

Modelowanie struktury systemu

dr hab. inż. Michał Śmiałek, prof. uczelni
dr inż. Kamil Rybiński



**Wydział
Elektryczny**
POLITECHNIKA WARSZAWSKA

Inżynieria oprogramowania



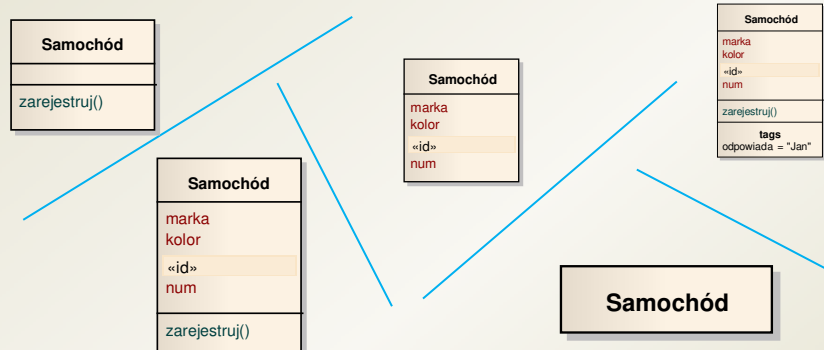
1

Modele struktury w języku UML

- **Model klas**
 - Uniwersalny model używany do opisu struktury klas obiektów i łączących je zależności
- **Model komponentów**
 - Przedstawienie struktury złożonych systemów
- **Model wdrożenia (montażu)**
 - Definiowanie fizycznych elementów budowanego systemu i połączeń między nimi
- **Model obiektów**
- **Model pakietów, Model składowych**
 - Stosunkowo rzadko używane modele opisujące szczegóły grupowania elementów oraz ich składników

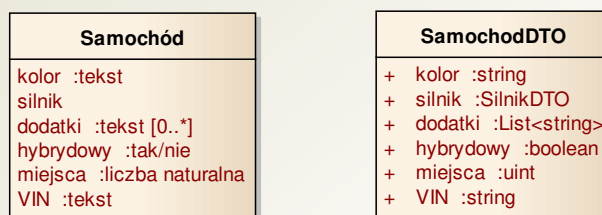
2

Model klas - klasa



- Różne poziomy szczegółowości reprezentacji klasy
 - Ta sama klasa może być pokazana na różnych diagramach na różnym poziomie szczegółowości

Atrybuty klasy



- Notacja atrybutów
 - Nazwa (obowiązkowa)
 - Typ (po dwukropku, opcjonalny)
 - Krotność (w nawiasach kwadratowych, opcjonalna)
- Uwaga: język UML nie definiuje dopuszczalnych typów danych
 - Typy mogą być zgodne np. z danym językiem programowania lub terminologią klienta

Operacje klasy

Samochód

```
oblicz cenę()  
zderz()  
zaparkuj()  
przemaaluj()  
uruchom silnik()  
sprzedaj()
```

MSamochod

```
# ObliczCene(znizka :float, podatek :float) :float  
+ ZmienKolor(nowyKolor :Kolor) :void  
- PobierzDodatki() :List<string>
```

- Operacje wyróżnione są nawiasami po nazwie
- Notacja operacji
 - Nazwa (obowiązkowa)
 - Parametry (w nawiasie, obowiązkowo, mogą być puste)
 - Typ rezultatu (opcjonalny)
- Notacja parametrów jak atrybutów
- Widoczność: + (publiczna), - (prywatna), # chroniona

Typy wyliczeniowe

Samochód

kolor: Kolor

«enumeration» Kolor

czerwony
zielony
niebieski

- Definicja typu poprzez określenie listy wartości
- Wyróżnione poprzez nadanie stereotypu «enumeration»
- Typy wyliczeniowe stosujemy tak, jak inne typy
- Uwaga: zwykłe klasy też można użyć jako typy


Stereotypy

«data transfer»
XCar

«UI element»
CarDataForm

«business uc»
Buy new car

«JavaBean»
CarBean

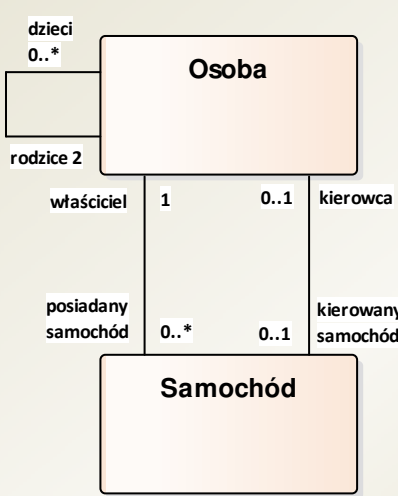

 CarDataForm

- Nadanie dodatkowego znaczenia elementowi modelu
- Można nadć dowolnym elementom (nie tylko klasom)
- Nazwa stereotypu umieszczana nad nazwą elementu
- Nadanie stereotypu może być związane ze zmianą kształtu (ikony) danego elementu

Politechnika
Warszawska
Inżynieria oprogramowania
7

7

Asocjacje

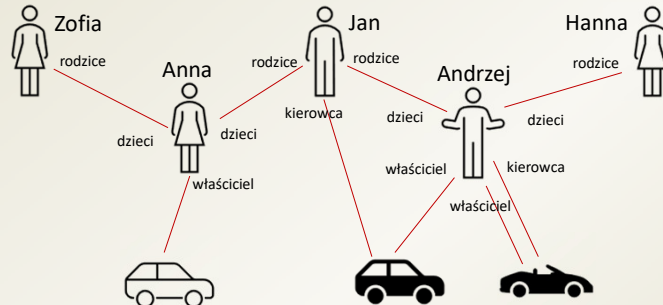


- Określenie powiązań między obiektami klas
- Krotność: liczba obiektów danej klasy w relacji z inną klasą (zakres)
- Rola: nazwa końca
- Asocjacje „zawinięte”: powiązania między obiektami tego samego typu

Politechnika
Warszawska
Inżynieria oprogramowania
8

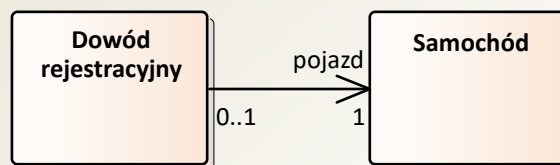
8

Powiązania między obiektami a klasy



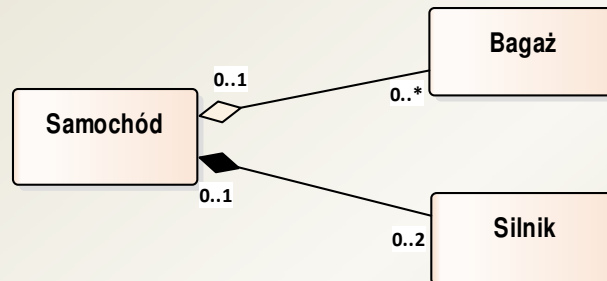
- Powiązania między obiektami muszą być zgodne z modelem klas
 - Dopuszczalne tylko powiązania zgodne z asocjacjami
 - Liczba powiązań mieści się w zakresie krotności

Nawigowalność asocjacji



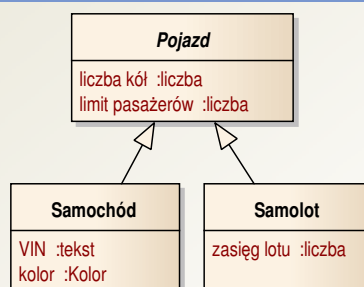
- Możliwość osiągnięcia obiektów jednej klasy przez obiekty drugiej klasy
- Np. dowód rejestracyjny „zna” dane samochodu
- Nawigowalność umożliwia korzystanie z operacji lub dostęp do atrybutów wskazywanej klasy

Agregacje



- Grupowanie elementów: relacja całość - części (składniki)
- Kompozycja: części są integralnymi składnikami całości
 - Części nie mogą być dzielone między różne całości
 - Czas życia części ograniczony czasem życia całości

Generalizacje



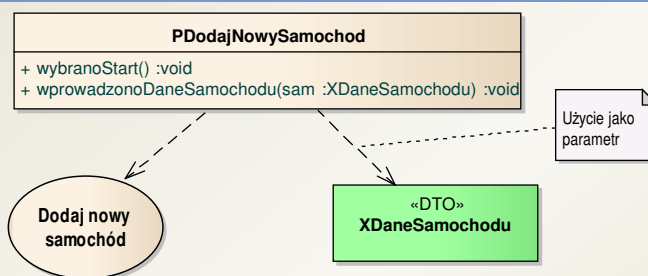
- Zależność między klasami obiektów ogólnych i specjalizowanych
- Klasy specjalizowane „dziedziczą” wszystkie atrybuty, operacje i relacje klasy ogólnej
- Możliwe dziedziczenie wielobazowe
 - Niektóre języki programowania na to nie pozwalają

Klasy abstrakcyjne i interfejsy



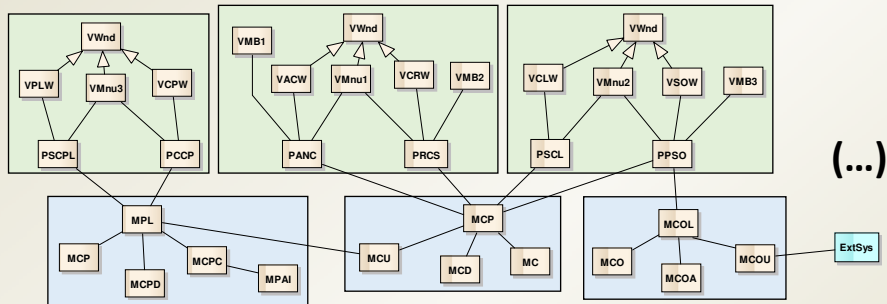
- Klasy abstrakcyjne opisują tylko cechy
 - Brak instancji (obiektów); nazwa pisana kursywą
- Interfejsy definiują zestawy operacji
 - Oddzielenie specyfikacji od jej realizacji
 - Klasy realizują interfejsy - relacja realizacji

Relacje zależności



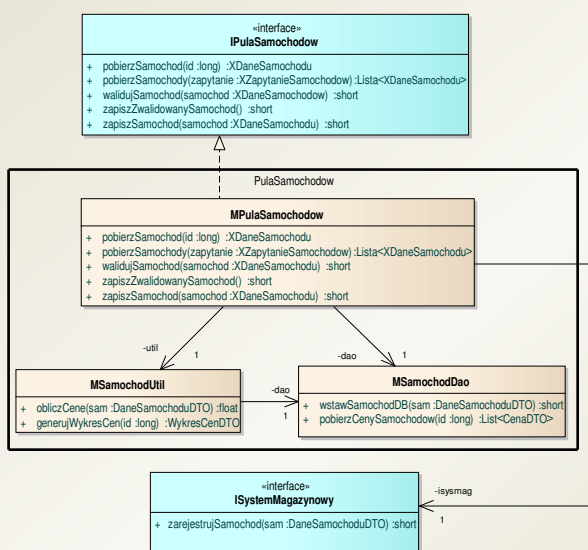
- Znaczenie jednego elementu zależy od innego
 - „Klient” relacji zależy od jej „dostawcy”
- Brak precyzyjnej semantyki tej relacji
- Generyczna - może być stosowana między dowolnymi elementami języka UML (nie tylko klasy)
- Uwaga: notatki mogą opisywać dowolny element

Modelowanie złożoności



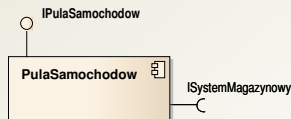
- Duże modele klas wymagają podziału
- Grupowanie klas silnie ze sobą współpracujących
- Komunikacja między grupami klas poprzez wąskie interfejsy

Komponenty z interfejsami



- Komponent dostarcza i korzysta z interfejsów
- Interfejsy są „fasadami” komponentów
- Komponent nie ma relacji z klasami innych komponentów

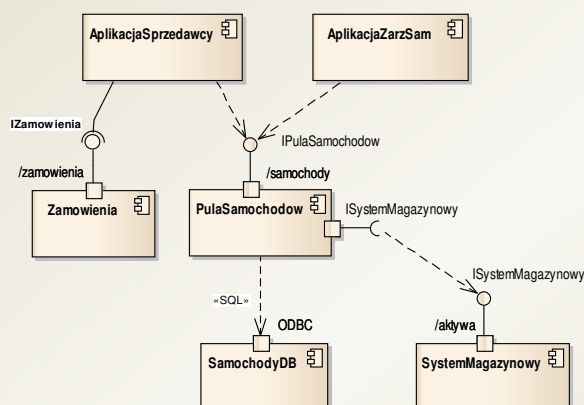
Komponenty



- Model komponentów określa ogólną strukturę systemu
 - Ukryte szczegóły wnętrza komponentu i operacje interfejsów
 - Komponent: podstawowa jednostka modelu - „czarna skrzynka”
 - Interfejs dostarczany - używany przez inne komponenty
 - Interfejs wymagany - konieczny do działania, dostarczany przez inny komponent

17

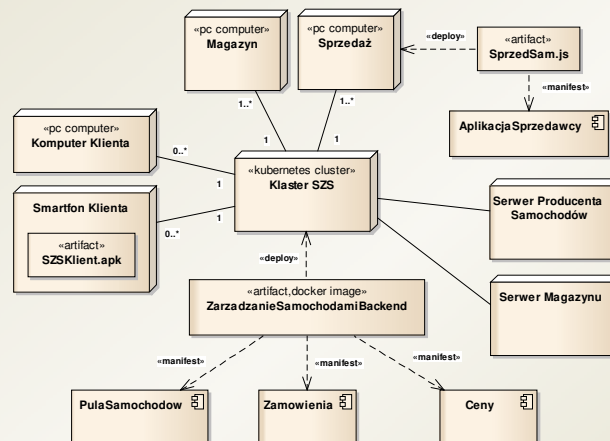
Model komponentów



- Powiązania między komponentami
 - Relacje zależności między interfejsami
 - Połączenia montażowe
- Porty
 - Punkty interakcji (udostępnienie usług)

18

Model wdrożenia (montażu)



- Opis fizycznej struktury systemu

Elementy modelu montażu

- Węzeł
 - Środowisko działania oprogramowania
 - Serwer, komputer PC, urządzenie mobilne, maszyna wirtualna, ...
- Artefakt
 - Element wykonywany w węzłach
 - Może wyrażać jeden lub więcej komponentów
 - Najczęściej - pliki wykonywalne i instalowalne
 - Połączony z węzłami relacjami montażu (deploy)
 - Połączony w komponentami relacjami wyrażania (manifest)
- Uwaga: węzły i artefakty warto opatrywać odpowiednimi stereotypami