MODELOWANIE OBIEKTOWE Z *UML*

Maciej Patan

Instytut Sterowania i Systemów Informatycznych
Uniwersytet Zielonogórski

Paradygmat obiektowy

- system – zbiór unikatowych obiektów („społeczność obiektów"),
- obiekt w czasie swego „cyklu życia":
  - ▷ jest nośnikiem informacji (atrybuty=dane),
  - ▷ może wykonać określone czynności (metody=przetwarzanie),
  - ▷ może komunikować się z innymi obiektami,
- odzwierciedlenie struktury obiektów i relacji zachodzących w świecie realnym.

Podstawowe koncepcje obiektowości

- abstrakcja – odfiltrowanie atrybutów i operacji nieistotnych,
- enkapsulacja – ukrycie nadmiernego poziomu szczegółowości,
- dziedziczenie – generalizacja
  - ▷ relacja hierarchiczna,
  - ▷ oszczędność nakładów modelowania,
- polimorfizm – wielość form operacji dla dziedziczonych klas
  - ▷ wirtualny mechanizm wywoływania funkcji,
  - ▷ naturalny system wyrażania czynności,
  - ▷ zmniejszenie nakładów programowania,

- komunikacja,
  - synchronizacja zdarzeń,
  - wymiana danych,
  - współpraca między obiektami,
- asocjacja (powiązanie) – relacja wiążąca klasy (obiekty),
- agregacja – powiązanie wielu komponentów w jedną całość,
  - agregacja całkowita – komponenty składowe istnieją tylko jako części całości.

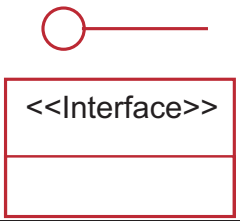
Modelowanie strukturalne

Model strukturalny: Opis systemu, podkreślający strukturę obiektów, włączając w to ich przynależność do klas, wzajemne powiązania, atrybuty i operacje.

Diagramy strukturalne

- Pokazują statyczną strukturę modelu:
  - jednostki istniejące (np. klasy, interfejsy, pakiety, komponenty, węzły),
  - wewnętrzną strukturę jednostek,
  - wzajemne związki między jednostkami,
- Nie pokazują informacji o dynamicznym stanie systemu
- Typy diagramów
  - Statyczne diagramy strukturalne,
    - diagramy klas (widok przynależności klasowych),
    - diagramy obiektów (widok systemu w danej chwili),
  - Diagramy implementacyjne,
    - diagramy komponentów,
    - diagramy wdrożenia,

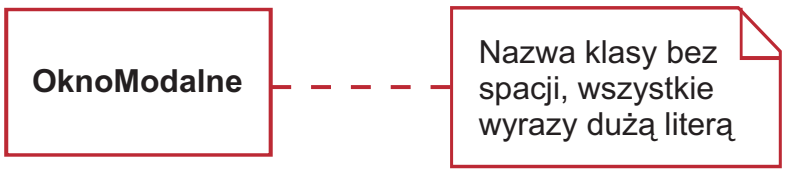
Podstawowe elementy statycznych diagramów strukturalnych

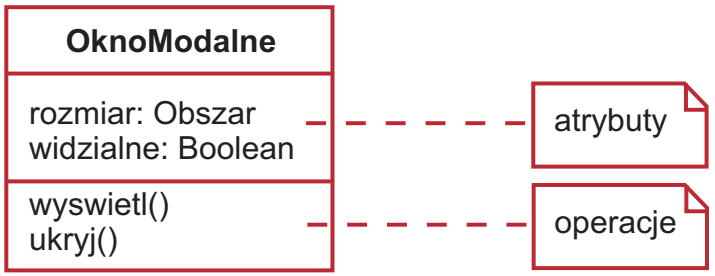
Konstrukt	Krótki opis	Składnia
klasa	opis zbioru obiektów, które dzielą te same atrybuty, operacje, metody, związki i semantykę	
interfejs	nazwany zbiór operacji charakteryzujący zachowanie się elementu	
pakiet	element grupujący inne elementy	
ograniczenie	warunek semantyczny lub restrykcja	

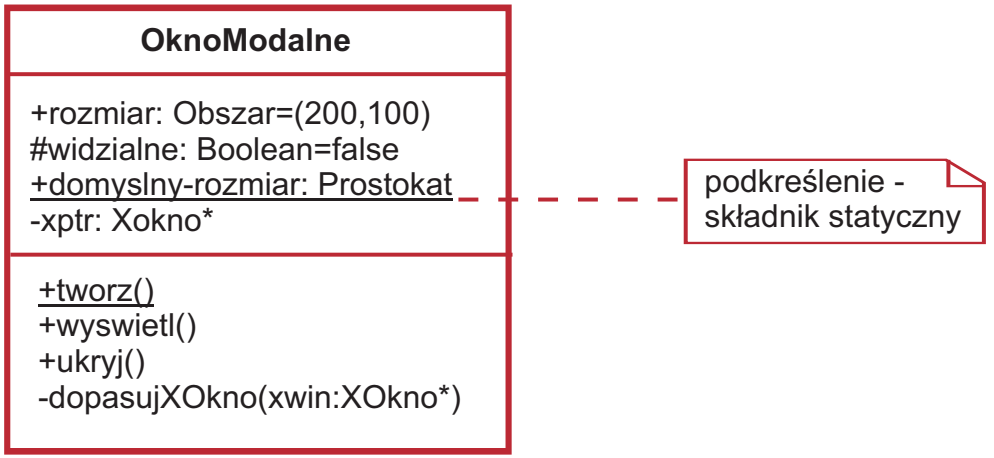
Podstawowe relacje statycznych diagramów strukturalnych

Konstrukt	Krótki opis	Składnia
asocjacja	związek pomiędzy klasami wymagający połączenia się ich instancji (obiektów)	
agregacja	specjalny przypadek asocjacji odpowiedzialny za relację zawierania (posiadania)	
generalizacja	systematyczna relacja między elementem szczegółowym a jego ogólniejszą postacią	
zależność	związek, w którym zmiana w jednym elemencie modelu (niezależnym) pociąga za sobą zmiany w drugim elemencie (zależnym)	
realizacja	zależność pomiędzy specyfikacją elementu i jego implementacją	

Wizualizacja klas







Składnia atrybutu

```
dostęp nazwa [liczność porządek] : typ = wartość_początkowa
```

- `dostęp` (+ public,− private,# protected),
- `liczność` (opcjonalna) – zakres wartości, które przyjmuje atrybut w postaci `gr_dolna..gr_gorna` (* oznacza zakres nieograniczony),
- `porządek` (opcjonalny) – (ordered, unordered),
- `typ` (opcjonalny) – Boolean, Integer, Real, String lub inna istniejąca klasa,
- `wartość_początkowa` (opcjonalna) – wartość inicjalizacyjna atrybutu.

Przykład: `+naturalna [0..* ordered]: Integer = 1`
– „naturalna” jest publicznym atrybutem przyjmującym nieograniczone i uporządkowane wartości całkowite w zakresie od 0 do ∞, a jego wartość początkowa jest równa 1.

Składnia operacji

```
dostęp nazwa_oper (lista_parametrów) : typ_zwracany
```

- `dostęp` (+ public, − private, # protected),
- `lista_parametrów` (opcjonalna) – każdy parametr ma postać

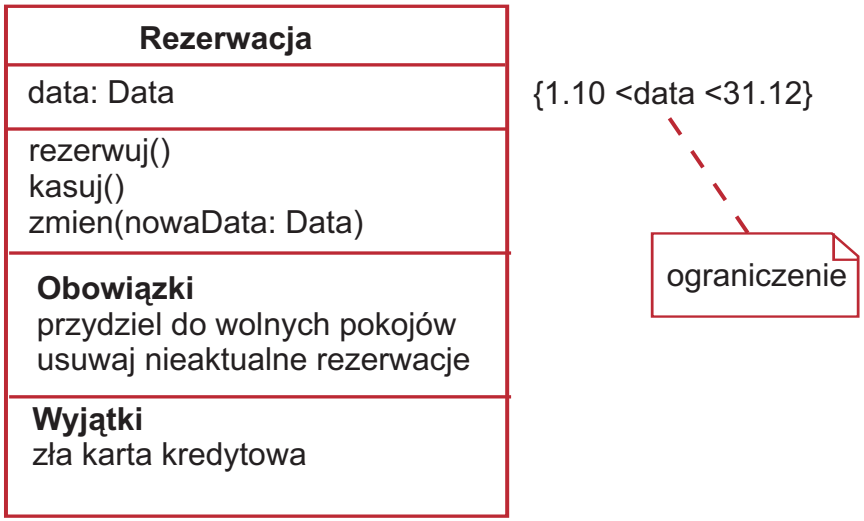
  `rodzaj nazwa : typ = wartość_domyślna`

  gdzie `rodzaj` $\in$ {in, out, inout} , `typ` i `wartość_domyślna` jak dla atrybutu,
- `typ_zwracany` (opcjonalny) – jw.

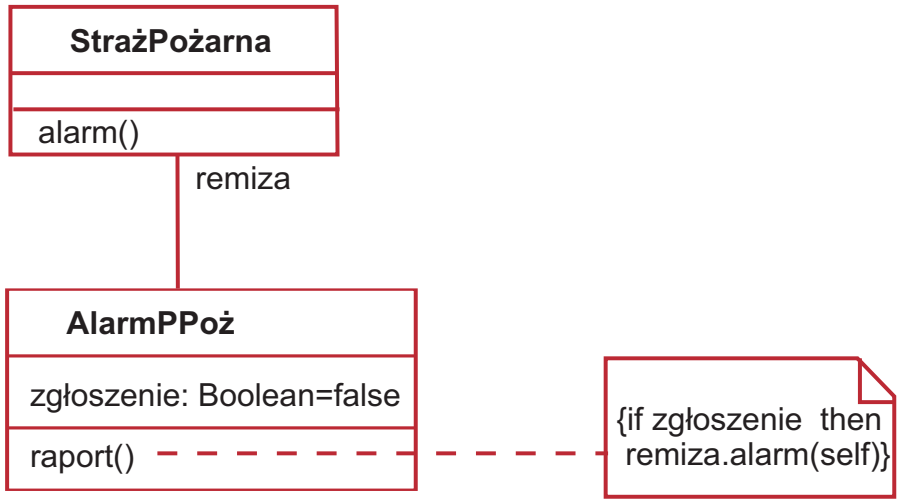
Przykład: `-dodaj(inout A: Macierz, inout B: Macierz): Macierz`
– „dodaj" jest prywatną operacją przyjmującą jako argumenty dwa atrybuty wejściowo-wyjściowe typu `Macierz` bez wartości domyślnych, a zwraca także element typu `Macierz`.

Wizualizacja klas (2)

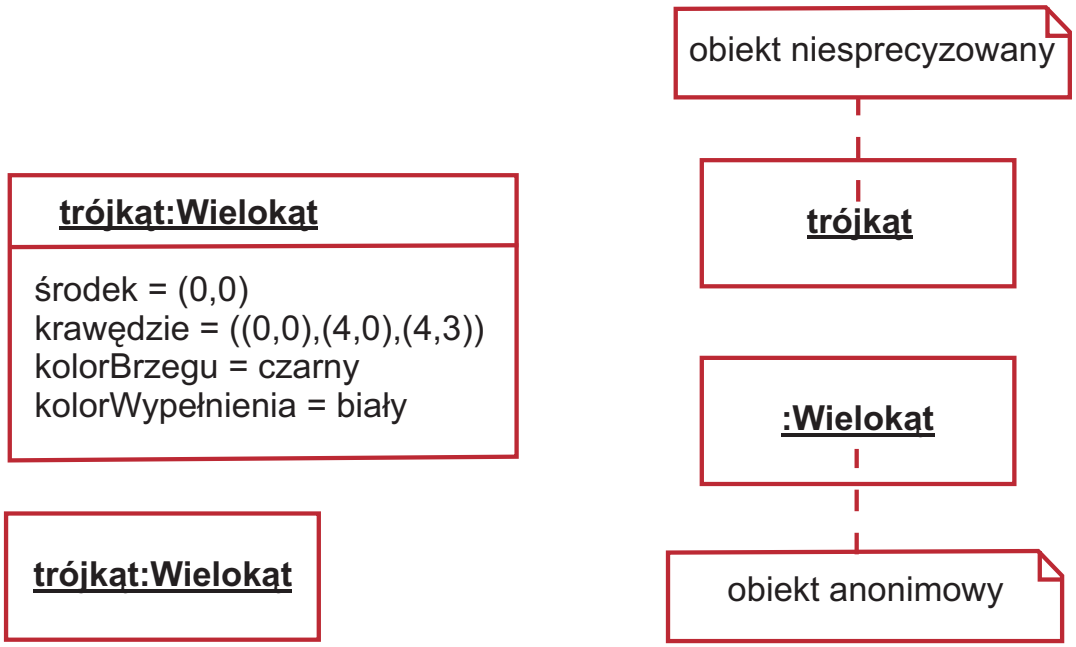
Dodatkowe dane



Ciała funkcji



Wizualizacja obiektów



Wartości atrybutów

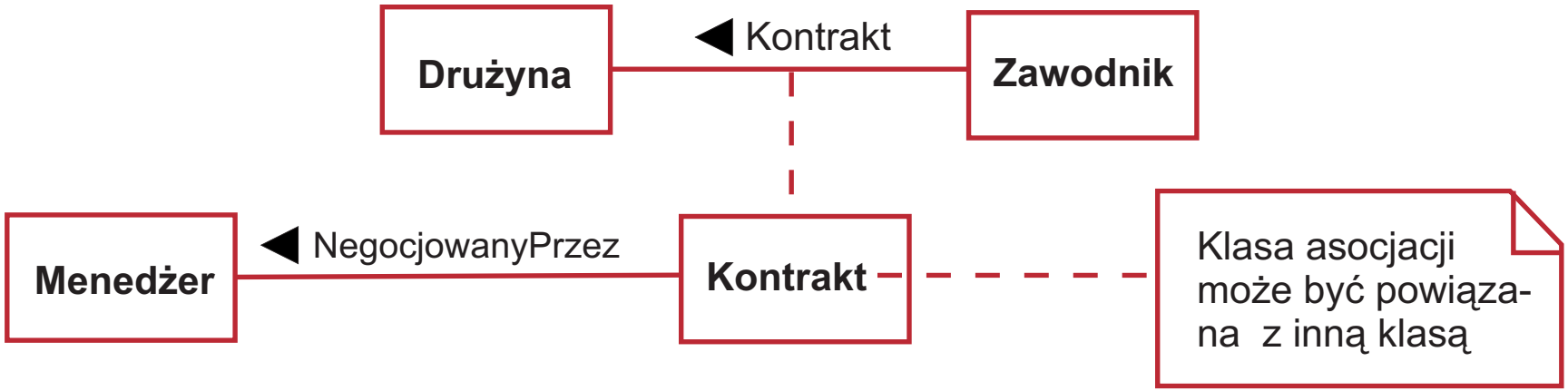
```
dostęp nazwa [indeks] : typ = wartość
```

Asocjacje

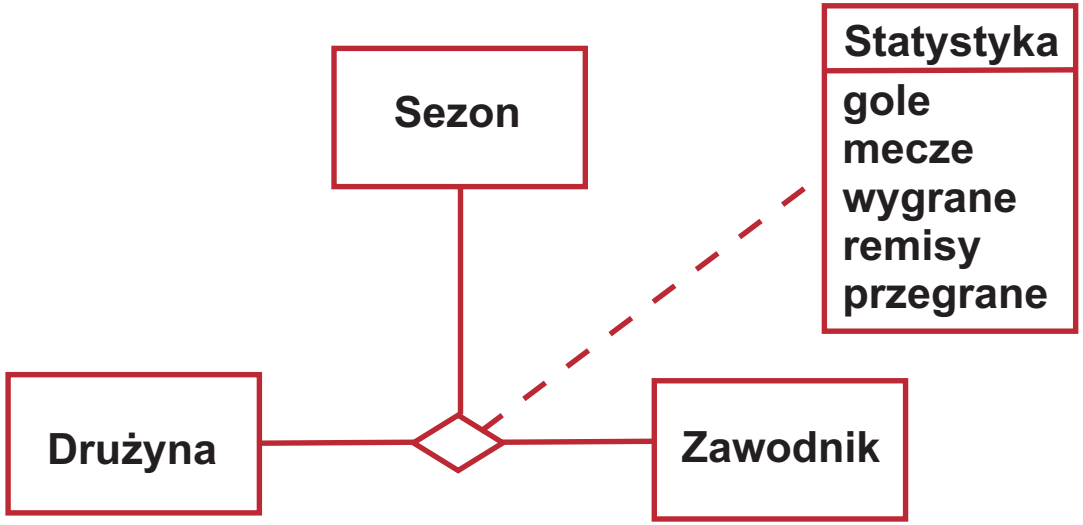
Asocjacja – ogólna relacja zachodząca pomiędzy klasami definiująca typ powiązania.

- asocjacje binarne – wiążą ze sobą dwie klasy,

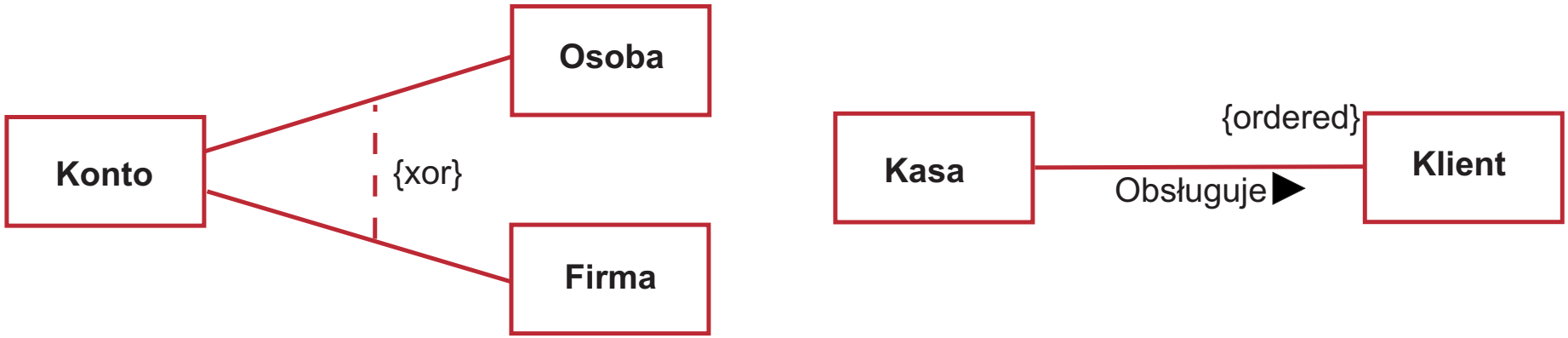
- klasy asocjacji – klasa reprezentująca atrybuty i operacje charakterystyczne dla asocjacji,



- asocjacje n-arne – wiążą ze sobą n-klas,



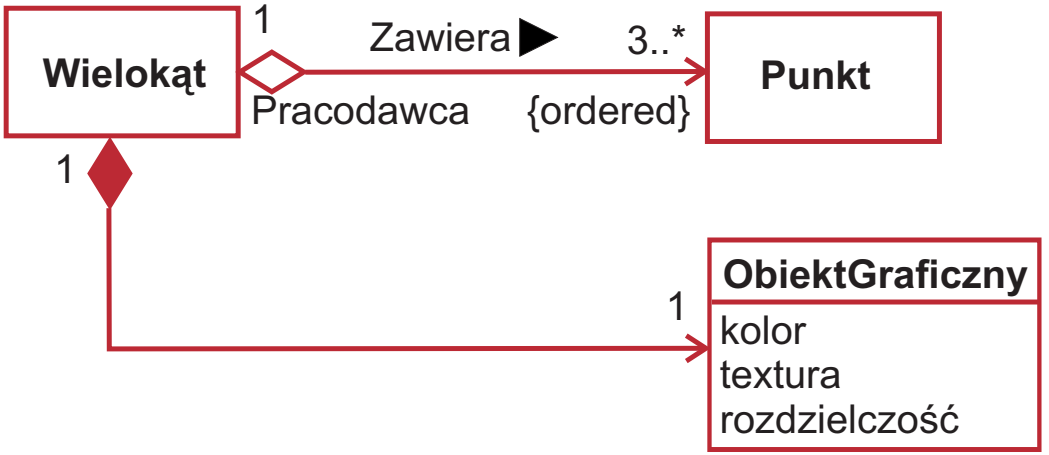
- ograniczenia asocjacji – wymagania nałożone na relacje



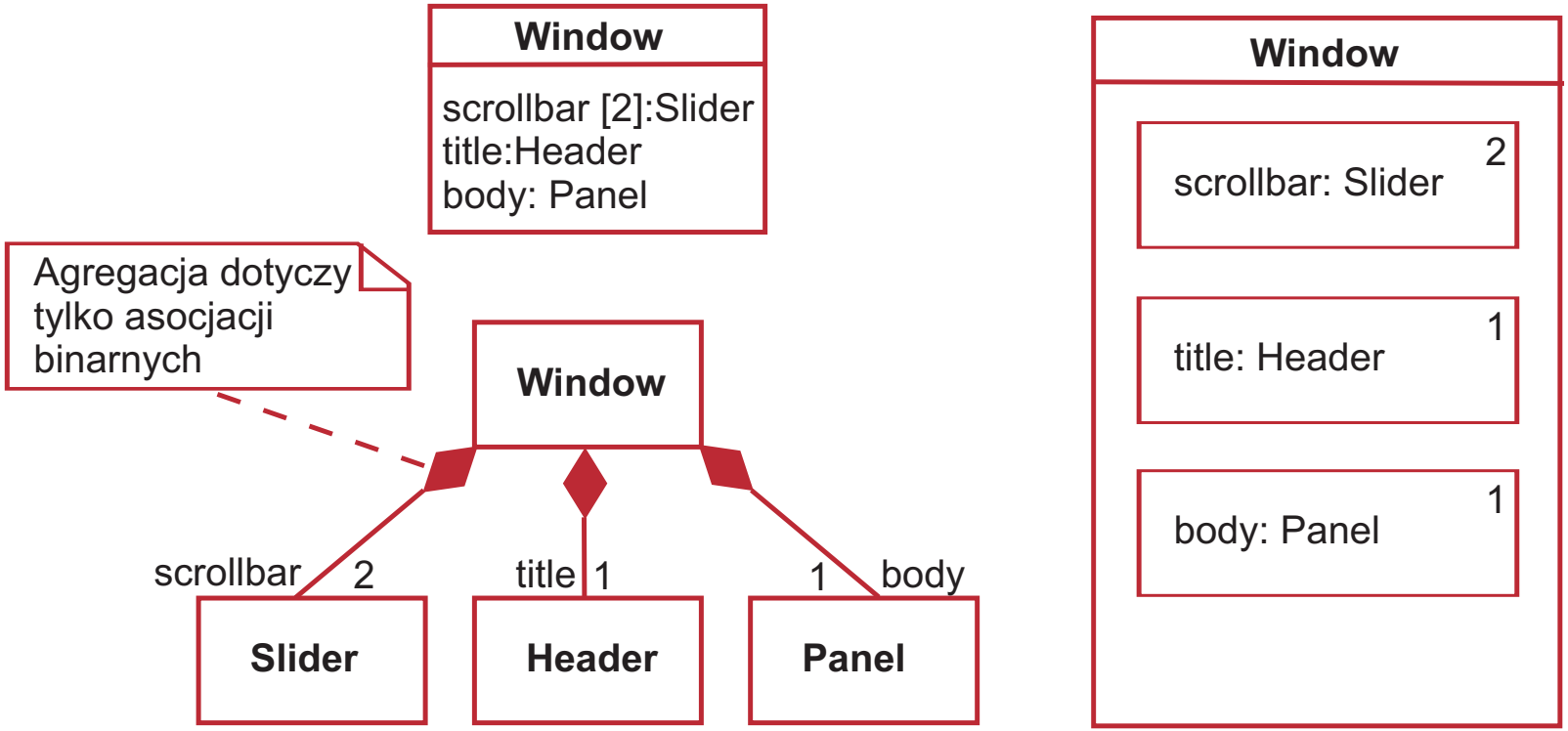
Zakończenia asocjacji

- role – opcjonalne nazwy zadań lub kontekstu występowania klas w ramach relacji,
- specyfikacje liczności – opcjonalne zakresy liczby obiektów z klasy mogących kojarzyć się z obiektami innej klasy w relacji,
  - ▷ literałowe wartości całkowite,
  - ▷ przedziały wartości całkowitych (*=nieograniczony zakres), np. $0..5, 1..*$ $(* = 0..*)$,

- strzałki nawigacyjne – opcjonalne, wskazują czy do danej klasy mogą odwoływać się inne klasy w rozważanej relacji,
- symbole agregacji i kompozycji,

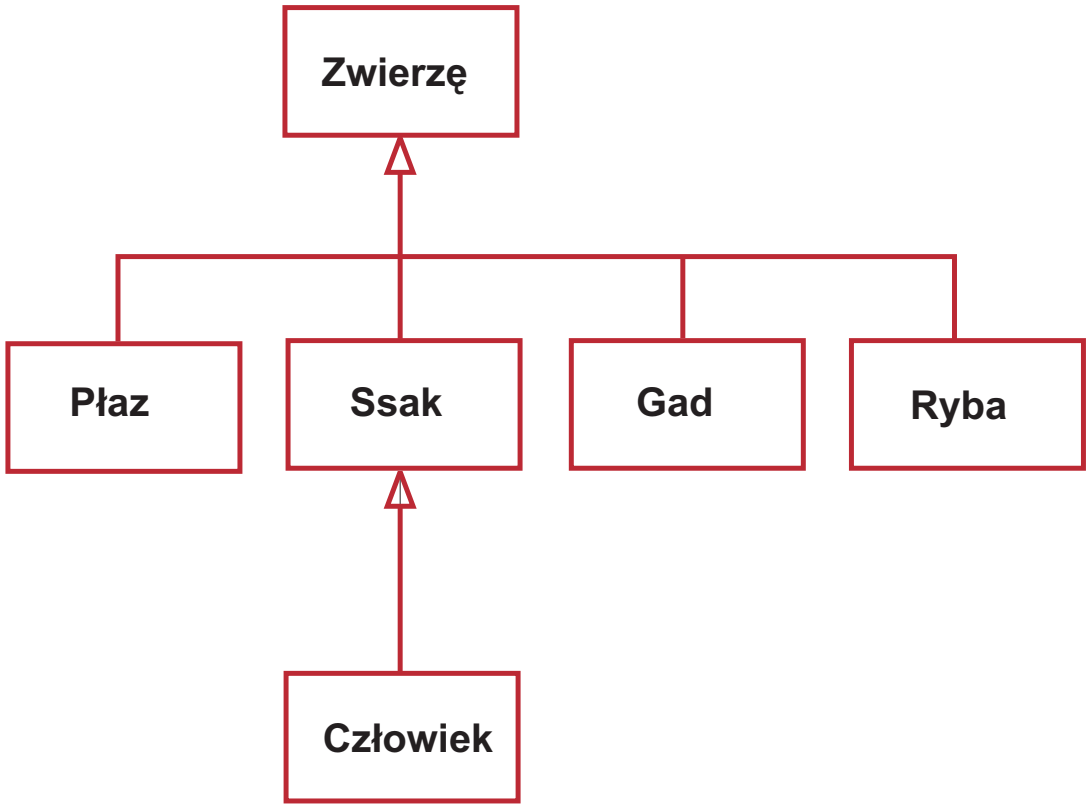


Kompozycja

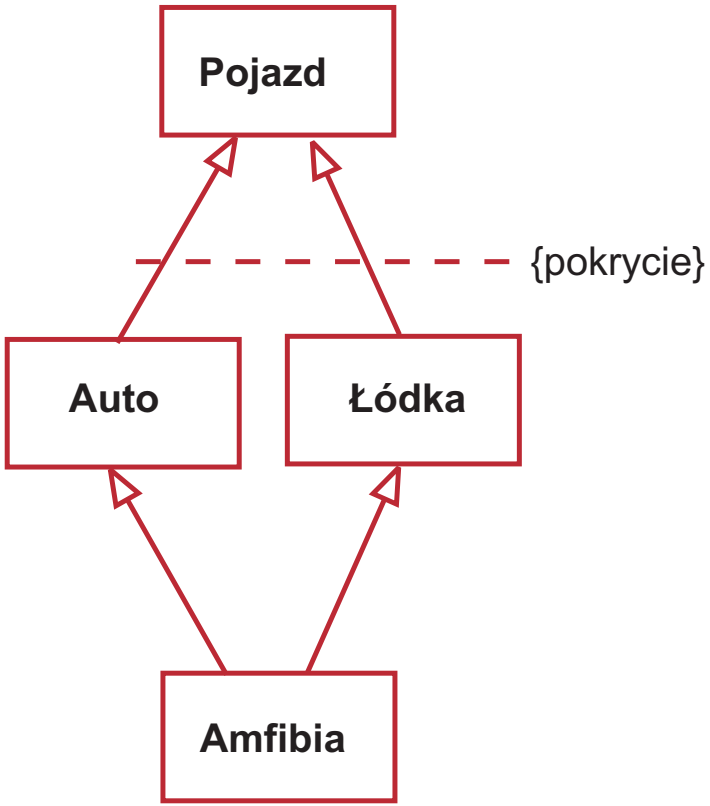

Window
scrollbar [2]:Slider
title:Header
body: Panel
Agregacja dotyczy tylko asocjacji binarnych
Window
scrollbar
2
title
1
1
body
Slider
Header
Panel
Window
scrollbar: Slider
2
title: Header
1
body: Panel
1

Generalizacje

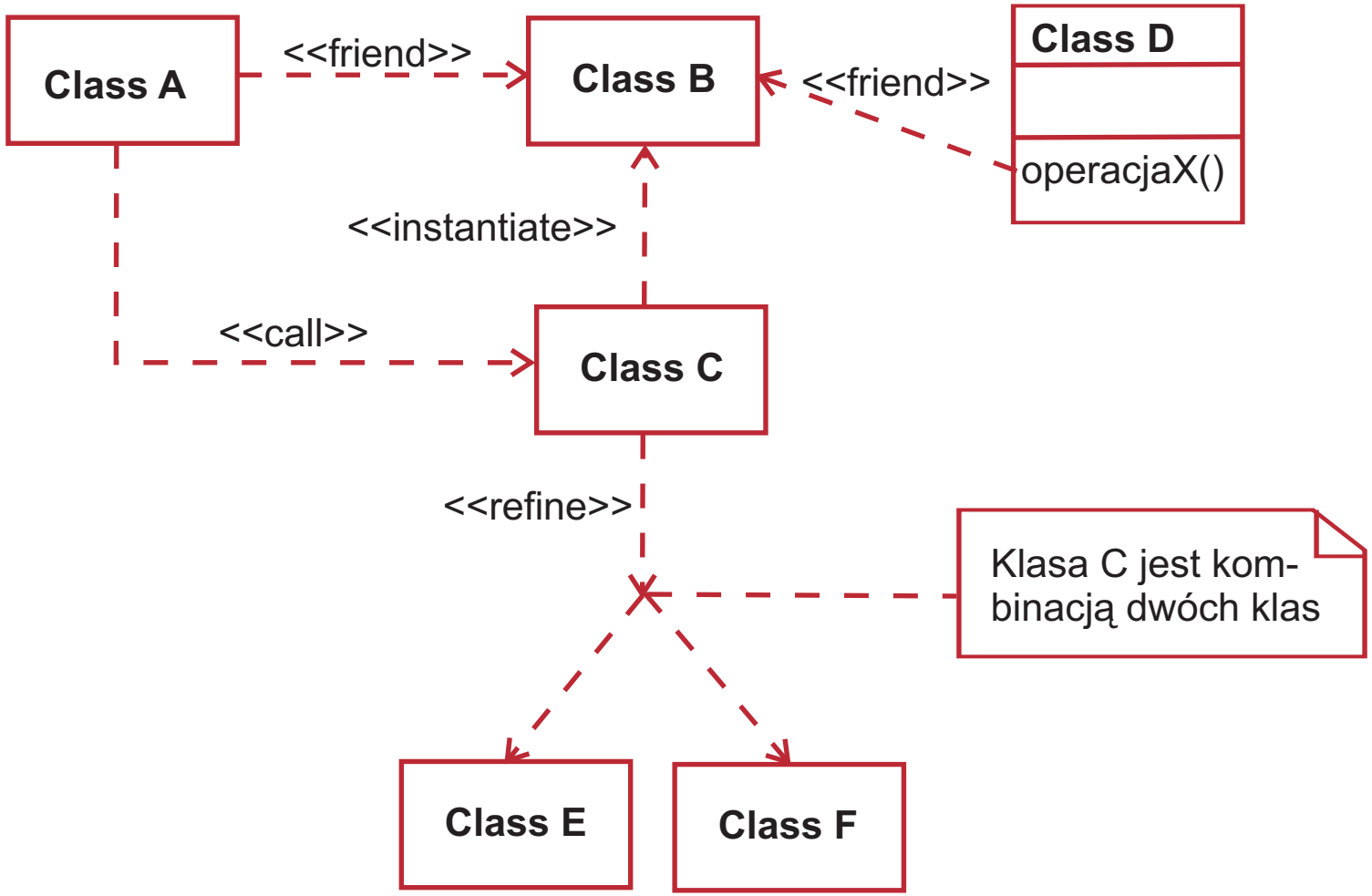
Związki generalizacji dotyczą uogólnienia związków dziedziczenia i mogą istnieć dla klas nierozróżnialnych oraz typów, klas implementacji i interfejsów.



Generalizacja wirtualna



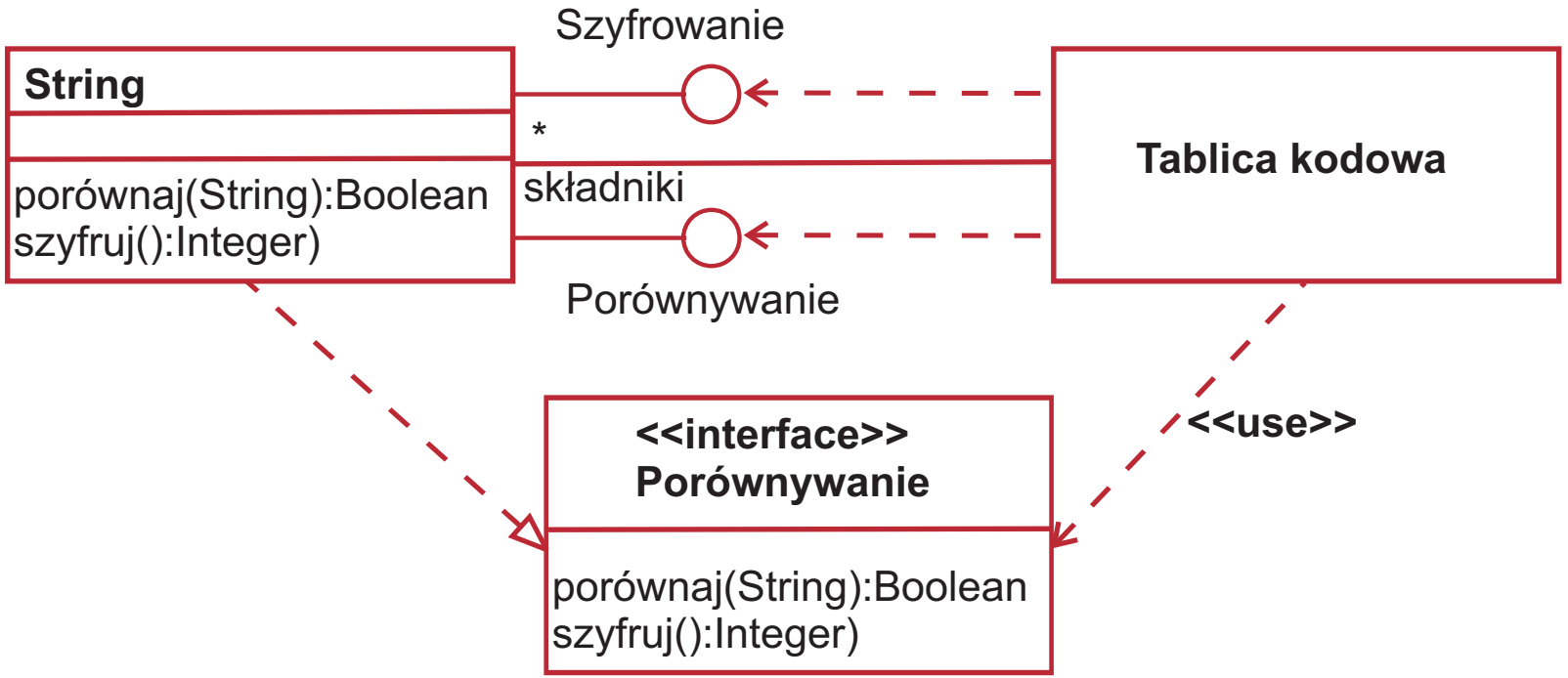
Zależności


Class A
<<friend>>
Class B
<<friend>>
Class D
operacjaX()
<<instantiate>>
<<call>>
Class C
<<refine>>
Klasa C jest kombinacją dwóch klas
Class E
Class F

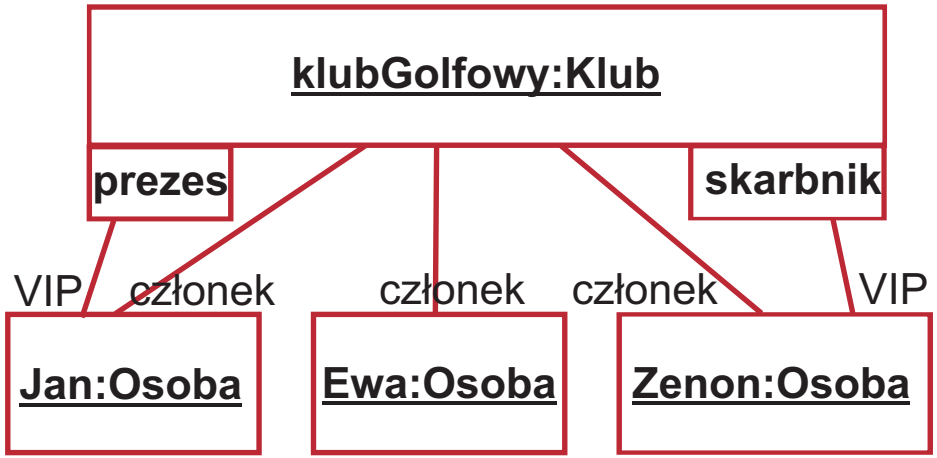
Interfejsy i realizacje

Interfejs – jest zbiorem operacji, które określają pewien aspekt zachowania klasy i które są udostępniane innym klasom. Jest to klasa, która może mieć operacje, lecz nie może mieć atrybutów, asocjacji ani metod.

Realizacja – związek pomiędzy specyfikacją klasy a jej implementacją (np. między klasą i interfejsem)



Powiązania



Pakiety i podsystemy

Pakiet jest elementem grupującym i organizującym, w którym umieszczane są inne elementy. Musi mieć unikatową nazwę.

Podsystem jest elementem grupującym i organizującym elementy, które razem świadczą usługi w taki sposób, że inne elementy mają dostęp wyłącznie do tych usług.