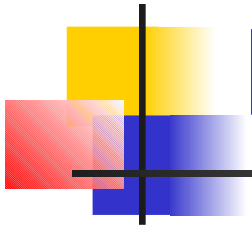
UML

zastosowanie i projektowanie
w języku UML



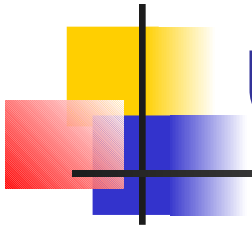
Plan

- Czym jest UML
- Diagramy przypadków użycia
- Diagramy sekwencji
- Diagramy klas
- Diagramy stanów
- Przykładowe programy
- Visual Studio a UML

Czym jest UML

- **UML** jest graficznym językiem pozwalającym na opisywanie systemów (nie tylko informatycznych, lecz głównie do tego celu jest on używany) za pomocą diagramów i słów. UML jest standardem dzięki czemu diagramy w nim zbudowane będą zrozumiałe dla każdego, kto miał z nim styczność.
  Język ten jest bardzo rozbudowany. Na szczęście w większości przypadków używa się tylko jego
  drobnego podzbioru. Nie sposób będzie zaprezentować wszystkich jego możliwości.

Diagramy przypadków użycia

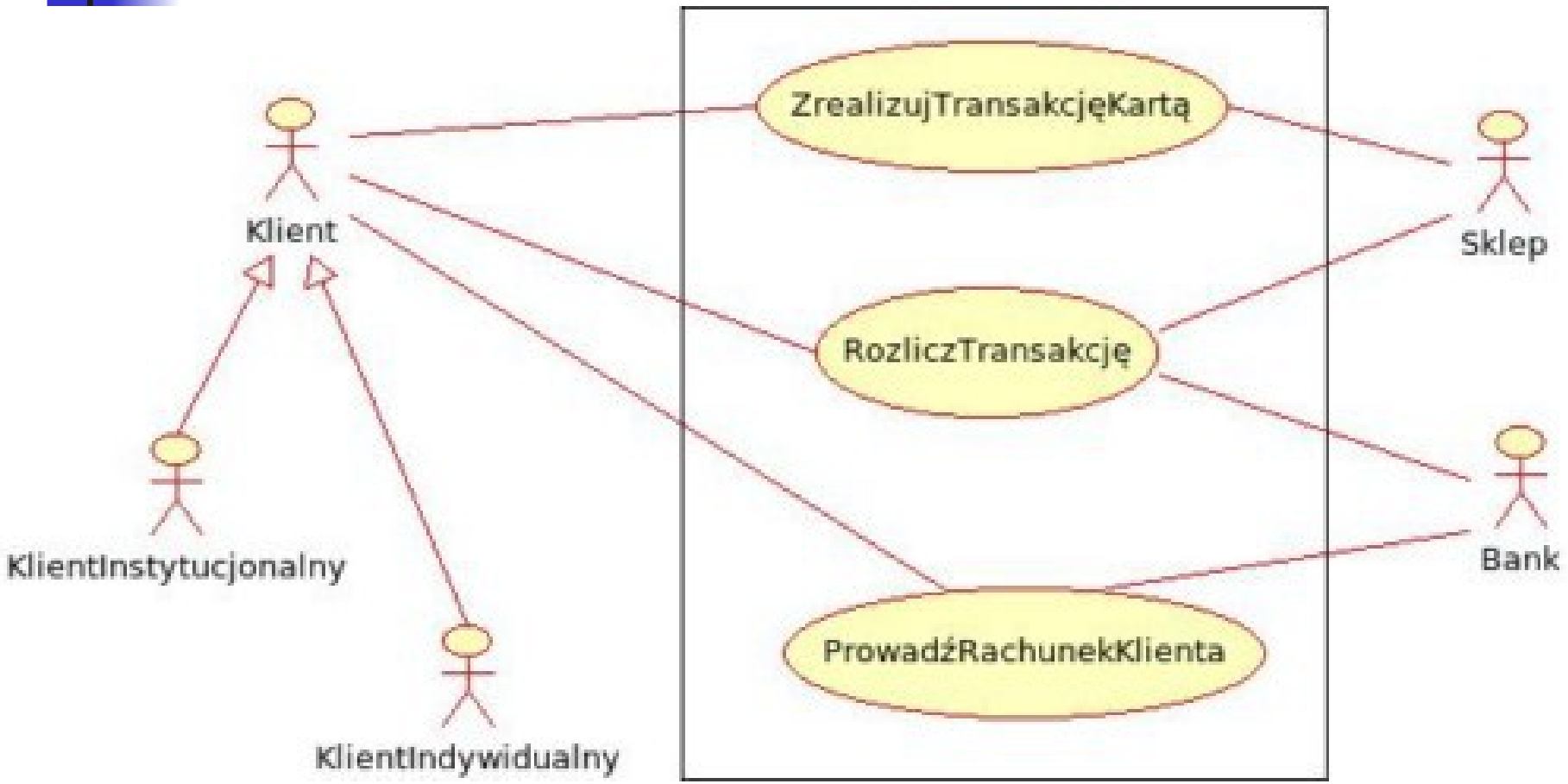


- definiują granice modelowanego systemu
- określają jego kontekst
- wymieniają użytkowników systemu i jednostki zewnętrzne
- przedstawiają funkcje dostępne dla użytkowników
- określają powiązania i zależności pomiędzy nimi

Diagramy przypadków użycia

- W diagramach przypadków użycia UML pojawiają się:
  - aktorzy – reprezentujący osoby, grupy osób lub elementy otoczenia systemu wchodzące w interakcje z systemem i odgrywający pewną rolę (jedna osoba może być reprezentowana przez kilku aktorów odgrywających różne role)
  - przypadki użycia – oznaczające zbiór scenariuszy związanych z konkretnym wykorzystaniem systemu
  - klasyfikatory – do których przynależą konkretne przypadki użycia
  - powiązania – łączące aktorów z przypadkami użycia, w których odgrywają swe role

Diagramy przypadków użycia



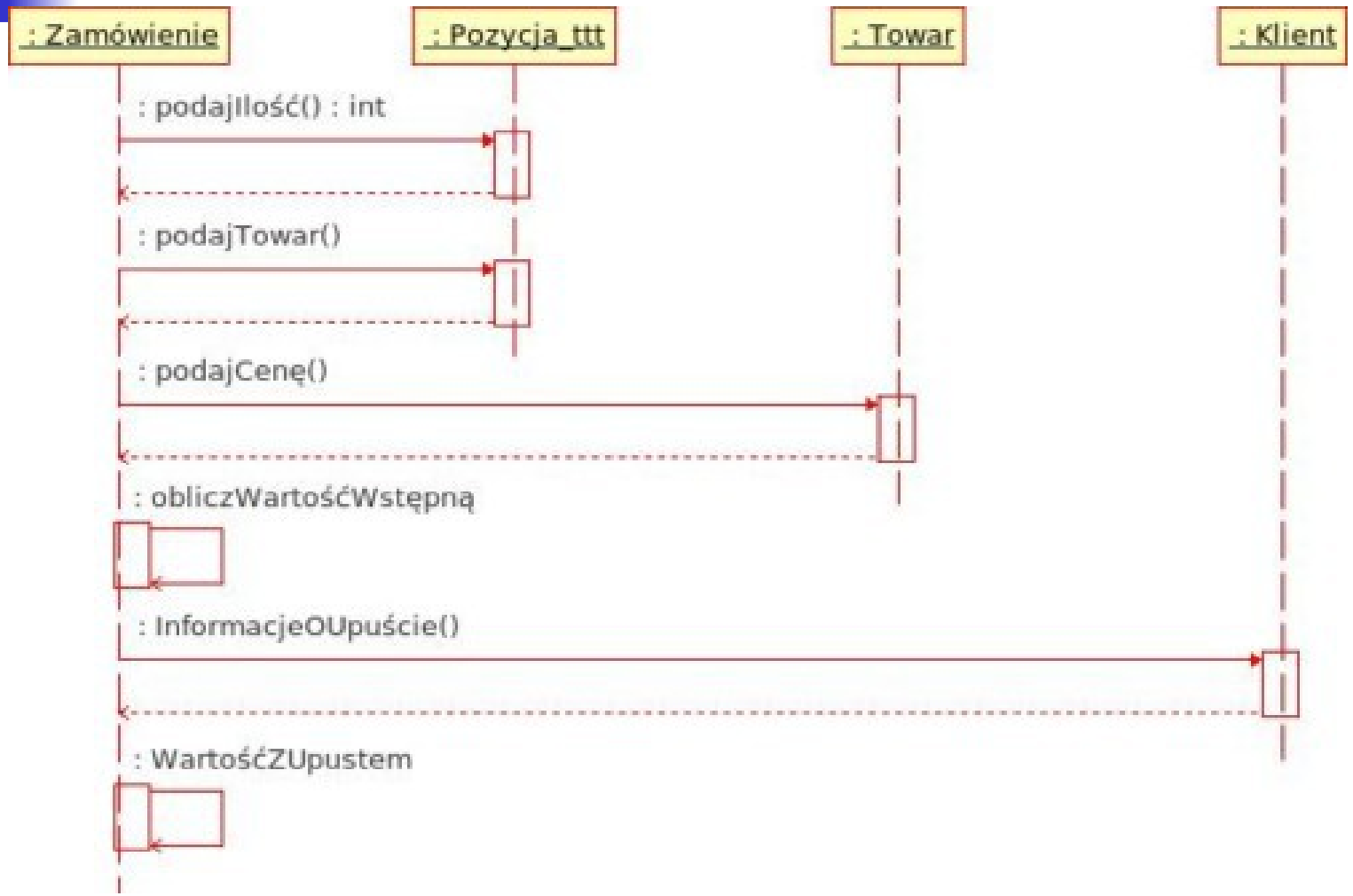
Diagramy sekwencji

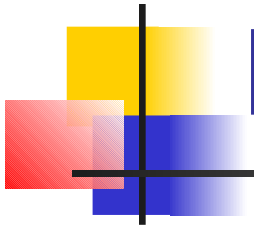
- Diagramy sekwencji dobrze ukazują dynamiczne aspekty realizacji scenariuszy, w których dochodzi do złożonych oddziaływań pomiędzy obiektami. Podstawowymi oddziaływaniami są wymiany komunikatów oznaczane następującymi symbolami:
  - przesłanie komunikatu asynchronicznego – strzałka zwykła
  - wywołanie funkcji – strzałka z wypełnionym grotem
  - powrót z funkcji – strzałka rysowana linią przerywaną

Diagramy sekwencji

- Na diagramach sekwencji w łatwy sposób ilustrować można tworzenie i niszczenie obiektów (poprzez odpowiednie umieszczenie obiektów i odpowiednie symbole). Diagramy sekwencji mogą zawierać dodatkowe konstrukcje umożliwiające zapis takich aspektów realizacji programu jak instrukcje warunkowe czy pętle. Realizuje się to poprzez umieszczenie na diagramie sekwencji odpowiednich regionów – ramek. Diagramy sekwencji mogą także zawierać informacje o ograniczeniach czasowych realizowanych działań (istnieją też specjalne diagramy do tego celu – timing diagrams.

Diagramy sekwencji - przykład





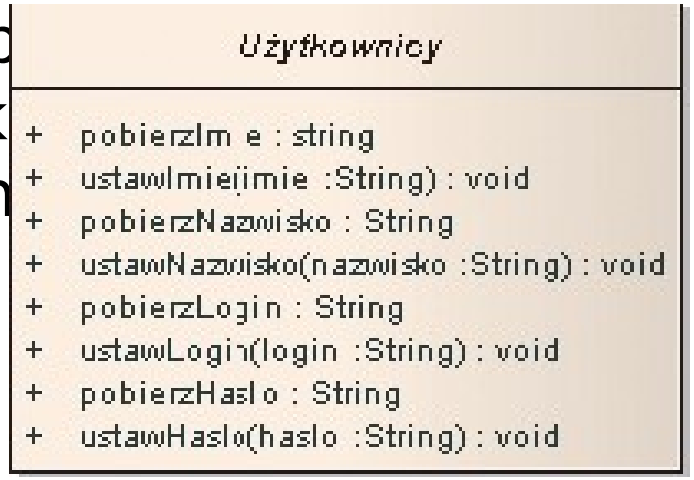
KLASY

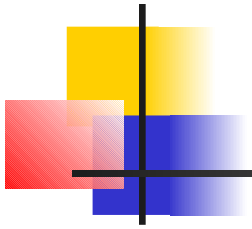
Klasa jest miejscem przechowywania cech obiektów, które są niezmienne. Klasa **nie jest** zbiorem obiektów i **nie jest** definicją zbioru obiektów. Stosunek klasa/podklasa oznacza, że obiekty podklasy posiadają wszystkie cechy nadklasy, plus swoje cechy. Np. klasa Student ma wszystkie cechy klasy Osoba, plus niektóre własne.

Klasy abstrakcyjne

- W **UML-u klasy** abstrakcyjne niewiele różnią się
od normalnych klas. Jedyną widoczną różnica jest
ich nazwa, napisana kursywą.

  Warto zauważyć, że można tworzyć
także operacje ab ich
nazwa będzie tak
abstrakcyjnych, n





Najważniejsze cechy obiektów:

Nazwa – językowy identyfikator

Typ – czyli statyczna budowa obiektu

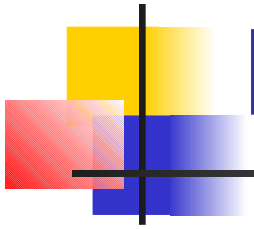
Metody – operacje które można wykonać na obiekcie

Zdarzenia, wyjątki – zachodzą w operacjach

Lista eksportowa – które atrybuty są dostępne z zewnątrz

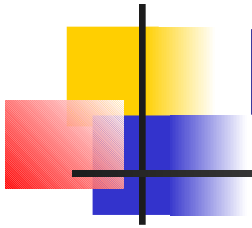
Ograniczenia – muszą podlegać obiekty

...



Diagramy klas

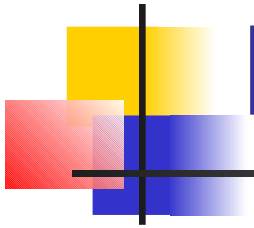
- Diagram klas służy do zobrazowania współpracy klas.
- są odmianą klasyczną diagramów encja-związek (rozbudowanymi o nowe elementy)
- dużo oznaczeń o charakterze pomocniczym (np.: notatki i ograniczenia)



Diagramy klas cd

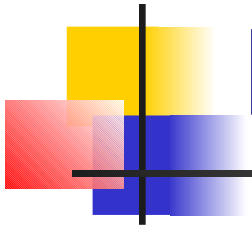
- Diagram klas stanowi statyczny, koncepcyjny widok projektowanej aplikacji
- Pokazuje kluczowe elementy (klasy) oraz ich związki w systemie
- Klasa jest to ogólne określenie dla grupy podobnych obiektów
- Obiekt jest instancja klasy
- Pojecie klasy na diagramie jest ściśle powiązane z technikami programowania zorientowanego obiektowo
- Jest to jeden z istotniejszych diagramów UML

Do czego służy diagram klas?



- Pozwala na sformalizowanie specyfikacji danych i zestawu metod
- Pełni role uniwersalnego środka graficznego wizualizującego implementacje klas, może bezpośrednio służyć jako graficzny środek pokazujący szczegóły implementacji
- Narzędzia projektowe zgodne ze specyfikacją UML umożliwiają, na podstawie diagramów klas, generowanie elementów źródeł w popularnych językach obiektowych, takich jak: C++, Java, C#

Do czego służy? cd



zapis modelu pojęciowego

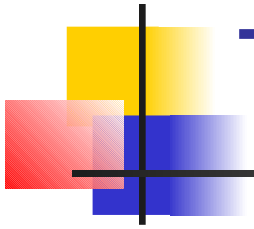
reprezentują pojęcia w dziedzinie zastosowań, które aktualnie podlegają analizie – sformalizowana wizja wyobrażeń powstających podczas myślenia nad problemem

sformalizowana specyfikacja danych i metod

dotyczy opisu zewnętrznego oprogramowania

bez szczegółów implementacyjnych

Tworzenie diagramu klas

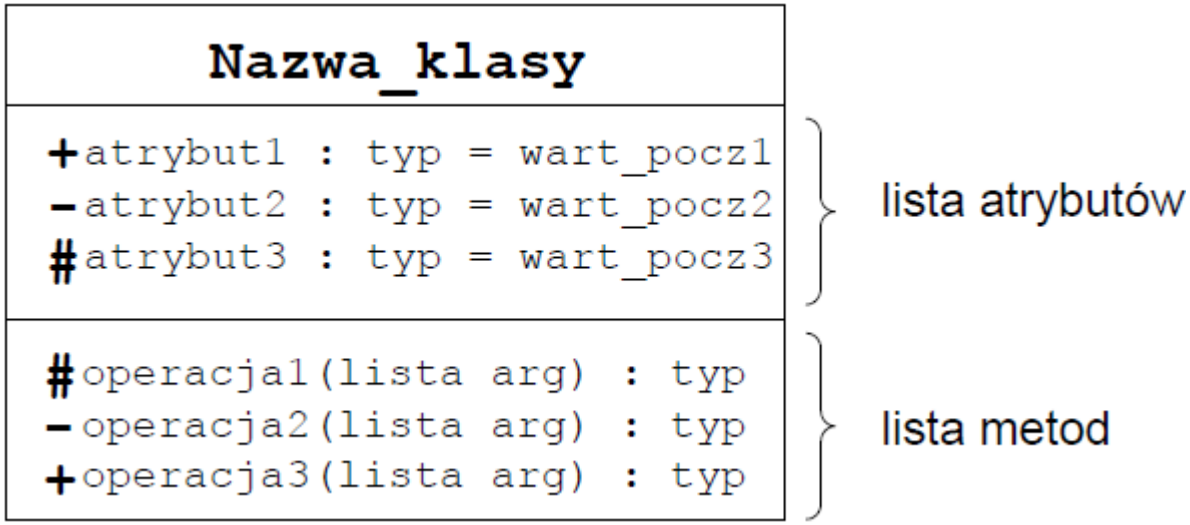


- Identyfikacja klas i obiektów
- Identyfikacja związków pomiędzy klasami
- Identyfikacja i definiowanie pól (atrybutów)
- Identyfikacja i definiowanie metod i komunikatów kolejność

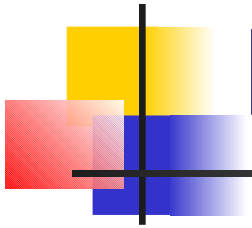
  kolejność wykonywania nie jest ustalona i zależy zarówno od stylu pracy, jak i od konkretnego problemu.

Reprezentacja graficzna

- Klasa obrazowana jest za pomocą podzielonego na części prostokąta –każda część reprezentuje cechy o zbliżonym przeznaczeniu



Reprezentacja graficzna



Widoczność atrybutów i metod określają następujące symbole:

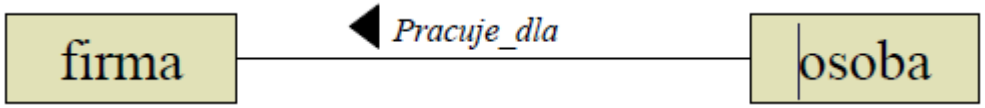
+ publiczna (public)

- prywatne (private)

chronione (protected)

Asocjacje – powiązania pomiędzy obiektami klas

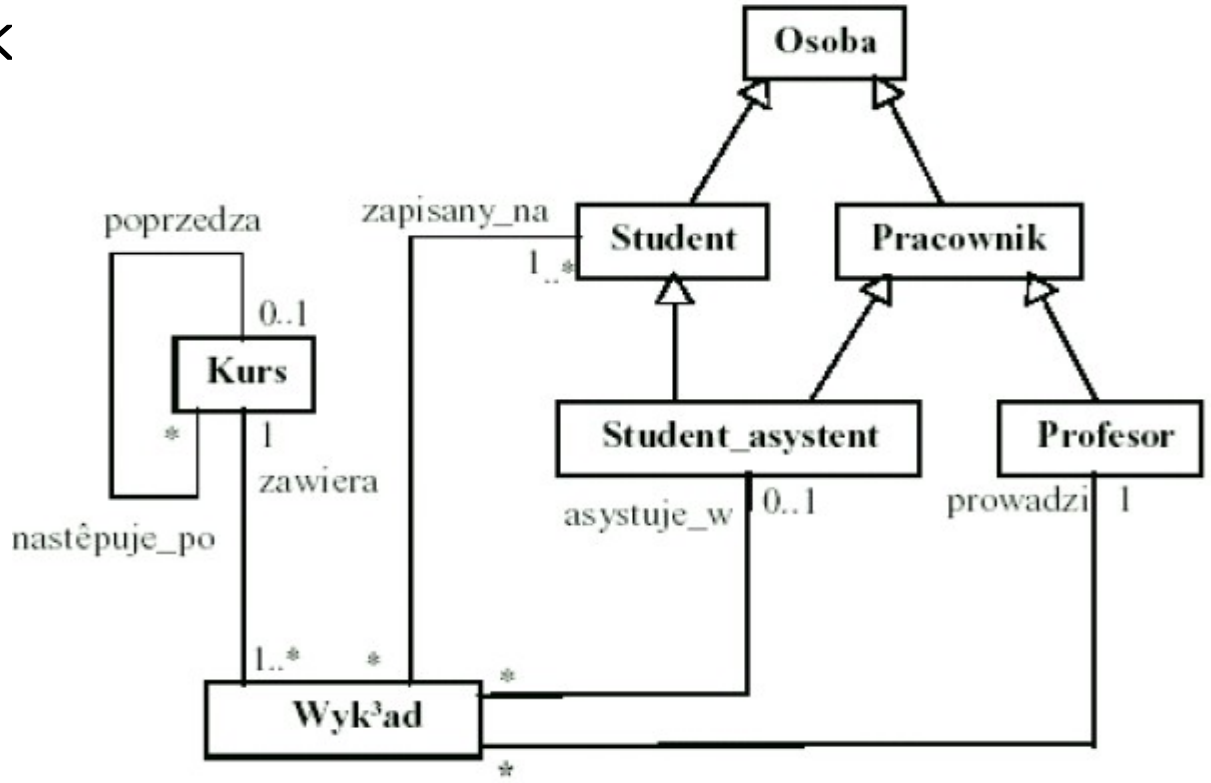
Poniżej widoczny jest przykład specyfikacji asocjacji pomiędzy obiektami klasy Osoba i obiektami klasy Firma. Czarny trójkącik określa kierunek wyznaczony przez nazwę powiązania.



dlaAsocjacje mają nazwy, które wyznaczają znaczenie tej asocjacji w modelu pojęciowym. Jeżeli to znaczenie jest oczywiste, wówczas nazwę asocjacji można pominąć.

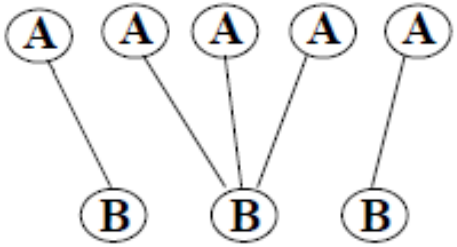
Asocjacje - liczności

Liczność oznacza, ile obiektów innej klasy może być powiązane z jednym obiektem danej klasy (para liczb oznaczająca ilość minimalną i mak

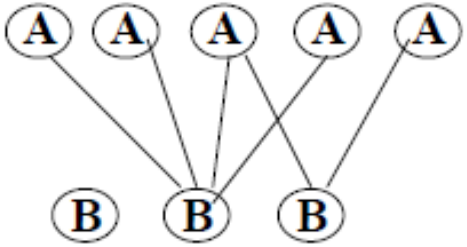


Asocjacje - przykłady

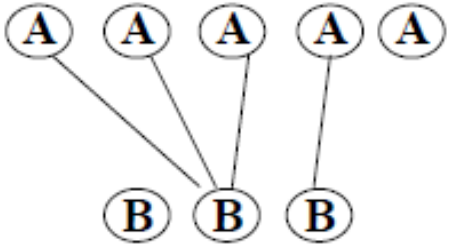
Asocjacje mogą być wyposażone w oznaczenia



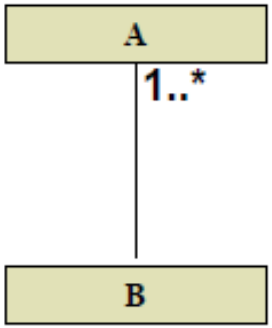
A→B : min=1, max=1
B→A : min=1, max=N

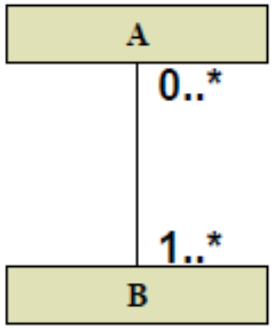


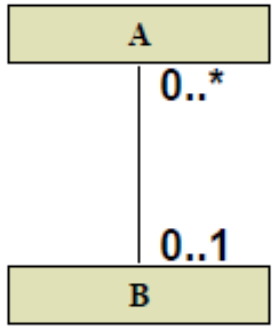
A→B : min=1, max=N
B→A : min=0, max=N



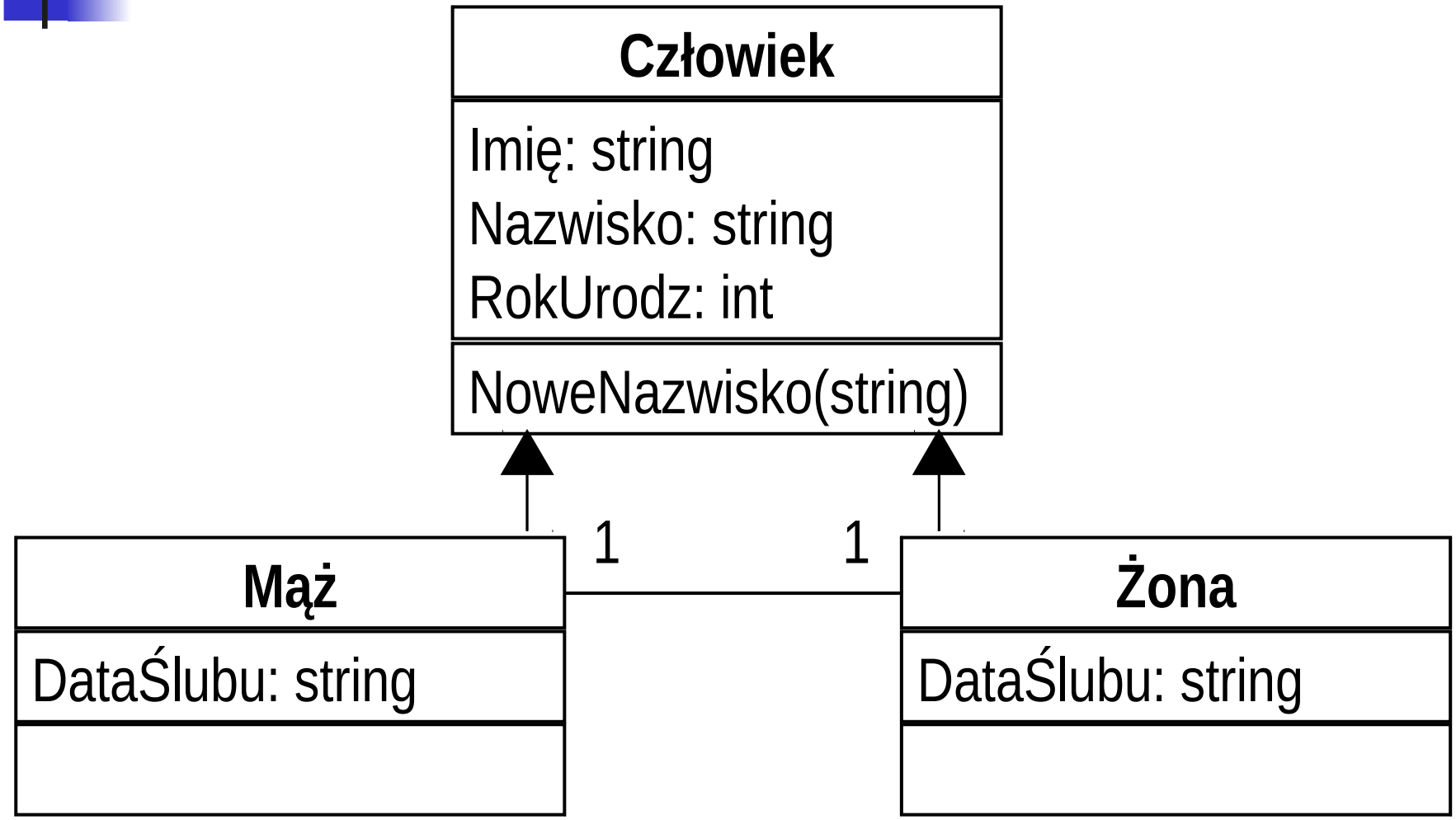
A→B : min=0, max=1
B→A : min=0, max=N





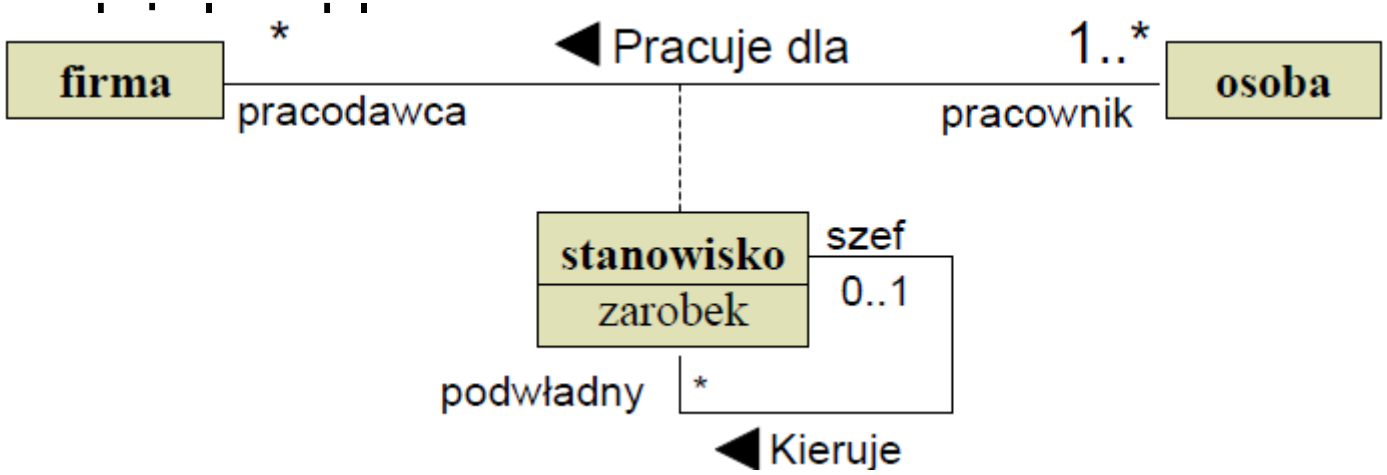


Asocjacje


Człowiek
Imię: string
Nazwisko: string
RokUrodz: int
NoweNazwisko(string)
1
1
Mąż
DataŚlubu: string
Żona
DataŚlubu: string

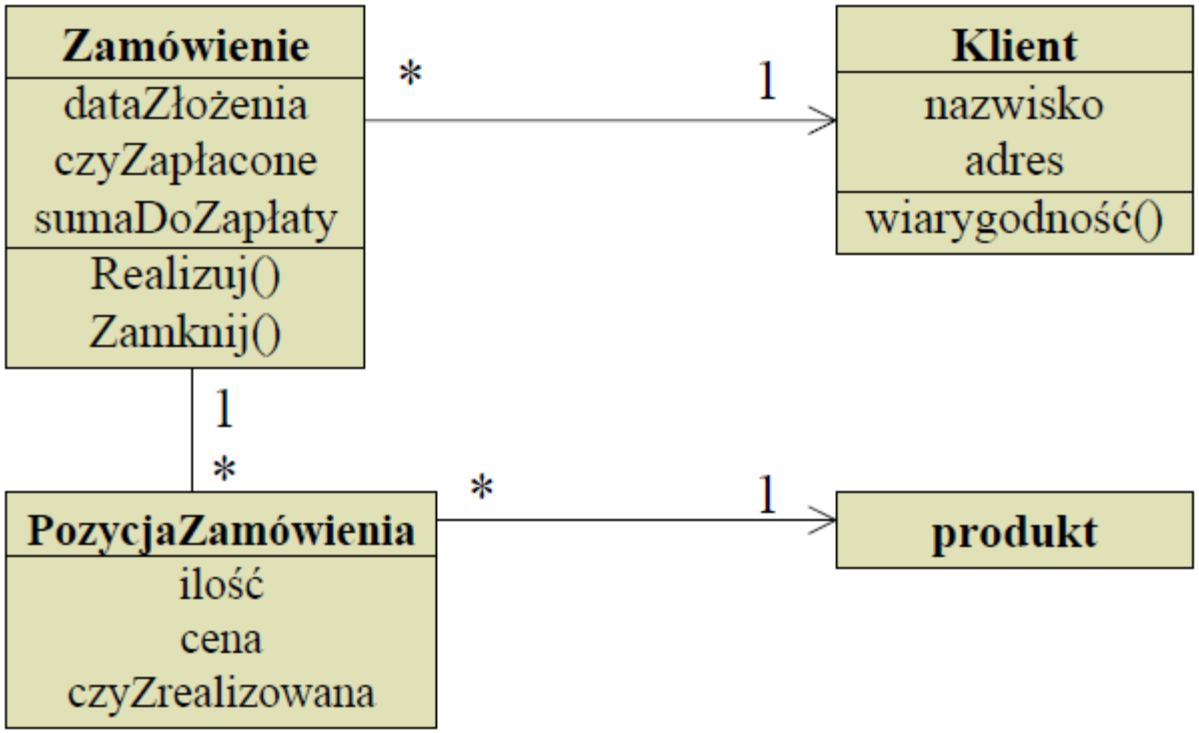
Asocjacje – nazwy ról

- Asocjacje mogą być także wyposażone w dodatkowe nazwy ról (przy odpowiednich końcach).
- Asocjacje mogą posiadać atrybuty. W tym celu przewidziano linię przerywaną łączącą daną asocjacją z klasą. Wewnątrz klasy asocjacji można zdefiniować atrybuty operacje i inne cechy połączenia. Klasa asocjacji może być używana za san



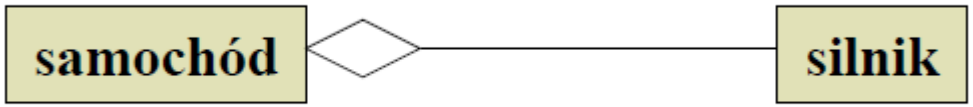
Asocjacje skierowane

Na diagramach UML można zaznaczyć kierunek nawigacji wzdłuż danej asocjacji



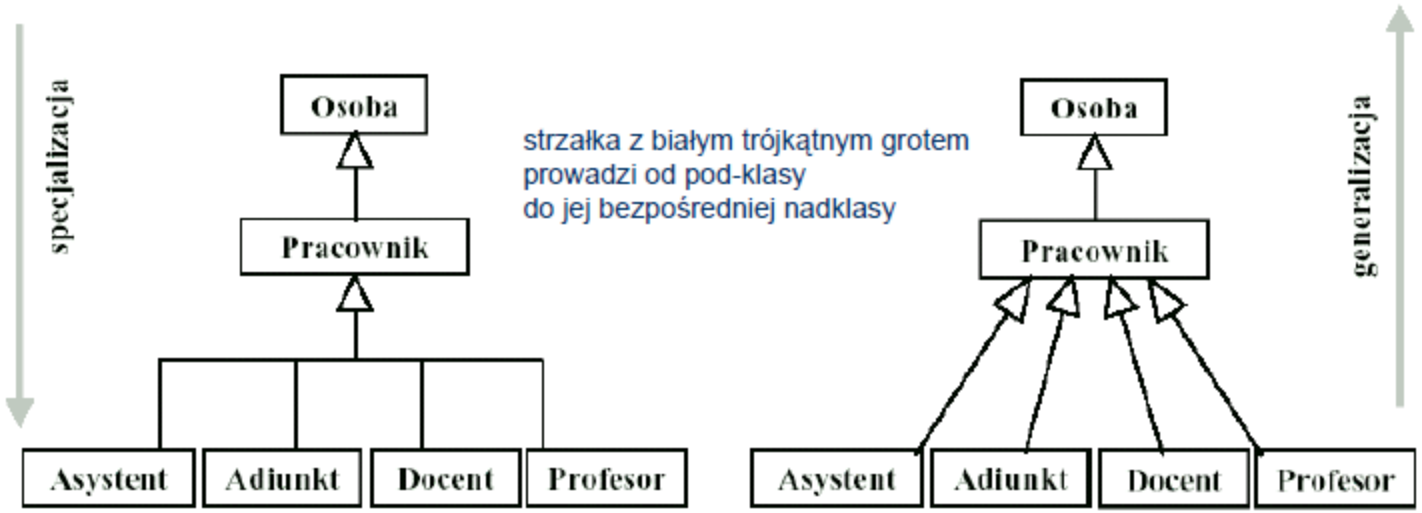
Agregacje

Szczególny przypadek asocjacji wyrażający zależność: część –całość. Oznacza się je za pomocą pustego rombu.



Dziedziczenie

Obiekt pod-klasy automatycznie dziedziczy wszystkie atrybuty, metody, asocjacje i agregacje z wszystkich jej nadklas.



Dziedziczenie

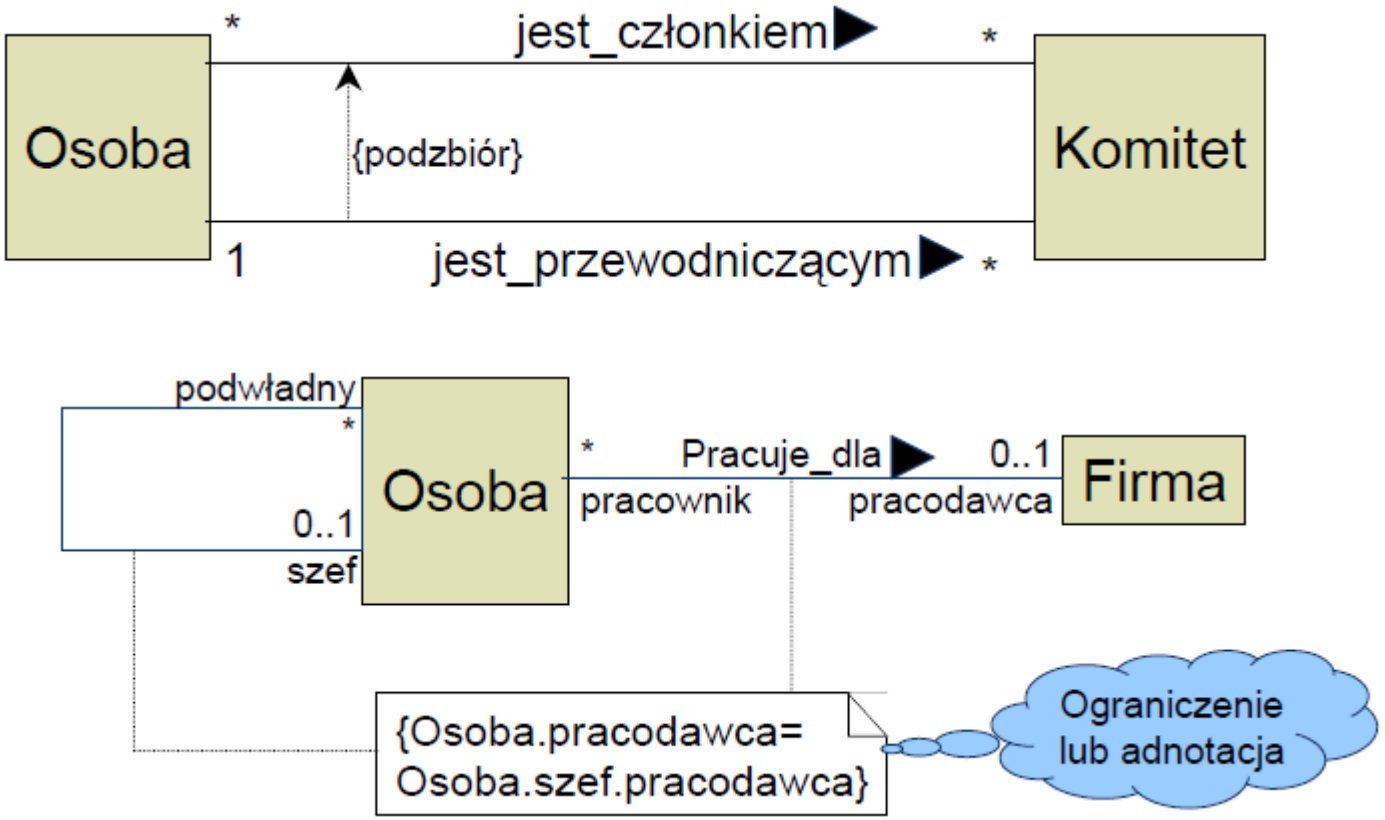
- SPECJALIZACJA

budowa pojęć bardziej szczegółowych, gdy mamy bardziej ogólne

- GENERALIZACJA

budowa pojęć bardziej ogólnych, gdy mamy bardziej szczegółowe

Ograniczenia

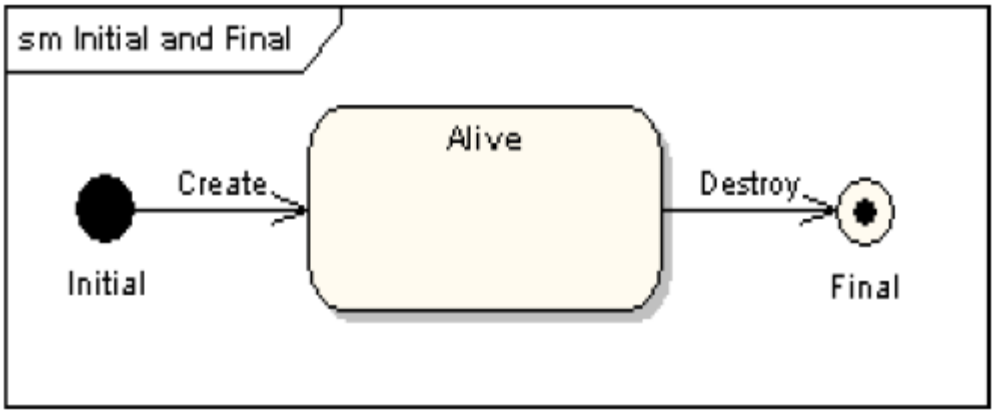


Diagramy stanów

- Diagram stanu opisuje zmiany stanu obiektu, podsystemu lub systemu pod wpływem działania operacji - iest szczególnie przydatny, gdy zachowanie obiektu jest złożne.
- Stan jest okolicznością lub sytuacją, w jakiej znajduje się obiekt
  - · jest rezultatem poprzedniej aktywności
  - · spełnia jakiś warunek
  - · jest określony przez wartości własnych atrybutów i powiązań do innych zadań
  - · wykonuje pewne czynności
  - · czeka na jakieś zdarzenie

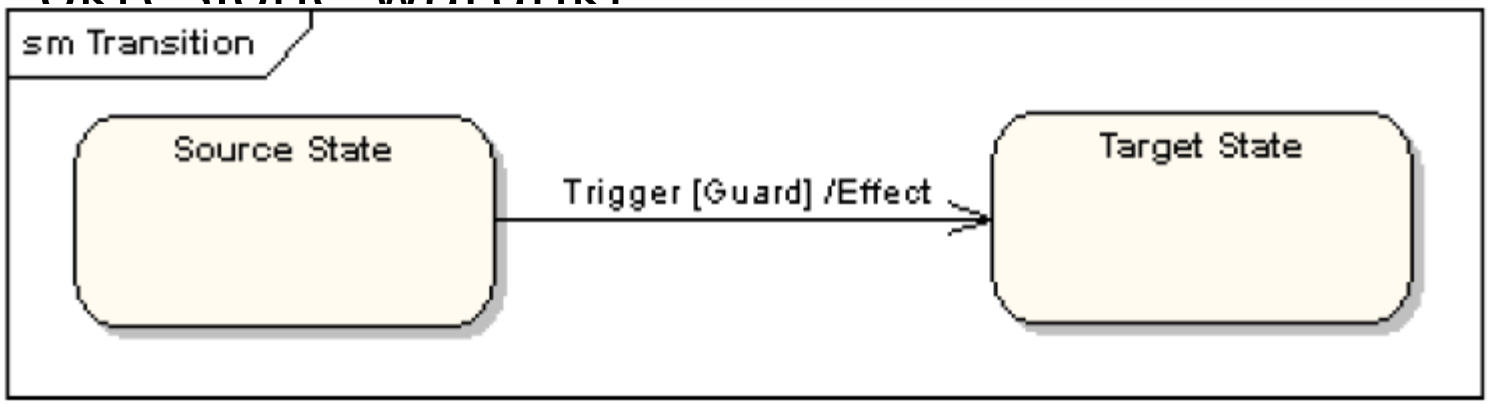
Diagramy stanów

- Stan początkowy – może być tylko jeden stan początkowy
- Stan końcowy – może być kilka stanów końcowych



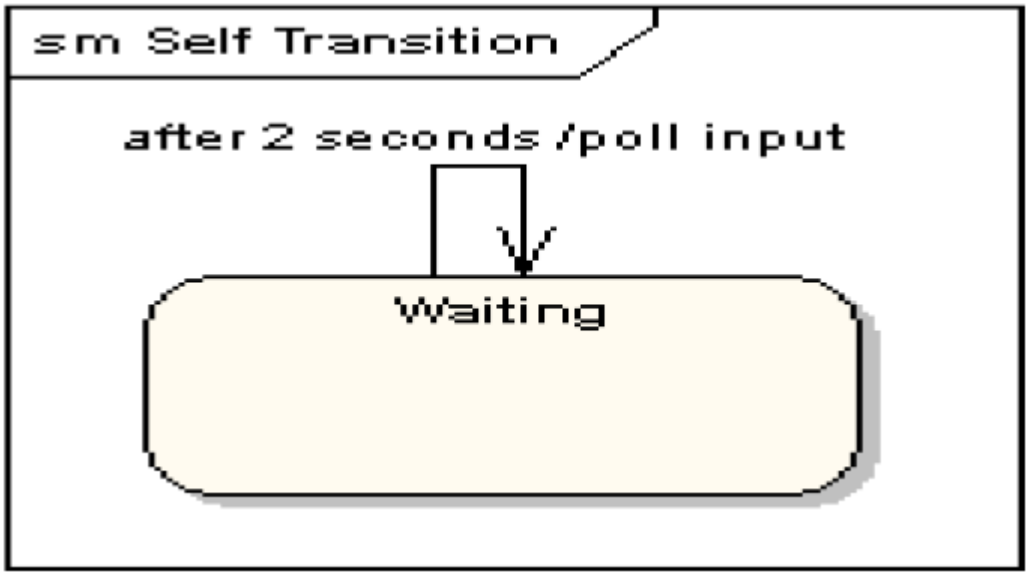
Diagramy stanów

- Przejście jest związkiem między dwoma stanami, który wskazuje, np. obiekt znajdujący się w pierwszym stanie wykona pewne akcje i przejdzie do drugiego stanu, ilekroć zaistnieje określone zdarzenie i będą spełnione określone warunki



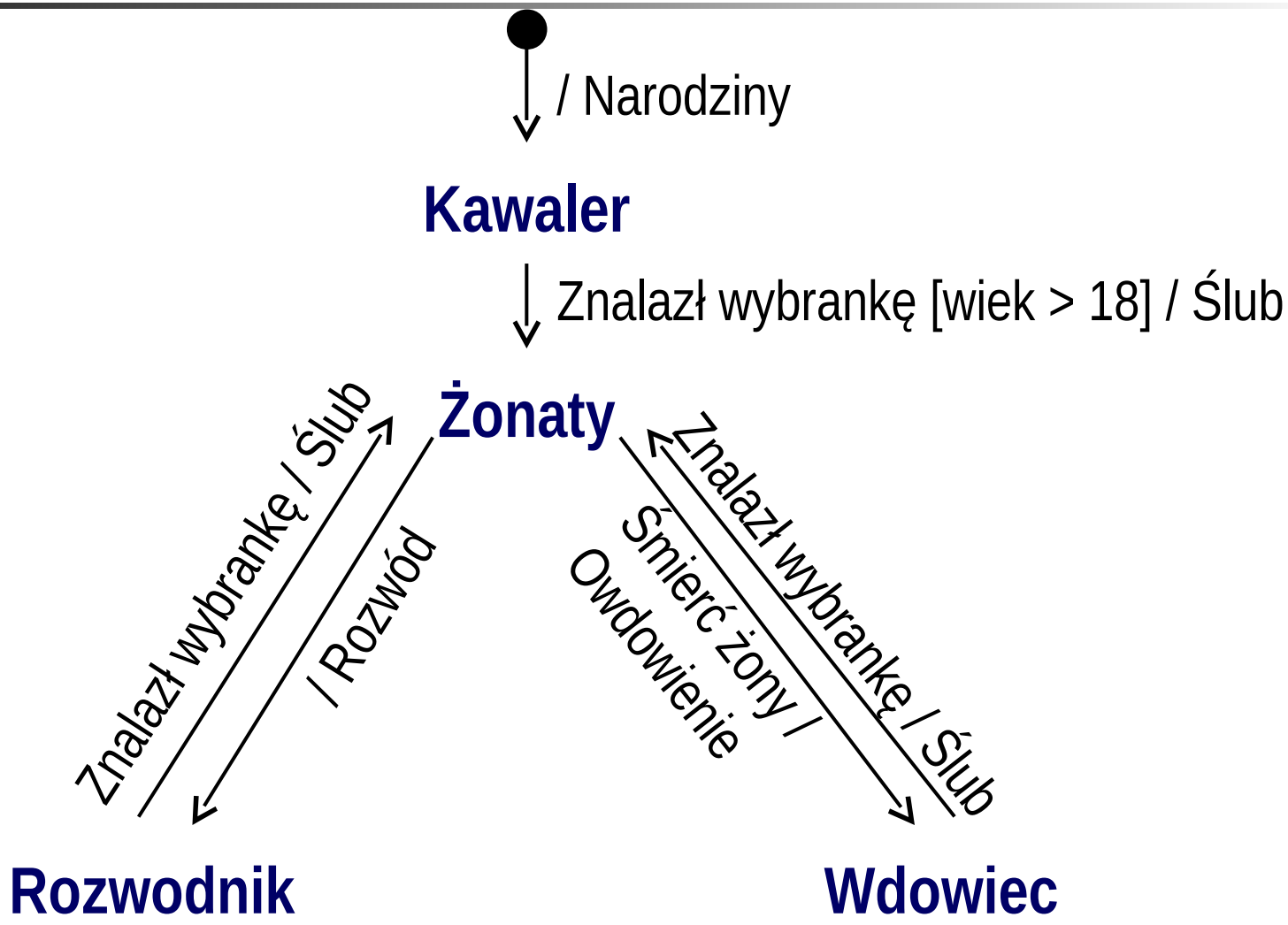
Diagramy stanów

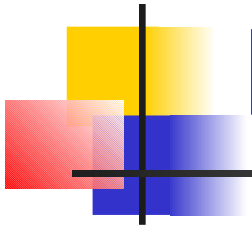
- Przejście własne jest związkiem między tym samym stanem, który wskazuje, np. obiekt znajdujący się w pewnym stanie wykona pewne akcje i powróci do tego samego stanu, ilekroć zaistnieje określone zdarzenie i będą spełn



Obiekty ze stanami

Diagramy stanów



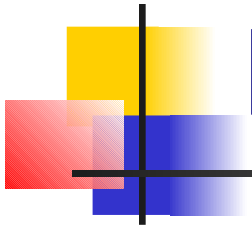


Programy

- Z programów darmowych warto przyjrzeć się:
  - ArgoUML (http://argouml.tigris.org) – jest to aplikacja napisana w Javie, a więc działająca pod niemal każdym systemem operacyjnym. Jej wadą jest jednak wsparcie jedynie dla UML-a w wersji 1.4, co uniemożliwi budowę niektórych diagramów.

Programy

- Poseidon (http://www.gentleware.com) - produkt komercyjny, tak jak ArgoUML stworzony w Javie. Oferujący jednak wsparcie dla UML-a w wersji 2.0. Istnieje kilka jego wersji, w tym Community Edition, która jest darmowa i doskonale nadaje się do nauki.
- IBM Rational Software (http://www.ibm.com/software/rational) – cała rodzina programów, wśród których znajduje się także aplikacja do budowy diagramów UML. Jest to produkt z najwyższej półki, o czym niestety nie pozwala zapomnieć także jego cena.



Programy

- Sparx Enterprise Architect (http://www.sparxsystems.com.au) – bardzo dobry program, oferujący wsparcie dla UML-a w wersji 2.1,
  dodatkowo pozwalający budować diagramy baz
  danych. A to wszystko za dość rozsądną cenę
  (w przypadku jednego stanowiska, od 135$ za
  wersję Desktop Edition).

Definicja klasy

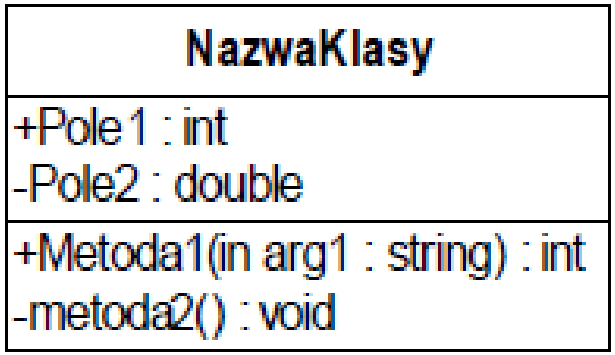
- W języku UML klasa jest reprezentowana w postaci prostokąta. W prostokącie możemy wyróżnić trzy części: nazwę klasy, pola (atrybuty) oraz metody. Do oznaczenia, że składowa jest publiczna, przed jej nazwą stawiamy znak plus (+), natomiast do oznaczenia składowych prywatnych używamy znaku minus (-).

  Następująca definicja klasy:

```
class NazwaKlasy
{
        public int Pole1;
        private double pole2;
        public int Metoda1(string arg1){}
        private void metoda2(){}
}
```

Definicja klasy

w języku UML ma następującą reprezentację:

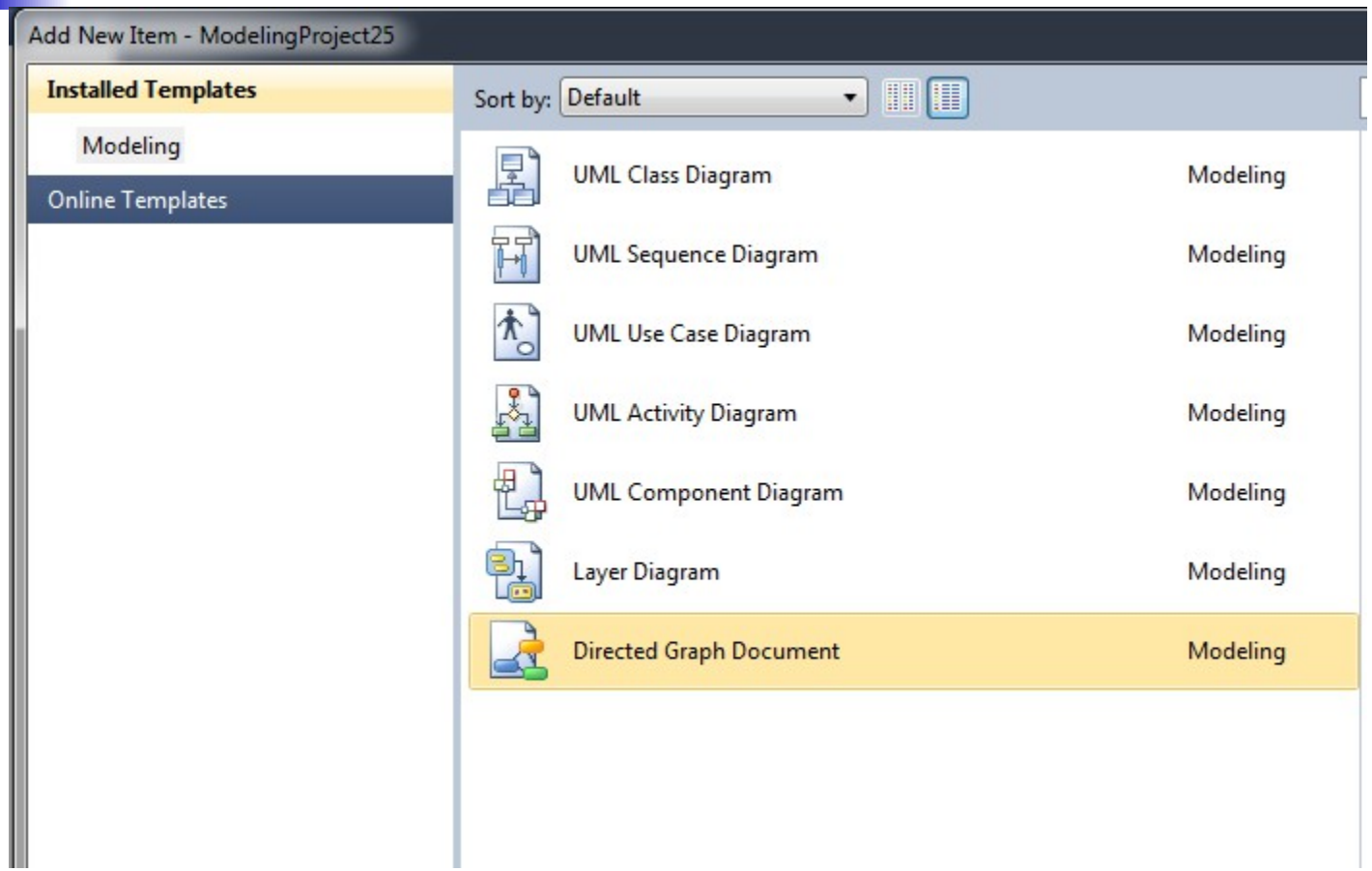


Reprezentacja klasy w języku UML

VS 2010 Ultimate

- Aby Visual Studio obsługiwało (imitowało) język UML i umożliwiało generowanie kodu mając projekt UML oraz generowanie projektów różnych diagramów należy w tym celu zainstalować wtyczkę Visual Studio 2010 Visualization & Modeling Feature Pack (x86 and x64) - (English), którą można pobrać ze strony Microsoft.

VS 2010 Ultimate



VS 2010 Class Diagram