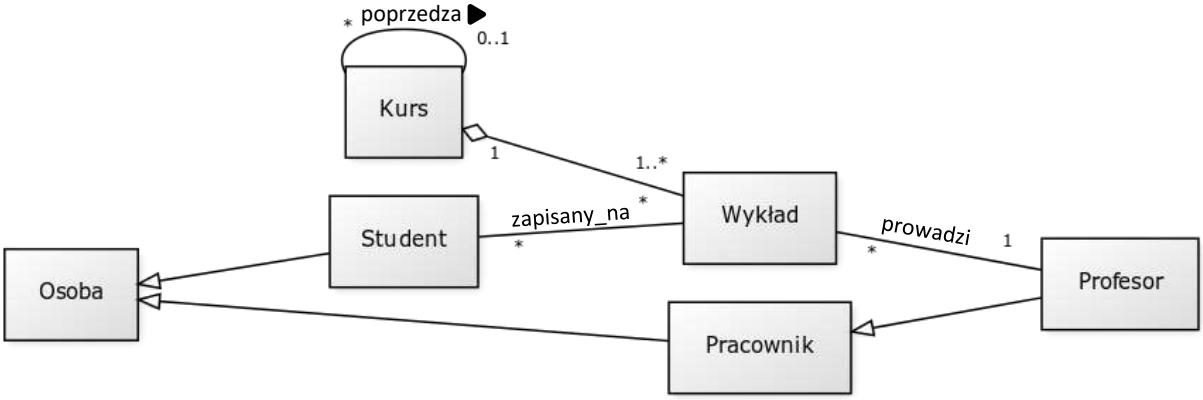
Oznaczenia klas w UML mogą być połączone liniami oznaczającymi asocjacje, czyli powiązania pomiędzy obiektami tych klas. Rys. 4 pokazuje specyfikację asocjacji *Pracuje_dla* pomiędzy obiektami klasy Osoba i obiektami klasy Firma. Czarny trójkącik określa kierunek wyznaczony przez nazwę powiązania (w danym przypadku określa on, że osoba pracuje dla firmy, a nie firma pracuje dla osoby). Asocjacje mają nazwy, takie jak Pracuje_dla, które wyznaczają znaczenie tej asocjacji w modelu pojęciowym. Jeżeli to znaczenie jest oczywiste, wówczas nazwę asocjacji można pominąć.



Rys. 4 Asocjacja i jej oznaczenie

Asocjacje mogą być wyposażone w oznaczenia liczności. Liczność oznacza, ile obiektów innej klasy może być powiązane z jednym obiektem danej klasy; zwykle określa się to poprzez parę liczb (znaków) oznaczającą minimalną i maksymalną liczbę takich obiektów. Zapis liczności w UML jest naturalny i nie wprowadza

specjalnych symboli graficznych. Oznaczenie **0..*** oznacza liczność od zera do dowolnie wielkiej liczby. Analogicznie, **1..*** oznacza liczność od jeden do dowolnie wielkiej liczby, zaś **0..1** oznacza liczność zero lub jeden. Generalnie, oznaczenia liczności mogą być zapisem dowolnego podzbioru liczb całkowitych nieujemnych, gdzie podwójna kropka oznacza „od...do...", zaś gwiazdka oznacza dowolną liczbę (z reguły większą od 1). Rysunek 5 pokazuje przykładowy prosty diagram klas pokazujący zależności pomiędzy wybranymi klasami w systemie obsługi uczelni.



Rys. 5 Diagram klas

W UML istnieje możliwość opisu asocjacji nie tylko poprzez jej nazwę ale również poprzez role jakie klasy grają w tym powiązaniu. Jak każdy inny sposób opisu nie jest on wymagany w przypadku gdy role są oczywiste, natomiast w sytuacji gdy asocjacja łączy ze sobą tę samą klasę opis ról jest obligatoryjny (Patrz rys 5 – asocjacja łącząca klasę Kurs – czytamy ją w następujący sposób: Co najwyżej jeden kurs poprzedza kurs; dowolna liczba kursów może występować po kursie).

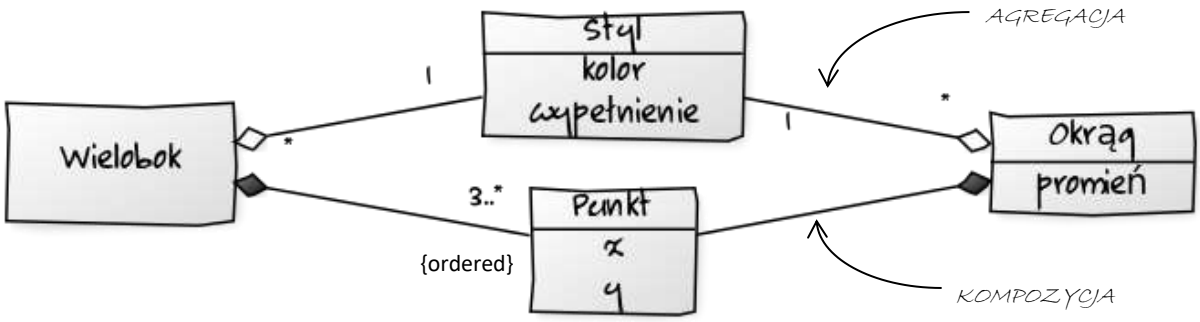
Agregacja i jej silniejszy wariant – kompozycja

Agregacja jest szczególnym przypadkiem asocjacji wyrażającym zależność część-całość, „zawiera", „składa się z". Np. silnik jest częścią samochodu, czyli obiekt-samochód jest agregatem obiektów będących jego częściami. Niestety, nie istnieje powszechnie akceptowana definicja agregacji, zaś wątpliwości co do jej znaczenia są zasadnicze.

Autorzy UML podjęli próbę uporządkowania agregacji w taki sposób, aby zmniejszyć nieco zamieszanie dookoła tego pojęcia. Pozostawiając klasyczne pojęcia agregacji znane z innych metodyk i notacji obiektowych (np. OMT),

wprowadzili oni mocniejszą formę agregacji, nazywając ją kompozycją. Związek kompozycji oznacza, że dana część może należeć tylko do jednej całości. Co więcej, usunięcie całości powoduje automatyczne usunięcie wszystkich jej części związanych z nią związkiem kompozycji.

Przykładem związku kompozycji jest uczelnia i wydział: wydział (technicznie mówiąc jego instancja) nie występuje w kilku uczelniach i znika w momencie likwidacji uczelni.



Rys. 6 Agregacja i kompozycja „by example”.

Rys.6 ilustruje zastosowanie agregacji i kompozycji. Każde wystąpienie obiektu Punkt należy albo do obiektu Wielobok albo do obiektu Okrąg; nie może należeć do dwóch obiektów naraz. Wystąpienie obiektu Styl może być dzielone przez wiele obiektów Wielobok i Okrąg. Usunięcie obiektu Wielobok powoduje kaskadowe usunięcie wszystkich związanych z nim obiektów Punkt, natomiast nie powoduje usunięcia związanego z nim obiektu Styl. Zwrócimy uwagę, że ograniczenie mówiące, iż obiekt Punkt może należeć do dokładnie jednego obiektu Wielobok lub Okrąg nie może być wyrażone w inny sposób.

Informacje dodatkowe

Polecane edytory:

- yuml.me

„Kody” do generacji bazy diagramów z rysunku 5 i 6 w narzędziu on-line yuml.me

```
[Osoba]^-[Student], [Osoba]^-[Pracownik], [Pracownik]^-[Profesor],
[Wykład]*-*[Student], [Kurs]<>1-1..*[Wykład], [Wykład]*-1[Profesor],
[Kurs]*-0..1[Kurs]

[Wielobok]++-3..*[Punkt|x;y]
[Wielobok]*<>-1[Styl|kolor;wypełnienie]
[Punkt]-++[Okrąg|promień]
[Styl]1-<>*[Okrąg]
```

- Draw.io link do edytora: https://app.diagrams.net