

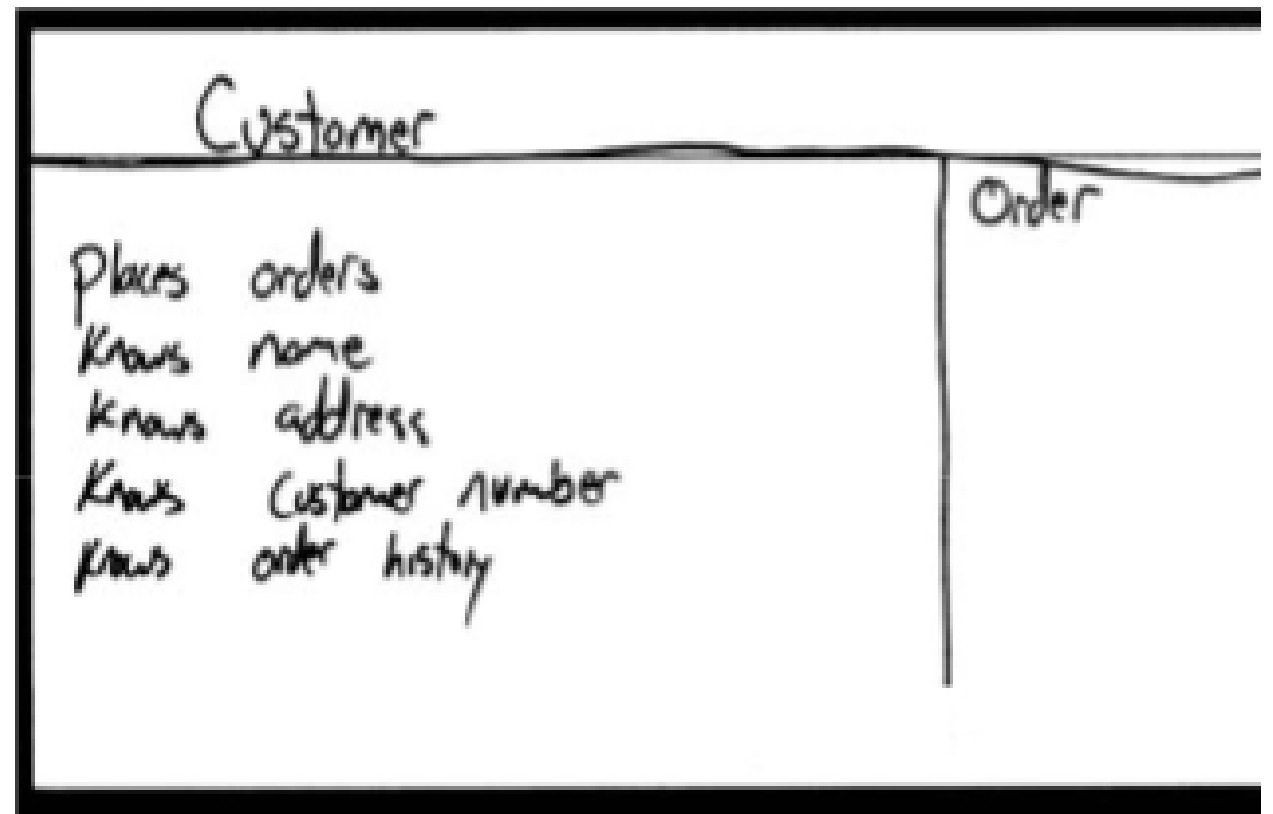
Uvod u UML

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
 - Interaction diagrams

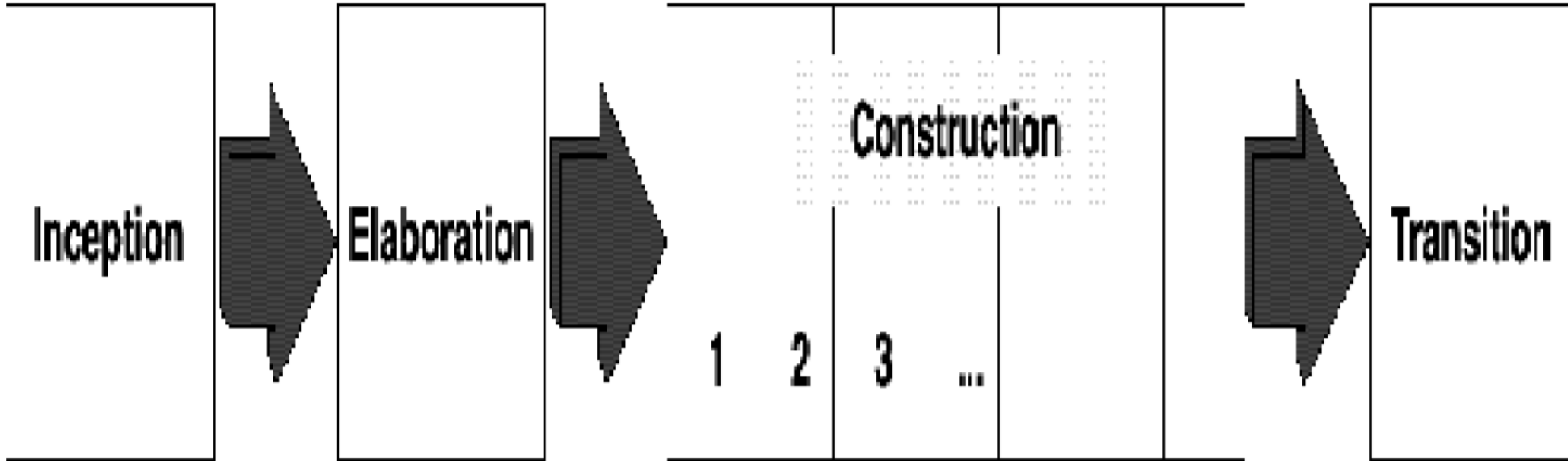
UML

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

Primjena

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

Proces razvoja softvera



Proces razvoja softvera (2)

- Inception
 - Inicijalna analiza da bi se odredio opseg, veličina, cijena, dobit iz projekta
 - Ograničena na nekoliko dana
 - Dokument sa zahtjevima

Proces razvoja softvera (3)

- Elaboration
 - Requirement risks
 - Definirati sve use case-ove (use case dijagrami)
 - Skicirati konceptualni model domena (dijagrami klasa, dijagrami aktivnosti)
 - Technological risks
 - Izrada prototipa, testiranje kompatibilnosti različitih komponenti
 - Definisanje arhitekture sistema (package, deployment dijagrami)
 - Skills risks
 - Political risks
- Kraj faze je kada
 - Moguća je procjena implementacije svakog use case-a
 - Identifikovani svi rizici i planovi u tim slučajevima

Proces razvoja softvera (4)

- Construction

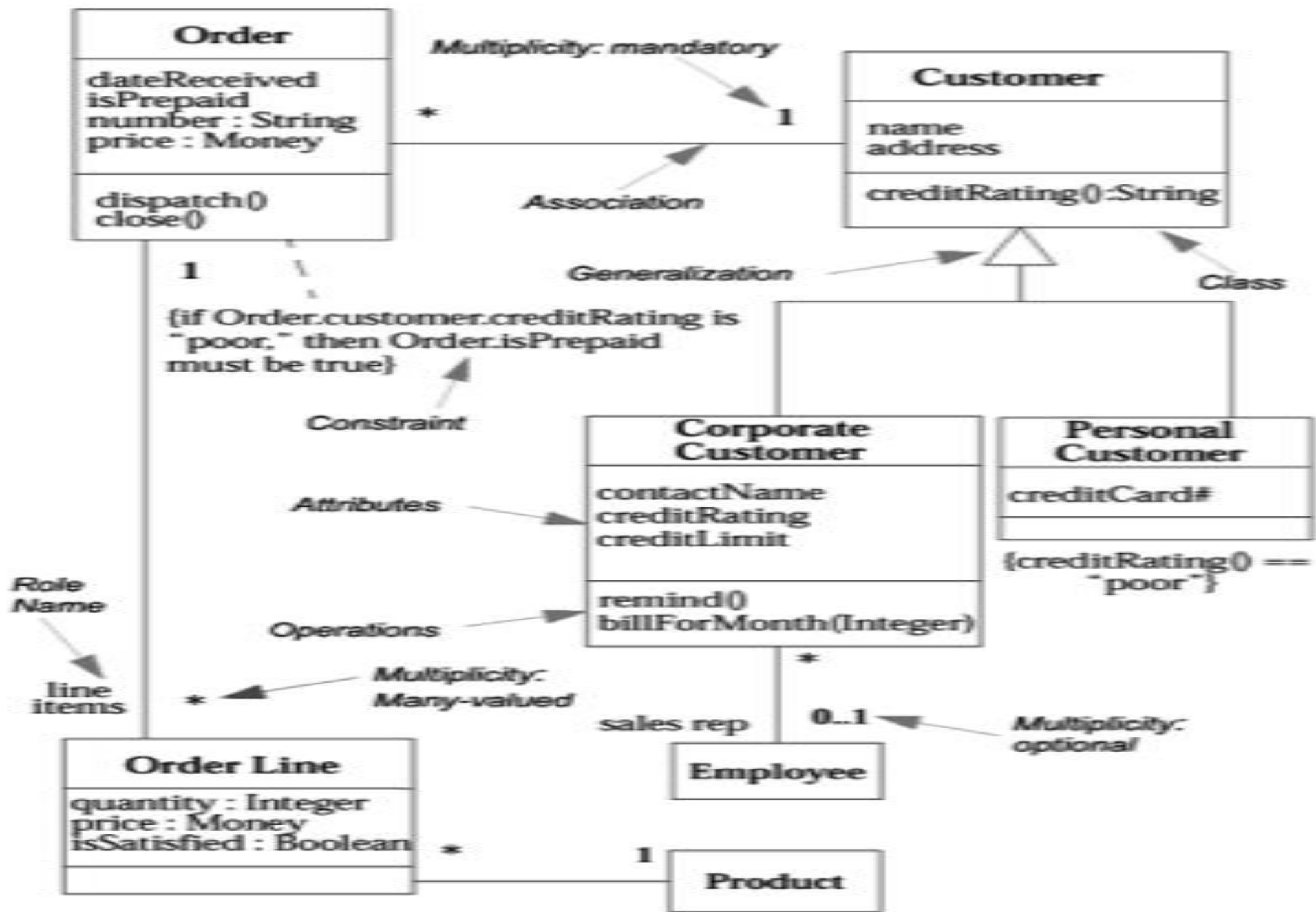
- Definisanje serije iteracija (npr. jedan use case = jedna iteracija) i definisanje funkcionalnosti koje u svakoj iteraciji treba da budu implementirane
- Procjena vremena za svaku iteraciju: analiza, dizajn, kodiranje, testiranje, integracija
- Iteracije su mini-projekti
 - Smanjenje rizika
 - Inkrementalne, sljedeća iteracije nadograđuju ranije razvijene use case-ove
 - Iterativne u smislu ponovnog pisanja nekih dijelova programskog koda

UML dijagrami klasa

- Dijagram klasa
 - Sadrži sve klase u sistemu
 - Attribute i metode
 - Veze između klasa
- Dijagram klasa ne sadrži
 - Detalje veza
 - Detalje algoritama kojima se implementira određeno ponašanje

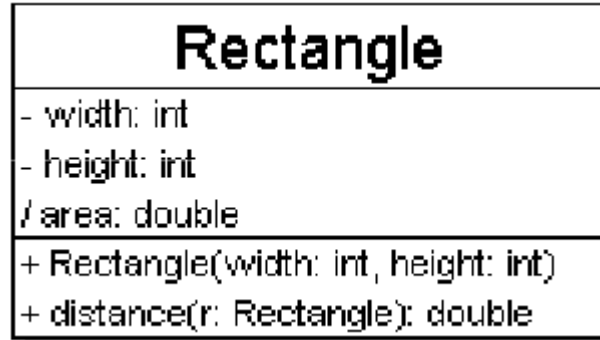
Dijagrami klasa

- Perspektive
 - Konceptualna, stereotip <<type>>
 - Specifikacija, stereotip <<type>>
 - Implementaciona, stereotip <<implementation class>>



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

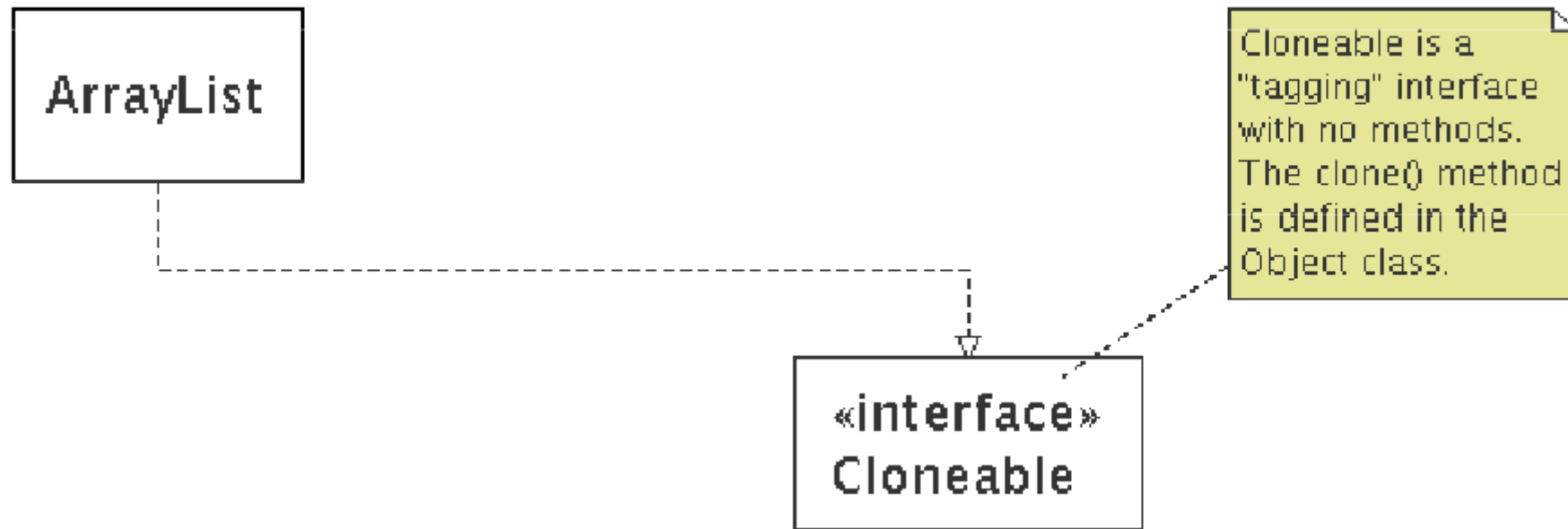
Rectangle
- width: int
- height: int
/ area: double
+ Rectangle(width: int, height: int)
+ distance(r: Rectangle): double

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

Rectangle
- width: int
- height: int
/ area: double
+ Rectangle(width: int, height: int)
+ distance(r: Rectangle): double

Komentari



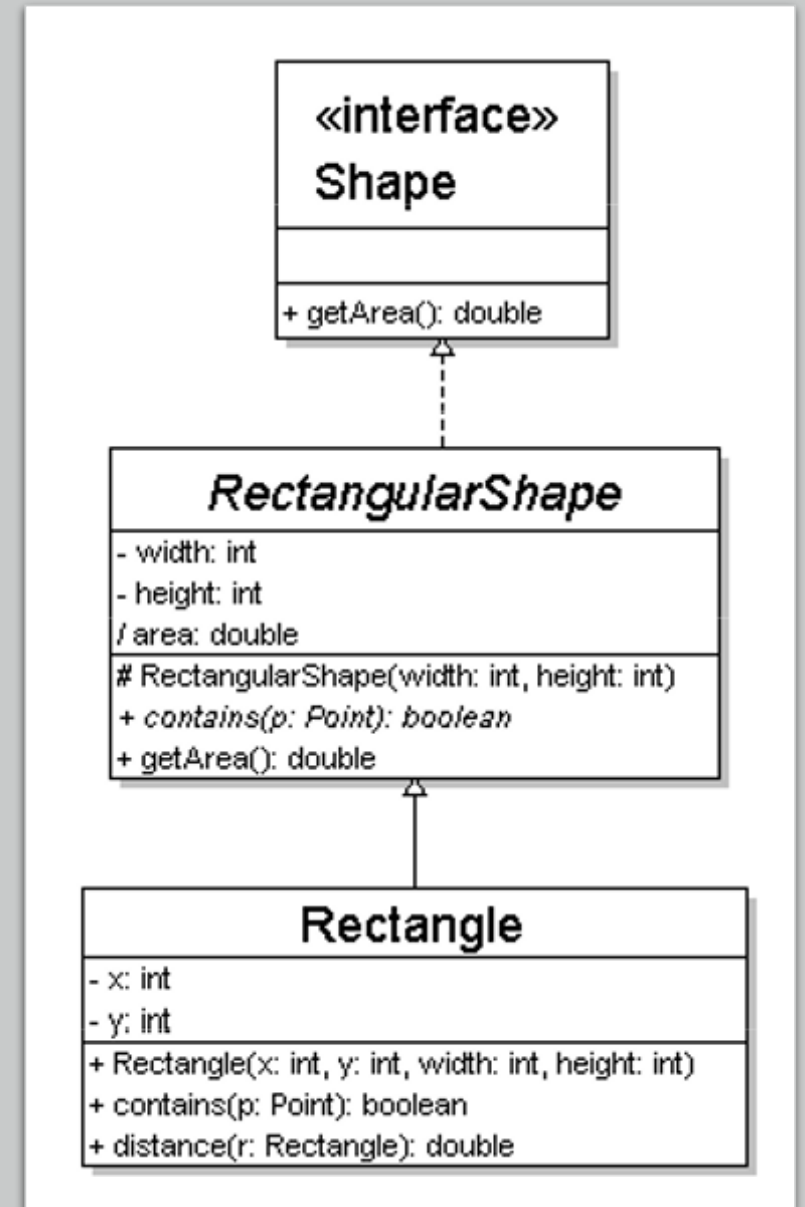
Veze između klasa

- Generalizacija
 - Nasleđivanje
 - Implementacija interfejsa

- Asocijacija
 - Zavisnost
 - Agregacija
 - Kompozicija

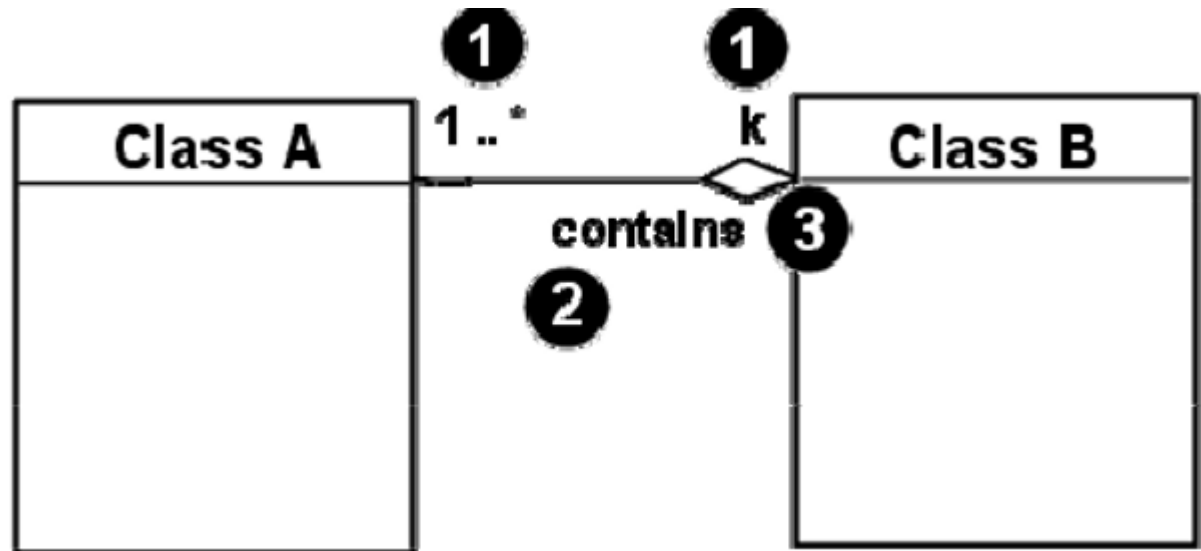
Generalizacija

- Top down hijerarhija
- Razlikuju se slučajevi kada je roditelj
 - Klasa
 - apstraktna klasa
 - Interfejs



Asocijacija

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

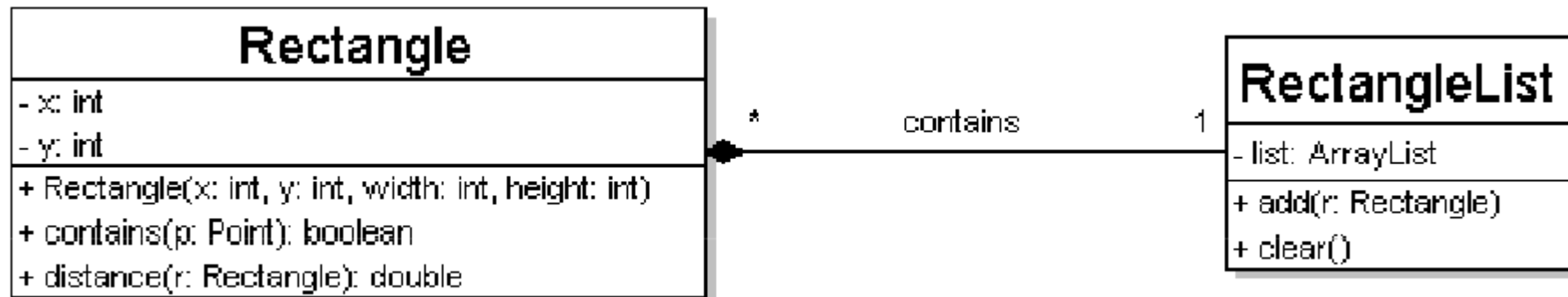


Multiplikativnost

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

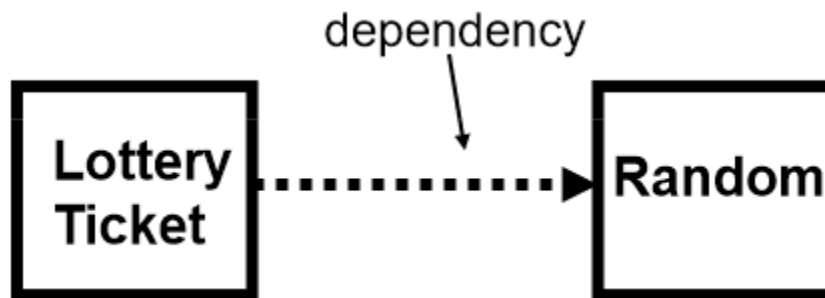
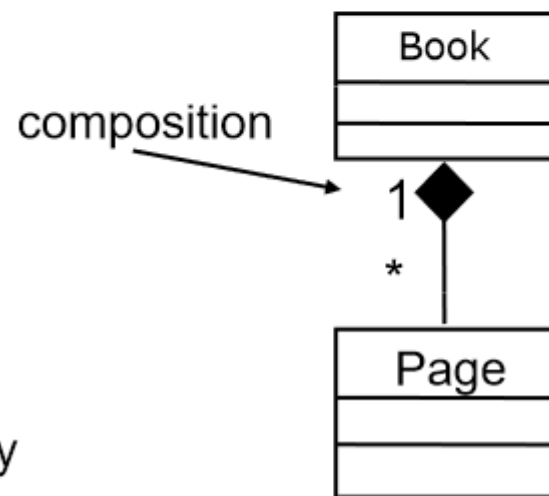
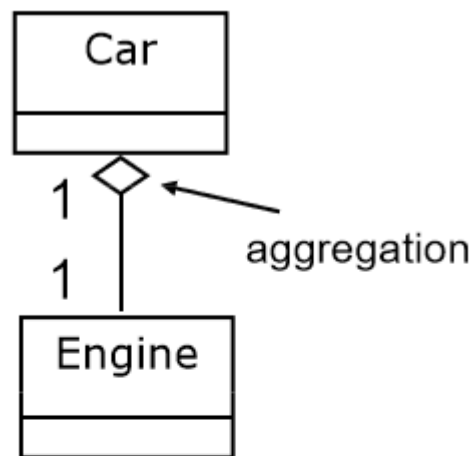


- one-to-many
 - Jedna lista sastoji se iz više pravougaonika

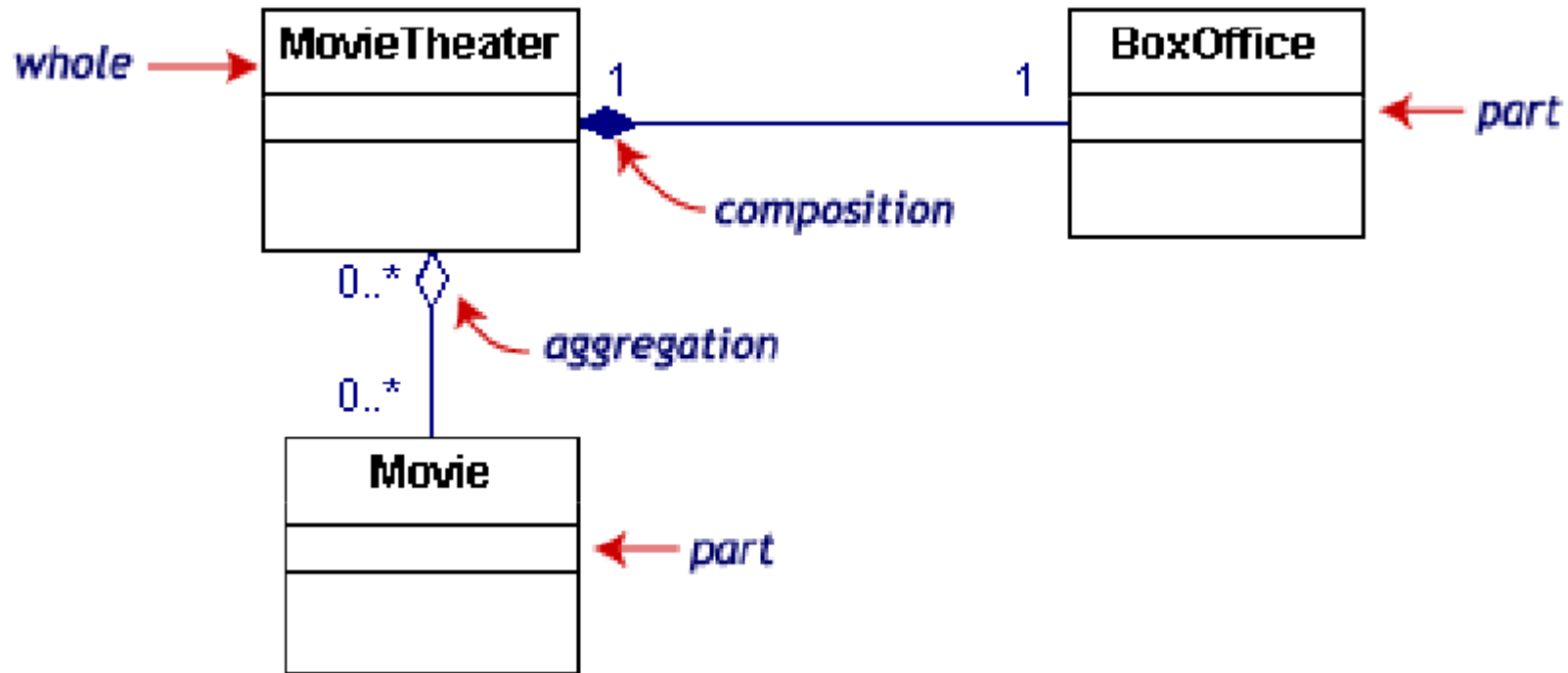


Tipovi asocijacije

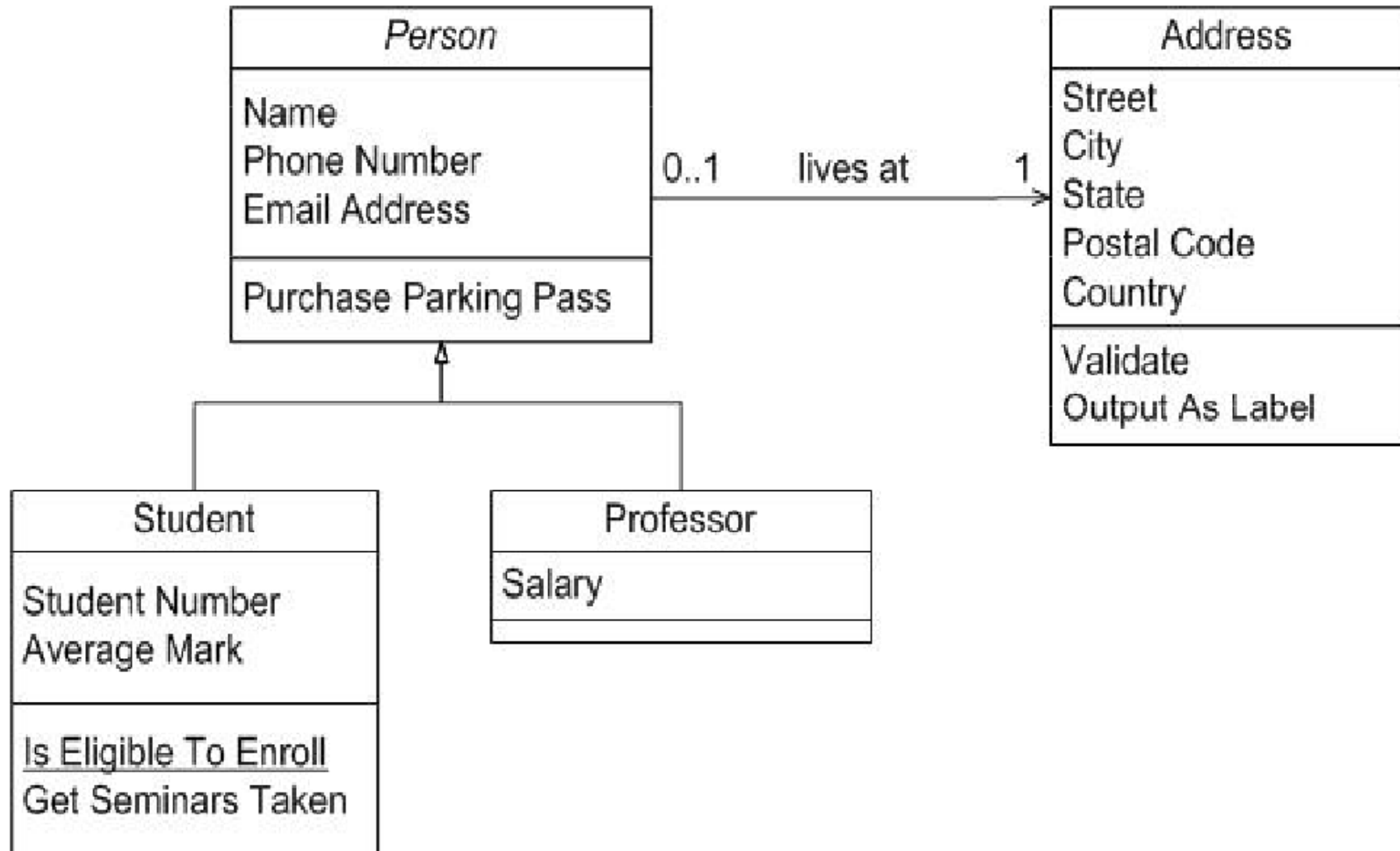
- Agregacija
 - „is part of“
- Kompozicija
 - „is entirely made of“
- Zavisnost
 - „uses temporarily“
 - Često predstavlja detalje implementacije



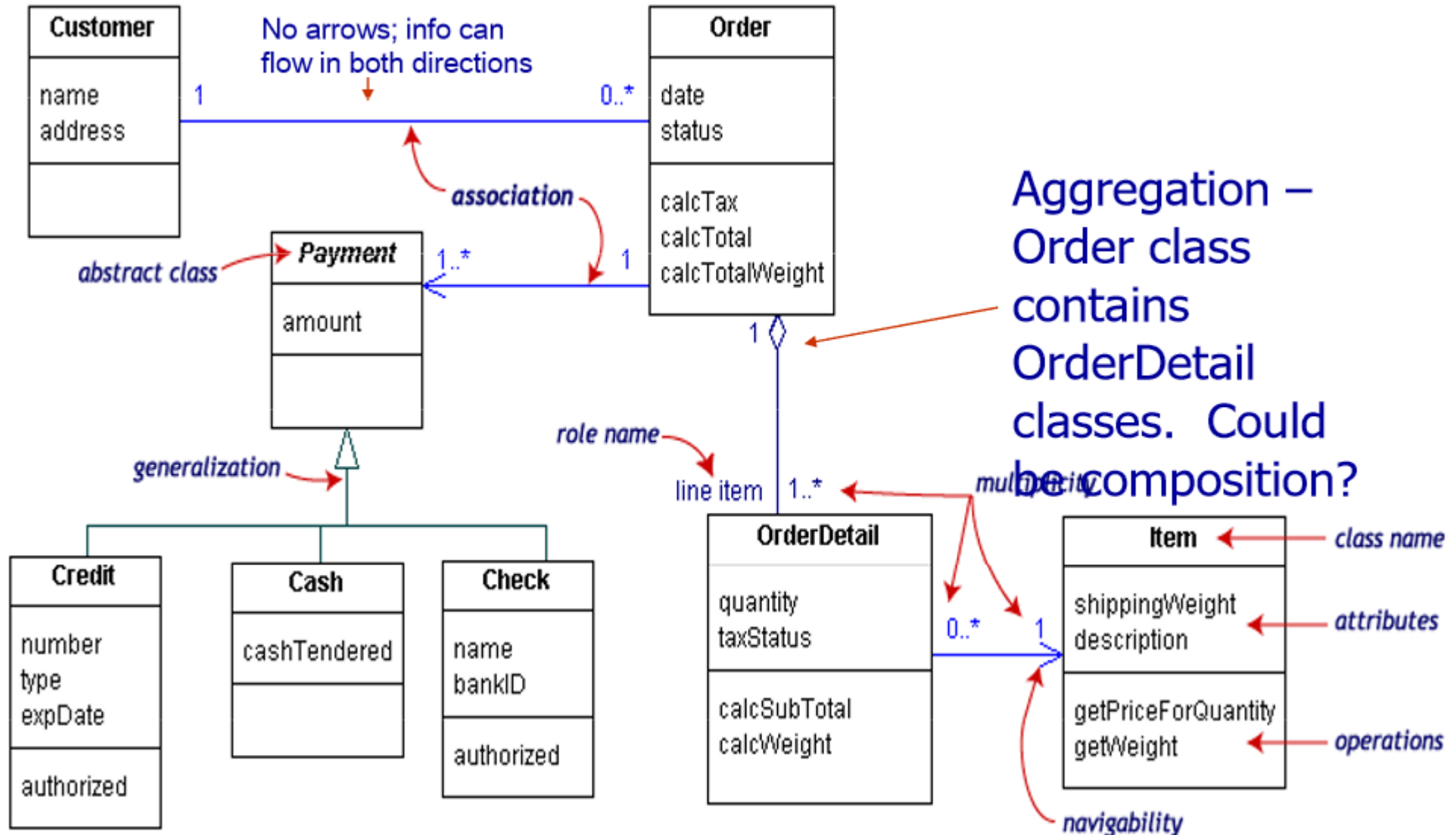
Kompozicija vs. Agregacija



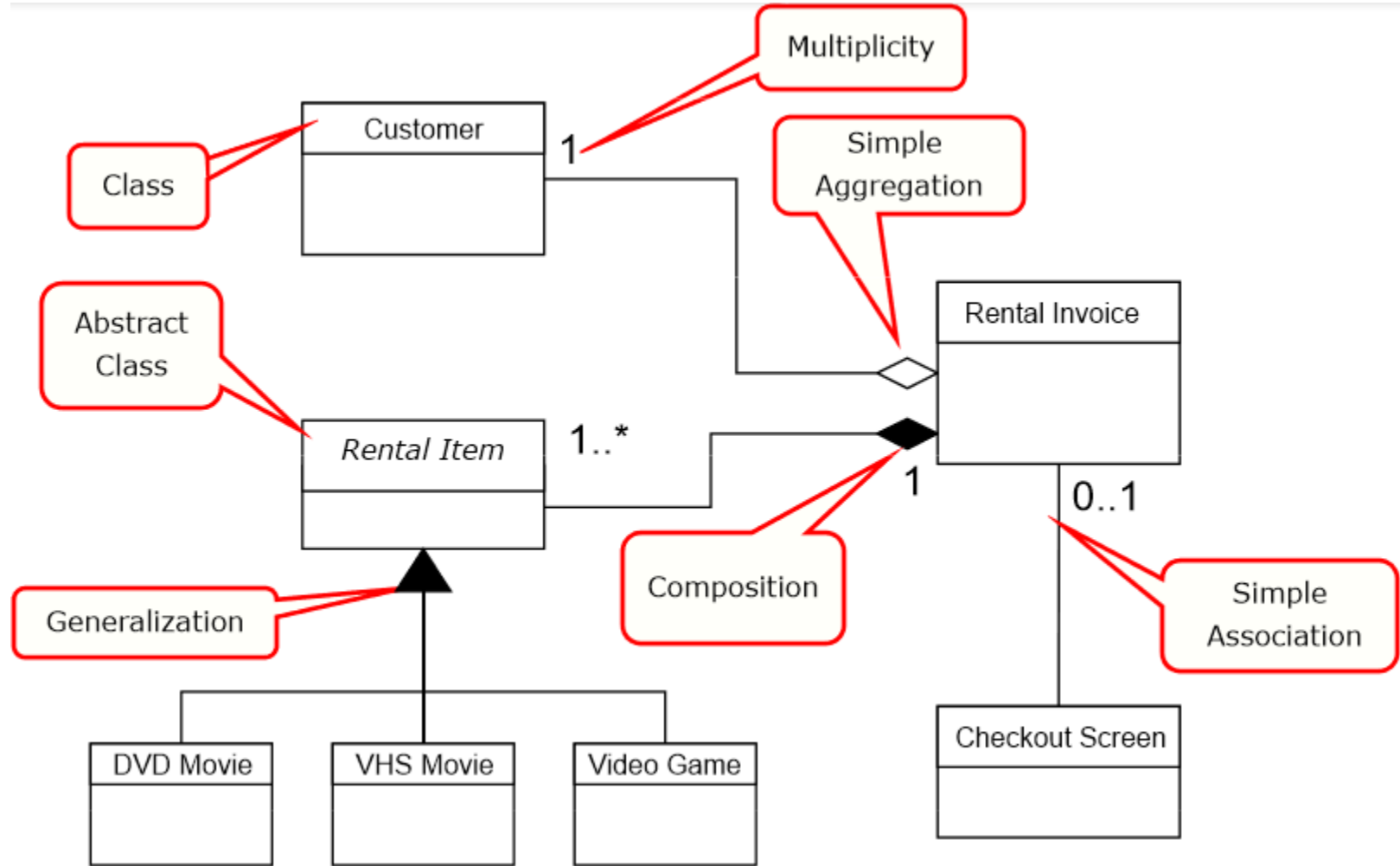
Primjer



Primjer 2



Primjer 3



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