Diagram klas

Diagram klas (class diagram) przedstawia klasy występujące w systemie i statyczne relacje pomiędzy nimi wraz z ograniczeniami. Jest podstawowym diagramem struktury logicznej systemu.

Diagram klas jest najczęściej używanym diagramem UML. Z reguły zawiera także największą ilość informacji i stosuje największą liczbę symboli. Na diagramie są prezentowane klasy, ich atrybuty i operacje oraz powiązania między klasami. Diagram klas przedstawia więc podział odpowiedzialności pomiędzy klasy systemu i rodzaj wymienianych pomiędzy nimi komunikatów. Z uwagi na rodzaj i ilość zawartych na tym diagramie danych jest on najczęściej stosowany do generowania kodu na podstawie modelu. **Do czego służy diagram klas:**

- pozwala na sformalizowanie specyfikacji danych i zestawu metod,
- pełni rolę uniwersalnego środka graficznego wizualizującego implementację klas,
- narzędzia projektowe zgodne ze specyfikacją UML umożliwiają, na podstawie diagramów klas, generowanie elementów źródeł w popularnych językach obiektowych, takich jak: C++, Java, C#.

Proces tworzenia diagramu klas składa się z ośmiu etapów:

1. zidentyfikowanie i nazwanie klasy
2. opcjonalne określenie zobowiązań klas
3. połączenie poszczególnych klas z wykorzystaniem związków asocjacji
4. zidentyfikowanie oraz nazwanie atrybutów i operacji
5. wyselekcjonowanie asocjacji z użyciem wszystkich jej cech (nazwa, rola, nawigacja, liczebność, agregacja, kwalifikacja)
6. opracowanie innych rodzajów związków (uogólnień, zależności, realizacji) 7. precyzyjne wyspecyfikowanie atrybutów i operacji
8. opcjonalne opracowanie diagramów obiektów.

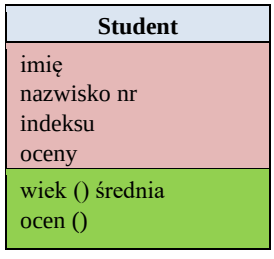
Symbol klasy

- symbolem klasy jest prostokąt, zwykle podzielony poziomymi liniami na trzy sekcje:
  - nazwy
  - atrybutów
  - operacji

nazwa

atrybuty

operacje



- w razie potrzeby może zostać uzupełniony dodatkowymi sekcjami (np. wyjątków).

Klasa: W celu zwiększenia czytelności, dowolny z nich można ukryć bądź dodać nowy (np. przechowujący zdarzenia lub wyjątki), choć zwykle są to właśnie trzy przedziały. Tradycyjnie

nazwa klasy zaczyna się z dużej litery, jest wytłuszczona, a w przypadku klasy abstrakcyjnej – także pochyła. Nazwa klasy to z reguły rzeczownik w liczbie pojedynczej.

Obiekt: jest instancją klasy, podobnie jak w przypadku programowania obiektowego. Nazwa obiektu jest umieszczana przed nazwą klasy i oddzielana od niej dwukropkiem.

Cechy klasy reprezentują informację jaką klasa przechowuje. Mogą zostać zapisane w postaci dwóch, w zasadzie równoważnych notacji: jako atrybuty klasy (umieszczane w przedziale atrybutów) lub jako relacje pomiędzy klasami (zapisywane w postaci linii łączącej klasy). Zwykle pierwsza notacja jest stosowana do typów prostych lub obiektów reprezentujących wartości, natomiast druga do typów złożonych.
Atrybuty reprezentują wartości proste lub niewielkie obiekty, asocjacje – obiekty złożone.

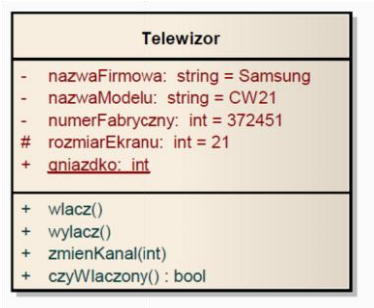
Widoczność nazwa : typ [krotność] {ograniczenia} = wartość domniemana

Widoczność: skąd atrybut jest widoczny?
Krotność: ile obiektów potrzeba i ile można umieścić w atrybucie?
Ograniczenia: jakie dodatkowe warunki spełniają wartości atrybutu?
Wartość domniemana: jaka jest wartość atrybutu gdy nie podano jej wprost?.

Operacje reprezentują usługi, jakie klasa oferuje. Ich realizacje – metody – dostarczają implementacji tych usług.

UML posiada **4 poziomy widoczności elementu:**

Poziom	Symbol	Opis
publiczny	+	obiekty wszystkich klas mają dostęp do atrybutu lub operacji
prywatny	–	tylko obiekty danej klasy mają dostęp do atrybutu lub operacji
chroniony	#	wyłącznie obiekty klas dziedziczących z danej klasy mają dostęp do atrybutu lub operacji
pakietowy	~	tylko składowe pakietu, do którego dana klasa należy, mają dostęp do atrybutu lub operacji



Elementy statyczne

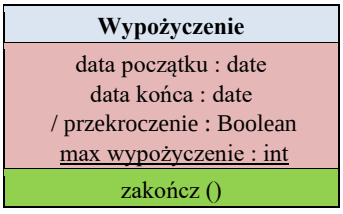
- elementy można zadeklarować jako statyczne - działające na rzecz klasy, nie obiektu
- statyczna metoda może odwoływać się tylko do innych elementów statycznych
- koncepcja identyczna jak w językach zorientowanych obiektowo
- reprezentacją graficzną jest **podkreślenie**

Krotność pozwala określić minimalną i maksymalną liczbę obiektów, jakie można powiązać z daną cechą:
dolna granica..górna granica – przedział od – do,
1 – dokładnie jeden obiekt,
0..1 – opcjonalnie jeden obiekt,
1..* – przynajmniej jeden obiekt, 1,
3, 5 – konkretne liczby obiektów, *
– dowolna liczba obiektów.
Krotność można określać jako ograniczenie dolne i górne, jednak oczywiste lub powtarzające się wartości graniczne można pomijać, np.. zapis 0..* jest skracany do *, a zapis 1..1 do 1.

Joanna Fabiś-Domagała

Niemal każdy element w UML może posiadać **dodatkowe właściwości i ograniczenia**, które szczegółowo opisują jego zachowanie i przeznaczenie. Są one zapisywane w nawiasach klamrowych:
{ordered} – obiekty wewnątrz cechy są uporządkowane
{unordered} – obiekty są nieuporządkowane
{unique} – obiekty wewnątrz cechy nie powtarzają się
{nonunique} – obiekty wewnątrz cechy mogą się powtarzać
{readOnly} – wartość atrybutu służy tylko do odczytu
{frozen} – wartość atrybutu nie może być zmodyfikowana po jej przypisaniu {sorted}
– wartość atrybutu oznacza, że są one w jakiś sposób posortowane.

Atrybuty pochodne (ang. derived) są zależne od innych atrybutów i ich wartości można obliczyć na podstawie tych atrybutów. Często w fazie implementacji są przekształcane w metody lub ich wartość jest obliczana na bieżąco. Nie ma zatem potrzeby ich zapamiętywania w klasie. Atrybuty pochodne sąoznaczane znakiem '/' umieszczonym przed nazwą atrybutu.



Przekroczenie = (data końca. Dni()- data początku. Dni())>max_wypożyczenie

Operacja to proces który klasa potrafi wykonać. Są opisywane podobnie jak atrybuty, oczywiście, z uwzględnieniem listy parametrów i pominięciem wartości domyślnej. Parametry są zapisywane identycznie jak atrybuty klasy, jednak są poprzedzone informacją o kierunku jego przekazania: in, out, inout i return. Domyślnym kierunkiem jest wejściowy.

widoczność nazwa(parametr1, parametr2,…) : typ {ograniczenia}

Parametr 1: kierunek nazwa typ [krotność] = wartość domniemana

Kierunki parametrów:

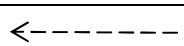
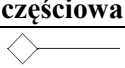
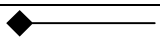
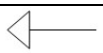
- wejściowy: in – nie może być modyfikowany
- wyjściowy: out – może być modyfikowany
- wejściowo-wyjściowy: inout – może być modyfikowany
- zwracany z metodą: return

Operacje, podobnie jak atrybuty, mogą posiadać dodatkowe właściwości i ograniczenia:
{query} –nie modyfikuje stanu obiektu, jest zapytaniem. Informacja taka jest bardzo ważna w czasie implementacji.
«exception» wraz z nazwą wyjątku jako opis klasy wyjątku – metoda może zgłaszać wyjątek

Operację można także opisywać przez dwa rodzaje warunków: wstępne i końcowe. Opisują one wymagany i oczekiwany stan części systemu wymagany odpowiednio przed i po wykonaniu operacji. Pozwala to na dokładniejsze opisane zadania realizowanego przez metodę, jej wymagań i rezultatów.

Związki między klasami:

← Słabszy związek między klasami — Silniejszy związek między klasami →

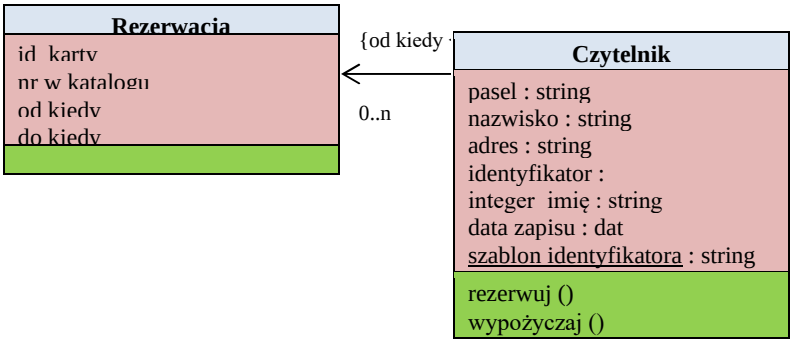
Zależność	Asocjacja	Agregacja częściowa	Agregacja całkowita	Dziedziczenie
				
obiekty jednej klasy działają stosując przelotnie obiekty innej klasy	obiekty jednej klasy działają stosując obiekty innej przez dłuższy czas	jedna klasa zawiera ale jednocześnie współdzieli odwołanie do obiektów innej	jedna klasa zawiera obiekt innej	kiedy jedna klasa jest rodzajem innej

Zależność jest najsłabszą formą relacji między klasami. Oznacza, że przez jakiś czas obiekty wskazanej klasy używają obiektów innej klasy. Przykładem takiej relacji jest użycie klasy jako parametru metody lub użycie obiektów danej klasy w metodzie innej klasy. Zmiany w interfejsie lub sposobie działania klasy, od której zależy klasa będąca początkiem relacji, może powodować konieczność zmian w klasie zależnej.

Zależność oznaczana jest linią przerywaną łączącą klasy będące w relacji a o rodzaju zależności decyduje słowo kluczowe umieszczone nad linią:

«call» - operacje w klasie A wywołują operacje w klasie B
«create» - klasa A tworzy instancje klasy B
«instantiate» - obiekt A jest instancją klasy B
«use» - do zaimplementowania klasy A wymagana jest klasa B

Asocjacje są silniejszymi relacjami niż zależności. Wskazują, że jeden obiekt jest związany z innym przez pewien okres czasu. Jednak czas życia obu obiektów nie jest od siebie zależny: usunięcie jednego nie powoduje usunięcia drugiego.



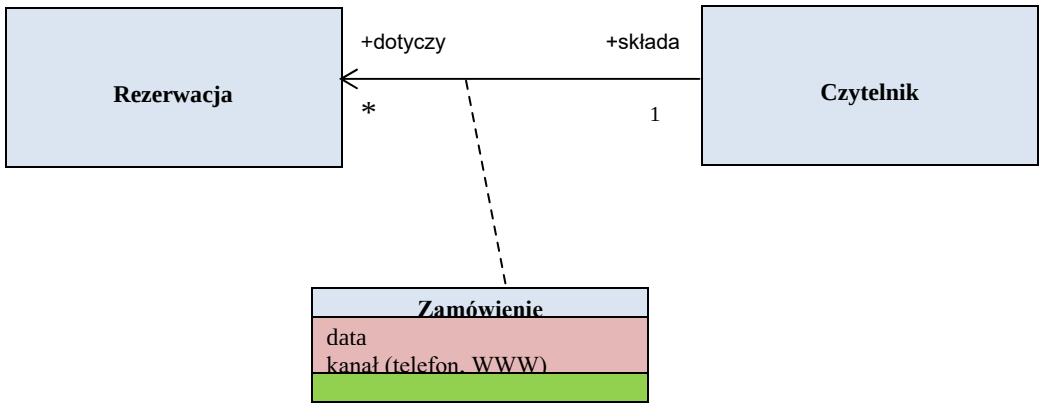
Asocjacje mogą posiadać nazwy, zwykle w postaci czasownika, który pozwala przeczytać w języku naturalnym jej znaczenie, np. „A posiada B". Często pomija się jedną z nazw asocjacji dwukierunkowej, jeżeli jest ona jedynie stroną bierną drugiej nazwy, np. „przechowuje" – „jest przechowywany".

Joanna Fabiś-Domagała
fabis@mech.pk.edu.pl

Asocjacja jest równoważna atrybutowi. Warto jednak przyjąć konwencję, w której obiekty reprezentujące wartości (np. daty) oraz typy proste (liczby, napisy, znaki) są modelowane jako atrybuty, natomiast obiekty dostępne poprzez referencje – są przedstawiane poprzez asocjacje.

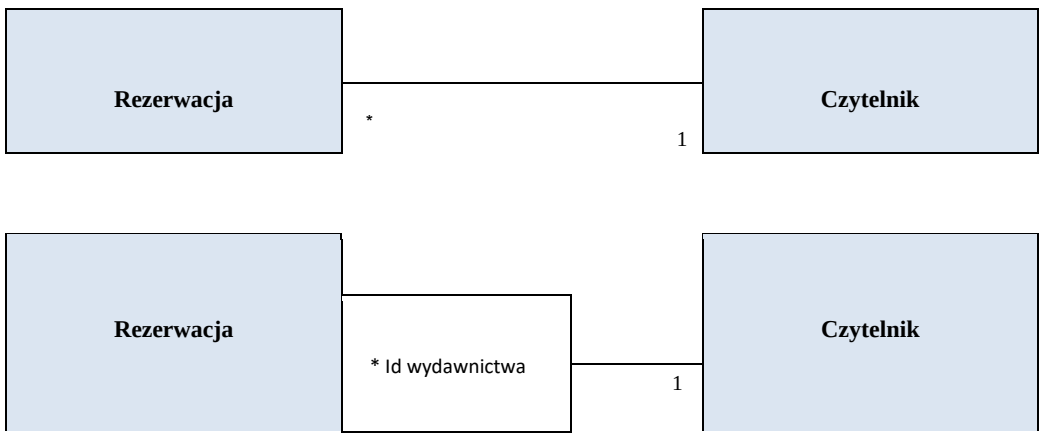
Asocjacje modelują względną równowagę pomiędzy połączonymi nimi obiektami, jednak nie oznacza to, że ich wiedza o sobie jest taka sama. Informację o kierunku relacji, czyli który obiekt może odwołać się do drugiego) opisuje kierunek asocjacji – **nawigowalność**. Wskazuje ona możliwość przejścia z obiektu klasy źródłowej A do docelowej B. Oznaczona jest otwartą strzałką, wskazującą możliwy kierunek przejścia. Domyślnie przyjmuje się dwukierunkowe przejście. W przypadku nawigowalności dwukierunkowej strzałki pomija się.

Klasa asocjacyjna umożliwia opisanie za pomocą atrybutów i operacji nie obiektu, ale samej asocjacji pomiędzy klasami. Informacje przechowywane w klasie asocjacyjnej nie są związane z żadną z klas uczestniczących w asocjacji, dlatego wygodnie jest stworzyć dodatkową klasę i powiązać ją z relacją.



Klasy asocjacyjne są reprezentowane graficznie jako klasy połączone linią przerywaną z relacją asocjacji, której dotyczą.

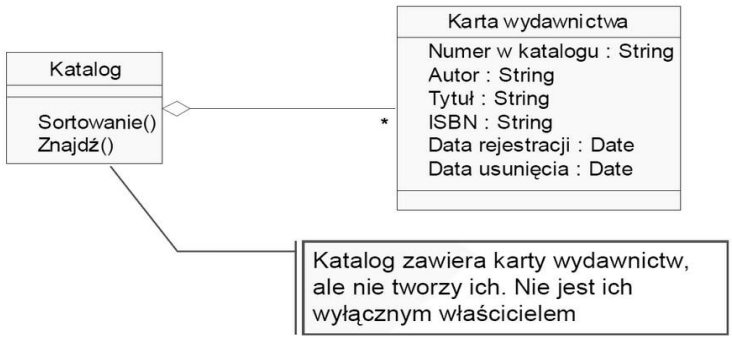
Asocjacja kwalifikowana jest rozszerzeniem zwykłej asocjacji o możliwość określenia, który z atrybutów jednej z klas decyduje o związku między nimi.



Joanna Fabiś-Domagała
fabis@mech.pk.edu.pl

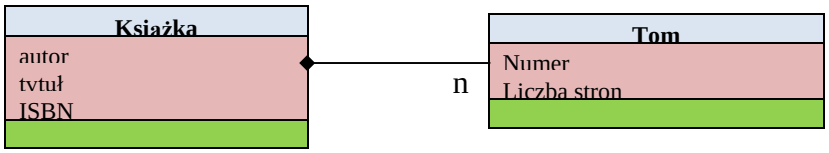
Składając Rezerwację, Czytelnik podaje listę Wydawnictw, które chciałby wypożyczyć. Jednak w danym momencie Czytelnik może zarezerwować dane Wydawnictwo tylko jeden raz – i dlatego atrybut id wydawnictwa jest kwalifikatorem tej relacji. W efekcie pomiędzy instancją Czytelnika a instancją Rezerwacji występuje relacja jeden-jeden, ponieważ konkretny Czytelnik rezerwuje konkretne Wydawnictwo w danym momencie tylko raz.

Agregacja jest silniejszą formą asocjacji. W przypadku tej relacji równowaga między powiązanymi klasami jest zaburzona: istnieje właściciel i obiekt podrzędny, które są ze sobą powiązane czasem swojego życia. Właściciel jednak nie jest wyłącznym właścicielem obiektu podrzędnego, zwykle też nie tworzy i nie usuwa go.



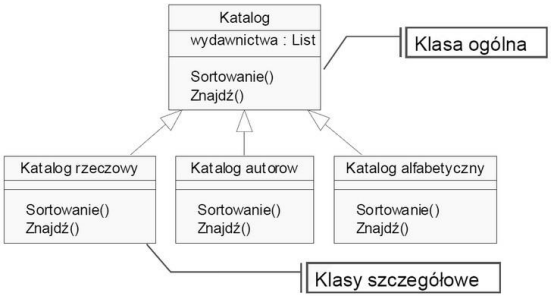
Relacja agregacji jest zaznaczana linią łączącą klasy/obiekty, zakończoną białym rombem po stronie właściciela.

Kompozycja jest silniejszym wariantem asocjacji typu ”całość - część”. Czas życia połączonych obiektów jest ze sobą ściśle związany. Obiekty pełniące rolę ”części” są niszczone gdy niszczona jest ”całość” i nie istnieją poza agregatem. Kompozycja wyróżniana jest pełnym symbolem romba po stronie klasy, która pełni rolę ”całości”.



Kompozycja jest przedstawiana na diagramie podobnie jak agregacja, przy czym romb jest wypełniony.

Uogólnienie tworzy hierarchię klas, od ogólnych do bardziej szczegółowych. Pozwala wyłączyć części wspólne klas.



Joanna Fabiś-Domagała
fabis@mech.pk.edu.pl

Klasyfikacja obiektu reprezentuje (w odróżnieniu od relacji uogólnienia/uszczegółowienia) związek pomiędzy obiektami a klasami. Klasyfikacja obiektu określa, z którymi typami (klasami) jest powiązany – poprzez dziedziczenie, interfejsy.

Ponieważ obiekt może jednocześnie uczestniczyć w wielu niezależnych klasyfikacjach (a zatem posiadać wiele typów, niekoniecznie poprzez dziedziczenie), dlatego do szczegółowego określenia klasyfikacji stosowane są uściślające słowa kluczowe:

{overlapping} – obiekt może należeć do kilku klas
{disjoint} – obiekt może należeć tylko do jednej klasy
{complete} – nie istnieje więcej podklas
{incomplete} – mogą powstać kolejne podklasy.

Szablon klasy określa, z jakimi innymi klasami (nie obiektami!) może współpracować podana klasa. Klasy te są przekazywane jako jej parametry. Klasa będąca uszczegółowieniem takiej klasy jest klasą związaną.

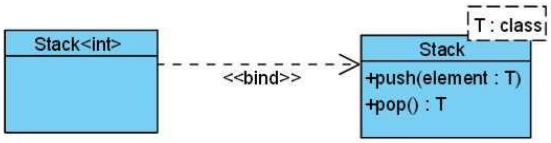


Diagram obiektów podobne do diagramu klas, przedstawiają jednak nie klasy, tylko konkretne obiekty będące instancjami klas systemu. Z punktu widzenia notacji diagramy obiektów używają elementów zapożyczonych z diagramów klas, chociaż często używają prostszej notacji. Skupiają się na obiektach a nie na związkach pomiędzy klasami. Większość z nich używa wyłącznie obiektów i asocjacji. Diagram jest więc wizualizacją hipotetycznego stanu systemu podczas jego działania. Służy do tworzenia przykładów pomagających zrozumieć diagram klasa przede wszystkim powiązań w nim występujących. Obiekty są identyfikowane poprzez umieszczenie przykładowej nazwy poprzedzonej dwukropkiem ':' przed nazwą klasy, którą reprezentują.

```
nazwa_obiektu : nazwa_klasy
nazwa_atrybutu1 = wartość_atrybutu1
nazwa_atrybutu2 = wartość_atrybutu2
...
nazwa_atrybutuN = wartość_atrubutuN
```

Diagram reprezentuje Jasia Kowalskiego i jego ulubieńców:

Joanna Fabiś-Domagała
fabis@mech.pk.edu.pl

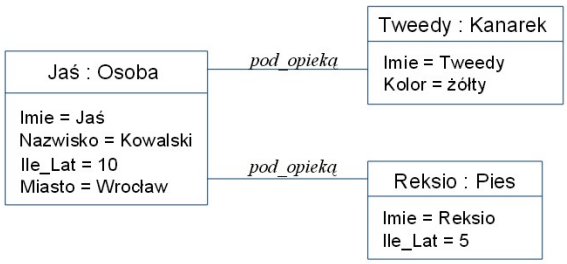


Diagram struktur złożonych przedstawia wewnętrzną strukturę obiektu oraz punkty interakcji z innymi obiektami w systemie.

Jezyk UML 2.0 w modelowaniu systemów informatycznych, S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Helion
http://wazniak.mimuw.edu.pl/index.php?title=Io-5-wyk-toc