

Wykład 4-5 – Modelowanie strukturalne

Pojęcia podstawowe: klasa, obiekt, zależność

Diagram klas

Proces modelowania struktury oprogramowania

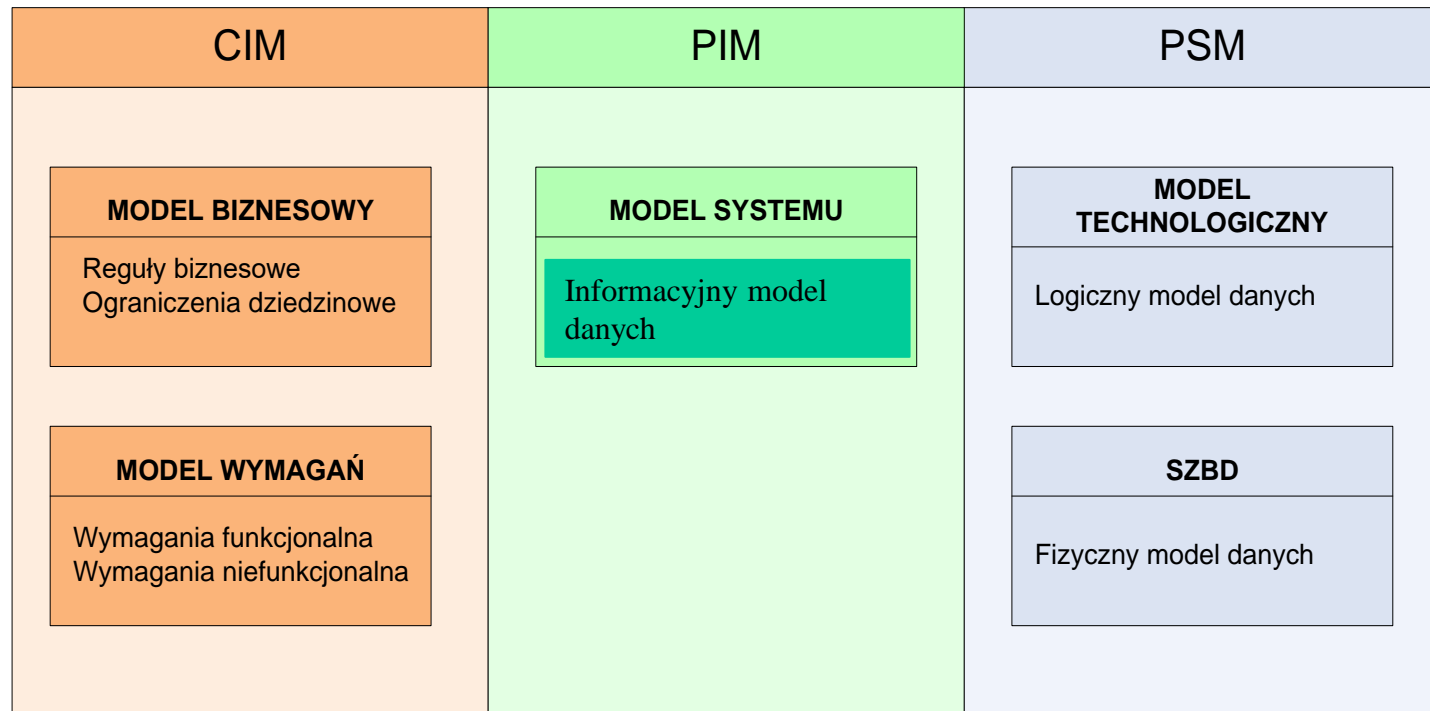
Przykłady

*Wszystko co można sobie wyobrazić, da się zamodelować za
pomocą UML*

G. Booch, J. Rumbaugh, I. Jacobson



Poziomy modelowania systemów informatycznych (MDA) – aspekt statyczny



Model statyczny

Opisuje statyczną strukturę systemu (wyrażony diagramem klas/obiektów)

- Elementy modelu

- **Klasa**

- **opis własności** (atrybuty, operacje, relacje, semantyka) **zbioru obiektów** (jest ich ‘wzorcem’)
 - opis typu

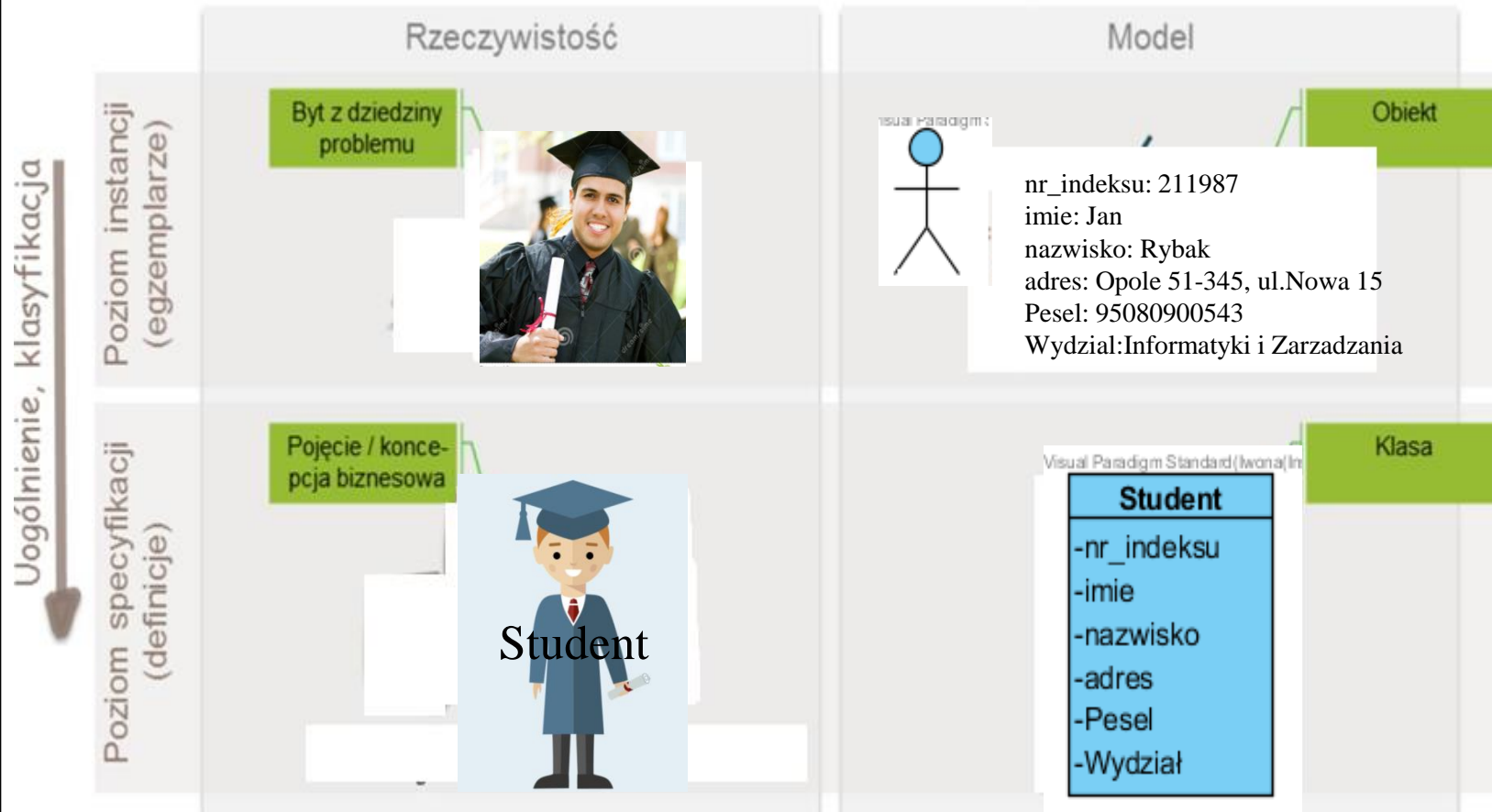
- **Obiekt**

- jednoznacznie identyfikowany byt, ściśle rozgraniczony od swego otoczenia, ukrywający własny stan i zachowanie;
 - stan obiektu jest reprezentowany przez jego atrybuty oraz relacje z innymi obiektami, a zachowanie przez operacje, metody oraz maszyny stanów
 - instancja pewnej klasy

- **Relacje**



Poziomy procesu modelowania - przykład



Diagramy klas - przeznaczenie

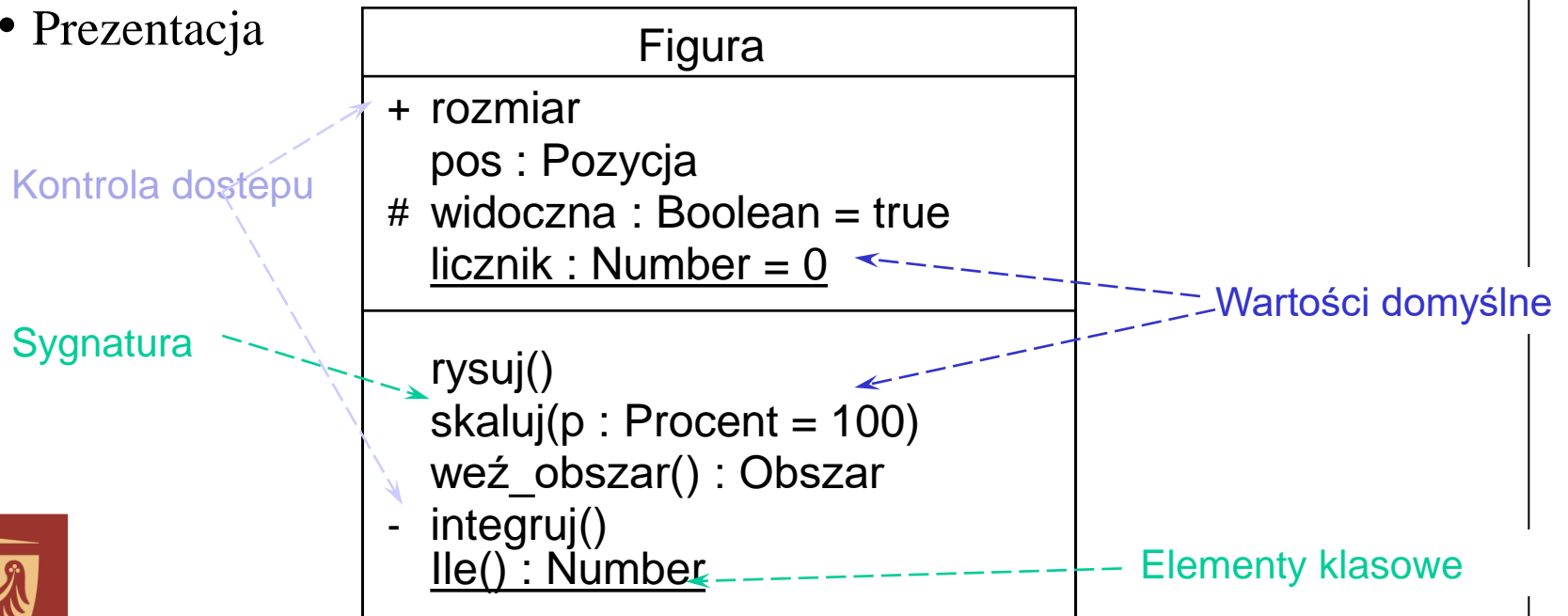
- Służą do obrazowania składników modelowanego systemu i zachodzących pomiędzy nimi powiązań statycznych
- Przedstawiają fizyczną strukturę systemu
- Należą do najczęściej używanych diagramów

Diagramy takie mogą powstawać w różnych sytuacjach projektowych, np. podczas tworzenia słownika pojęć, budowy modelu domenowego modelowania logicznego danych itp.



Klasa - struktura i prezentacja

- Struktura
 - nazwa
 - atrybuty
 - operacje
- Prezentacja



Klasa – Składnia atrybutu

`[widoczność][/] nazwa [krotność] [:typ] [= wartość_domyślna]
[{określenie_właściwości}]`

gdzie:

widoczność: {+, -, #, ~} oznaczające odpowiednio: public, private, protected i package

/ oznacza atrybut pochodny

typ: typ atrybutu, np. klasa, albo typ wbudowany

krotność: liczba egzemplarzy atrybutu, (domyślnie 1) np. [2], [0..1], [*]

UWAGA:

dla atrybutów o krotności większej niż jeden, można dodatkowo podać informacje o uporządkowaniu i unikatowości:

{ *ordered* } atrybuty są posortowane (domyślnie są {*unordered*})

{ *unique* } – atrybuty są unikalne (domyślna właściwość) przeciwieństwo: {*nonunique*}.



Klasa – Składnia atrybutu (cd)

wartość_domyślna : domyślna wartość atrybutu

{określenie_właściwości}: wskazuje wartości właściwości, które mogą być stosowane dla atrybutu:

{id}, **{readOnly}**, {union},
{subsets <property-name>},
{redefines <property-name>},
{ordered}, {bag},
{seq} lub {sequence},
{composite}.



Klasa – Składnia operacji

[widoczność] **nazwa** [(lista_parametrów)] [:typ_wyniku]
[{określenie-właściwości}]

gdzie:

widoczność: {+, -, #} oznaczające odpowiednio: public, private, protected,

typ_wyniku: typ zwracany przez funkcje (lub void); opcjonalny

Składnia deklaracji parametru ma postać:

[tryb] **nazwa** : **typ** [= wartość_początkowa]

Tryb może być postaci:

in - parametr wejściowy; nie może być modyfikowany

out - parametr wyjściowy; może być modyfikowany w celu przekazania informacji wywołującemu

inout - parametr wejściowy; może być modyfikowany

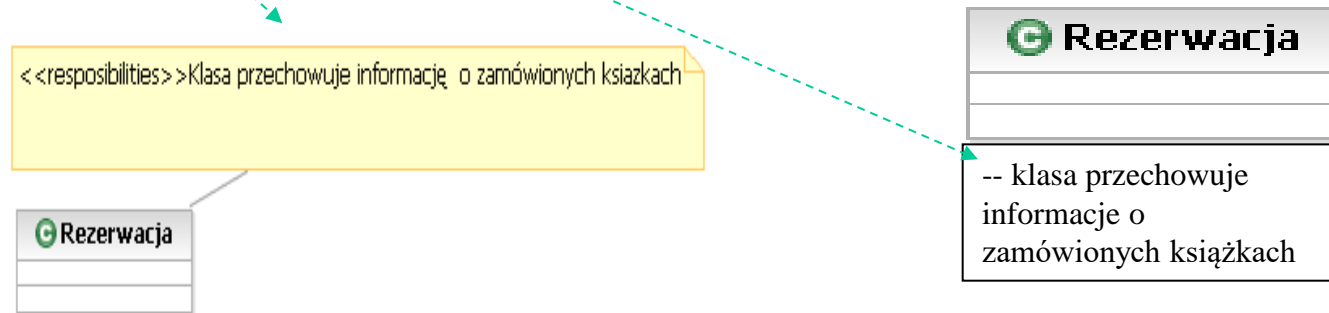
wartość_początkowa : domyślna wartość parametru



Klasa - zobowiązania

Atrybuty i operacje klasy służą do wypełniania przez nią zobowiązań

Zobowiązania można umieszczać albo w dodatkowej (np. czwartej) części prostokąta klasy, albo w dołączonej do niego notatce.



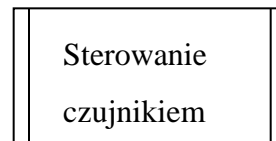
Klasa abstrakcyjna

Klasa abstrakcyjna to klasa, która nie posiada swoich obiektów. Stosuje się ją, by zdefiniować przodka (generalizację) dla innych klas, których obiekty będą wykorzystywane w systemie. Klasę abstrakcyjną wyróżnia się pisząc jej nazwę *kursywą*.

Klasa aktywna

Klasa, której instancje są obiektami aktywnymi tzn. obiektami posiadającymi własny wątek sterowania i mogącymi inicjować sterowanie.

Jest oznaczana jako:



Klasy – zalecane konwencje nazewnnicze

Nazwa klasy:

- pisana z **dużej** litery
- wyśrodkowana w górnej części
- pisana ‘tłustym’ drukiem
- nazwy klas abstrakcyjnych pisane *kursywą*

Atrybuty klasy:

- rzeczowniki pisane z małej litery,
- jeśli nazwa atrybutu składa się z więcej niż jednego słowa, to łączy się je, a każde kolejne jest pisane z dużej litery np.: dataOtrzymania

Operacje klasy:

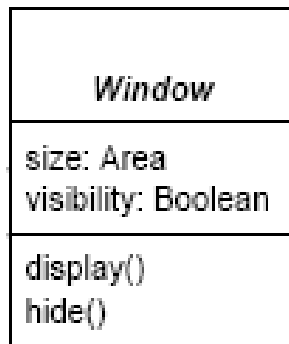
- zazwyczaj wyrażenie czasownikowe,
- pisane z małej litery,
- jeśli nazwa atrybutu składa się z więcej niż jednego słowa, to łączy się je a każde kolejne jest pisane z dużej litery np.: wyslijRezerwacje()



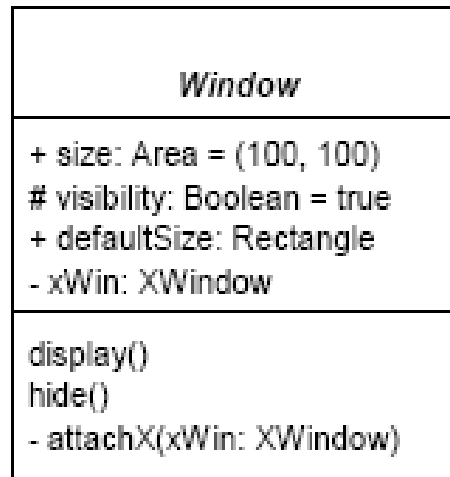
Notacje klasy

różne poziomy specyfikacji

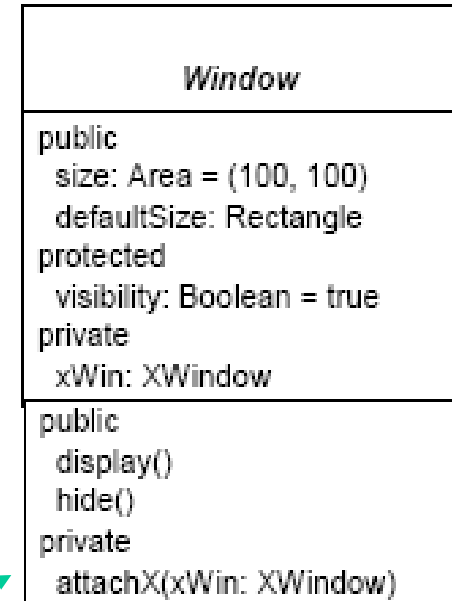
skrócony



analityczny



implementacyjny



Relacje w UML

- Zależność (dependency)
 - semantyczna relacja pomiędzy dwoma elementami podstawowymi, w której zmiana jednego elementu (niezależnego -dostawcy/serwera) może wpływać na semantykę elementu drugiego (zależnego - klienta)
- Asocjacja (association)
 - strukturalna zależność **opisująca zbiór połączeń**;
 - semantyczna zależność pomiędzy dwoma lub więcej klasyfikatorami, która pociąga powiązania pomiędzy instancjami tych klasyfikatorów

Może mieć następujące cechy:

- nazwa
- role
- kierunek nawigacji
- liczebność
- agregacja/kompozycja



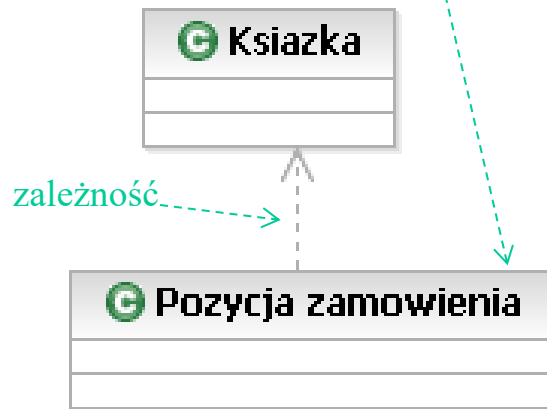
Relacje w UML

- Generalizacja (generalization)
 - związek pomiędzy elementem ogólnym i jego specjalizacją
 - zależność, w której obiekty będące instancjami elementu specjalizacji (dziecko) **mogą zastępować** obiekty będące instancjami elementu ogólnego - generalizacji (rodzica)
 - element specjalizacji może zawierać jedynie dodatkowe własności
- Realizacja (realization)
 - semantyczna relacja pomiędzy dwoma klasyfikatorami, w którym jeden z klasyfikatorów specyfikuje kontrakt gwarantowany przez inny klasyfikator.



Relacja zależności

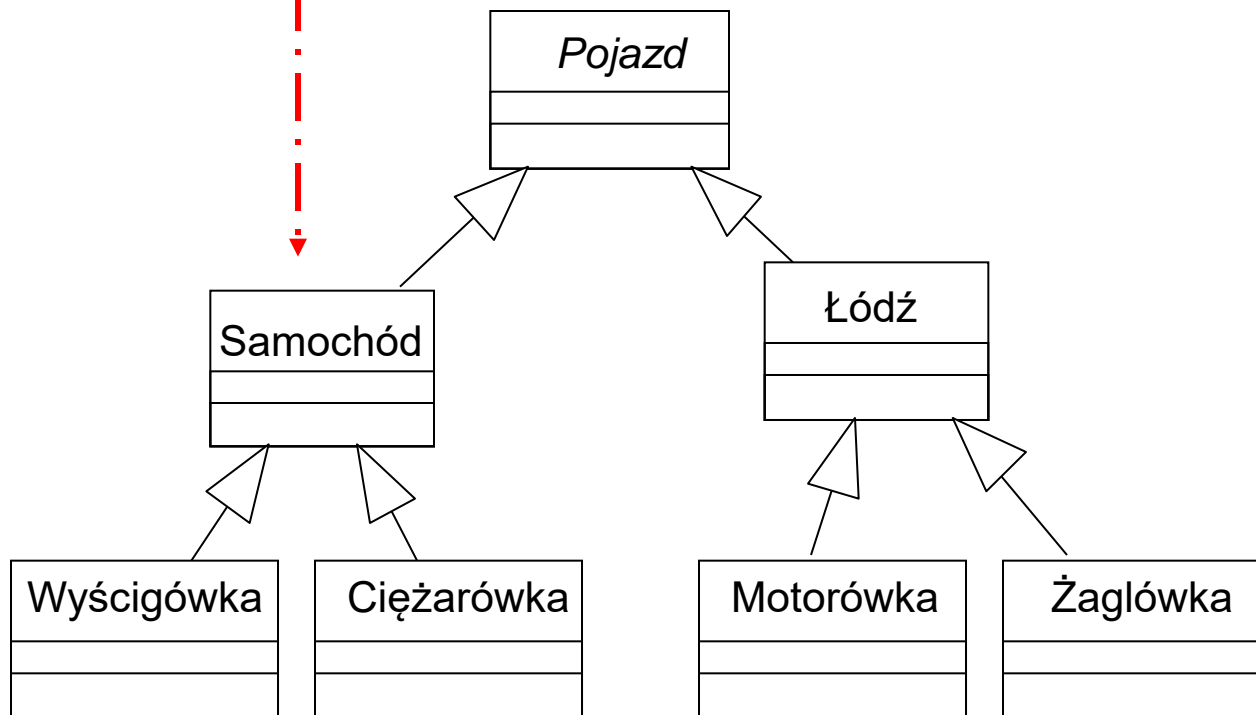
semantyczna relacja pomiędzy **dwoma elementami podstawowymi**, w której zmiana jednego elementu (niezależnego) może wpływać na semantykę elementu drugiego (zależnego)



Relacja generalizacji

Relacja pomiędzy klasami – „dziedziczenie” (inheritance)

- klasa **specjalizacja** (podklasa) dziedziczy wszystkie elementy: atrybuty, operacje, relacje od **generalizacji** (nadklasy)



Relacja generalizacji (cd)

Standardowe ograniczenia na relację generalizacji :

complete disjoint – wszystkie specjalizacje w generalizacji zostały uwzględnione i specjalizacje nie mają wspólnych instancji;

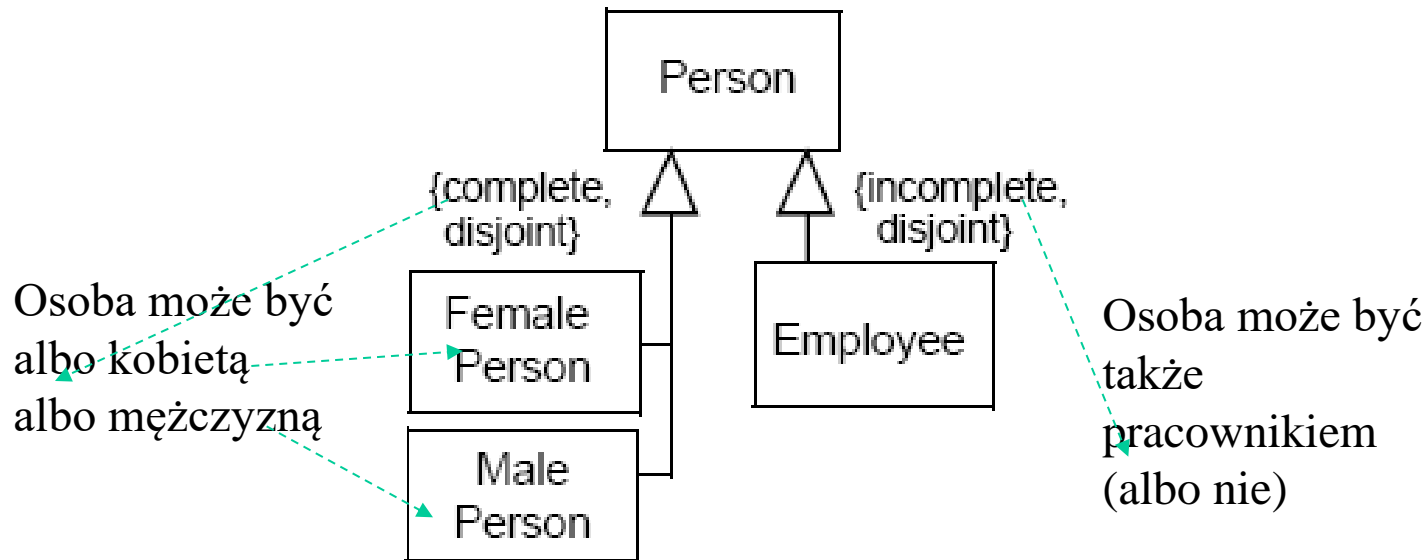
incomplete disjoint – nie wszystkie specjalizacje zostały w generalizacji uwzględnione i specjalizacje nie mają wspólnych instancji;

complete overlapping - wszystkie specjalizacje zostały w generalizacji uwzględnione ale specjalizacje mają wspólne instancje;

incomplete overlapping- nie wszystkie specjalizacje zostały w generalizacji uwzględnione i specjalizacje współdzielą instancje;

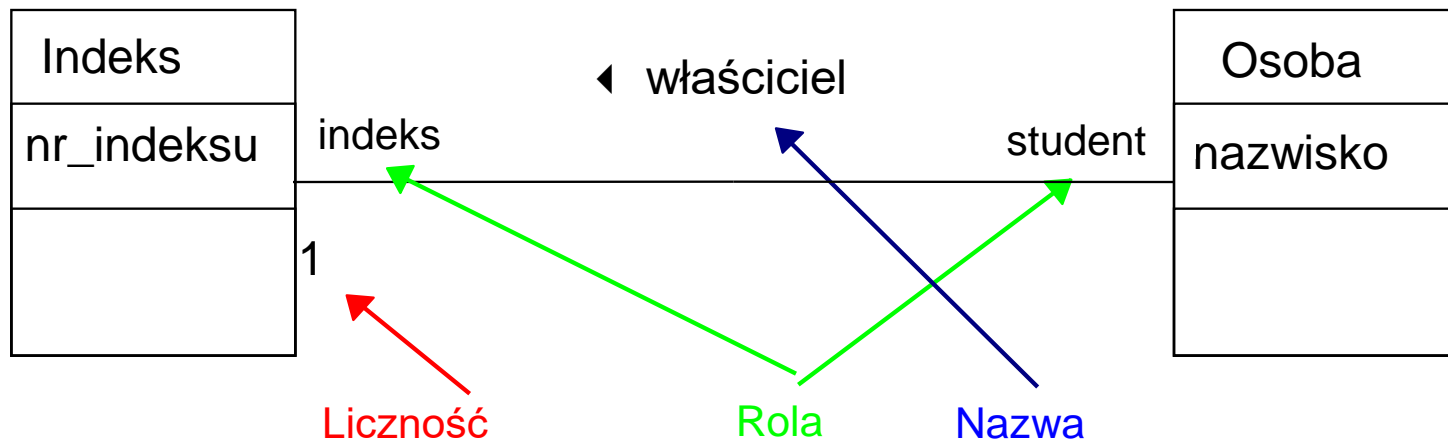


Relacja generalizacji - przykładowe ograniczenia



Relacja asocjacji

- strukturalna zależność opisująca zbiór połączeń;
- semantyczna zależność pomiędzy dwoma lub więcej klasami, która pociąga powiązania pomiędzy obiektami (instancje tych klas)



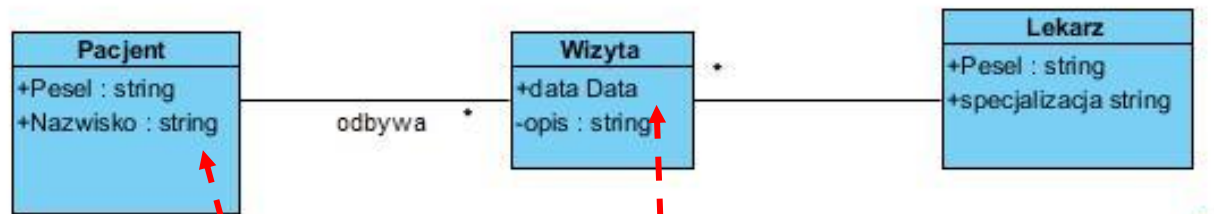
Liczność:

1 ; *

0..1 ; 0..* ; 1..*



Relacja asocjacji - przykład



Powered By Visual Paradigm Community Edition

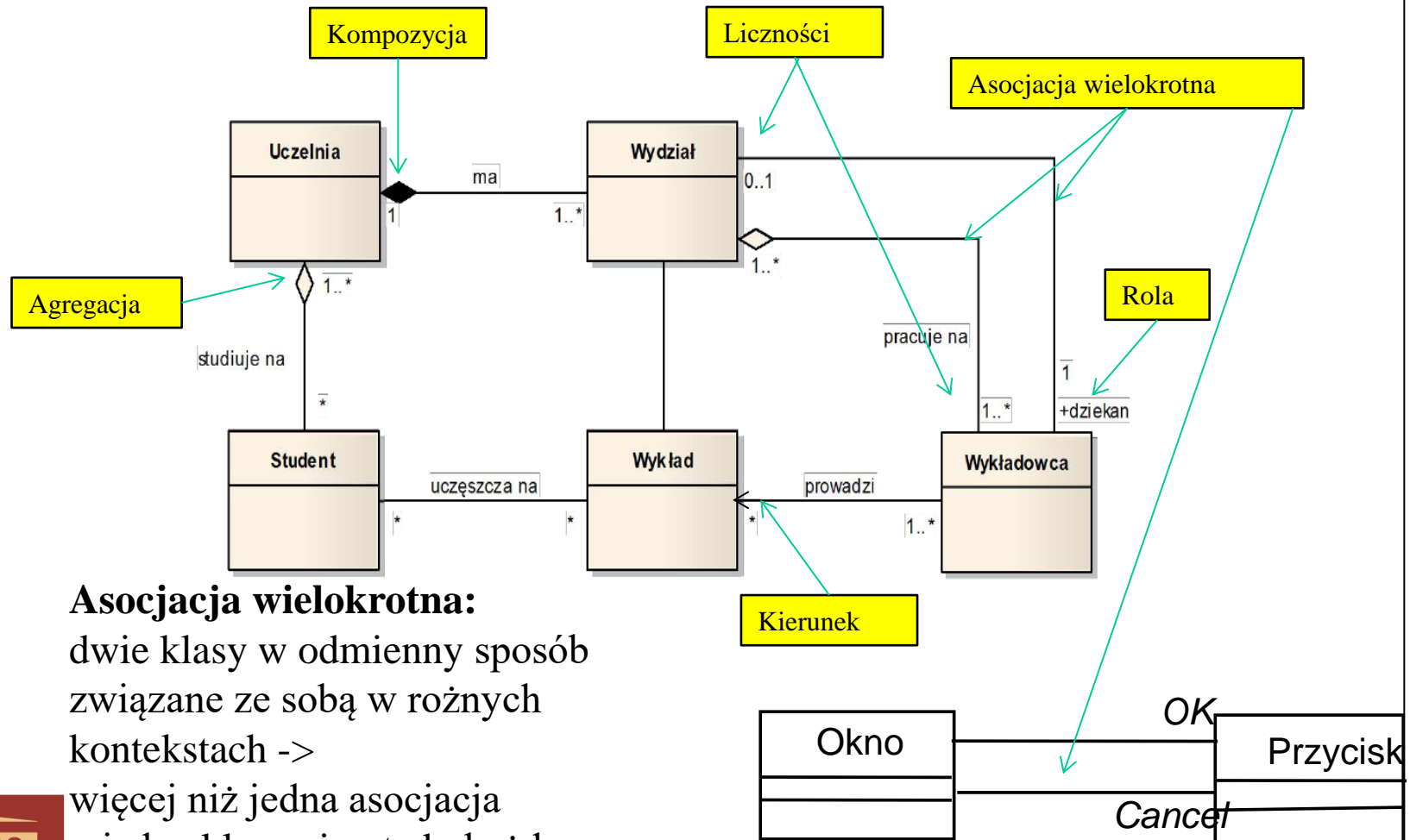
Złożone typy danych - struktury



Typy proste predefiniowane w języku UML



Relacja asocjacji -składowe



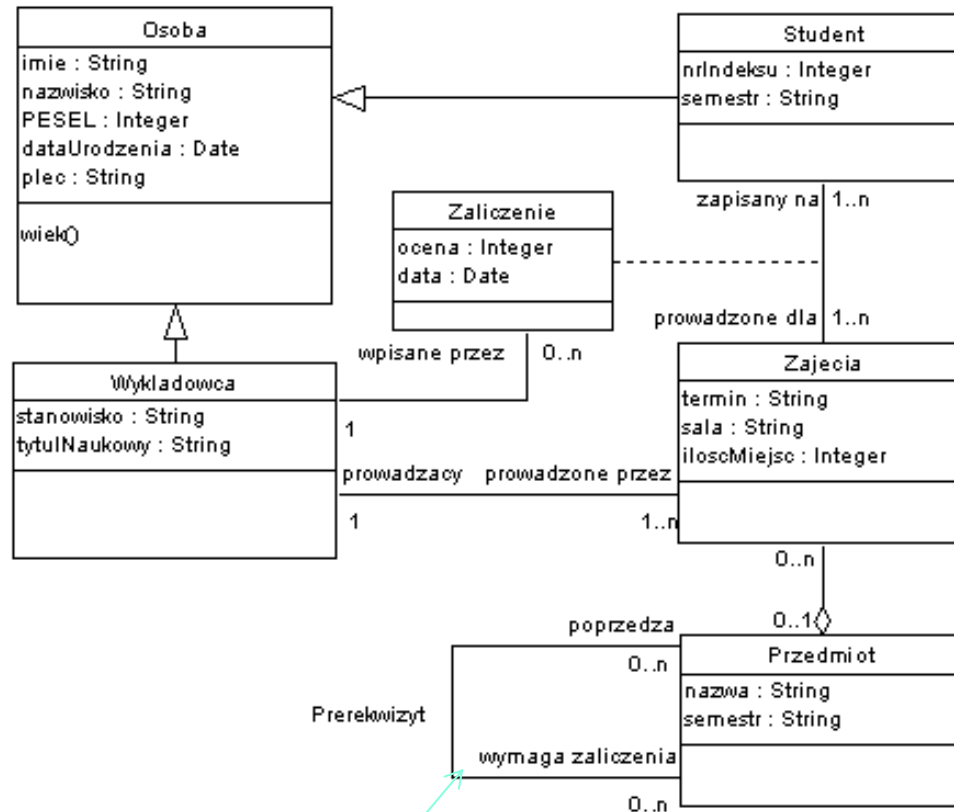
Asocjacja wielokrotna:

dwie klasy w odmienny sposób związane ze sobą w różnych kontekstach ->

więcej niż jedna asocjacja między klasami; wtedy każda powinna być nazwana.



Diagram klas- przykład

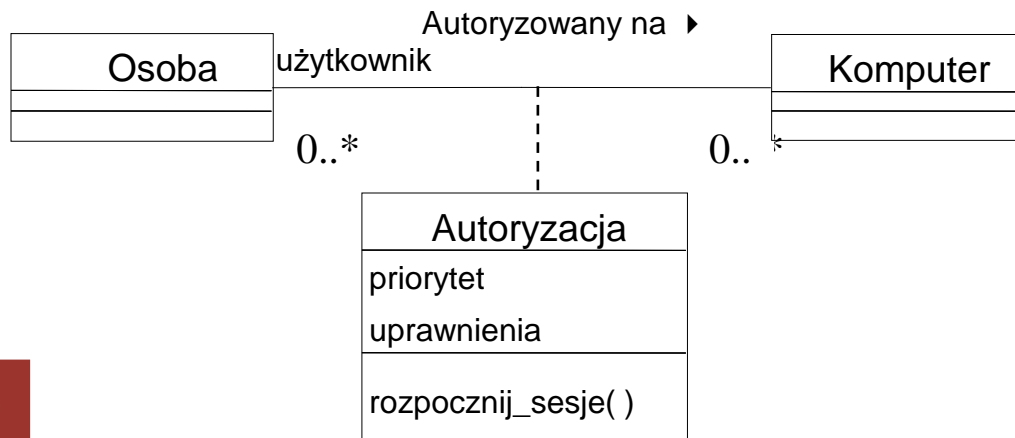
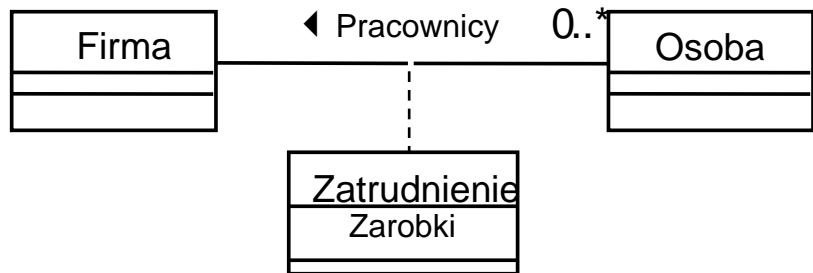


Asocjacja rekurencyjna(zwrotna) –
połączone są obiekty tej samej klasy

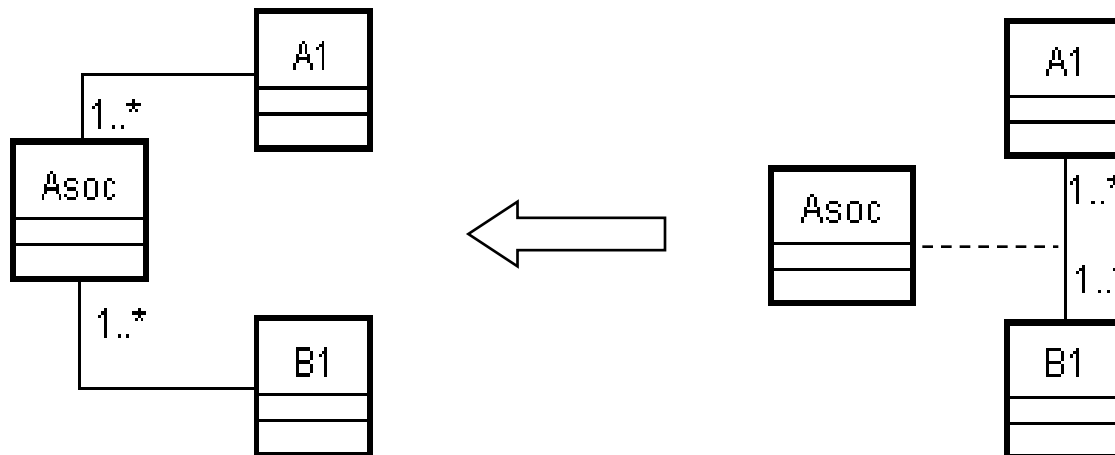


Klasa asocjacji

asocjacja może być reprezentowana przez klasę, aby umożliwić opisanie atrybutów i operacji związanych z asocjacją

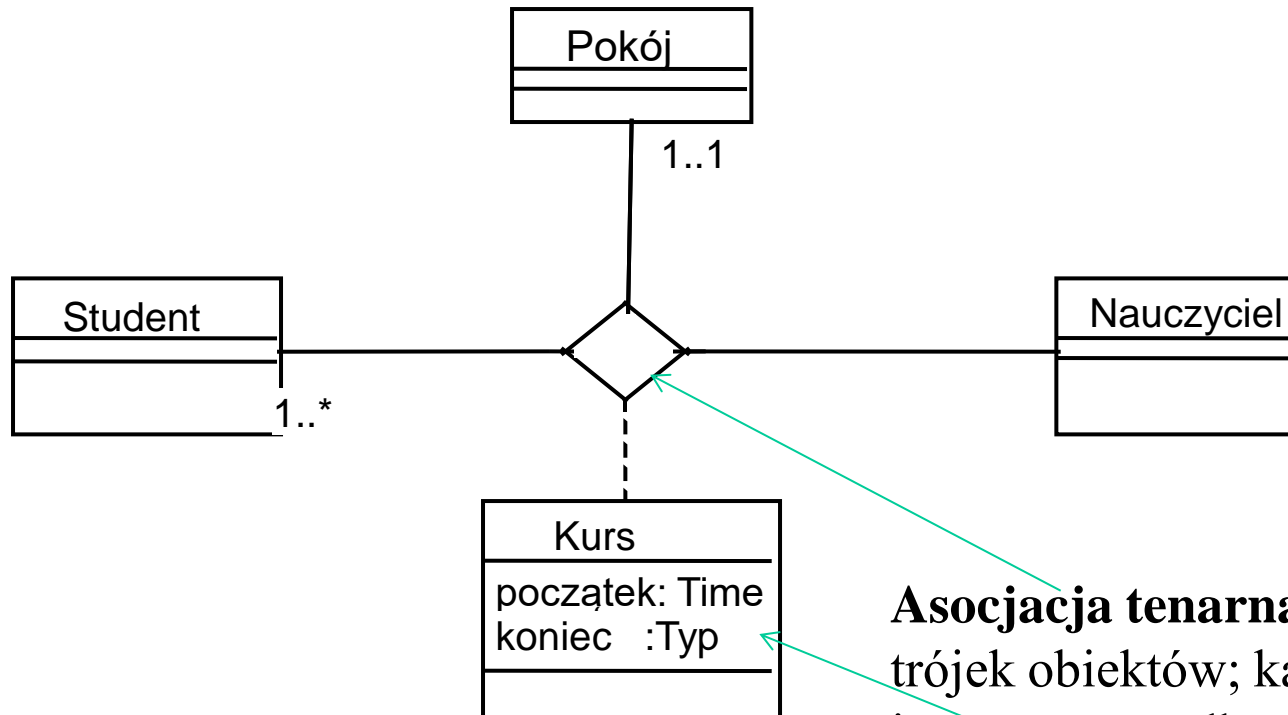


Klasa asocjacji przekształcenie na asocjacje binarne



Asocjacja N-arna

Dana klasa może być w asocjacji z wieloma innymi klasami



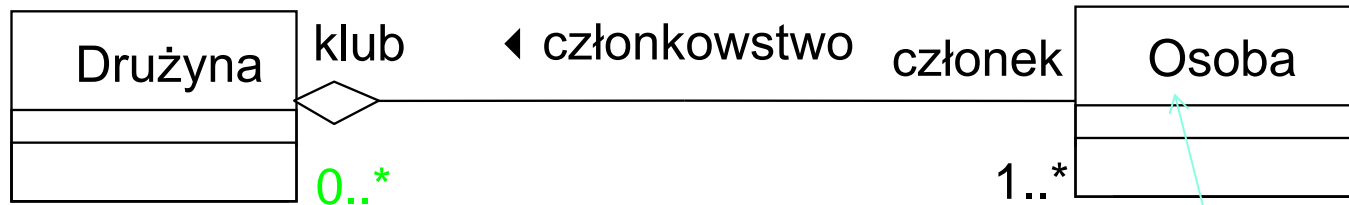
Asocjacja tenarna – zbiór trójek obiektów; każdej trójce jest przyporządkowany pewien zbiór wartości



Relacja agregacji

Agregacja - opisuje relację “całość-część” pomiędzy **instancjami**

ta sama część może być składową różnych agregatów -
jeden obiekt może być też zawierany przez wiele innych
(*analogia: zawieranie wskaźnika - bądź referencji- do
innego obiektu*).

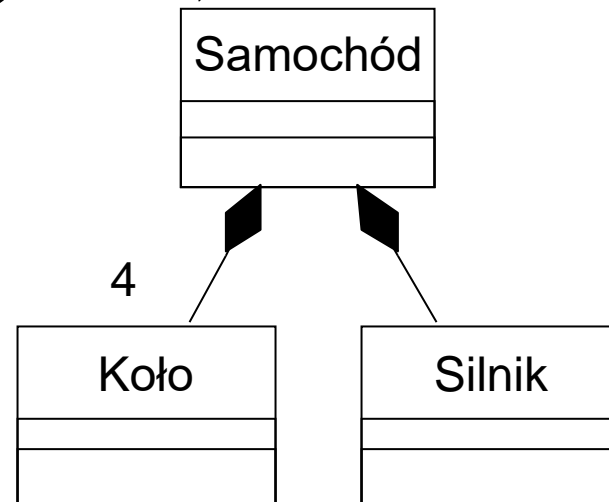


Ta sama osoba
może być
członkiem wielu
drużyn



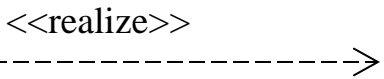
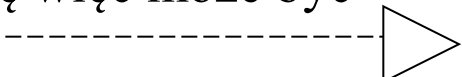
Agregacja silna (kompozycja)

- relacja całkowitej własności;
- jedność czasu życia całości i części: części powstają po utworzeniu całości, a po jej usunięciu są one także usuwane (nie mogą istnieć jeśli symbol określający całość jest usunięty - *analogia: zawieranie obiektu przez inny obiekt*).



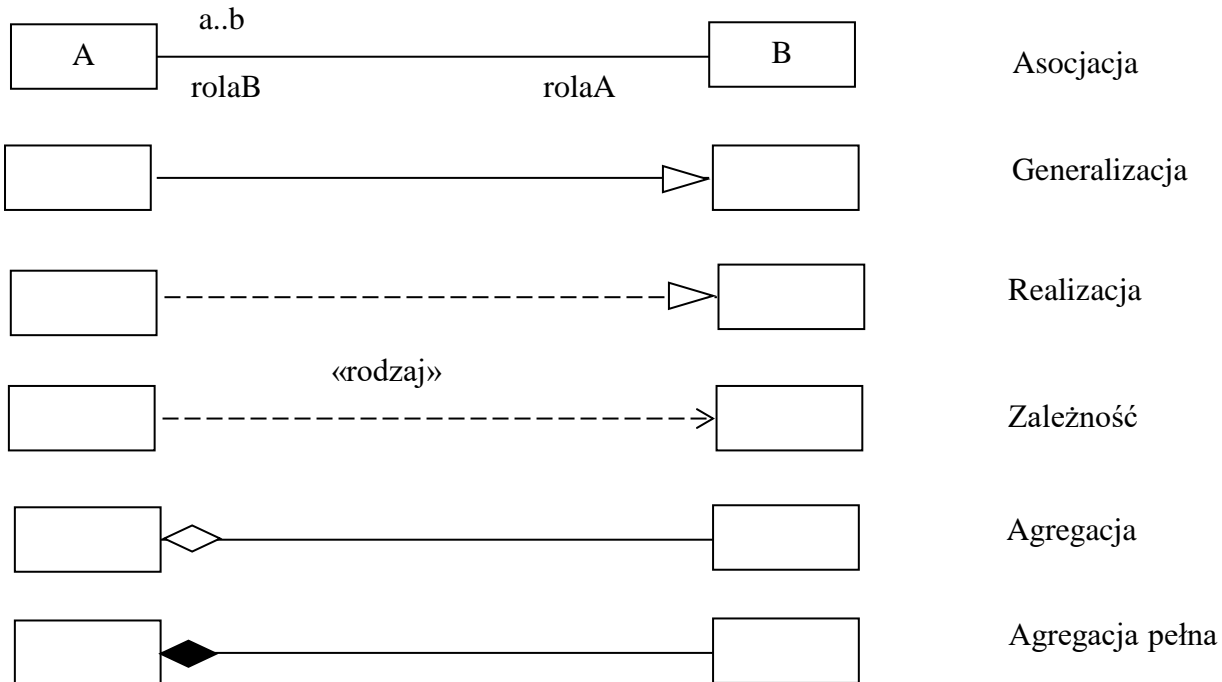
Relacja realizacji

Realizacja – semantyczna relacja pomiędzy dwoma klasami, w którym jedna z klas specyfikuje kontrakt gwarantowany przez drugą klasę ; jest specjalizacją relacji abstrakcji.

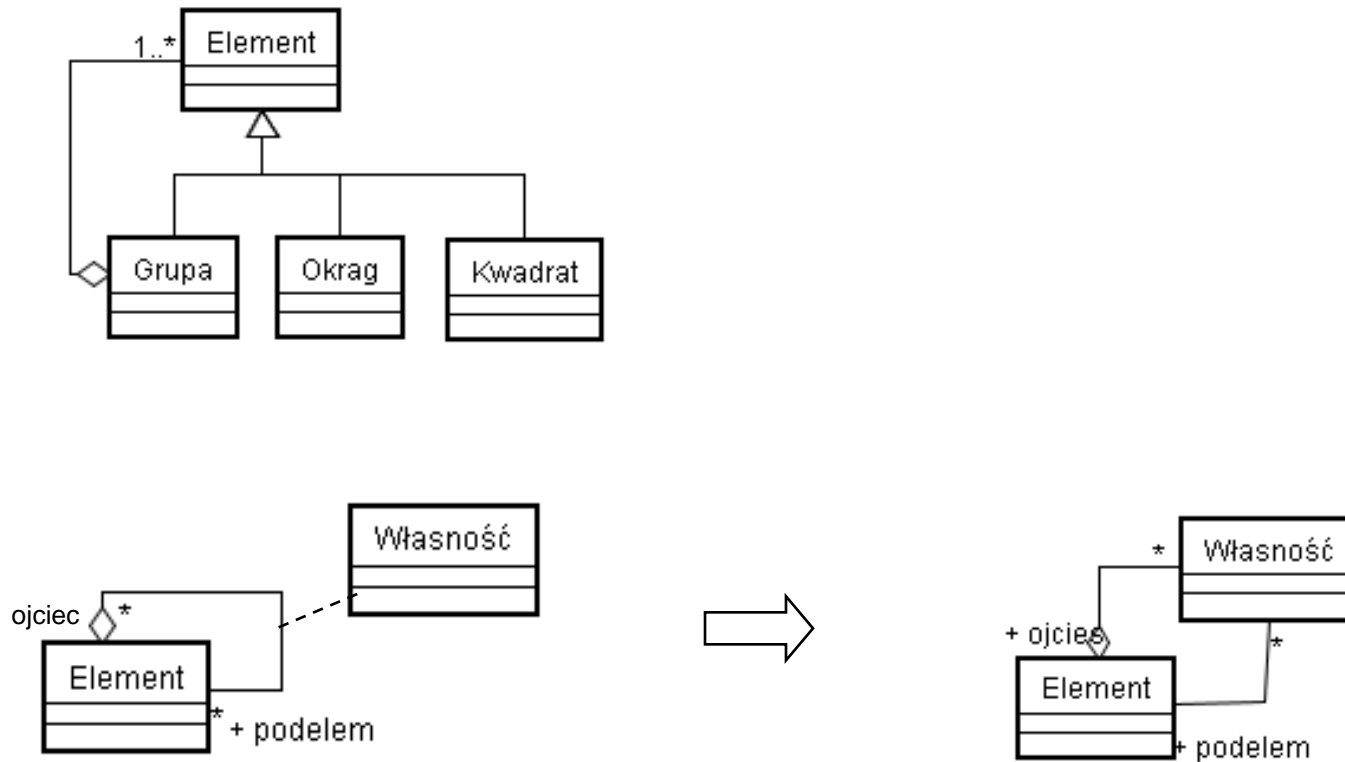
Uważana za pewnego rodzaju relację zależności, 
z innej zaś strony przypomina generalizację więc może być
prezentowana jako: 



Relacje - zestawienie symboli graficznych



Wzorce strukturalne – przykłady asocjacji rekurencyjnej



Interfejs

deklaracja podzbioru operacji, które łącznie definiują pewien zestaw usług oferowanych przez instancję (klasy, przypadku użycia, komponentu)

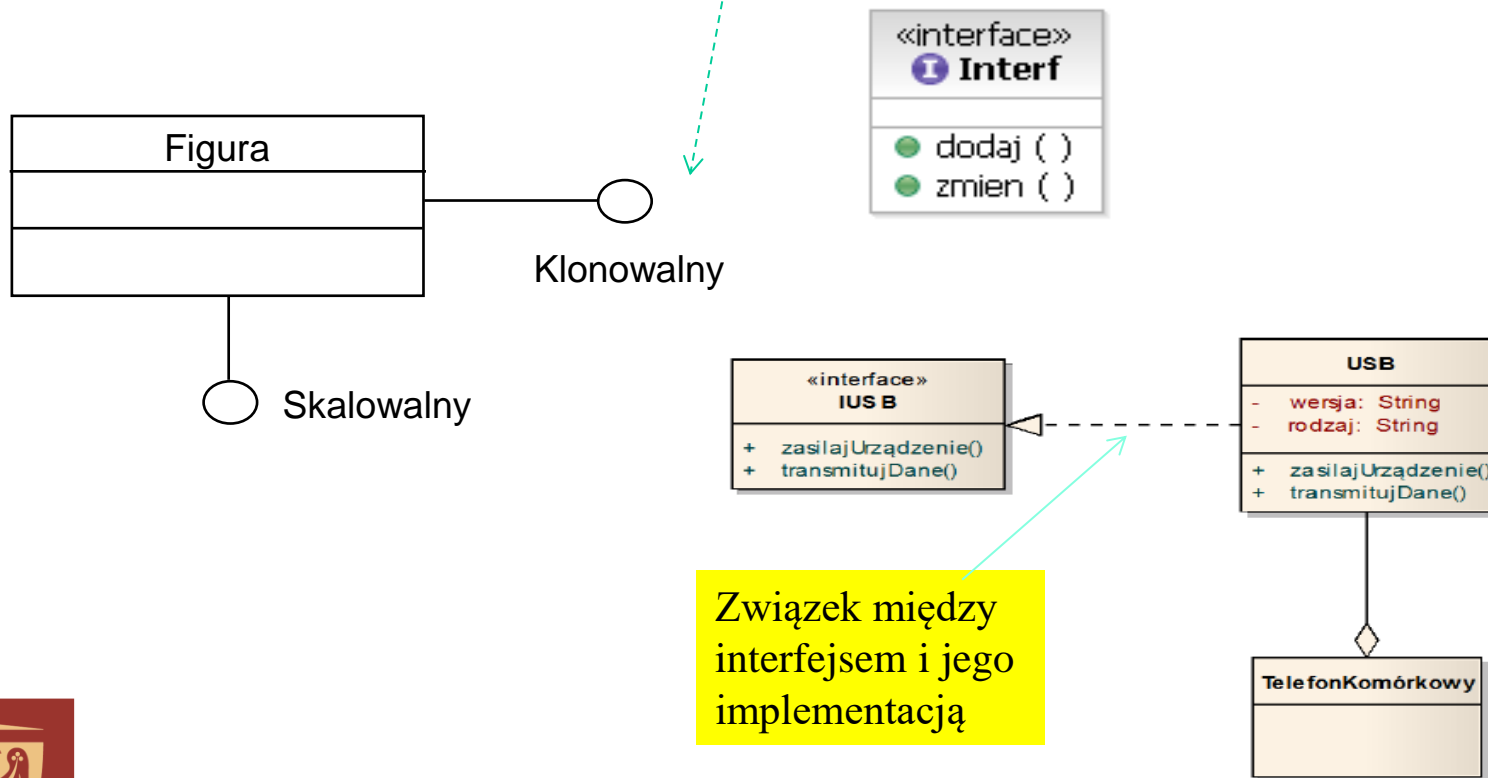
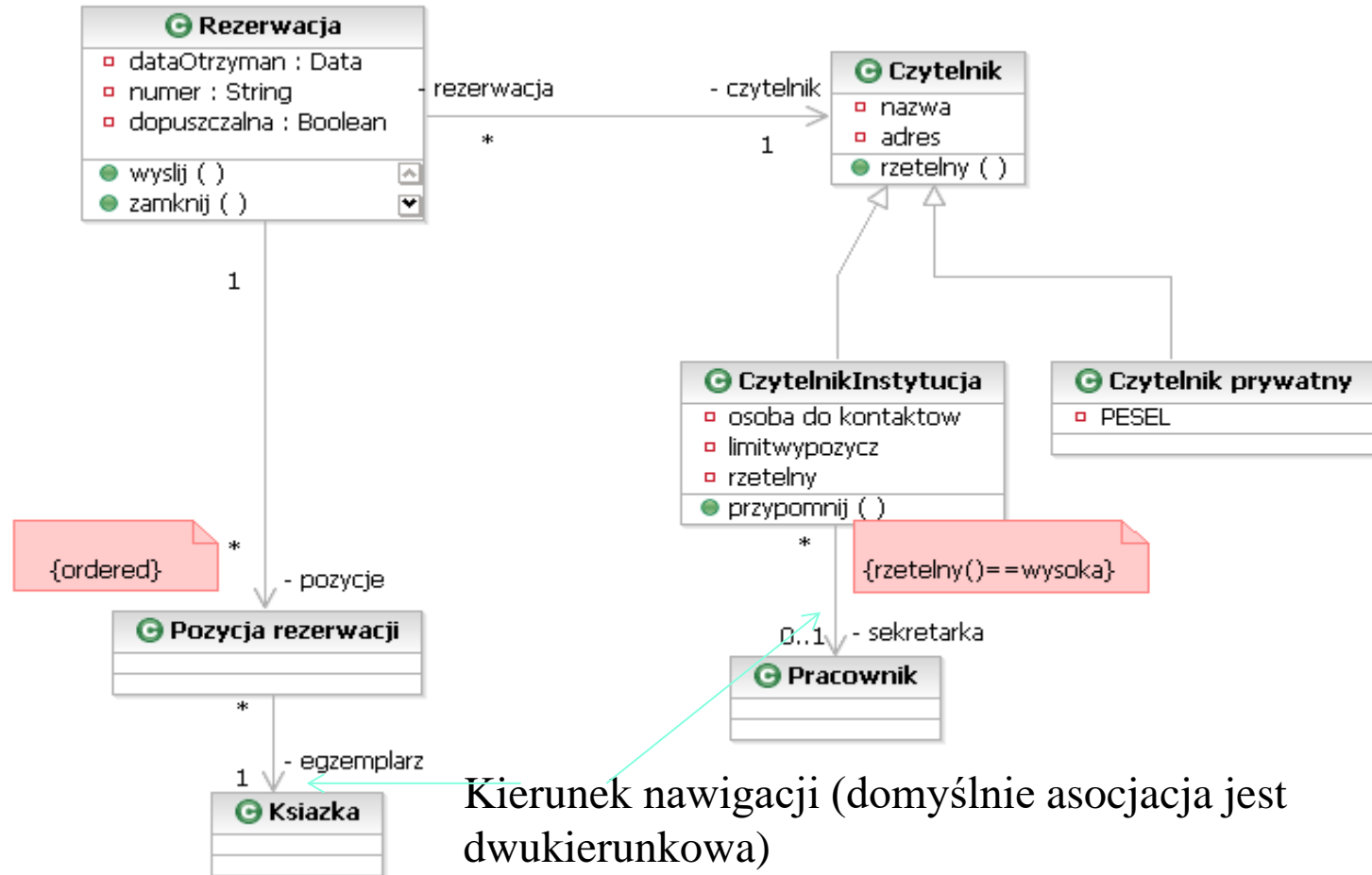


Diagram klas -przykład



Kierunek nawigacji (domyślnie asocjacja jest dwukierunkowa)

-jednokierunkowa; dodaje się strzałkę (oznacza to że **komunikacja jest jednokierunkowa**)



Diagram klas -przykład

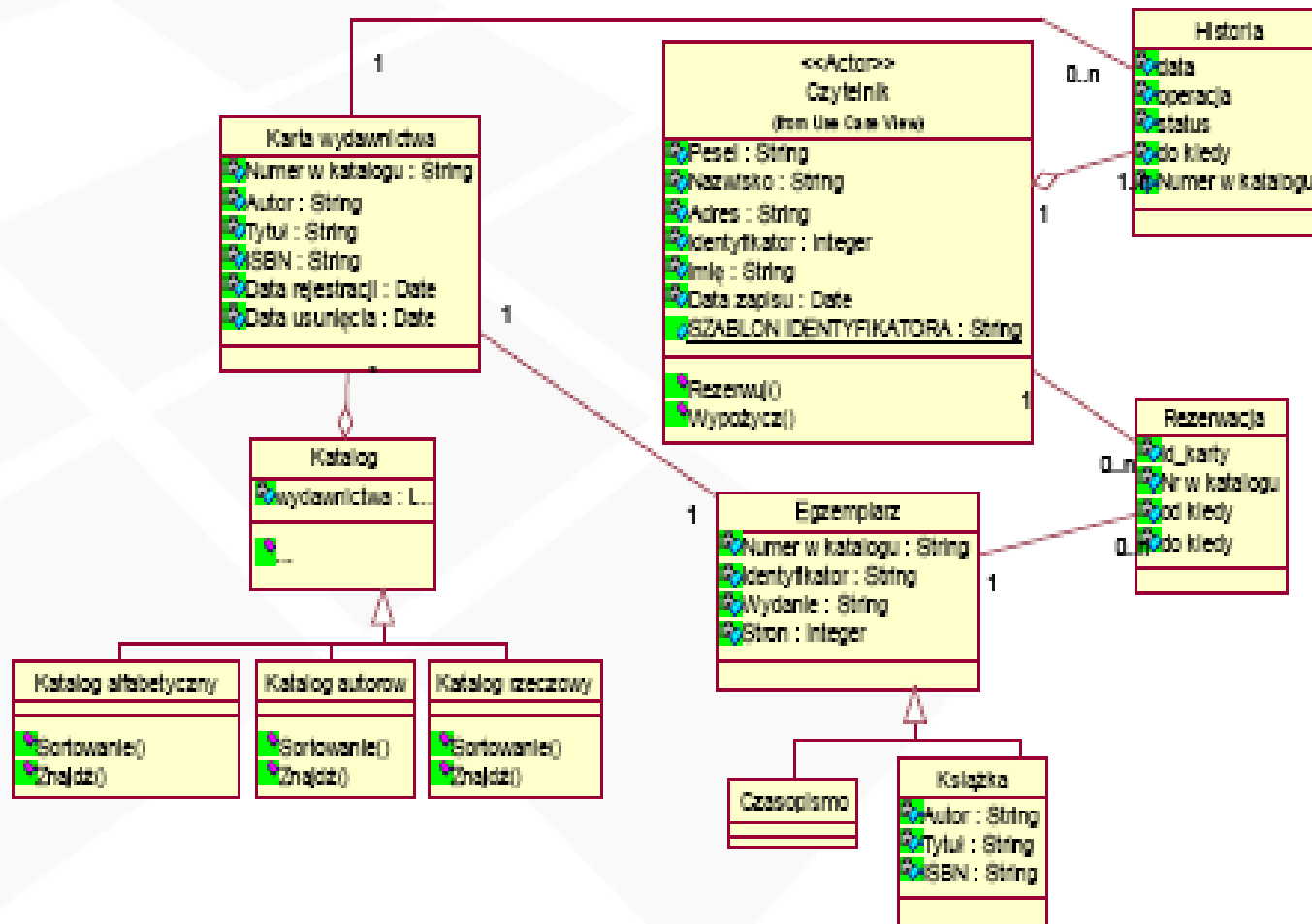
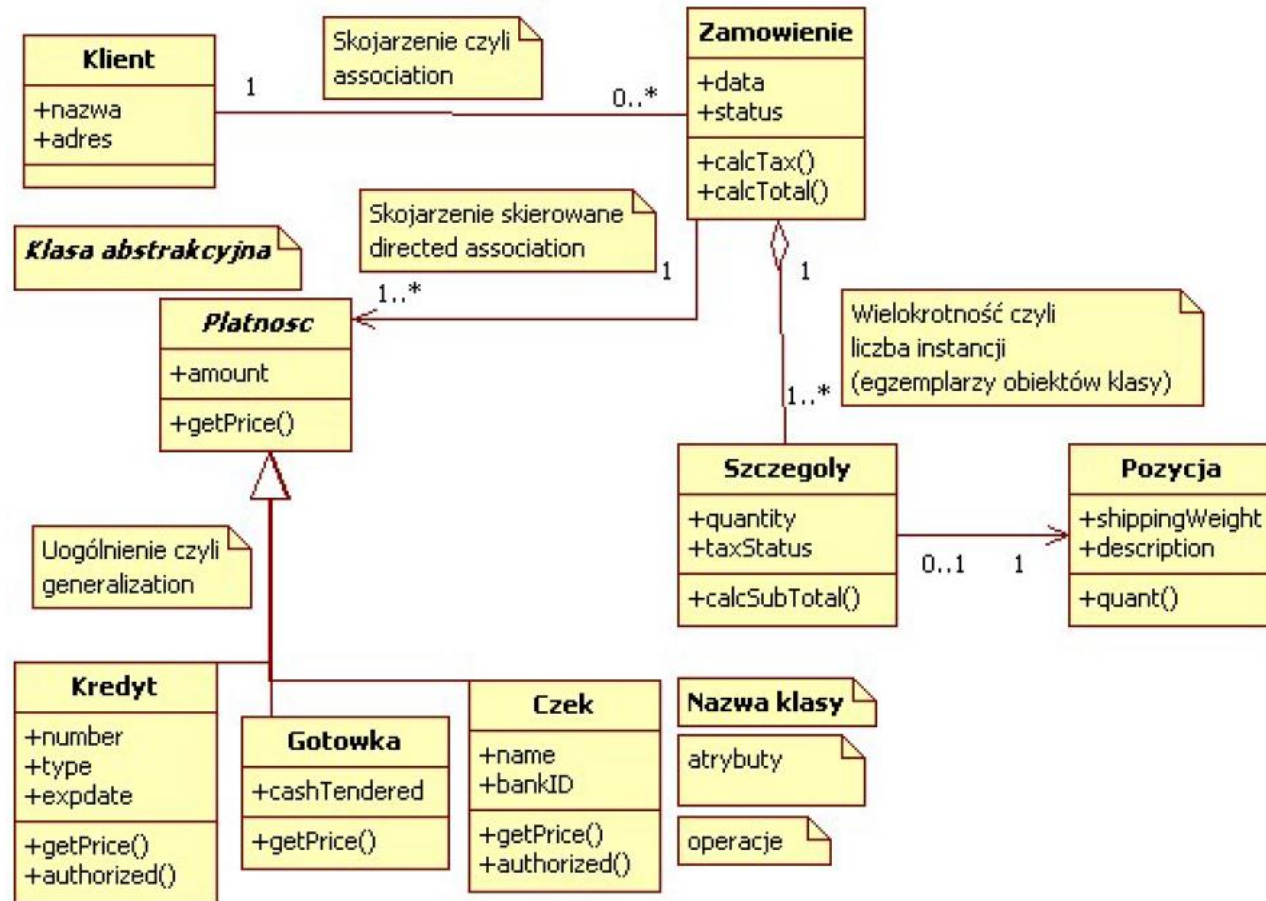
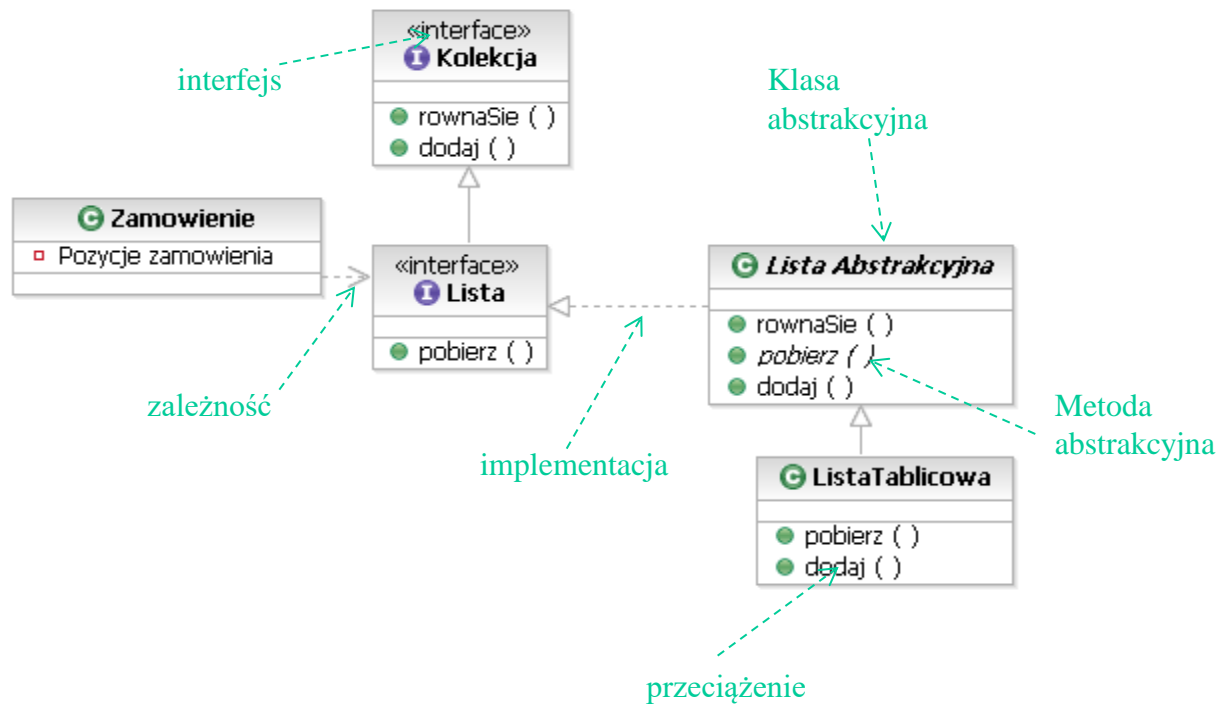


Diagram klas -przykład



Interfejs – przykład zastosowania



private Lista pozycjeZamowienia =

new ListaTablicowa();

[Fowler, 2005]



Diagram klas - zalecenia

- Diagram powinien mieć nazwę określającą jego przeznaczenie
- Umieszczać na diagramie **tylko szczegóły istotne dla danego aspektu systemu**
- Zachować równowagę pomiędzy diagramami obrazującymi statyczne i dynamiczne właściwości systemu
- **Unikać zbyt dużych diagramów** (do 15 elementów maksymalnie!)
- **Nie przesadzać z ilością związków** na jednym diagramie
- Elementy diagramu powinny być ułożone tak, aby **zminimalizować liczbę przecinających się linii**
- Elementy zbliżone znaczeniowo powinny znajdować się z pobliżu siebie
- **Można (a nawet należy) używać kolorów i notatek**

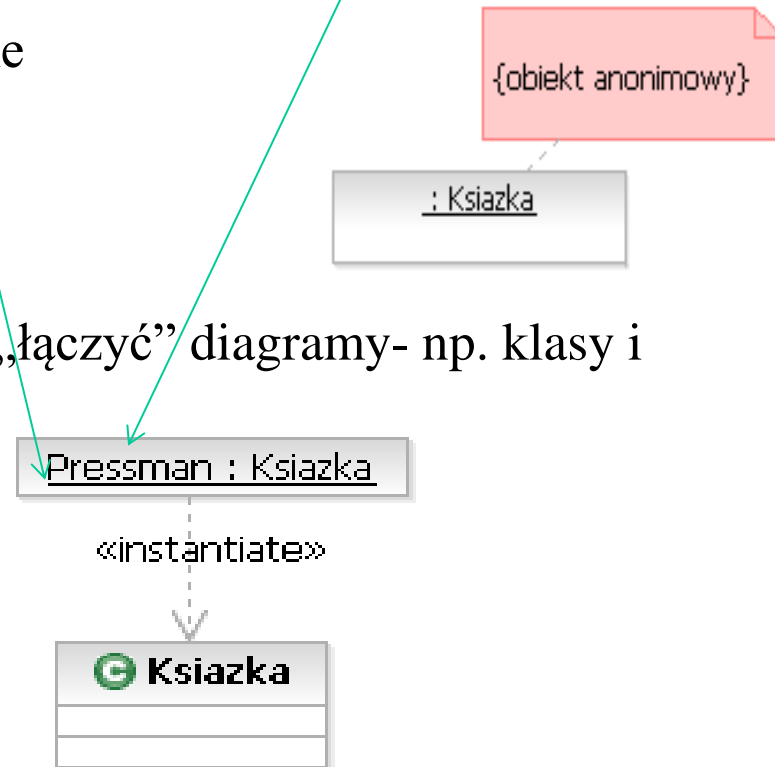


Obiekt – egzemplarz klasy

Każdy egzemplarz klasy- (obiekt) może mieć własną nazwę, ale często na diagramach używa się **obiektów anonimowych**

Nazwy obiektów są podkreślone

Jeśli zachodzi potrzeba można „łączyć” diagramy- np. klasy i obiekty na jednym diagramie:



Opis systemu rzeczywistego – diagram obiektów

- Instancja modelu statycznego
- Elementy
 - obiekty
 - połączenia (links)

Diagram Obiektów pokazuje zbiór obiektów i ich związków w ustalonej chwili. Podobnie jak inne diagramy może zawierać ograniczenia, notatki, pakiety/podsystemy.

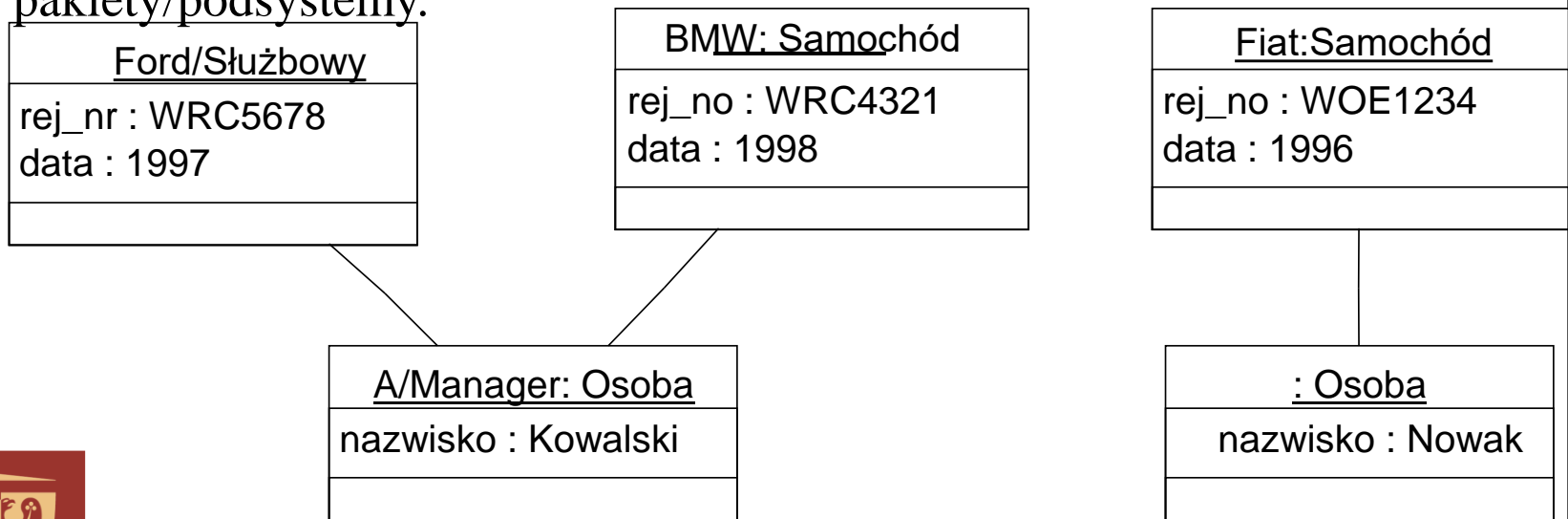
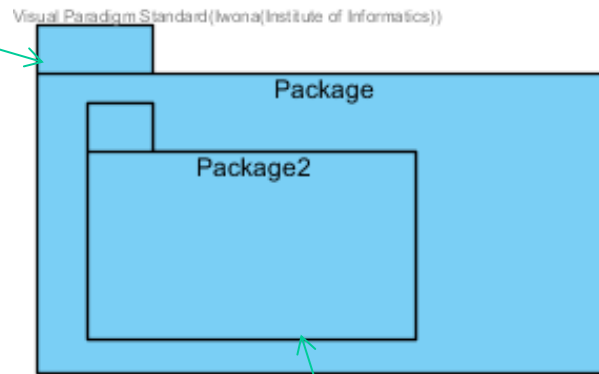


Diagram pakietów - charakterystyka

Pakiet - podstawowy element organizacyjny modelu systemu w UML.



Cały system to pakiet zawierający wszystkie inne pakiety, diagramy i elementy.

Jeden pakiet może zawierać podrzędne pakiety, diagramy lub pojedyncze elementy.



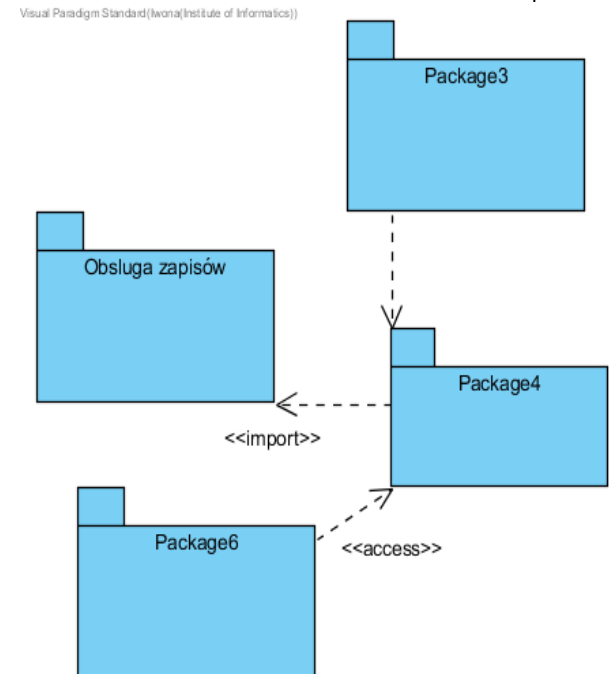
Diagram pakietów - charakterystyka

Pakiety mogą być łączone przez trzy zależności (opisane przez stereotypy):

1. **<<import>>** - oznacza, że elementy pakietu wskazywanego mogą być bezpośrednio użyte w pakiecie wskazującym

2. **<<merge>>** - oznacza, że pomiędzy elementami pakietów, które mają tę samą nazwę, zachodzi - generalizacja (dziedziczenie)

3. **<<access>>** - analogicznie jak <<import>>, ale nazwy należy poprzedzać nazwą pakietu



Modelowanie statyczne - proces

Przedstawione:

- pojęcia: klasa, obiekt, relacja
- diagramy klas
- typy relacji

stosuje się podczas **procesu modelowania struktury** systemu składającego się z kroków:

1. identyfikacja obiektów w systemie i jego otoczeniu,
2. opisanie ich struktury,
3. dokonanie klasyfikacji obiektów → klasy,
4. opisanie struktury powiązań między klasami.



Modelowanie statyczne - podsumowanie

Modelowanie statyczne:

- opisuje elementy i strukturę wnętrza modelowanego systemu (zawiera informacje o statycznych związkach między elementami – klasami)
- wykorzystuje **diagramy klas, obiektów, komponentów i rozmieszczenia**
- ściśle powiązane z technikami programowania zorientowanego obiektowo

