Programowanie obiektowe

Wykład 11

Marcin Młotkowski

30 maja 2019

Plan wykładu

Plan wykładu


```
import java.awt.*;

public class Aplikacja implements Runnable
{
   private int xPos;
   private int yPos;

   private String tempDir = "/tmp";

   public static void main(String[] args)
   {
      Aplikacja app = new Aplikacja();
      app.run();
   }

}
```

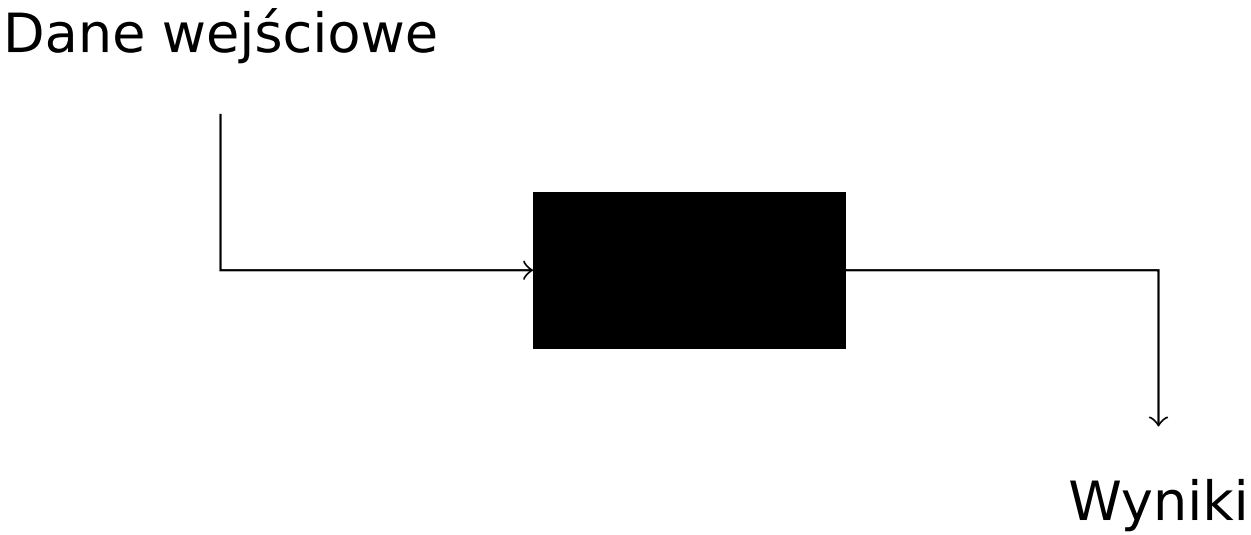
Przypomnienie (akwarystyka)

Specyfikacja programu

„Napisz mi program, w którym bym notował elektronicznie różne fakty związane z moim akwarium, na przykład liczbę i gatunki ryb, temperaturę, częstotliwość karmienia”.

Metody projektowania: prosta dziedzina

Przykłady: numeryczna analiza danych, przetwarzanie danych finansowych.



Przypomnienie (programowanie funkcyjne)

Definicja silni

definicja matematyczna

$$\text{silnia}(n) = \begin{cases} 1 & \text{gdy } n = 0 \\ n * \text{silnia}(n-1) & \text{wpp} \end{cases}$$

Implementacja w Ocaml'u

```
let rec silnia n =
  if n=0 then 1
  else n*silnia(n-1);;
```

Metody projektowania

Dekompozycja funkcjonalna

podział modułów ze względu na funkcje

Graf przepływu danych

Ustalenie kolejności wykonywania operacji na danych

Rozbudowana dziedzina

- Modelowanie danych;
- modelowanie związków między danymi.

Przypomnienie

Specyfikacja programu

„Napisz mi program, w którym bym notował elektronicznie różne fakty związane z moim akwarium, na przykład liczbę i gatunki ryb, temperaturę, częstotliwość karmienia”.

Przypomnienie

Specyfikacja programu

„Napisz mi program, w którym bym notował elektronicznie różne fakty związane z moim akwarium, na przykład liczbę i gatunki ryb, temperaturę, częstotliwość karmienia”.

Przypomnienie

Specyfikacja programu

„Napisz mi program, w którym bym notował elektronicznie różne fakty związane z moim akwarium, na przykład liczbę i gatunki ryb, temperaturę, częstotliwość karmienia”.

Analiza problemu

- Proces poznawania dziedziny;
- zrozumienie szczegółów;
- ustalenie granic dziedziny.

Przyczyny kłopotów z tworzeniem oprogramowania

- Rozbudowa oprogramowania: zwiększanie zarówno dziedziny jak i funkcjonalności;
- błędne założenia o czasie użytkowania;
- zmienność potrzeb, w tym zmienność prawa;
- problemy zapanowania nad dużym projektem.

Cechy projektu

- Abstrakcja danych i abstrakcja operacji
- Enkapsulacja: ukrycie detali implementacyjnej
- Modularność: podział programu na wyraźne fragmenty
- Hierarchia pojęć

Waga dobrego projektu

- Dziedzina zmienia się wolno
- Funkcjonalność może się szybko zmieniać

Plan wykładu

Podział projektu na warstwy

1. Warstwa klas i obiektów
2. Warstwa związków
3. Warstwa tematów
4. Warstwa atrybutów
5. Warstwa usług

Klasy i obiekty

- Osoby, przedmioty, pojęcia
- Zdarzenia
- Role
- Jednostki organizacyjne

Warstwa związków

- generalizacja
- specjalizacja
- agregacja
- kompozycja
- asocjacja

Warstwa tematów

Grupowanie klas w tematyczne klastry

Warstwa tematów

Grupowanie klas w tematyczne klastry

Wielkość tematu: wskazówka

7 ± 2

Warstwa atrybutów

Atrybuty proste

Np imię, nazwisko

Warstwa atrybutów

Atrybuty proste

Np imię, nazwisko

Atrybuty złożone

Np. adres

Warstwa atrybutów

Atrybuty proste

Np imię, nazwisko

Atrybuty złożone

Np. adres

Atrybuty obliczane

Np. wiek,

Warstwa atrybutów

Atrybuty proste

Np imię, nazwisko

Atrybuty złożone

Np. adres

Atrybuty obliczane

Np. wiek,

Unikatowe identyfikatory

Własny ’numer seryjny’ obiektu.

Warstwa atrybutów

Atrybuty proste

Np imię, nazwisko

Atrybuty złożone

Np. adres

Atrybuty obliczane

Np. wiek,

Unikatowe identyfikatory

Własny ’numer seryjny’ obiektu.

Referencje do innych obiektów

Warstwa usług

- Inicjowanie
- Dostęp do atrybutów
- Zmiana stanu

Plan wykładu

Unified Modelling Language

Notacja graficzna, umożliwiająca ilustarcję zagadnień związanych z oprogramowaniem.

Do czego służą diagramy

Przezentacja koncepcji

Do czego służą diagramy

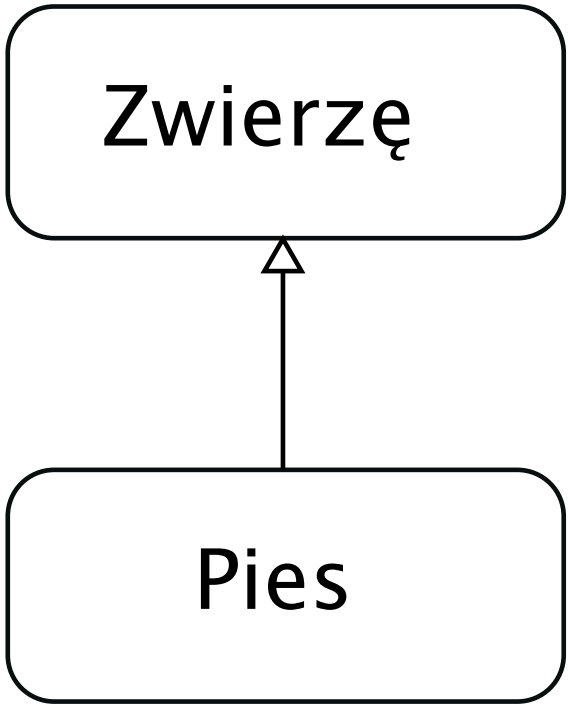
Przezentacja koncepcji

Przentacja specyfikacji.

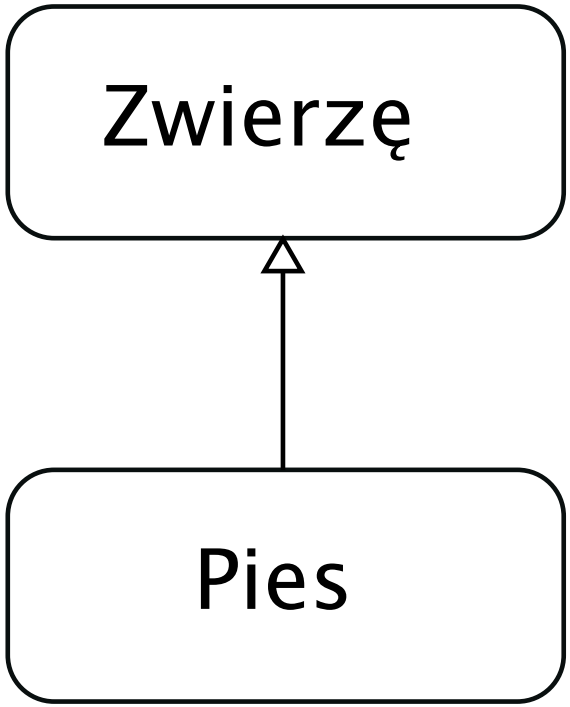
Do czego służą diagramy

- Przezentacja koncepcji
- Przentacja specyfikacji.
- Prezentacja implementacji.

Rola UML



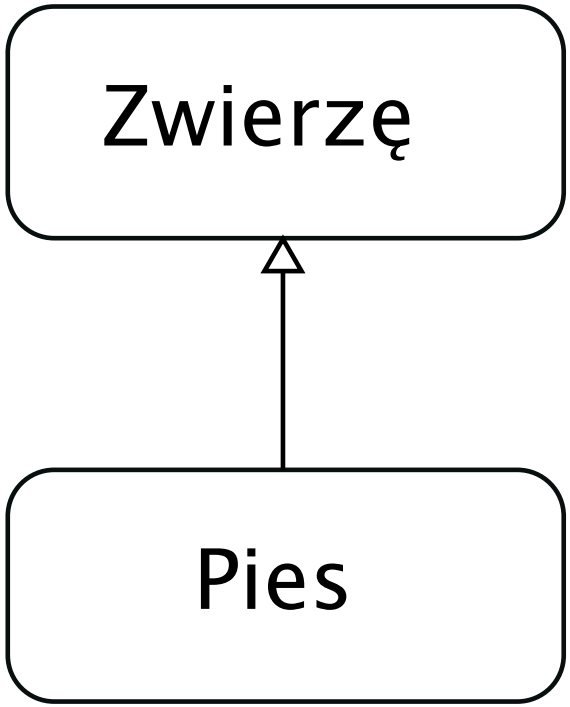
Rola UML



Poziom koncepcyjny

Pies jest zwierzęciem

Rola UML



Poziom koncepcyjny

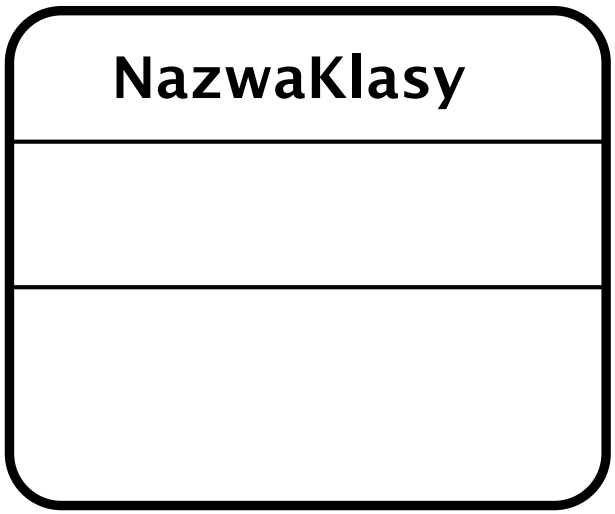
Pies jest zwierzęciem

Poziom specyfikacji/implementacji

```
public class Zwierzę {}

public class Pies : Zwierzę {}
```

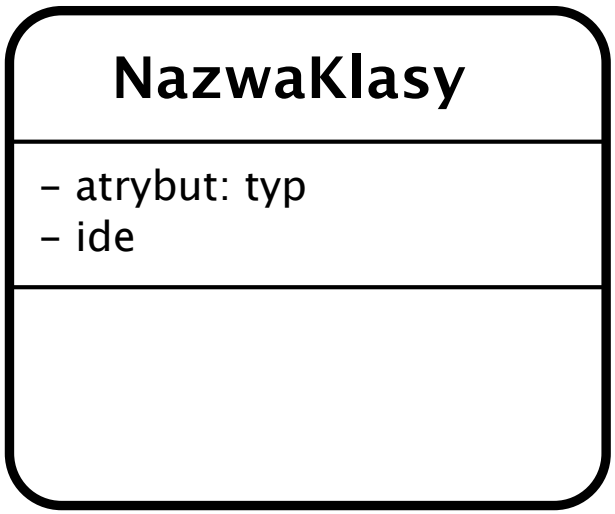
Opis klasy



Opis klasy

- nazwa klasy;

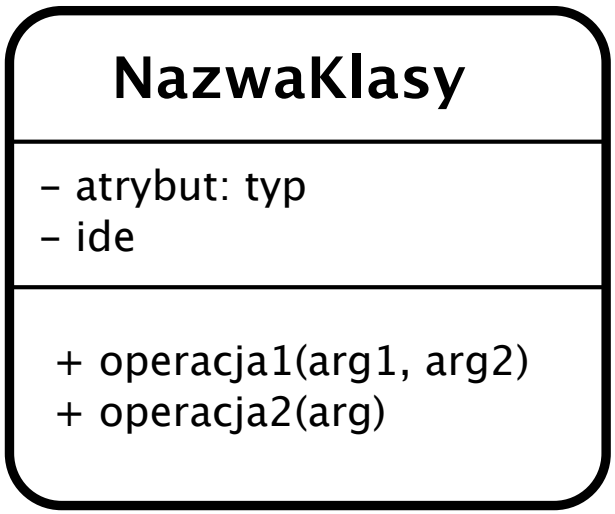
Opis klasy



Opis klasy

- nazwa klasy;
- atrybuty klasy, opcjonalnie z typem atrybutu i widzialnością (+: publiczny,#: zabezpieczony,–: prywatny);

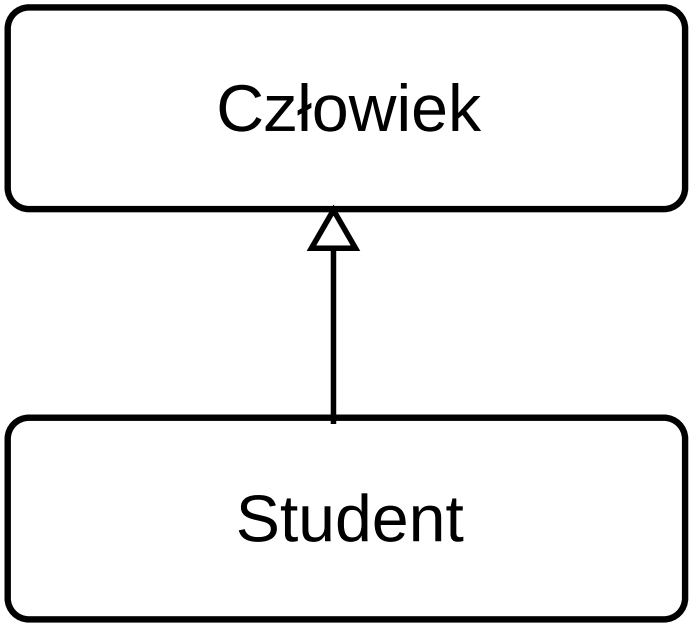
Opis klasy



Opis klasy

- nazwa klasy;
- atrybuty klasy, opcjonalnie z typem atrybutu i widzialnością (+: publiczny,#: zabezpieczony,–: prywatny);
- operacje na klasie, opcjonalnie z typami i widzialnością.

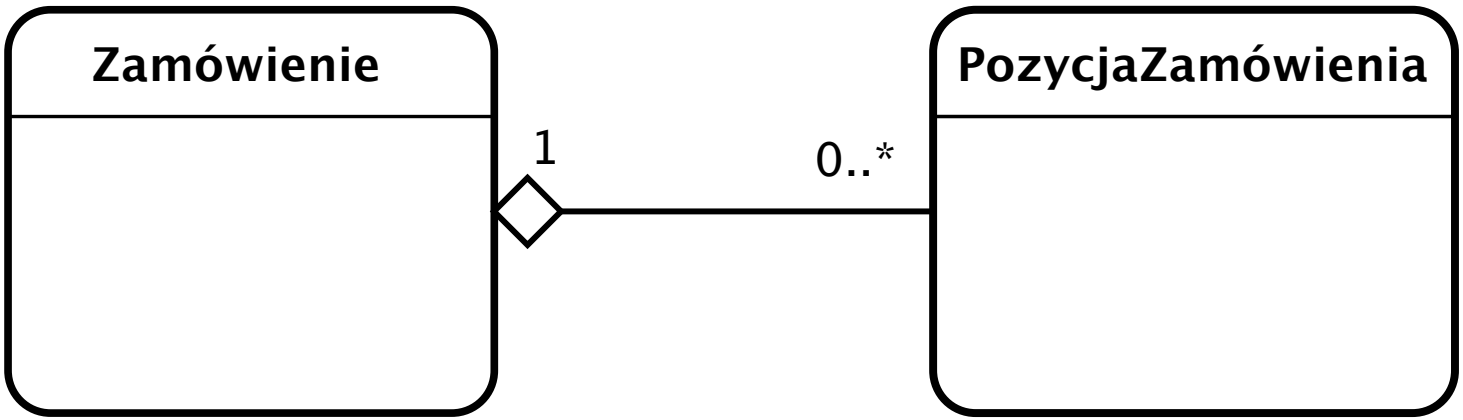
Generalizacja/Specjalizacja



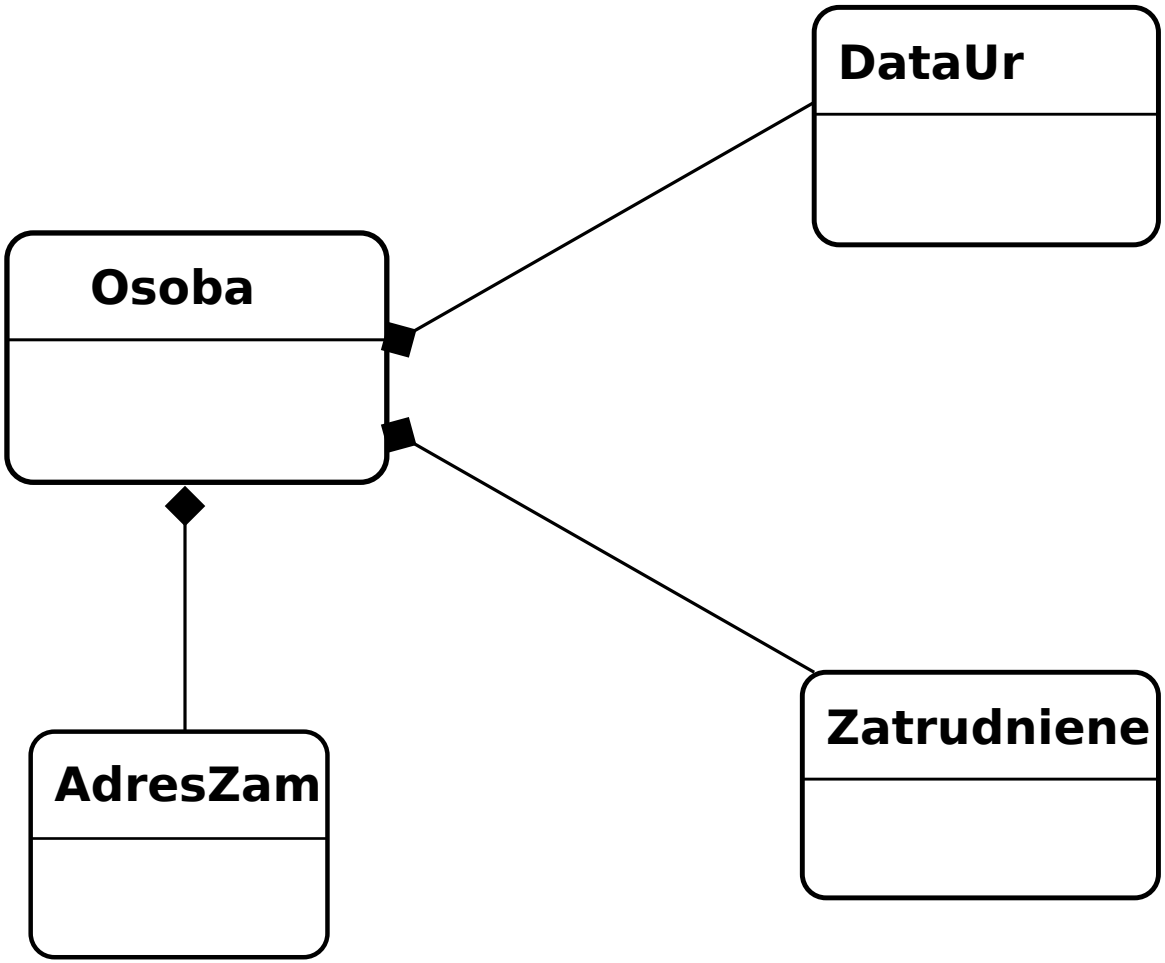
Generalizacja

Specjalizacja

Agregacja



Kompozycja



Opis asocjacji

Opis klasy

- kierunek asocjacji;

Opis asocjacji

Opis klasy

- kierunek asocjacji;
- nazwa zmiennej z referencją;

Opis asocjacji

Opis klasy

- kierunek asocjacji;
- nazwa zmiennej z referencją;
- "liczność asocjacji".

Drzewo binarnych poszukiwań

```
public class DrzewoBinarne<Key, Val> {
  private Węzeł<Key, Val> root = null;
  public void Dodaj(IComparable<Key> klucz,
                    Val wartość) { ... }
}
```

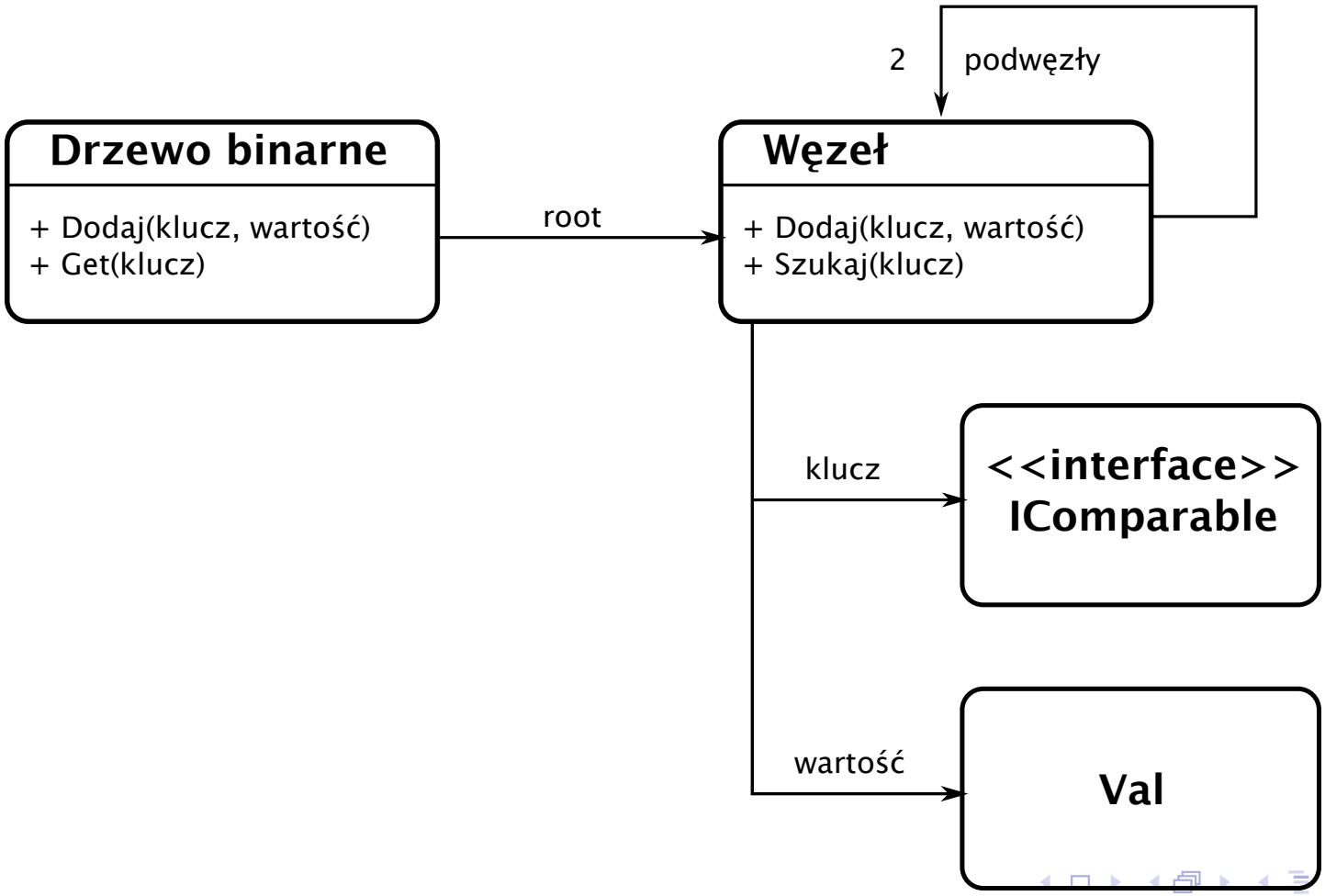
Drzewo binarnych poszukiwań

```
public class DrzewoBinarne<Key, Val> {
  private Węzeł<Key, Val> root = null;
  public void Dodaj(IComparable<Key> klucz,
                    Val wartość) { ... }
}
```

```
class Węzeł<Key, Val> {
  Val wartość;
  IComparable<Key> klucz;

  Węzeł<Key, Val>[] podwęzły = new Węzeł<Key, Val>[2];
}
```

Diagram klas



Plan wykładu

Biblioteka osiedlowa

- Książki i czasopisma
- Czytelnicy
- Karty biblioteczne
- Tymczasowi czytelnicy