

Introduktion till UML

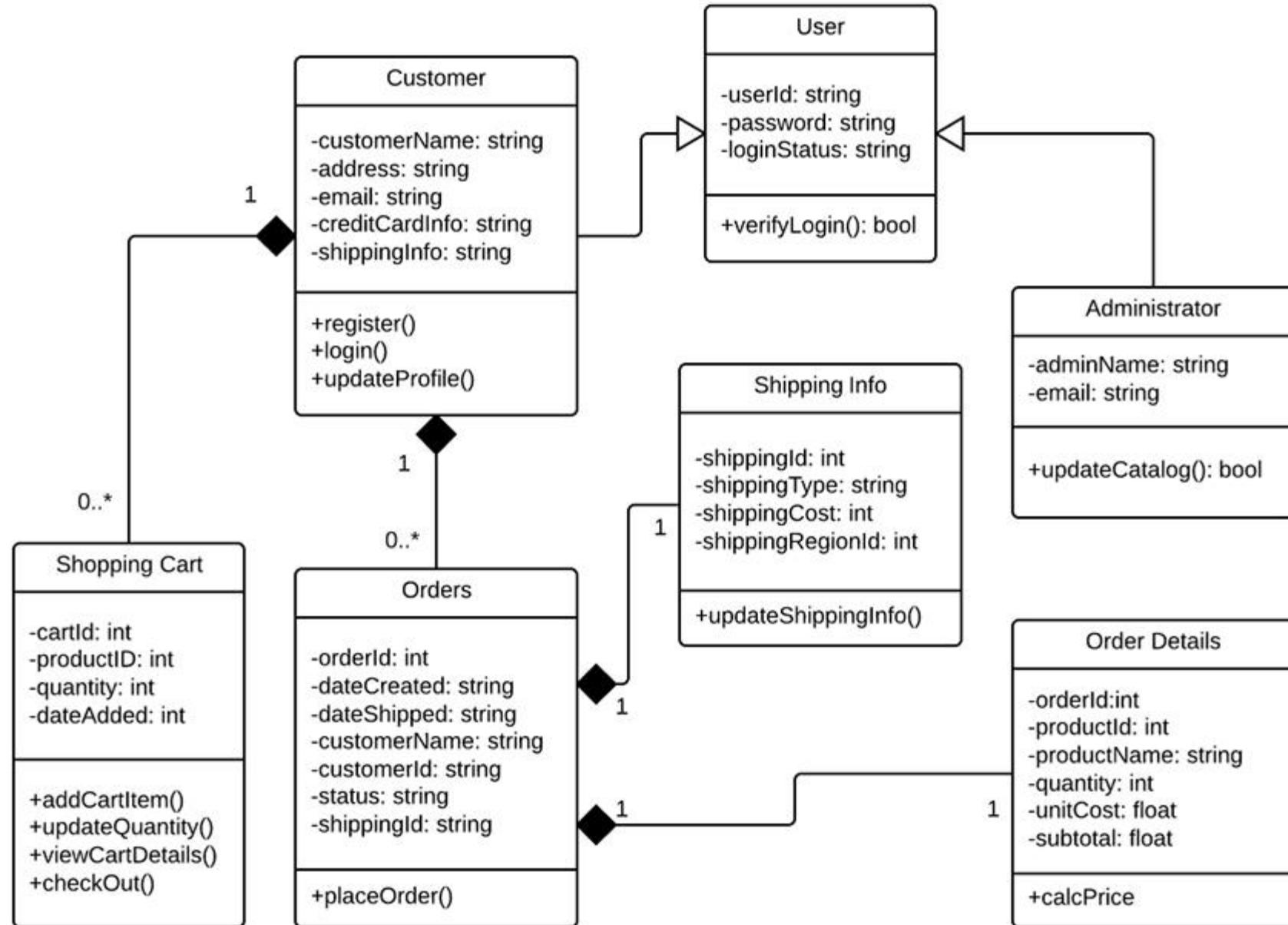
Unified Modeling Language

Momentet UML i kursen

- Dagens föreläsning
- Torsdagens workshop
- Återkoppling genom resterande del av kursen
 - Föreläsningar, laborationer, inlämningsuppgifter







Vad innebär arkitektur av ett system för:

- Slutanvändare
- Projektledare
- Systemingenjör
- Utvecklare
- Testare
- Underhållare av system

Modellerings – Modell av verkligheten

- Målet med UML är att skapa en visuell bild av ett mjukvara
 - Men kan användas till mer!
- Vi vill göra våra mjukvaror enkla att förstå
 - För oss själva
 - För våra kollegor
 - För våra kunder
 - ...
- Vi gör ofta förenklingar, det viktigaste är att förstå hur systemet hänger ihop & hur de olika delarna relaterar till varandra.

Modellerings

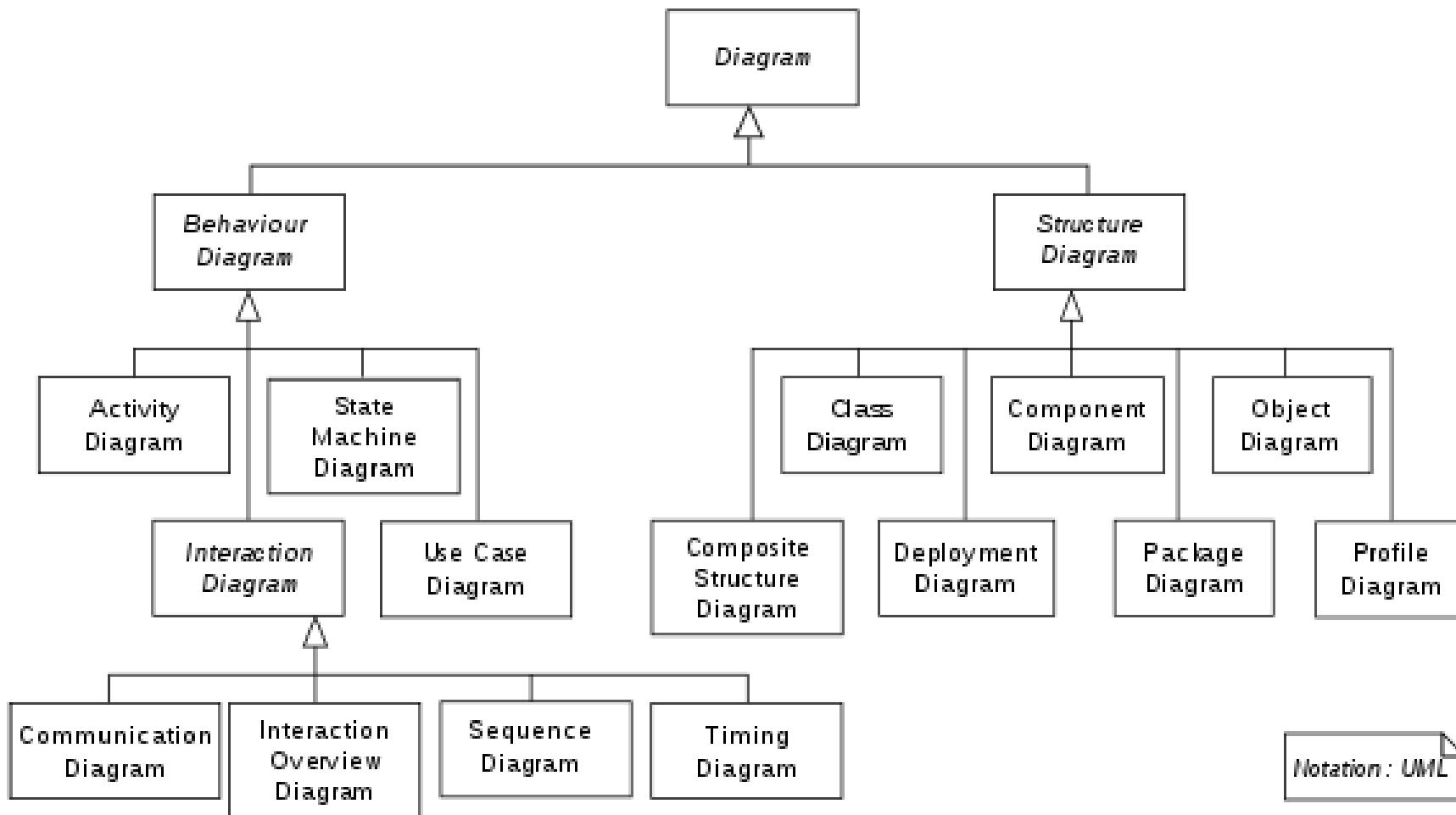
- Modellerings innebär att vi **designar mjukvara innan** vi programmerar dem
- Modellerings innebär en högre nivå av abstraktion av ett system
 - Vilket också gör att "icke-programmerare" har möjlighet att se hur ett system hänger ihop
- Modellerings blir således en "**analog**" bild av systemet
 - Jämför med ritningar för en byggnad – vi gör ritningarna **innan** vi bygger själv byggnaden
- Modellerings används för att kontrollera att **alla krav** blir uppfyllda innan vi faktiskt börjar att bygga vårt system
 - Riskerar att bli **svindyrt** om man missar krav här…
- Modellerings är också ett verktyg att presentera sitt system för olika intressenter

UML – En standard för att modellera system

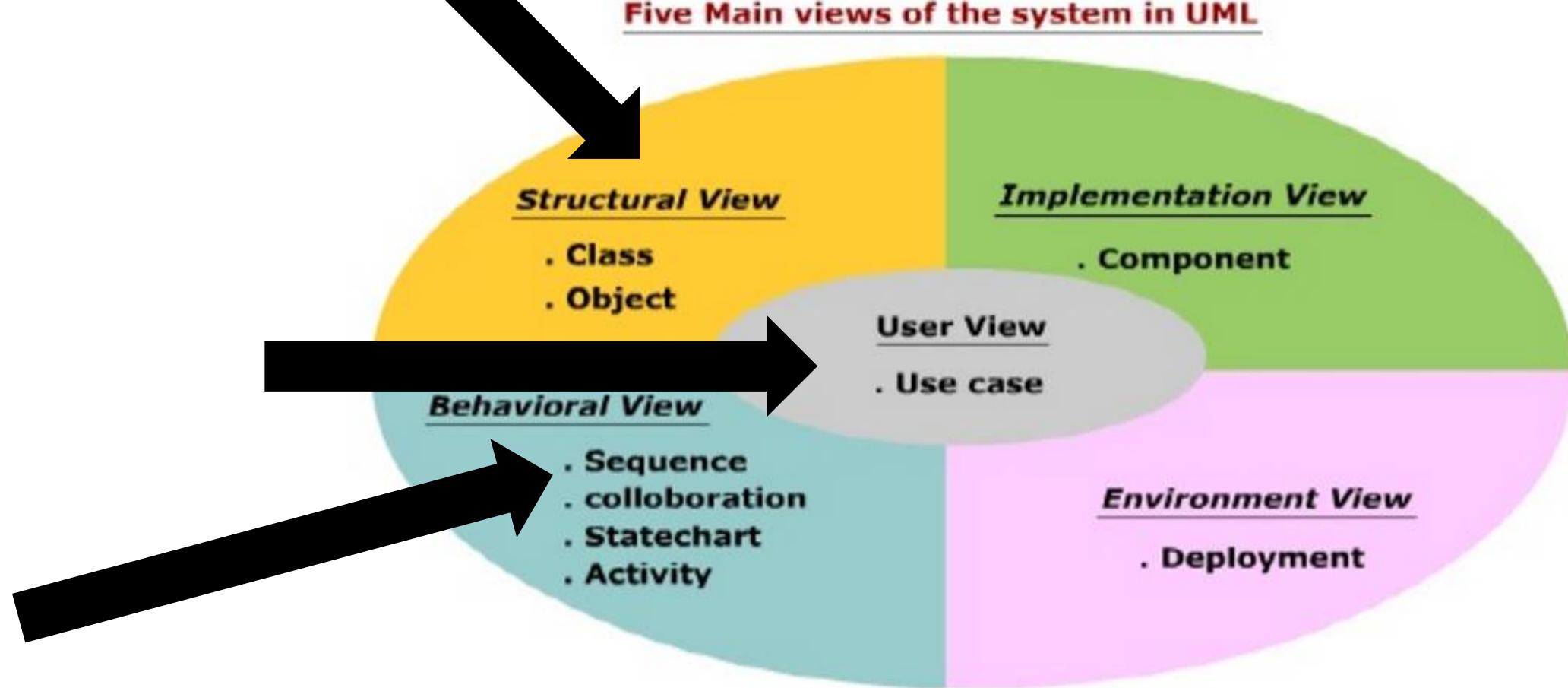
UML - Bakgrund

- UML – Syftet är att erbjuda ett standardiserat sätt att visuellt beskriva ett system.
- Skapades 1994-1995 av
 - Grady Booch
 - Ivar Jacobson
 - James Rumbaugh
- Används ofta tillsammans med objektorientering
- Ett utbrett och standardiserat sätt att modellera mjukvara

Olika diagram inom UML



Olika delar i UML



**Hur har ni modellerat upp
era system tidigare?**

UML inom projekt

Små projekt

- Enkla att förstå
- Enkla att få översikt över
- Lätta att lära sig
- Lite kod

Stora projekt

- Komplexa att förstå
- Svåra att få översikt över
- Svåra att lära sig
- Massor av kod

Nyckelord för UML

- Abstraktion
- Förenkling
- Hantera komplexitet
- En brygga mellan olika domäner
 - Programmerare & designer
 - Utvecklare & kunder
 - Etc.
- Diagram
 - Struktur
 - Beteende

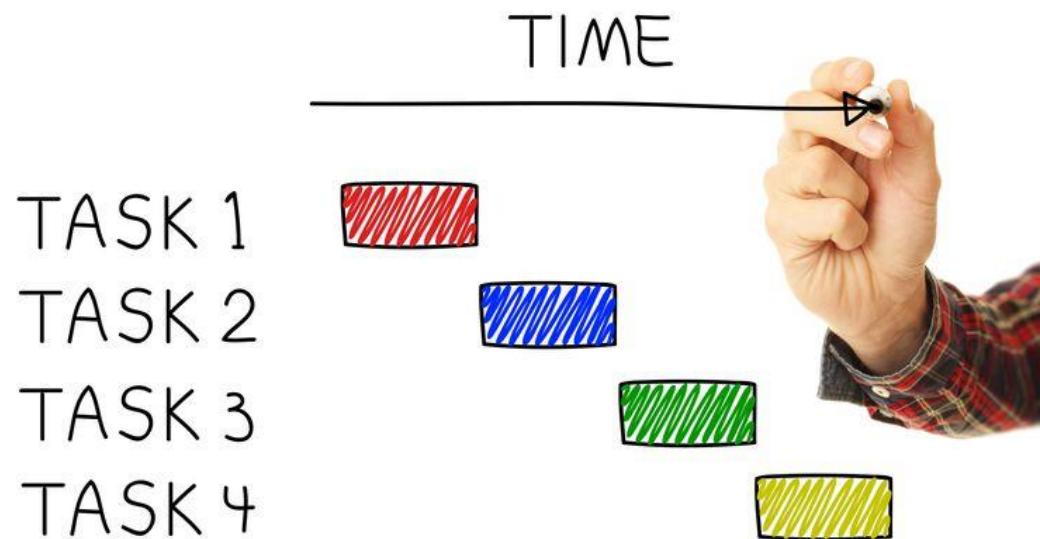


Varför ska man välja UML?

- Det är ett formellt språk
 - Varje del av språket är definierat, vilket leder till färre egna tolkningar
- Det är koncist
 - Hela språket är uppbyggt av enkla figurer och notation
- Det är heltäckande
 - Olika delar täcker olika aspekter av system, vilket gör att det "räcker" med "bara" UML för ett system
- Det är skalbart & plattformsberoende
- Det är ett mycket språk som har lärt sig av sina misstag
- Det är en standard

Olika sätt att driva projekt

- Vattenfallsmetoden
 - Analys
 - Design
 - Kodning
 - Testning
- Iterativt (t.ex. Scrum)
 - Lös problemet i mindre bitar



Ni ska i mindre grupper (3-4) beskriva It's Learning som system, visuellt. Hur gör ni?

Målgrupper

- Slutanvändare
- Projektledare
- Systemingenjör
- Utvecklare
- Testare
- Underhållare av system

Del 2. Hur skulle ni strukturera upp er kod?

It's L

Anwände

• login

Student

• kommentar
• konto info
• lösung

admin
lärare

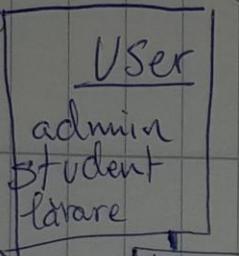
• gör test
• registrera
• läs upp

Uppgift

lösning

Uppgift

läs upp



Login

It's learning

fäsa
lämna dokument
lämna meddelande

skapa innehåll
skapa grupper
skapa sidor

skapa/tabort användare
ändra
inlägg

Admin

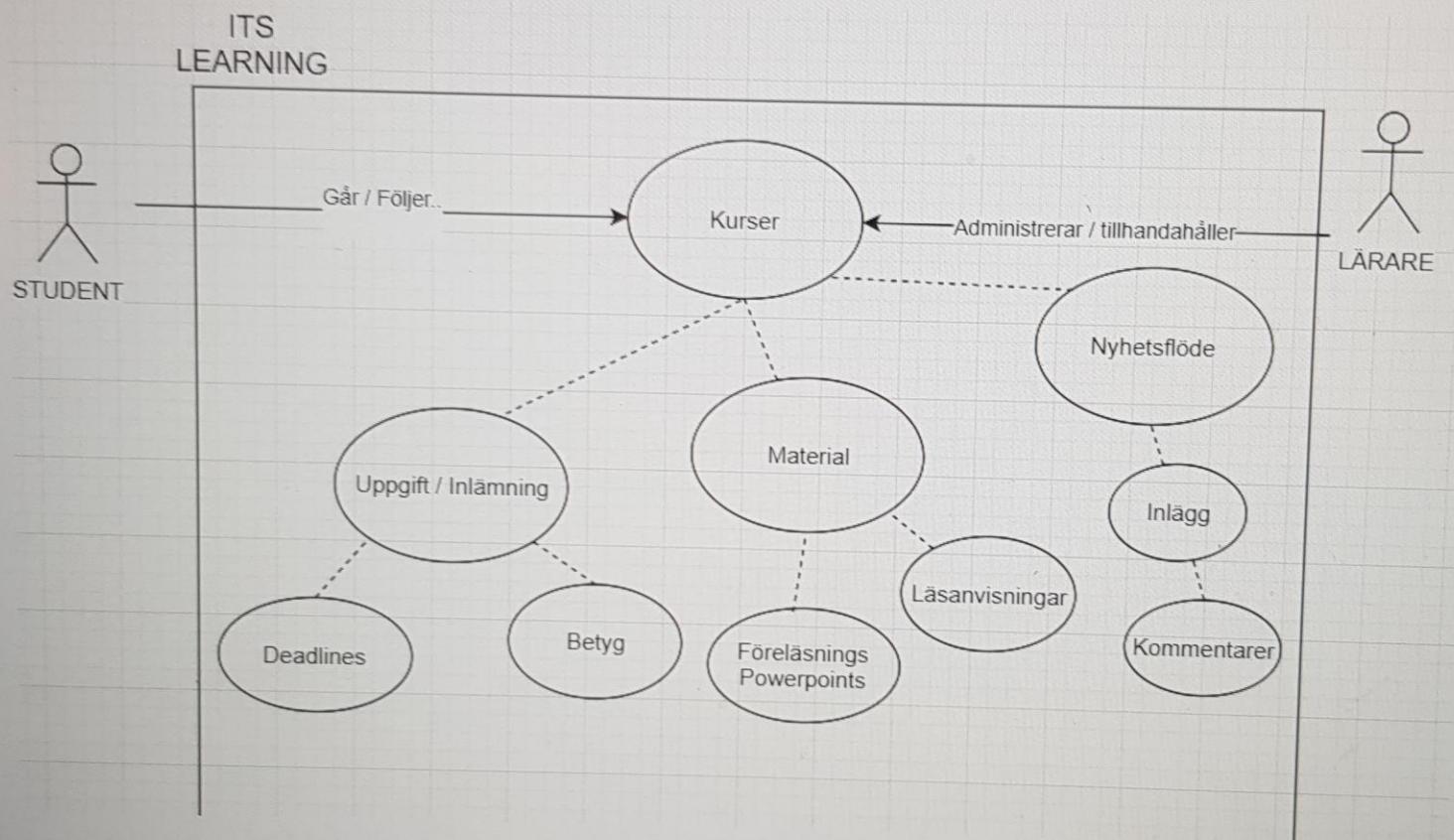
- Allt lärare & studenter kan
- ~~Hantera användare~~
- Hantera innehåll

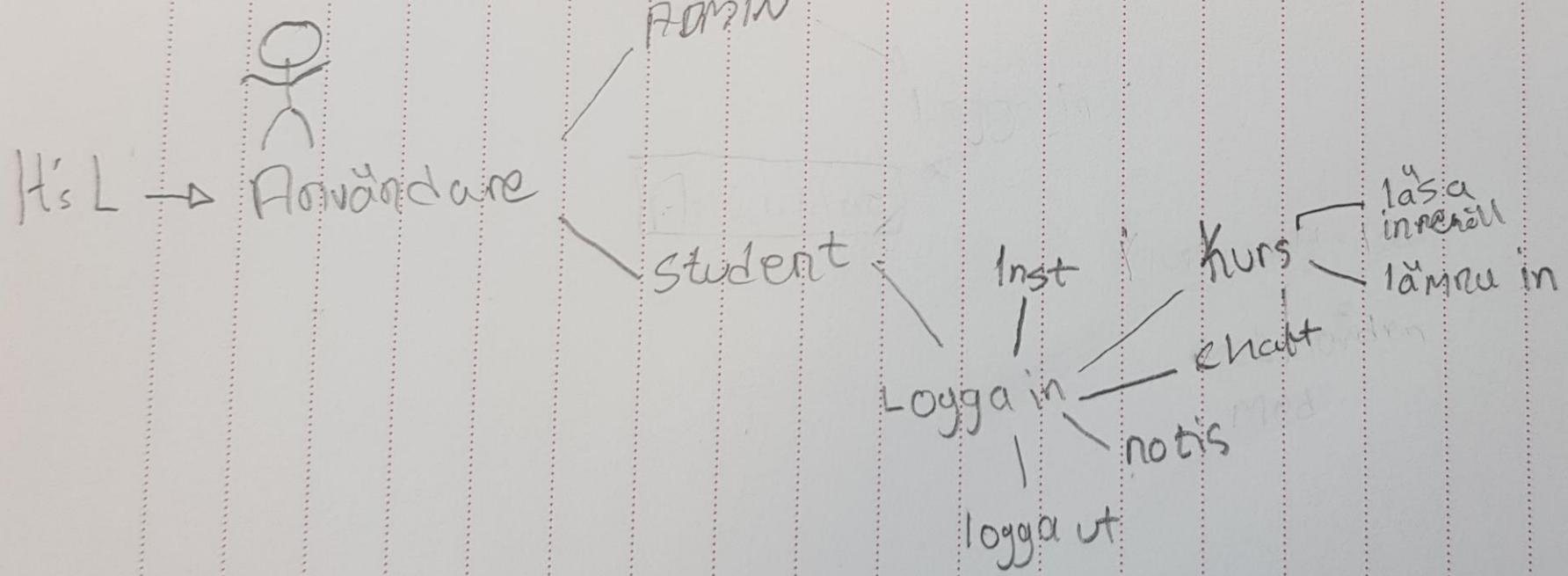
lärare

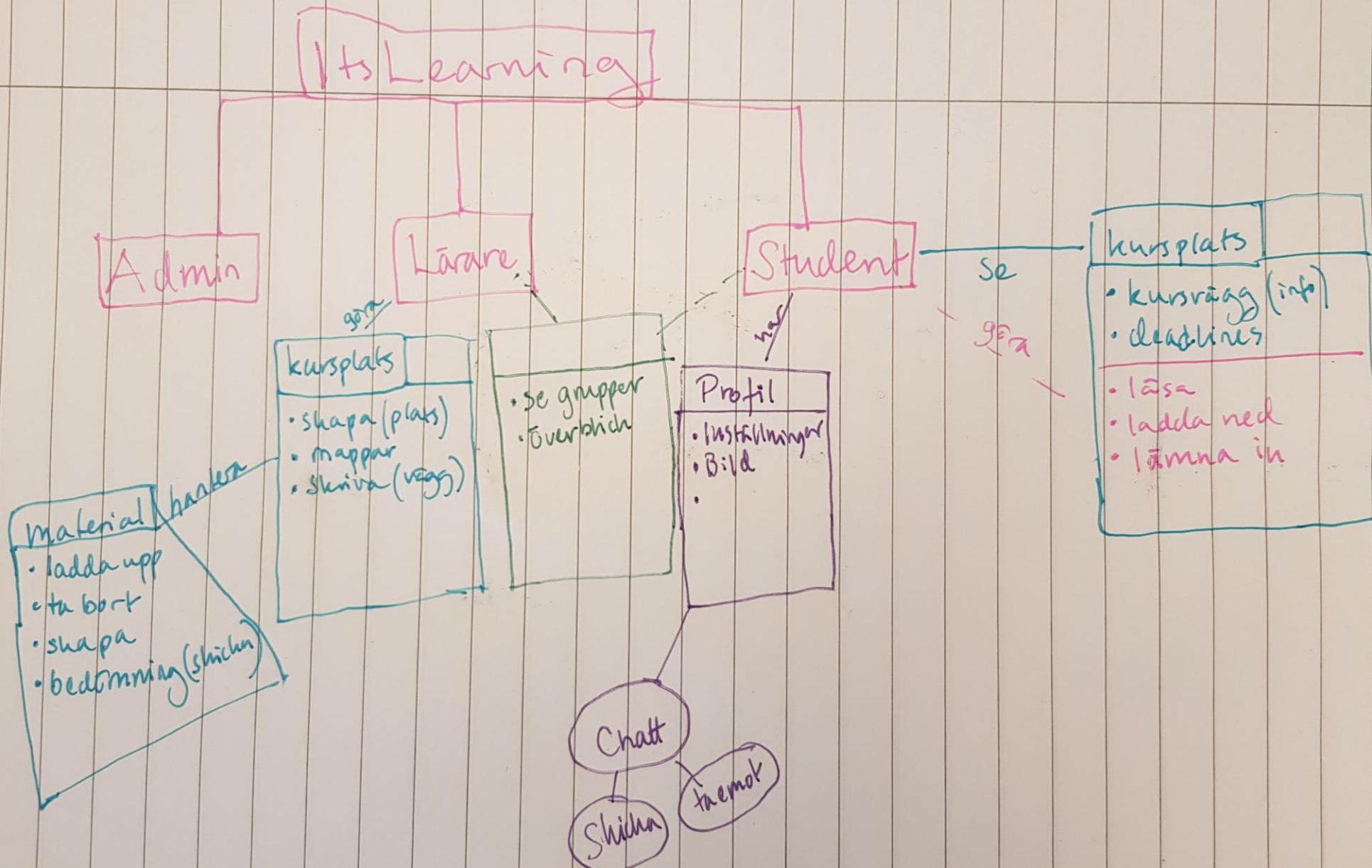
- Allt student kan göra
- + • skapa uppgifter
- avsluta uppgifter
- förändra uppgifter

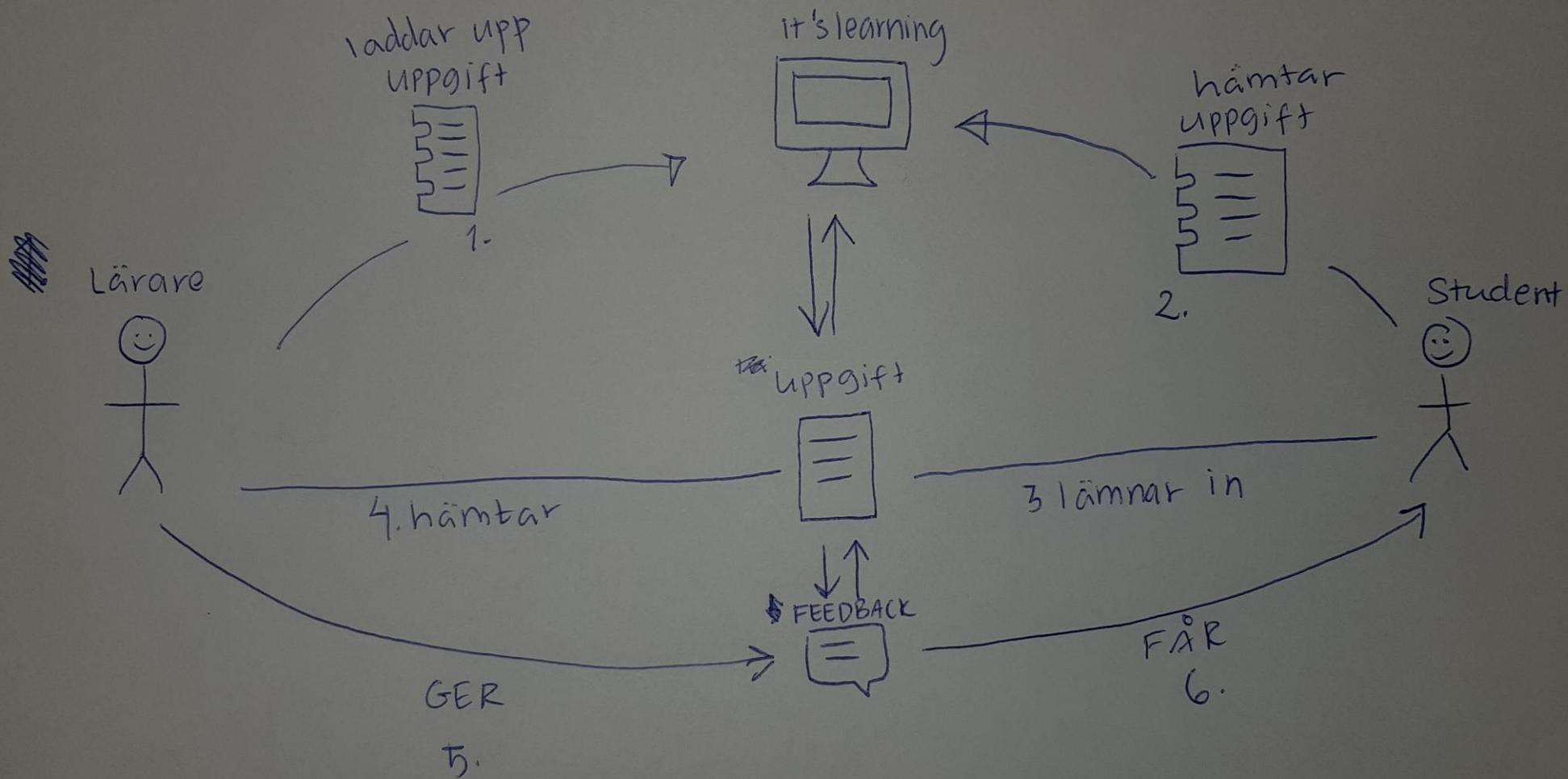
Student

- se inlägg
- kommentera inlägg
- ~~fäsa~~
- notisar
- Addera kommentarer och filer till skapad uppgift
- Be flöde / lista valbar information
- In/utlogg
- ~~Hantera användarprofil~~





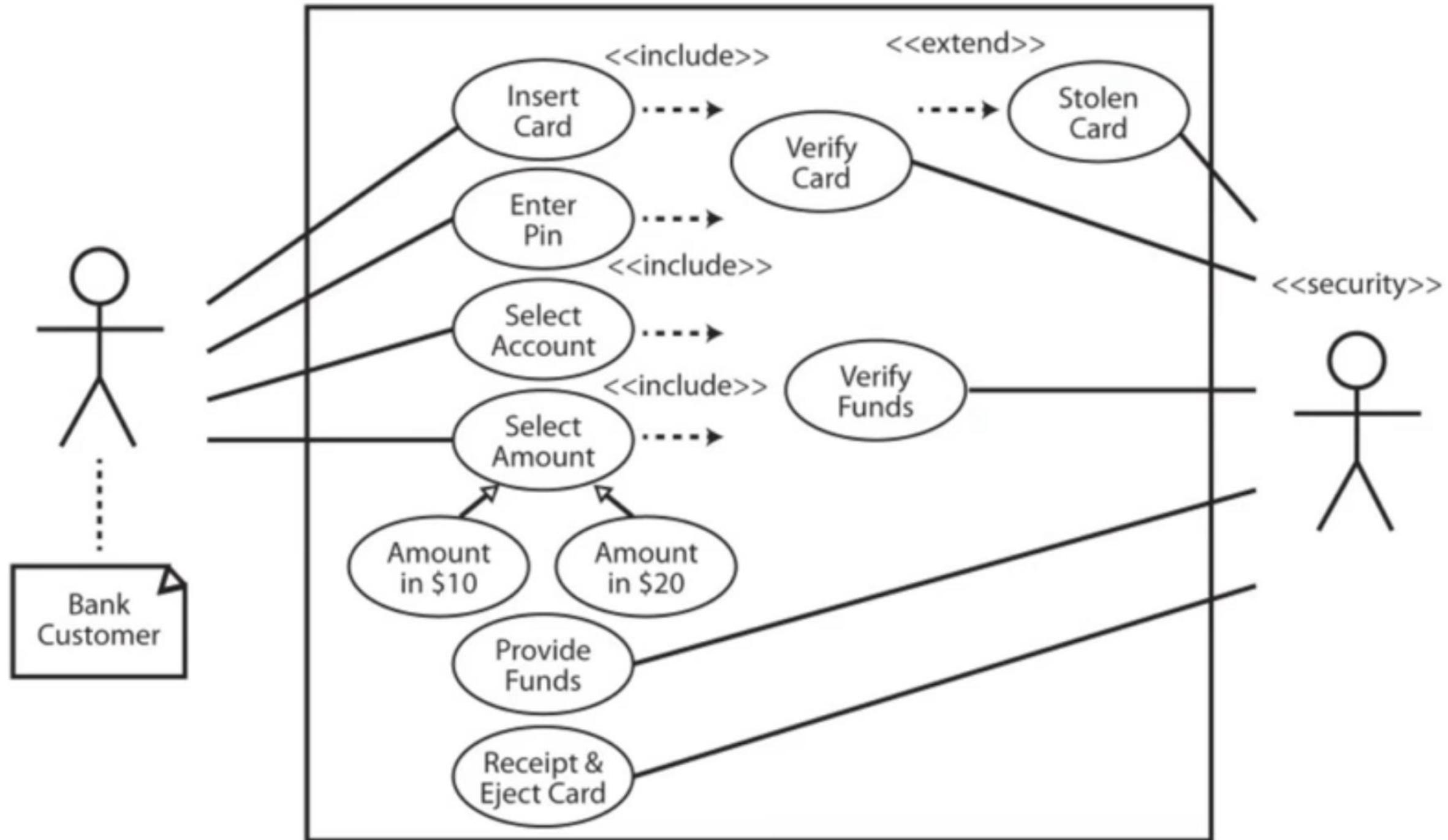




Use Case-diagram

Use Case-diagram

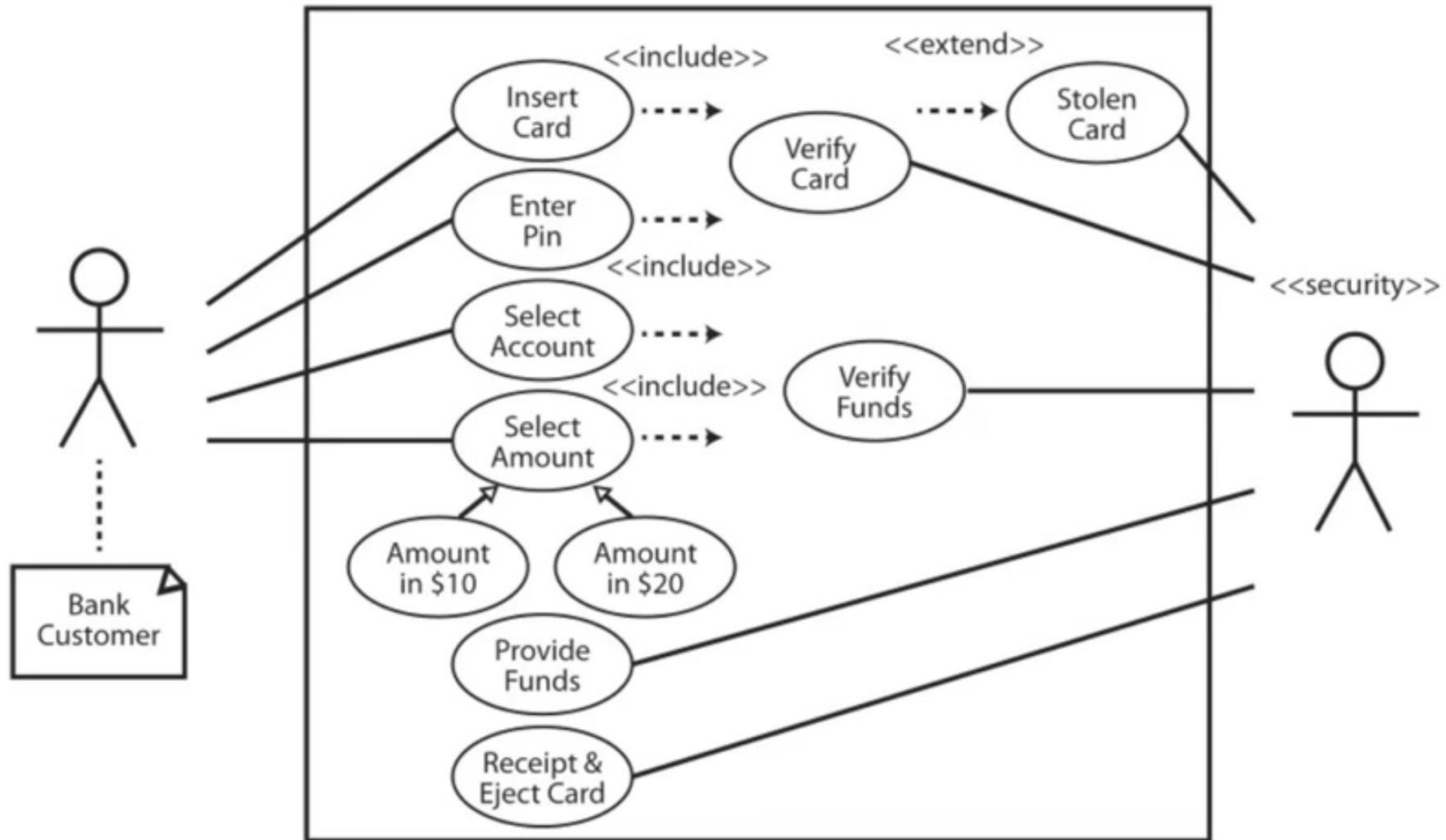
- Vilka är krav finns på systemet?
- Vilka olika krav finns?
 - Måste-krav
 - Önska-krav
- Aktörer
 - Användare
 - Externa system



Use Case

Use Case

- Exemplifiering av en funktion i systemet
- Beskriver funktionen genom b.la.
 - Handling som "startar" ett Use Case
 - Förutsättningar
 - Standardflöde
 - Avvikande flöde
 - Aktörer



Klassdiagram

Vad är en klass???

OOA => OOD => OOP

OBJECTS

EVERYWHERE

Objektorientering

Objektorientering är ett sätt att beskriva för en dator hur den ska se på världen, hur data ska struktureras och hur metoder ska skapas och användas. Även deras relationer till varandra beskrivs.

Objektorientering ska försöka likna en människas sätt att se på saker, att man grupperar saker som hör ihop, man skapar objekt.

Ett objekt

- Är av en "typ" => Klass
- Har egenskaper => Attribut
- Kan göra saker => Metoder (funktioner)



Ett objekt == En sak

En klass

En klass är en **mall** för hur ett **objekt** får se ut och bete sig

Klass: Människa

Attribut

- Namn
- Ålder
- Kön

Metoder

- Gå
- Prata
- Sova

Objektet

Attribut

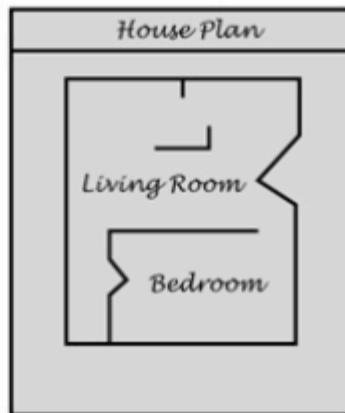
- Namn: Anton Tibblin
- Ålder: 28
- Kön: Man

Metoder

- Gå => ...
- Prata => "Bla bla bla"
- Sova => "Zzzzz"

Class

Blueprint that describes a house



3 objects /
instances /
individuals

Instances of the house described by the blueprint



Klassdiagram

Vad är en klass???

Klassdiagram

- Beskriver hur våra klasser ser ut
- Klassnamn: Namnet på mallen
- Attribut: Egenskaper för alla objekt av klassen
- Metoder: Saker som alla objekt av klassen kan göra



Person

+ Name: string

+ Age: integer

+ Gender: string

+ go()

+ talk()

+ sleep()

```
names = ["Jane", "John", "Elizabeth"]
```

names : list
"Jane", "John", "Elizabeth"
append count insert remove reverse

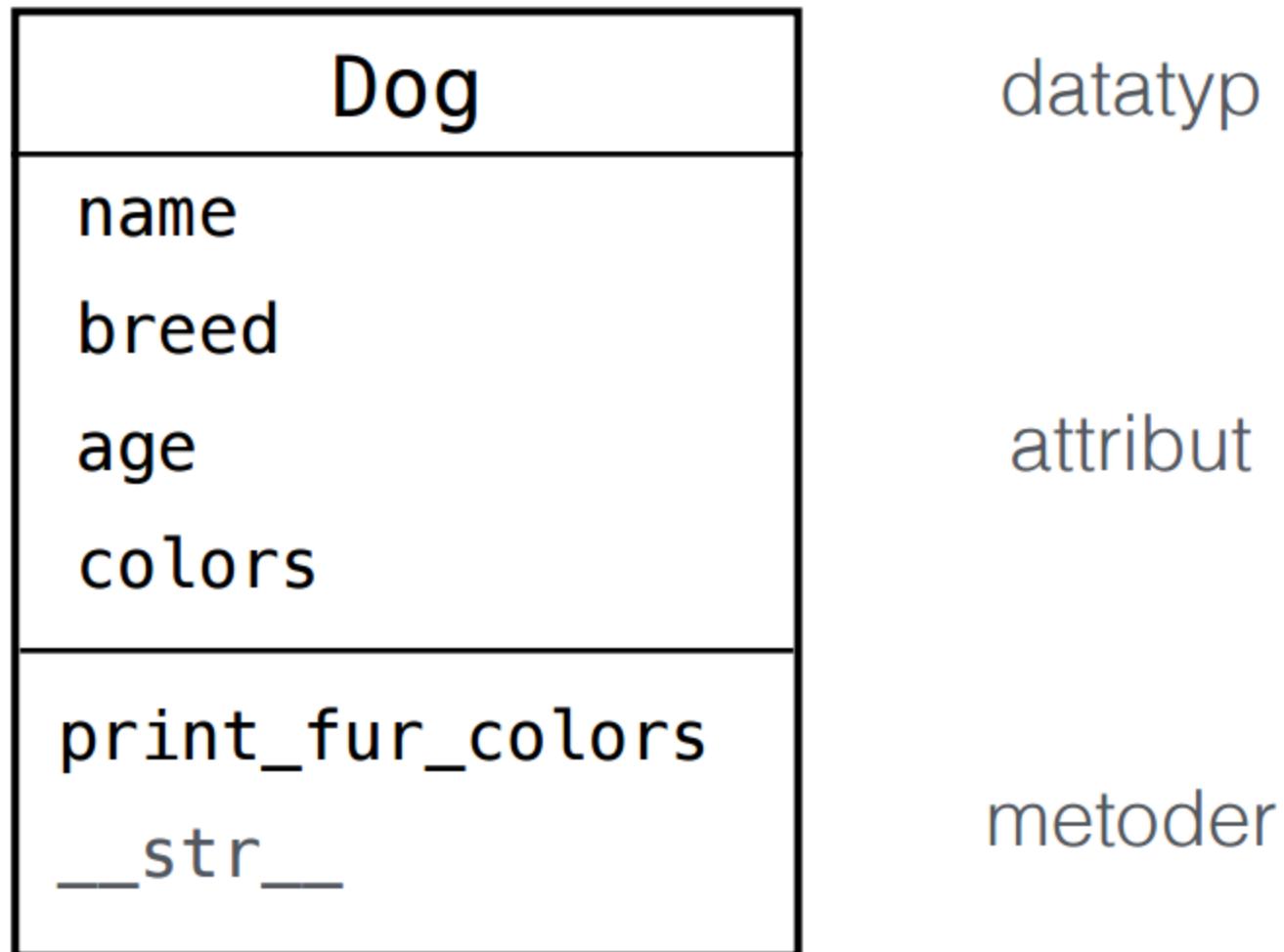
Typ

Data (attribut)

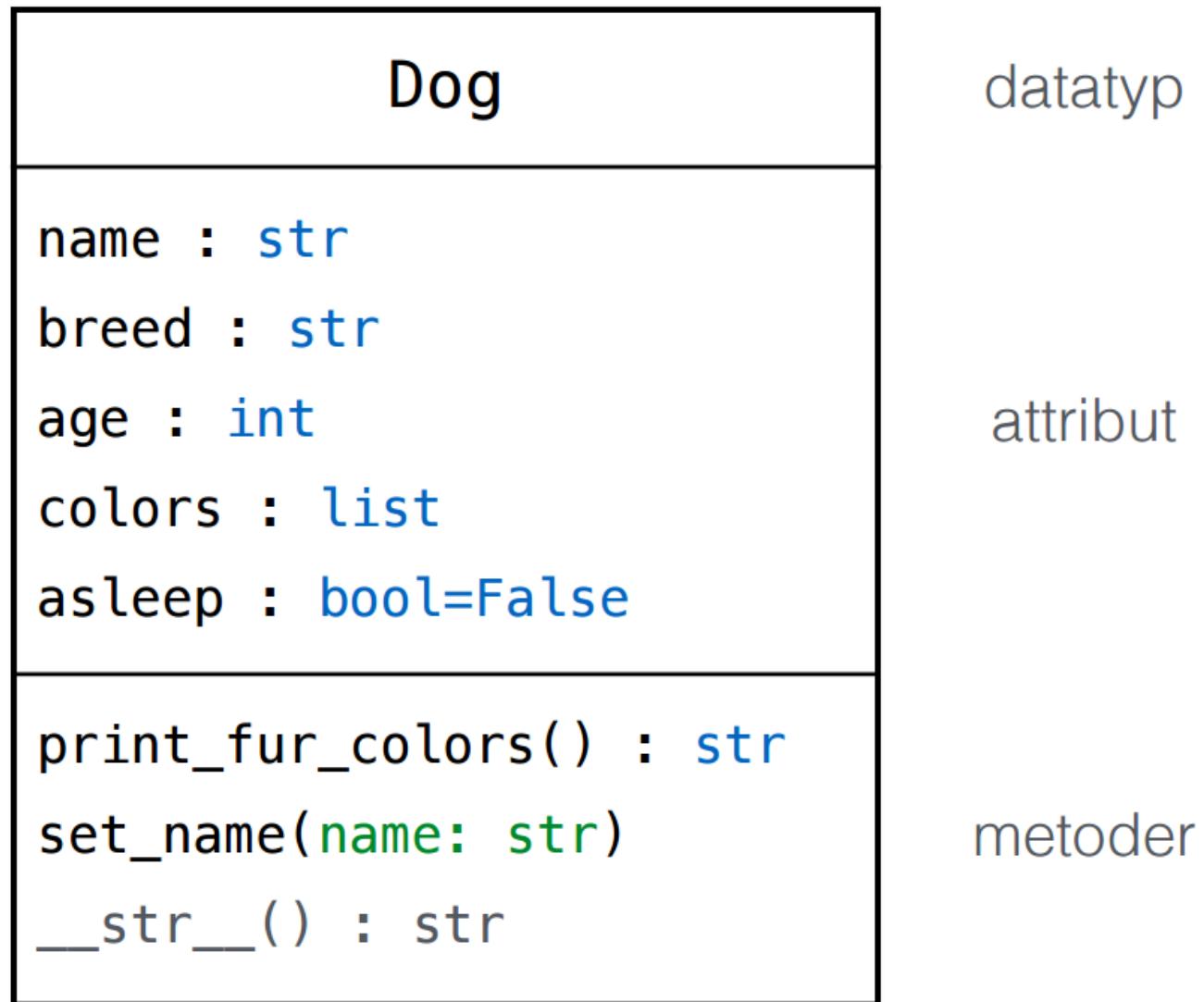
Metoder (funktioner)

Fler exempel

Klassdiagram



Klassdiagram

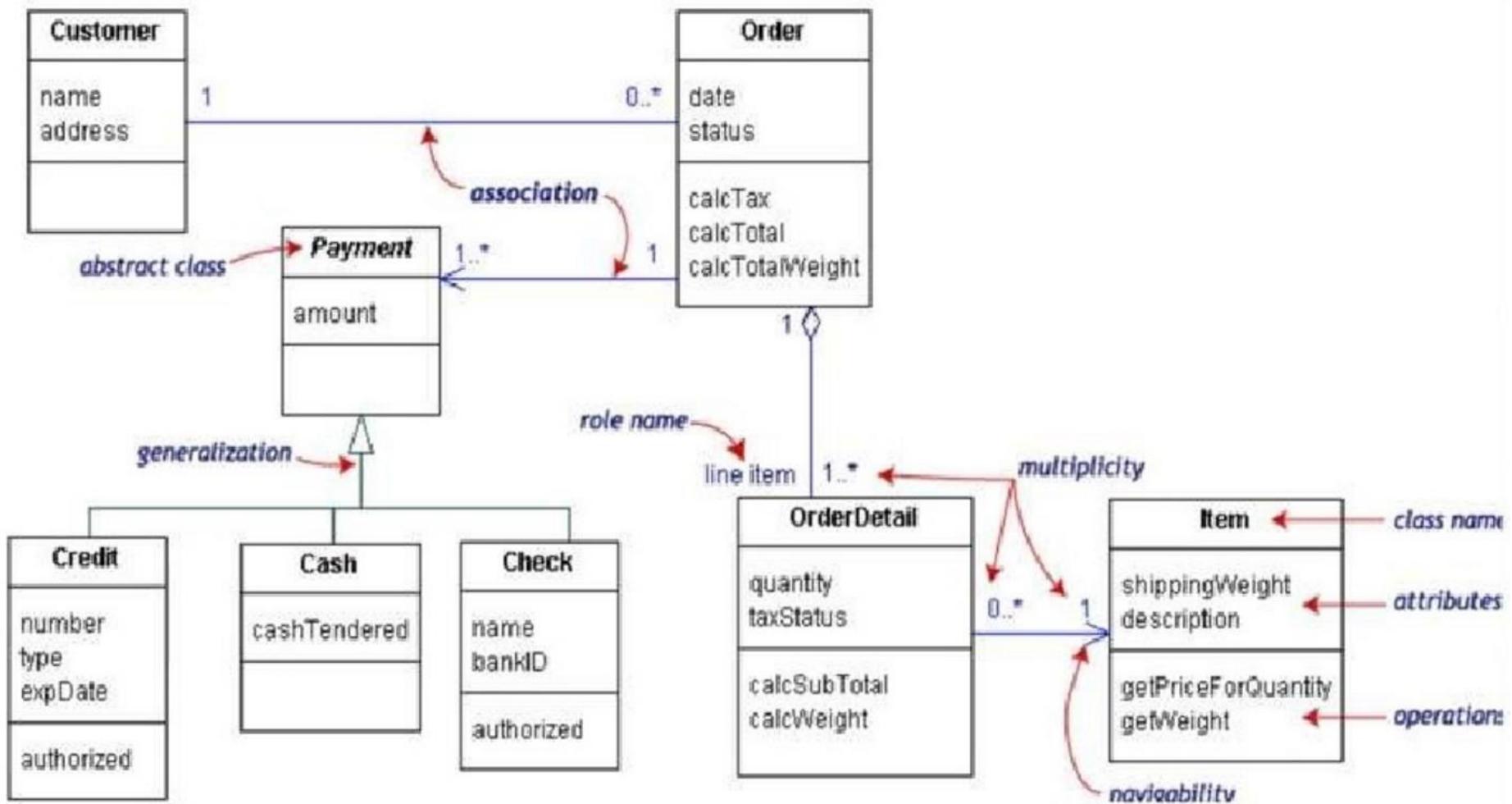


Men detta var ju bara en klass... Hur modellerar man ett helt system?

Arbetsflöde

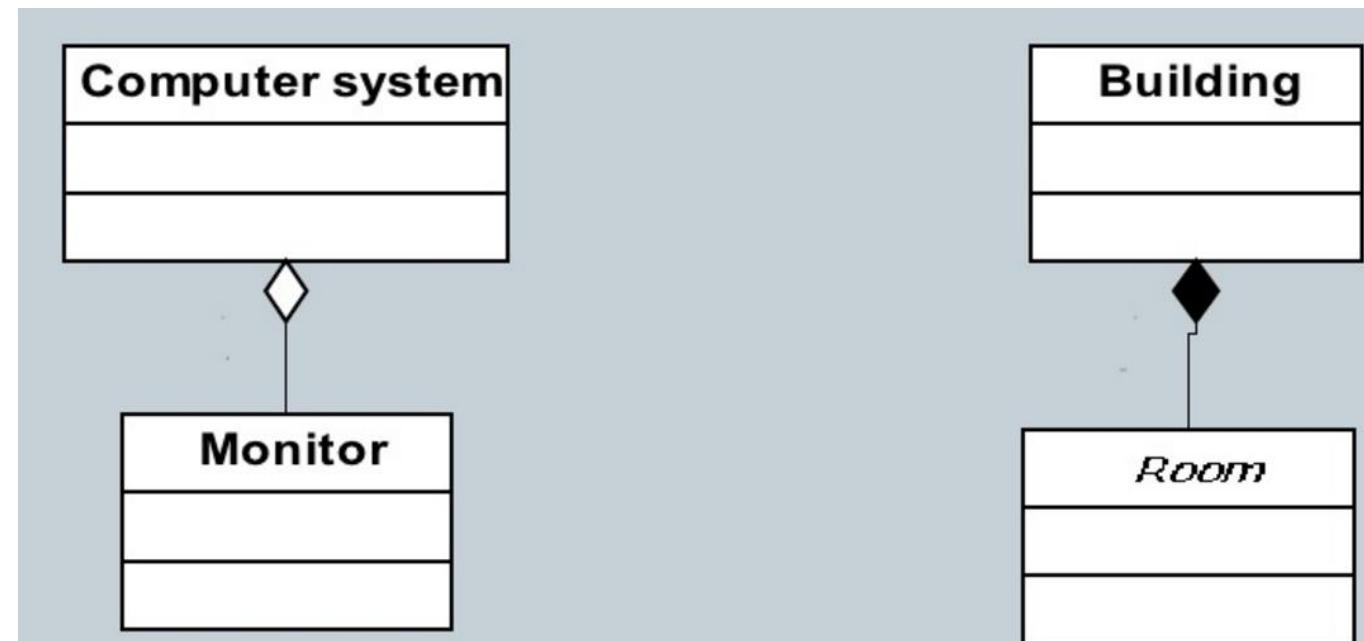
- Identifiera vad som ska modelleras (substantiv)
- Skissa upp ett klassdiagram
- Implementera (stubb)
- Vidareutveckla klassdiagram och implementation

Exempel på klassdiagram



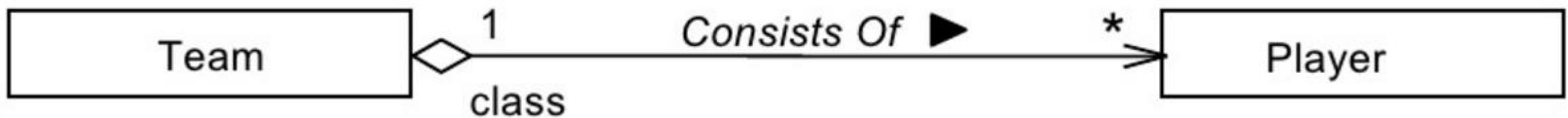
Aggregation och Komposition

- Det finns olika typer av associationer:
 - Aggregation
 - Komposition



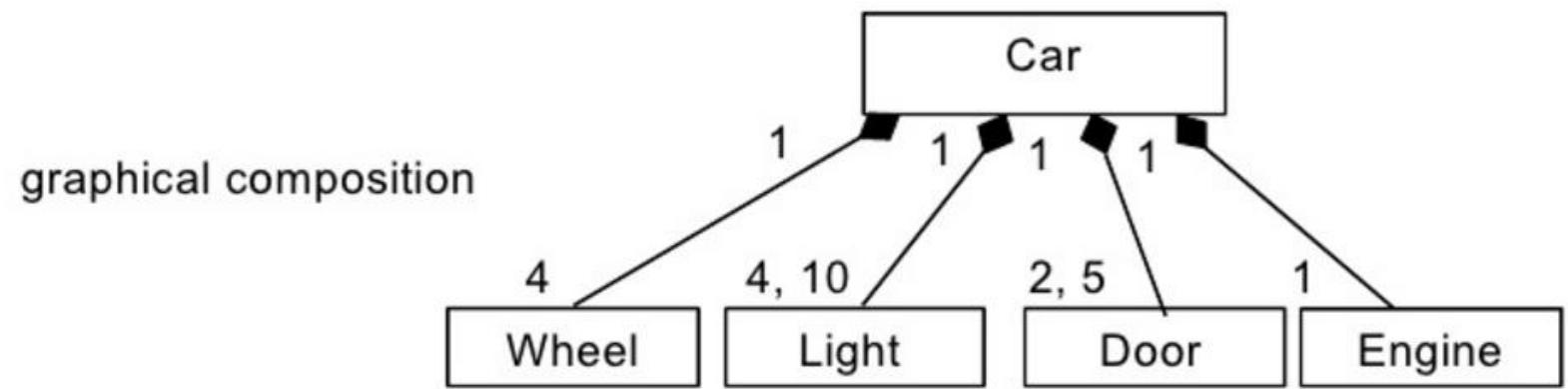
Aggregation

- - "är en del av"-association

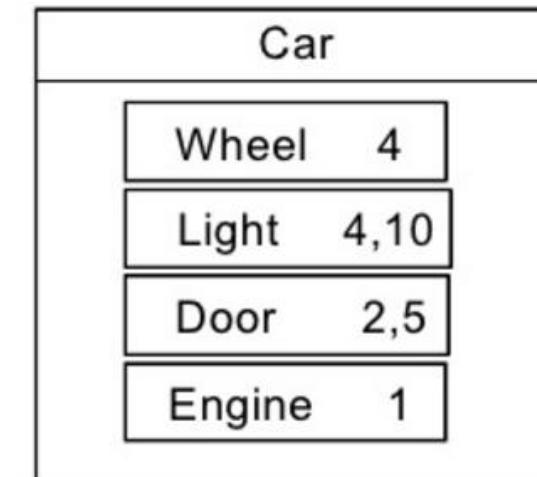


Komposition

- Komposition innebär "består av" (ägande)



nested composition



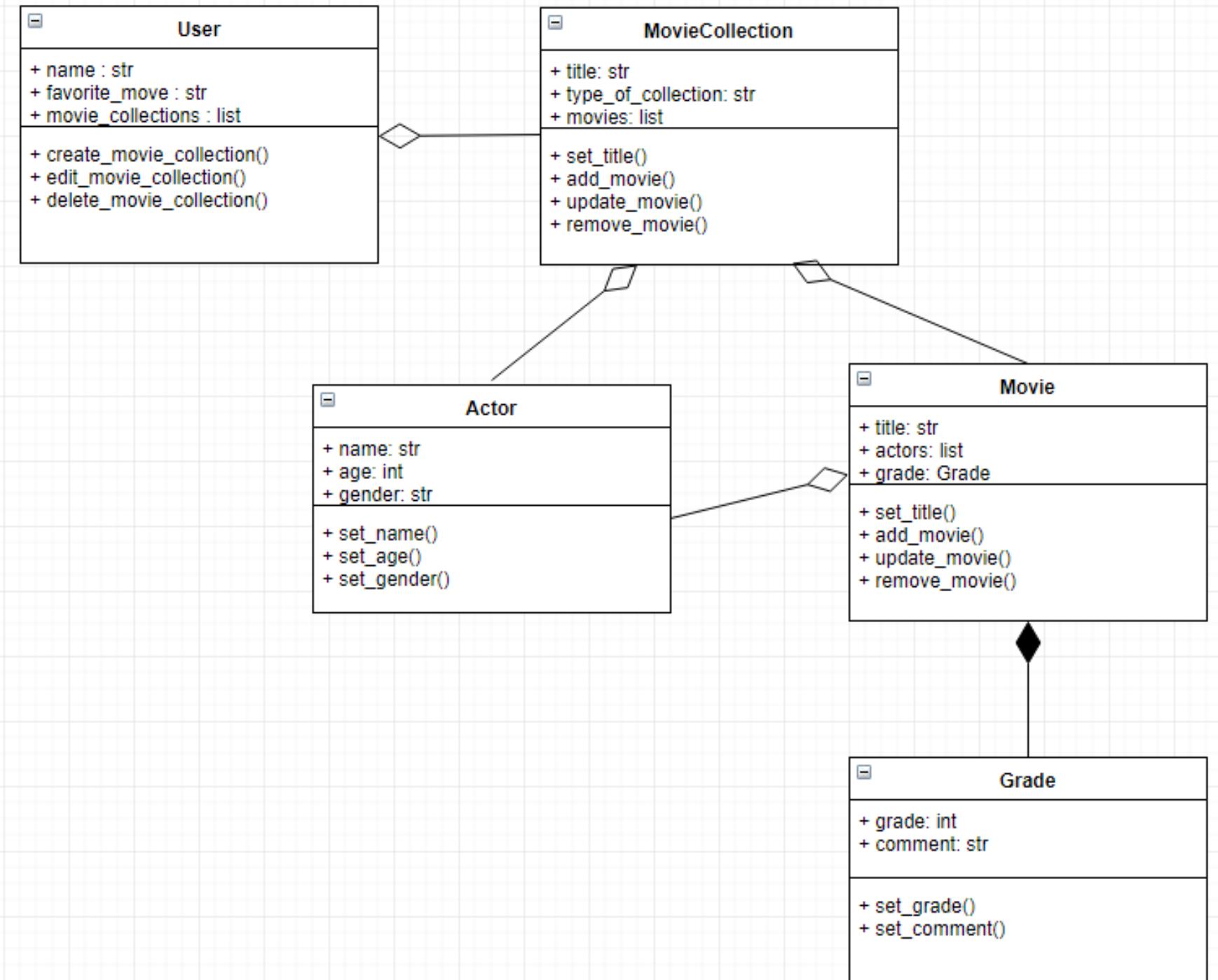
**Med hjälp av detta kan vi
modellera system! =)**

Att modellera en filmsamling

Jag skulle vilja skapa ett system för att hålla reda på mina filmer. Jag vill kunna se varje film och dess skådespelare, men även kunna betygsätta filmerna.

Att modellera en filmsamling

Jag skulle vilja skapa ett **filmsamling** för att hålla reda på mina **filmer**. Jag vill kunna se varje **film** och dess **skådespelare**, men även kunna ge filmerna **betyg**.

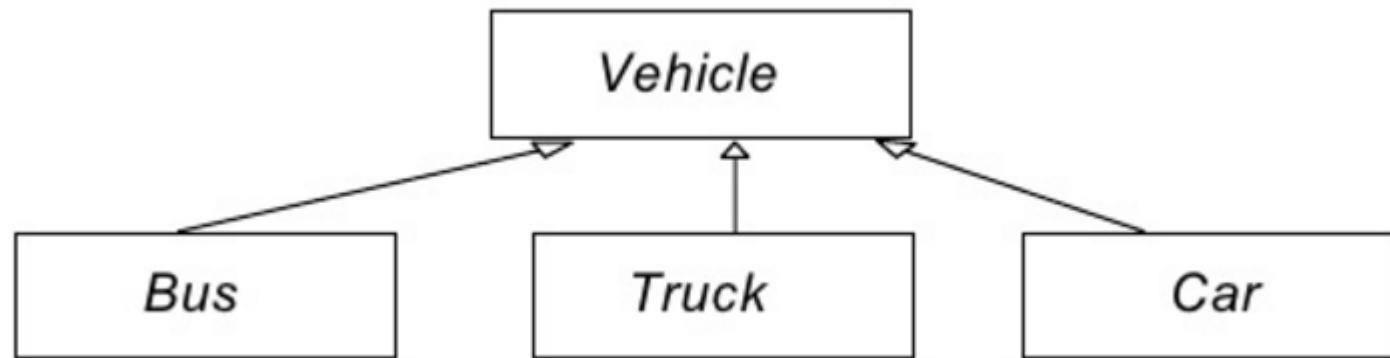


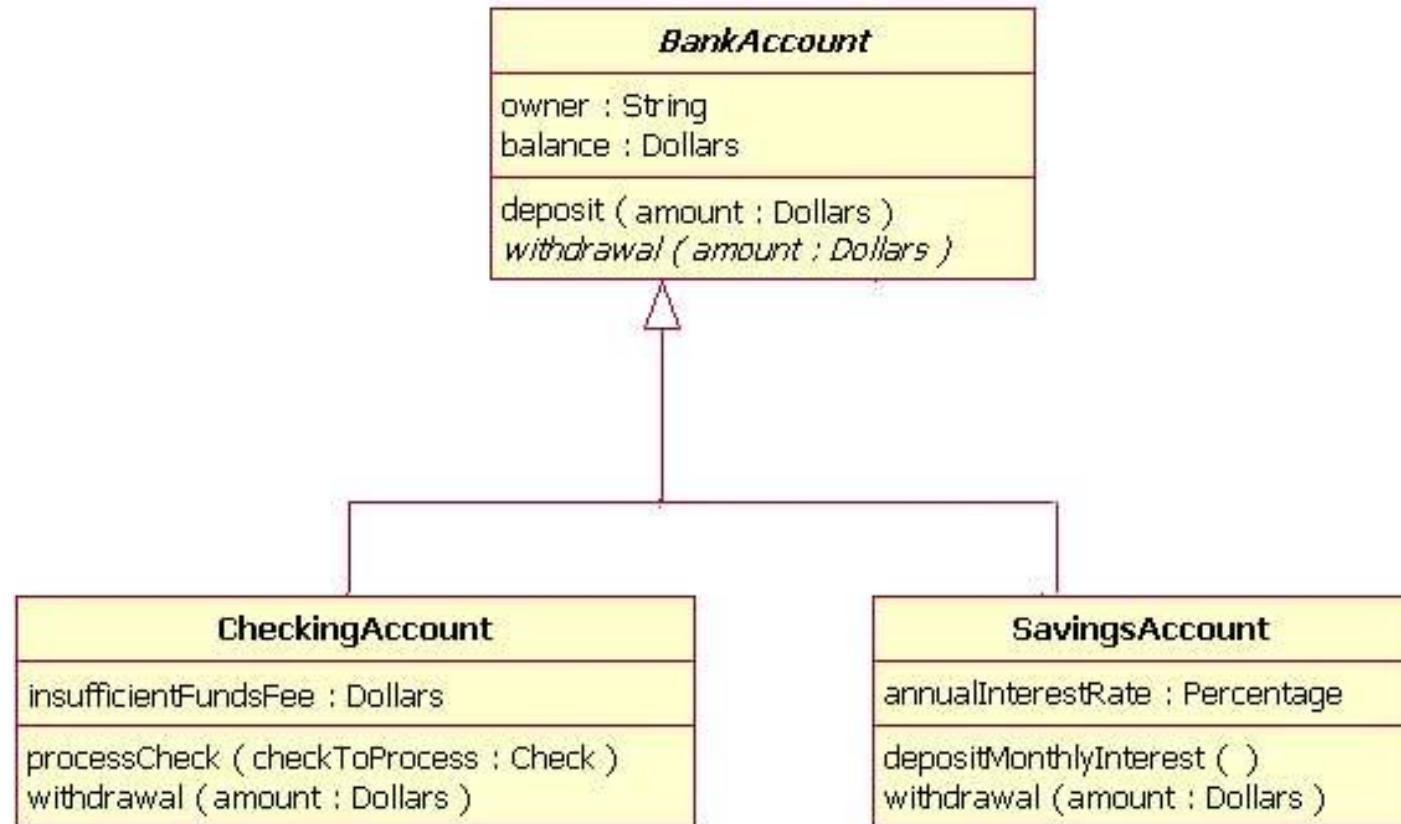
Arv & polymorfism

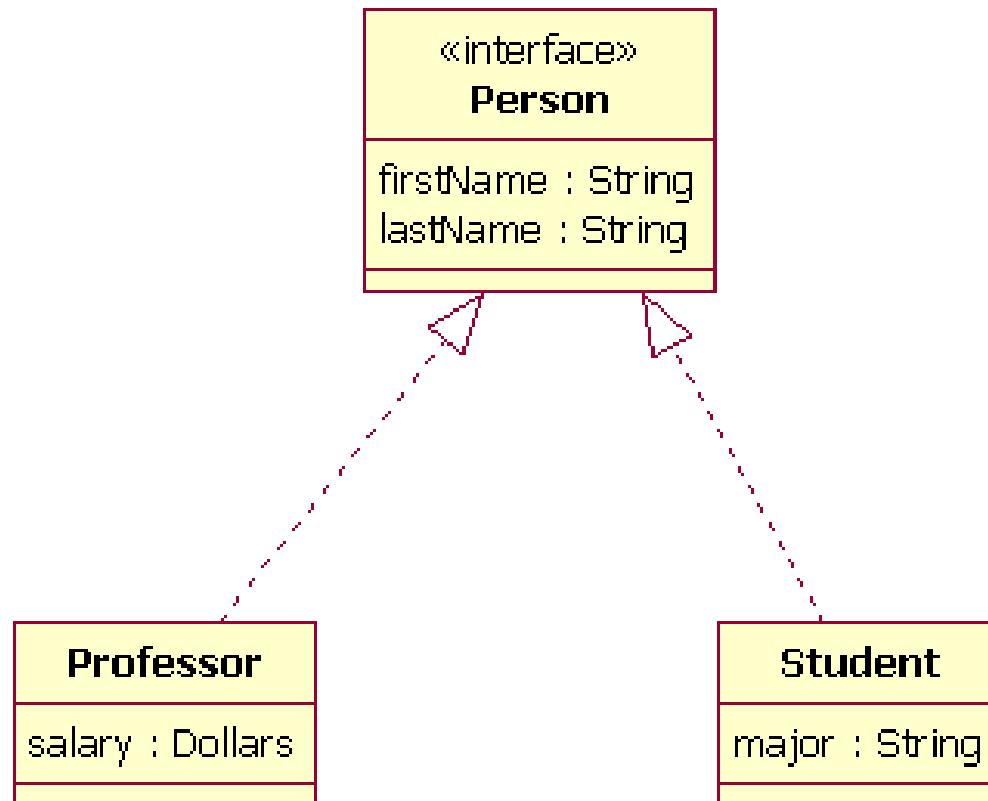
Arv

Arv

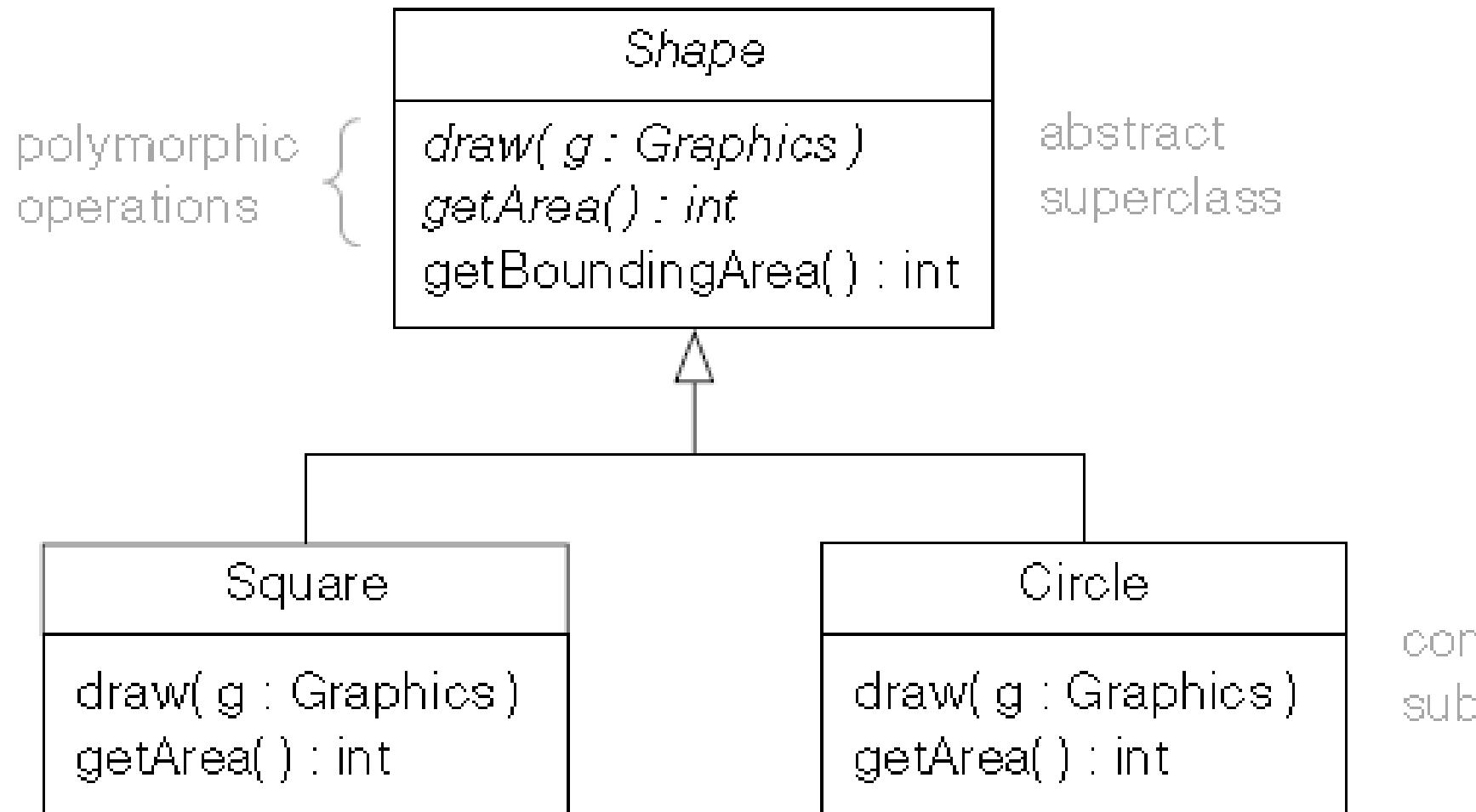
- Arv innebär att man ärver attribut & funktioner från en annan klass

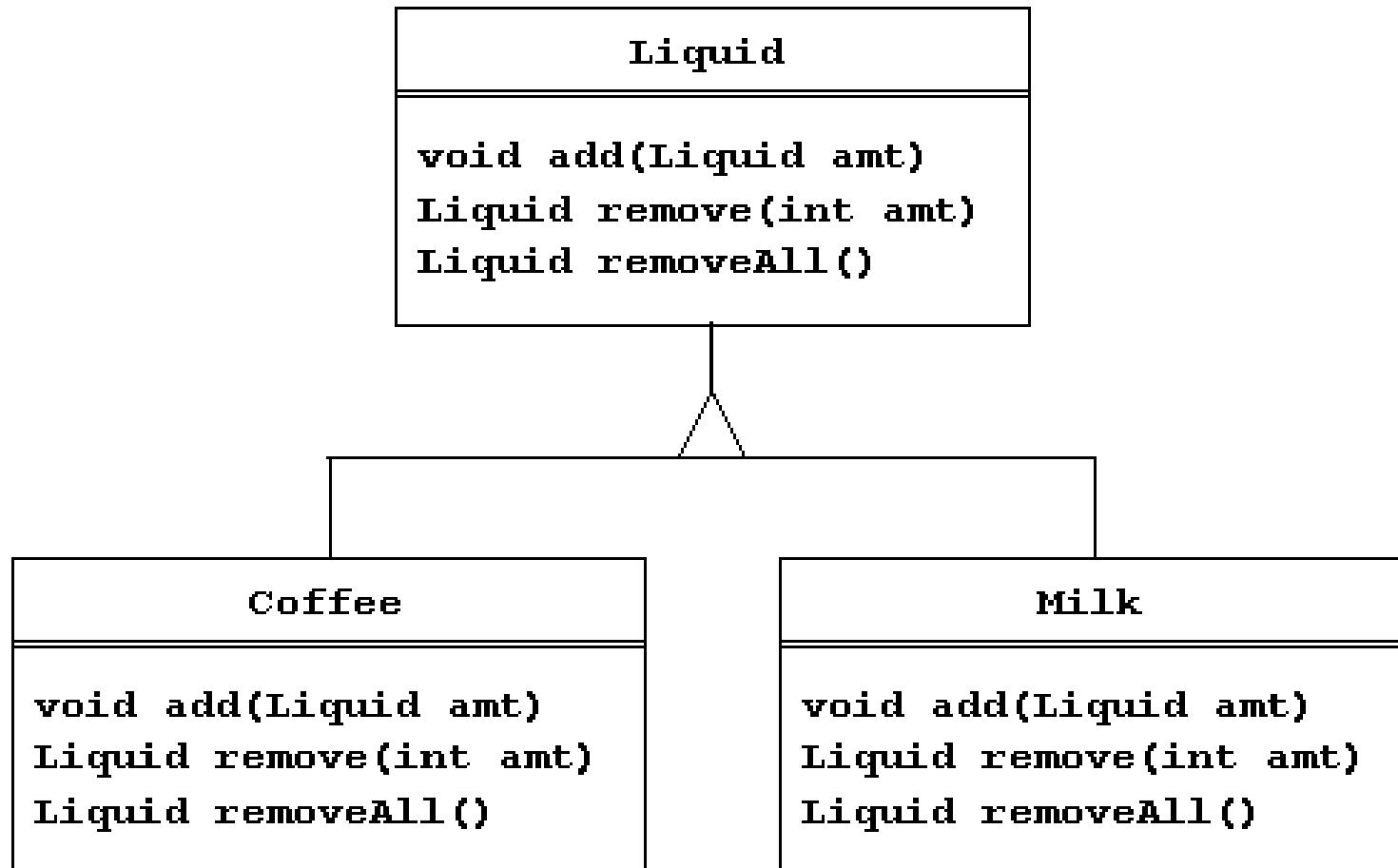


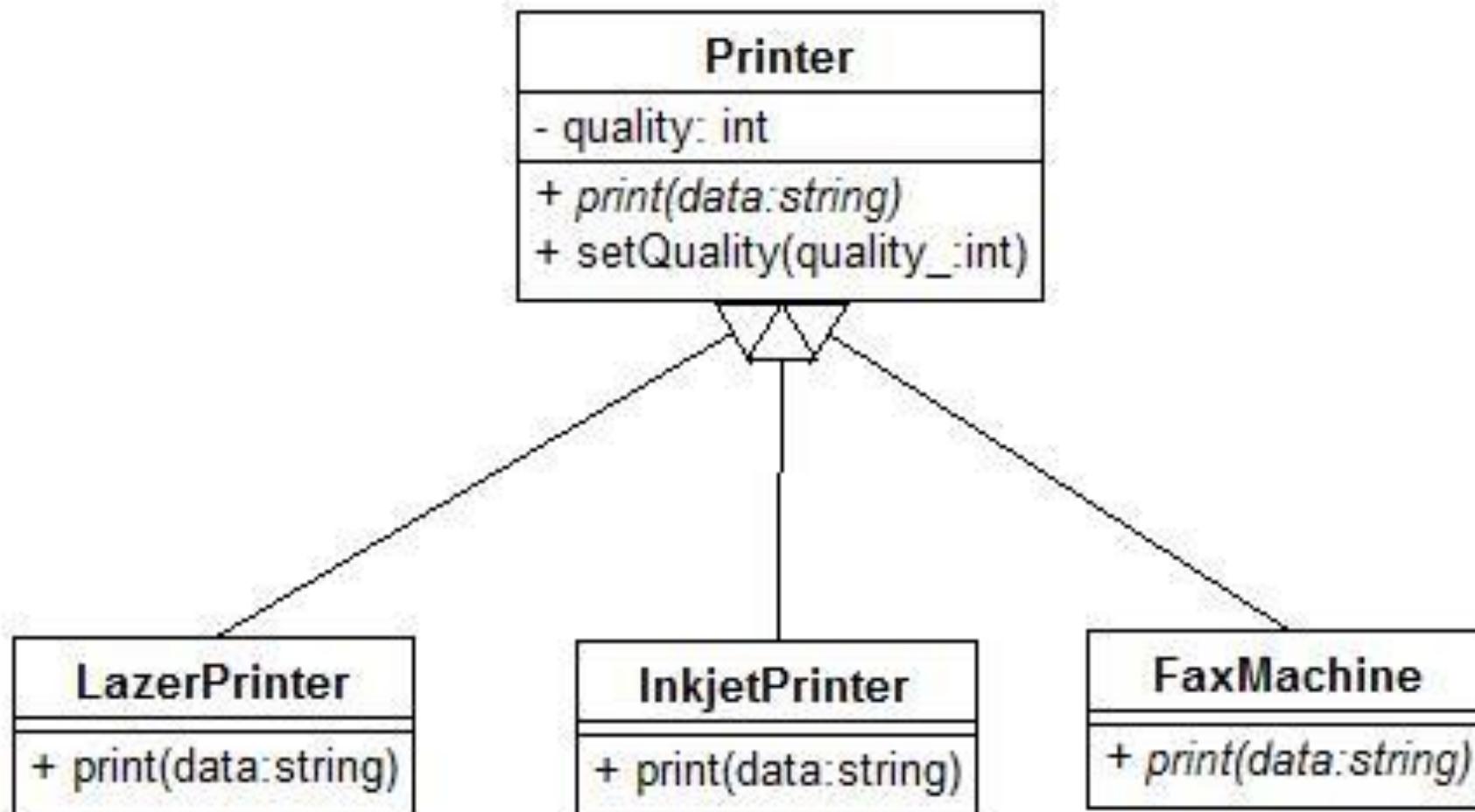




Polymorfism



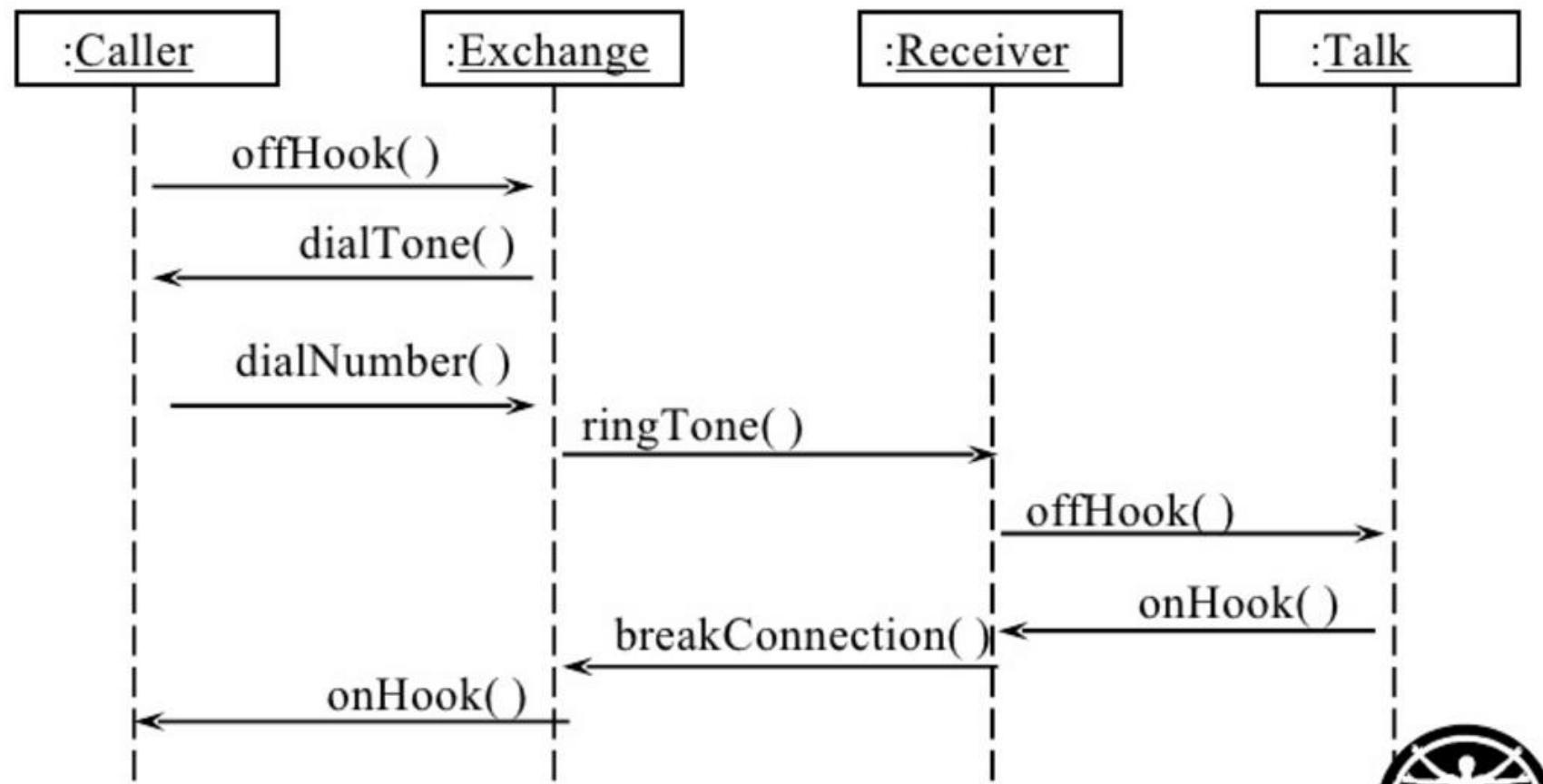




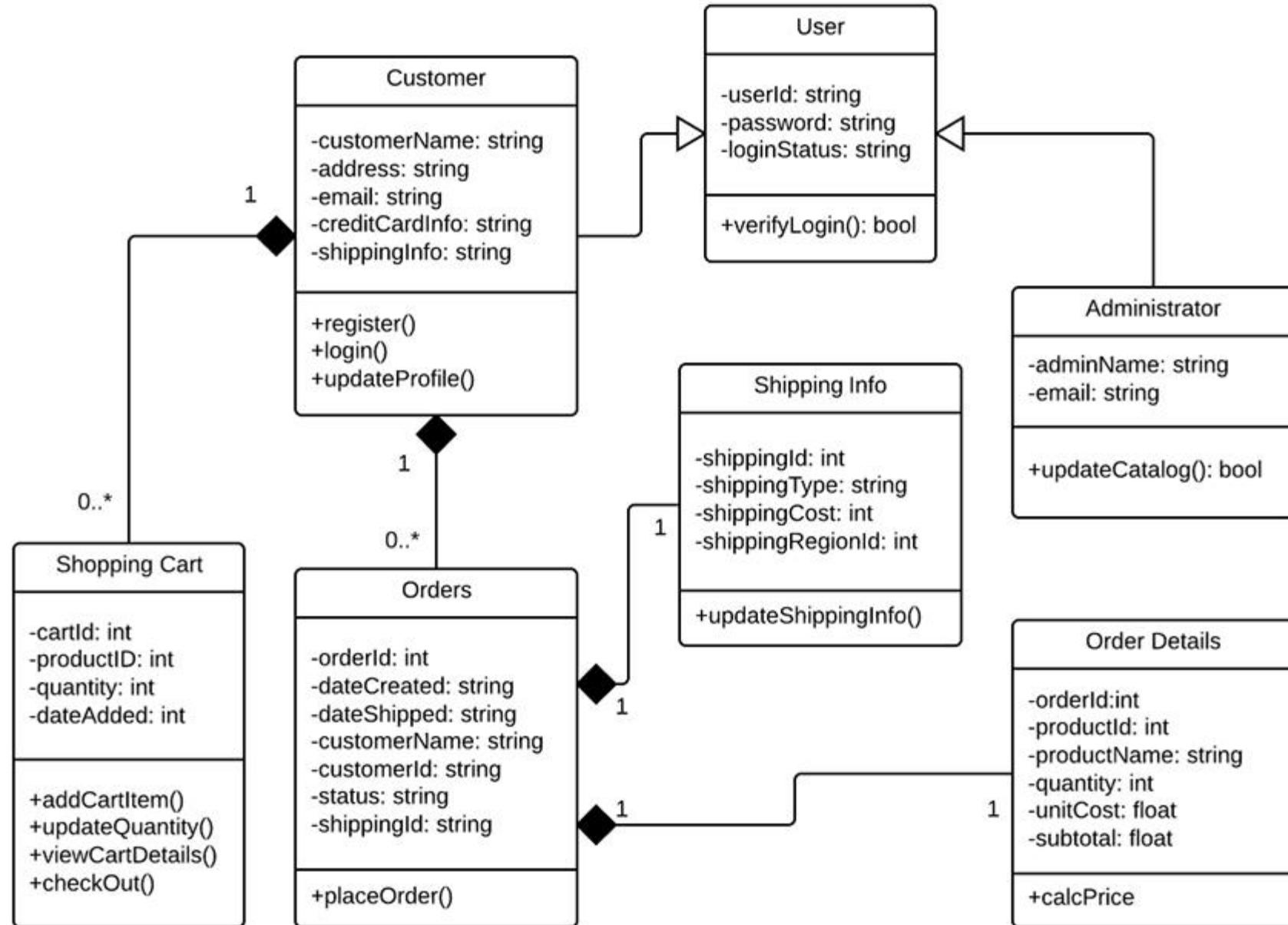
Sekvensdiagram

Sekvensdiagram => Betéende

Telephone Call



Frågor?



Workshop på torsdag

Förberedande inför inlämningsuppgiften