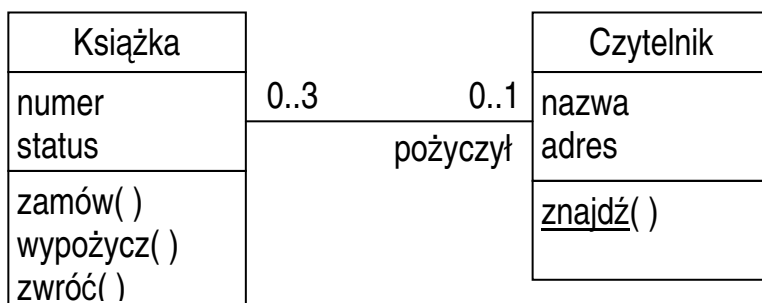


## Diagram klas

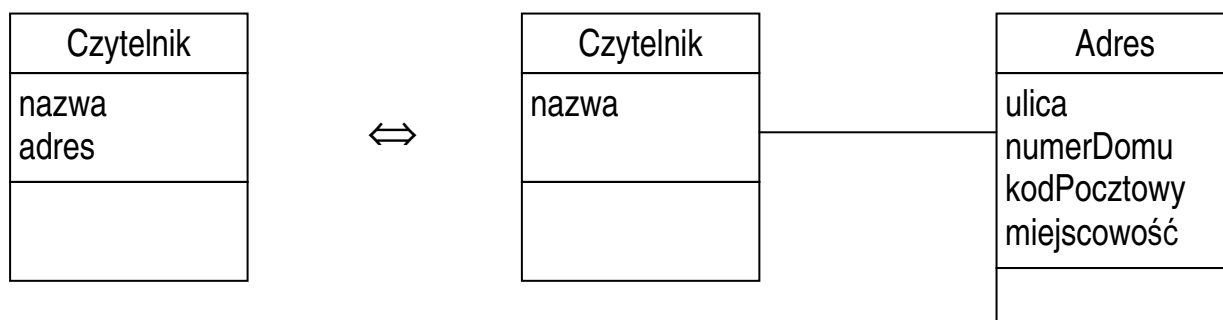
- Model statycznej struktury problemu
- Elementy modelu
  - klasy
  - relacje (*uogólnienie, stowarzyszenie*)
- Reprezentacja graficzna



## Perspektywy widzenia modelu

- **Model pojęciowy**

- model analizy
- pokazuje klasy i atrybuty, asocjacje i generalizacje



- **Model specyfikacyjny**

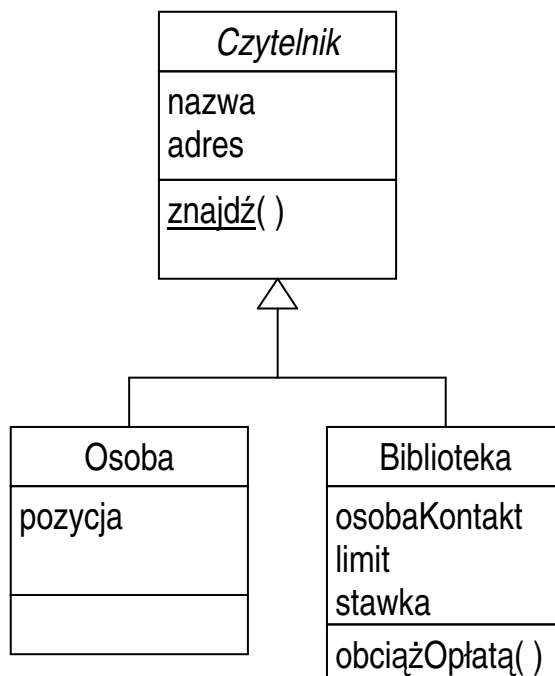
- model projektu (architektury)
- pokazuje interfejsy i ich klasyfikację
- nie pokazuje metod

- **Model implementacyjny**

- model projektu i implementacji
- określa strukturę danych i dziedziczenie
- operacje są metodami klasy

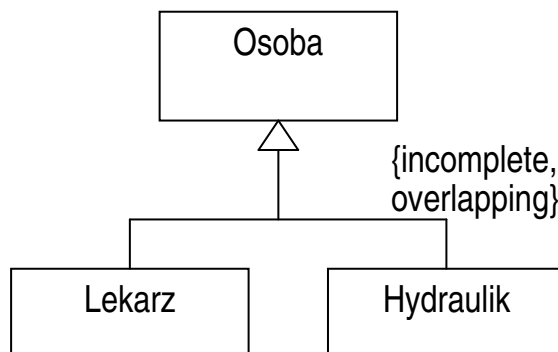
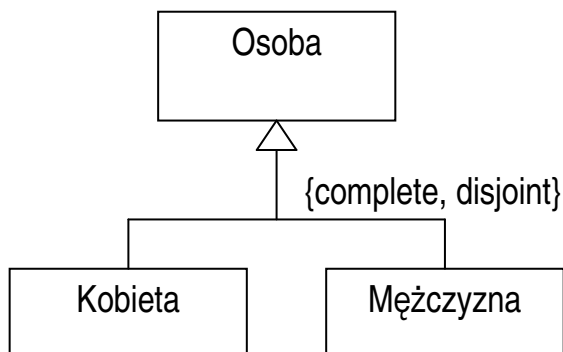
## Uogólnienie

Klasyfikacja, modele na różnych poziomach abstrakcji



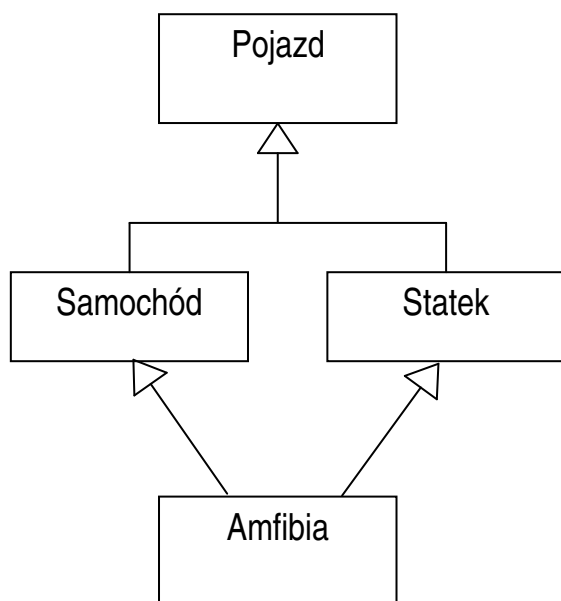
- **Model pojęciowy**
  - klasyfikacja bytów
- **Model specyfikacyjny**
  - klasyfikacja interfejsów (operacji)
- **Model implementacyjny**
  - hierarchia dziedziczenia języka programowania

- Ograniczenia klasyfikacji



domyślnie: incomplete, disjoint

?? Spójność diagramu



## **Atrybuty**

- **Model pojęciowy** – nazwa i nieformalny opis znaczenia
- **Model implementacyjny** – pełna specyfikacja postaci:

widoczność nazwa [ krotność ] : typ = wartość-domyślna { właściwości }

**widoczność** określa dostępność atrybutu w modelu:

- +      dostęp publiczny
- #      dostęp chroniony
- dostęp prywatny

**krotność** określa liczbę wartości danego atrybutu, np.:

- 1              atrybut obowiązkowy
- 0..1          atrybut opcjonalny
- k lub k..n    postać ogólna

**właściwości** definiują dodatkowe cechy atrybutu:

- changeable    atrybut modyfikowalny bez ograniczeń
- addOnly        brak możliwości usuwania (krotność > 1)
- frozen          atrybut stały (ustawiany podczas inicjalizacji)

Wyróżnienia:

- atrybuty klasowe:      podkreślenie

## Operacje

- **Model pojęciowy** – brak lub odpowiedzialności
- **Model implementacyjny** – pełna specyfikacja postaci:  
widoczność nazwa ( lista-argumentów ) : typ { właściwości }

**widoczność** określa dostępność operacji w całym modelu

**lista-argumentów** określa argumenty operacji:

sposób-przekazywania nazwa : typ = wartość-domyślna

### sposób-przekazywania

in przekazanie przez wartość

out przekazanie przez referencję

inout przekazanie przez referencję

**właściwości** definiują dodatkowe cechy operacji:

leaf operacja nie jest polimorficzna

isQuery operacja nie zmienia atrybutów

sequential wymaga sekwencyjnego działania obiektu

guarded wykonywana rozłącznie z innymi

concurrent wykonywana współbieżnie z innymi

### Wyróżnienia:

– operacje klasowe: podkreślenie

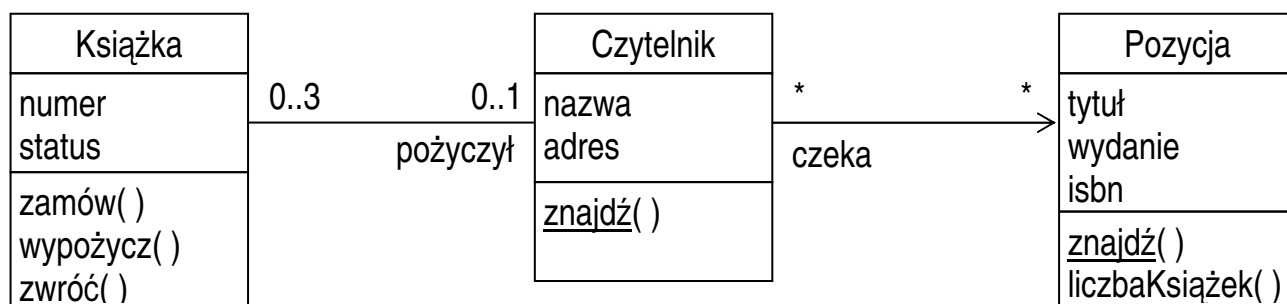
– operacje abstrakcyjne: *kursywa*

## Asocjacja

Trwałe powiązania między obiektami, niekoniecznie typu 1–1

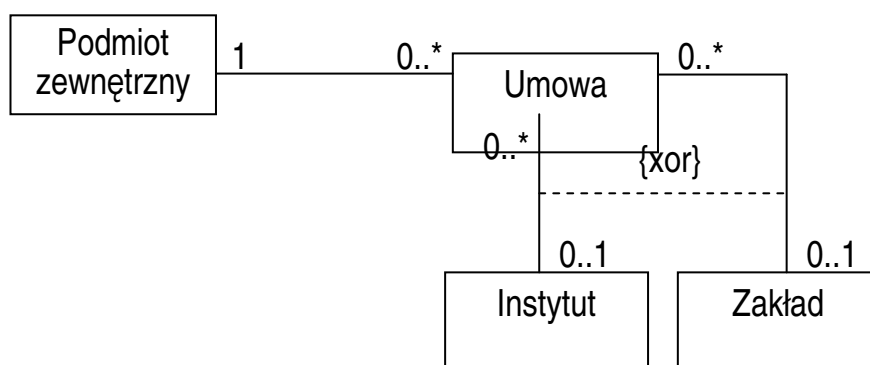
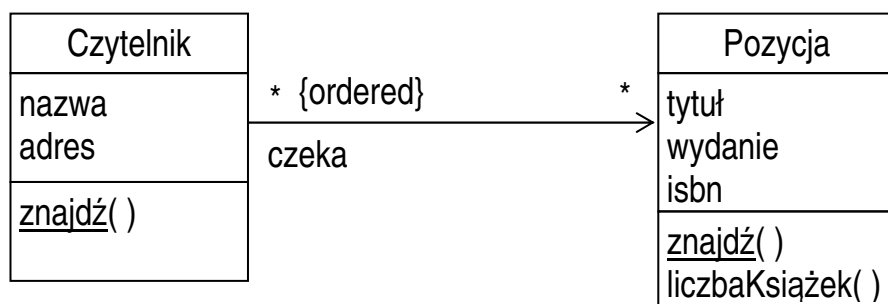
Opis relacji:

- krotność
- kierunek
- etykieta objaśniająca rolę



- **Model pojęciowy**
  - związek pojęciowy między klasami
- **Model specyfikacyjny**
  - operacje nawigowania między obiektami
  - operacje umożliwiające tworzenie relacji
  - nieokreślona realizacja (dane/kod)
- **Model implementacyjny**
  - atrybut wskazujący związany obiekt

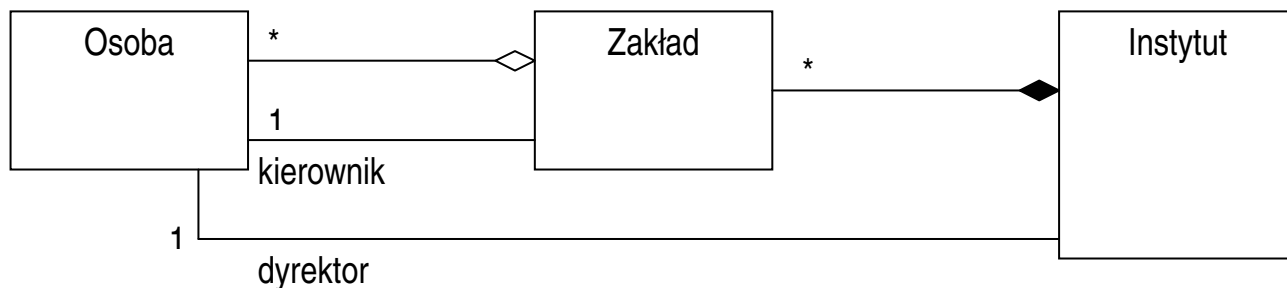
- Ograniczenia asocjacji



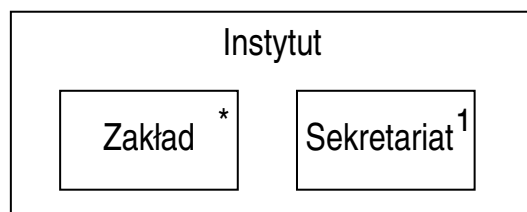


## Agregacja

Szczególny przypadek asocjacji: związek typu całość – część



Złożenie lub zawieranie (*composition*)



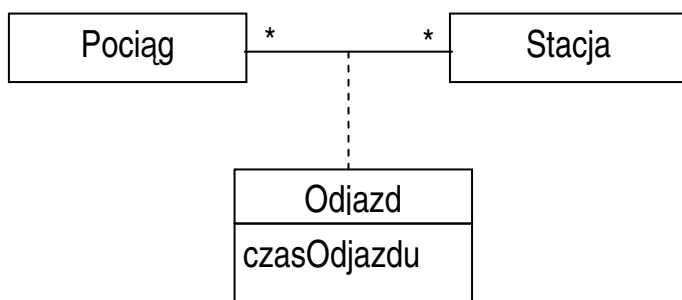
## Klasy asocjacyjne

- Klasa jest zbiorem obiektów
- Obiekt (instancję klasy) charakteryzują wartości atrybutów
- Asocjacja dwóch klas jest relacją  
— zbiorem par obiektów należących do tych klas.
- Asocjacja nie ma atrybutów  
— parę obiektów (instancję asocjacji) charakteryzują atrybuty tych obiektów.

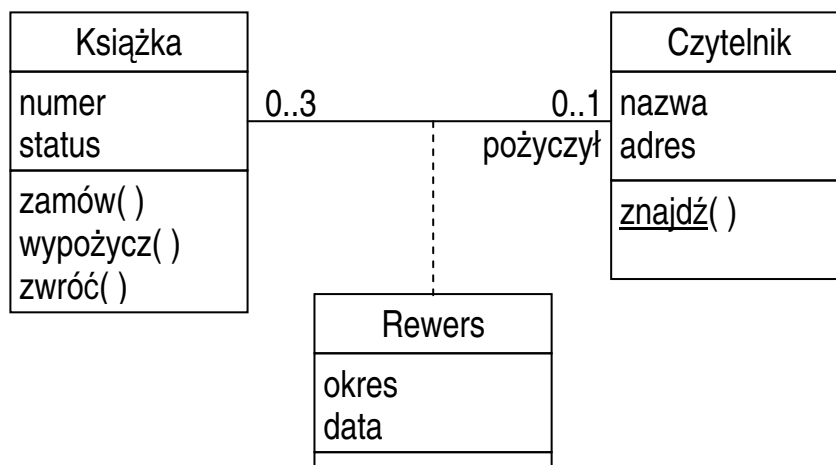
Ostatnia reguła nie opisuje dobrze świata, np.:



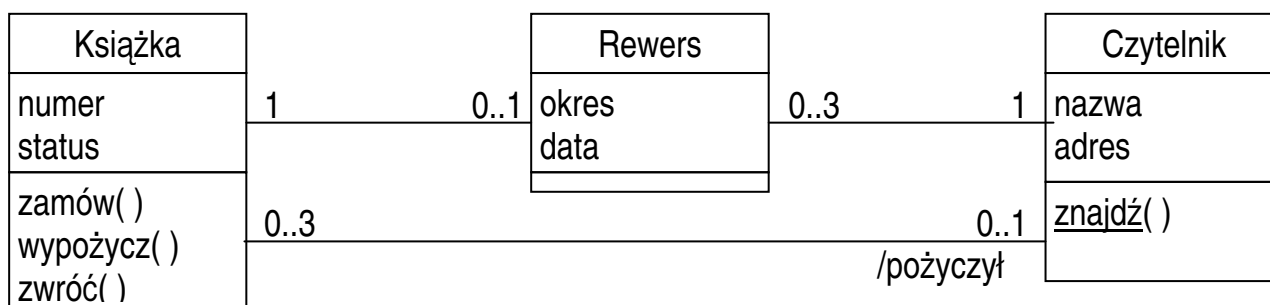
- do kogo należy atrybut: **czasOdjazdu** ?



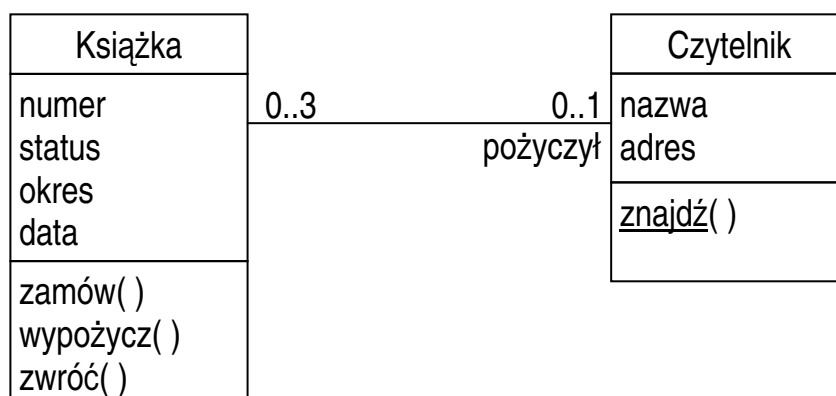
- Klasa asocjacyjna**



- Wariant alternatywny — zwykła klasa.**



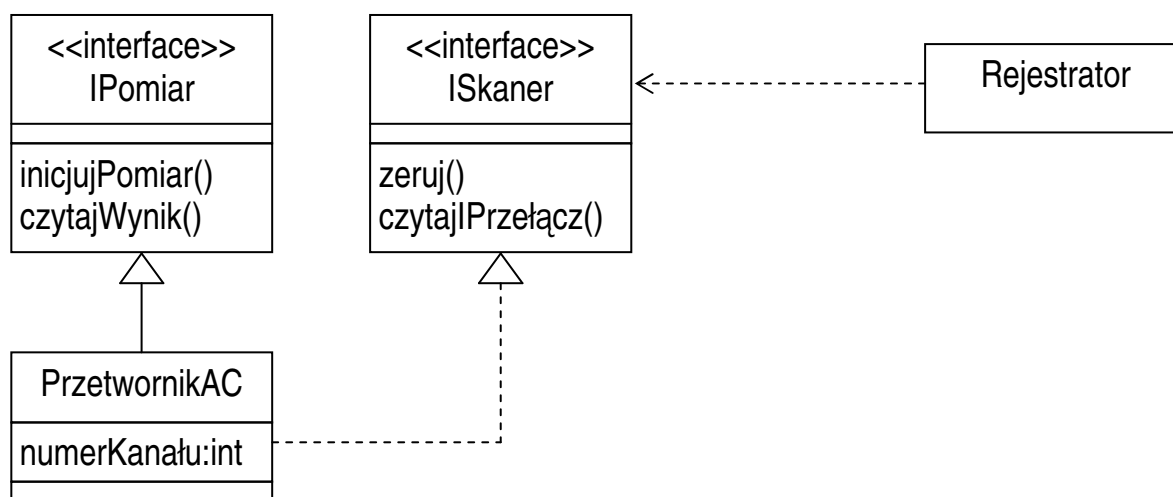
- Trzeci wariant — dodatkowe atrybuty**



## Interfejsy

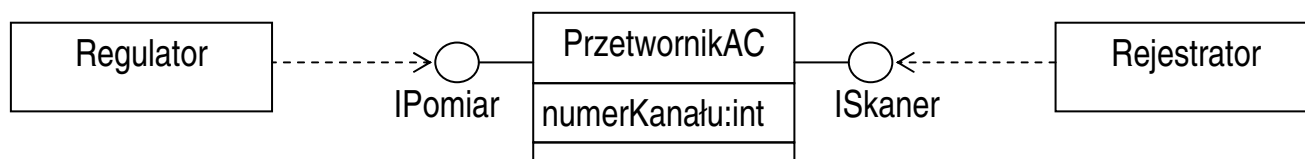
Klasa definiująca operacje udostępniane innym obiektom

- Metod realizujących operacje interfejsu może dostarczyć:
  - klasa pochodna (relacja **generalizacji**)
  - inna klasa lub komponent systemu (relacja **realizacji**)



- Klasa używająca interfejsu jest z nim w relacji **zależności**
  - zmiana interfejsu może wymagać zmiany w kodzie klienta
  - zmiana metody wykonania operacji nie wpływa na klienta

Notacja alternatywna (bez pokazania operacji):



### Pojęcia podobne

- klasa abstrakcyjna {abstract}
- typ <<type>>

## **Zastosowanie diagramów klas**

### **1. Model dziedziny (model pojęciowy)**

- klasy z dziedziny problemu
- nieformalny opis odpowiedzialności

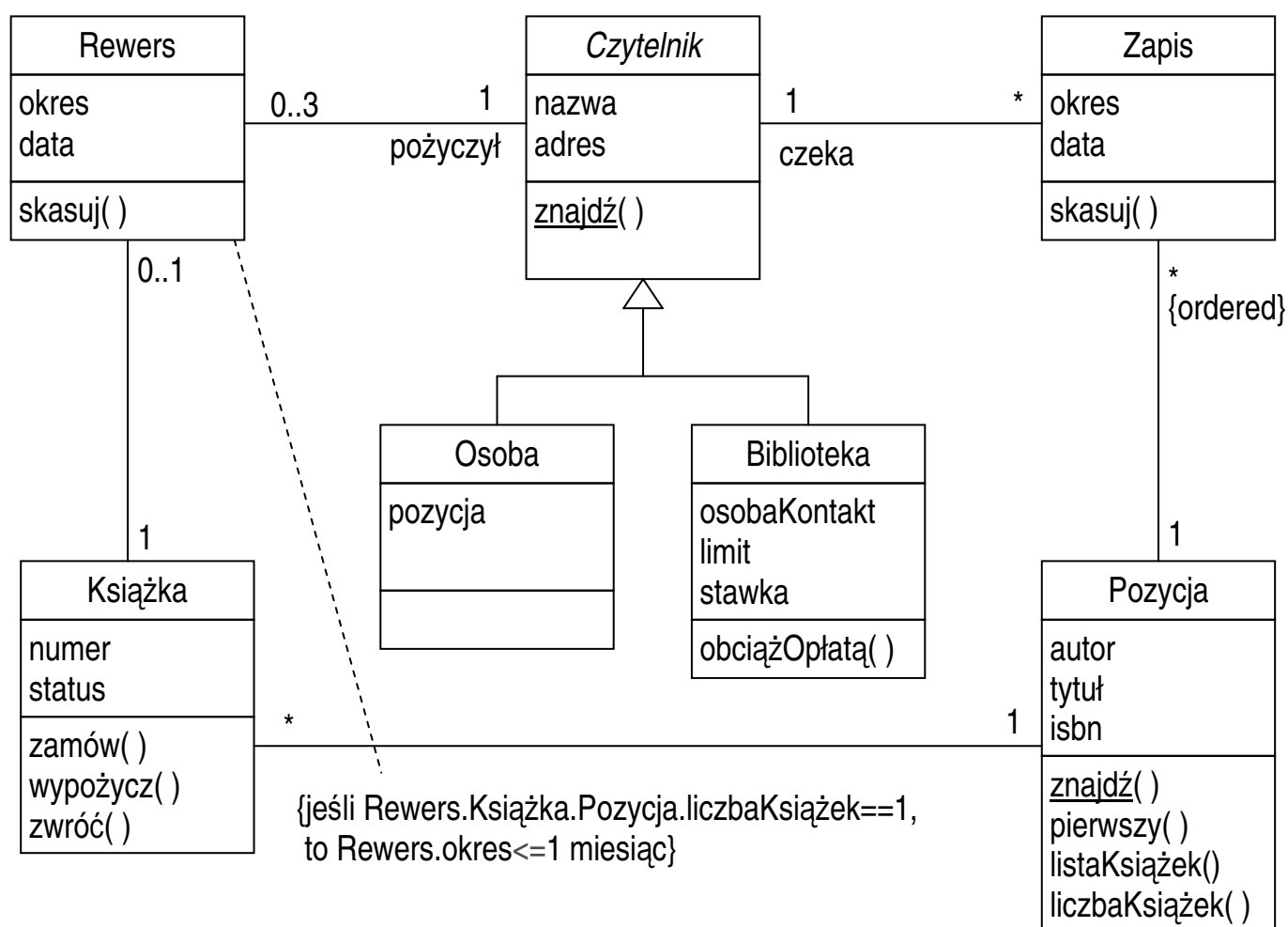
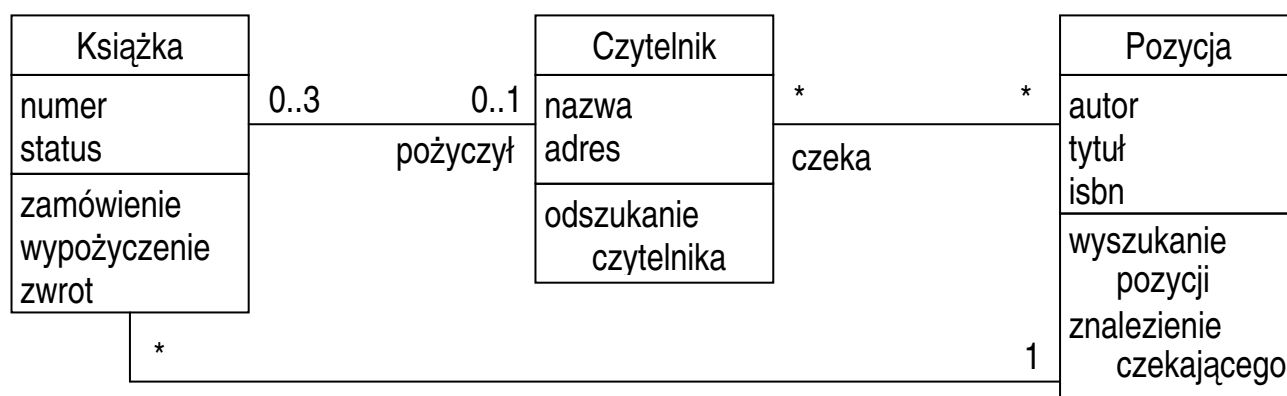
### **2. Model analizy**

- uzupełnienie opisu klas (operacje)
- uzupełnienie hierarchii klas
- dodawanie klas pomocniczych (np. klasy asocjacyjne)
- łączenie modeli różnych obszarów tematycznych
- budowa modeli zachowania

### **3. Model projektowy i implementacyjny**

- określenie komponentów i ich interfejsów
- określenie klas implementujących operacje
- uporządkowanie hierarchii dziedziczenia
- określenie klas implementujących operacje

## Przykład: system biblioteczny



## Dodatkowe elementy opisu klas

- stereotypy

<<interface>>	tylko operacje (brak atrybutów i metod)
<<type>>	tylko atrybuty i operacje (brak metod)
<<business>>	klasa bierna, nie inicjuje działań

- metki

{ abstrakt }	klasa abstrakcyjna (też <i>kursywa</i> )
{ root }	klasa najwyższa w hierarchii generalizacji
{ leaf }	klasa najniższa w hierarchii generalizacji
{ persistent }	klasa reprezentująca obiekty bazy danych

## Diagram obiektów

Diagram klas — ogólne związki między obiektami

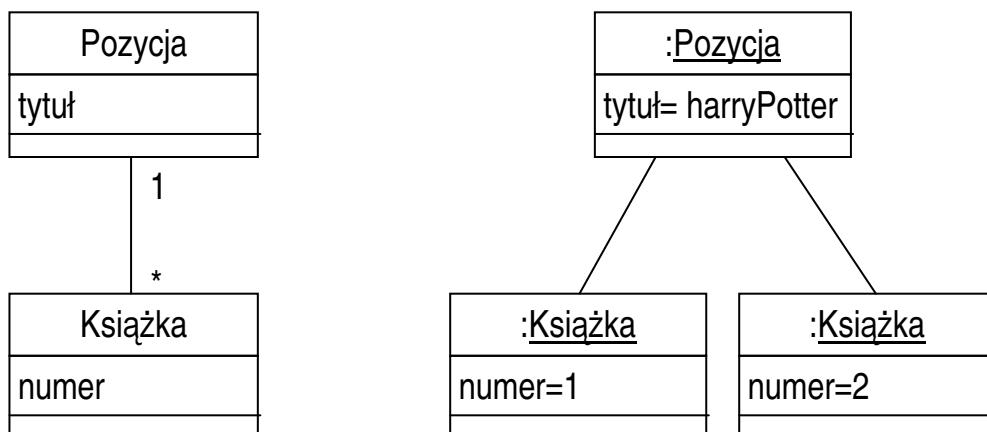
Diagram obiektów — chwilowa konfiguracja obiektów

- Notacja

- ta sama symbolika, prostokąty reprezentują obiekty
- nazwy obiektów obiekt:klasa

- Użyteczność

- w prostych przypadkach niewielka



- w złożonych modelach rekurencyjnych duża



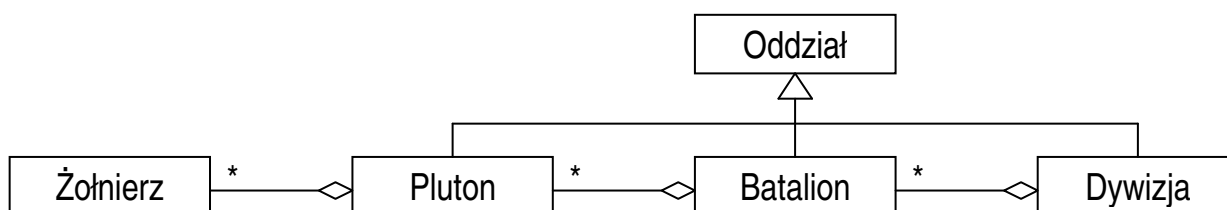
## Przykład: gra komputerowa

- żołnierze, o zdolności bojowej charakteryzowanej przez uzbrojenie i wydajność (0...1),
- oddziały, których zdolność bojowa jest sumą zdolności bojowej żołnierzy.

System zna zdolność bojową żołnierzy i oddziałów i oferuje jednolite metody posługiwania się żołnierzami i oddziałami

### I wariant: prosty diagram klas

Ogólna klasa Oddział i agregacja oddziałów i żołnierzy



### Wady:

- sztywna hierarchia oddziałów (reorganizacja po bitwie)
- brak jednolitego interfejsu oddziałów i żołnierzy

## II wariant: wzorzec projektowy *Composite*

- wspólna nadklasa dla oddziałów i żołnierzy (Jednostka)
- model rekurencyjny

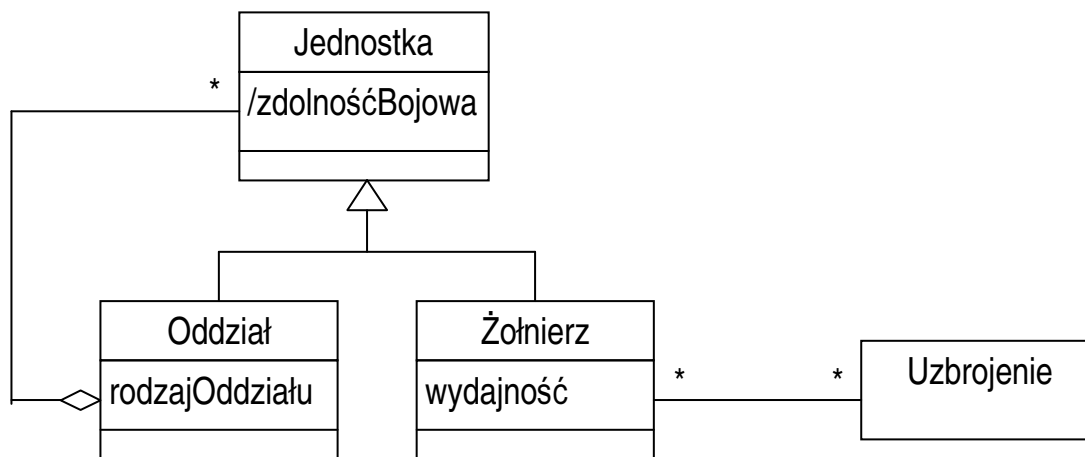
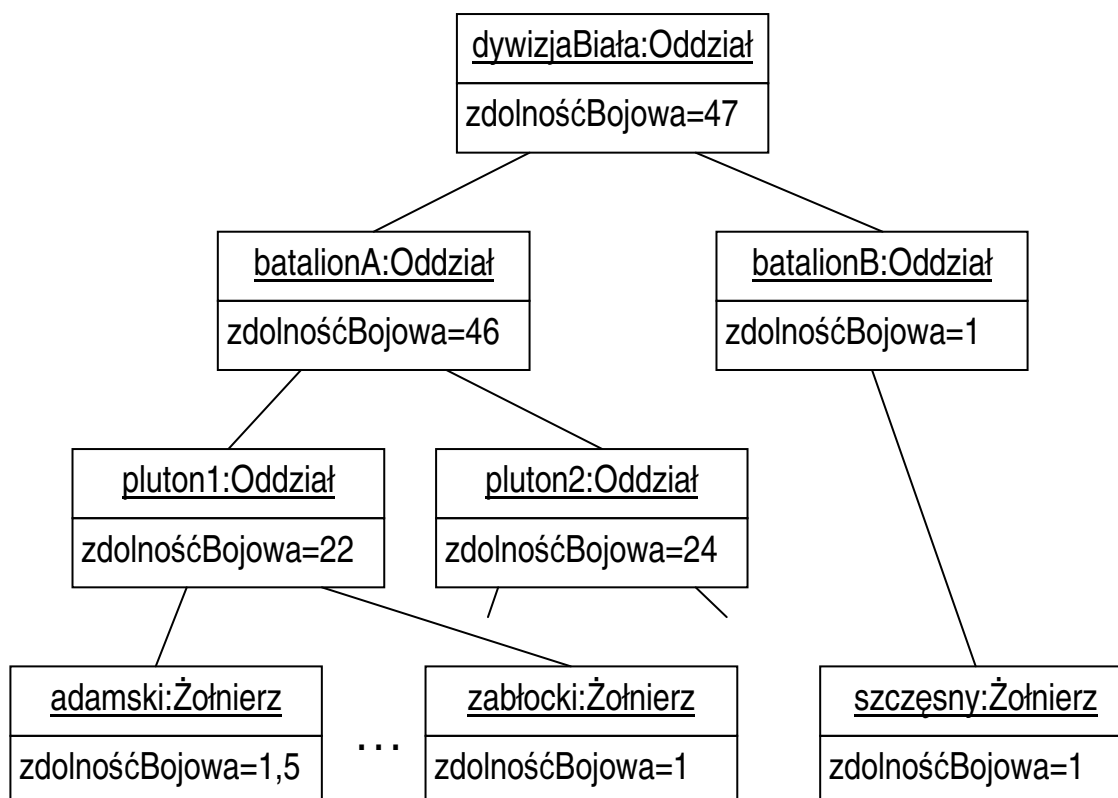


Diagram obiektów pokazuje chwilową konfigurację



→ Konfiguracja obiektów a reprezentacja to nie jest to samo!