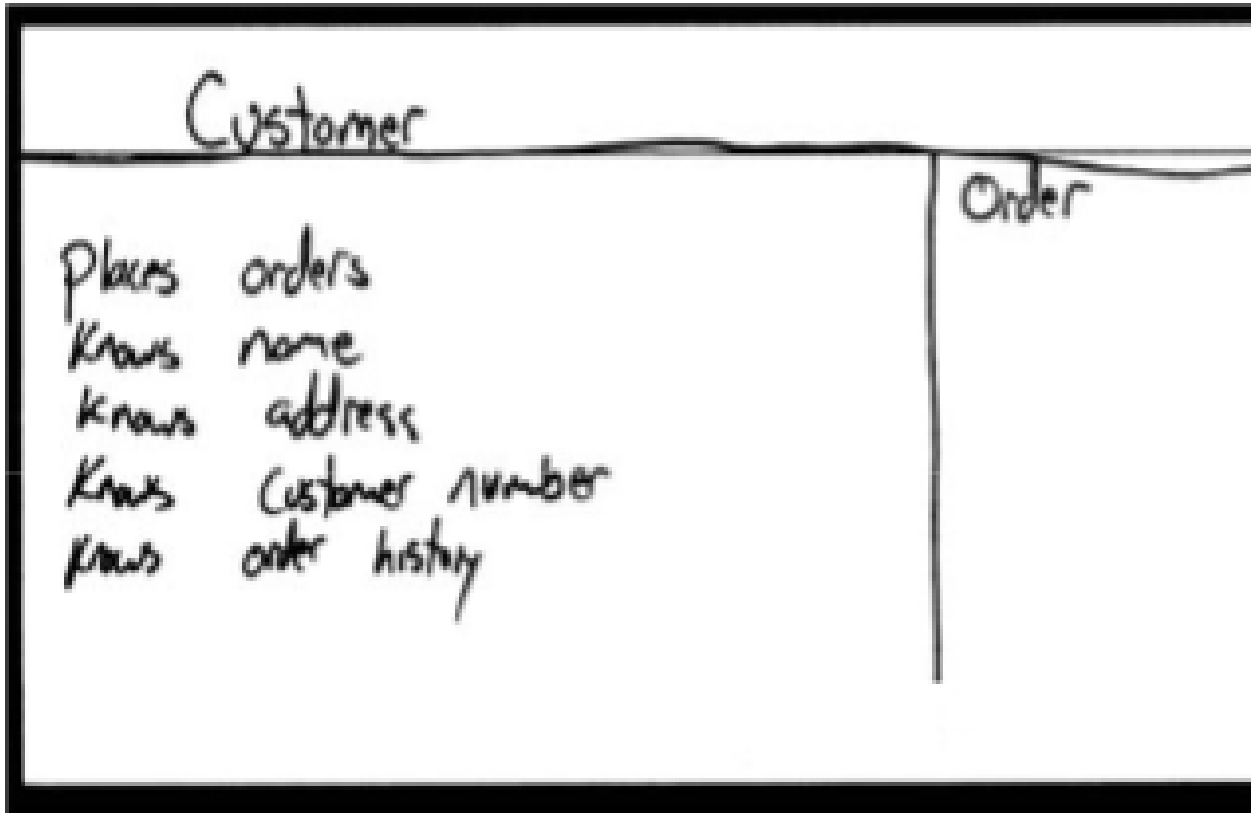
Uvod u UML

Motivacija

- Šta je UML? Da li se stvarno koristi?
- Koji su UML dijagrami? Koje informacije sadrže? Kako i kada se kreiraju?
- Faza dizajniranja softverskog proizvoda

Primjer - dijagrami klasa

- Koje klase je potrebno implementirati da bi sistem zadovoljio postavljene zahtjeve?
- Koji su atributi i metode?
- Interacije između klasa
- Kako se dizajniraju klase
  - Identifikacija klasa
  - CRC card
    - Responsibilities
    - Collaborations



UML

- Grady Booch (BOOCH)
- Jim Rumbaugh (OML: object modeling technique)
- Ivar Jacobsen (OOSE: object oriented software eng)

UML – Unified Modeling Language

- Union of Modeling Languages
  - Use case diagrams
  - Class diagrams
  - Object diagrams „
  - Sequence diagrams „
  - Collaboration diagrams „
  - Statechart diagrams
  - Activity diagrams
  - Component diagrams „

UML

- UML je slika OO sistema
  - Programski jezici nijesu dovoljno apstraktni za OO dizajn
  - UML je široko prihvaćen standard
  - Descriptive
  - Perscriptive (shaped by usage and convention)

Primjena

- Skica sistema
  - Forward/backward design
- Implementacija kompletnog dizajna sistema
- Programski jezik
  - Generisanje programskog koda

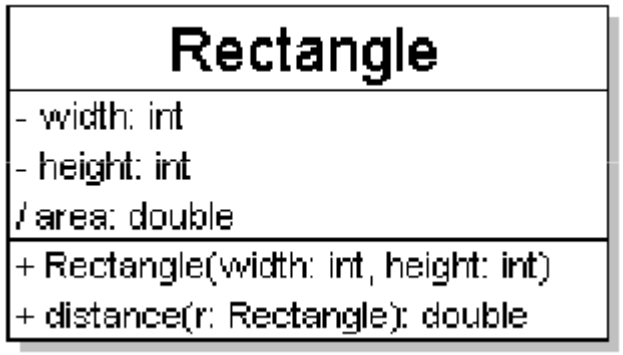
UML dijagrami klasa

- Dijagram klasa
  - Sadrži sve klase u sistemu
  - Atribute i metode
  - Veze između klasa

- Dijagram klasa ne sadrži
  - Detalje veza
  - Detalje algoritama kojima se implementira određeno ponašanje

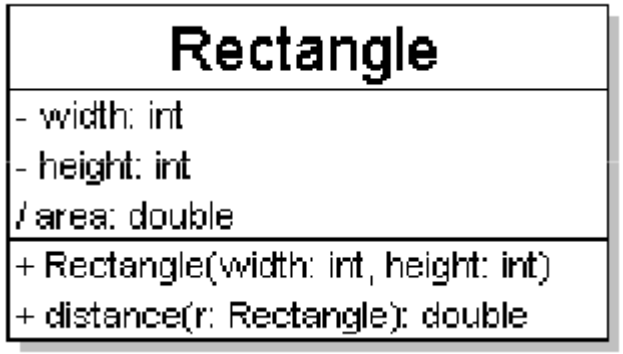
Primjer dijagrama

- Ime klase počinje velikim slovom
  - Stereotip <<interface>>
  - Italic font za abstraktnu klasu
- Atributi
- Metodi



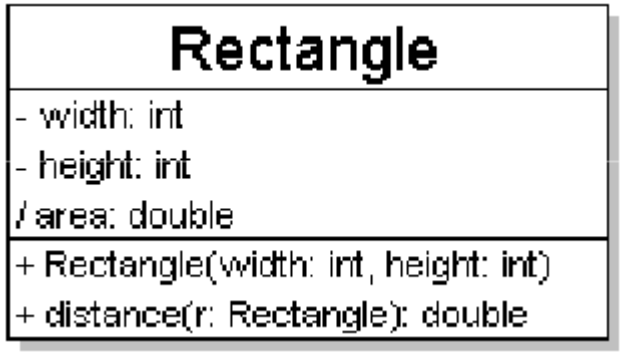
Atributi

- Sintaksa: *visibility name : type [count] = default_value*
- Primjer: balance : double = 0.0
- Visibility: +, #, -, /
- Underline – statički atributi klase



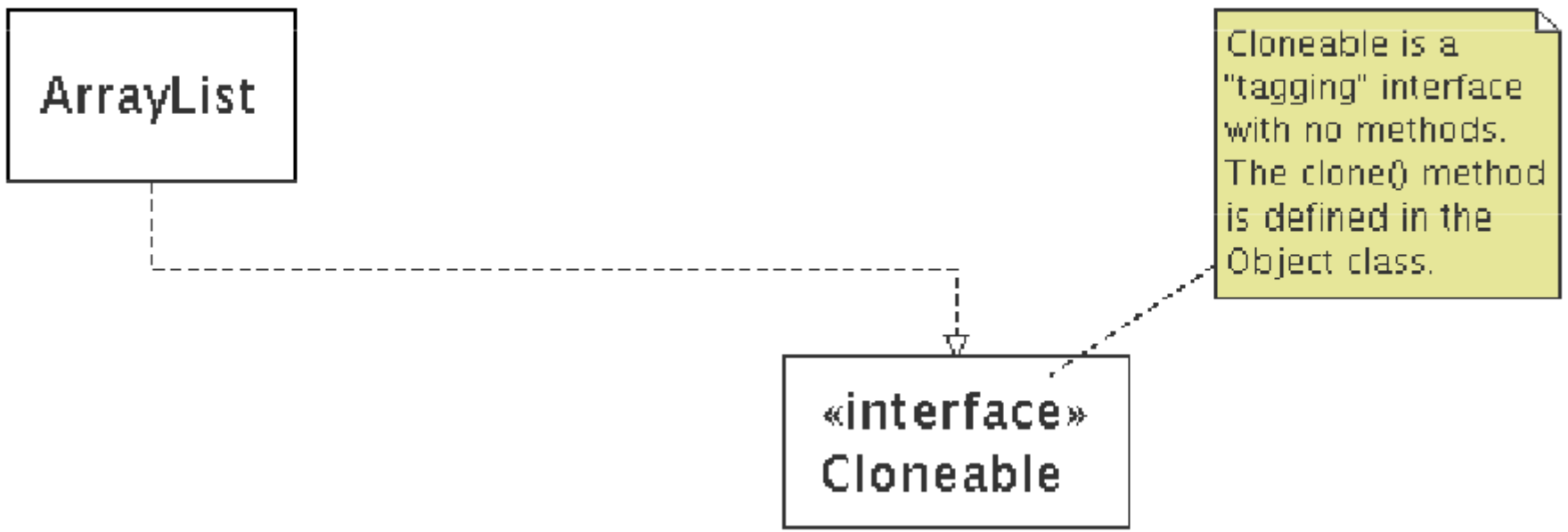
Metodi

- Sintaksa: *visibility name(parameters) : return_type*
- Primjer: + distance(p1: Point, p2: Point): double
- Visibility: +, #, -
- Underline: statičke metode



- Parametri se zadaju kao lista parova name:type

Komentari


ArrayList
«interface»
Cloneable
Cloneable is a "tagging" interface with no methods. The clone() method is defined in the Object class.

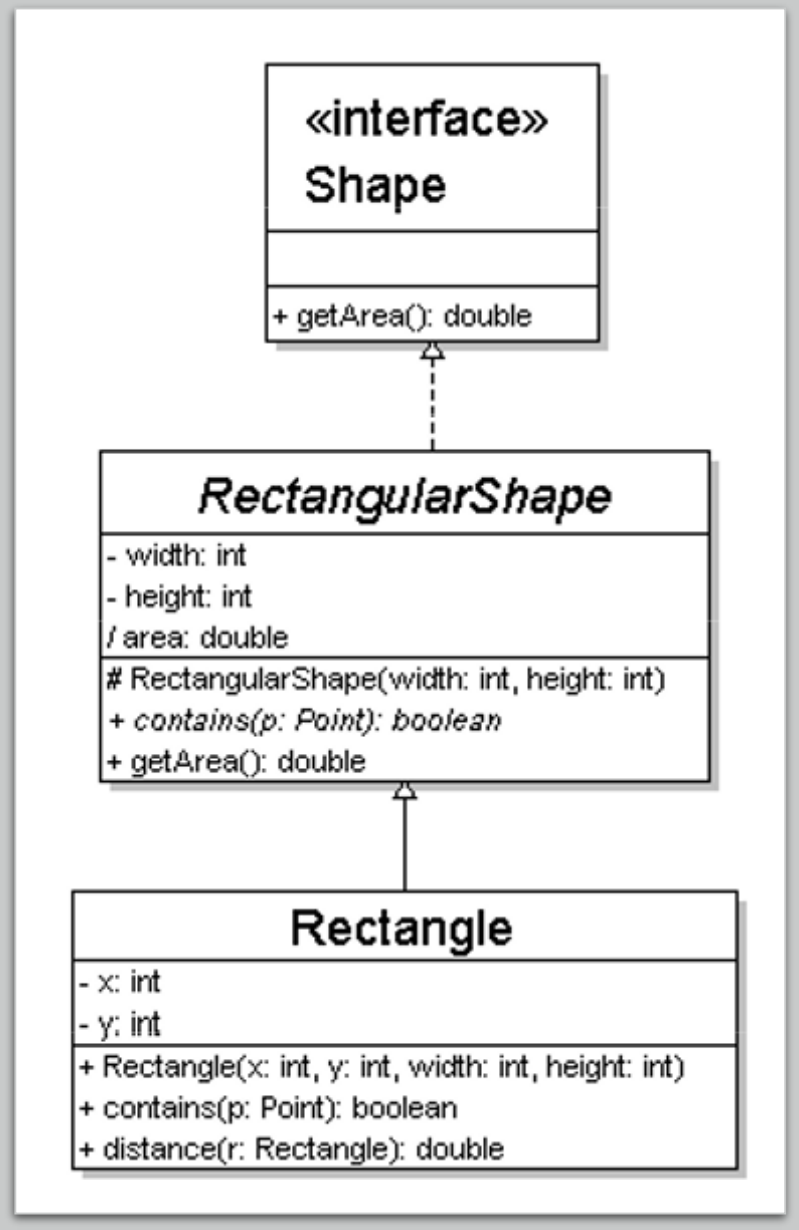
Veze između klasa

- Generalizacija
  - Nasleđivanje
  - Implementacija interfejsa
- Asocijacija
  - Zavisnost
  - Agregacija
  - Kompozicija

Generalizacija

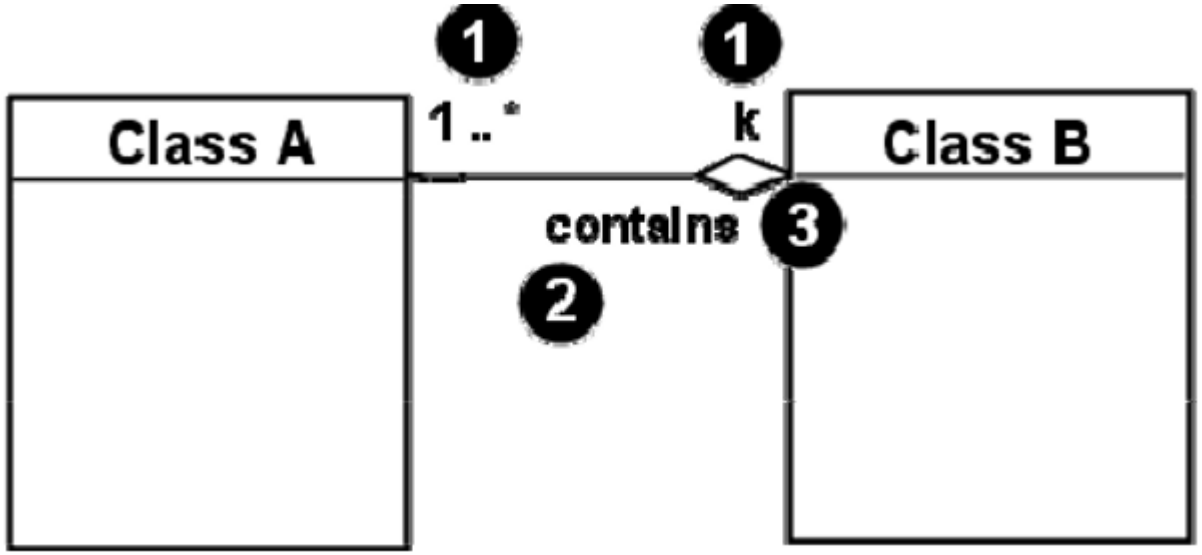
- Top down hijerarhija

- Razlikuju se slučajevi kada je roditelj
  - Klasa
  - apstraktna klasa
  - Interfejs



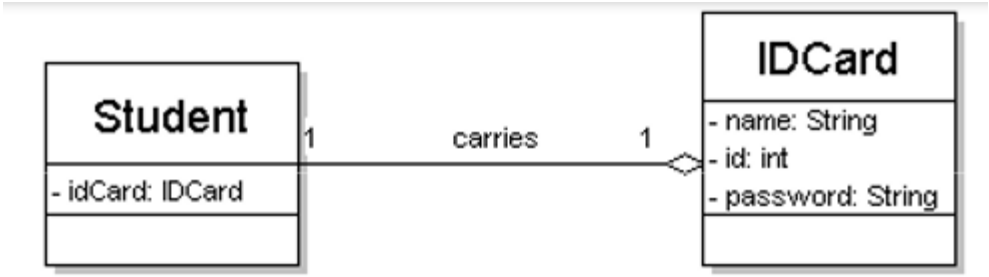
Asocijacija

- Asocijativna veza
  - Multiplikativnost: *, 1, 2..4, 3..*
  - Naziv
  - Smjer

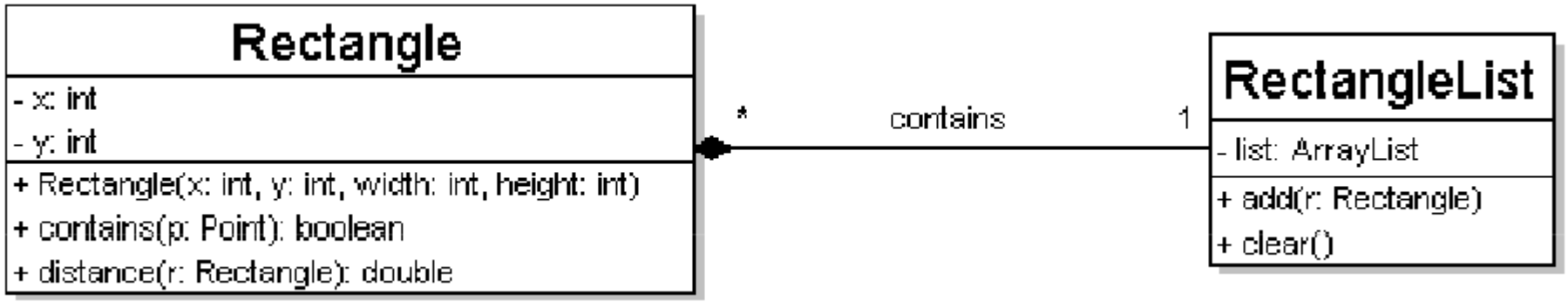


Multiplikativnost

- one-to-one
  - Svaki student ima tačno jednu ID karticu

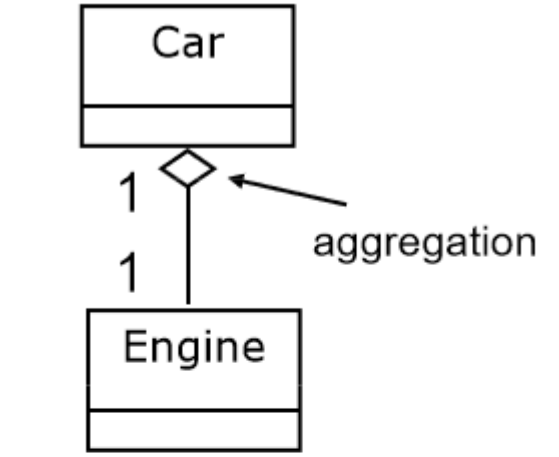


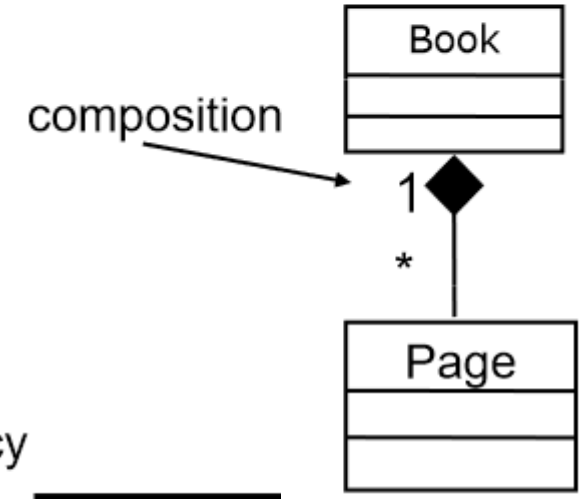
- one-to-many
  - Jedna lista sastoji se iz više pravougaonika

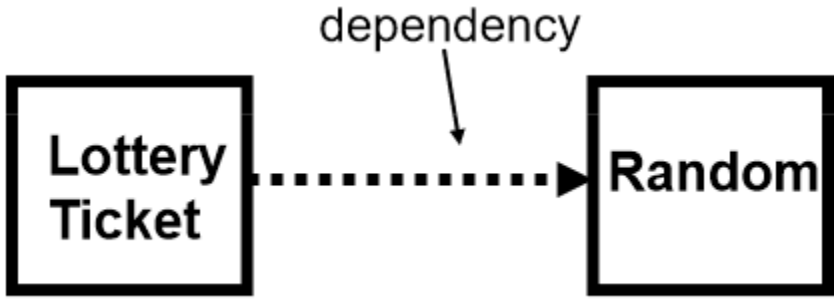


Tipovi asocijacije

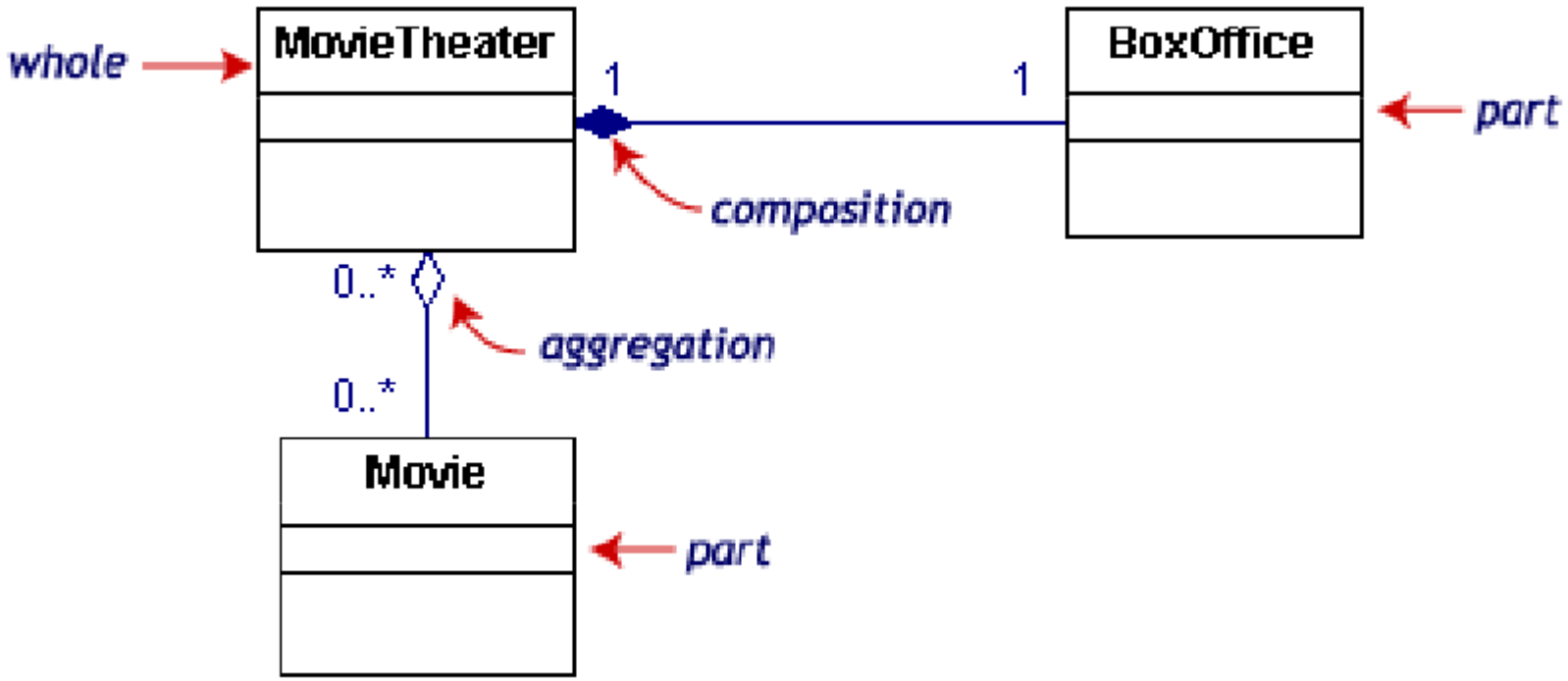
- Agregacija
  - „is part of“
- Kompozicija
  - „is entirely made of“
- Zavisnost
  - „uses temporarily“
  - Često predstavlja detalje implementacije



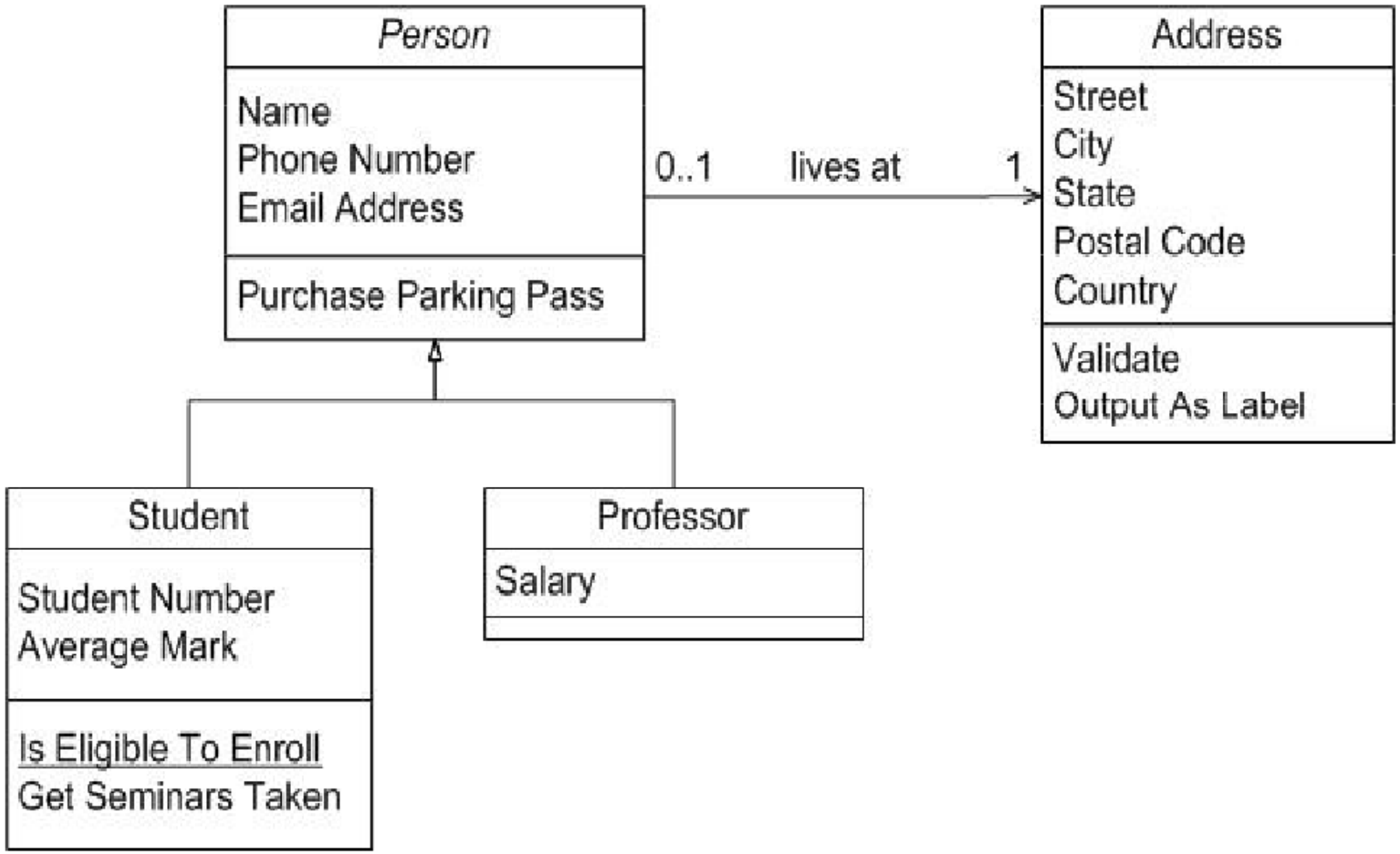




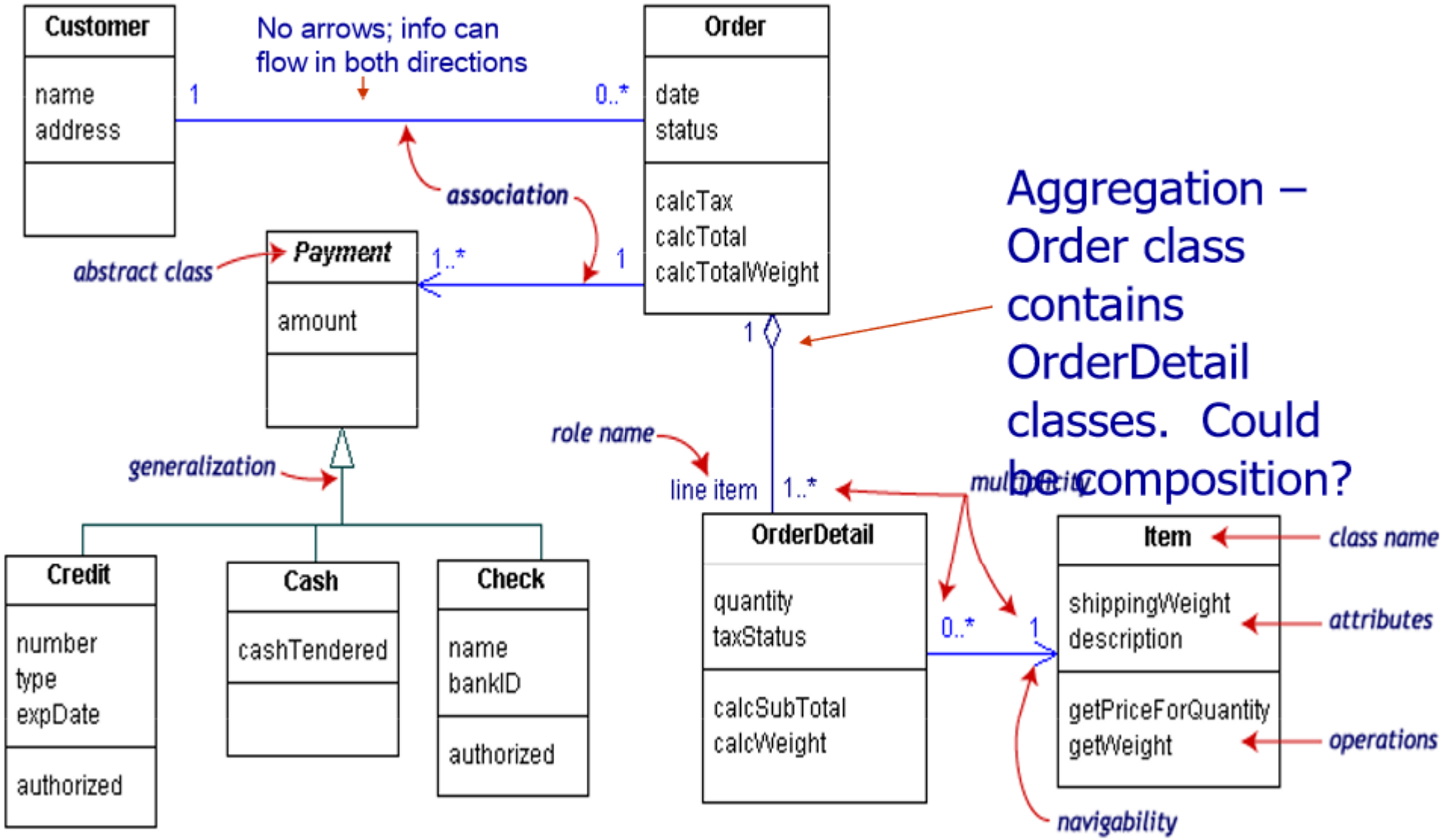
Kompozicija vs. Agregacija



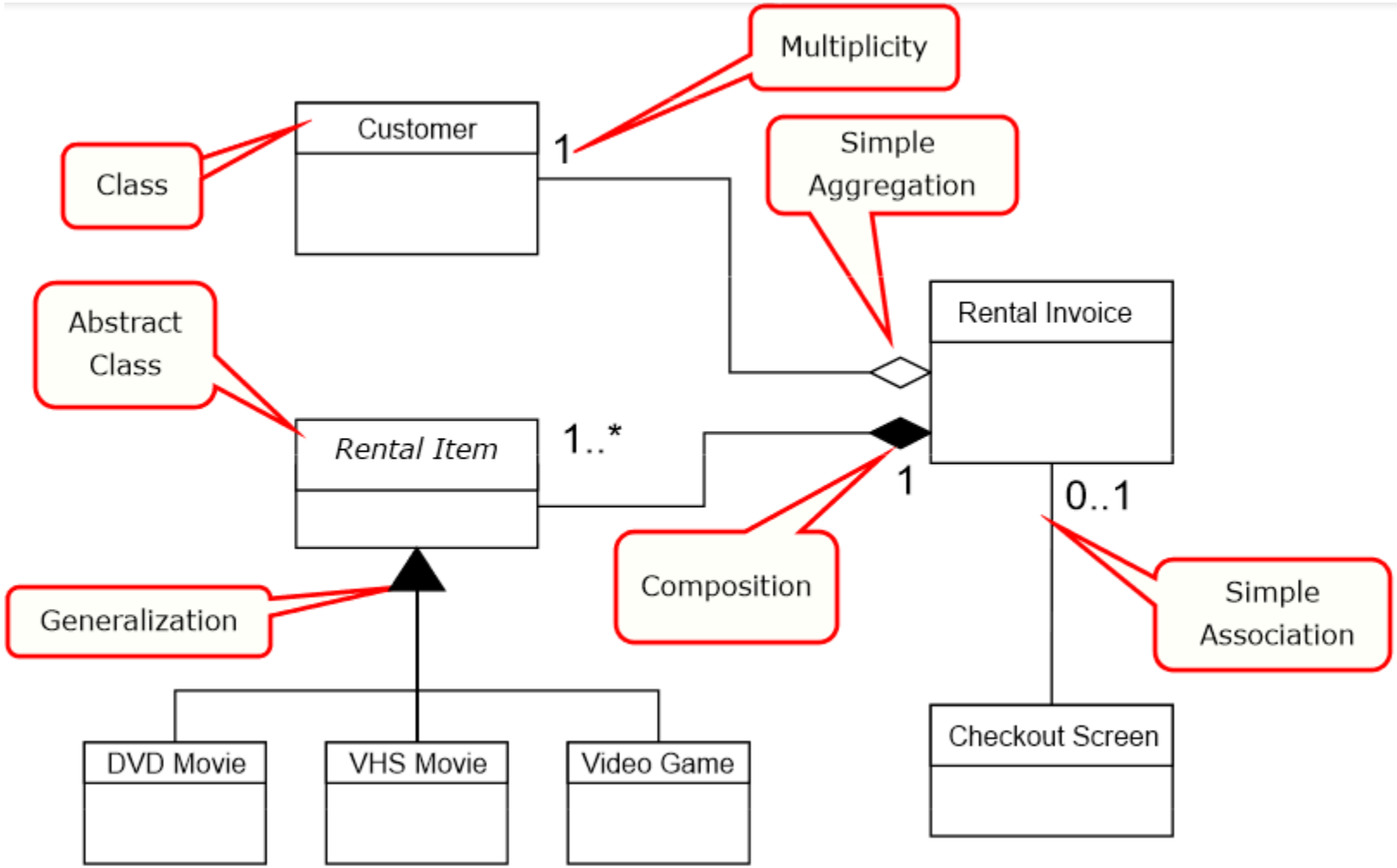
Primjer


Person
Name
Phone Number
Email Address
Purchase Parking Pass
0..1
lives at
1
Address
Street
City
State
Postal Code
Country
Validate
Output As Label
Student
Student Number
Average Mark
Is Eligible To Enroll
Get Seminars Taken
Professor
Salary

Primjer 2


Customer
name
address
No arrows; info can flow in both directions
1
0..*
Order
date
status
calcTax
calcTotal
calcTotalWeight
association
1..*
1
abstract class
Payment
amount
generalization
role name
line item
1..*
multiplicity
1
Aggregation – Order class contains OrderDetail classes. Could be composition?
Credit
number
type
expDate
authorized
Cash
cashTendered
Check
name
bankID
authorized
OrderDetail
quantity
taxStatus
calcSubTotal
calcWeight
0..*
1
Item
shippingWeight
description
getPriceForQuantity
getWeight
class name
attributes
operations
navigability

Primjer 3


Multiplicity
Customer
1
Class
Simple
Aggregation
Abstract
Class
Rental Invoice
Rental Item
1..*
1
0..1
Composition
Simple
Association
Generalization
DVD Movie
VHS Movie
Video Game
Checkout Screen

Zadatak

- Poker sistem
  - 2-8 igrača, igrač može da bude kompjuter
  - Svaki igrač ima naziv i skup žetona
  - Kompjuter se podešava na nivoe: lak, srednji, težak
  - Osnovni tok jedne „runde“
    - Prikupljanje uloga, miješanje karata, dijeljenje po 2 karte
    - „betting round“, dijeljenje još 3 karte
    - Dodatne „betting rounds“, igrač može da odustane, preskoči ili poveća ulog
    - Na kraju runde, ako su ostala bar dva igrača, upoređuju se karte, najbolje pokarte pobjeđuju i osvajaju sav ulog