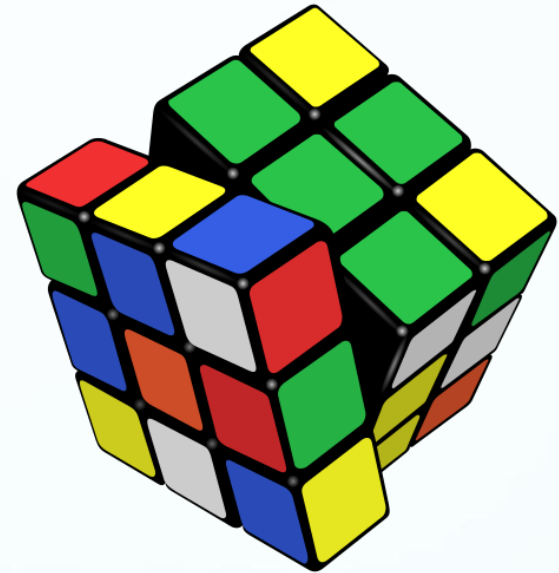


Analiza i Projektowanie Obiektowe w UML

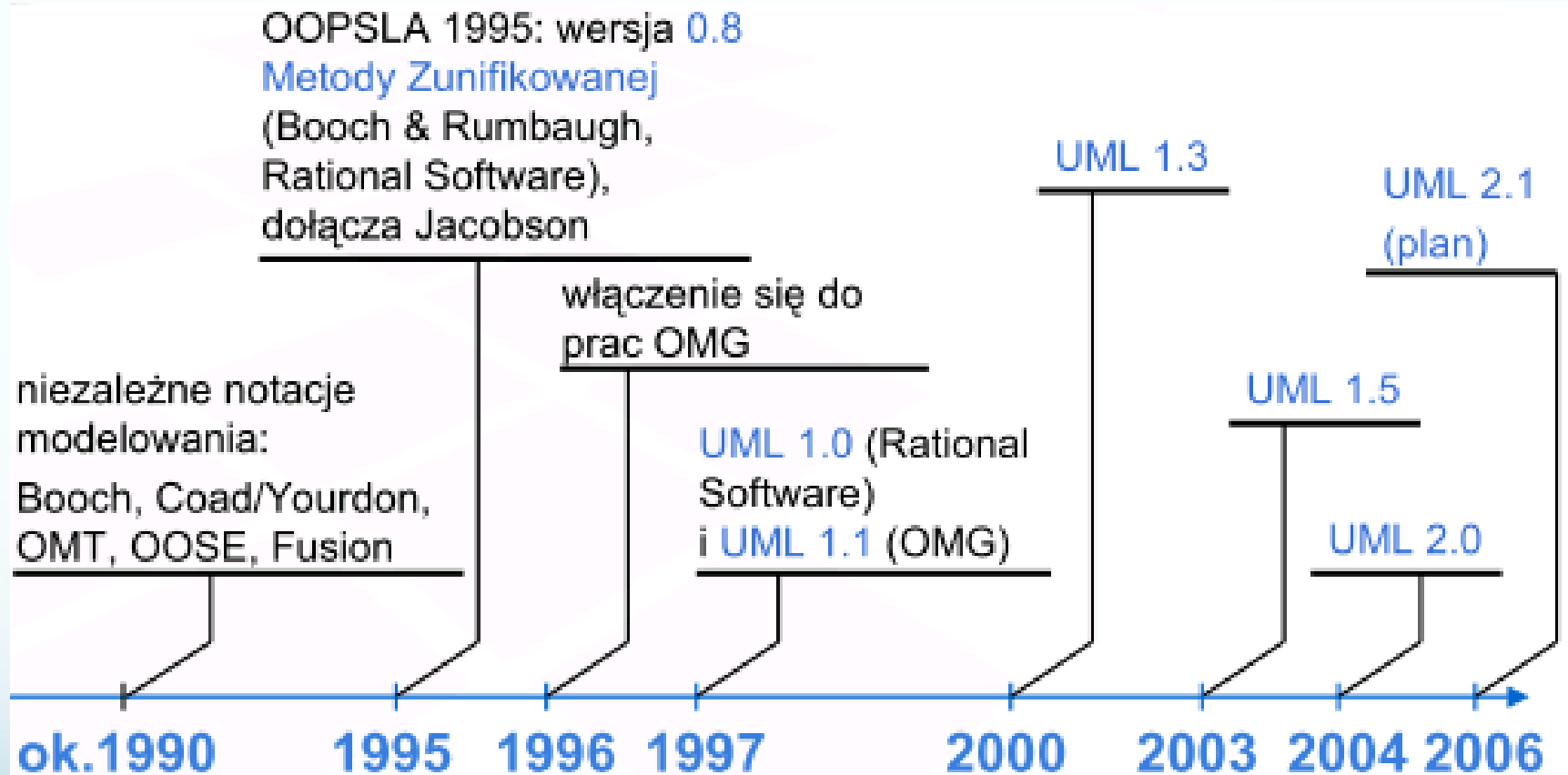


Język modelowania UML

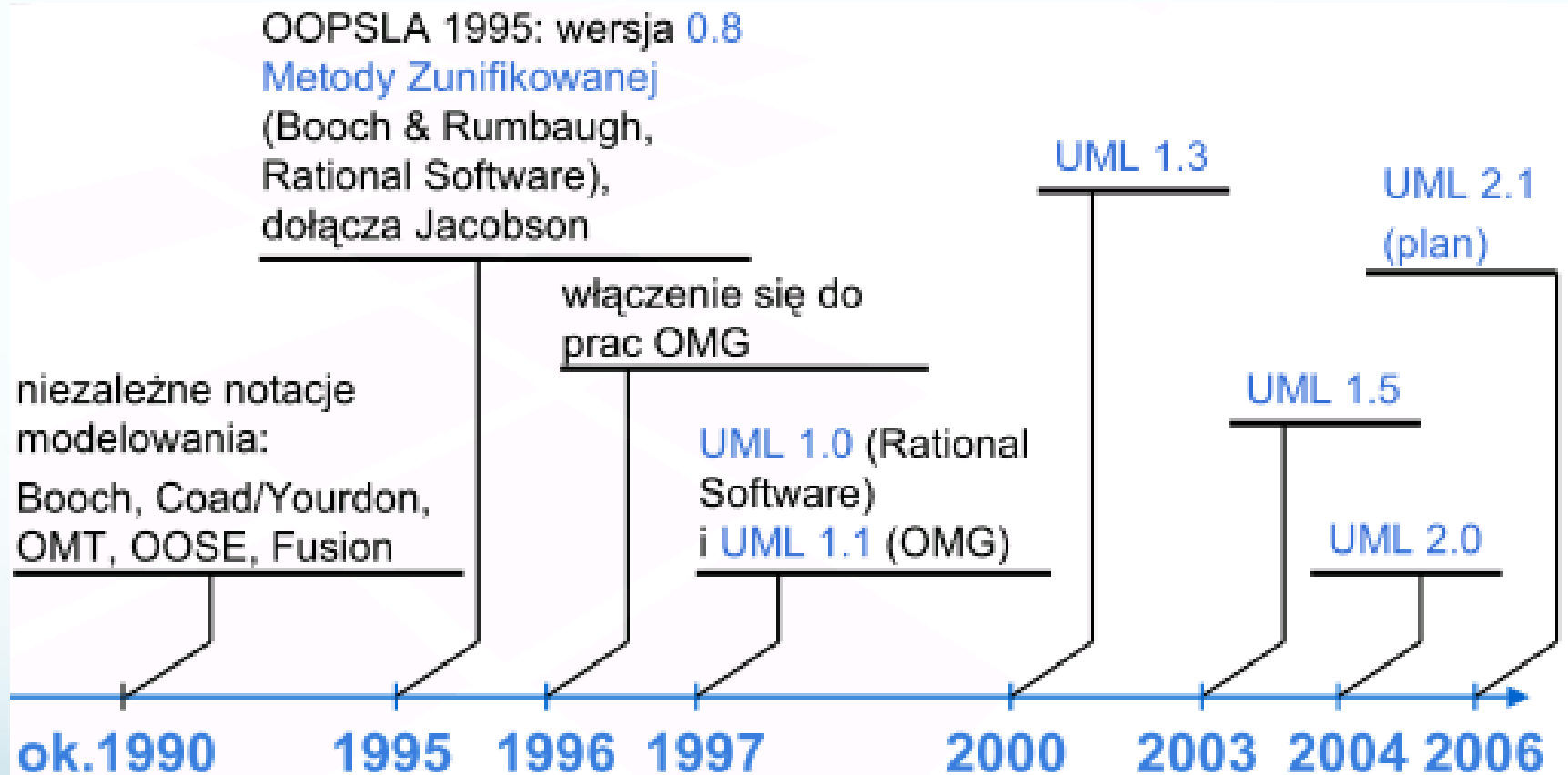
Agenda

- Historia i geneza UML
- Koncepcja modeli UML
- Diagramu przypadków użycia (wprowadzenie)
- Diagram klas
- Diagram obiektów
- Diagram struktur złożonych
- Podsumowanie

Historia UML



Historia UML



Trzej autorzy UML



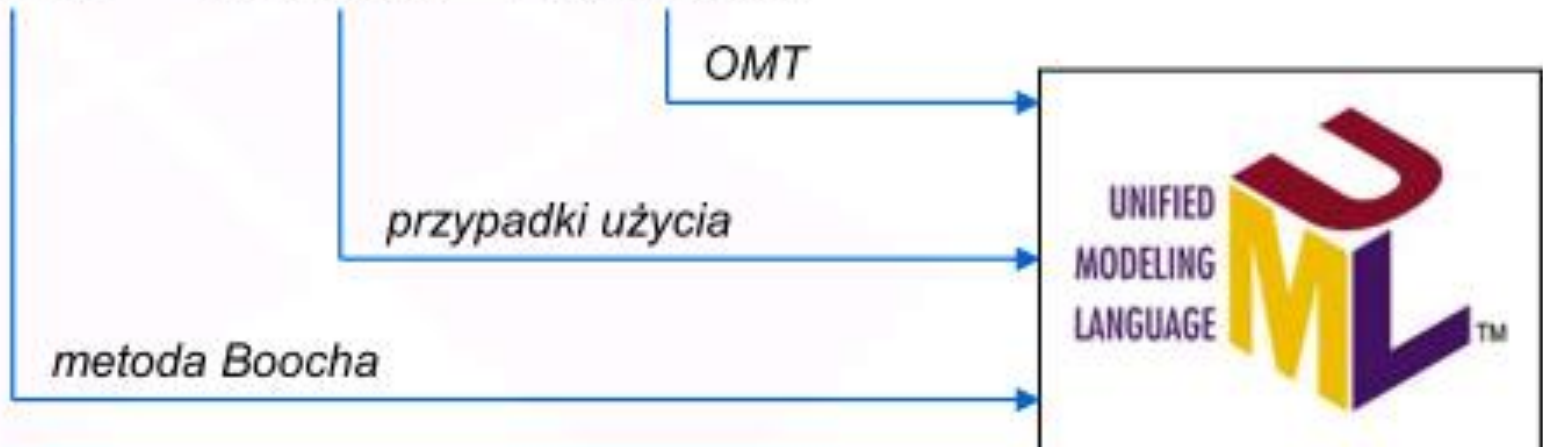
G. Booch



I. Jacobson



J. Rumbaugh



Czym jest UML?

metodyką

- UML nie określa metody modelowania
- zaleca jedynie stosowanie podejścia przyrostowego

narzędziem

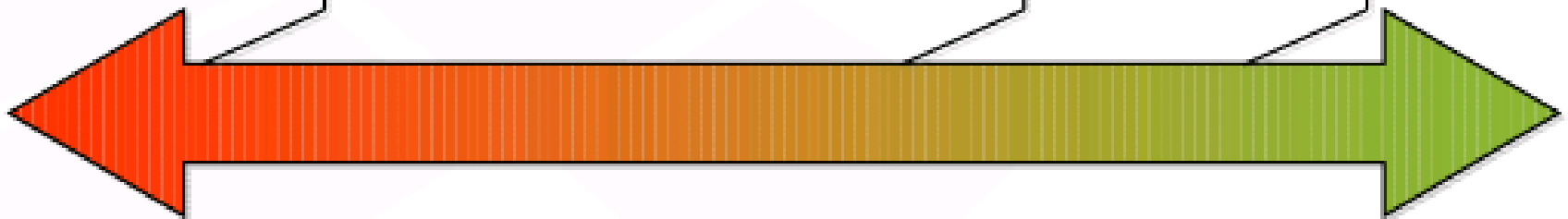
- UML to specyfikacja dla narzędzi

notacją graficzną

- UML określa sposób zapisu modeli

językiem programowania

- generowanie kodu z modelu stosowane obecnie na niewielką skalę



UML nie jest

UML jest

Konstrukcja UML

UML składa się z dwóch podstawowych elementów:

notacja

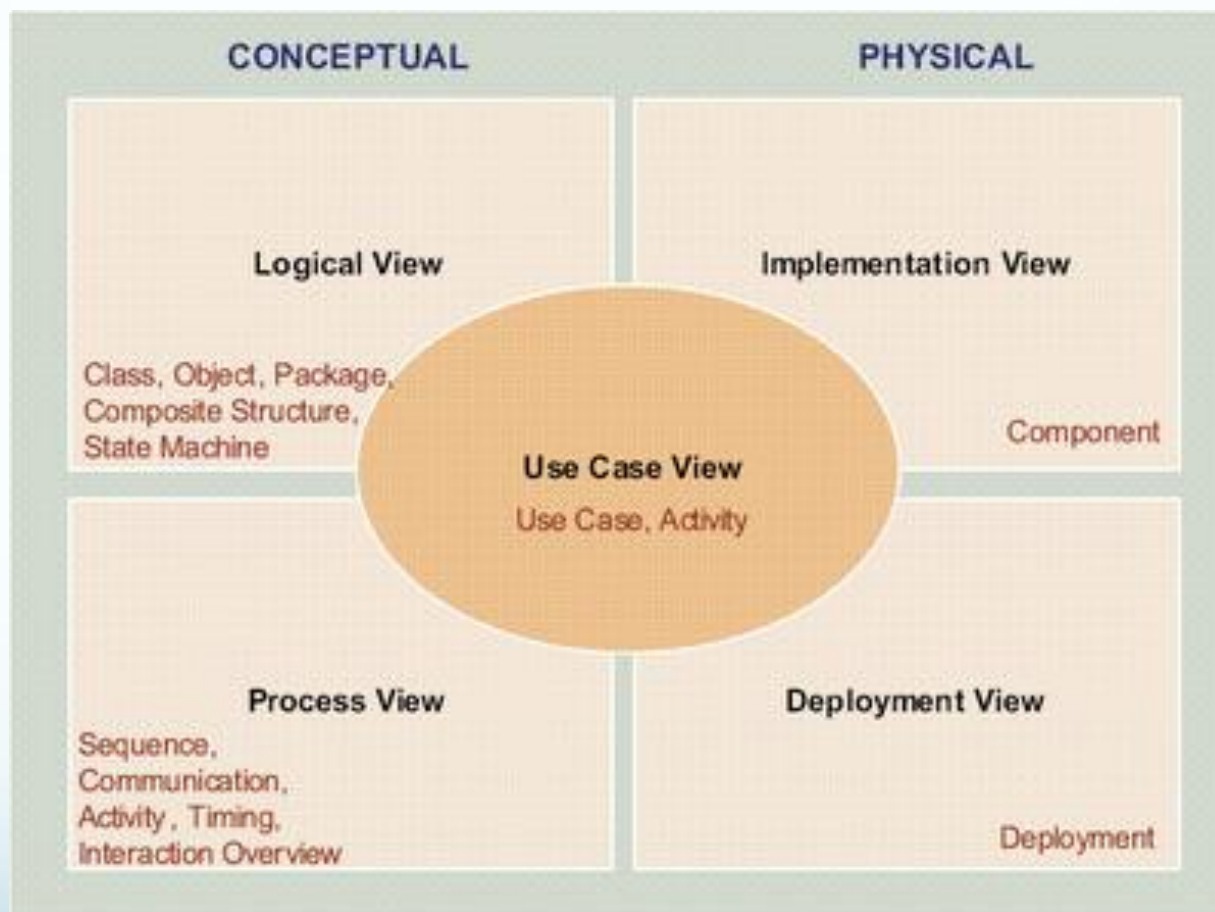
- elementy graficzne
- składnia języka modelowania
- istotna przy szkicowaniu modeli

metamodel

- definicje pojęć języka i powiązania pomiędzy nimi
- istotny przy graficznym programowaniu

- Z punktu widzenia modelowania ważniejsza jest notacja.
- Z punktu widzenia generacji kodu – metamodel.

Perspektywy 4 + 1



Diagramy UML

UML (2.4.1) obejmuje 14 rodzajów diagramów

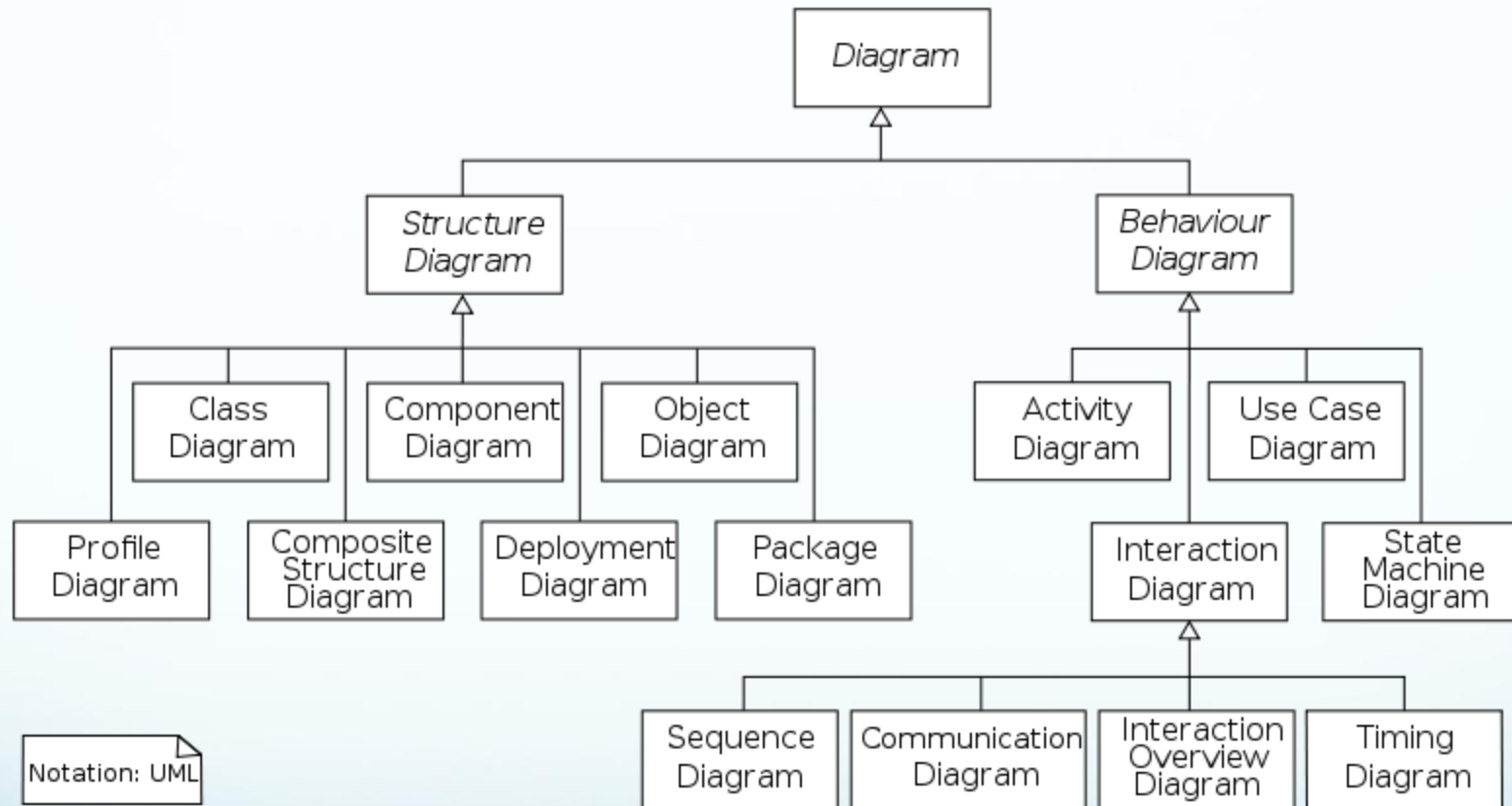
Modelowanie strukturalne

- diagram klas (ang. class diagram)
- diagram obiektów (ang. object diagram)
- diagram komponentów (ang. component diagram)
- diagram wdrożenia (ang. deployment diagram)
- diagram struktur złożonych (ang. composite structure diagram, UML 2.0)
- diagram pakietów (ang. package diagram, UML 2.0)
- diagram profili (ang. profile diagram, UML 2.2)

Modelowanie behawioralne

- diagram czynności (ang. activity diagram)
- diagram przypadków użycia (ang. use case diagram)
- diagram maszyny stanów (ang. state machine diagram)
- diagram interakcji (diagram abstrakcyjny)
 - diagram komunikacji (ang. communication diagram)
 - diagram sekwencji (ang. sequence diagram)
 - diagramy czasowe (ang. timing diagram)
 - diagram przeglądu interakcji (ang. interaction overview diagram)

Diagramy UML

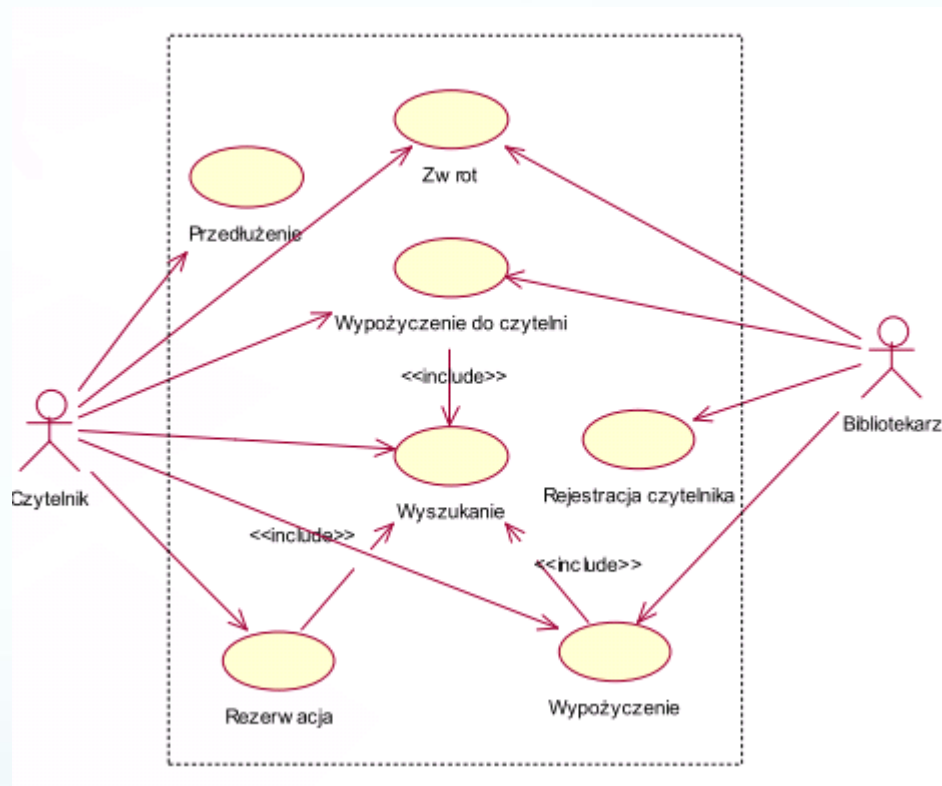


From Wikipedia, the free encyclopedia

Diagram przypadków użycia

Diagram przypadków użycia

- definiuje **granice** modelowanego systemu
- określa jego **kontekst**
- wymienia **użytkowników** systemu i jednostki zewnętrzne
- przedstawia **funkcje** dostępne dla użytkowników
- określa **powiązania i zależności** pomiędzy nimi



Aktor

Aktor

- inicjuje wykonanie funkcji systemu
- wymaga dostępu do systemu
- reprezentuje punkt widzenia na system
- jest osobą fizyczną, rolą w systemie lub systemem zewnętrznym



Bibliotekarz



Czytelnik



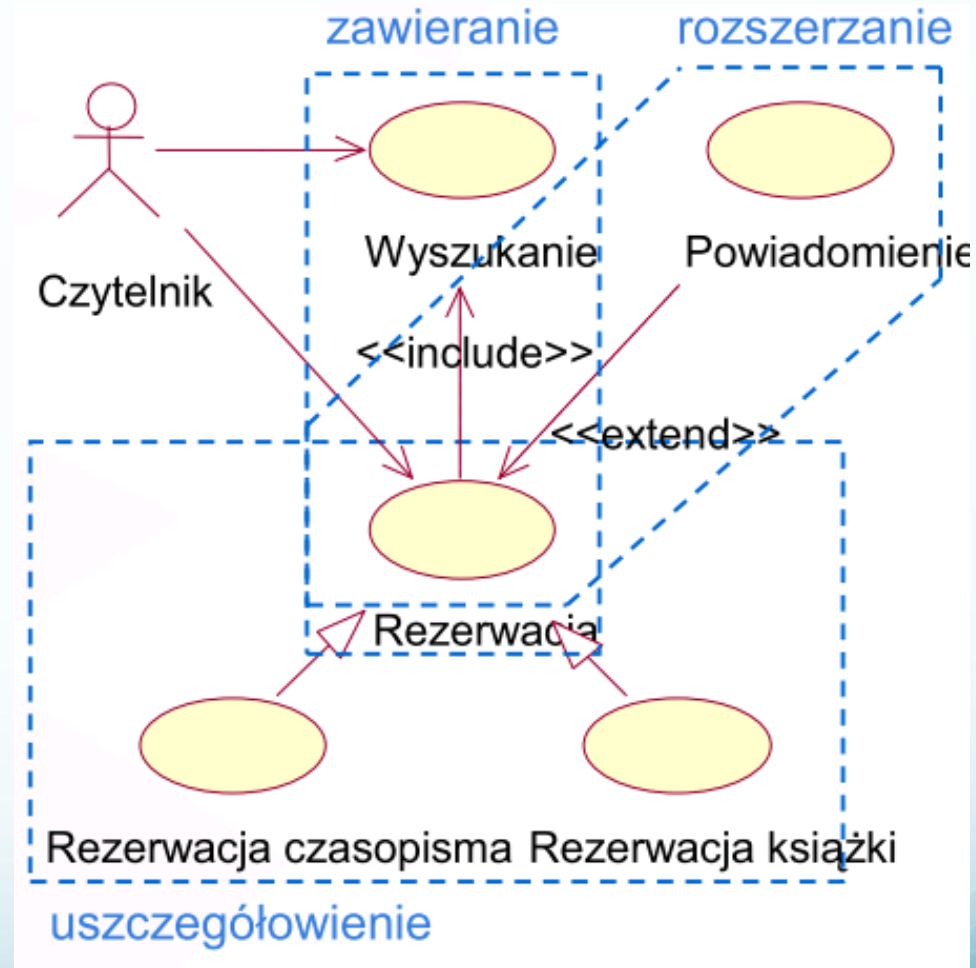
Zegar

Przypadek użycia

Przypadek użycia reprezentuje kompletną funkcję dostępną dla aktora.

Przypadki użycia mogą być powiązane zależnościami:

- **Uszczegółowienie:** specjalizowana wersja przypadku użycia
- **Rozszerzanie:** dodatkowa funkcjonalność przypadku użycia
- **Zawieranie:** wykonanie jednego przypadku użycia przez drugi



Przykład

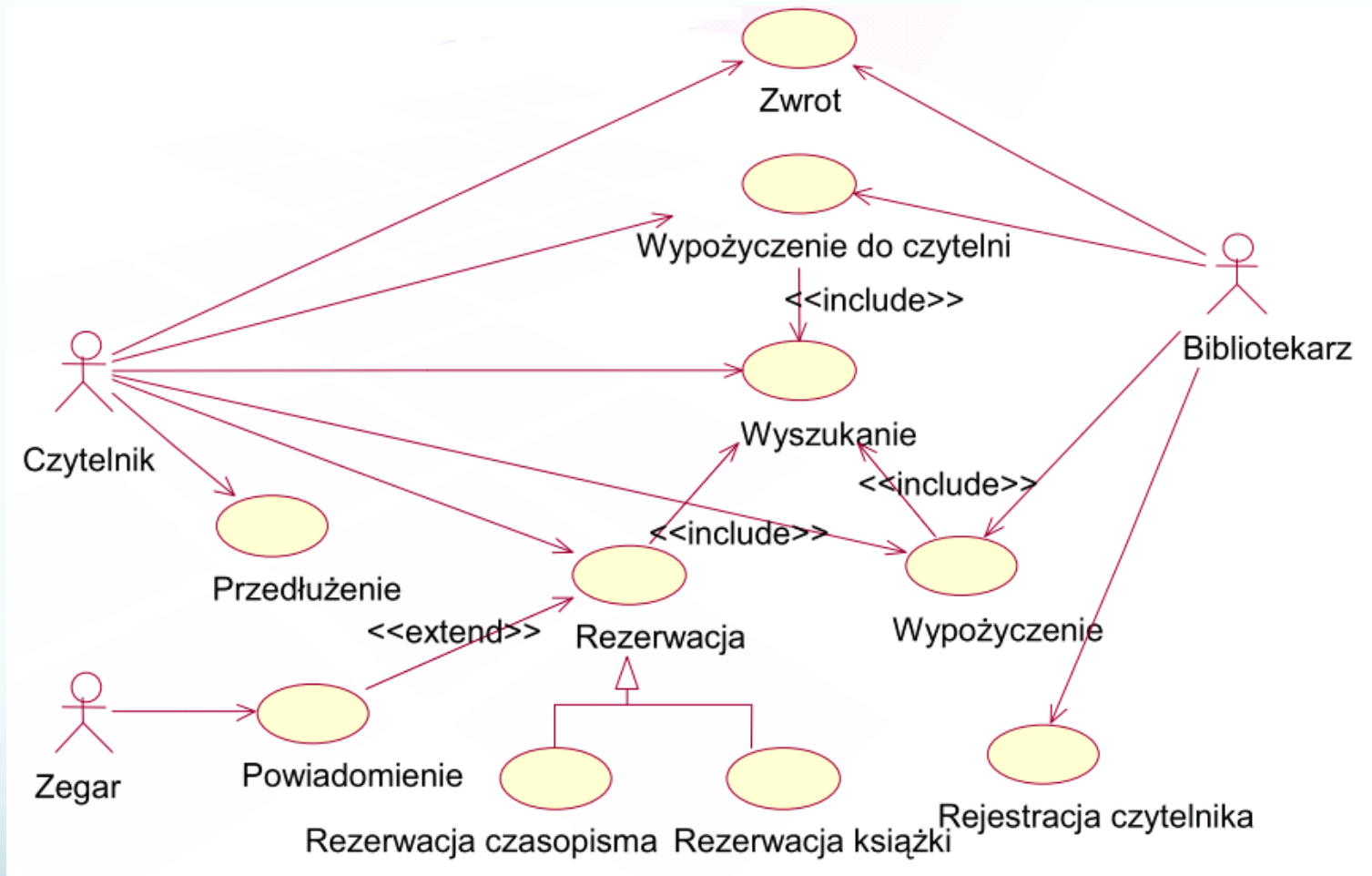
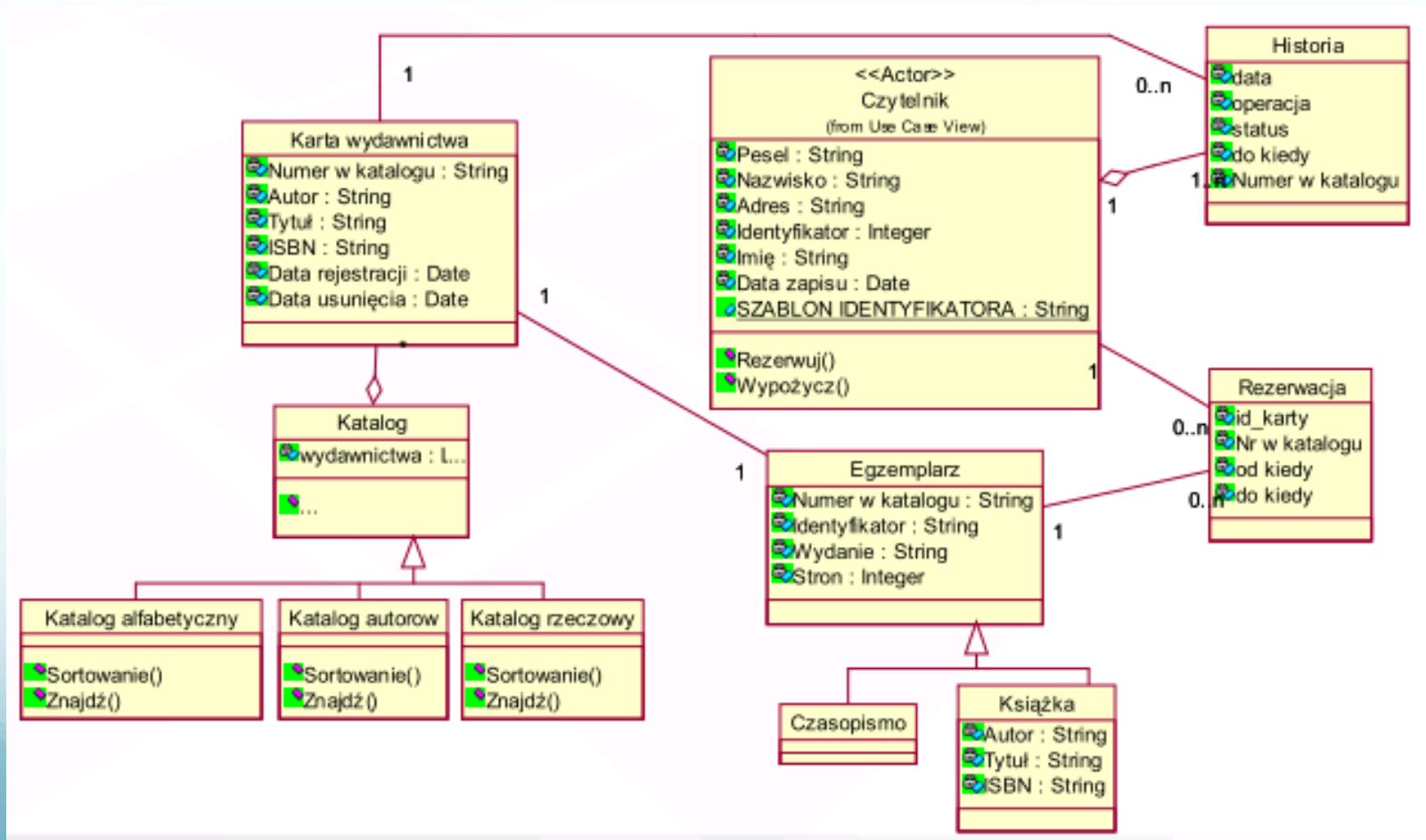
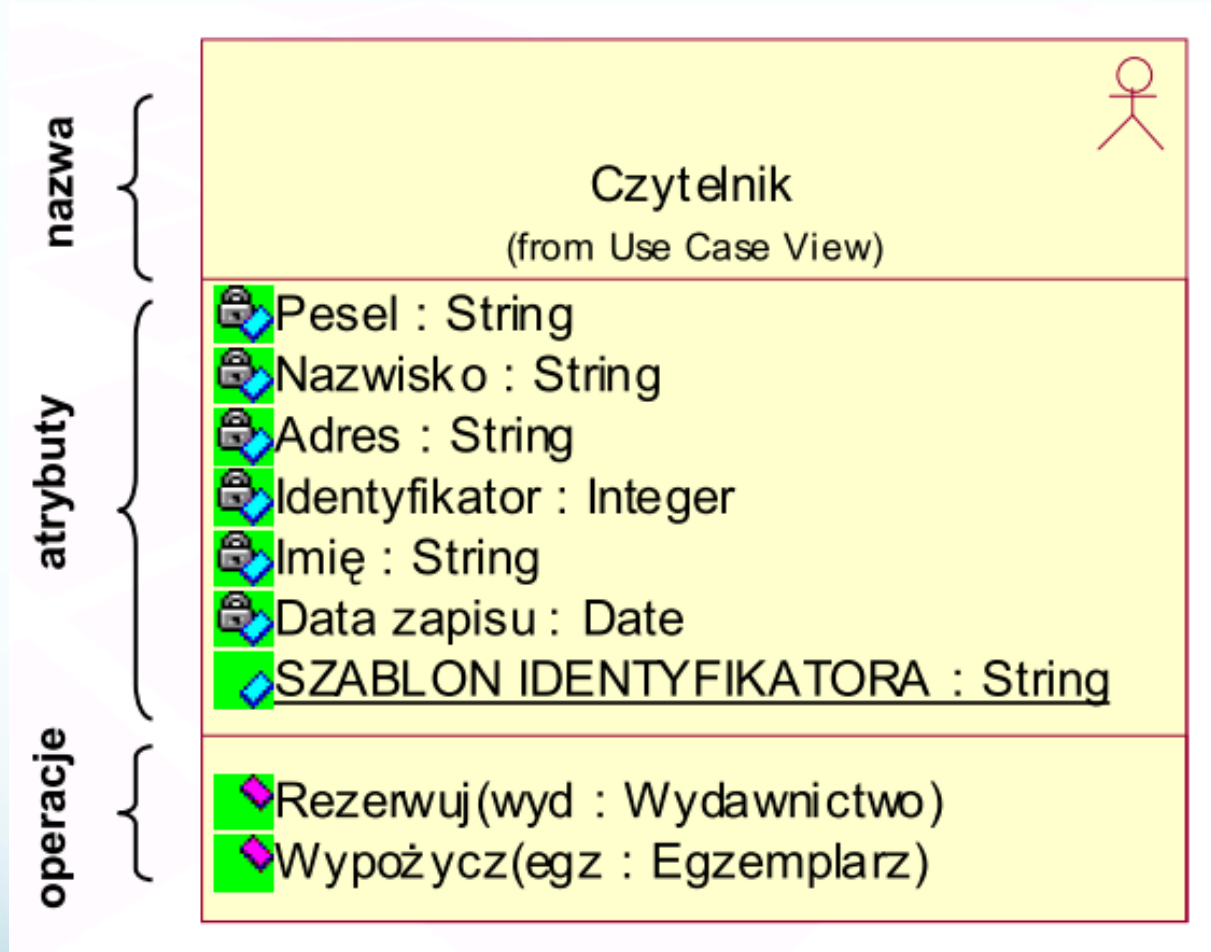


Diagram klas

- Diagram klas przedstawia klasy występujące w systemie i statyczne relacje pomiędzy nimi wraz z ograniczeniami. Jest podstawowym diagramem struktury logicznej systemu.



Klasa i obiekt



Atrybuty klasy

- Cechy klasy są zapisywane w postaci atrybutów klasy lub asocjacji z innymi klasami. Atrybuty reprezentują wartości proste lub niewielkie obiekty, asocjacje – obiekty złożone

widoczność nazwa : typ[krotność] {ograniczenia} = wartość dom.

Skąd atrybut jest widoczny?

Ile obiektów trzeba i ile można umieścić w atrybucie?

Jakie dodatkowe warunki spełniają wartości atrybutu?

Jaka jest wartość atrybutu, gdy nie podano jej wprost?

Poziom widoczności

UML definiuje **4 poziomy widoczności** cech i metod

- **+ publiczny** – element jest widoczny z każdego miejsca w systemie
- **# chroniony** – element jest widoczny we własnej klasie i jej podklasach
- **– prywatny** – element jest widoczny tylko we własnej klasie
- **~ publiczny wewnątrz pakietu** – element jest widoczny tylko wewnątrz własnego pakietu

Krotność

Krotność pozwala określić minimalną i maksymalną liczbę obiektów, jakie można powiązać z daną cechą:

- **dolna granica..górna granica** – przedział od-
do
- **1** – dokładnie jeden obiekt
- **0..1** – opcjonalnie jeden obiekt
- **1..*** – przynajmniej jeden obiekt
- **1, 3, 5** – konkretne liczby obiektów
- ***** – dowolna liczba obiektów






Właściwości i ograniczenia atrybutów

Z atrybutem mogą być związane dodatkowe ograniczenia, które określają jego właściwości, np.:

- **{ordered}** – obiekty wewnątrz cechy są uporządkowane
- **{unordered}** – obiekty są nieuporządkowane
- **{unique}** – obiekty wewnątrz cechy nie powtarzają się
- **{nonunique}** – obiekty wewnątrz cechy mogą się powtarzać
- **{readOnly}** – wartość atrybutu służy tylko do odczytu
- **{frozen}** – wartość atrybutu nie może być zmodyfikowana po jej przypisaniu

Atrybuty pochodne

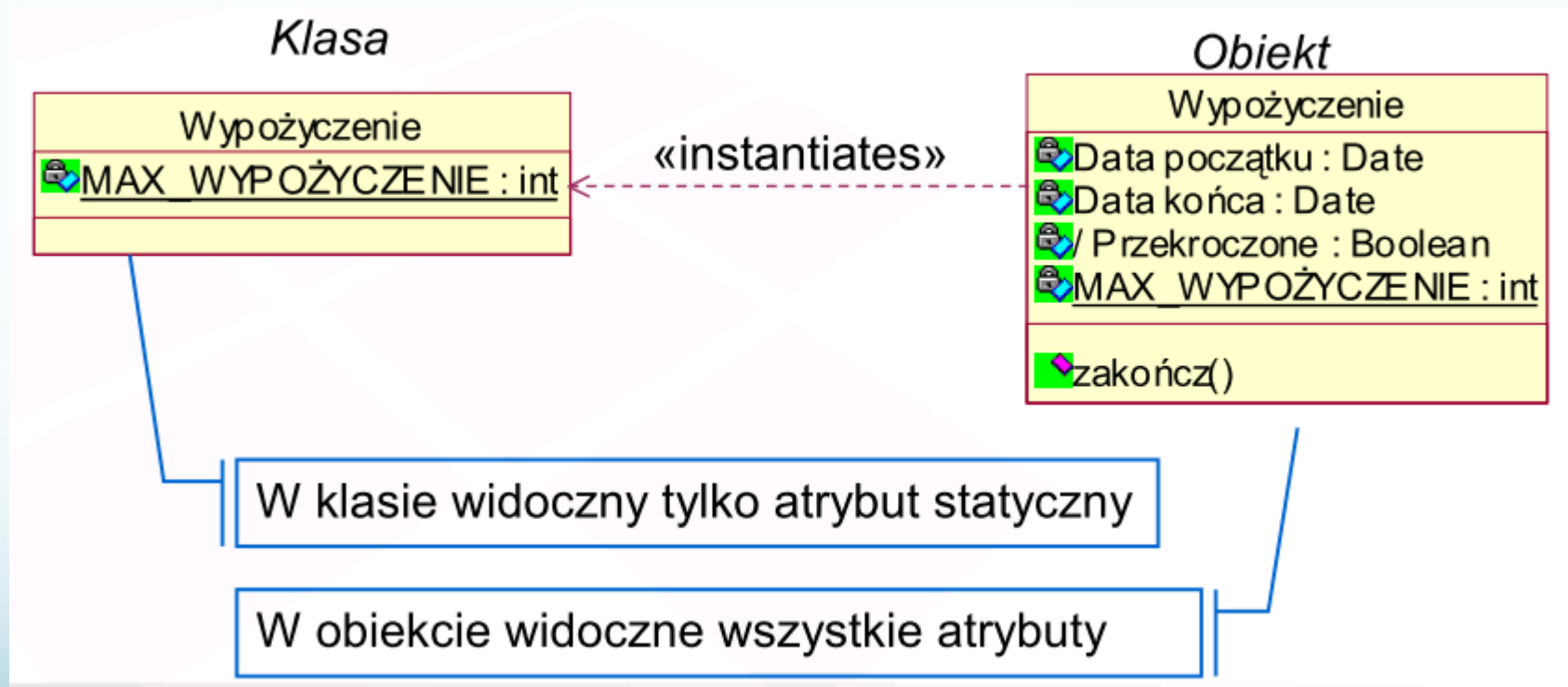
Atrybut pochodny (wywiedziony) może zostać obliczony na podstawie innych atrybutów. Atrybutów pochodnych nie trzeba implementować.

Wypożyczenie	
	Data początku : Date
	Data końca : Date
	/ Przekroczone : Boolean
	<u>MAX_WYPOŻYCZENIE : int</u>
	zakończ()

$$\text{przekroczone} = (\text{Data końca.Dni()} - \text{Data początku.Dni()}) > \text{MAX_WYPOŻYCZENIE}$$

Składowe statyczne

- **Składowe (atrybuty i operacje) statyczne** są widoczne także wewnątrz klasy, nie tylko wewnątrz obiektów.



Operacje

- **Operacja** to proces, który klasa potrafi wykonać.

widoczność nazwa(parametr1, parametr2,...) : typ {ograniczenia}

kierunek nazwa typ[krotność] = wartość dom.
kierunki parametrów:

- in: wejściowy
- out: wyjściowy
- inout: wejściowo-wyjściowy
- return: zwracany z metody

Właściwości i ograniczenia operacji

- Operacje, podobnie jak atrybuty, mogą posiadać **dodatkowe właściwości i ograniczenia**:
- **{query}** – operacja nie modyfikuje stanu obiektu – jest zapytaniem
- **«exception»** – metoda może zgłaszać wyjątek

Warunki wstępne i końcowe

Warunki wstępne

- opisują stan systemu wymagany przed wykonaniem operacji

Warunki końcowe

- gwarancje dotyczące stanu systemu po wykonaniu operacji

wyszukaj(tytuł: String) : Wydawnictwo[]

pre:

tytuł != null

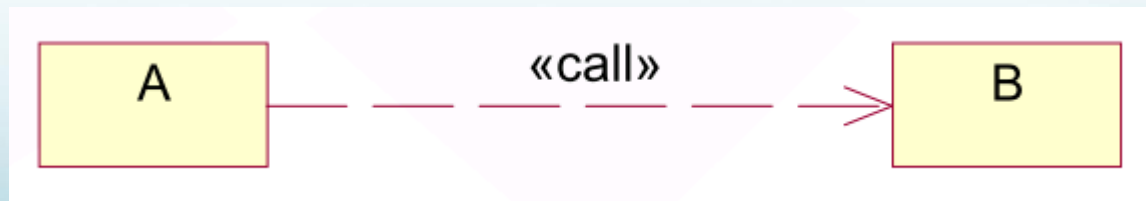
post:

wynik != null

wynik *instanceof* Wydawnictwo[]

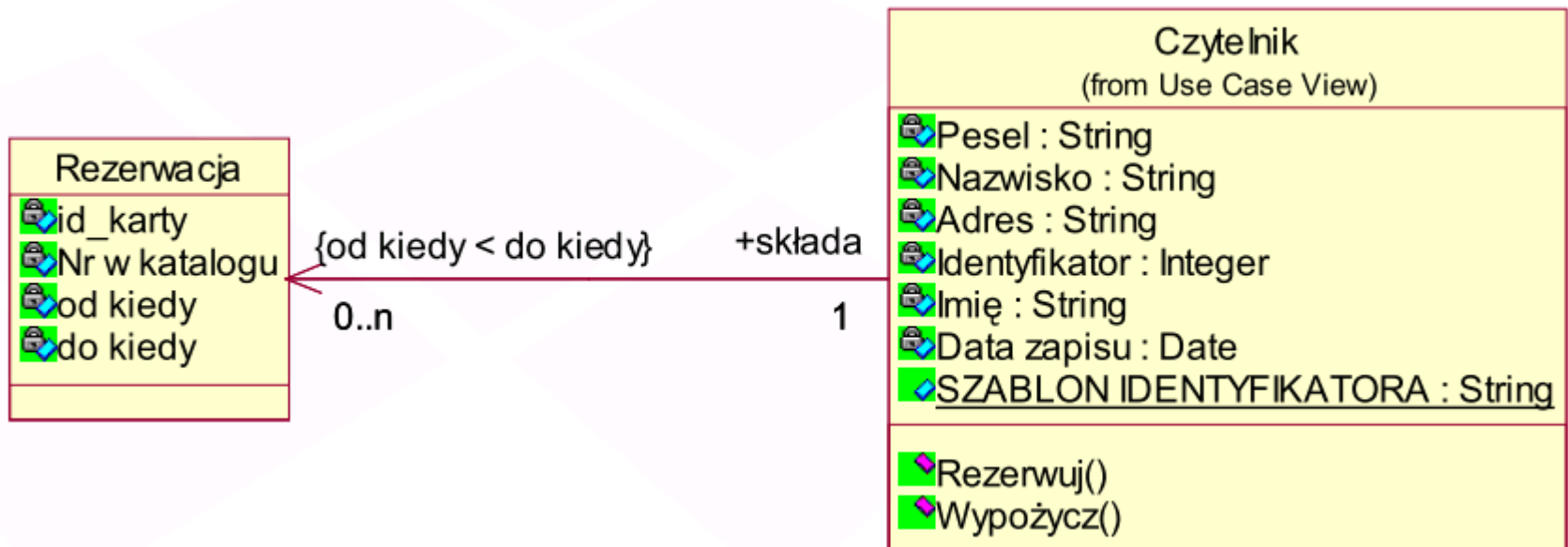
Zależność

- **Zależności** są najprostszym i najstabszym rodzajem relacji łączących klasy. Oznaczają, że zmiana jednej z nich w pewien sposób wpływa na drugą, np.
- **«call»** - operacje w klasie A wywołują operacje w klasie B
- **«create»** - klasa A tworzy instancje klasy B
- **«instantiate»** - obiekt A jest instancją klasy B
- **«use»** - do zaimplementowania klasy A wymagana jest klasa B



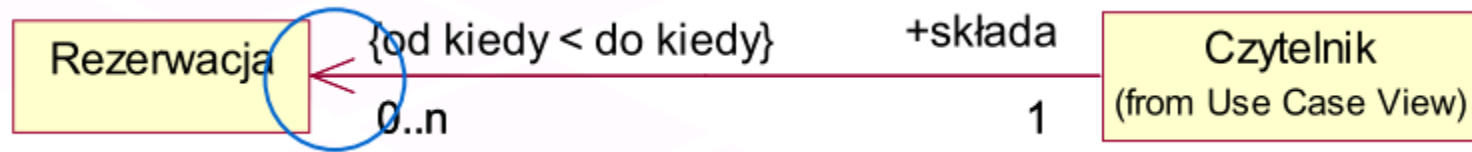
Asocjacja

- **Asocjacja** reprezentuje czasowe powiązanie pomiędzy obiektami dwóch klas. Obiekty związane asocjacją są od siebie niezależne.
- Asocjacja jest też używana jako alternatywny (obok atrybutu) i równorzędny sposób zapisu cech klasy.

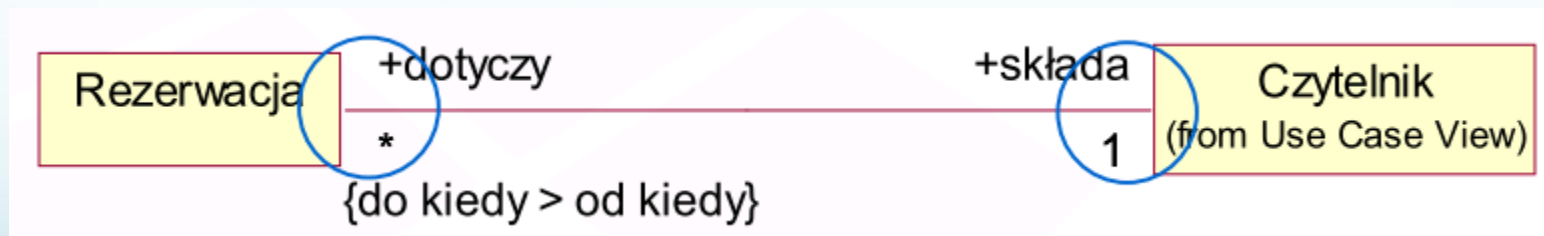


Nawigowalność asocjacji

- **Nawigowalność** określa wiedzę o sobie nawzajem obiektów uczestniczących w relacji.



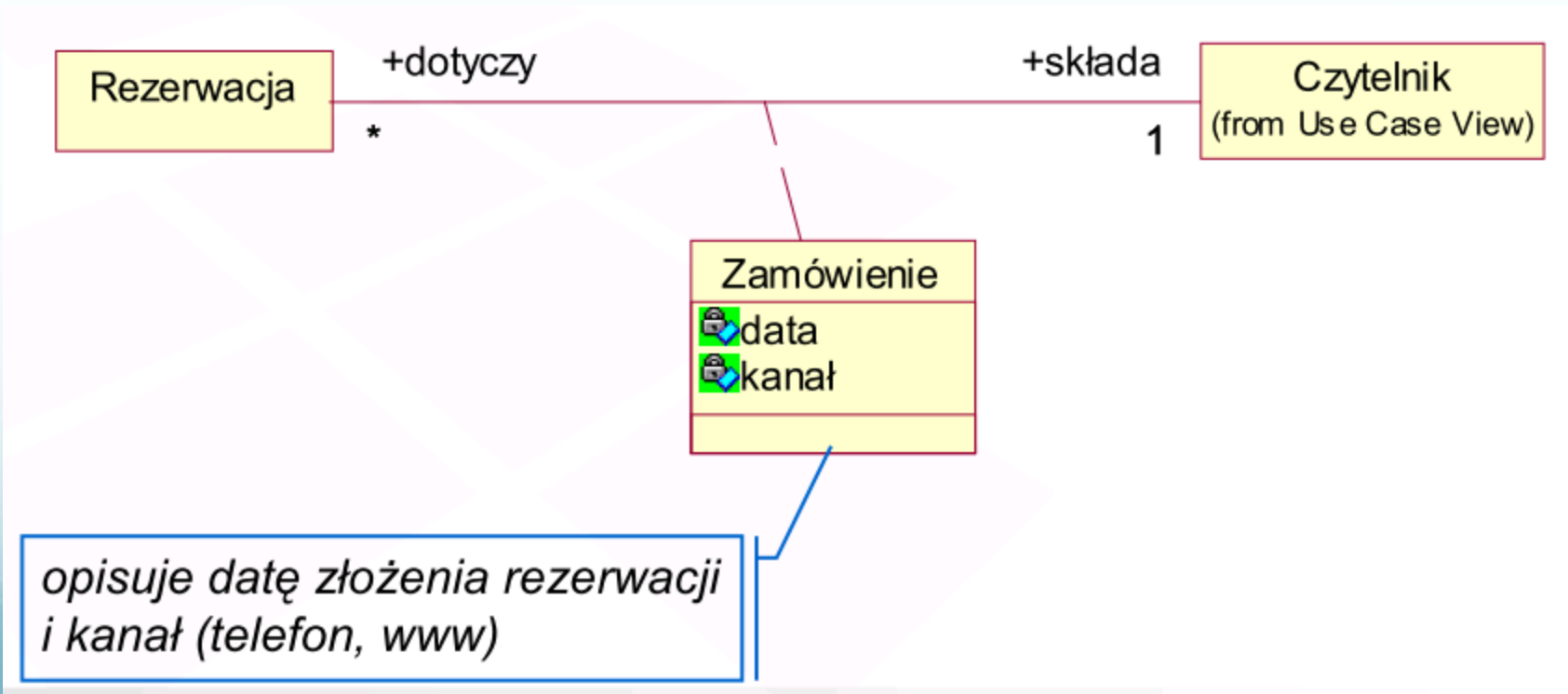
- Czytelnik wie o swoich rezerwacjach.
- Rezerwacja nie odwołuje się do czytelnika.



- Oba obiekty pozwalają na nawigację do siebie nawzajem.

Klasa asocjacji

- Klasy asocjacyjne są związane z relacją asocjacji i opisują jej właściwości.
- Mają dostęp do innych obiektów uczestniczących w relacji



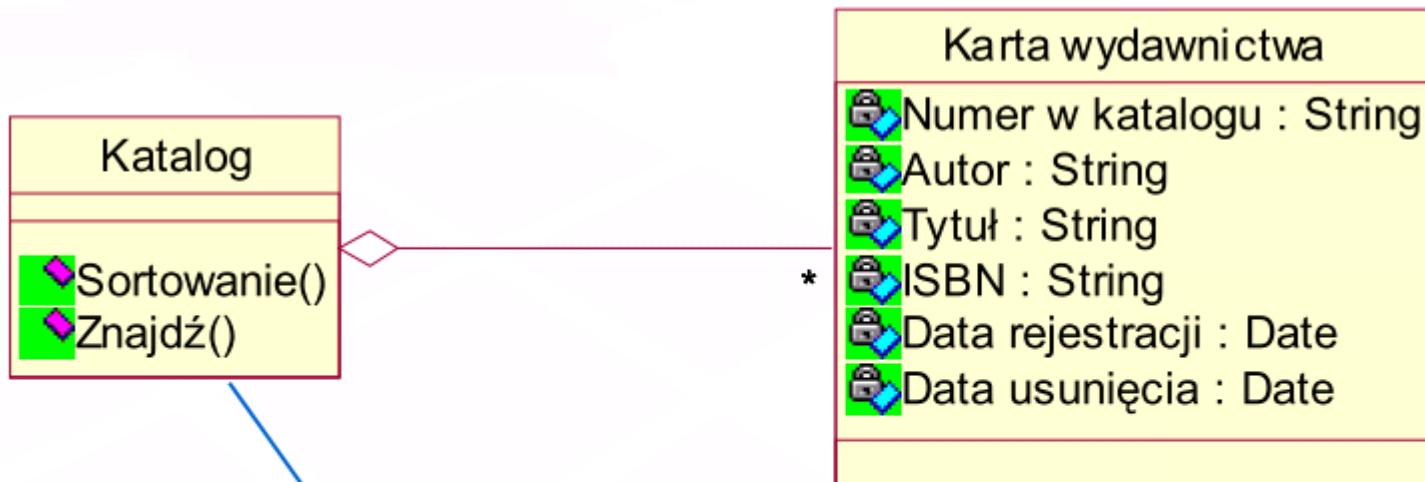
Asocjacja kwalifikowana

- **Asocjacja kwalifikowana** pozwala wskazać, który atrybut jednej z klas służy do zapewnienia unikatowości związku (jest jego kwalifikatorem).



Agregacja

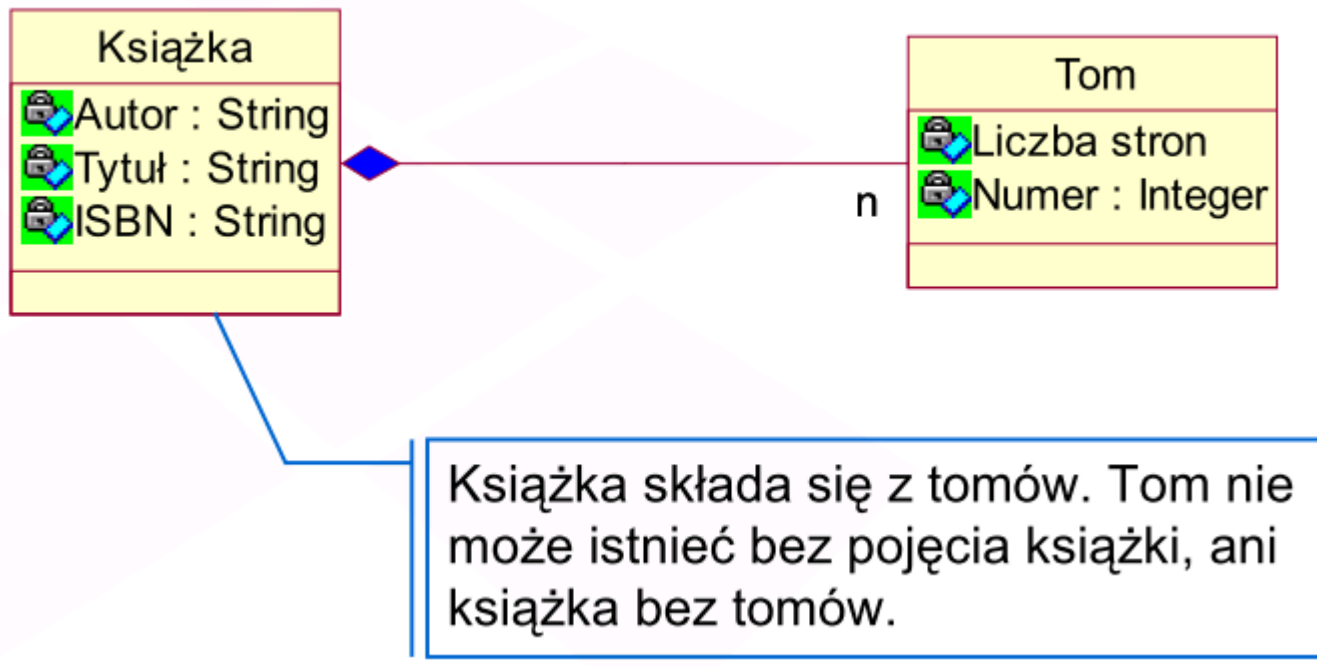
- **Agregacja** reprezentuje relację typu całość-część, w której część może należeć do kilku całości, a całość nie zarządza czasem istnienia części.



Katalog zawiera karty wydawnictw, ale nie tworzy ich. Nie jest ich wyłącznym właścicielem

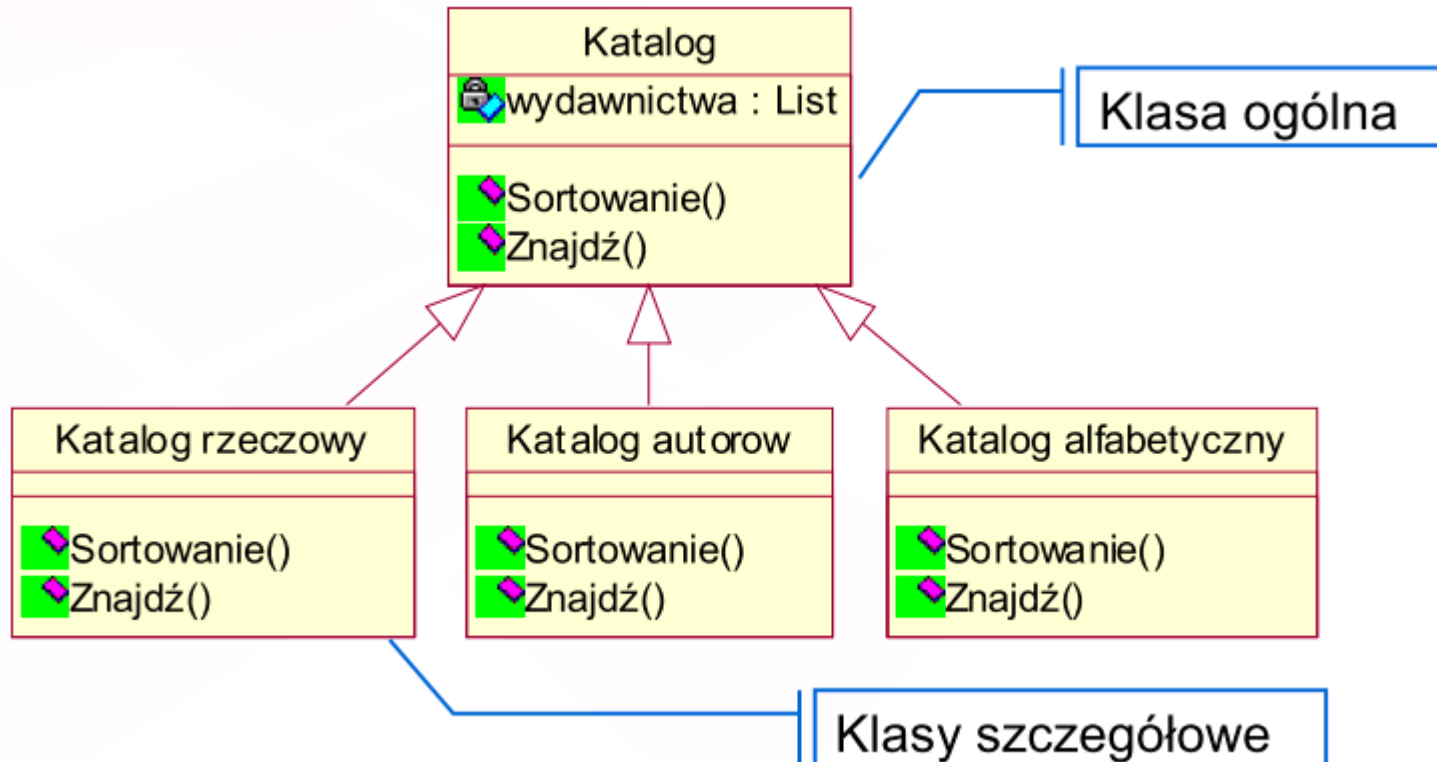
Kompozycja

- **Kompozycja** jest relacją typu **całość-część**, w której całość jest wyłącznym właścicielem części, tworzy je i zarządza nimi.



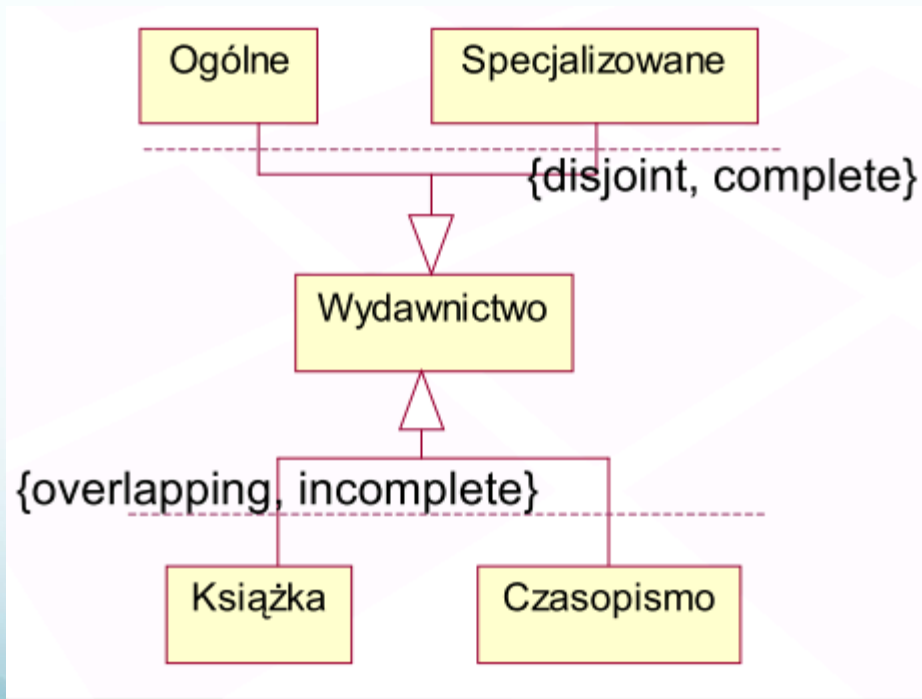
Uogólnienie

- **Uogólnienie** tworzy hierarchię klas, od ogólnych do bardziej szczegółowych. Pozwala wyłączyć części wspólne klas.



Klasyfikacja

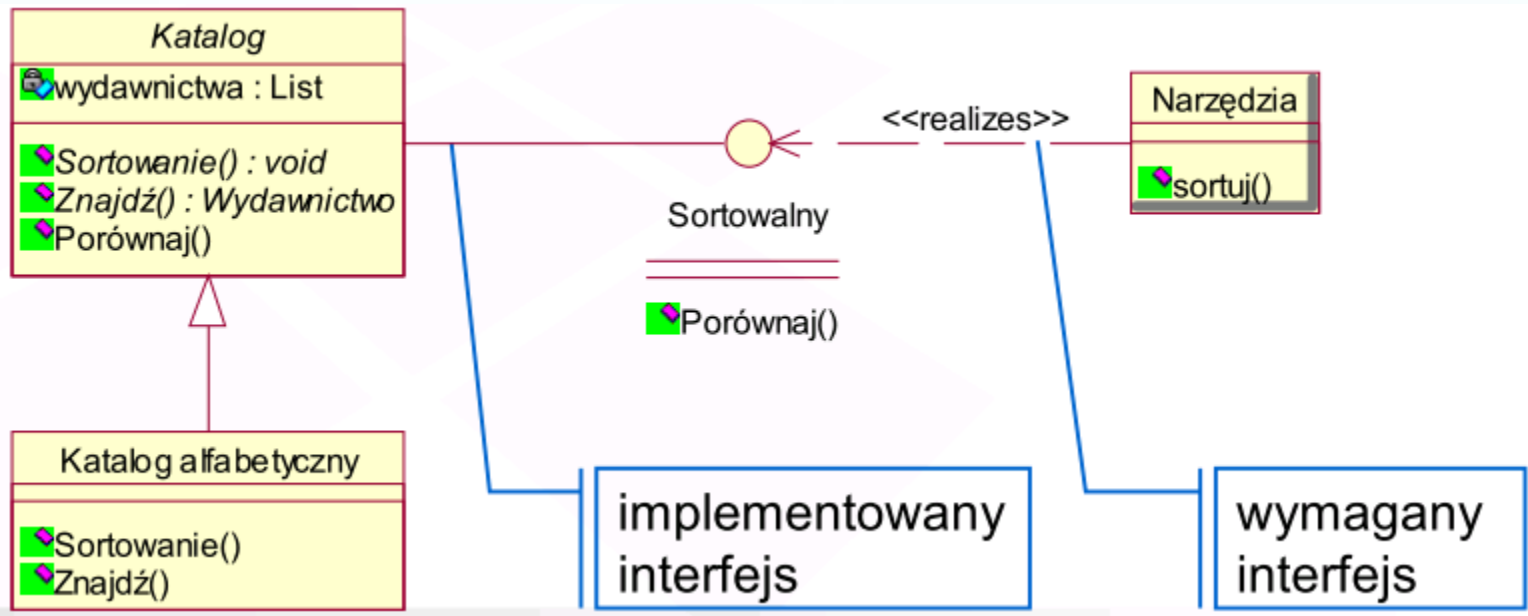
- **Klasyfikacja** określa związek między obiektem a jego typem (klasą).
- Obiekt może należeć jednocześnie do wielu typów.



- **{overlapping}** – obiekt może należeć do kilku klas
- **{disjoint}** – obiekt może należeć tylko do jednej klasy
- **{complete}** – nie istnieje więcej podklas
- **{incomplete}** – mogą powstać kolejne podklasy

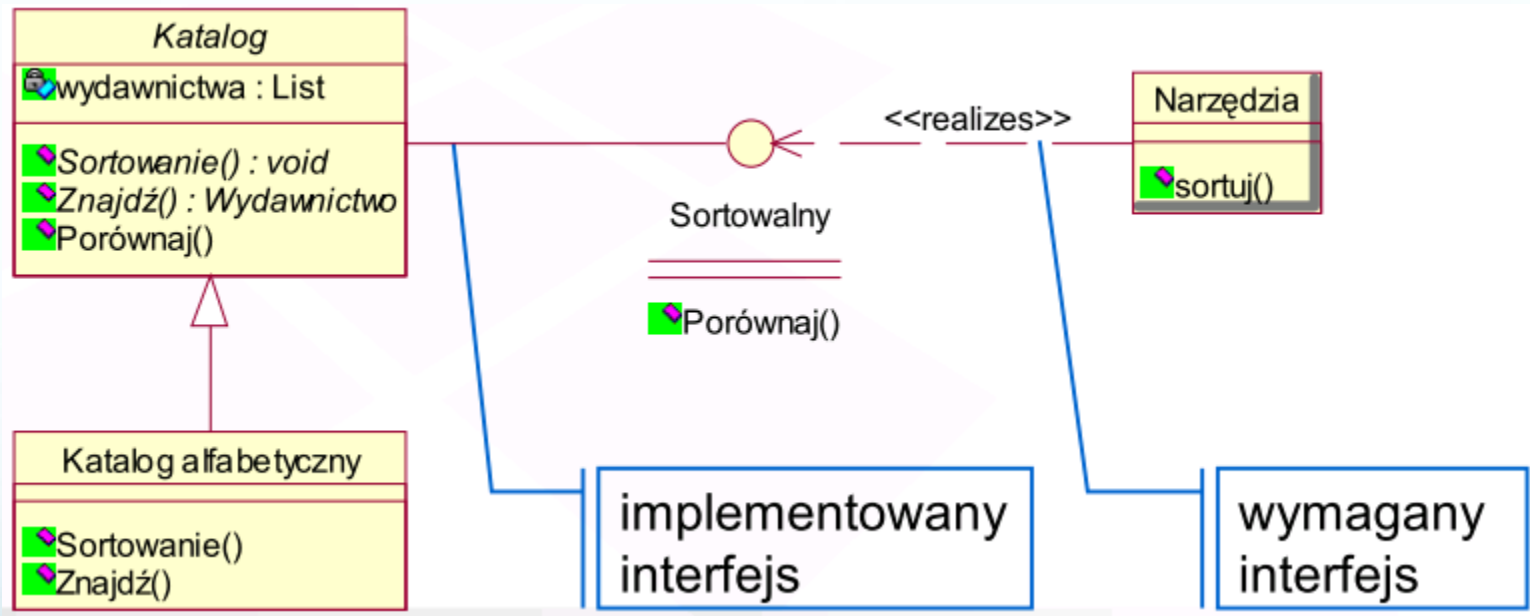
Interfejsy i klasy abstrakcyjne

- **Klasa abstrakcyjna** deklaruje wspólną funkcjonalność grupy klas. Nie może posiadać obiektów i musi definiować podklasy.
- **Interfejs** deklaruje grupę operacji bez podawania ich implementacji.



Interfejsy i klasy abstrakcyjne

- **Klasa abstrakcyjna** deklaruje wspólną funkcjonalność grupy klas. Nie może posiadać obiektów i musi definiować podklasy.
- **Interfejs** deklaruje grupę operacji bez podawania ich implementacji.



Szablony klas

- **Szablon klasy** określa, z jakimi innymi klasami (nie obiektami!) może współpracować podana klasa. Klasy te są przekazywane jako jej parametry.
- Klasa będąca uszczegółowieniem takiej klasy jest **klasą związaną**.

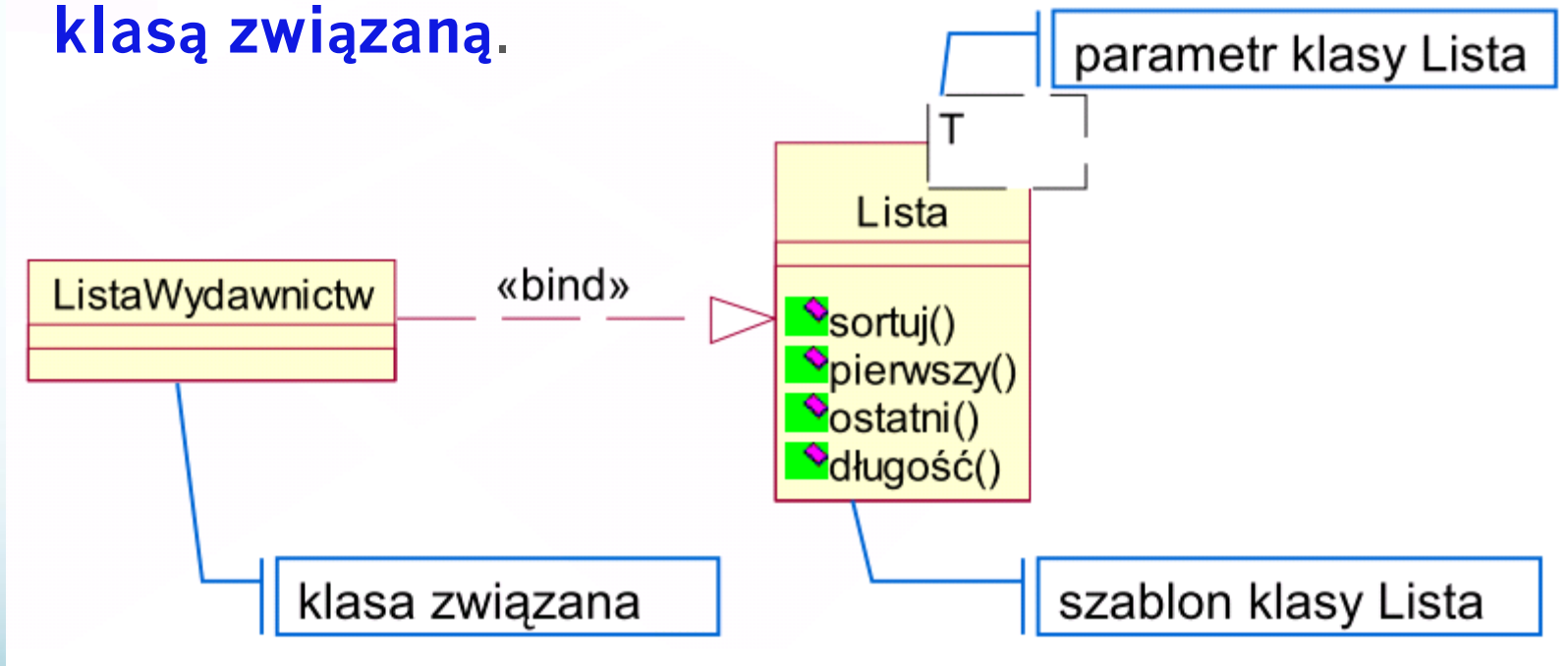


Diagram obiektów

- **Diagram obiektów** przedstawia możliwą konfigurację obiektów występujących w systemie.
- Relacje pomiędzy obiektami są bardziej dynamiczne niż w przypadku klas.

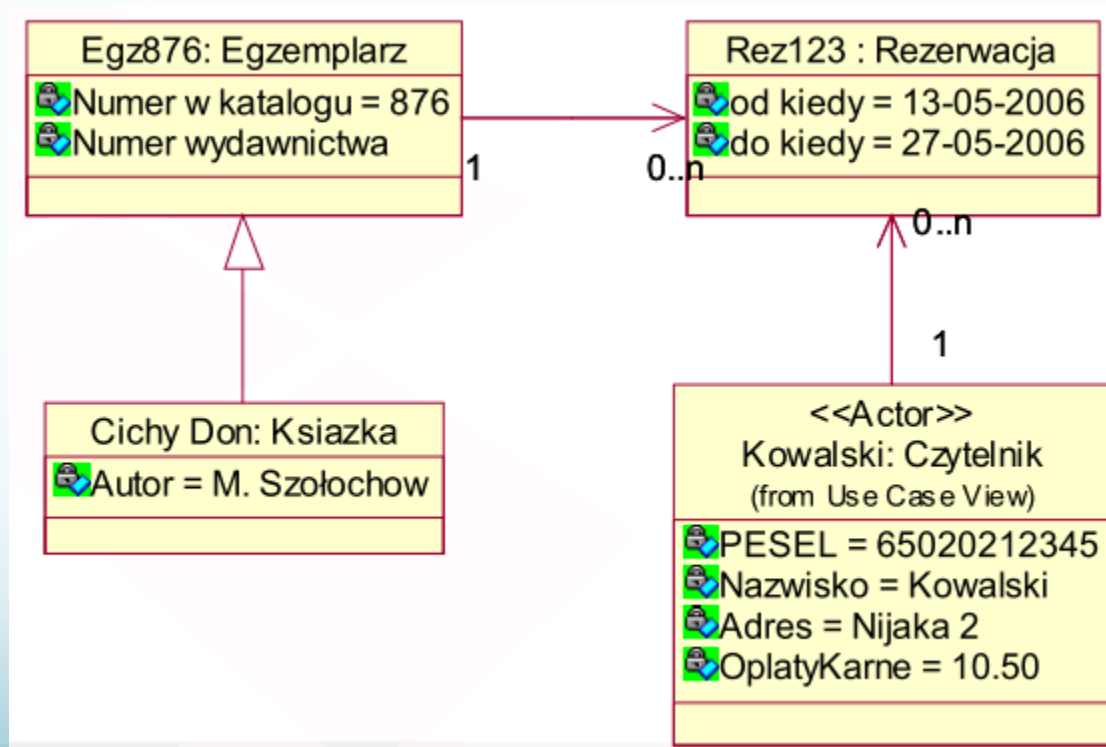
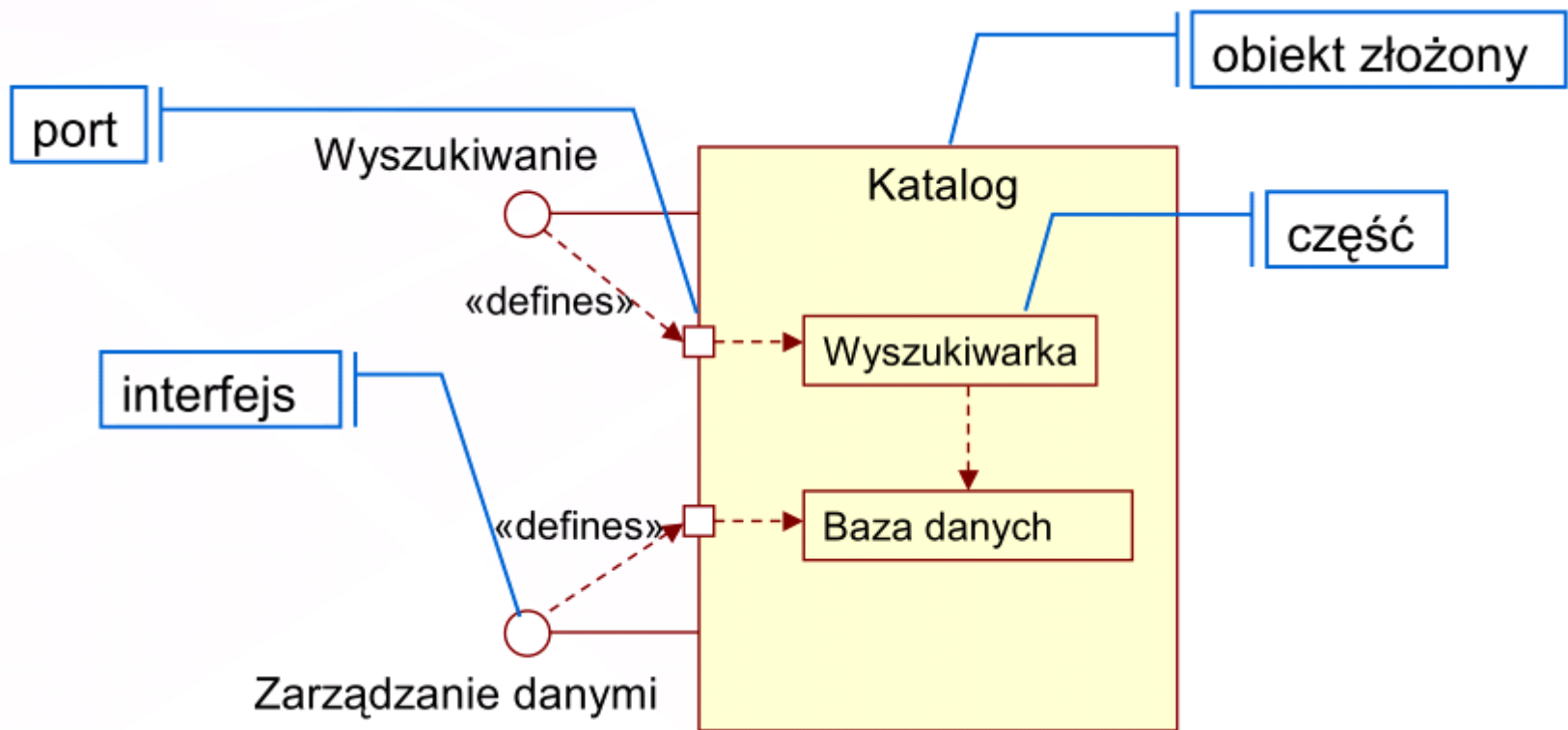


Diagram struktur złożonych

- **Diagram struktur złożonych** przedstawia wewnętrzną strukturę obiektu oraz punkty interakcji z innymi obiektami w systemie.



Podsumowanie



- UML powstał w wyniku połączenia różnych notacji i metodyk modelowania
- UML opisuje system w postaci 4 perspektyw wewnętrznych i 1 zewnętrznej
- Diagram przypadków użycia opisuje funkcje systemu i jego użytkowników
- Diagram klas określa statyczną strukturę logiczną systemu
- Diagram obiektów pokazuje możliwą konfigurację obiektów w systemie

Wykorzystane materiały

- Materiały opracowane na portalu edukacyjnym Uczelnia Online <http://wazniak.mimuw.edu.pl/>
- Wikipedia, wolna encyklopedia <http://pl.wikipedia.org/>
- Google Search Image <http://google.com/>
- Inne...



Literatura

- Booch G., Rumbaugh J., Jacobson I., *UML przewodnik użytkownika*. Wydawnictwa Naukowo-Techniczne, Warszawa 2001



- Wrycza S., *UML 2.1 – ćwiczenia*. Wydawnictwo Helion, Gliwice 2006

Koniec

