Podstawy modelowania w języku UML

dr hab. Bożena Woźna-Szcześniak, prof. UJD

Uniwersytet Humanistyczno-Przyrodniczy im. Jana Długosza w Częstochowie

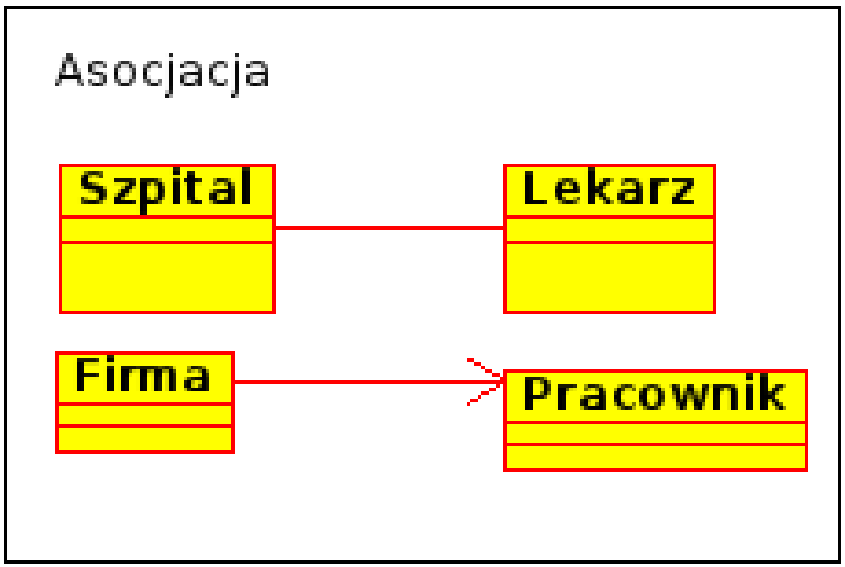
Wykład 2

Związki między klasami

- Asocjacja (ang. Associations)
- Uogólnienie, dziedziczenie (ang. Generalizations)
- Agregacja (ang. Aggregations)
- Kompozycja (ang. Composite aggregation)
- Zagnieżdżenia (ang. Nestings)
- Klasy asocjacyjne (ang. Association Classes)
- Zależności (ang. Dependencies)

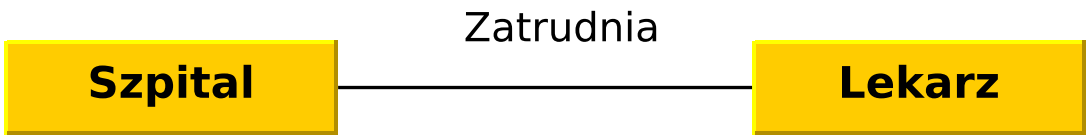
Asocjacja

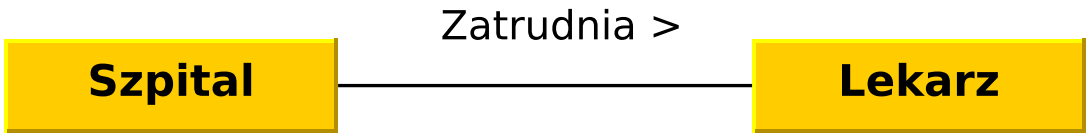
- Asocjacja jest podstawowym rodzajem związków/relacji między klasami i oznacza istnienie trwałego powiązania pomiędzy nimi.
- Asocjacja może zawierać nazwane role na każdym końcu, liczebności, kierunki i ograniczenia.
- Przykłady asocjacji:
  - Student studiuje na uczelni
  - Piłkarz gra w drużynie piłkarskiej
  - Lekarz pracuje w szpitalu, itd



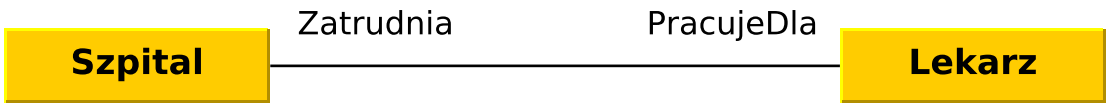
Nazwy, role i liczebność asocjacji

- **Nazwa asocjacji** wskazuje bezpośrednio czynność, jaka zachodzi pomiędzy klasami. Może także wskazywać kierunek.



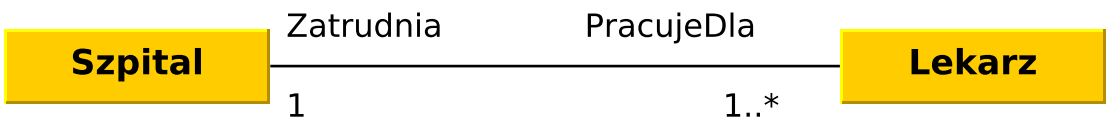


- **Role** określają jaką rolę pełni dana klasa w asocjacji.



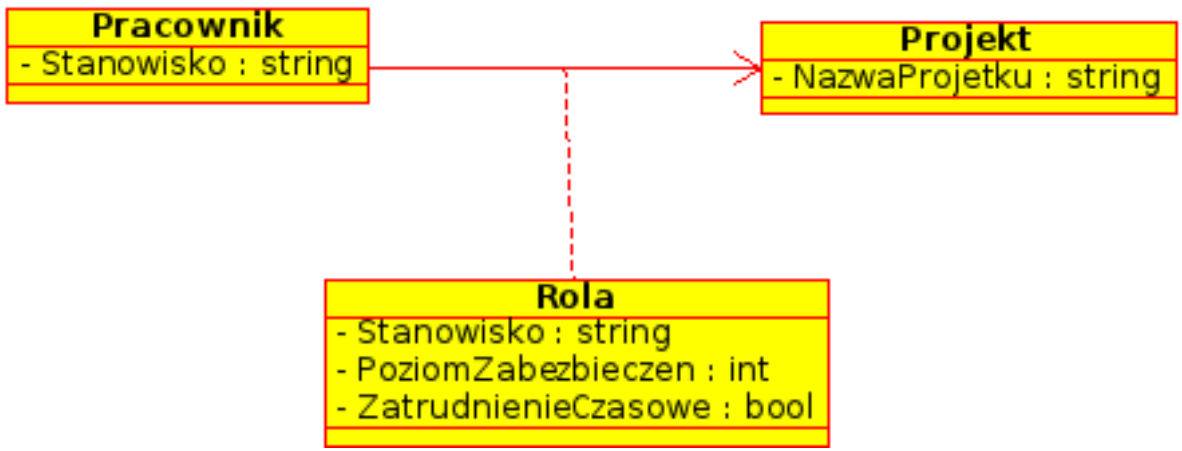
Nazwy, role i liczebność asocjacji

- **Liczebność** określa, ile obiektów jednej klasy bierze udział w asocjacji drugiej klasy.
- Liczebność można określić stałą cyfrą, np.: 1, 2, 3, 4, itd.
- Liczebność można określić jako nieskończoność - oznaczane przez *
- Liczebność można określić przedziałem, np.
  - 0..4 - od 0 do 4
  - 2..5 - od 2 do 5
  - 10..1000 - od 10 do 1000
  - 1..* - od 1 do nieskończoność (minimum 1), itd



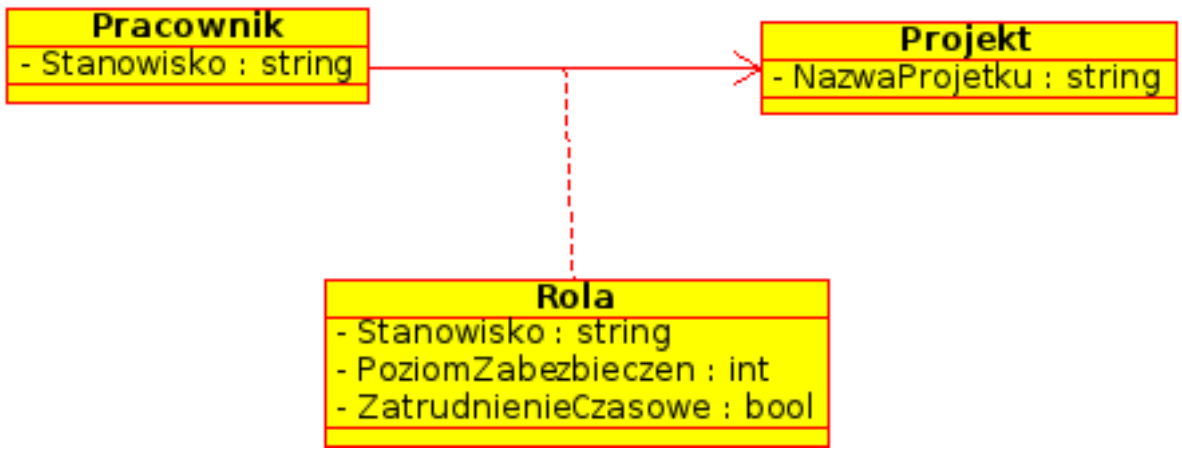
Klasy asocjacyjne

- **Klasa asocjacyjna** to klasa powiązana za pomocą linii przerywanej z asocjacją.
- **Klasa asocjacyjna** opisuje asocjacje pomiędzy klasami, których ta asocjacja dotyczy.
- **Klasa asocjacyjna** umożliwia asocjacji posiadanie operacji i atrybutów.



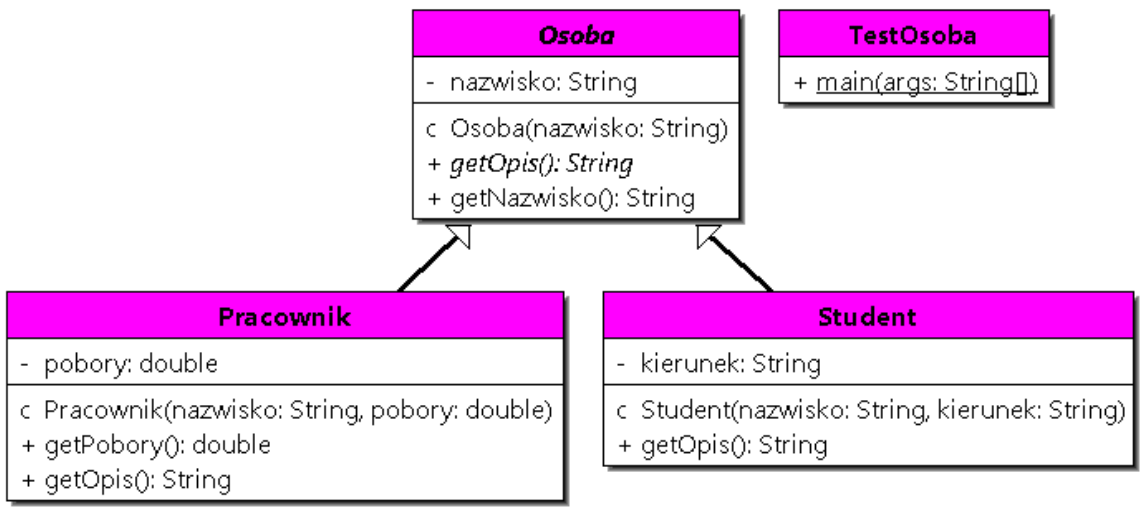
Klasy asocjacyjne

- W przypadku występowania klasy asocjacyjnej w kodzie programu, klasy biorące udział w asocjacji mogą, ale nie muszą, posiadać bezpośrednie połączenia pomiędzy sobą.
- Zatem nasza klasa Pracownik nie musi mieć bezpośredniego połączenia z klasą Projekt. Klasa Pracownik może mieć na przykład połączenie z klasą Rola, zaś klasa Rola z klasą Projekt. Dzięki temu przez pośrednictwo klasy asocjacyjnej Rola, klasa Pracownik jest powiązana z klasą Projekt.



Dziedziczenie (generalizacja)

- **Generalizacja** odpowiada dziedziczeniu znanemu z języków programowania.
- **Generalizacja** to związek pomiędzy bardziej ogólną klasą (rodzicem) a klasą bardziej szczegółową (dzieckiem).
- Przykłady generalizacji:
  - Kwadrat jest Figurą
  - Fiat jest Samochodem
  - Pies jest Zwierzęciem, itd.



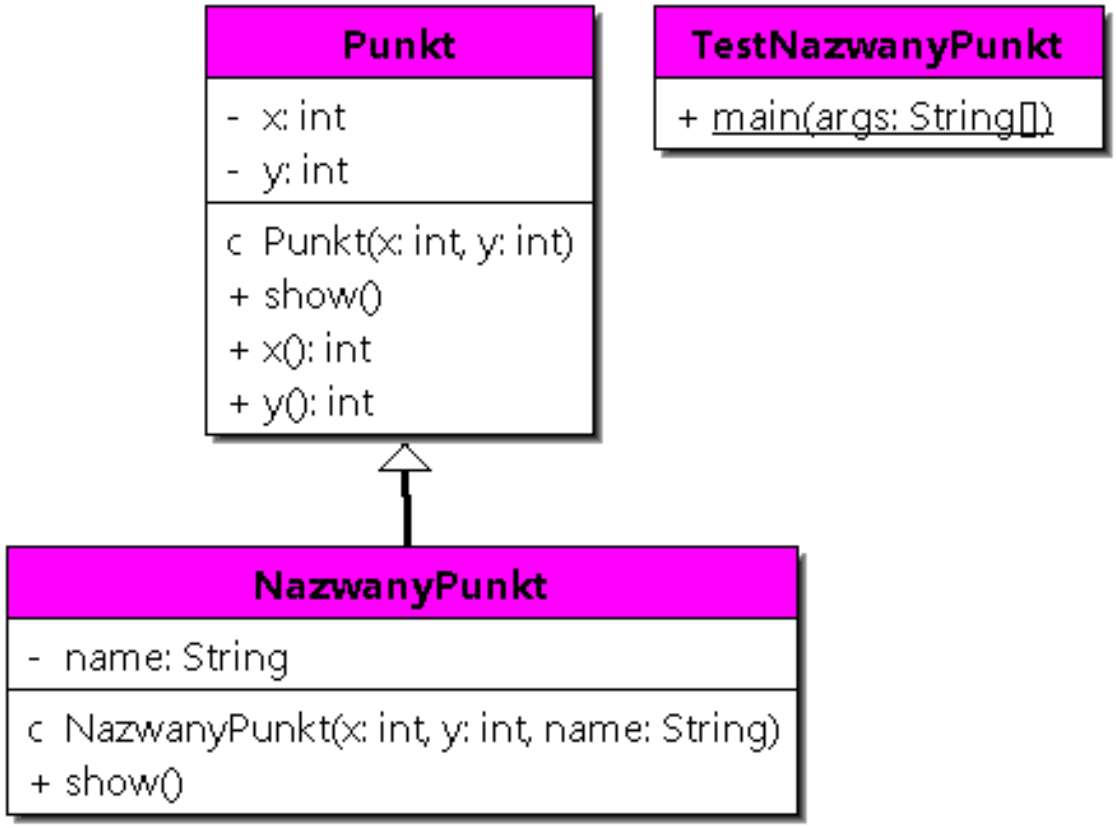
Klasa TestNazwanyPunkt, autor: dr hab. Andrzej Zbrzezny, prof UJD I

```
public class TestNazwanyPunkt {
    public static void main(String[] args)
    {
        NazwanyPunkt a = new NazwanyPunkt(3, 5, "port");
        a.show();
        Punkt b = new Punkt(3, 5);
        b.show();
        Punkt c = new NazwanyPunkt(3, 6, "tawerna");
        c.show();
        a = (NazwanyPunkt) c;
        a.show();
    }
}

class Punkt {
    Punkt(int x, int y) {
        this.x = x; this.y = y;
    }
    public void show()  {
        System.out.println("<" + x + ", " + y + ">");
    }
```

Klasa TestNazwanyPunkt, autor: dr hab. Andrzej Zbrzezny, prof UJD II

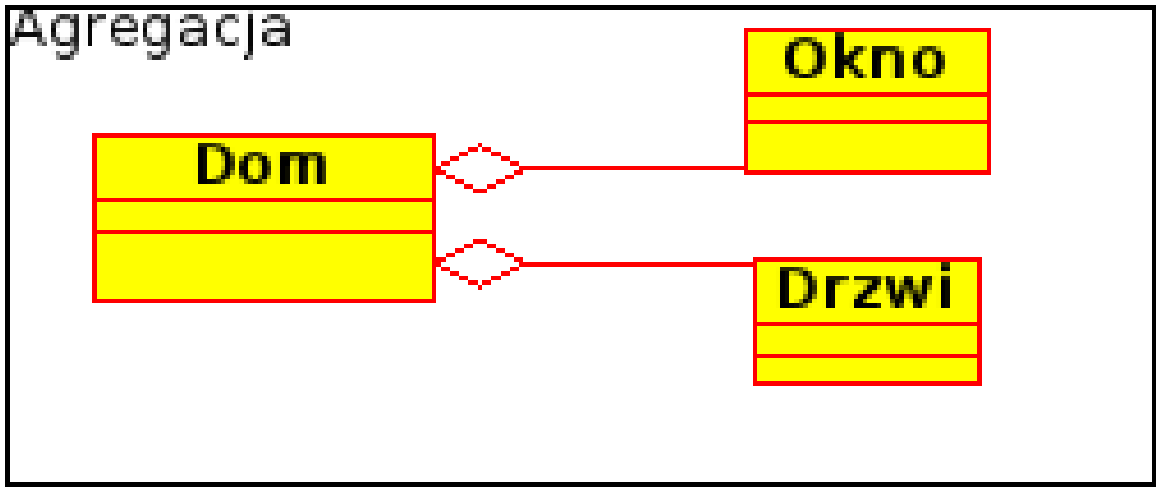
```
    public int x() { return x; }
    public int y() { return y; }
    private int x, y;
}

class NazwanyPunkt extends Punkt {
    NazwanyPunkt(int x, int y, String name) {
        super(x, y);
        this.name = name;
    }
    public void show() {
        System.out.println(name + ":<" + x() + ", " + y() + ">");
    }
    private String name;
}
```

Diagram klas dla klasy `TestNazwanyPunkt`



Agregacja

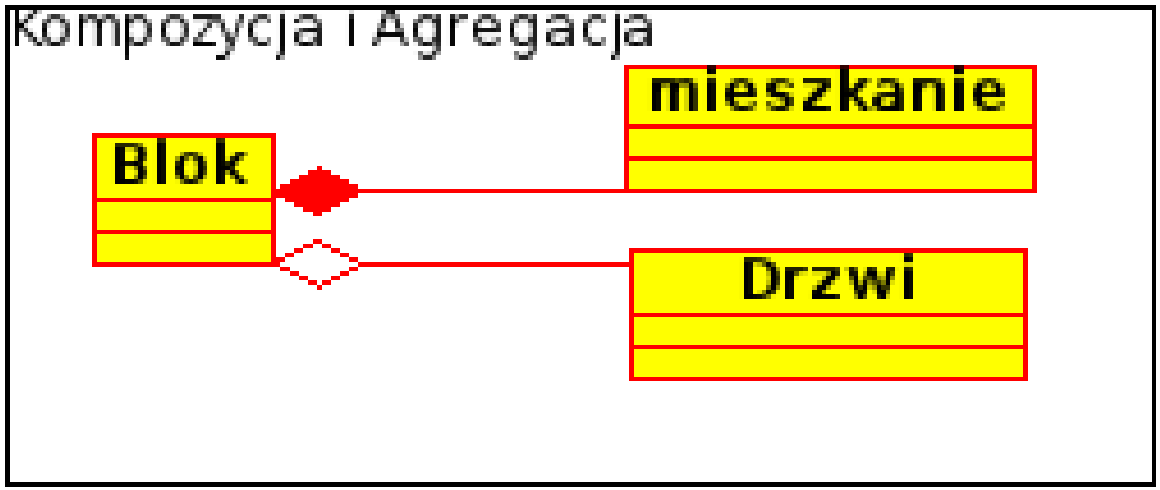
- Agregacje są wykorzystywane w celu przedstawienia elementów, które są złożone z mniejszych elementów.
- Agregacja, w skrócie, oznacza zawieranie, np.
  - Dom zawiera okno
  - Sygnalizacja świetlna zawiera żarówkę
  - Mieszkanie zawiera telefon
- Agregację na diagramie UML oznacza się strzałką z końcem zakończonym pustym rombem, skierowany w kierunku klasy rodzicielskiej.



Kompozycja I

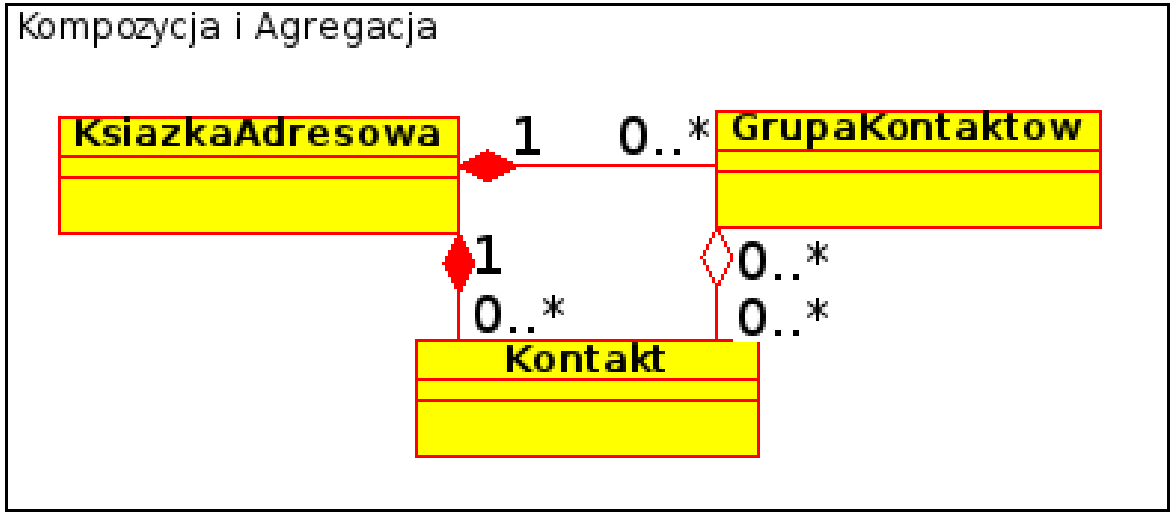
- Kompozycja jest szczególnym przypadkiem agregacji.
- Kompozycja od agregacji różni się tym, że klasa posiada obiekty (składa się z obiektów), które bez tej klasy nie mogły by istnieć.
- Jeśli rodzic w kompozycji jest usuwany, zwykle wszystkie jego części są usuwane z nim; jednakże część może zostać indywidualnie usunięte z kompozycji, bez konieczności usuwania całej kompozycji.
- Kompozycja jest relacją przechodnią, asymetryczną i może być rekurencyjne.
- Kompozycję na diagramie UML oznacza się strzałką z końcem zakończonym wypełnionym rombem, skierowany w kierunku klasy rodzicielskiej.

Kompozycja II



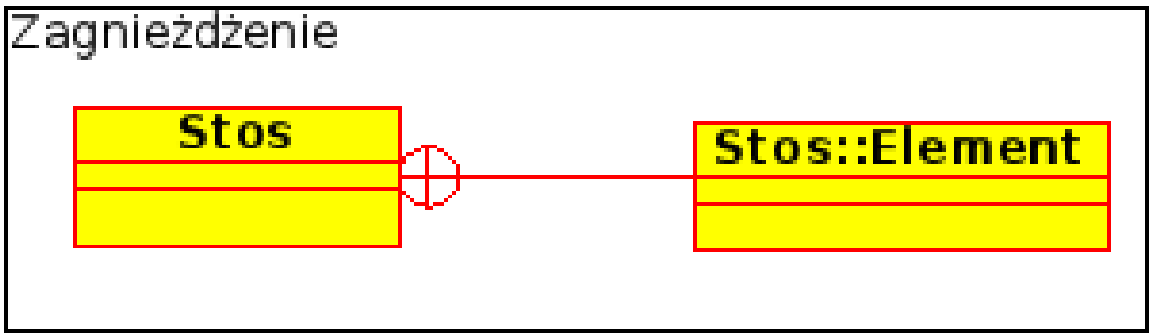
Różnica między agregacją i kompozycją - przykład

Książka adresowa składa się z wielu kontaktów i grup kontaktów. Grupa kontaktów to wirtualne grupowanie kontaktów; każdy kontakt może być zawarty w więcej niż jednej grupie kontaktów. Usunięcie książki adresowej powoduje, że wszystkie kontakty i grupy kontaktowe zostaną usunięte. Usunięcie grupy kontaktów, nie powoduje usunięcia żadnego kontaktu.

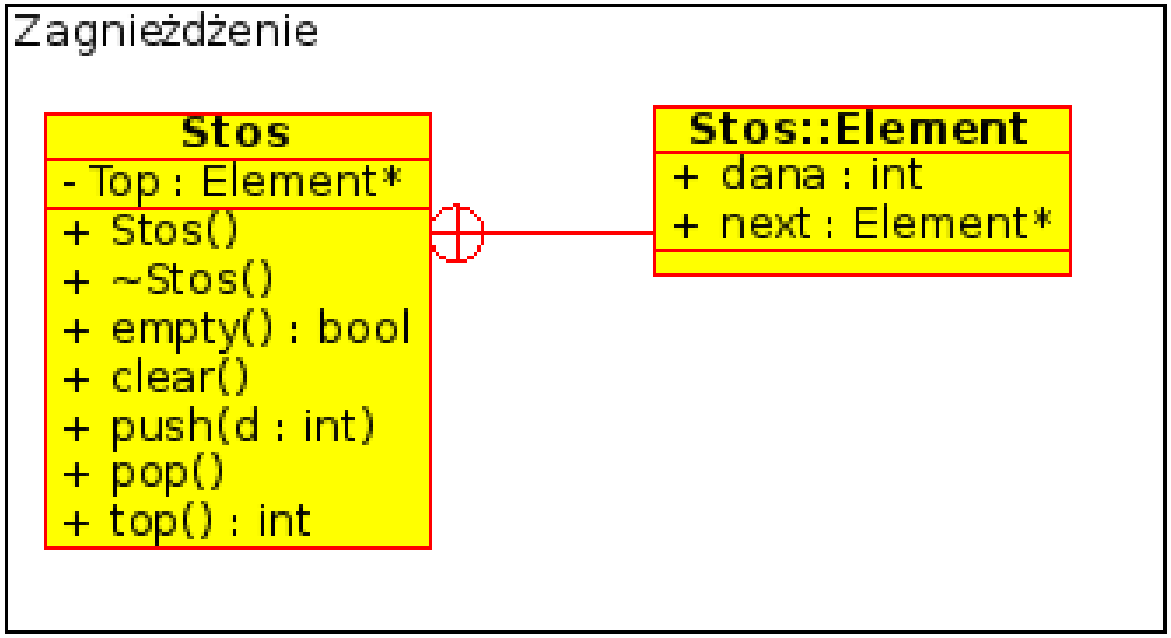


Zagnieżdżenia

Zagnieżdżenie jest relacją pokazującą, że element źródłowy (np. klasa Element) jest zagnieżdżona w elemencie docelowym (np. klasa Stos).



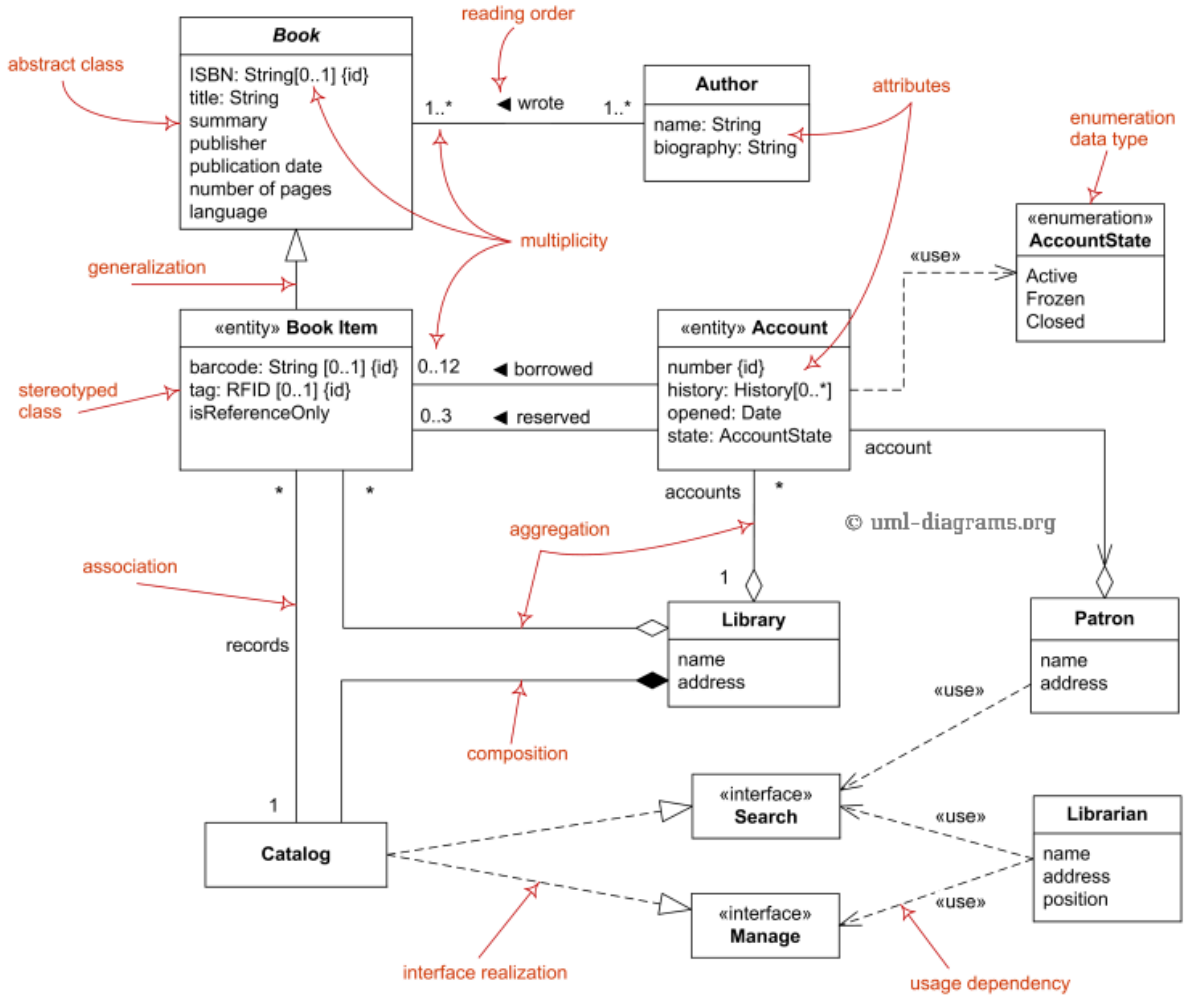
Zagnieżdżenia I



Zależności

- Zależności używane są do modelowania szerokiego zakresu związków pomiędzy elementami modelu.
- Zwykle stosowane są na wczesnym etapie procesu projektowania, gdy wiadomo, że istnieje jakiś związek między dwoma elementami, ale jest zbyt wcześnie, aby dokładnie wiedzieć, jaki jest to związek.
- W późniejszym etapie procesu projektowania zależności zostaną uszczegółowione lub zastąpione bardziej konkretnym typem związku.

Diagramy klas i związki- przykład



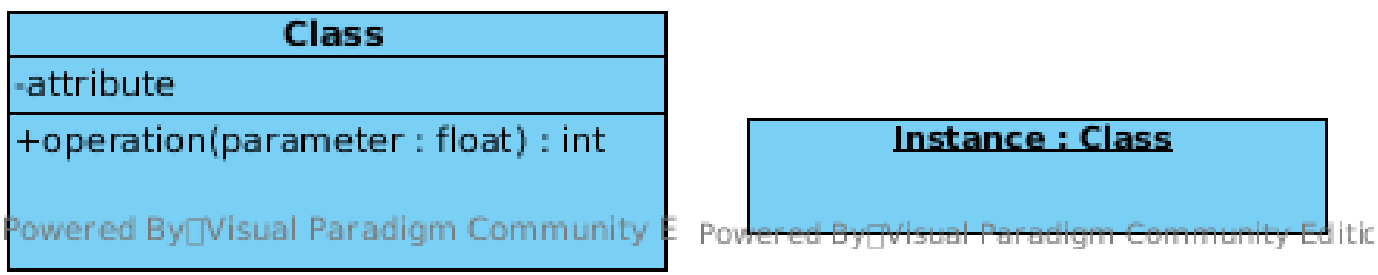
Źródło:http://www.uml-diagrams.org/class-diagrams-overview.html

Diagramy obiektów

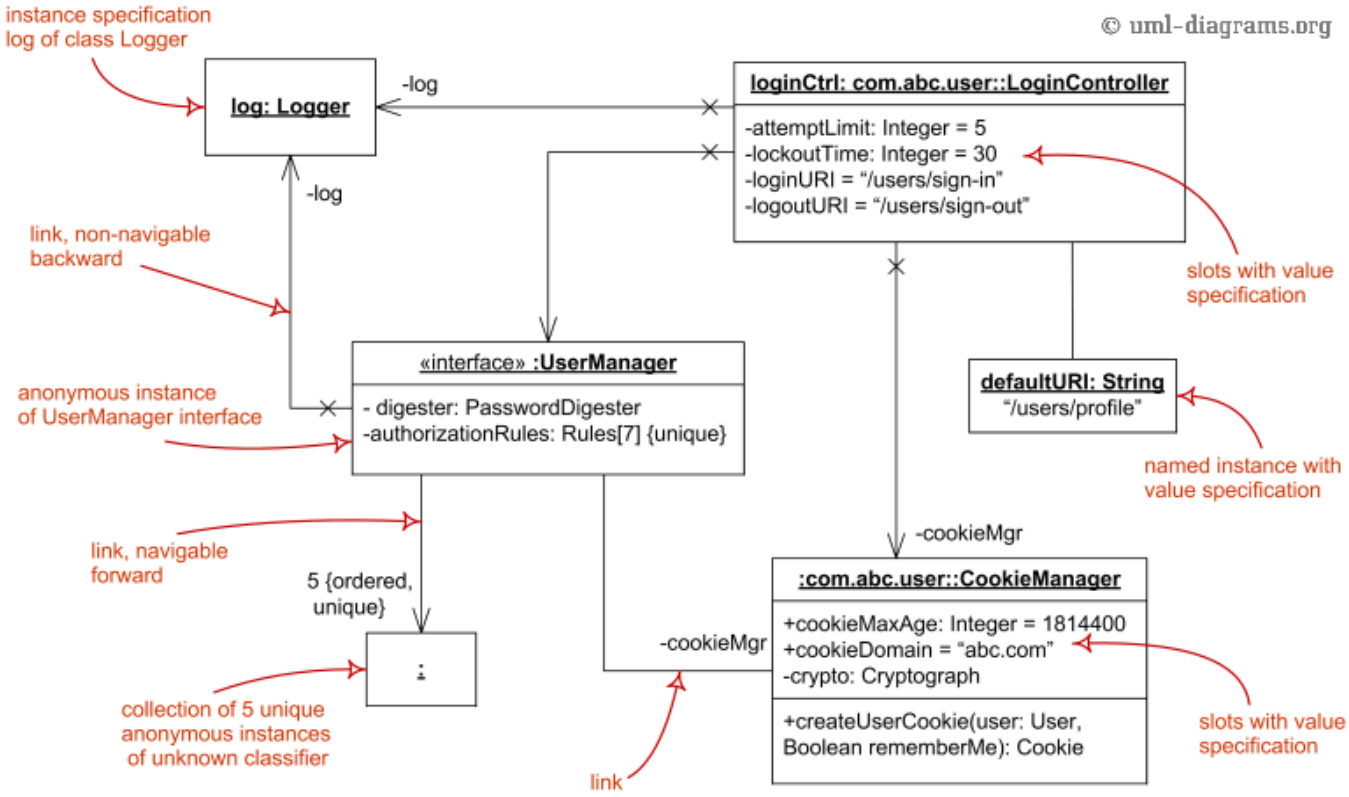
- **Diagram obiektu** może być uważane za szczególny przypadek diagramu klasy.
- **Diagramy obiektu** wykorzystują podzbiór elementów diagramu klas, aby podkreślić związek pomiędzy instancjami klas w pewnym momencie czasowym.
- **Diagramy obiektu** nie pokazują niczego architektonicznie innego do diagramów klas, ale odzwierciedlają wielość i role.

Diagramy obiektów - różnice w wyglądzie pomiędzy klasą i obiektem

- Klasy składa się z trzech części: nazwa klasy, atrybuty i operacje.
- Domyślnie obiekty nie mają takiego przedziału, posiadają tylko nazwę.
- Obiekty na diagramie prezentuje się za pomocą prostokątów, w których nazwa jest podkreślona. Po niej, po dwukropku, znajduje się nazwa klasy, której obiekt reprezentuje dany prostokąt.



Diagramy obiektów - przykład



Źródło:`http://www.uml-diagrams.org/class-diagrams-overview.html#object-diagram`