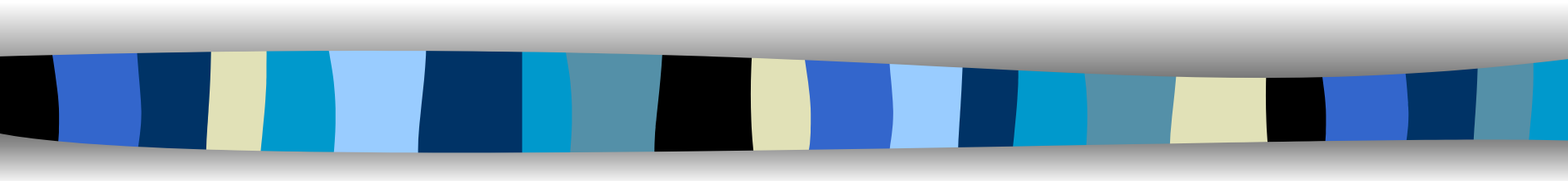
# Diagramy klas i obiektów



**zastosowanie do modelowania w języku UML**

Agnieszka Niegowska
&
Grzegorz Widziszowski

# Klasy

**Klasa jest** miejscem przechowywania cech obiektów, które są niezmienne (inwariantów). Klasa **nie jest** zbiorem obiektów i **nie jest** definicją zbioru obiektów. Stosunek klasa/podklasa oznacza, że obiekty podklasy posiadają wszystkie inwarianty nadklasy, plus swoje inwarianty. Np. klasa *Student* ma wszystkie inwarianty klasy *Osoba*, plus niektóre własne.

| | | |
|---|---|---|
| najważniejsze inwarianty | **nazwa** | językowy identyfikator obiektu |
| | **typ** | czyli statyczna budowa obiektu (atrybuty) |
| | **metody** | operacje, które można wykonać na obiekcie |
| inne możliwe | **zdarzenia lub wyjątki** | mogące zajść w operacjach na obiekcie |
| | **lista eksportowa** | określająca, które atrybuty dostępne są z zewnątrz |
| | **Ograniczenia** | którym msi podlegać każdy obiekt |
| | **...** | |

# Diagramy klas

- są odmianą klasyczną diagramów encja-związek (entity-relationship) rozbudowanymi o nowe elementy
- dużo oznaczeń o charakterze pomocniczym (np.: notatki i ograniczenia)
- rodzajem diagramów klas są diagramy pakietów (package diagrams)

Żadna klasa nie żyje w izolacji – działa w kooperacji z innymi, aby zrealizować działanie niemożliwe do wykonania w pojedynkę

Diagram klas służy do zobrazowania współpracy klas.

# Zastosowanie diagramów klas

## zapis modelu pojęciowego

reprezentują pojęcia w dziedzinie zastosowań, które aktualnie podlegają analizie – sformalizowana wizja wyobrażeń powstających podczas myślenia nad problemem

## sformalizowana specyfikacja danych i metod

dotyczy opisu zewnętrznego oprogramowania bez szczegółów implementacyjnych

## implementacja

może bezpośrednio służyć jako graficzny środek pokazujący szczegóły implementacji

# Tworzenie diagramu klas

- Identyfikacja klas i obiektów
- Identyfikacja związków pomiędzy klasami
- Identyfikacja i definiowanie pól (atrybutów)
- Identyfikacja i definiowanie metod i komunikatów

kolejność wykonywania nie jest ustalona i zależy zarówno od stylu pracy, jak i od konkretnego problemu.

**Dobrze zbudowany diagram klas:**

- uwypukla jeden statyczny aspekt perspektywy projektowej systemu
- obejmuje tylko byty niezbędne do zrozumienia tego aspektu
- zawiera szczegóły odpowiednie do przyjętego poziomu abstrakcji, z dodatkami koniecznymi do zrozumienia tego, na czym zależy projektantowi
- nie jest zbyt ogólny

# Reprezentacja graficzna

Klasa obrazowana jest za pomocą podzielonego na części prostokąta – każda część reprezentuje inwarianty o zbliżonym przeznaczeniu


Na diagramie klasy dodatkowo określa się widoczność atrybutów i metod przez umieszczanie przed nimi odpowiedniego symbolu:

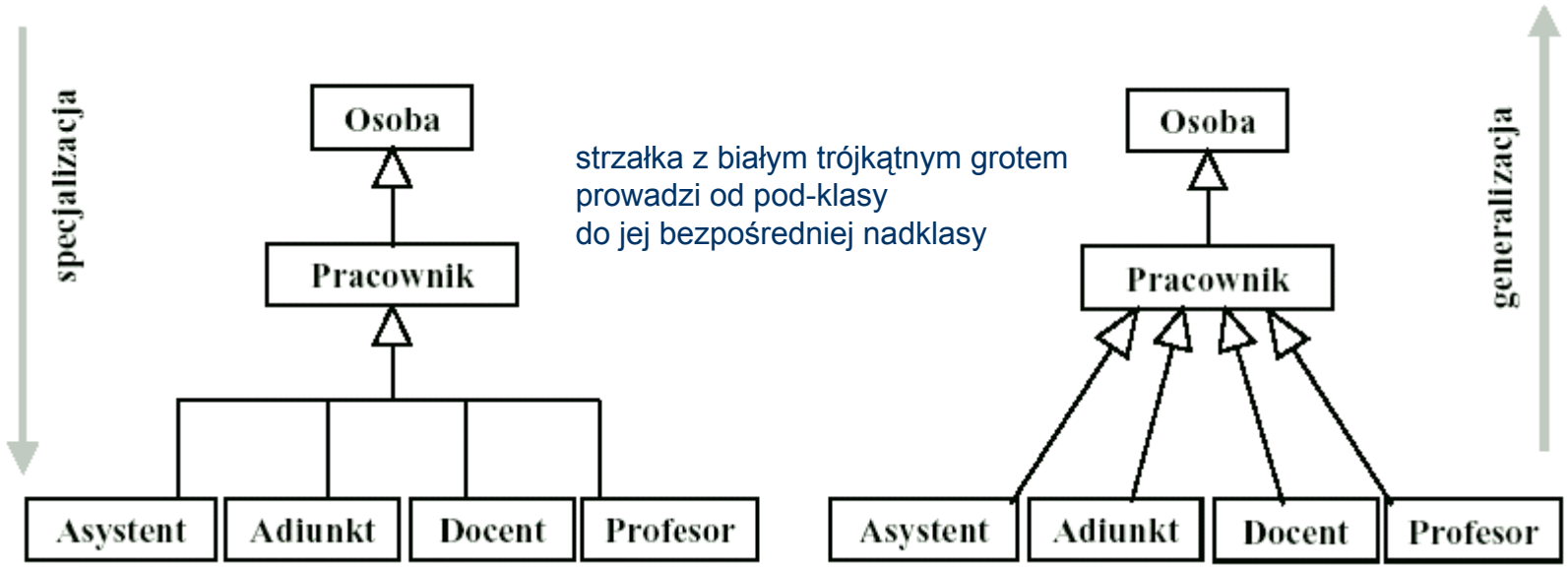
**+** publiczne *(public)*

**-** prywatne *(private)*

**#** chronione *(protected)*

# Dziedziczenie

Obiekt pod-klasy automatycznie dziedziczy wszystkie atrybuty, metody, asocjacje i agregacje z wszystkich jej nadklas.

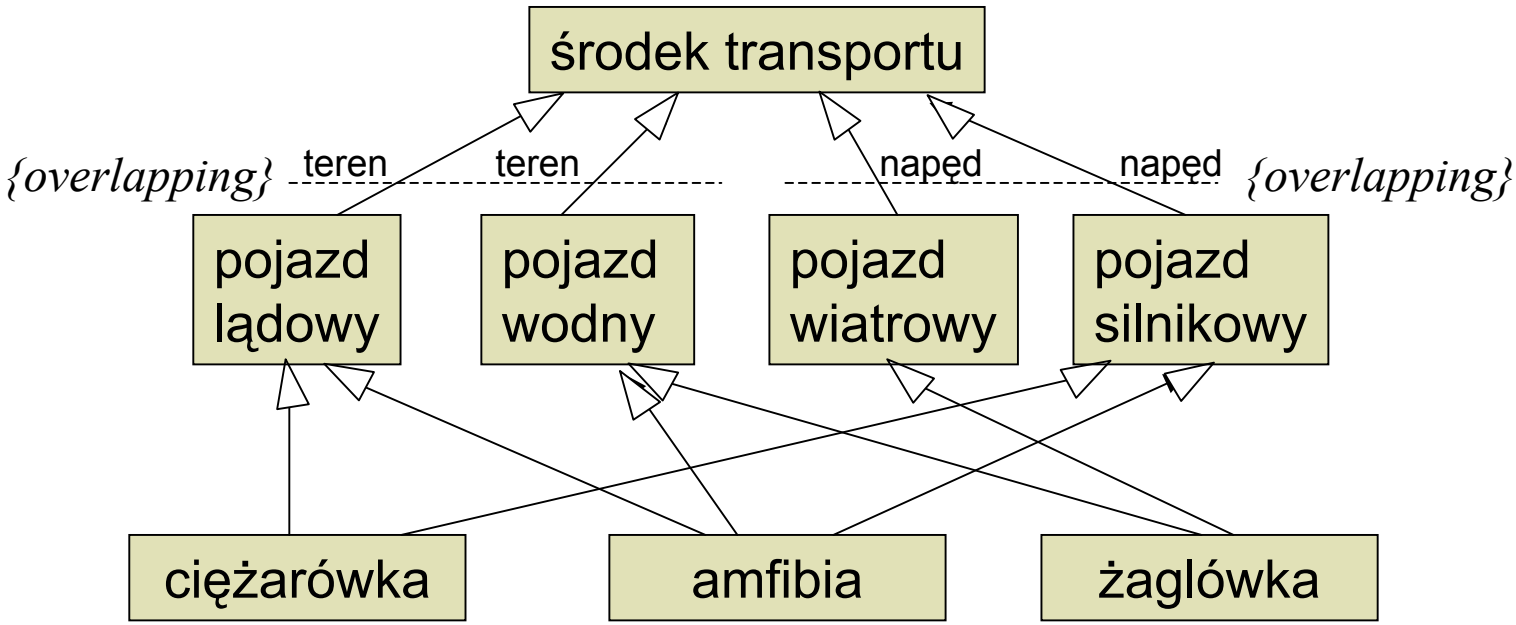



SPECJALIZACJA
– budowa pojęć bardziej szczegółowych, gdy mamy bardziej ogólne

GENERALIZACJA
– budowa pojęć bardziej ogólnych, gdy mamy bardziej szczegółowe

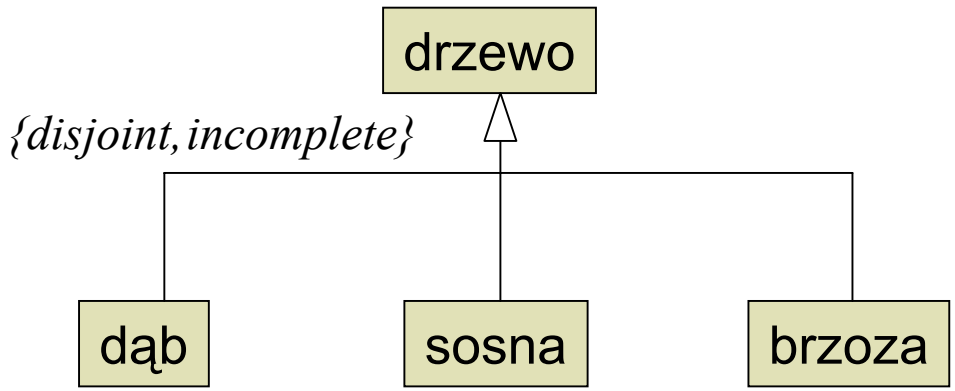
# Dziedziczenie wieloaspektowe

Stosuje się specjalne oznaczenia, gdy zakresy znaczeniowe klas nie są rozłączne. Można wtedy zadeklarować ***aspekt*** według którego specjalizuje się dany obiekt oraz oznaczyć ten fakt




*overlapping* – podział nierozłączny (przecięcie zakresów znaczeniowych nie jest zbiorem pustym)

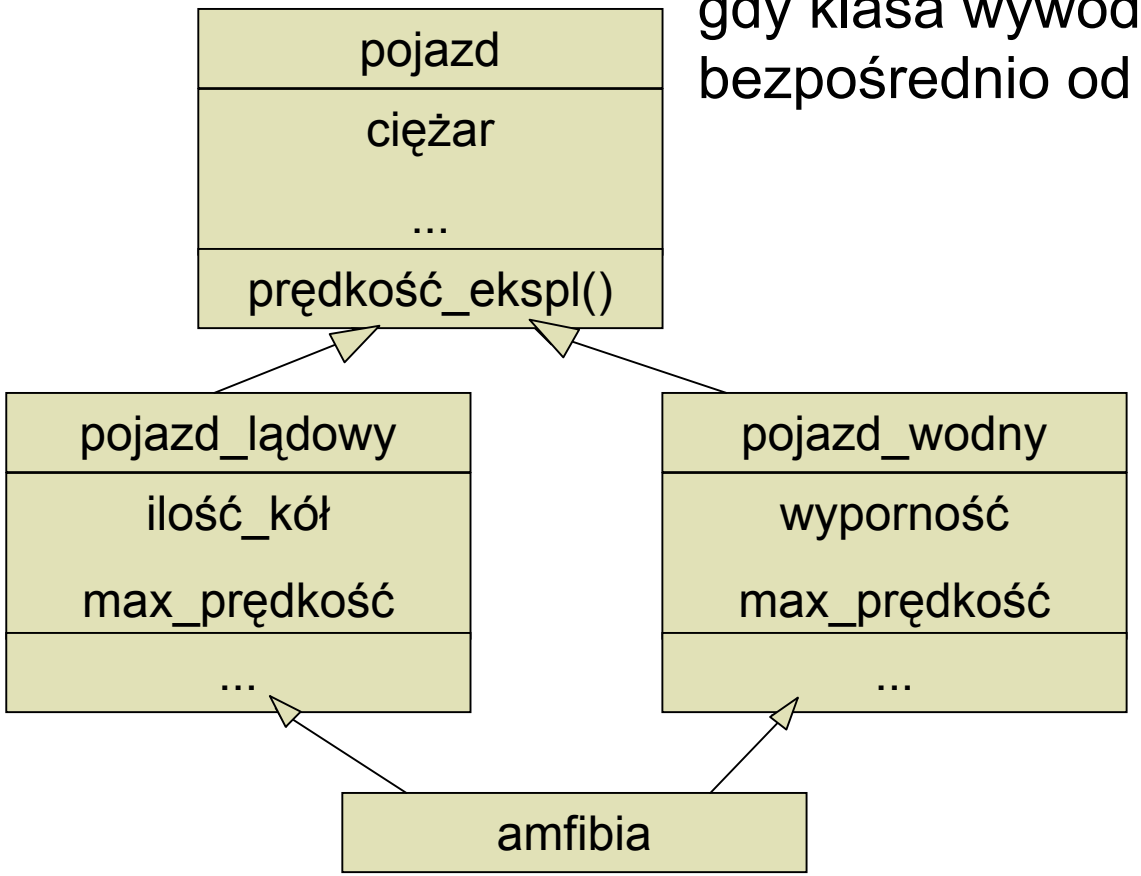
# Dziedziczenie wieloaspektowe




Dodatkowo można również odnotować fakty, że:

- podklasy są rozłączne *(disjoint)*
- nie pokrywają całego zakresu znaczeniowego ich nadklasy *(incomplete)*

# Dziedziczenie wielokrotne

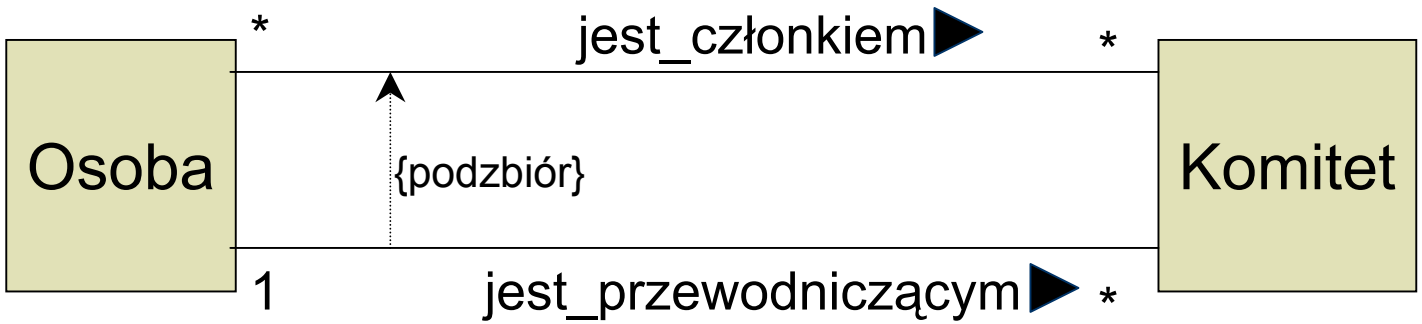


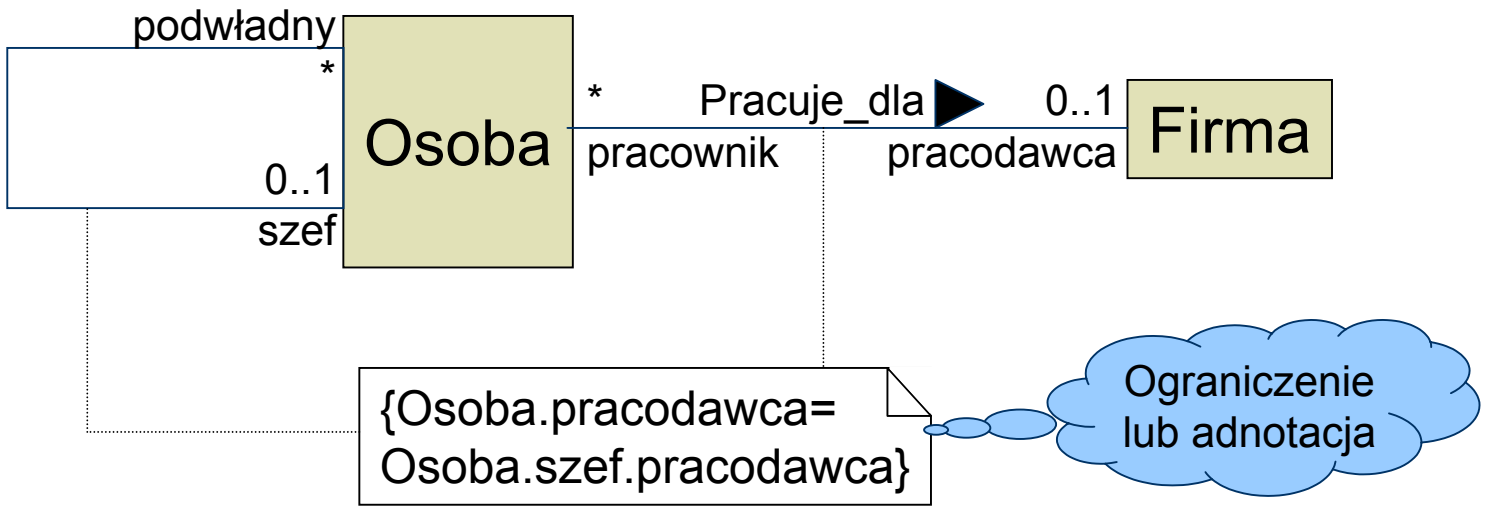

gdy klasa wywodzi się bezpośrednio od **więcej niż 1** klasy

Wielokrotnie dziedziczenie prowadzi do anomalii i wad koncepcji. W większości anomalie są konsekwencją faktu, że przy pomocy wielodziedziczenia próbuje się opanować koncepcję dynamicznych ról obiektu.

# Ograniczenia

Na diagramie klas można zawrzeć ograniczenia dotyczące klas oraz związków między nimi.

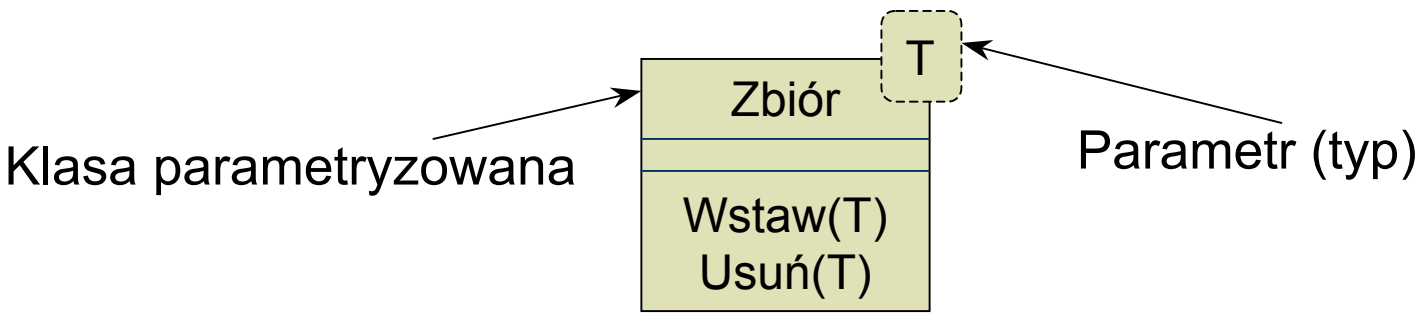


# Szablony klas

Niektóre języki obiektowe, wśród nich C++, wprowadzają użyteczne pojęcie klasy parametryzowanej (zwane też szablonem, *template*). UML wprowadza specjalne oznaczenie dla klas parametryzowanych, które może być użyte w diagramie klas.

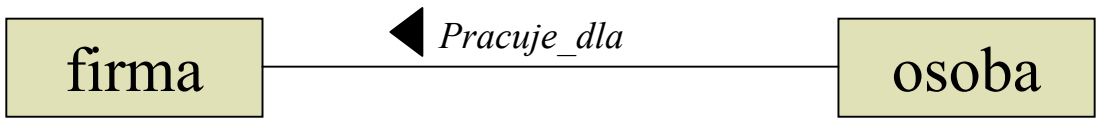
```
class Zbiór <T> {
                    void wstaw ( T nowyElement );
                    void usuń ( T pewienElement );
}
```



# Asocjacje

czyli powiązania pomiędzy obiektami klas

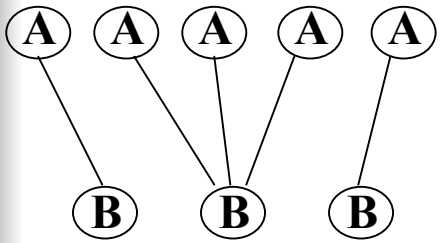
Poniżej widoczny jest przykład specyfikacji asocjacji pomiędzy obiektami klasy *Osoba* i obiektami klasy *Firma*. Czarny trójkącik określa kierunek wyznaczony przez nazwę powiązania.



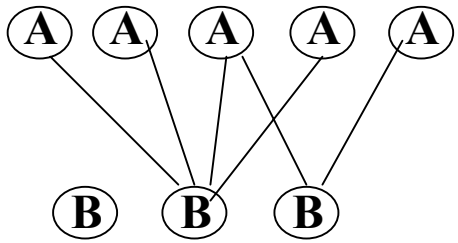

Asocjacje mają nazwy, które wyznaczają znaczenie tej asocjacji w modelu pojęciowym. Jeżeli to znaczenie jest oczywiste, wówczas nazwę asocjacji można pominąć.

# Asocjacje - liczność

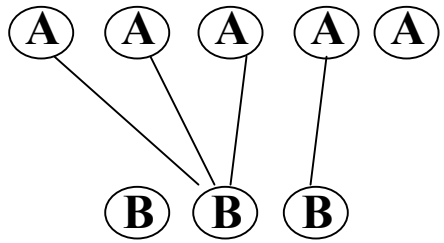
Asocjacje mogą być wyposażone w oznaczenia liczności:



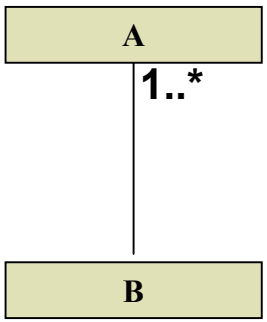

A→B : min=1, max=1
B→A : min=1, max=N

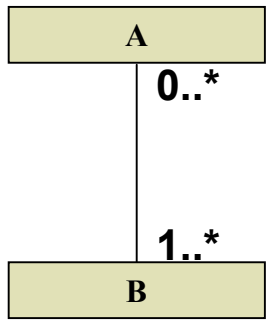


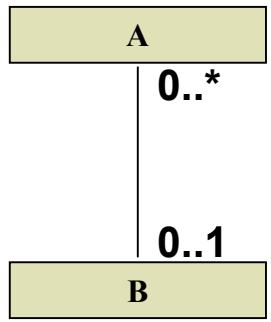

A→B : min=1, max=N
B→A : min=0, max=N




A→B : min=0, max=1
B→A : min=0, max=N

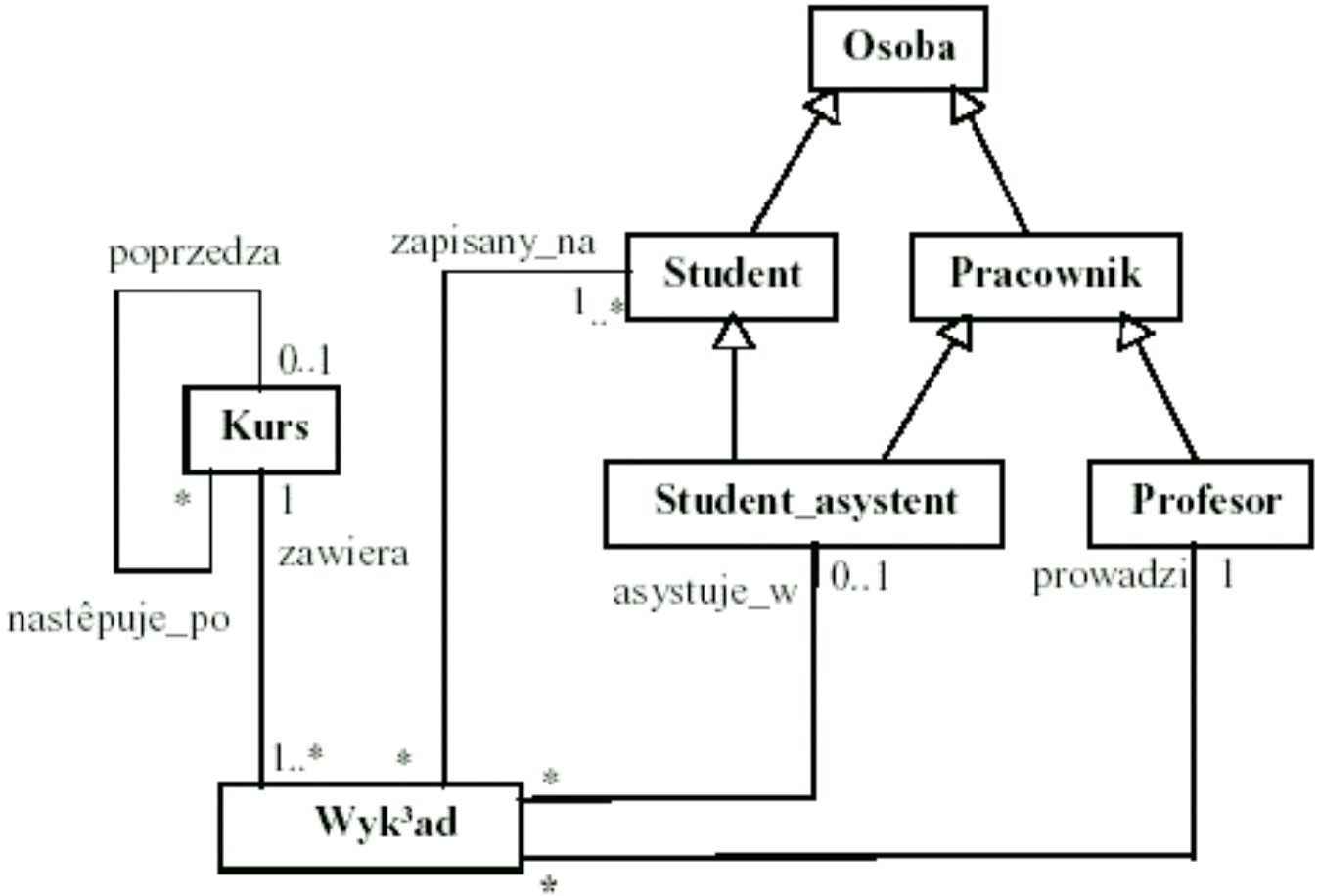


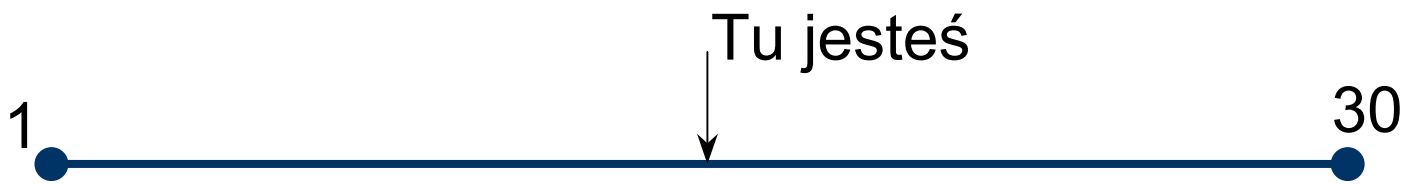




Liczność oznacza, ile obiektów innej klasy może być powiązane z jednym obiektem danej klasy (para liczb oznaczająca ilość minimalną i maksymalną)

# Przykład

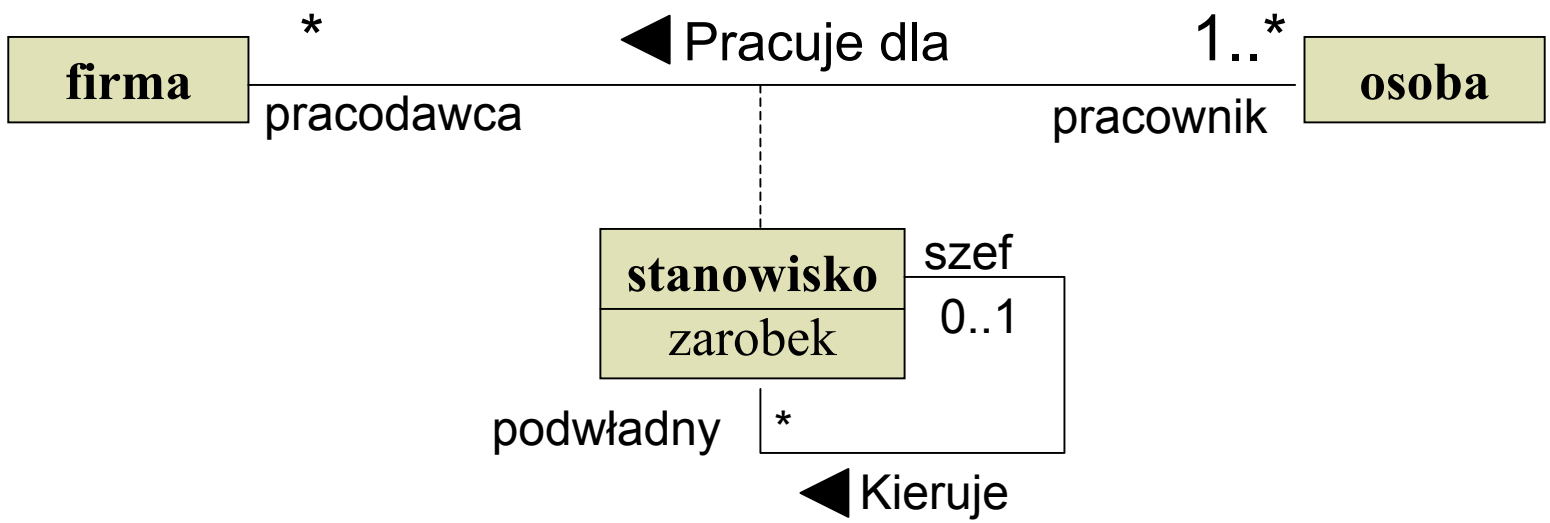

Osoba
Student
Pracownik
Student_asystent
Profesor
Kurs
Wyk³ad
poprzedza
0..1
*
1
zawiera
nastêpuje_po
zapisany_na
1..*
*
1..*
*
asystuje_w
0..1
prowadzi
1
*
*

# To już połowa



# Asocjacje – nazwy ról

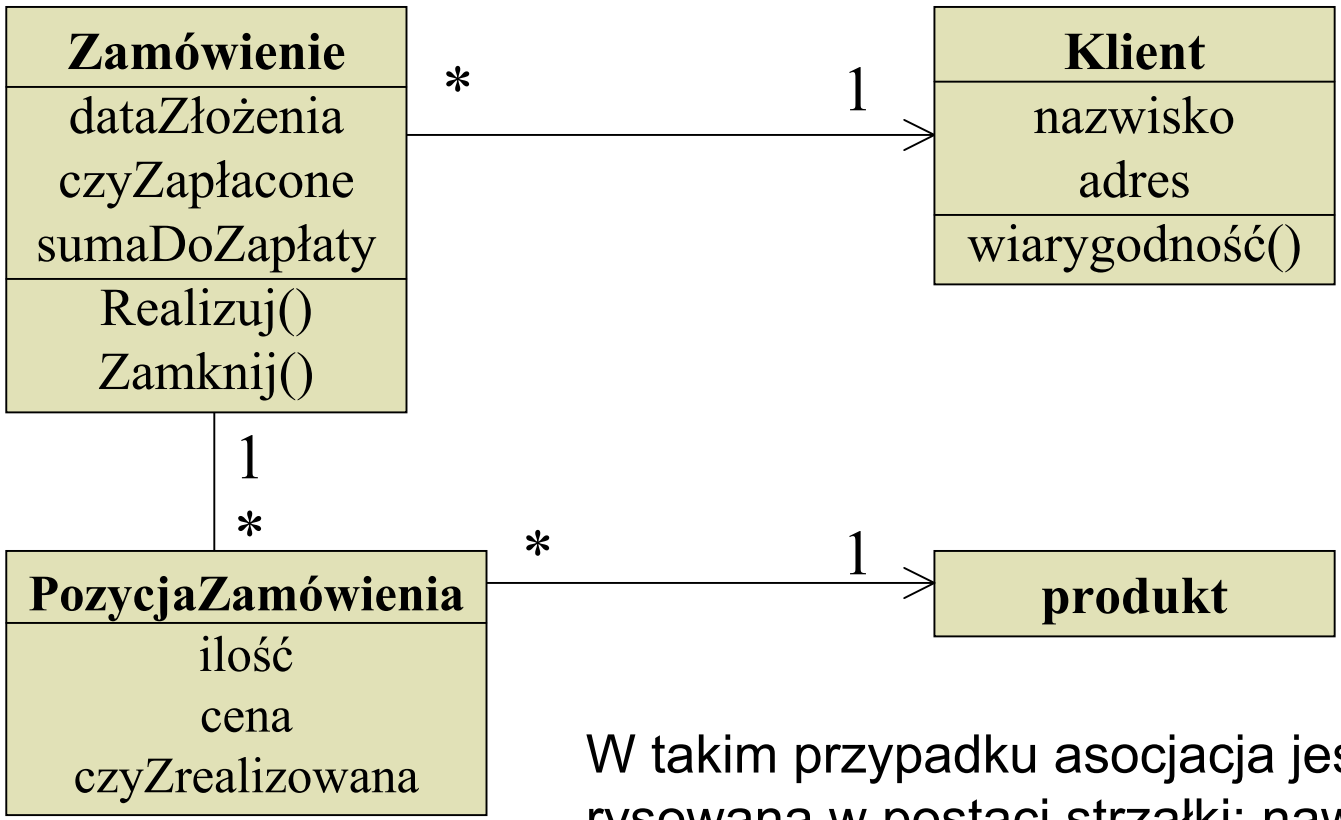
Asocjacje mogą być także wyposażone w dodatkowe nazwy ról (przy odpowiednich końcach).




Asocjacje mogą posiadać atrybuty. W tym celu przewidziano linię przerywaną łączącą daną asocjacją z klasą, określaną jako: „klasa asocjacji”. Wewnątrz klasy asocjacji można zdefiniować atrybuty, operacje i inne cechy asocjacji. Klasa asocjacji może być uważana za samodzielną klasę - w szczególności podlega związkom dziedziczenia i asocjacji. Asocjacje mogą być również n-arne (oznaczenie - pusty w środku romb).

# Asocjacje skierowane

Na diagramach UML można zaznaczyć kierunek nawigacji wzdłuż danej asocjacji




W takim przypadku asocjacja jest rysowana w postaci strzałki; nawigacja jest możliwa zgodnie z jej kierunkiem, ale nie odwrotnie.

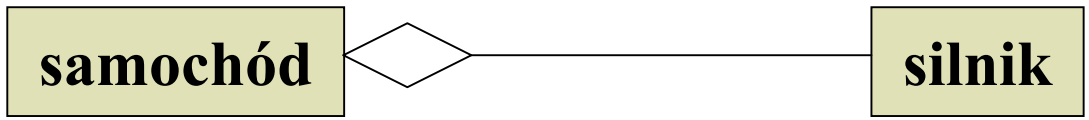
# Agregacje

szczególny przypadek asocjacji wyrażający zależność: część – całość.

Oznacza się je za pomocą pustego rombu.

*Przykład 1*

samochód jest agregatem swoich części




*Przykład 2*

Klasa emerytowanych pracowników dziedziczy zarówno od klasy Emeryt, jak i od klasy Pracownik, wówczas obiekt emerytowanego pracownika zawiera jako swoją „część” inny obiekt grupujący informację o cechach osoby jako emeryta.

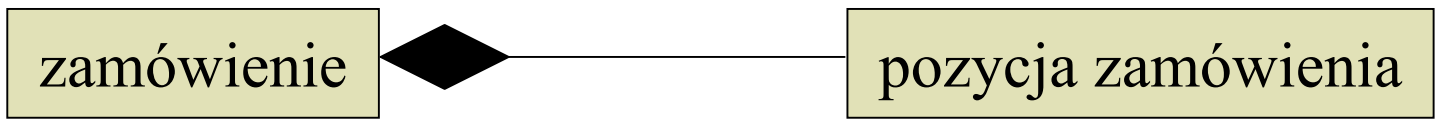
Mówi się, że obiekty pracownika i emeryta pozostają ***w związku agregacji*** (emeryt „jest częścią” pracownika).

# Kompozycja

*bardziej restrykcyjna agregacja*

dana część może należeć tylko do jednej całości

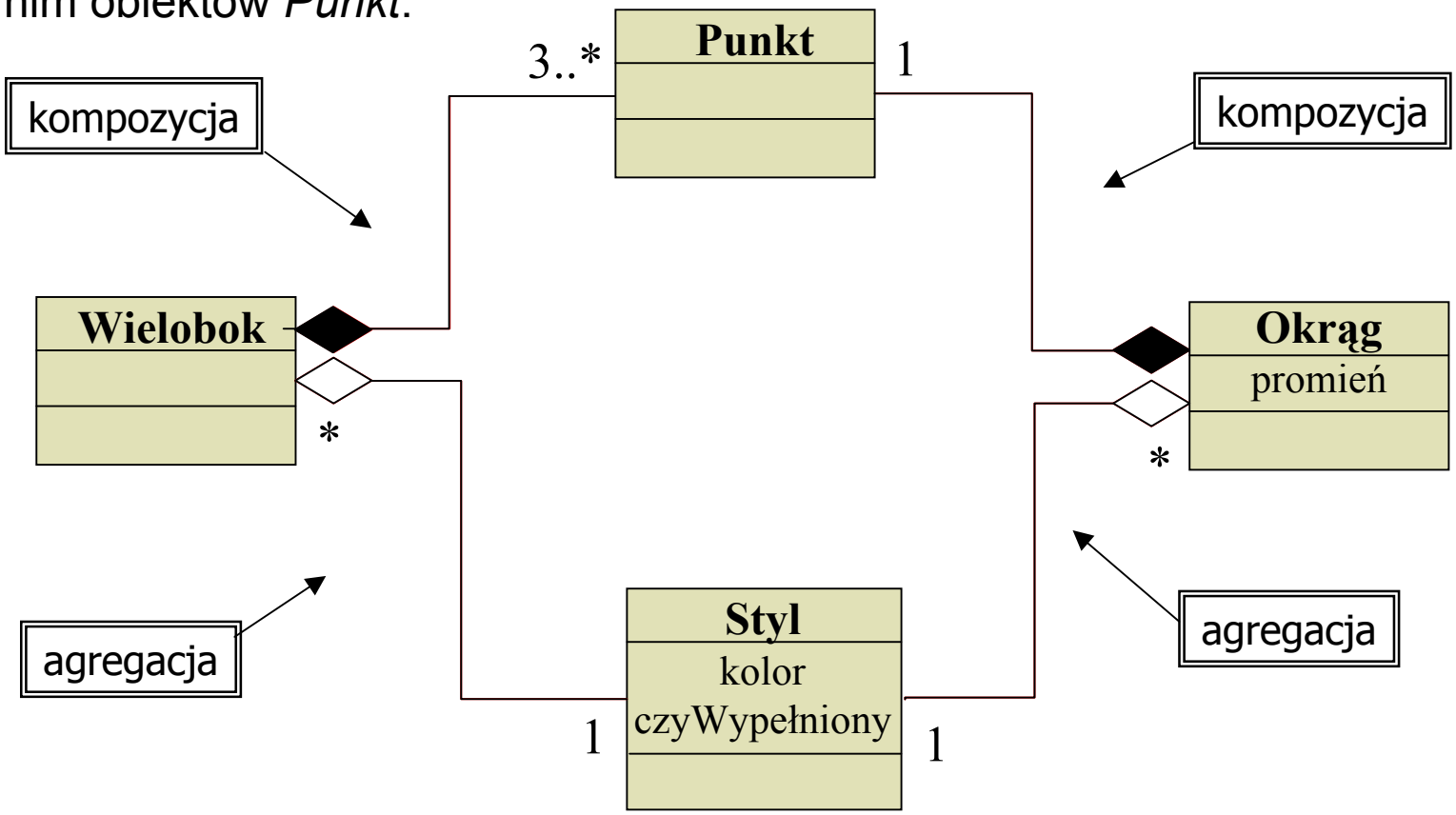
Co więcej, część nie może istnieć bez całości
– pojawia się i jest usuwana wraz z całością.
Zatem usunięcie całości powoduje automatyczne usunięcie wszystkich części związanych z nią związkiem kompozycji.




Kompozycje oznacza się za pomocą wypełnionego rombu.

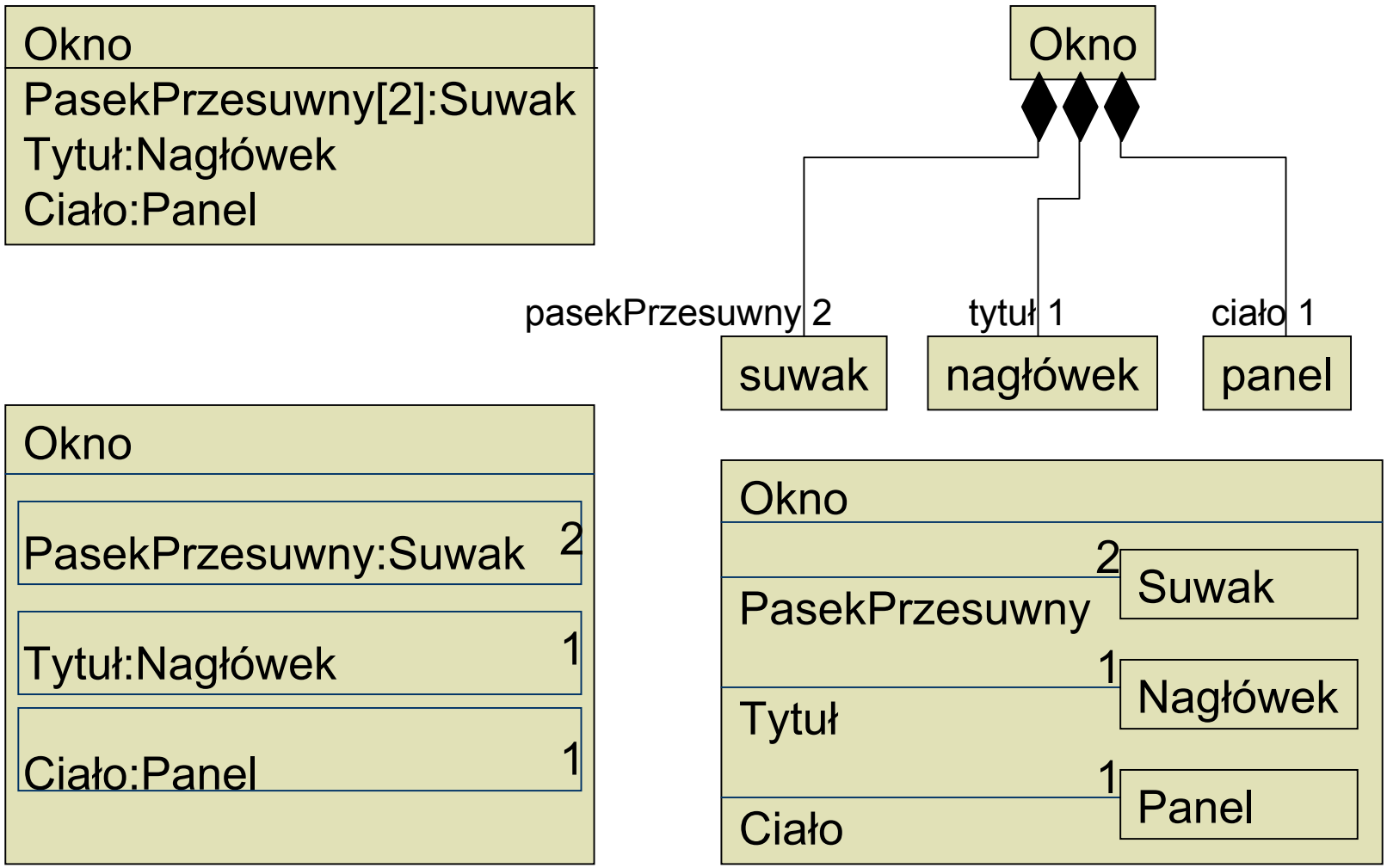
# Przykład–agregacja i kompozycja

Każde wystąpienie obiektu *Punkt* należy do obiektu *Wielobok* albo do obiektu *Okrąg* - nie może należeć do dwóch obiektów naraz a usunięcie obiektu *Wielobok (Okrąg)* powoduje kaskadowe usunięcie wszystkich związanych z nim obiektów *Punkt*.




Wystąpienie obiektu *Styl* może być dzielone przez wiele obiektów *Wielobok* i *Okrąg,* których usunięcie nie powoduje usunięcia związanego z nimi obiektu *Styl*.

# Różna postać zapisu kompozycji

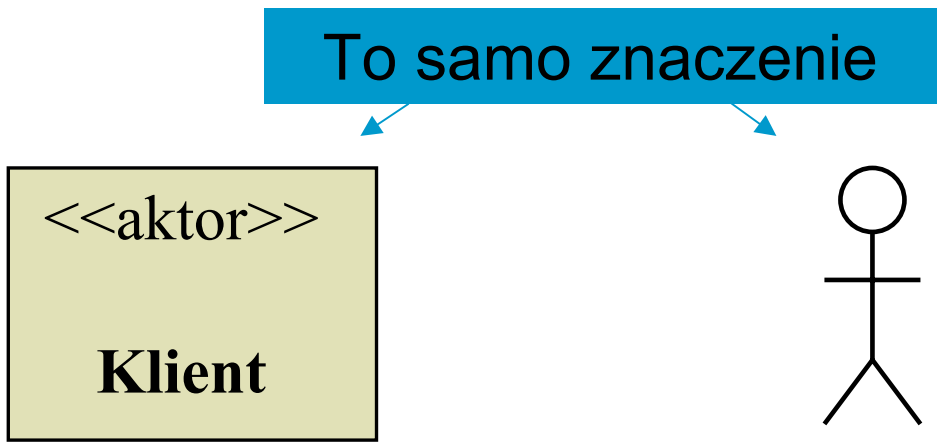


# Stereotypy

Idea : ustalenie meta-klasyfikacji obiektów
i wprowadzenie oznaczeń graficznych a nią zgodnych

Oznaczeniami są **ciągi znaków** wewnątrz nawiasów « »(np. «control object»)
Mogą one występować w różnych kontekstach oraz mogą być zastąpione przez specjalne **ikony**.(nie są określone - mogą być dowolnie wybrane)

Wewnątrz diagramów klas mogą występować stereotypy klas, asocjacji i generalizacji, itd. Stereotypy są pewnymi wspólnymi nazwanymi własnościami tych bytów, dzięki czemu ich definicja może ulec uproszczeniu lub uszczegółowieniu.



# Rodzaje stereotypów

WĘZŁÓW

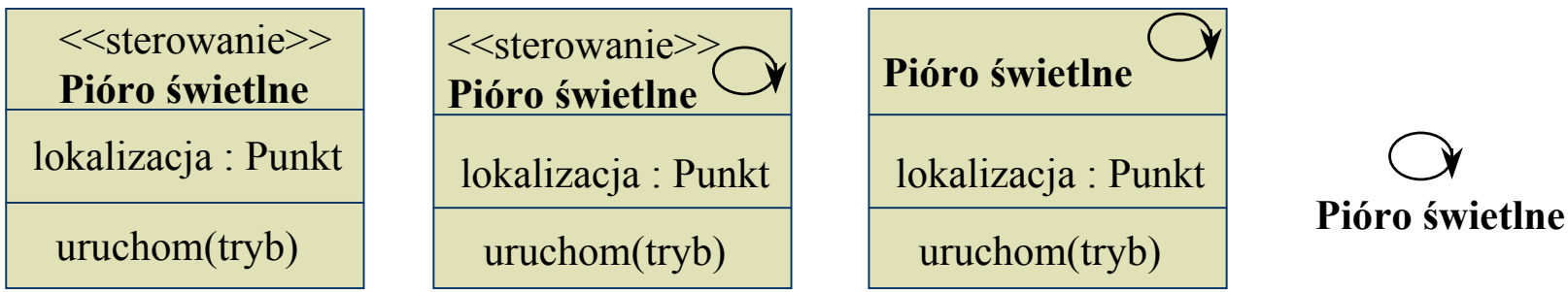
KLAS I OBIEKTÓW
- *zdarzenie*
- *wyjątek*
- *interfejs*
- *metaklasa*
- *udogodnienie*

TYPU OBIEKTÓW
- *obiekty rzeczywiste*
- *obiekty sterujące*
- *obiekty interfejsu*

PAKIETÓW

ZADAŃ
- *proces*
- *wątek*

*Przykład równoważnych oznaczeń:*



# Diagramy obiektów (Object diagram)

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 klas a przede wszystkim powiązań w nim występujących.

# Obiekty i asocjacje

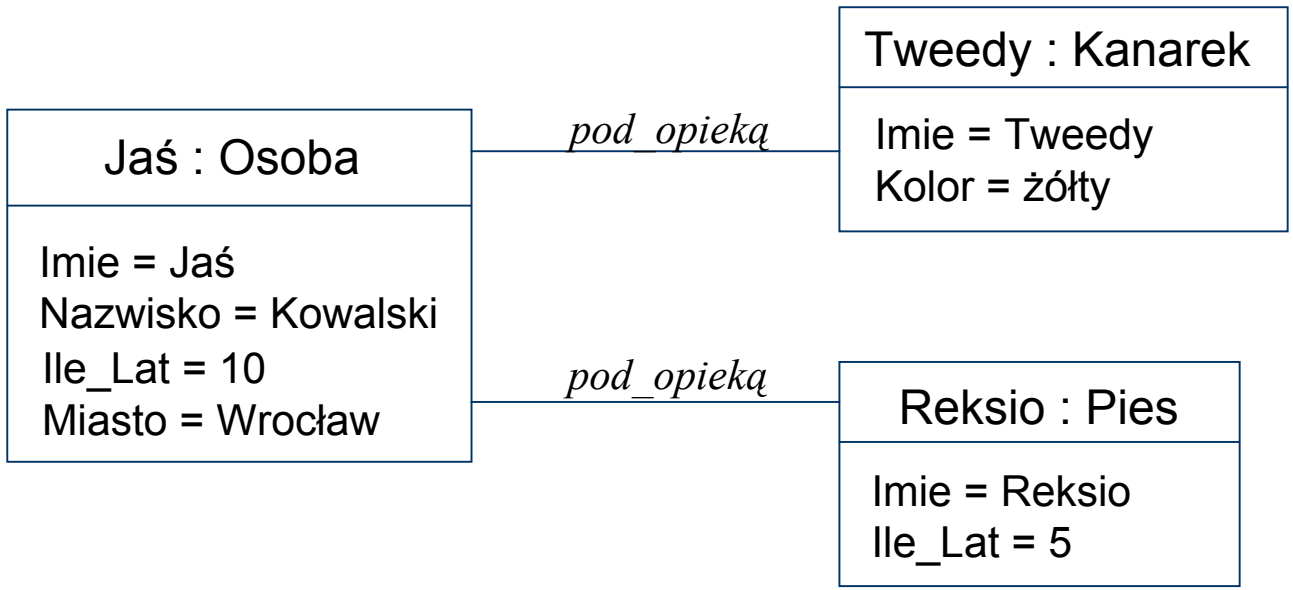
OBIEKTY są identyfikowane poprzez umieszczenie przykładowej nazwy poprzedzonej dwukropkiem ‘**:**’ przed nazwą klasy, którą reprezentują

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

ASOCJACJE oznaczane są za pomocą odcinków łączących obiekty. Na odcinkach tych dodatkowo umieszcza się nazwy asocjacji.

# Przykład diagramu obiektów

Diagram reprezentuje Jasia Kowalskiego i jego ulubieńców. Jaś ma 10 lat i mieszka we Wrocławiu. Opiekuje się dwoma zwierzakami: psem-Reksiem i kanarkiem-Tweedym



# Diagramy pakietów

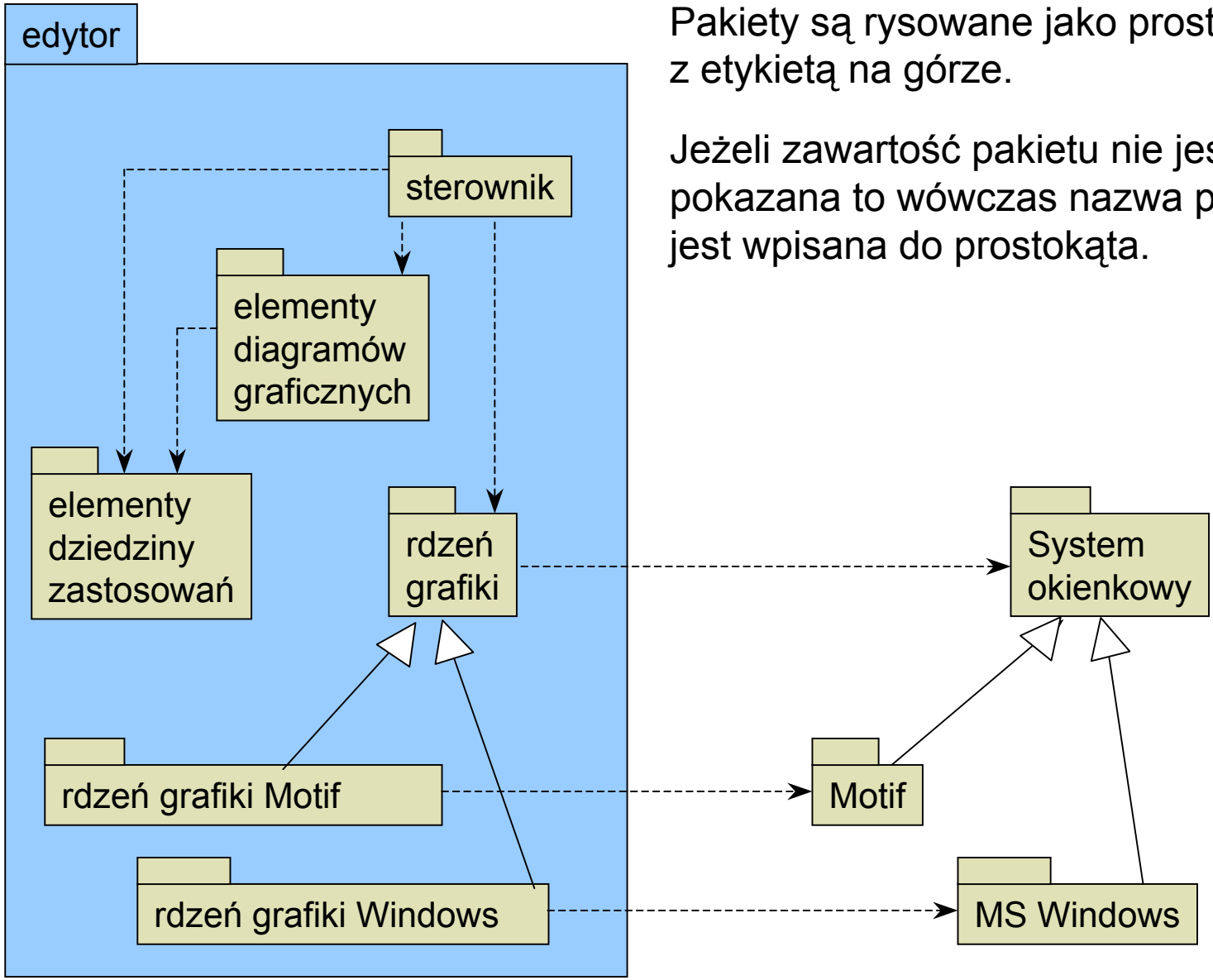
•Pakiety są zestawami elementów modelu wraz z zachodzącymi pomiędzy nimi zależnościami.

•Diagramy pakietów mogą być istotne dla dużych projektów, składających się z wielu modułów funkcjonalnych ze złożonymi zależnościami pomiędzy modułami.

• Pakiety mogą być zagnieżdżane. Zależności pomiędzy pakietami są przedstawiane w postaci strzałki z przerywaną linią.

•Pakiety są traktowane jako obiekty należące do swoich klas, które mogą podlegać związkom dziedziczenia.

# Diagramy pakietów




Pakiety są rysowane jako prostokąty z etykietą na górze.

Jeżeli zawartość pakietu nie jest pokazana to wówczas nazwa pakietu jest wpisana do prostokąta.


Koniec