MODELOWANIE OBIEKTOWE

(Wykład na podstawie literatury: M.Śmiałek „Zrozumieć UML 2.0”, Helion 2005)

UML – *Unified Modeling Language*

(język do specyfikowania, wizualizowania, konstruowania i dokumentacji tzw. „artefactów” oraz czynności biznesowych lub innych z obszaru działalności człowieka

→ **Artefact** *(UML)* ***- fizyczny element informacyjny wyprodukowany lub używany w procesie wytwarzania oprogramowania (pliki .sor .com , elementy bazy danych, dokumenty, fragmenty modeli, itp.***

→ **Klasyfikatory** *(UML)* ***- elementy składowe modelu opisujące jego zachowanie lub strukturę posiadające swoją reprezentacje geometryczną.***

W szczególności zaliczamy do nich: **klasy, aktorzy, przypadki użycia, relacje.**

→ **Obiect** *(paradygmat obiektowości)* ***- „kawałek kodu” posiadający:*** **stan, zachowanie tożsamość**

stan ***– definiowany przez wartości atrybutów;***

zachowanie ***– operacje i usługi świadczone przez obiekt na żądanie innych o.***

tożsamość ***– unikalna cecha obiektu rozróżniająca obiekty o tych samych atrybutach i operacjach*** *(dwa obiekty mogą być równe ale nie identyczne)*

MODELOWANIE OBIEKTOWE

→ **Obiect** *(paradygmat obiektowości)*- ***„kawałek kodu”***

stan – ***definiowany przez wartości atrybutów;***

zachowanie – ***operacje i usługi świadczone przez obiekt na żądanie innych o.***

tożsamość – ***unikalna cecha obiektu rozróżniająca obiekty o tych samych atrybutach i operacjach*** *(dwa obiekty mogą być równe ale nie identyczne)*

Klasyfikacja modeli obiektowych:

Model stanu systemu (*State models*) – *statyczny stan systemu*

Model zachowania systemu *(Behavior models)*

Model zmiany stanu systemu *(State change models)*

(Modele te są reprezentowane przez jeden lub wiele diagramów)

UML – *wprowadza sześć typów diagramów:* struktura stanu ***(state structure)*** przypadki użycia ***(use case),*** **wykres (rejestr) stanu *(statechart),*** diagram współpracy ***(collaboration, communication diagrams),*** wykres aktywności ***(activity diagrams),*** wykres implementacji ***(implementation diagram)***

MODELOWANIE OBIEKTOWE

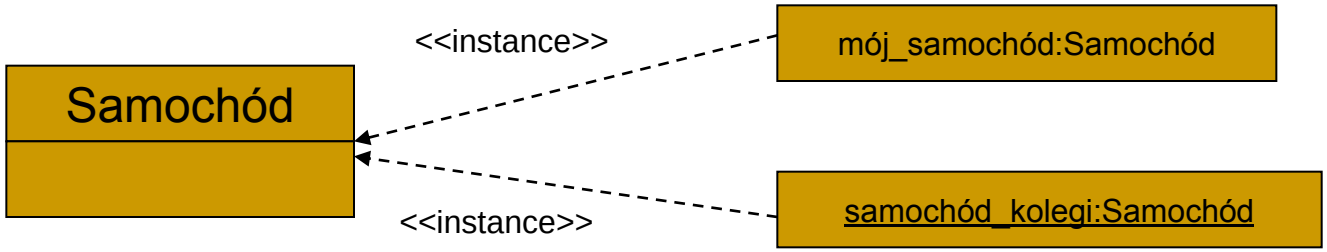
**STATYCZNA STRUCTURA MODELU (*State models*$_{(UML)}$*) wyraża się poprzez:*
Diagramy klas (*Class Diagrams*) – *obrazują klasy (interfejsy), ich wewnętrzna strukturę i relacje do innych klas.*

→ **Klasy** $_{(UML)}$**-** ***grupa obiektów o tych samych własnościach, ograniczeniach i semantyce.***
- ***deskryptor grupy obiektów o podobnej strukturze, zachowaniu i relacjach***

Do **własności** ***klasy zaliczamy*** **atrybuty** ***i*** **operacje**
Atrybuty – własność typu strukturalnego
Operacje – własność typu behawioralnego klasy, aktorzy, przypadki użycia, relacje.

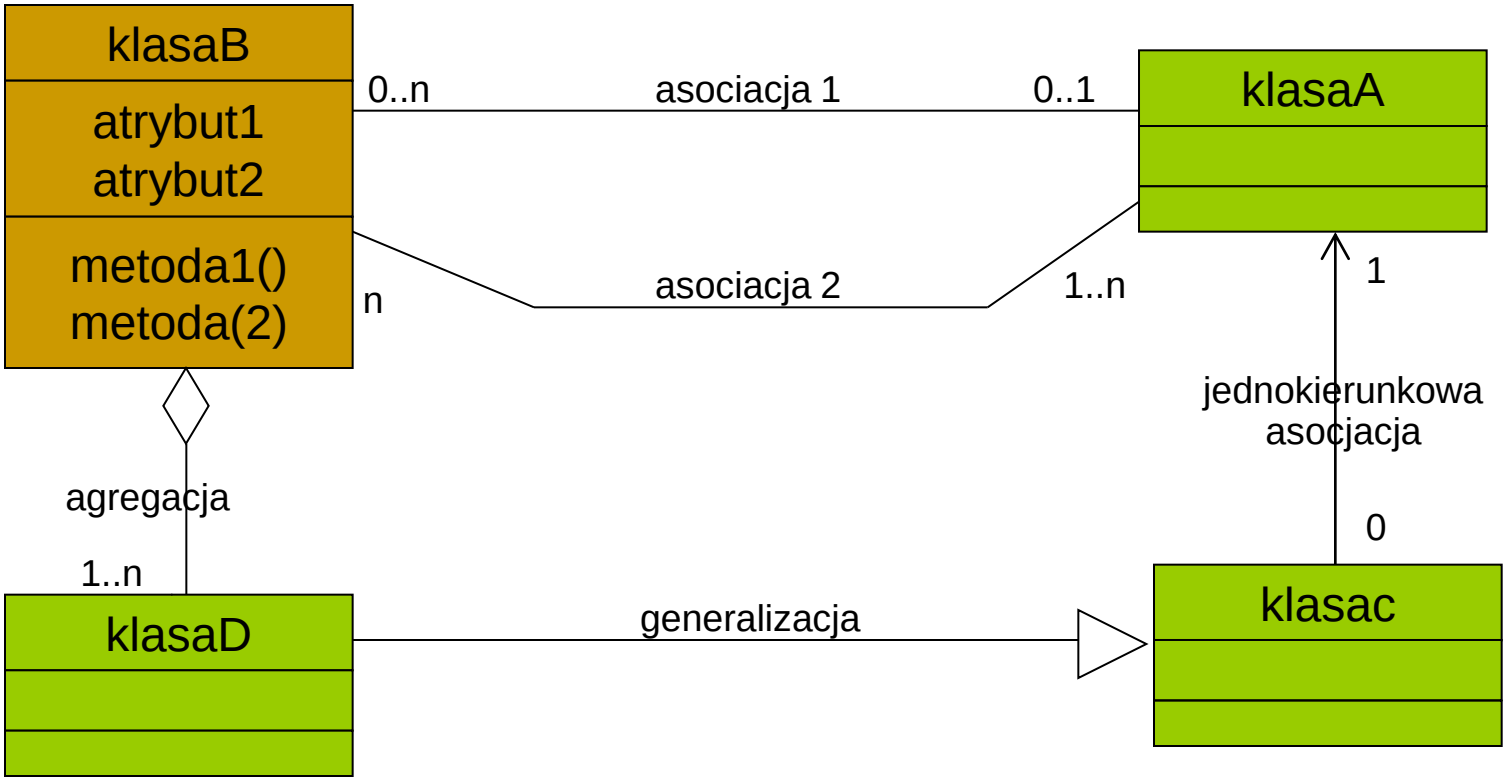
Relacje ***dotyczą połączeń pomiędzy klasyfikatorami (np. klasami w diagramach klas)***

→ **Instancja danej klasy** ***(instance)*** $_{(UML)}$***- obiekt przypisany do określonej klasy obiektów***



MODELOWANIE OBIEKTOWE

ELEMENTY DIAGRAMU KLAS



Asocjacja *(UML)* ***- zależność łącząca dwie klasy lub więcej (łącząca instancje!!)***

Mnogość ***– określa jak wiele instancji jednej klasy jest w relacji do jednej instancji drugiej klasy*** *(0 w mnogości określa asocjacje opcjonalną)*

Agregacja *(UML)* ***- jest rodzajem asocjacji wskazujacej na zawieranie się klas. Nadklasa zawiera wszystkie instancje podklasy***

MODELOWANIE OBIEKTOWE

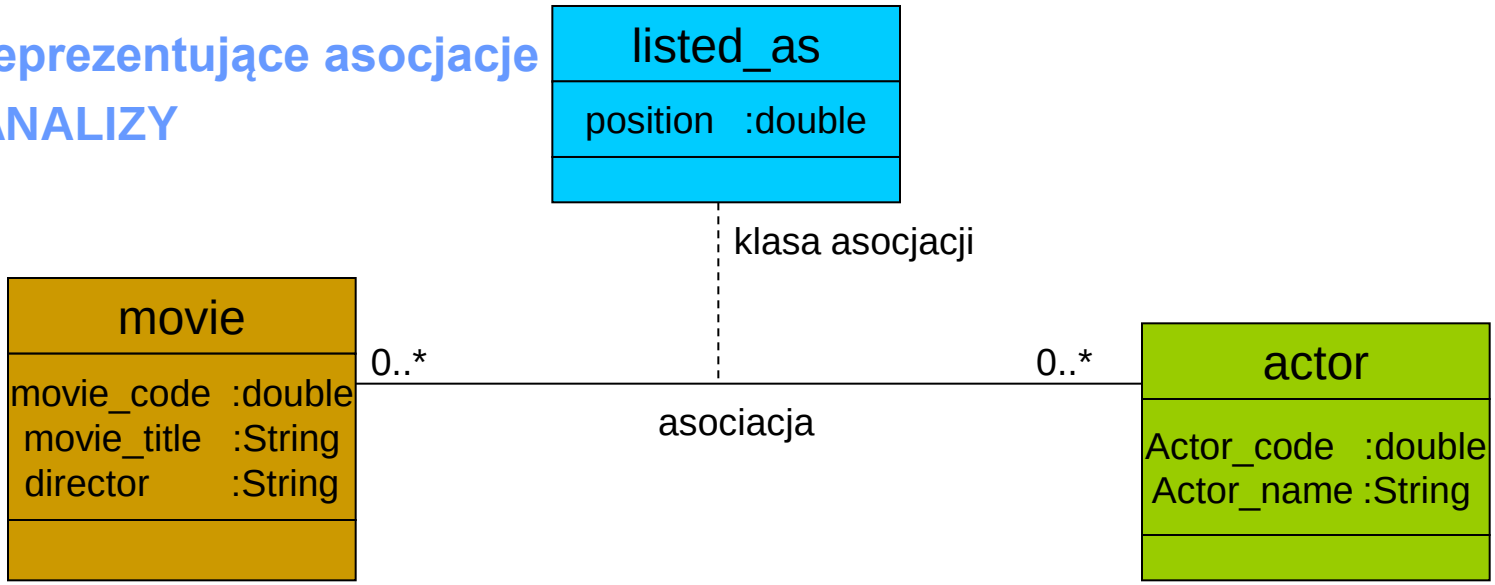
ELEMENTY DIAGRAMU KLAS

Generalizacja *(UML)* ***- jest rodzajem związku pomiędzy klasyfikatorami (klasami a nie instancjami) Nie definiujemy więc w nim mnogości.***

DIAGRAMY KLAS ***– pełnią dominującą rolę w OOM jak DFD's w modelowaniu strukturalnym. Diagramy klas jak i DFD's definiują operacje i struktury danych – podstawowa różnica: koncentracja odpowiednio na operacjach i strukturach danych.***

Klasy reprezentujące asocjacje

FAZA ANALIZY



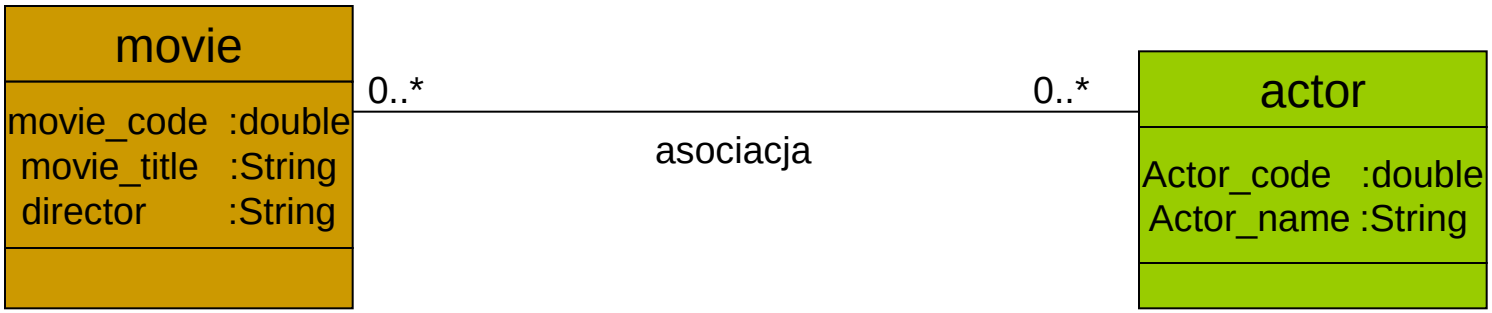
MODELOWANIE OBIEKTOWE

FAZA PROJEKTU (klasy Javy)

Projekt klasy Movie

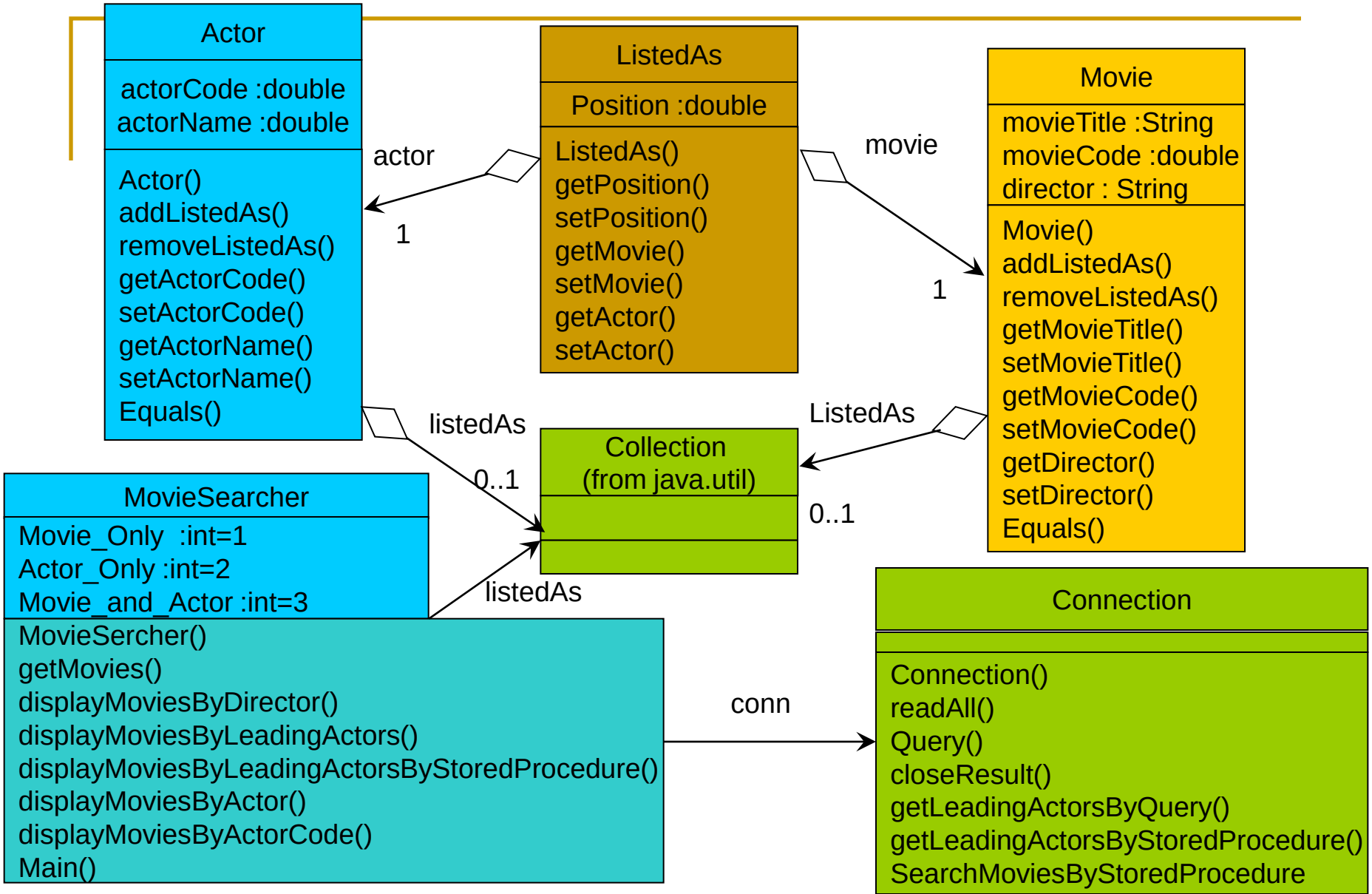
- **własności stanu i zachowania**
- **atrybuty**
- **konstructor obiektów**
- **Metody z sygnaturami (argumenty i ich typy)**
- **W UML na etapie projektowania nie określamy zwracanego przez funkcję typu**

Movie
movie_code :double movie_title :String director :String
Movie(movieCode :double, title :String, director :String) addListedAs(l : ListedAs) :void removeListedAs(l:ListedAs) :void getMovieTitle() :String setMovieTitle(property1 :String) :void getMovieCode() :double setMovieCode(property1 :double) :void getDirector() :String SetDirector(property1 :String) :void Equals(o :Object) :boolean



MODELOWANIE OBIEKTOWE

FAZA PROJEKTU (klasy Javy) MovieActor application



MODELOWANIE OBIEKTOWE

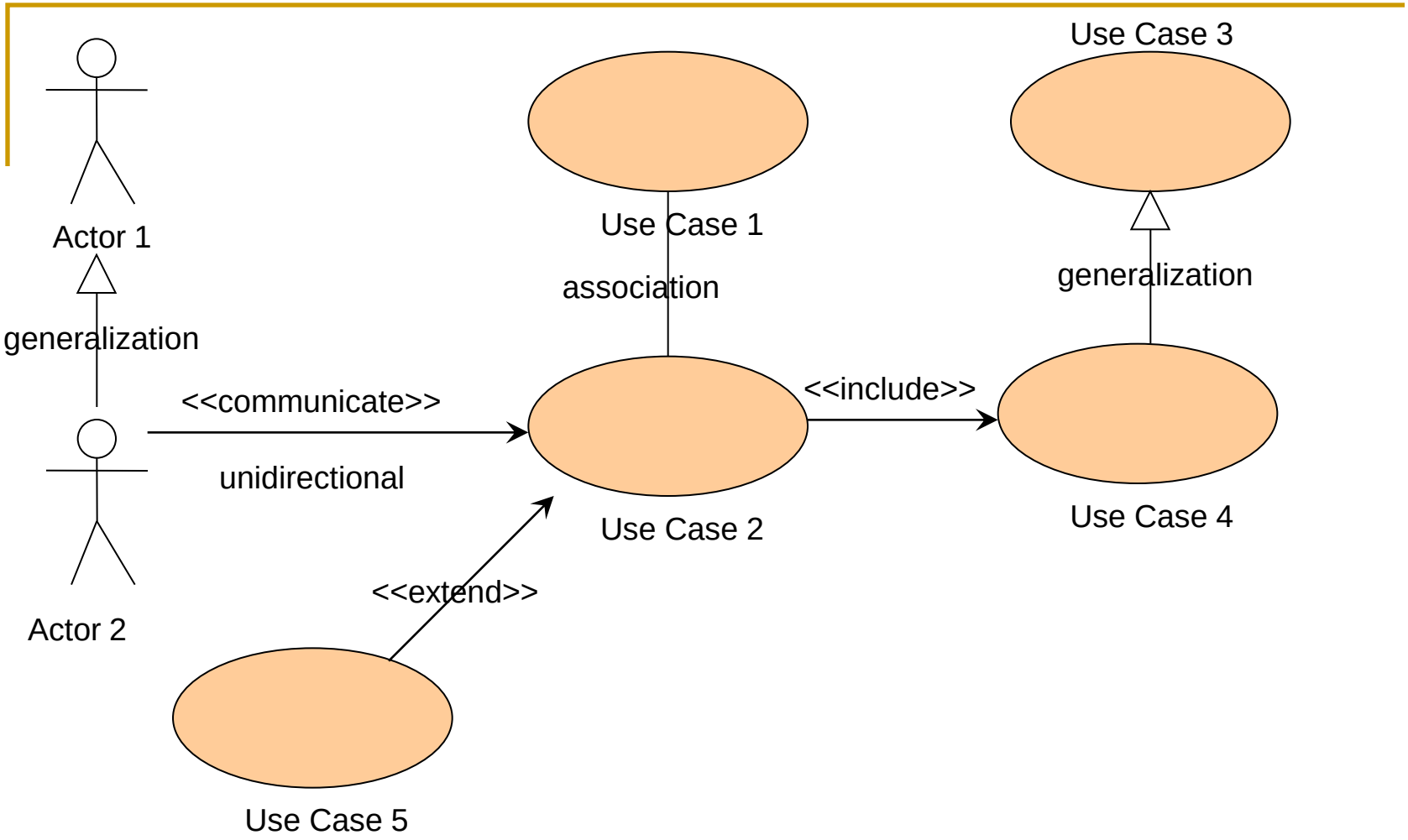
DIADRAMY PRZYPADKÓW UŻYCIA **Model zachowania systemu** **(nie generują hierarchicznej struktury systemu jak DFD's)**

- **Tekstowa specyfikacja przypadków użycia – składane w repozytorium**
- **Pozwalają na właściwe prowadzenie analizy, projektowania i implementacji systemu**
- **Dają możliwość określenia przypadków testowania, rejestrowania uszkodzeń systemu oraz identyfikacji przyszłych rozszerzeń (*enhancements)* systemu**

PRZYPADKi UŻYCIA – główne elementy funkcjonalności systemu

- **aktor – ktoś (coś) gra role, komunikuje się (przekazuje) z przypadkami użycia (*via <<communicate>>)* oczekując reakcji – wartości lub widocznego efektu**

MODELOWANIE OBIEKTOWE



MODELOWANIE OBIEKTOWE