



# UML – WARSZTATY PRAKTYCZNE W ZAKRESIE NIEZBĘDNYM DO REALIZACJI ZADAŃ ZWIĄZANYCH Z INSPIRE

Marek Strzelecki, Przemysław Biegański  
Wrocławski Instytut Zastosowań Informatyki  
Przestrzennej i Sztucznej inteligencji



Warszawa, wrzesień-październik 2016 r



# PROGRAM SZKOLENIA

- Wprowadzenie do UML.
- Notacja UML – podstawy stosowanych elementów graficznych.
- Najczęściej stosowane diagramy UML.
- Szczegółowe omówienie diagramów:
  - przypadków użycia,
  - aktywności,
  - interakcji,
  - klas,
  - pakietów.
- Omówienie schematów aplikacyjnych UML w szczególności wykorzystywanych w INSPIRE.
- Podstawy języka OCL.
- Rola UML w INSPIRE, zastosowanie UML dla potrzeb INSPIRE.
- Interpretacja istniejących diagramów UML, mających praktyczne zastosowanie w INSPIRE.



WROCLAW GŁÓWNY

budimex  
50 LAT

# WPROWADZENIE DO UML



# PROCES TWORZENIA OPROGRAMOWANIA

- Wraz ze wzrostem skomplikowania systemów informatycznych proces tworzenia oprogramowania jest bardzo rozbudowany.
- Wymaga zorganizowanego podejścia m.in. z wykorzystaniem technik i narzędzi inżynierii oprogramowania.
- Jest zadaniem zespołowym, w którego skład wchodzi nie tylko programiści.





# MODELOWANIE

- Systemy informatyczne mają na celu rozwiązanie określonego problemu.
- Na potrzeby systemu tworzony jest **model pojęciowy** (konceptualny), ukazujący pojęcia i związki między nimi.
- Analitik musi dokładnie wyobrazić sobie dany problem i opracować metodę jego rozwiązania.
- Posługuje się przybliżonym obrazem rzeczywistości za pomocą:
  - struktur danych,
  - operacji i algorytmów.



# MODELOWANIE

- W jaki sposób wyrazić przemyślenia analityka?
- Czy język naturalny jest wystarczająco ekspresywny?
- Jak komunikować się w rozbudowanym zespole?
- Jak zapewnić w nim odpowiedni przepływ informacji?
- Z jakich perspektyw należy opisywać system informatyczny?



# UNIFIED MODELING LANGUAGE

- Wzrost znaczenia podejścia obiektowego (początek lat 90).
- Mnogość metod modelowania obiektowego (1989-1994 ok. 50 zidentyfikowanych metod) – próba ujednolicenia.
- 1994 – rozpoczęcie prac nad UML (Booch, Jacobson, Rumbaugh).
- 1995 – Unified Method 0.8 .
- 1997 – UML 1.0 .
- **2003 – UML 2.0 .**
- 2009 – UML 2.2 .



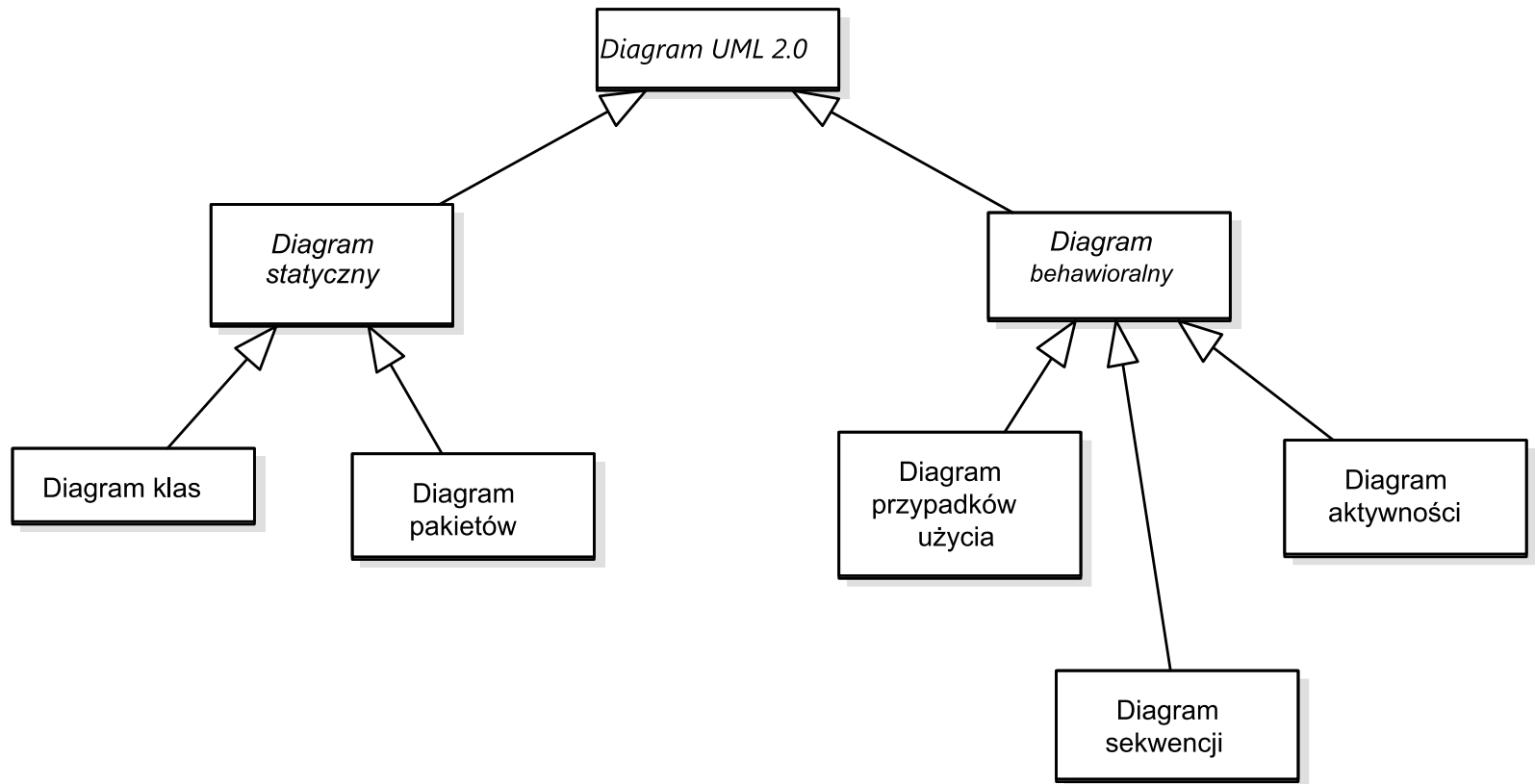
# UNIFIED MODELING LANGUAGE

- Ujednolicony język modelowania systemów.
- Wykorzystuje notacje graficzne do tworzenia diagramów opisujących różne aspekty modelowanych systemów.
- Wykorzystuje podejście obiektowe.
- Rozważamy wersję UML 2.0
- Typy diagramów UML.





# WYBRANE DIAGRAMY UML





# DIAGRAM UML

RAMKA Z ETYKIETĄ I TYPEM

**class DiagramUML**

ZAWARTOŚĆ DIAGRAMU



# KLASYFIKATORY

- Diagramy UML tworzone są z elementów zwanych klasyfikatorami.
- Klasyfikatory to elementy opisujące strukturę lub dynamikę systemu posiadające reprezentację graficzną.
- Należą do nich m.in.:
  - Aktorzy,
  - Przypadki użycia,
  - Klasy,
  - Związki,
  - Relacje.



# DIAGRAM PRZYPADKÓW UŻYCIA





# SPECYFIKACJA WYMAGAŃ

- Specyfikacja wymagań to zbiór wymagań, czyli co oprogramowanie powinno robić.

## Typy wymagań:

- Wymagania funkcjonalne – funkcje systemu widziane od strony użytkownika:
  - zakup książki,
  - drukowanie raportu,
  - możliwość przybliżenia mapy.
- Wymagania niefunkcjonalne – wymagania niezwiązane bezpośrednio z funkcjami systemu ale mające wpływ na działanie całego systemu:
  - zgodność ze standardami,
  - konieczność zastosowania pewnych technologii,
  - niezawodność,
  - bezpieczeństwo.



# WYMAGANIA FUNKCJONALNE – PRZYKŁAD DEFINIOWANIA

Definiowanie za pomocą opisu w języku naturalnym:

System powinien:

- umożliwiać zakup książek,
- umożliwiać drukowanie raportu,
- mieć modyfikowalny interfejs,
- posiadać możliwość zalogowania użytkownika,
- umożliwiać zaznaczenie obiektów przestrzennych

...

Dobry początek, ale nie sprawdza się dla złożonych systemów informatycznych.





# WYMAGANIA FUNKCJONALNE – PRZYKŁAD DEFINIOWANIA

**Przypadek użycia** (use case) – przedstawia interakcje pomiędzy użytkownikiem (**aktorem**) a systemem.

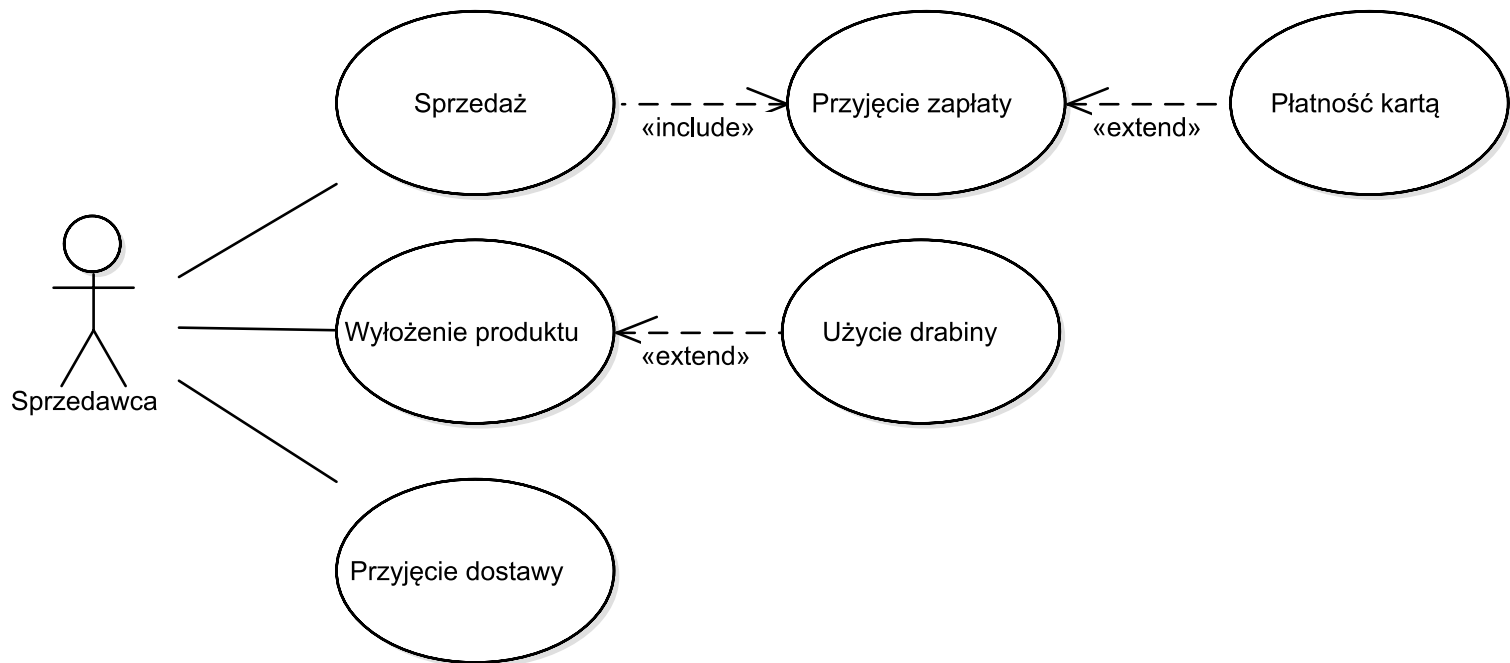
Najpowszechniejszy sposób definiowania wymagań dla systemu informatycznego. Możliwe jest jego utworzenie na podstawie opisu systemu.

1. Opis interakcji **aktora** z systemem prowadzący do konkretnego **celu**.
2. Brak szczegółów związanych z technologią i interfejsem użytkownika.
3. Odpowiednio dobrany **poziom szczegółowości**.



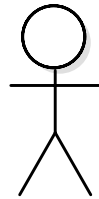
# DIAGRAM PRZYPADKÓW UŻYCIA

- Diagram Przypadków to narzędzie UML pozwalające na przedstawienie zidentyfikowanych przypadków użycia.

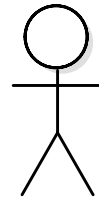




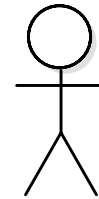
# AKTORZY



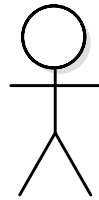
Aktor



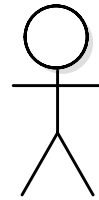
Użytkownik



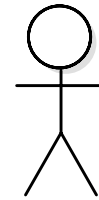
Administrator



Sprzedawca



Pracownik



Specjalista ds kadrowo-  
płacowych

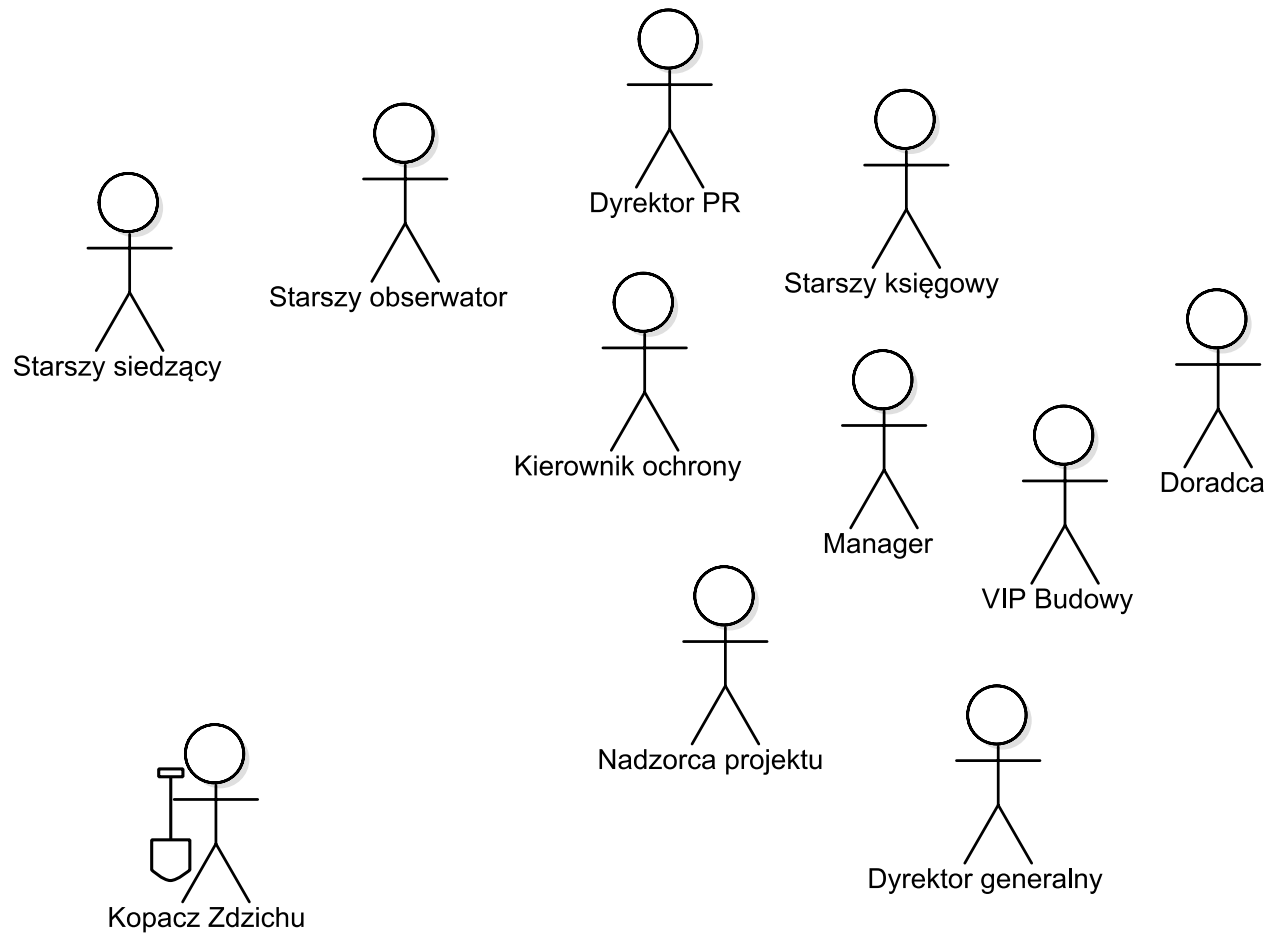


# AKTORZY





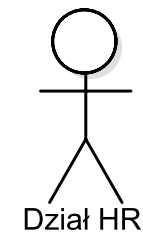
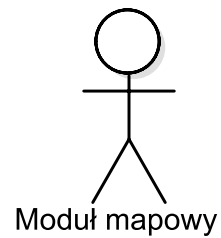
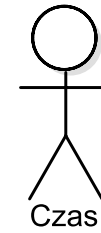
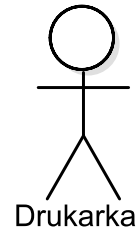
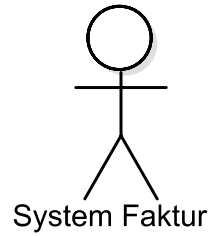
# AKTORZY





# AKTORZY

- Aktorzy nieosobowi:

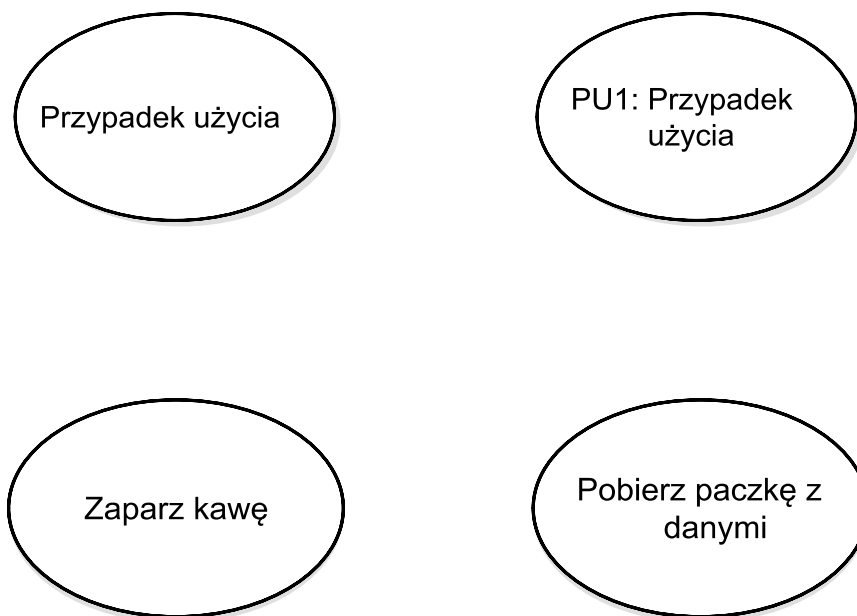






# PRZYPADEK UŻYCIA

"Opis interakcji między aktorem a systemem."





# PRZYPADEK UŻYCIA

## POPRAWNE NAZWY:

Logowanie  
użytkownika

Zaloguj się

Generowanie  
raportu

Generuj raport



# PRZYPADEK UŻYCIA

## NIEPOPRAWNE NAZWY:

Mój przypadek  
użycia

Użytkownik loguje  
się do systemu

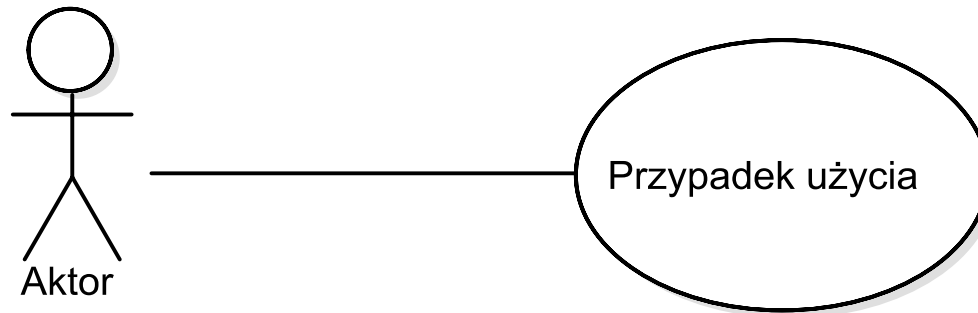
Zarządzanie

Zapłata za wybrany  
produkt i jego  
wysyłka



# ASOCJACJA

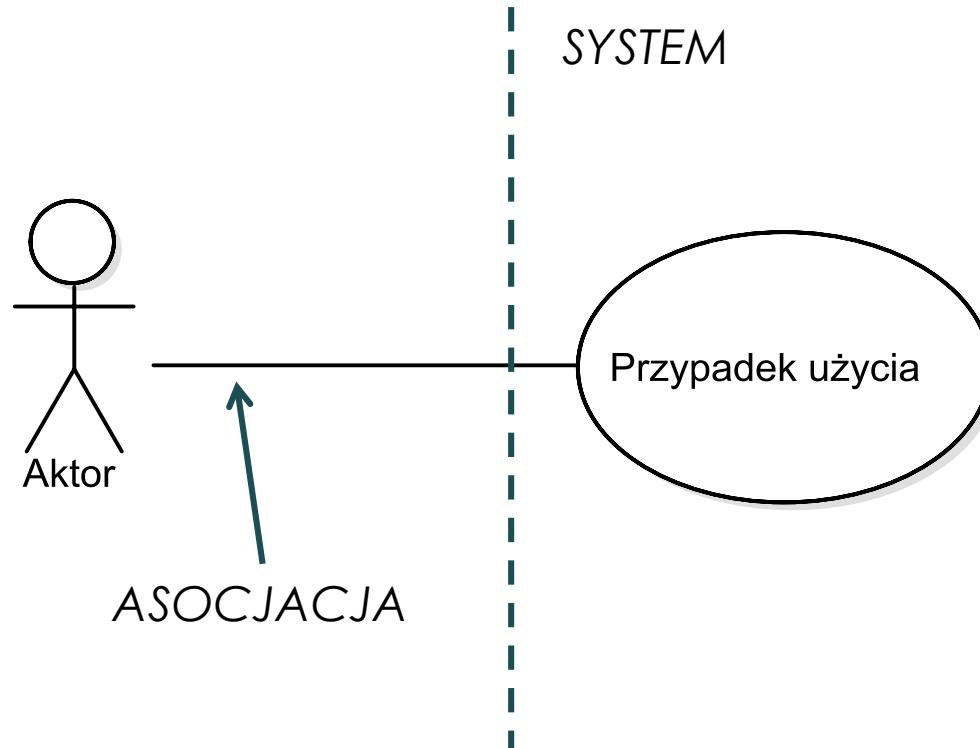
- "Opis **interakcji** między aktorem a systemem."





# ASOCJACJA

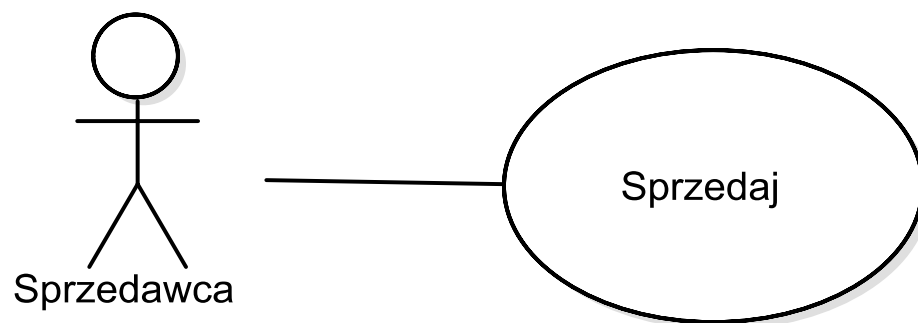
- "Opis **interakcji** między aktorem a systemem."





# ASOCJACJA

- Prosta asocjacja:

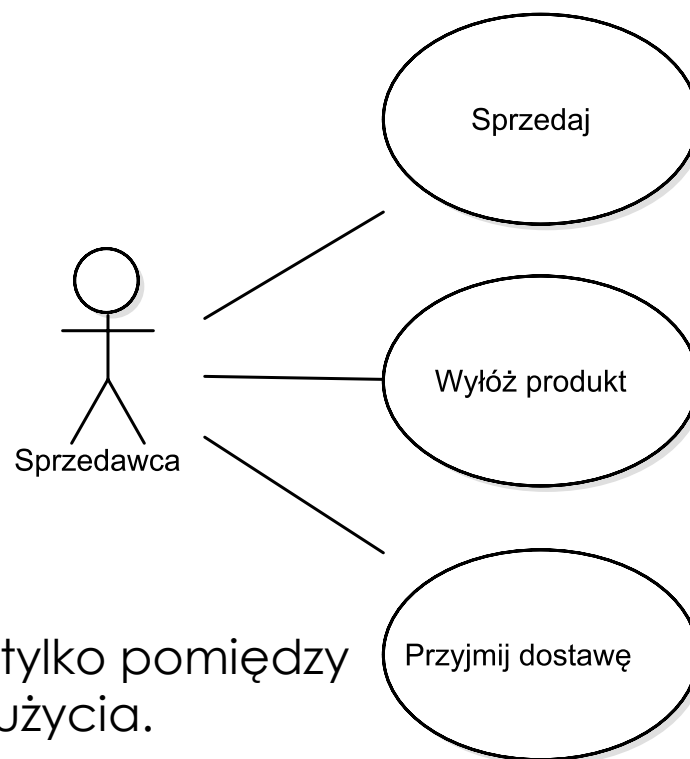






# ASOCJACJA

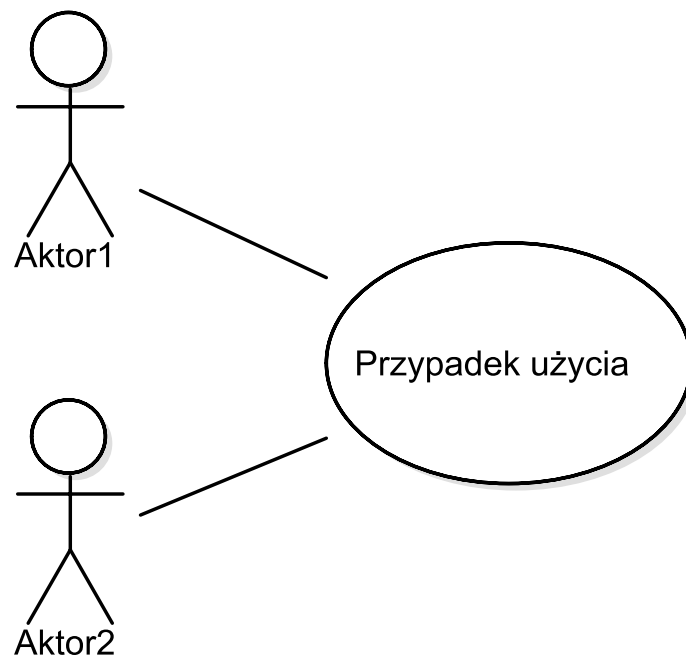
- Sprzedawca inicjuje wystąpienie kilku interakcji z systemem.



- Asocjacja występuje tylko pomiędzy aktorem i przypadkiem użycia.



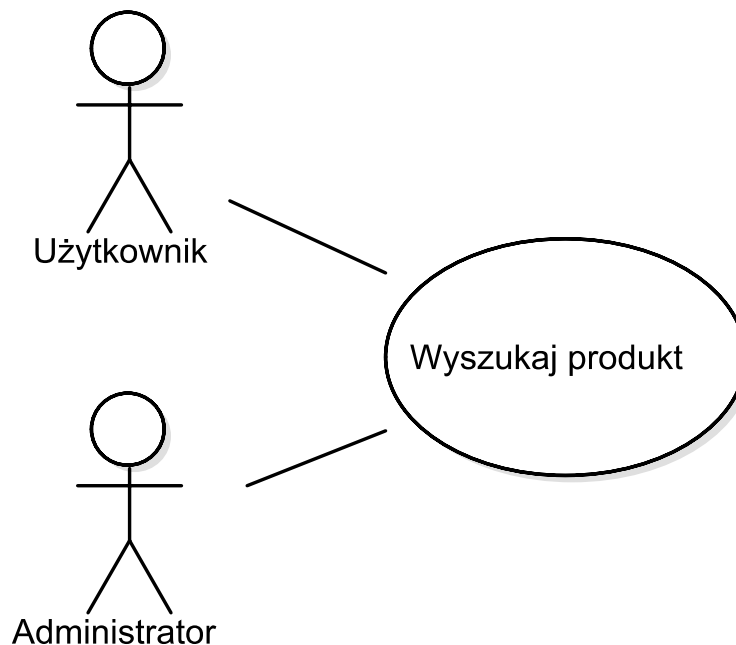
# ASOCJACJA





# ASOCJACJA

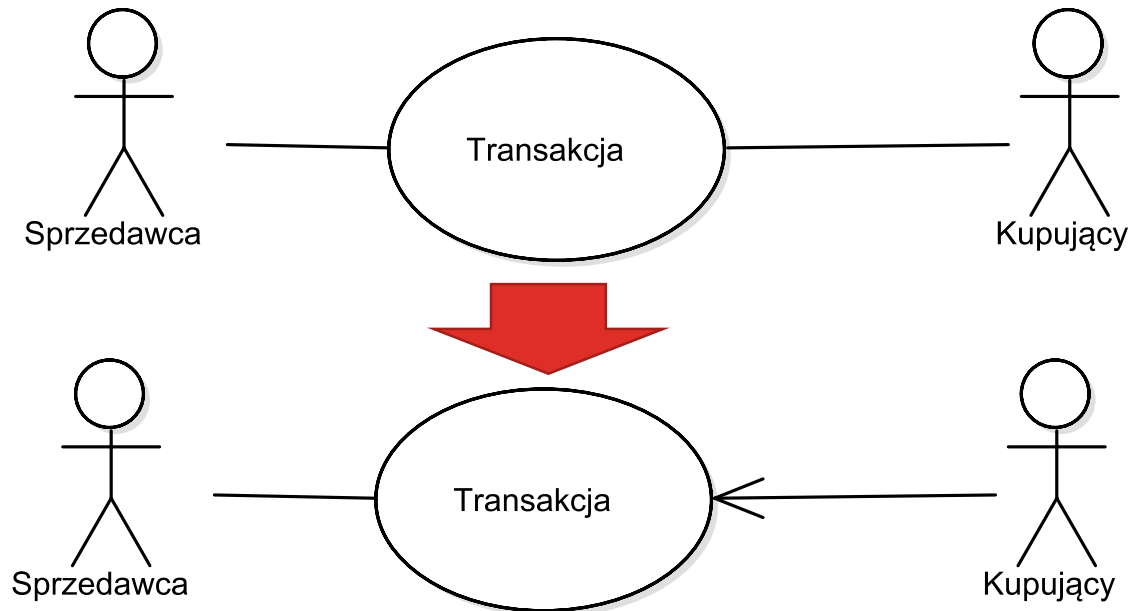
- Zarówno użytkownik i administrator mogą wyszukać produkt w katalogu.





# ASOCJACJA SKIEROWANA

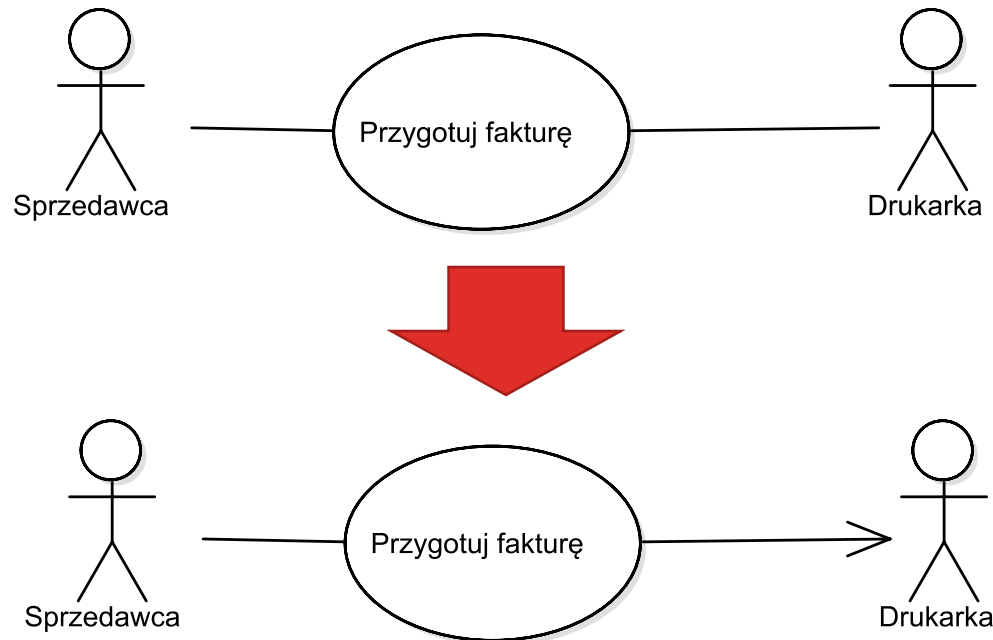
- Opcjonalnie można wskazać wprost kierunek inicjacji interakcji.





# ASOCJACJA SKIEROWANA

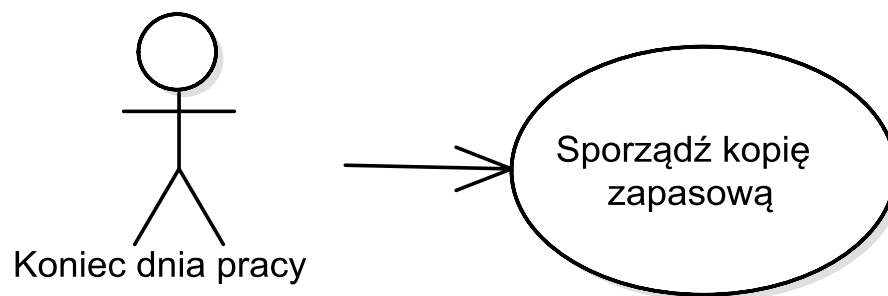
- "Kontrowersyjna" sytuacja (PU inicjuje interakcje):





# ASOCJACJA SKIEROWANA

- Aktor "CZAS":

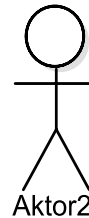
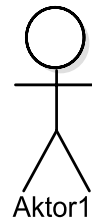






# UOGÓLNIENIE (GENERALIZACJA)

- Sposób przedstawiania hierarchii klasyfikatorów (głównie aktorów):

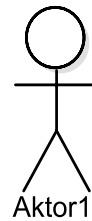




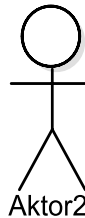
# UOGÓLNIENIE (GENERALIZACJA)

- Sposób przedstawiania hierarchii klasyfikatorów (głównie aktorów):

Dziecko "dziedziczy" po rodzicu.



"Rodzic"

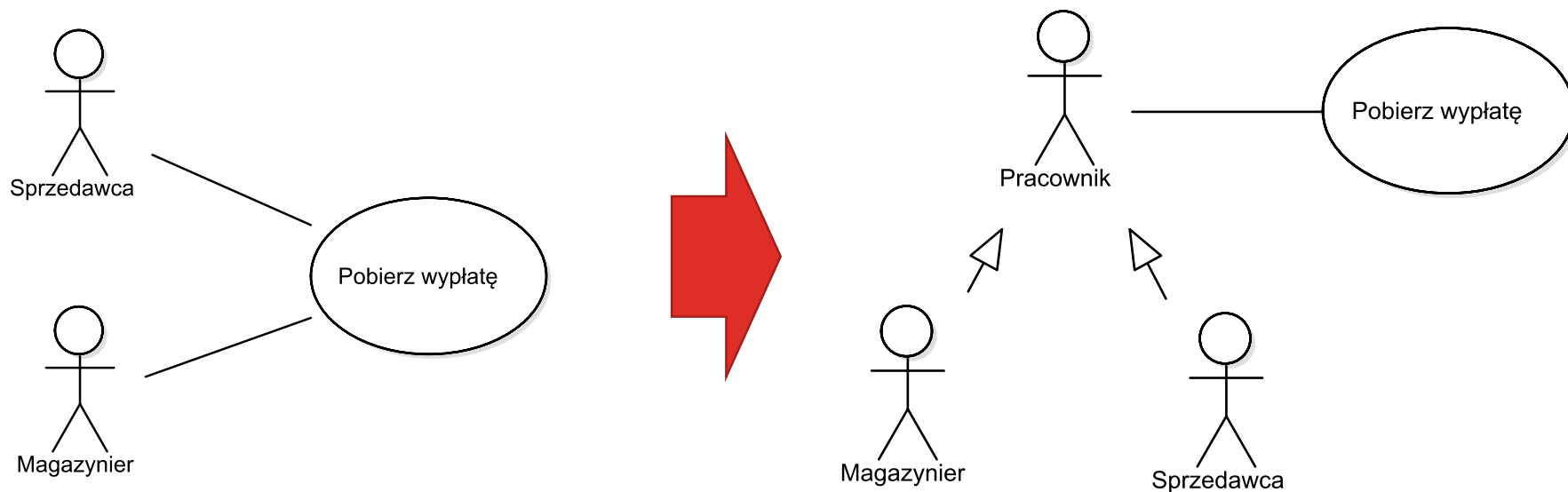


"Dziecko"



# UOGÓLNIENIE (GENERALIZACJA)

- Zarówno Magazynier i Sprzedawca są Pracownikami.



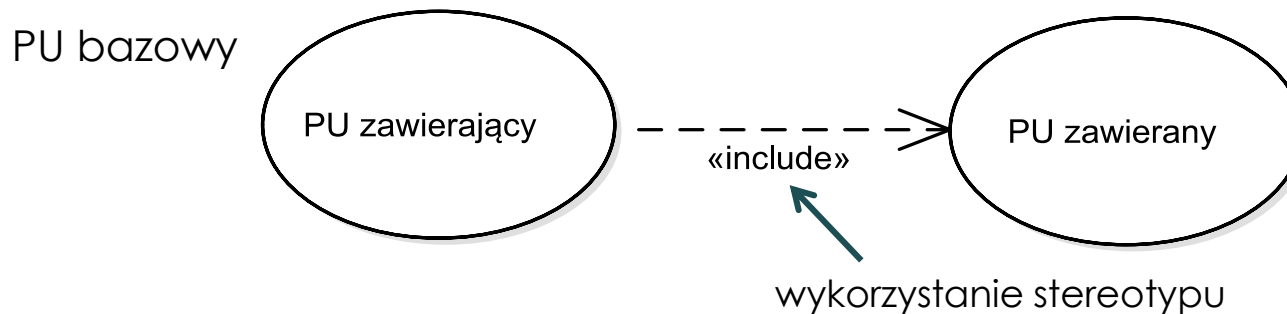


# ZALEŻNOŚCI

- Zwykłe asocjacje i generalizacje mogą nie być wystarczająco ekspresywne do wyrażenia złożoności wymagań stawianych wobec systemu.
- Możliwe jest wykorzystanie zależności między przypadkami użycia.
- Zależność to związek pomiędzy dwoma klasyfikatorami, w którym zmiana jednego z nich wpływa na drugi. Możemy wyróżnić:
  - Zależność zawierania,
  - Zależność rozszerzania.



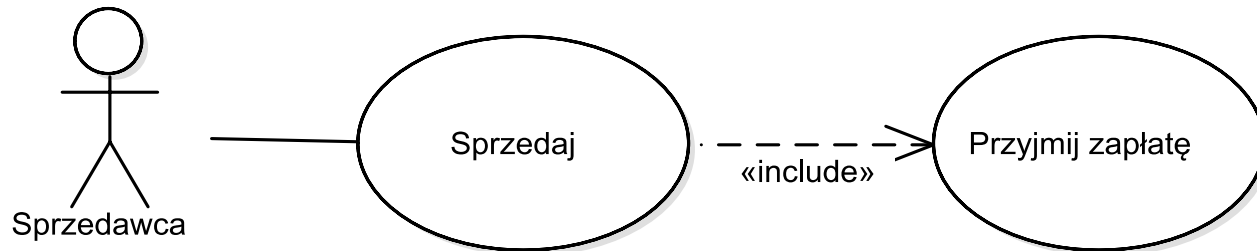
# ZALEŻNOŚĆ ZAWIERANIA



- Jest to związek obligatoryjny.
- "PU zawierany" wykonywany jest **ZAWSZE**, ale tylko wtedy gdy wykonywany jest "PU zawierający".
- "PU zawierany" nie jest samodzielny.



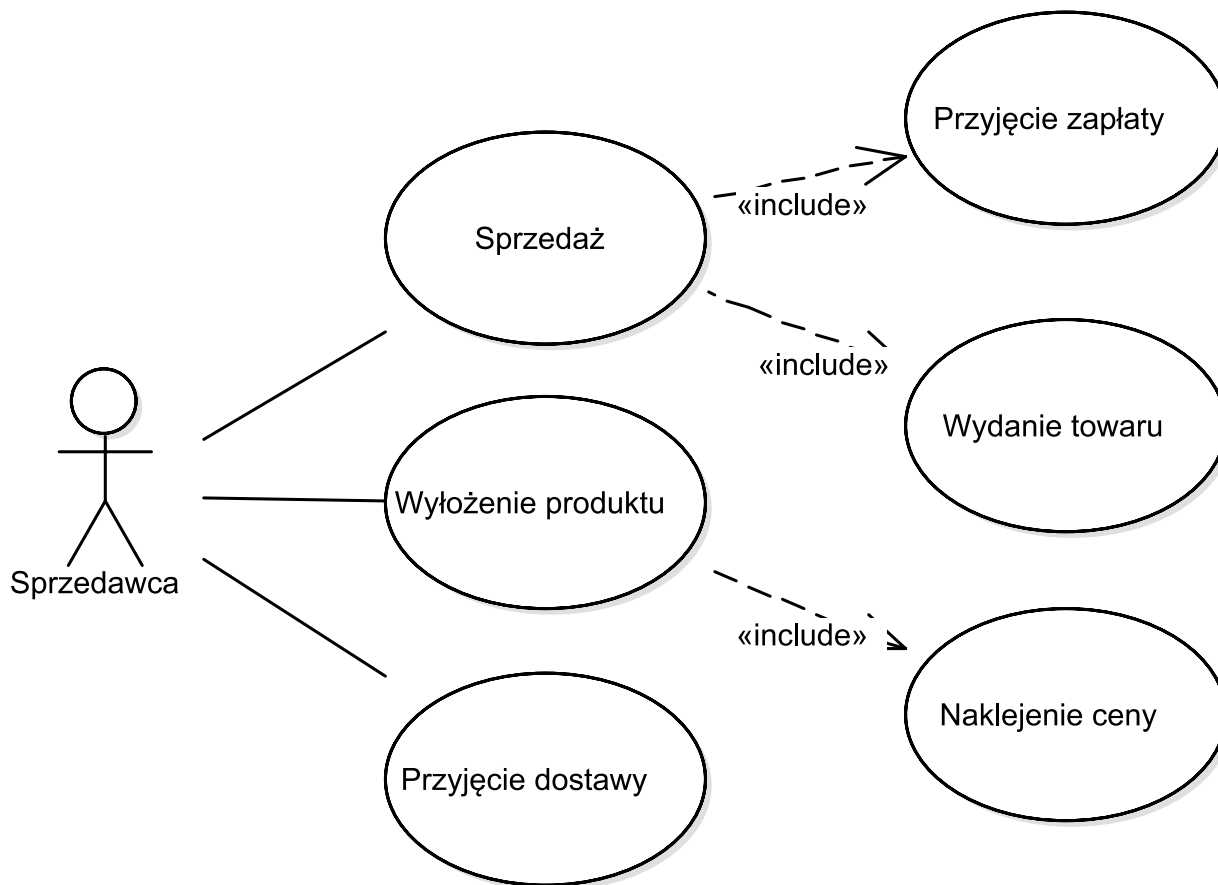
# ZALEŻNOŚĆ ZAWIERANIA



- Gdy przeprowadzana jest sprzedaż, **zawsze** pobierana jest zapłata za towar.

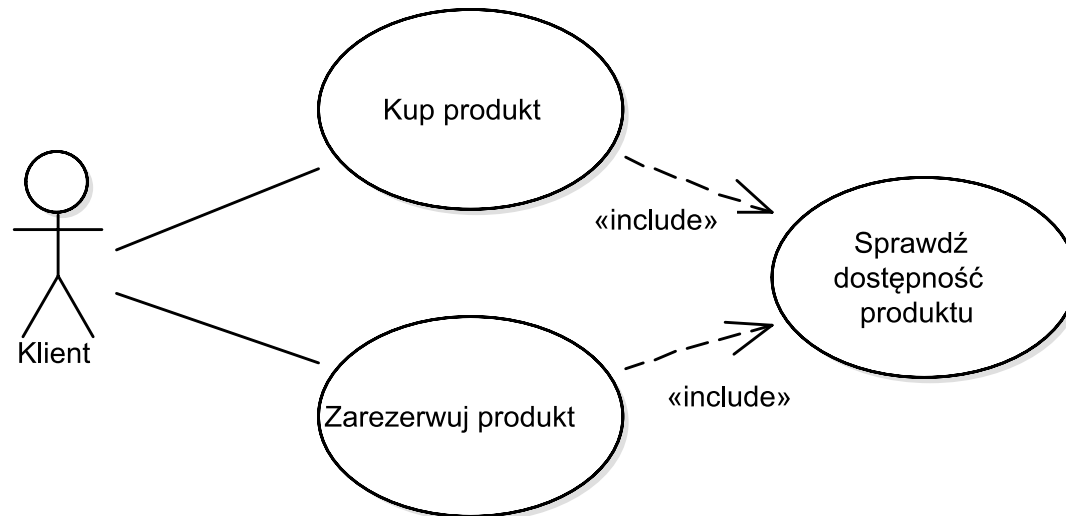


# ZALEŻNOŚĆ ZAWIERANIA





# ZALEŻNOŚĆ ZAWIERANIA



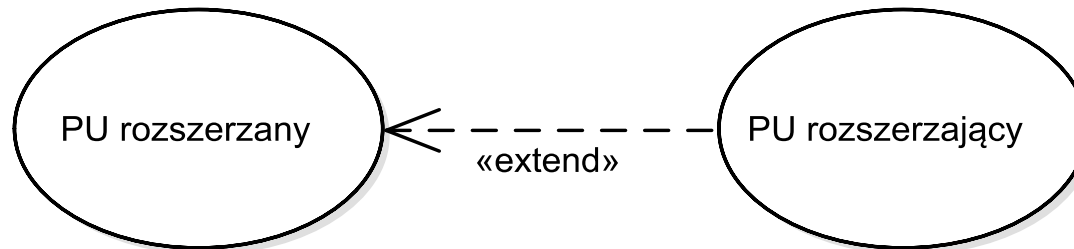
- PU zawierany może być wykorzystywany wielokrotnie.





# ZALEŻNOŚĆ ROZSZERZANIA

PU bazowy

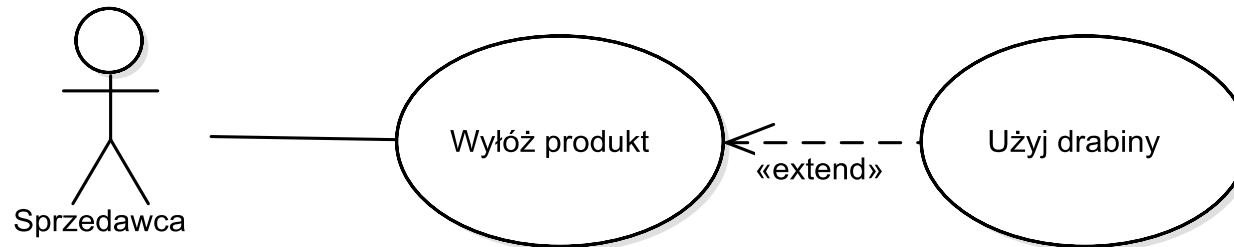


- Jest to związek opcjonalny.
- "PU rozszerzający" **może, ale nie musi** wystąpić podczas wykonywania "PU rozszerzany".
- Kiedy wystąpi? Jeśli zajdzie odpowiedni warunek\*.

\* - "To zależy."



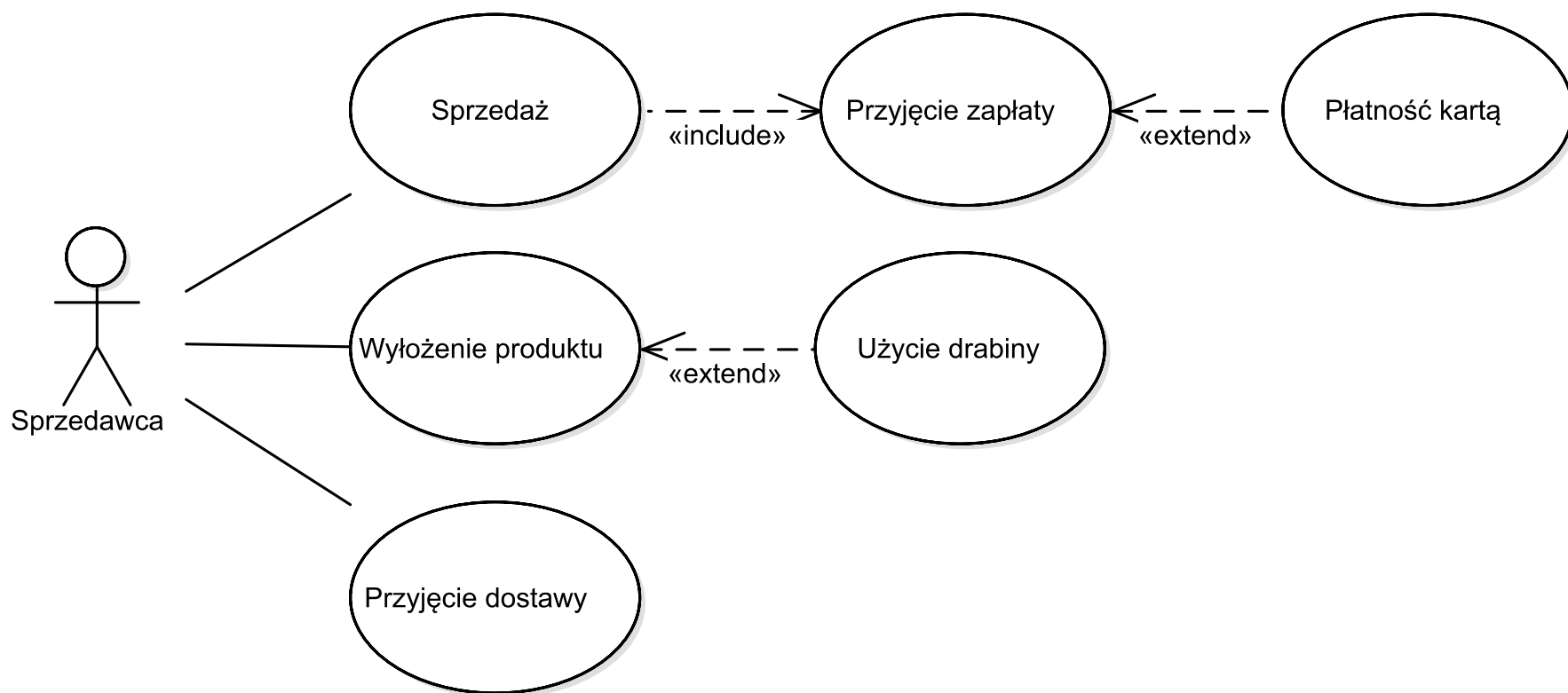
# ZALEŻNOŚĆ ROZSZERZANIA



- Kiedy sprzedawca wyklada produkt, **w niektórych przypadkach** będzie mógł użyć drabiny.

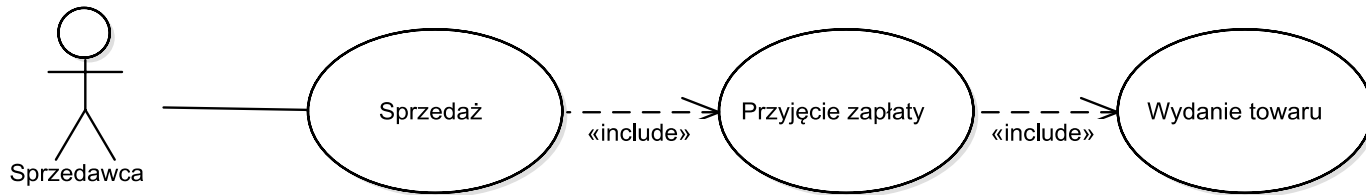


# ZALEŻNOŚĆ ROZSZERZANIA

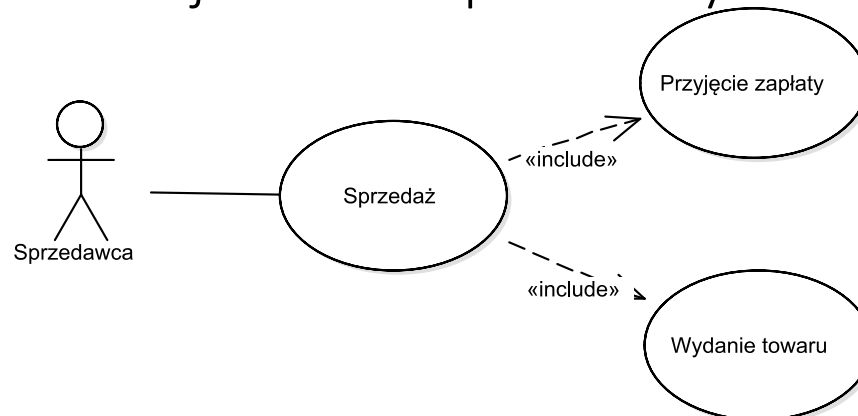




# INTERPRETACJA DIAGRAMU PU

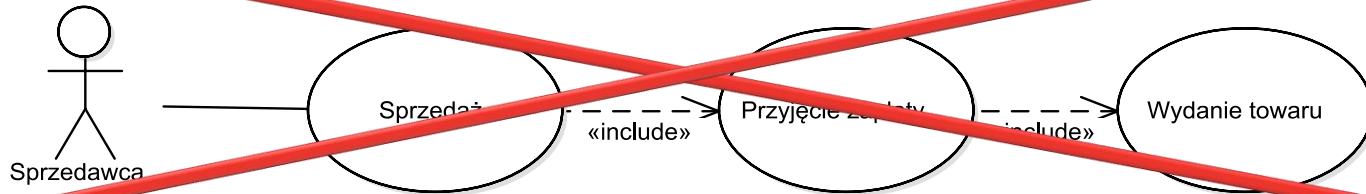


- Diagramy PU **nie służą** do przedstawiania sekwencji akcji (procesów biznesowych).
- Pokazują jakie funkcjonalności posiada system.

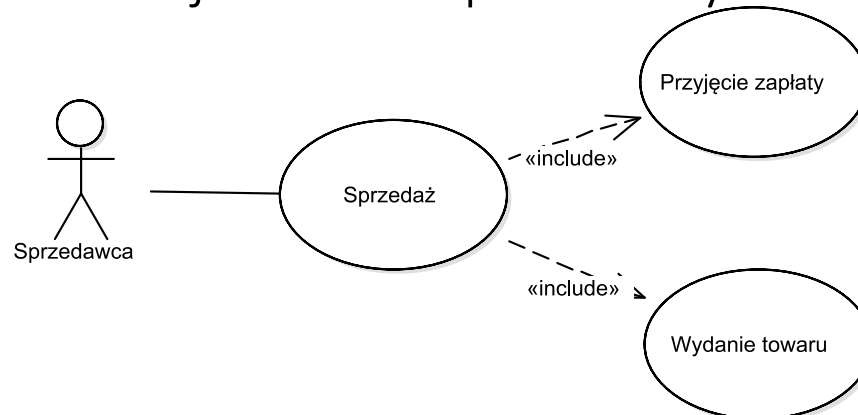




# INTERPRETACJA DIAGRAMU PU



- Diagramy PU **nie służą** do przedstawiania sekwencji akcji (procesów biznesowych).
- Pokazują jakie funkcjonalności posiada system.





# HISTORYJKI UŻYTKOWNIKA

- Pomagają "wychwycić" przypadki użycia.
- Zdefiniowane w formie zdań:

Jako "**użytkownik**" chcę mieć możliwość "**funkcjonalność**", abym mógł "**warunek satysfakcji**".

- Jako *użytkownik niezalogowany* chcę mieć możliwość *przeglądania aktualnych ofert* abym mógł *wybrać ofertę, która mnie interesuje*.
- Jako *administrator* chcę mieć możliwość *dodawania nowych kategorii produktów*, abym mógł *dokonać odpowiedniej hierarchizacji*.



# SPECYFIKACJA PRZYPADKÓW UŻYCIA

Na podstawie:

- opisu systemu,
- dokumentu "system powinien",
- specyfikacji aktorów,
- historyjek użytkownika.

Istotne jest przygotowanie formalnego dokumentu opisującego przypadki użycia.

Warto stosować przyjęty szablon dla dokumentów projektowych.



# DOKUMENTACJA PRZYPADKÓW UŻYCIA

PATRZ ĆWICZENIE 2

Nazwa PU	Pełna nazwa przypadku użycia
Numer:	Numer ID przypadku użycia
Twórca:	Dane osobowe twórcy PU
Poziom ważności:	Poziom istotności PU z perspektywy użytkownika (niski, średni, wysoki, krytyczny)
Typ przypadku użycia:	Określenie przypadku użycia z perspektywy jego złożoności (ogólny, szczegółowy)
Aktorzy:	Lista aktorów biorących udział w danym PU
Krótki opis:	Ogólna charakterystyka przypadku
Warunki wstępne:	Warunki niezbędne do zainicjowania przypadku
Warunki końcowe:	Charakterystyka stanu systemu po realizacji przypadku
Główny przyptyw zdarzeń:	Scenariusz główny przypadku. Wypunktowana lista zdarzeń zachodzących podczas realizacji PU.





# CZEŚĆ PRAKTYCZNA (INSTRUKTAŻ ORAZ ĆWICZENIA)

## **Materiały szkoleniowe – Zeszyt ćwiczeń: Rozdział I – Diagram przypadków użycia.**

Instruktaż:

- Przykład „System aukcji Internetowej”
- Przykład INSPIRE „Strefy Zagrożenia Naturalnego”
- Przykład INSPIRE „Urządzenia do monitorowania środowiska”

Ćwiczenia:

- Ćwiczenie 1 – Specyfikacja wymagań
- Ćwiczenie 2 – Przypadki użycia
- Ćwiczenie 3 – Diagram przypadków użycia

# DIAGRAM AKTYWNOŚCI





# SCENARIUSZ PRZYPADKU UŻYCIA

Opis przypadku użycia musi zawierać główny scenariusz PU, czyli sekwencje kroków, która musi zostać wykonana do osiągnięcia celu.

## PU1: Prezentacja warstwy danych w GIS

Aktor: Użytkownik, Administrator

Scenariusz PU:

1. System GIS prezentuje listę dostępnych warstw.
2. Użytkownik wskazuje warstwę danych.
3. System GIS prezentuję wybraną warstwę danych.



# ROZSZERZENIA PU

Rozszerzenia opisują sytuacje wyjątkowe, które mogą nastąpić w przypadku niespełnienia pewnych warunków np.:

- wybór złej pozycji,
- wpisanie złego hasła,
- błędny kod pocztowy.

## PU1: Wybór warstwy do wyświetlenia

Rozszerzenia PU:

3.1 Użytkownik wybrał już wyświetloną warstwę.

3.1.1 System nie ładuje ponownie tej warstwy.



# WARUNKI POCZĄTKOWE I KOŃCOWE

Warunki początkowe to specyfikacja jaki musi być stan systemu przed wykonaniem PU np.:

- Użytkownik jest zalogowany,
- Mapa jest aktywowana.

Warunki końcowe to końcowy efekt PU:

- Warstwa jest prezentowana.
- Dokonany zostaje zakup przedmiotu.



# OPIS PU

## PU1: Wybór warstwy do wyświetlenia

Aktor: Użytkownik, Administrator

Warunki początkowe:

- Użytkownik

Warunki końcowe:

- Prezentowana jest wybrana warstwa.

Scenariusz PU:

1. System prezentuje listę dostępnych warstw.
2. Użytkownik wskazuje warstwę.
3. System prezentuje wybraną warstwę.

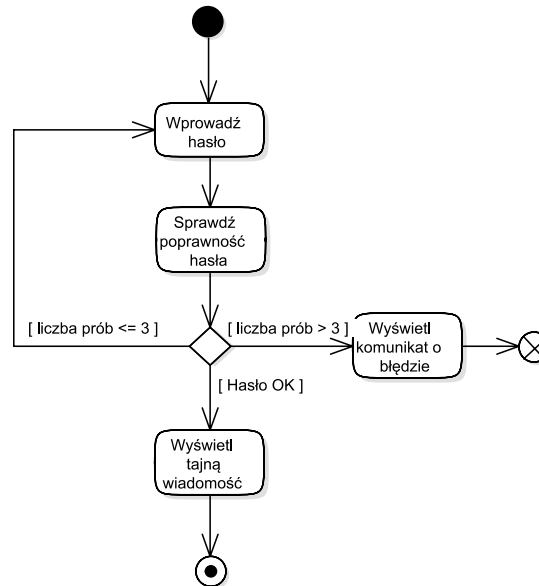
Rozszerzenia PU:

3.1 Użytkownik wybrał już wybraną warstwę.

3.1.1 System nie ładuje ponownie tej warstwy.



# DIAGRAM AKTYWNOŚCI



- Diagram aktywności wykorzystywany jest do modelowania:
  - scenariuszy przypadków użycia,
  - procesów biznesowych,
  - algorytmów.



# CZYNNOŚCI I AKCJE

## CZYNNOŚCI

Generuj raport

Zrób jajecznicę



## AKCJE

$SUMA = A + B$

Rozbij jajko

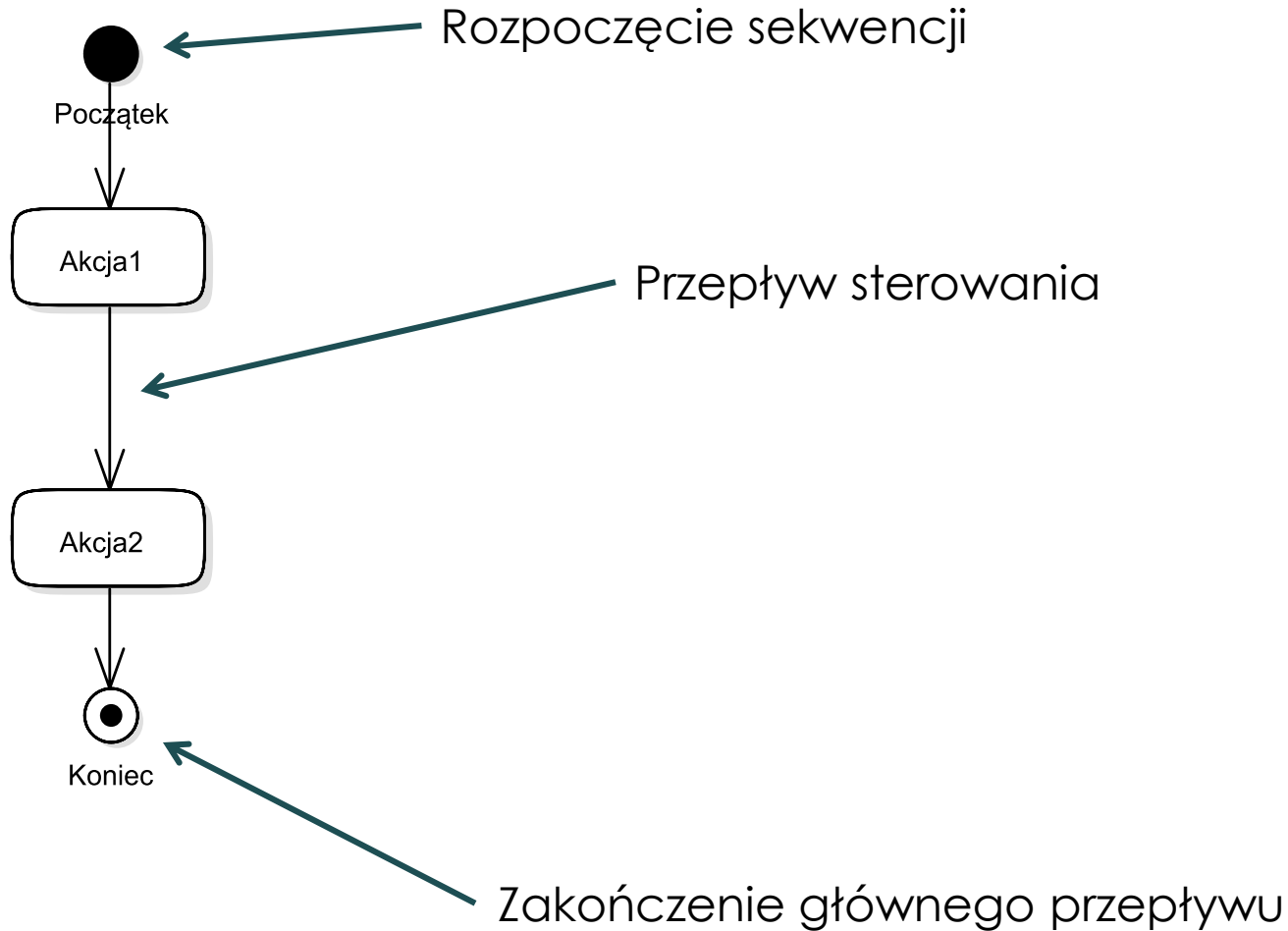
- Złożone procesy biznesowe.
- Może być dekomponowana.

- Atomowa operacja.
- Nie może być zdekompnowana.





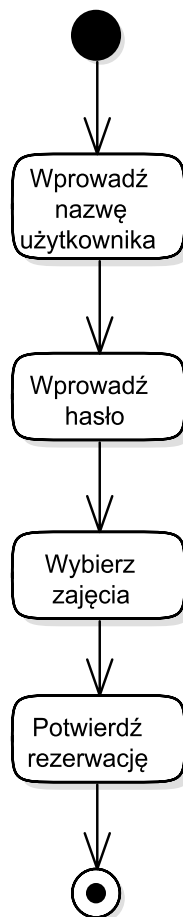
# DIAGRAM AKTYWNOŚCI





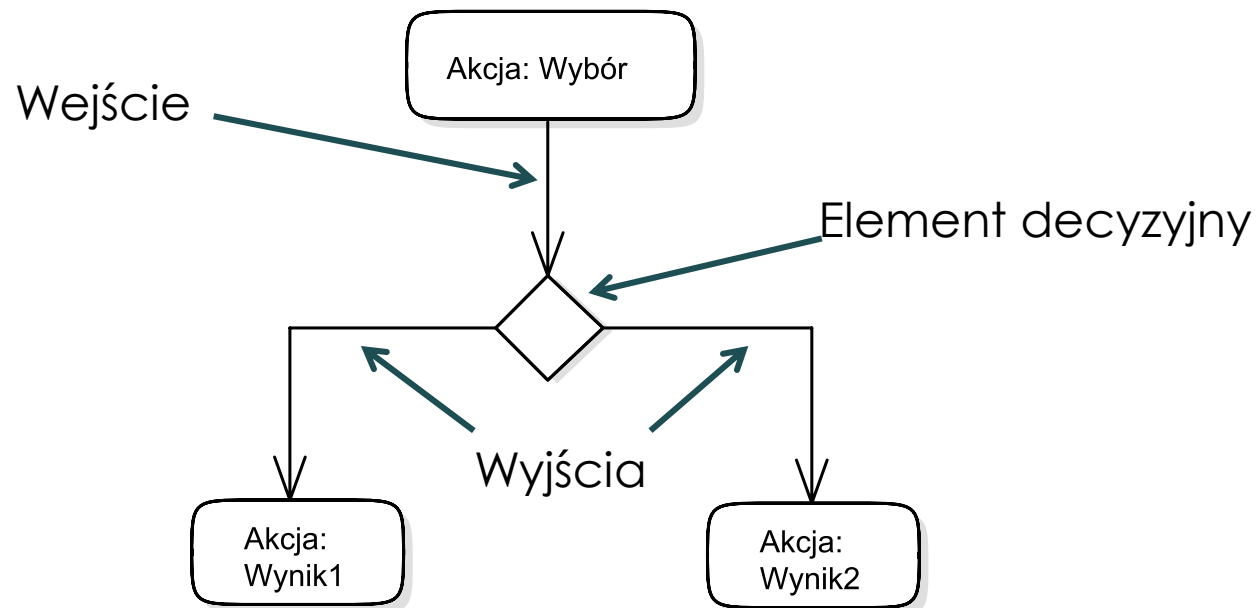
# DIAGRAM AKTYWNOŚCI

Proces: Zapis na zajęcia.



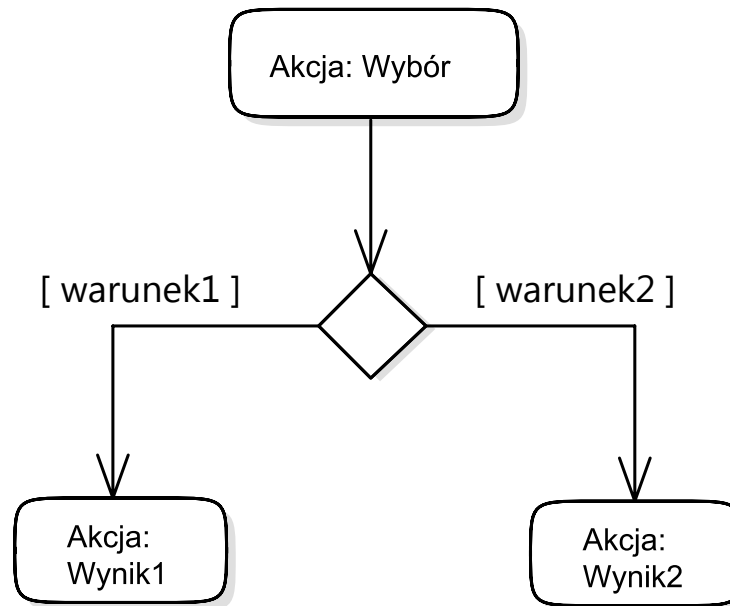


# DECYZJA



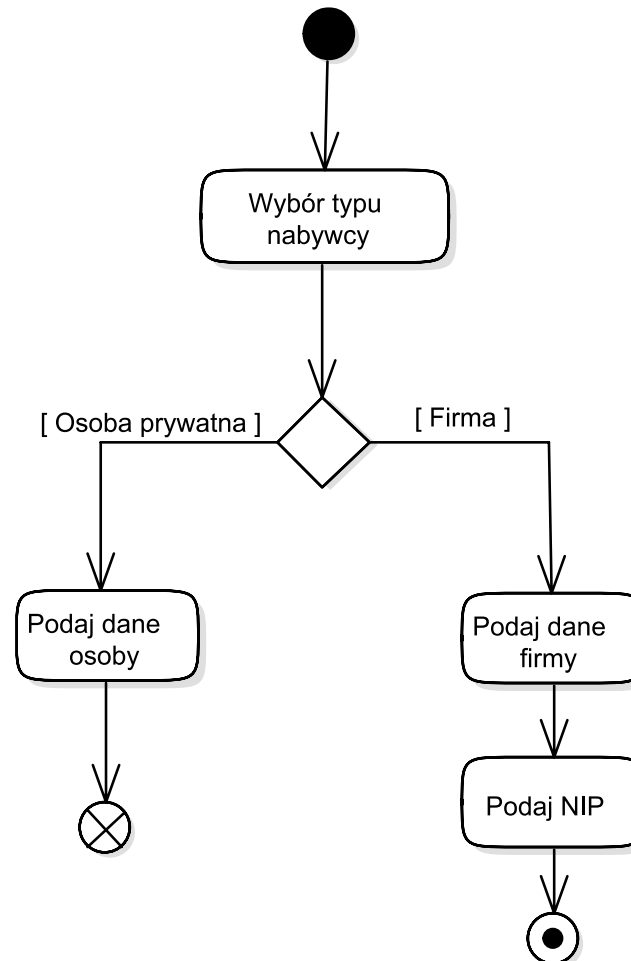


# DECYZJA



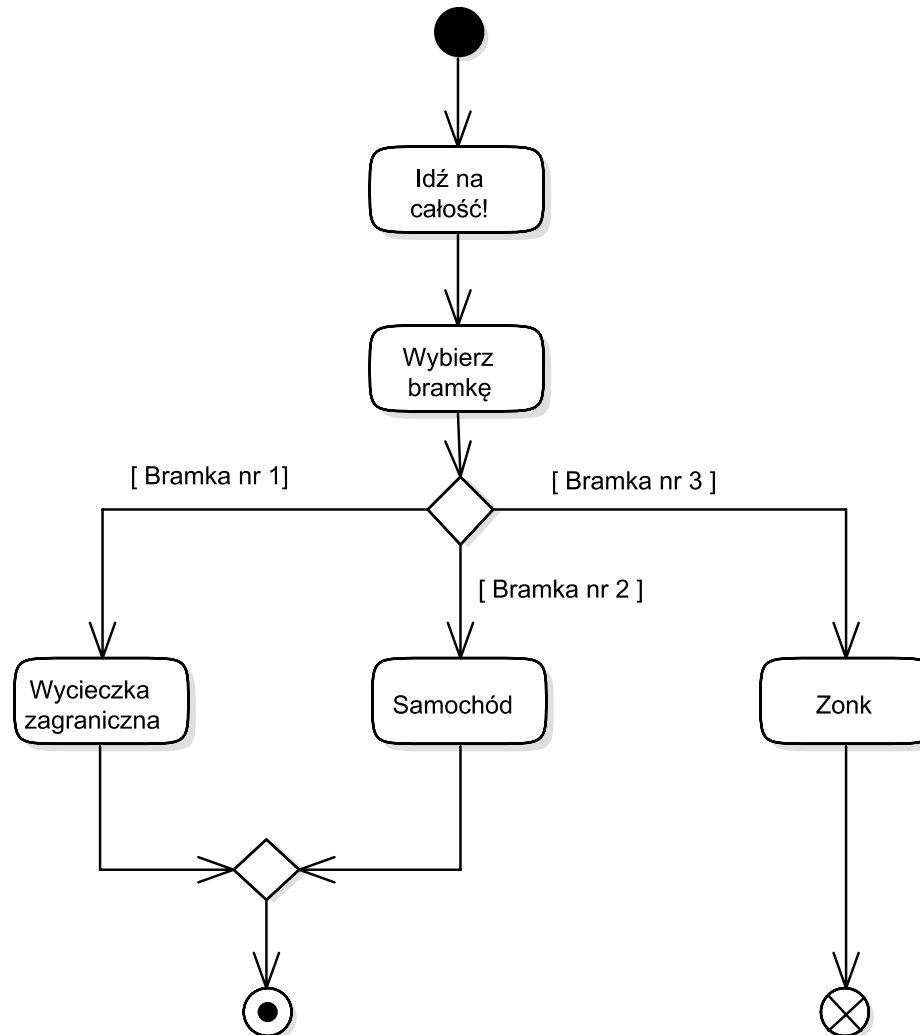


# DECYZJA



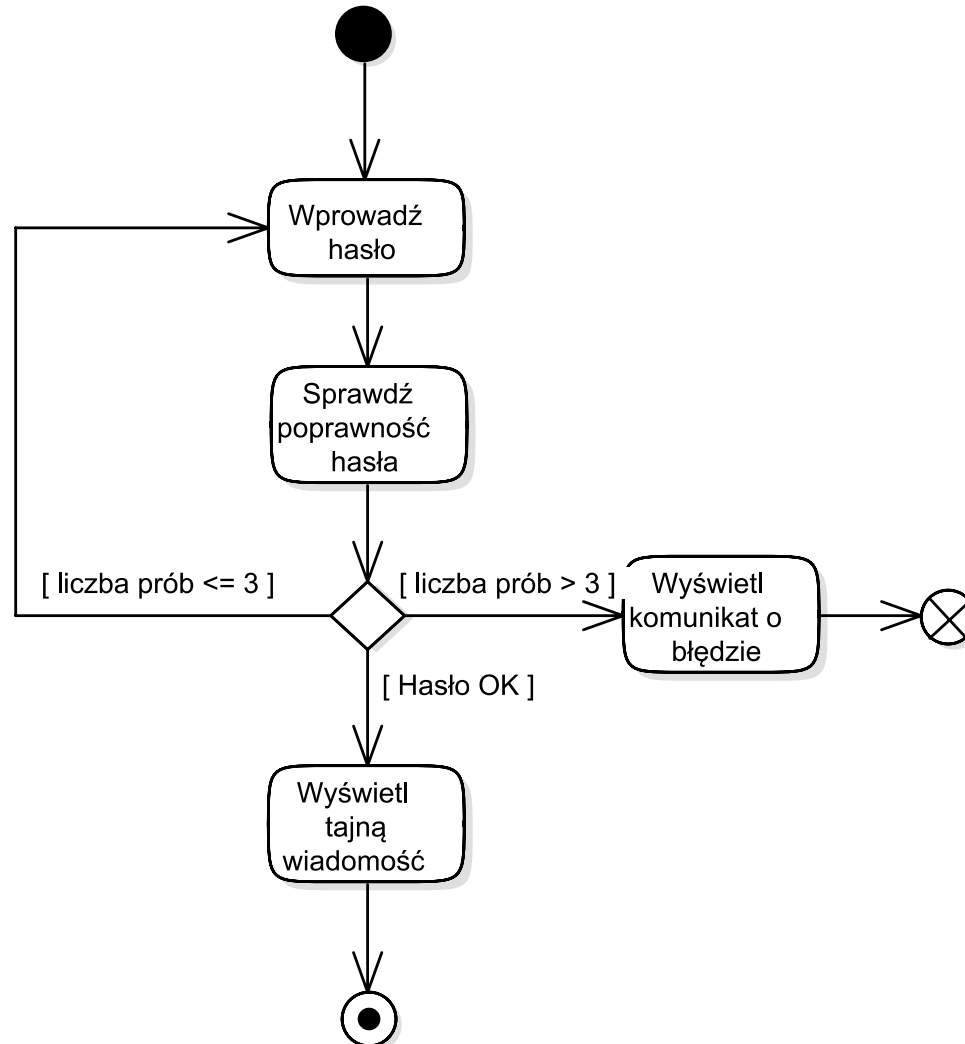


# DECYZJA





# DECYZJA

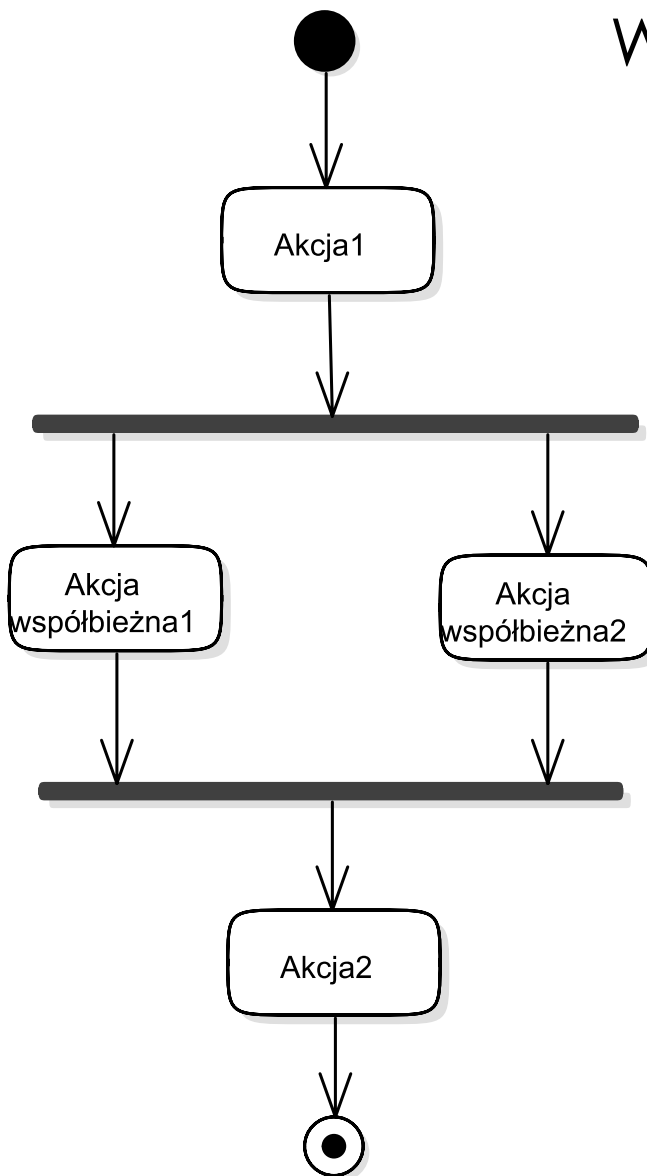




# WSPÓŁBIEŻNOŚĆ

Rozwidlenie

Scalenie

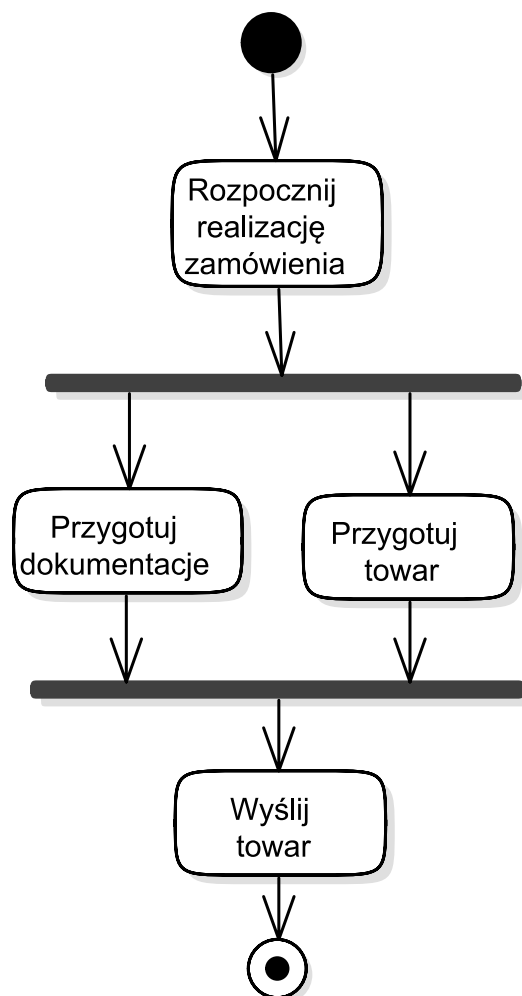


Czynności realizowane równolegle (w tym samym czasie)





# WSPÓŁBIEŻNOŚĆ





# CZEŚĆ PRAKTYCZNA (INSTRUKTAŻ ORAZ ĆWICZENIA)

## **Materiały szkoleniowe – Zeszyt ćwiczeń: Rozdział II – Diagram aktywności.**

Instruktaż:

- Przykład „Rejestracja użytkownika w sklepie internetowym”
- Przykład z INSPIRE

Ćwiczenia:

- Ćwiczenie 1 – Uzupelnienie diagramu
- Ćwiczenie 2 – Wykonanie diagramu
- Ćwiczenie 3 – Współbieżność

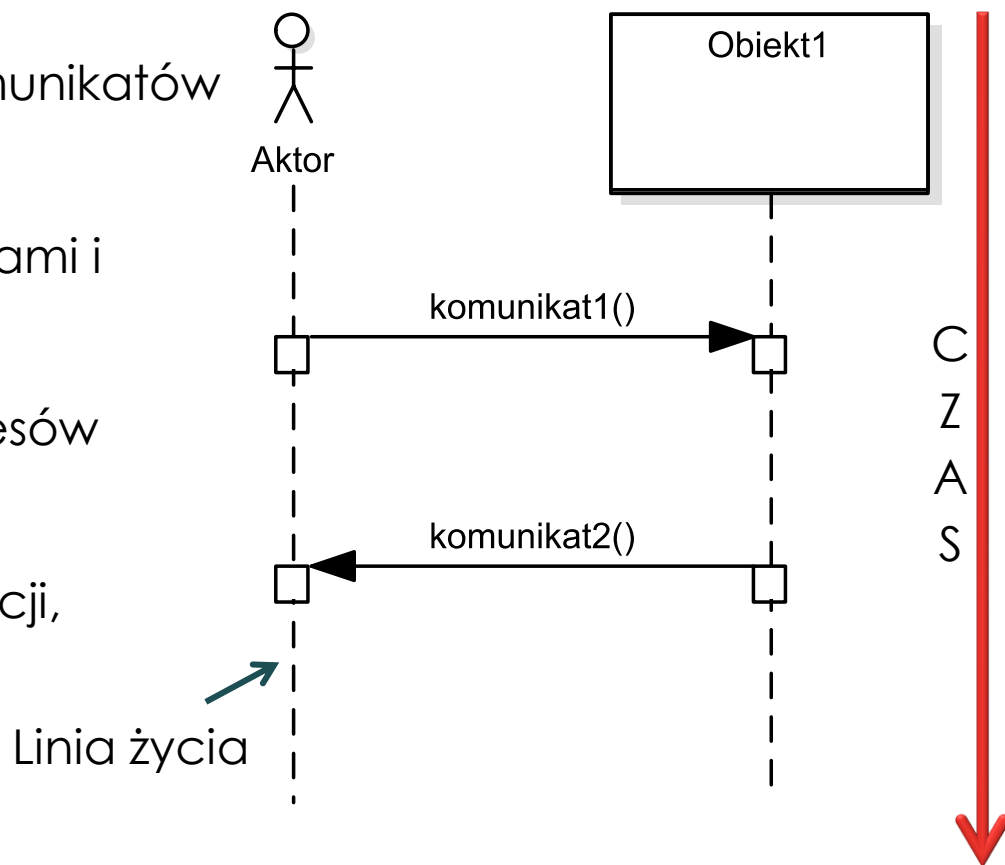


# DIAGRAM INTERAKCJI



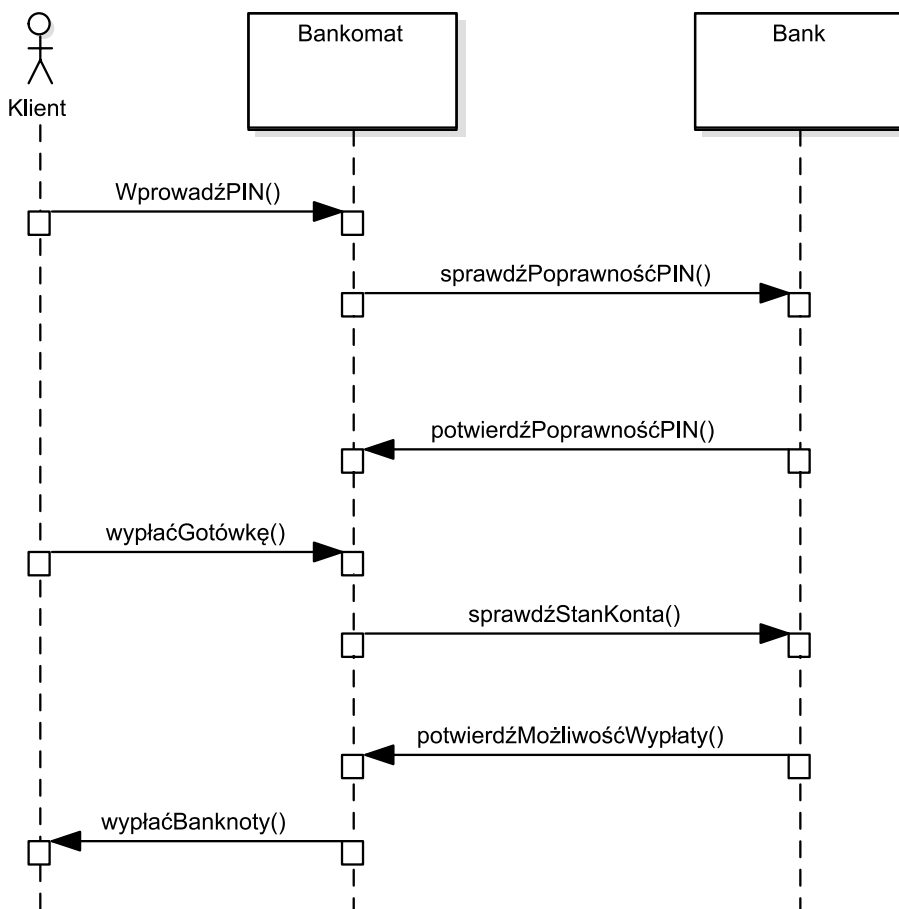
# DIAGRAM INTERAKCJI

- Opisuje sekwencje wysyłania komunikatów między klasyfikatorami w czasie.
- Wymiana informacji między aktorami i elementami systemu.
- Pozwala na przedstawianie procesów biznesowych.
- Linia życia: rola uczestnika interakcji, jaką pełni w czasie jej trwania. Reprezentuje współuczestnika interakcji i czas jego istnienia podczas realizacji scenariusza.





# DIAGRAM INTERAKCJI (WERSJA UPROSZCZONA)

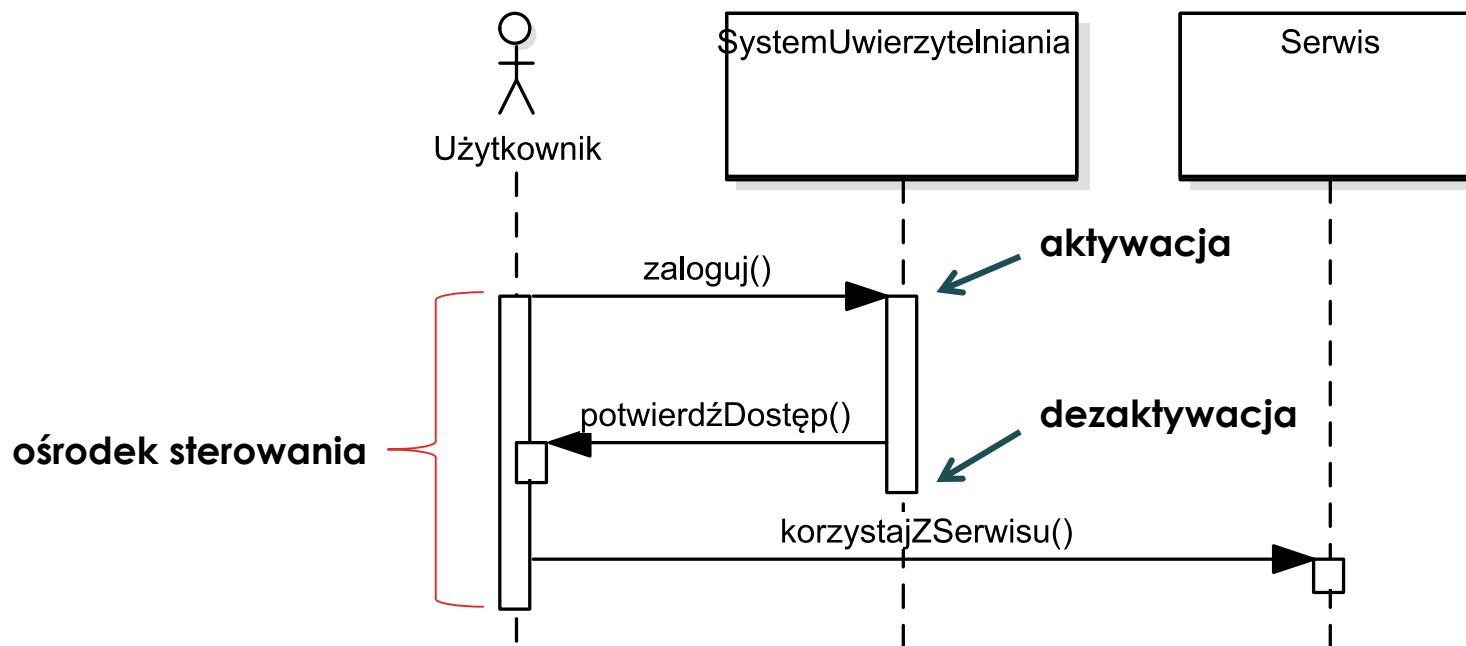






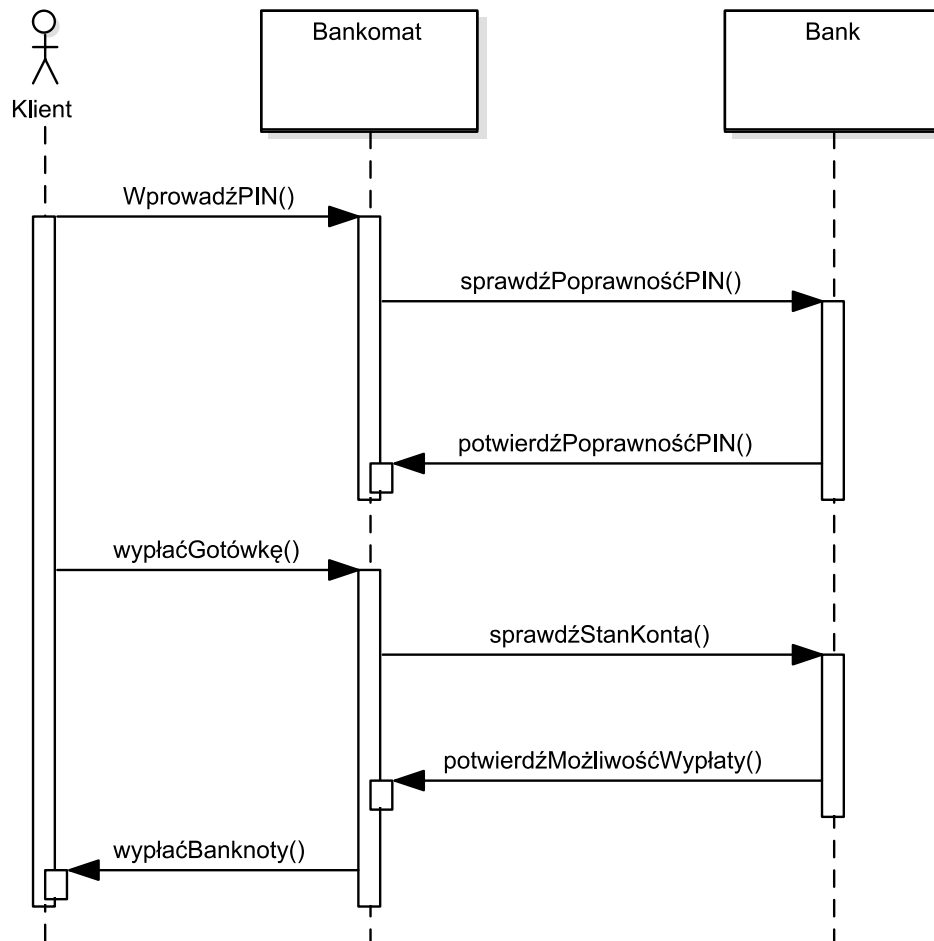
# OŚRODEK STEROWANIA

- Wskazanie okresu aktywacji instancji klasyfikatora.



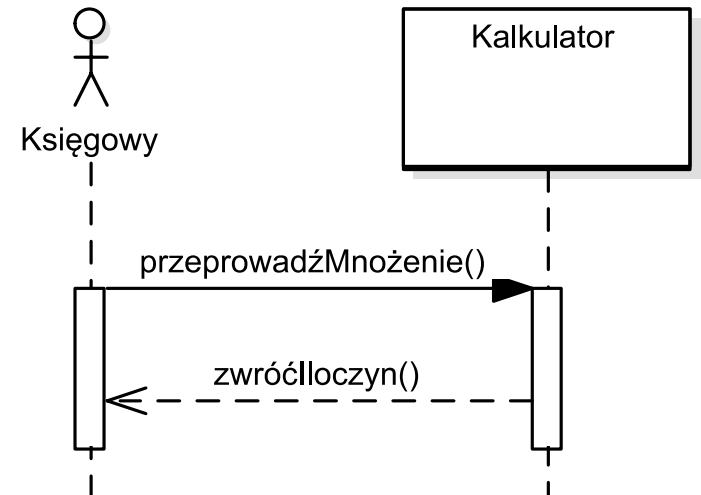
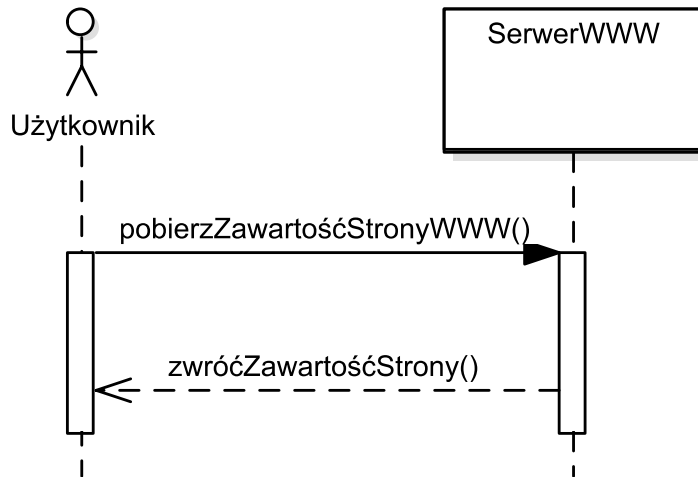


# DIAGRAM INTERAKCJI





# KOMUNIKAT ZWROTNY





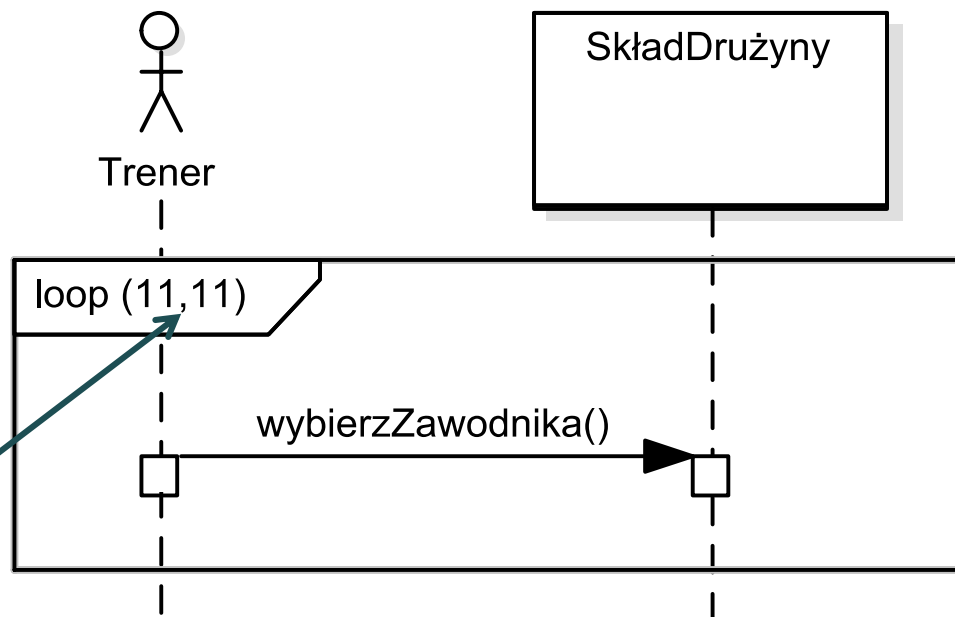


# FRAGMENTY

- Bardziej zaawansowane elementy diagramu interakcji.
- Wyodrębniona część diagramu zawierająca fragment wykonywany wg. określonych reguł.
- Pozwalają na definiowanie m.in.:
  - Pętli ("LOOP"),
  - Alternatywnego przebiegu ("ALT"),
  - Krytyczne fragmenty.



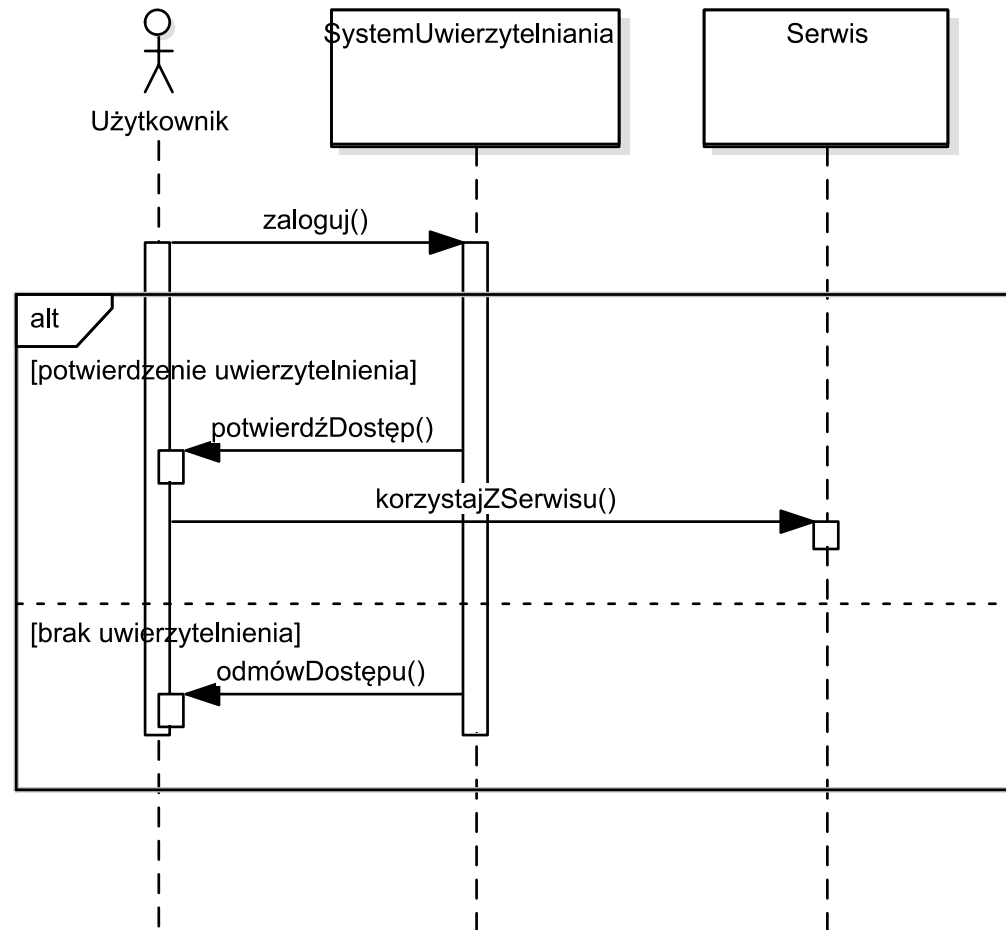
## FRAGMENT "LOOP"



Min i max liczba powtórzeń, którą należy wykonać, aby warunek został spełniony



# FRAGMENT "ALT"





# CZEŚĆ PRAKTYCZNA (INSTRUKTAŻ ORAZ ĆWICZENIA)

## **Materiały szkoleniowe – Zeszyt ćwiczeń: Rozdział III – Diagram interakcji.**

Instruktaż:

- Przykład „Wypłata z bankomatu”

Ćwiczenia:

- Ćwiczenie 1 – Uzupełnienie diagramu
- Ćwiczenie 2 – Stwórz diagram

# DIAGRAM KLAS





## MODELOWANIE CD.

- Systemy informatyczne rozwiązują problemy operując na danych.
- W pewnych sytuacjach konieczne jest opisywanie fragmentów rzeczywistości.
- Konieczne jest zastosowanie narzędzia pozwalające na przedstawienie deklaratywnych elementów dziedziny przedmiotowej.
- Modelowanie obiektowe jako bardziej naturalny sposób modelowania zaobserwowanych konceptów.



# MODELOWANIE OBIEKTOWE

- Jedno z podejść modelowania rzeczywistości spopularyzowane razem z rozwojem języków programowania zorientowanych obiektowo.
- Podstawowe pojęcia:
  - **Obiekt** – element posiadający stan i zachowanie.
  - **Klasa obiektu** – definicja zbioru obiektów. Określa ich dopuszczalny stan i zachowanie.
  - **Instancjonowanie** – utworzenie nowego obiektu na podstawie klasy.





# INSTANCJONOWANIE

Klasa „Samochód”

Atrybuty:

- Marka
- Model
- Rok produkcji
- Kolor

Operacje:

- Jedź
- Hamuj

**instancjonowanie**

Obiekt „Mój Bugatti Veyron”

*instancja klasy „Samochód”*



Pola:

- Marka - Bugatti
- Model – Veyron 16.4
- Rok produkcji – 2010
- Kolor - czarny

Metody:

- Jedź
- Hamuj





# DIAGRAM KLAS UML

- Diagram przedstawiający strukturę klas w systemie i ich wzajemne związku.
- Jeden z najczęściej wykorzystywanych diagramów UML.
- Wykorzystywany do opisywania struktury danych (INSPIRE) i definiowania schematów pojęciowych oraz aplikacyjnych.
- Podstawowym klasyfikatorem jest Klasa.



# KLASA W DIAGRAMACH UML

Nazwa

NazwaKlasy

Atrybuty

+ atrybut1  
+ atrybut2

dopuszczalny stan

Operacje

+ operacja1()  
+ operacja2()

dopuszczalne zachowanie



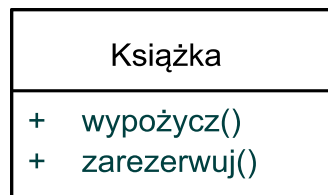
# KLASA W DIAGRAMACH UML



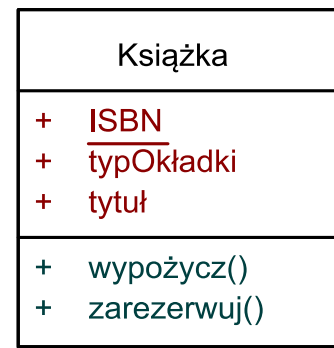
brak atrybutów  
i operacji



tylko atrybuty



tylko operacje



atrybuty  
i operacje



# ZWIĄZKI MIĘDZY KLASAMI

- Realizacja PU "Wypożyczanie książki" z perspektywy statycznej.

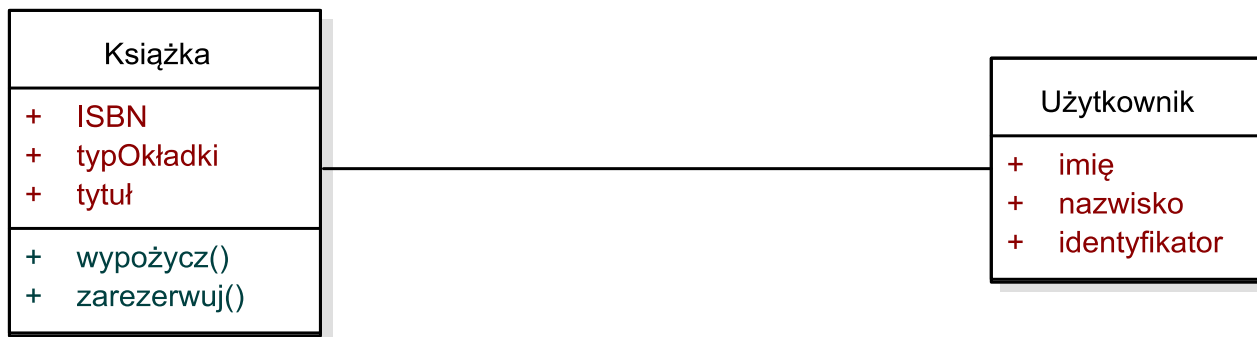
Książka
+ ISBN
+ typOkładki
+ tytuł
+ wypożycz()
+ zarezerwuj()

Użytkownik
+ imię
+ nazwisko
+ identyfikator

- W jaki sposób odnotować, że Użytkownik może mieć wypożyczoną Książkę?



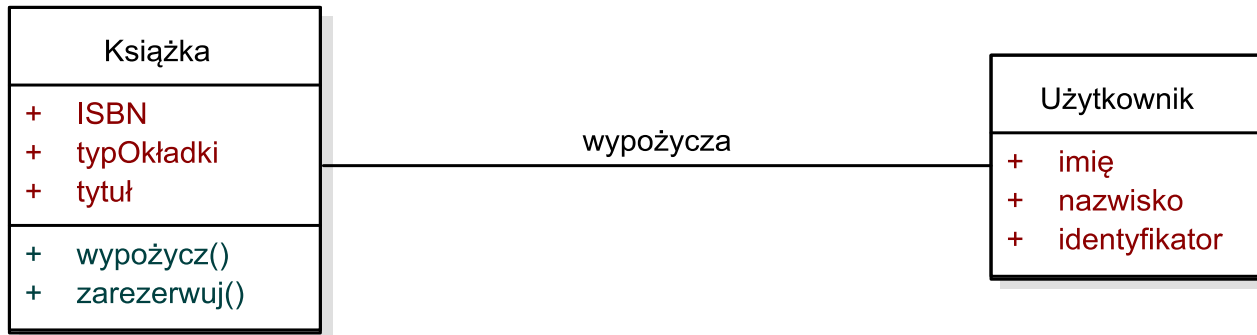
# ASOCJACJA



- Związek dwukierunkowy.
- Obie klasy są świadome związku.
- W praktyce rzadko stosowane.



# ASOCJACJA



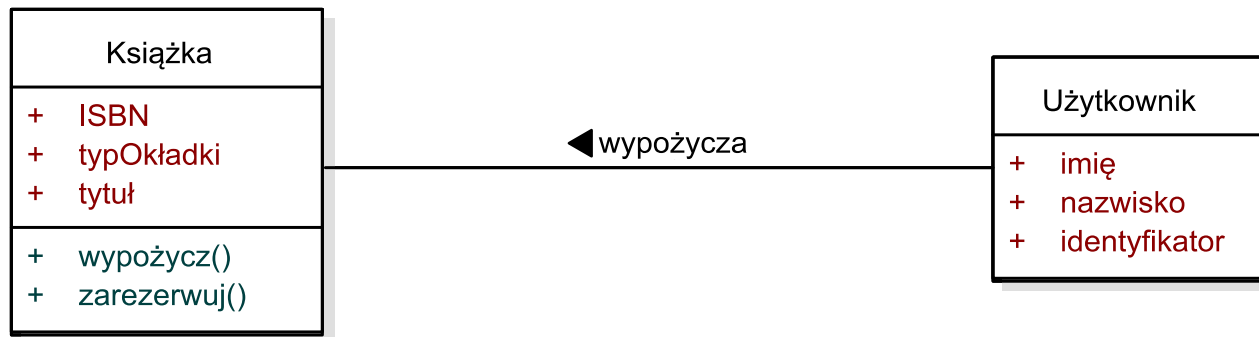


# ASOCJACJA





# ASOCJACJA

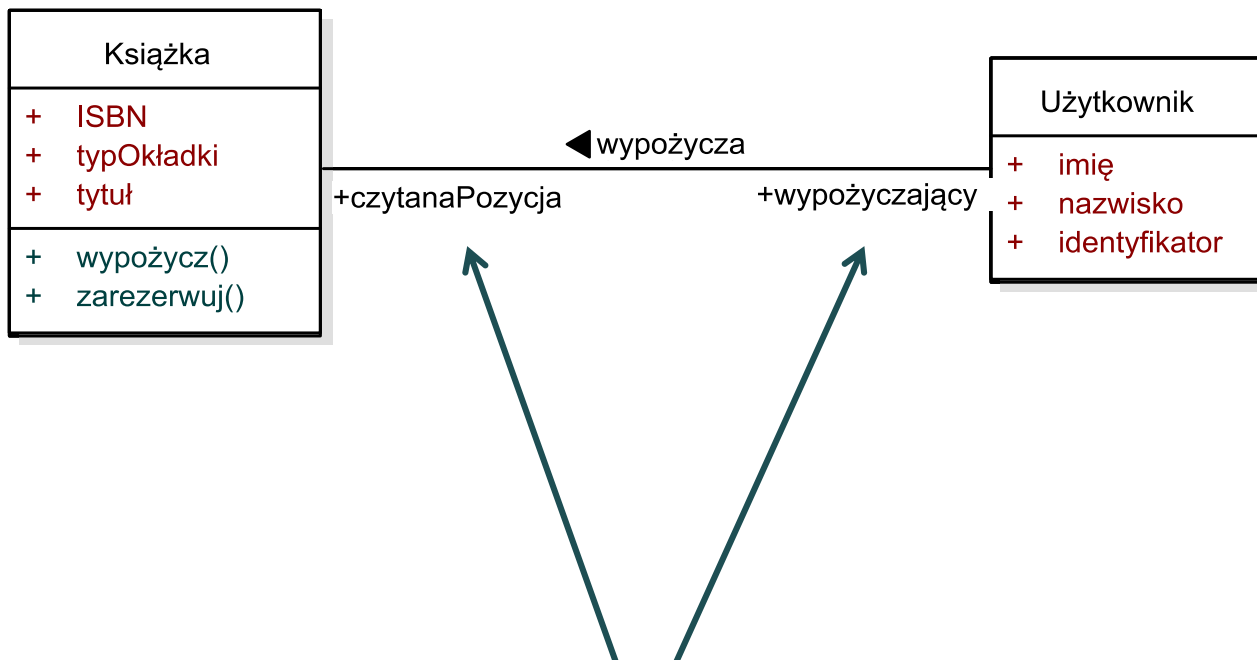


Użytkownik wypożycza książkę.





# ASOCJACJA



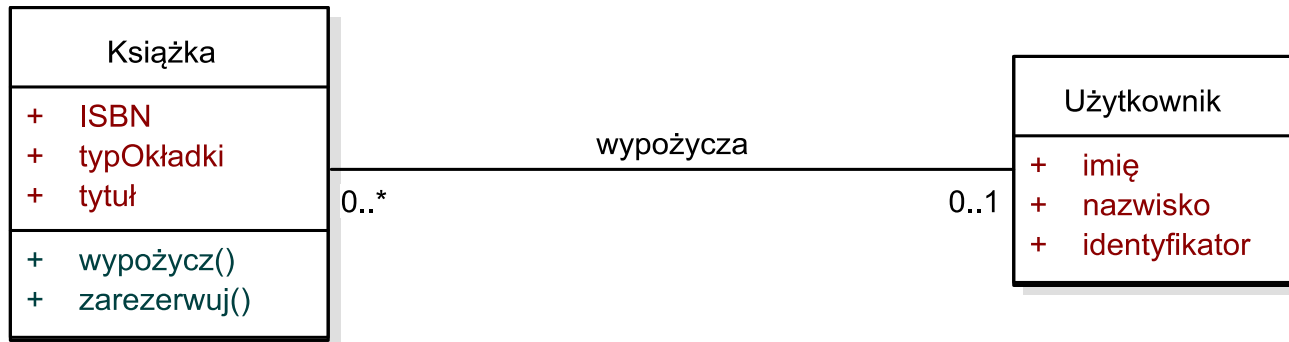
wskazanie roli w asocjacji



# LICZEBNOŚĆ

- Ograniczenie nakładane na końce asocjacji specyfikujące dopuszczalną liczbę obiektów biorących udział w danym związku.

→  
*Książka może być wypożyczona przez jednego Użytkownika lub nikogo.*



←  
*Użytkownik może wypożyczyć wiele książek lub żadną.*

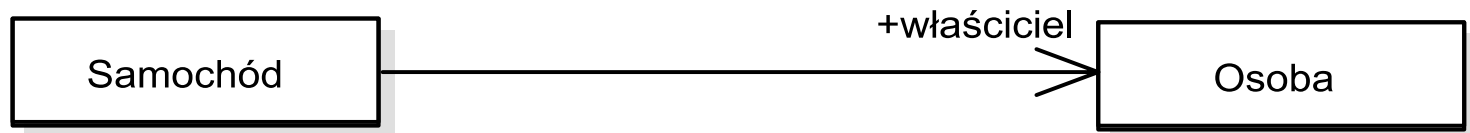


# LICZEBNOŚĆ

1	dokładnie jeden
1..*	jeden lub wiele
0..*	zero lub wiele
0..1	zero lub jeden
*	wiele
N	dokładnie n
1..n	jeden do n
0..n	zero do n
n..m	od n do m
n..*	n lub wiele



# NAWIGACJA



- Związek jednokierunkowy.
- Tylko jedna klasa jest świadoma związku.
- Często wykorzystywana w praktyce.
- Możemy "przejsć" od Samochodu do Osoby:  
"Posiadając nr tablicy rejestracyjnej jesteśmy w stanie odszukać właściciela samochodu."



# NAWIGACJA

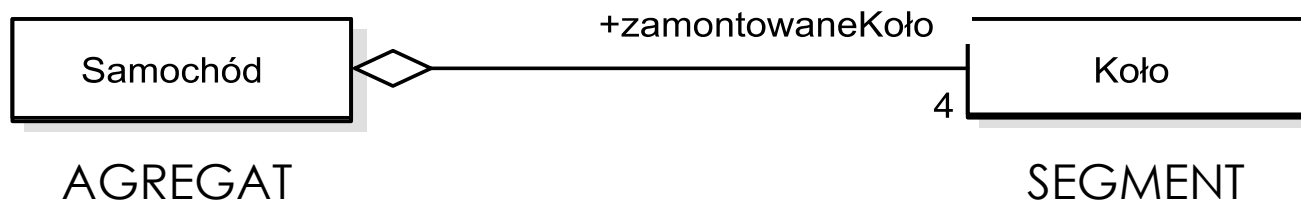


*"referencja"*



# AGREGACJA

- Związek opisujący relacje typu całość-część między klasami.

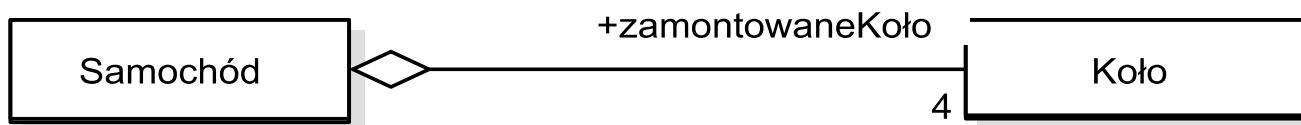


- Dostępne są 2 typy agregacji:
  - agregacja częściowa,
  - agregacja całościowa (kompozycja).



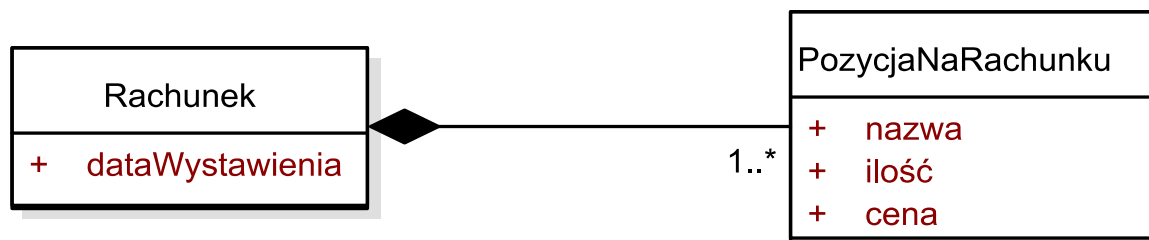
# AGREGACJA

- Agregacja częściowa:



Segment (Koło) może istnieć bez Agregatu.

- Kompozycja:

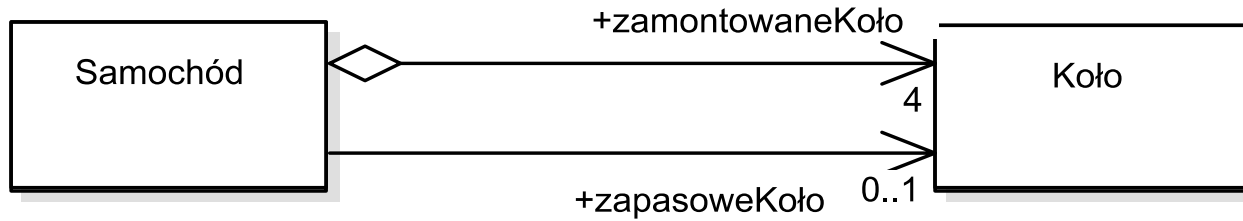


Segment nie może istnieć bez Agregatu.



# AGREGACJA I NAWIGACJA

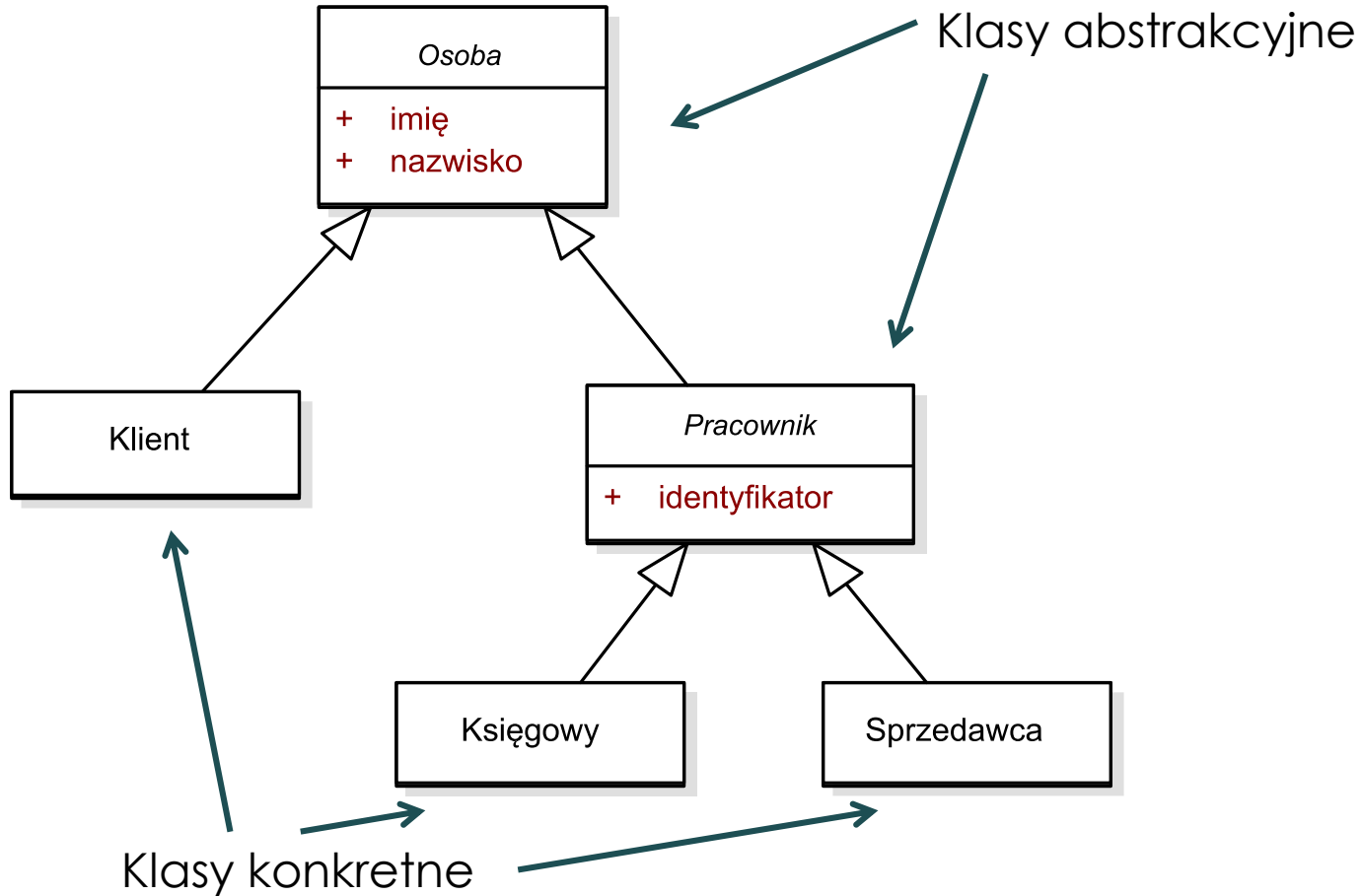
- W praktyce:







# GENERALIZACJA





# GENERALIZACJA

- Klasy abstrakcyjne nie mogą być instancjonowane.
- Obiekty powstałe na podstawie klas konkretnych przejmują atrybuty klas nadrzędnych:
  - Sprzedawca posiada atrybuty:
    - Identyfikator (z klasy Pracownik),
    - Imię (z klasy Osoba),
    - Nazwisko (z klasy Osoba).



# TYPY ATRYBUTÓW

- Atrybuty klas mogą przyjąć dodatkowe ograniczenia związane z określeniem **zbioru dozwolonych wartości**.

Osoba
+ imię: string + nazwisko: string + wiek: integer + dataRejestracji: date



# TYPY ATRYBUTÓW

- Typy atrybutów wywodzą się z typów danych.
- Przykładowe typy proste:

string	łańcuch znaków, tekst	"Kowalski"
integer	Liczba całkowita	123
single	Liczba zmiennoprzecinkowa pojedynczej precyzji	3.141592
date	data	2016-09-05
dateTime	data i czas	2016-09-05 09:00:00
boolean	Wartość logiczna – prawda lub fałsz.	true/false



# TYPY ATRYBUTÓW

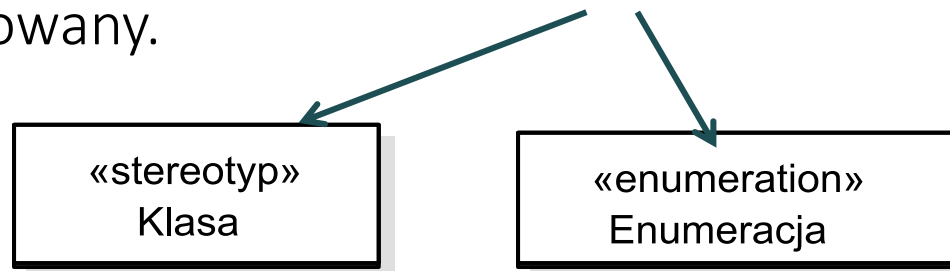
- Dodatkowo możliwe jest określenie krotności występowania atrybutów, poprzez umieszczenie jej w nawiasach kwadratowych.

Klasa
+ nazwa: string + alternatywneNazwy: string [1..*]



# STEREOTYPY

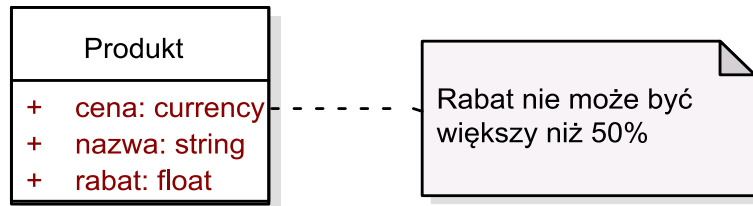
- Dodatkowa informacja (etykieta) dostarczająca informacje w jaki sposób klasyfikator UML powinien zostać zinterpretowany.





# OGRANICZENIA

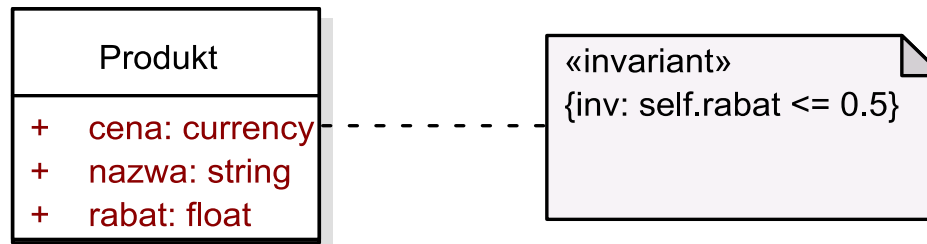
- UML umożliwia nakładanie dodatkowych ograniczeń na klasy.
- Ograniczenia są definiowane w formie warunków dla obiektów danej klasy:
  - "rabat nie może być większy niż 50%",
  - "atrybut nazwisko nie może być pusty".
- Ograniczenia mogą być definiowane za pomocą:
  - języka naturalnego,
  - języka OCL.





# JĘZYK OCL

- Object Constraint Language – deklaratywny język definiowania ograniczeń.
- Formalizuje zapis ograniczeń dla klas – posiada rozbudowaną składnię.
- Pozwala automatyzować proces sprawdzania spełnialności ograniczeń.







# JĘZYK OCL - PRZYKŁADY

Język naturalny:

*Geometry and representativePoint cannot both be empty.*

Zapis w OCL:

inv: geometry ->notEmpty() or  
representativePoint ->notEmpty()

Przykład pochodzi, ze specyfikacji  
„Urządzenia do monitorowania  
środowiska”, strona 34.



# JĘZYK OCL - PRZYKŁADY

Język naturalny:

If geometry has no value, a reference to a spatial object needs to be provided.

Zapis w OCL:

```
inv: self.geometry->isEmpty() implies  
self.spatialObject->notEmpty()
```

Przykład pochodzi, ze specyfikacji  
„Rozmieszczenie gatunków”, strona 32.



## JĘZYK OCL - PRZYKŁADY

Język naturalny:

Either the qualitative value or the quantitative value must be completed.

Zapis w OCL:

```
inv: self.qualitativeValue.isEmpty() implies  
self.quantitativeValue.notEmpty() and  
self.quantitativeValue.isEmpty() implies  
self.qualitativeValue.notEmpty()
```

Przykład pochodzi, ze specyfikacji  
„Strefy zagrożenia naturalnego”, strona 44.



# CZEŚĆ PRAKTYCZNA (INSTRUKTAŻ ORAZ ĆWICZENIA)

## **Materiały szkoleniowe – Zeszyt ćwiczeń: Rozdział IV – Diagram klas.**

Instruktaż:

- Przykład „Księgarnia”,
- Przykład pochodzący z INSPIRE

Ćwiczenia:

- Ćwiczenie 1 – Podstawy
- Ćwiczenie 2 – Uogólnienia
- Ćwiczenie 3 – Zaawansowane
- Ćwiczenie 4 – Uzupełnienie katalogu obiektów na podstawie diagramu klas
- Ćwiczenie 5 – Diagram klas na podstawie katalogu obiektów

# DIAGRAM PAKIETÓW





# PAKIET

- Mechanizm służący do organizowania klasyfikatorów w grupy.
- Może zawierać:
  - Kompletne diagramy,
  - Podsystemy,
  - Modele.



- Wykorzystywany do porządkowania diagramów.



# DIAGRAM PAKIETÓW

## Pracownicy

- + Księgowy
- + Monter
- + Pracownik
- + Sprzedawca

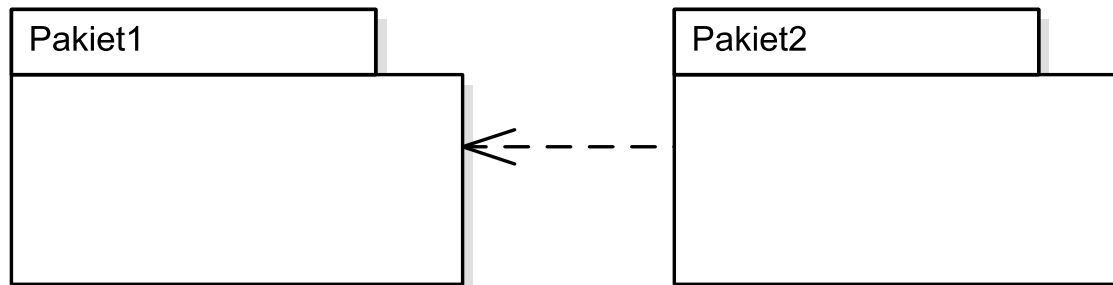
## Moduły systemu

- + BazaFaktur
- + Faktura
- + KatalogProduktów



# ZALEŻNOŚCI

- Wskazanie relacji pomiędzy pakietami – jeden pakiet jest zależny od drugiego – wykorzystuje jego elementy.







# CZEŚĆ PRAKTYCZNA (INSTRUKTAŻ ORAZ ĆWICZENIA)

## **Materiały szkoleniowe – Zeszyt ćwiczeń: Rozdział V – Diagram pakietów.**

Instruktaż:

- Przykład podstawowy „Firma”,
- Przykład diagramu pakietów ze specyfikacji INSPIRE

# SCHEMATY APLIKACYJNE INSPIRE



# INSPIRE

- Jaka jest rola UML w INSPIRE?
- Systemy Informacji Geograficznej są podstawą budowy Infrastruktury Danych i Infrastruktury Informacji Przestrzennej.
- Jednym z najważniejszych wymagań związanych z realizacją zadań INSPIRE jest **zapewnienie interoperacyjności**.

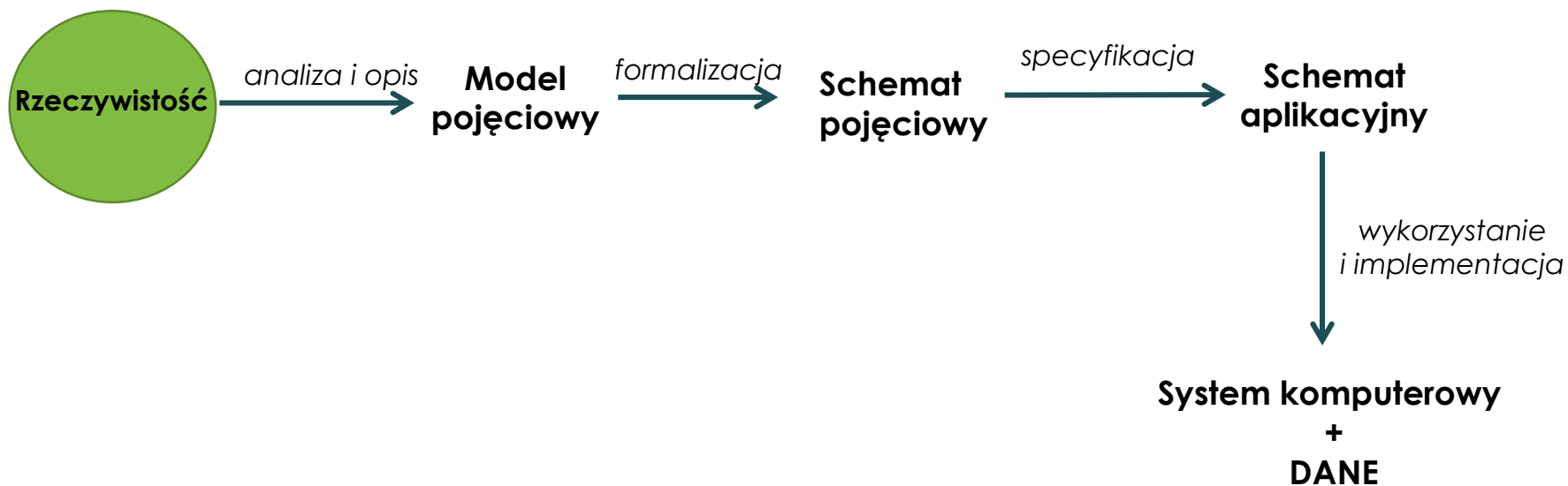


# INTEROPERACYJNOŚĆ

- Zdolność do współdziałania poprzez zapewnienie komunikacji i możliwości dokonywania transferu danych pomiędzy systemami.
- Istotna w procesie budowy Infrastruktury Danych Przestrzennych.
- Realizowana poprzez wdrażanie norm ISO z serii 19100.
- Zapewniana m.in. przez korzystanie ze wspólnych schematów pojęciowych.



# INTEROPERACYJNOŚĆ







# MODEL POJĘCIOWY

- Opis wycinka rzeczywistości – fragmentu pewnej dziedziny.
- Abstrakcyjny opis rzeczywistych obiektów.
- Nie ma formalnej reprezentacji – tworzony najczęściej przez analityka w postaci notatek.
- Skupia się na wymaganiach – pomocne jest wykorzystanie diagramów PU.



# SCHEMAT POJĘCIOWY

- Formalna reprezentacja modelu pojęciowego.
- Przygotowywany z wykorzystaniem odpowiednich narzędzi – języka schematu pojęciowego.
- Przykładem takiego języka jest UML.



# SCHEMAT APLIKACYJNY

- Schemat pojęciowy dla specyficznego zakresu przedmiotowego.
- Wykorzystywany przez różne aplikacje.
- Zawiera definicje klas obiektów i strukturę informacji geograficznej.
- Stosowane w celu osiągnięcia powszechnego i poprawnego rozumienia danych – jednoznaczną ich interpretację.
- **Definiuje zawartość danych.**





# NORMY ISO 19100

- Normy ISO 19103 i ISO 1909 definiują zasady tworzenia schematów aplikacyjnych.
- Porządkują i określają jakie elementy UML mają zostać wykorzystane do modelowania schematów aplikacyjnych.
- Dostarczają rozszerzeń dla UML.



# NORMA ISO19103

- Definiuje "Język schematu pojęciowego".
- Wykorzystanie języka UML do reprezentacji informacji geograficznej.
- Definiuje podstawowe typy danych jakie mogą przyjąć atrybuty klas definiujących fragmenty informacji przestrzennej.
- Definiuje strukturę pakietów.
- Zawiera sposób wykorzystania języka OCL.



# NORMA ISO19109

- Definiuje w jaki sposób tworzyć schematy aplikacyjne przy pomocy języka schematów pojęciowych zdefiniowanego w ISO 19103.
- Definiuje podstawowe stereotypy i elementy w postaci metamodelu GFM (General Feature Model) np.:
  - Klasa obiektu przestrzennego (FeatureType) reprezentująca elementy świata rzeczywistego.



# TWORZENIE SCHEMATU APLIKACYJNEGO

1. Przeprowadzenie analizy wymagań dla danych z pewnej dziedziny (modelowanie rzeczywistości).
2. Opracowanie modelu pojęciowego przy pomocy konceptów (metaklas) zdefiniowanych w General Feature Model (ISO 19109):
  - Identyfikacja obiektów przestrzennych,
  - Identyfikacja właściwości,
  - Identyfikacja ograniczeń.
3. Utworzenie schematu aplikacyjnego z wykorzystaniem UML.
4. Integracja z innymi ustandaryzowanymi schematami.



# SCHEMAT APLIKACYJNY W JĘZYKU UML

- Wykorzystywane są przede wszystkim:
  - Diagram klas,
  - Diagram pakietów.
- Tworzenie schematu polega na odpowiednim wykorzystaniu narzędzi UML z uwzględnieniem:
  - Stereotypów,
  - Zdefiniowanych typów.
- Podstawowym elementem jest **obiekt przestrzenny**.



# OBIEKT PRZESTRZENNY

«FeatureType»  
ObiektPrzestrzenny

«FeatureType»  
Budynek

«FeatureType»  
ObszarChroniony

Wykorzystanie stereotypu <<FeatureType>>



# OBIEKT PRZESTRZENNY

- Definiujemy strukturę danych, więc nie ma potrzeby definiowania operacji dla klas.
- Wymagane jest dobranie odpowiednich typów dla atrybutów zdefiniowanych w ISO 19103.

«FeatureType» Zabytek
+ identyfikator: Integer + nazwa: CharacterString + dataPowstania: Date



# TYPY ATRYBUTÓW

- Do dyspozycji mamy typy proste takie jak:

CharacterString	Wartość tekstowa
Integer	Liczba całkowita
Real	Liczba rzeczywista
Decimal	Ułamek dziesiętny
Date	Data
DateTime	Data i czas
Boolean	Prawda/Fałsz





# TYPY ATRYBUTÓW

- Jednostki miar:

Angle	Kąt
AngularVelocity	Prędkość kątowna
Area	Powierzchnia
Currency	Waluta
Distance	Odległość
Length	Długość
Scale	Skala
Time	Czas
Velocity	Prędkość
Volume	Objętość
Weight	Waga



# TYPY ATRYBUTÓW PRZESTRZENNYCH

- Dodatkowo norma ISO 19107 definiuje następujące typy geometryczne m.in.:

GM_Point	Punkt
GM_LineString	Łamana
GM_Surface	Powierzchnia
GM_Curve	Krzywa
GM_Circle	Okrąg



# TYPY ATRYBUTÓW GEOMETRYCZNYCH

«FeatureType» Zabytek
+ identyfikator: Integer + nazwa: CharacterString + dataPowstania: Date + lokalizacja: GM_Point



# ZŁOŻONE TYPY DANYCH

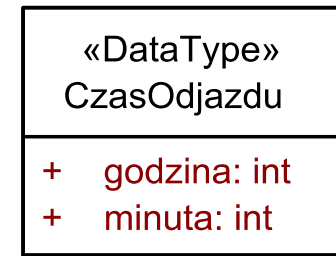
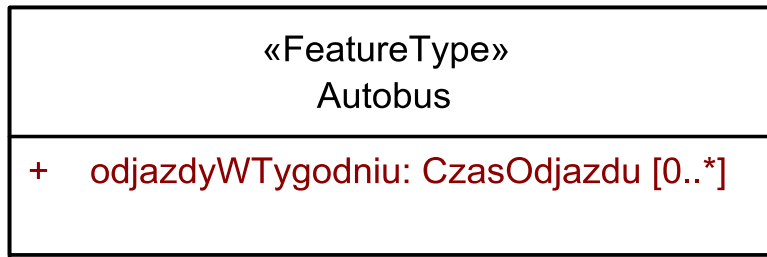
- W razie potrzeby możliwe jest utworzenie własnego, złożonego typu danych.



- Różnica pomiędzy wartością a referencją (nawigacją).



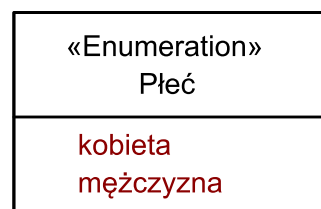
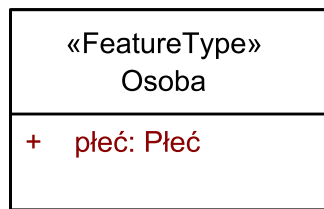
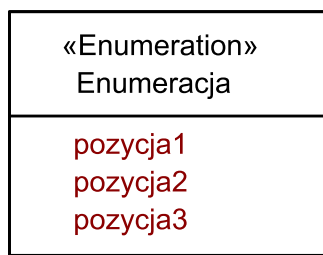
# ZŁOŻONE TYPY DANYCH





# ENUMERACJE

- Typ wyliczeniowy, którego nie można rozszerzać.





# LISTY KODOWE

- Typ wyliczeniowy, który może być zmieniony np. poprzez zmodyfikowanie słownika w bazie danych.

«CodeList» ListaKodowa
+ pozycja1
+ pozycja2
+ pozycja3

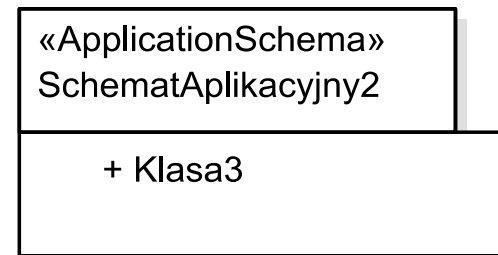
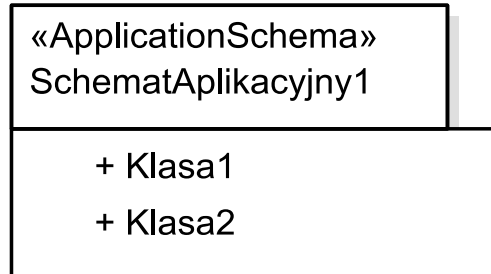
«CodeList» ObsługiwanyJęzyk
+ polski
+ angielski
+ niemiecki

«FeatureType» AutomatBiletowy
+ wybranyJęzyk: ObsługiwanyJęzyk



# ORGANIZACJA PAKIETÓW

- Pakiety ze zdefiniowanymi schematami aplikacyjnymi otrzymują stereotyp <<ApplicationSchema>>.







## STEREOTYP <<VOIDABLE>>

- Stosowany dla atrybutów, które nie są obowiązkowe (pomimo wymaganej liczności np. 1). Zamiast wartości atrybutu zostanie zapisana informacja o przyczynie braku wartości.

unpopulated	Nie stosuje się
missing	Brak danych
template	Tymczasowy brak danych
unknown	Wartość nieznana
withheld	Wartość zastrzeżona



# POZOSTAŁE STOSOWANE STEREOTYPY

- <<union>> - strukturalny typ danych, dla którego dokładnie jeden z atrybutów musi wystąpić,
- <<type>> - w normach grupy ISO 19100 – stereotyp przeznaczony do określenia dziedziny obiektów będących egzemplarzami tej klasy,
- <<placeholder>> - klasa, która tymczasowo „trzyma miejsce” dla klasy która zostanie zdefiniowana w przyszłości. Zazwyczaj posiada co najmniej definicję.



# CZEŚĆ PRAKTYCZNA (INSTRUKTAŻ ORAZ ĆWICZENIA)

## **Materiały szkoleniowe – Zeszyt ćwiczeń: Rozdział VI – Schematy aplikacyjne INSPIRE.**

Instruktaż:

- Analiza schematu aplikacyjnego „Regiony biogeograficzne”
- Analiza schematu aplikacyjnego dla tematu „Rozmieszczenie gatunków”