Lektion 2

Objektorienterad Analys och Design

Utbildare: Mahmud Al Hakim

NACKADEMIN

Lektionstillfällets mål och metod

Mål med lektionen – Mer om OOAD

- **Systemdesign/arkitektur**
- **Modellering**: Introduktion till UML
  - Modelleringsverktyg
- Användningsfall (Use Case)
- Användningsfallsdiagram (Use Case Diagram)
- Klassdiagram (Class Diagram)

Lektionens arbetsmetoder

- Föreläsning, diskussioner och övningar
- Laboration 2 (Modellering UML Klassdiagram)
  Redovisning i små grupper under lektionen.

Att läsa: https://www.ida.liu.se/divisions/sas/groups/upp/info/prog/OOP-Nutshell.pdf

Objektorienterad **Analys** och Design

Vad är Analys?

- ”Analys” betecknar i vanligt språkbruk för det mesta en aktivitet i vilken någonting *plockas isär och beskrivs*.
- Vi behöver analysera **kravspecifikationen (kravanalys)**.
- Syftet med analysen är att erhålla en modell av systemet i form av en mer exakt och mer fullständig specifikation och därigenom en bättre förståelse av systemet och dess relation till omgivningen.
- Under analys tar man fram **användningsfall** (use cases) med tillhörande figurer och beskrivningar, liksom olika diagram, som **klassdiagram**.

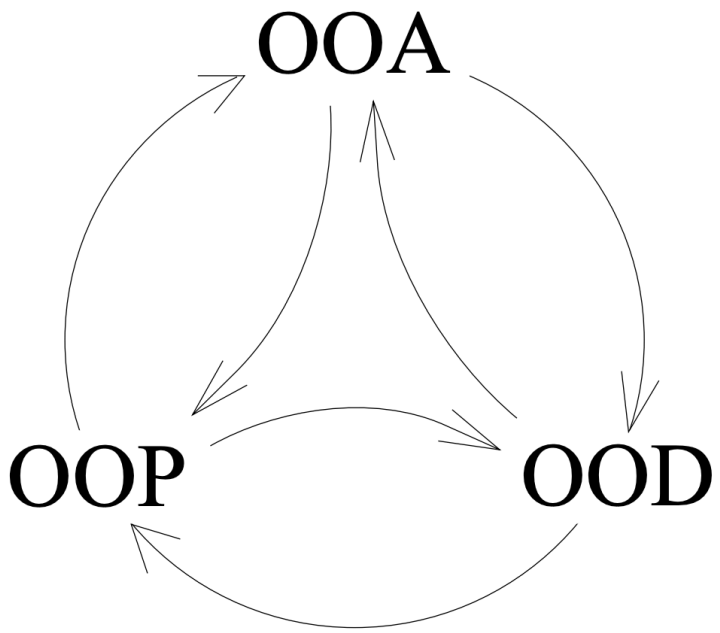
Objektorienterad Analys och Design

Vad är Design?

- En fördel med den objektorienterade modellen är att diagrammen från analysfasen kan användas även i designfasen.
- **Under designfasen förfinas och modifieras klasserna som analysfasen givit upphov till och det tillkommer normalt ytterligare klasser vars behov man inser först under designarbetet.**
- Resultatet av designfasen är detaljerade beskrivningar av systemet, dess gränssnitt mot omvärlden, klasser och objekt, hur data ska lagras permanent, etc.
- Efter designfasen kodas systemet med objektorienterade programmeringsmetoder.

OOAD är en iterativ process

- OOAD är en iterativ process där man vid behov går tillbaka och gör om tidigare steg.
- Det är också troligt att man under arbetets gång har behov av att koda prototyper eller implementera klasser för att prova en viss konstruktion eller ett gränssnitt.
- Detta arbetssätt kallas ***basebollmodellen.***



Objektorienterad design (OOD)

- Efter analysfasen följer designfasen, med dokumentationen från analysfasen som utgångspunkt.

”Gränsen mellan analys och design är ej skarp och ibland beskrivs båda faserna under begreppet **objektorienterad design”**

– Tommy Olsson 2014

Systemdesign/Arkitektur

- **Systemdesign är processen för att definiera arkitekturen.**
- Under systemdesign diskuterar man:
  - Hårdvarukrav (servrar, minne, lagring)
  - Säkerhet (inloggning/rättigheter/backup/återställning)
  - Hur ska systemet distribueras (webbsida/mobilapp/programvara).
  - Grafisk användargränssnitts utformning.
  - Hur permanent datalagring ska göras.
  - Databaser och ER-diagram.
  - Hur kommunikation mellan olika delar i systemet ska ske, etc.

Modellering

- En modell över ett specifikt problemområde kan hjälpa oss designa bra objekt.
- En modell hjälper oss diskutera och resonera kring det riktiga problemet.
- Vi kan känna igen objekt som är otydliga/otillräckliga.

UML – Unified Modeling Language

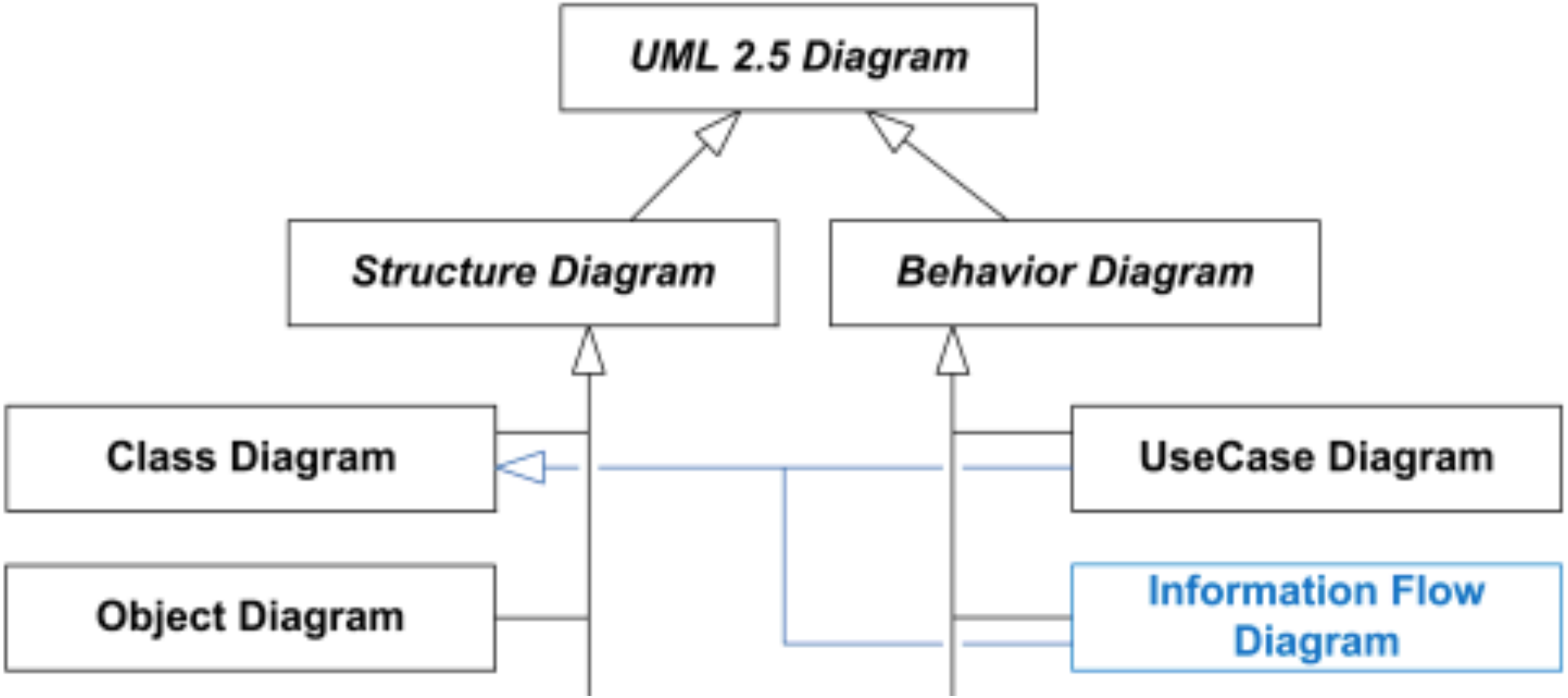
- UML är ett visuellt modelleringsspråk för att specificera, konstruera och dokumentera artefakter i ett system.
- UML är ett grafiskt notationsspråk för objektorienterad design.
- UML kan bl.a. användas för att modellera:
  - Användningsfall (use cases)
  - Klasser och objekt
  - Interaktionen mellan objekt

Varför UML?

- Människor har lättare att förstå bilder än abstrakt programkod.
- UML är ett gemensamt språk för IT-arkitekter och utvecklare.
- Gemensamt språk för att diskutera, dokumentera och förstå affärsprocess, systemkontext och vilka artefakter som ingår.
- Bra källa med många exempel: https://www.uml-diagrams.org/

UML-diagram

- UML-diagram kan delas upp i två kategorier
  1. **Structure Diagram (Struktur diagram)**
     Statisk representation av strukturen i ett system.
     Struktur diagram visar en överblick av ett system, hur det är levererat och vilka komponenter det består av.
  2. **Behavior Diagram (Beteende diagram)**
     Dynamiska diagram för att visualisera användningsfall, aktivitet, flöden och processer.
     Beteende diagram visar vad som händer i systemet.

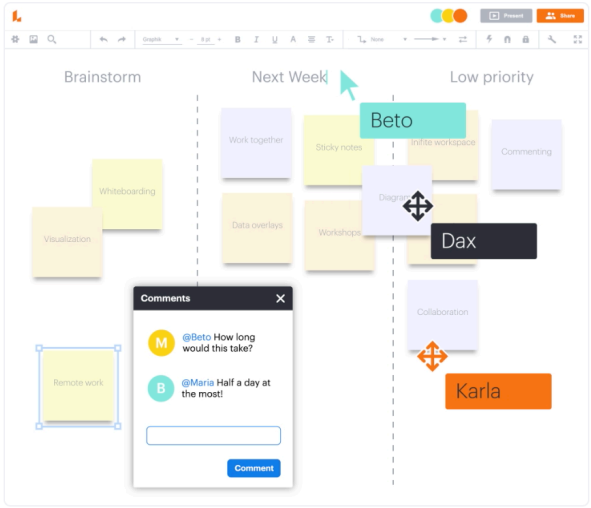


OBS! Det finns flera typer här:
https://www.uml-diagrams.org/uml-25-diagrams.html

Modelleringsverktyg
https://www.lucidchart.com

Lucidchart Enterprise Solutions Resources Pricing Sign up free Log in

The visual workspace for remote teams

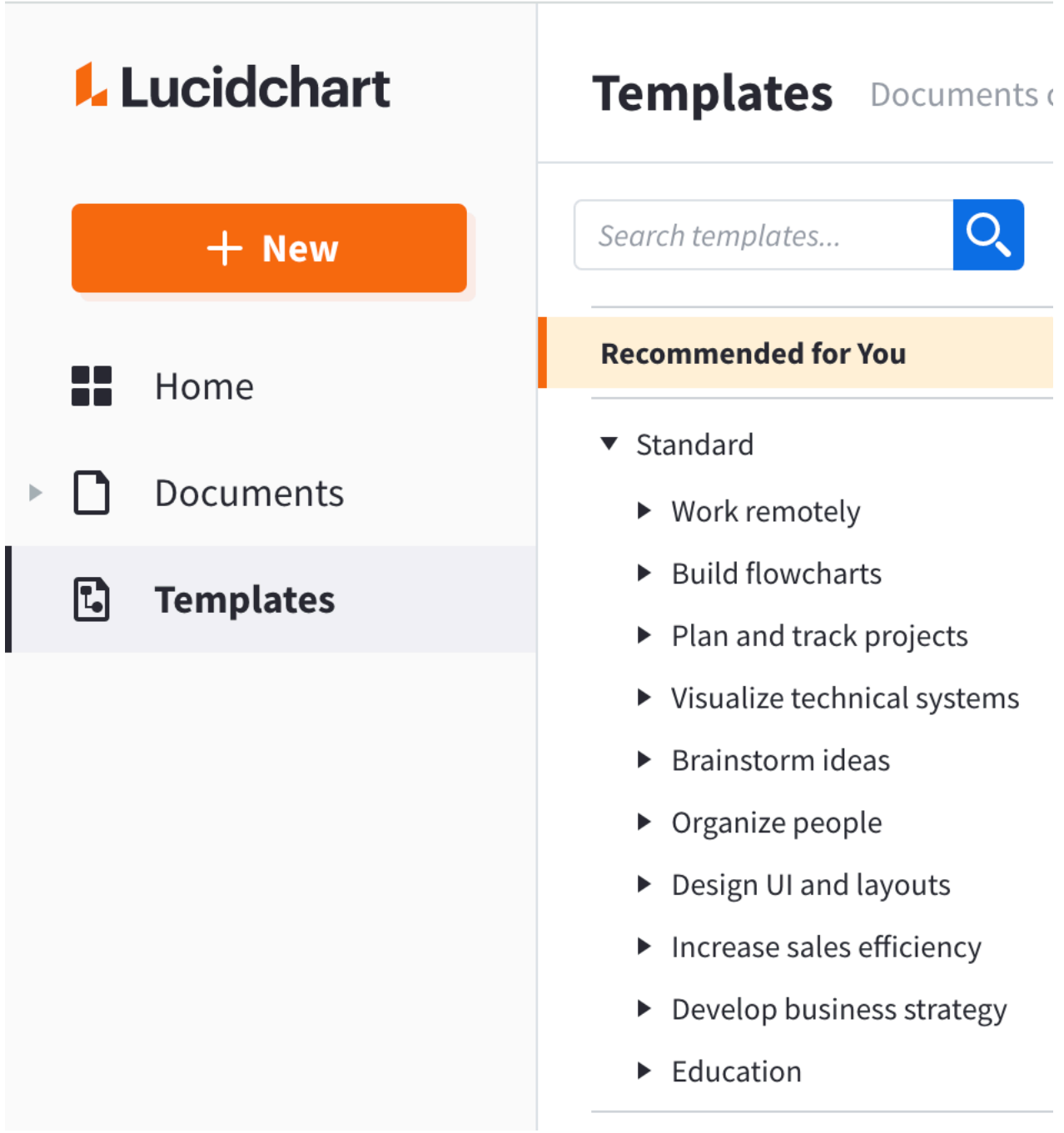


Accomplish more by collaborating in real time. Accelerate understanding and drive innovation with powerful diagramming, whiteboarding, and data visualization. Sign up for a free trial today.

Sign up free

Övning

- Skapa ett gratiskonto på www.lucidchart.com
- Leta efter några UML-mallar (Templates)
- Vilka diagramtyper hittar du?



Vad är ett användningsfall?

- Ett användningsfall (**use case**) är en interaktion mellan en användare och systemet.
- Om man arbetar med ordbehandlingsprogram kan ett användningsfall vara att ”**ändra ett textstycke till kursiv stil**” eller ”**skapa en innehållsförteckning**”.
- Ett användningsfall
  - Utgör en funktion som är synlig för användaren.
  - Uppnår ett distinkt mål för användaren.
  - Kan vara stort eller litet.

Användningsfallsmodellering

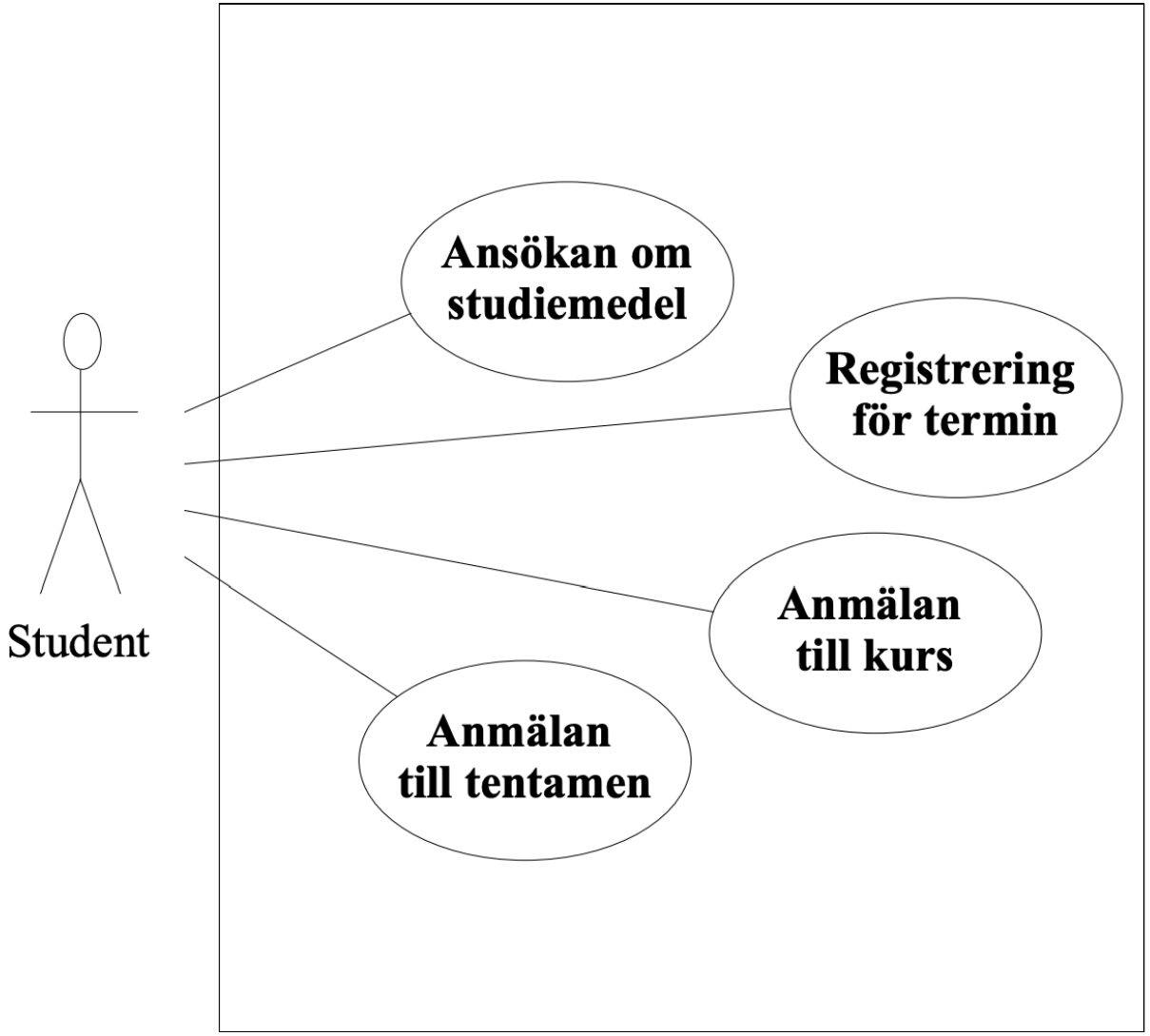
- I enkla fall kan man hitta användningsfall genom att tala med användare och diskutera vad de vill kunna göra med systemet.
- Varje distinkt sak en användare vill göra ges ett namn.
  Man skriver också en kort textuell beskrivning på några få meningar.
  Med en sådan kortfattad beskrivning av ett användningsfall som utgångspunkt utvecklar man sedan iterativt användningsfallet i detalj.
- Resultatet av en sådan användningsfallsmodellering är ett antal aktörer och en katalog med användningsfall.

Aktörer

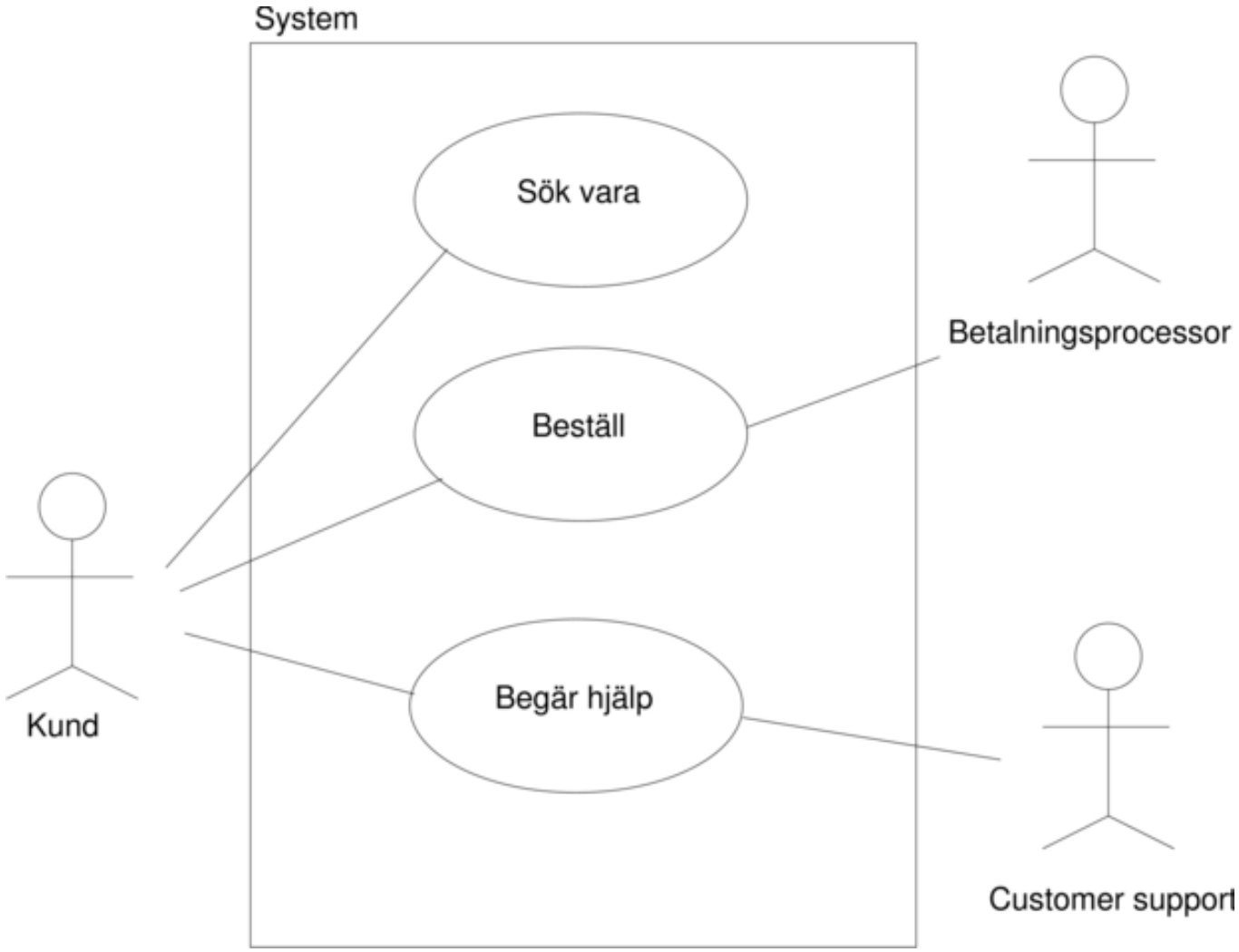
- En aktör är en roll som en användare har i förhållande till systemet.
- Aktörer är externa i förhållande till systemet och utbyter information med systemet.
- En aktör kan utföra olika användningsfall och ett användningsfall kan utföras av flera olika aktörer.
- Ibland kan det vara givande att först hitta aktörerna i ett system och utifrån dessa komma fram till olika användningsfall.

Användningsfallsdiagram (Use Case Diagram)

- Ett användningsfall utgör en viss funktionalitet i systemet som en aktör använder.
- Det initieras av aktören och består av en sekvens av händelser i systemet.
- Användningsfall visualiseras i form av användningsfallsdiagram (use case diagram).



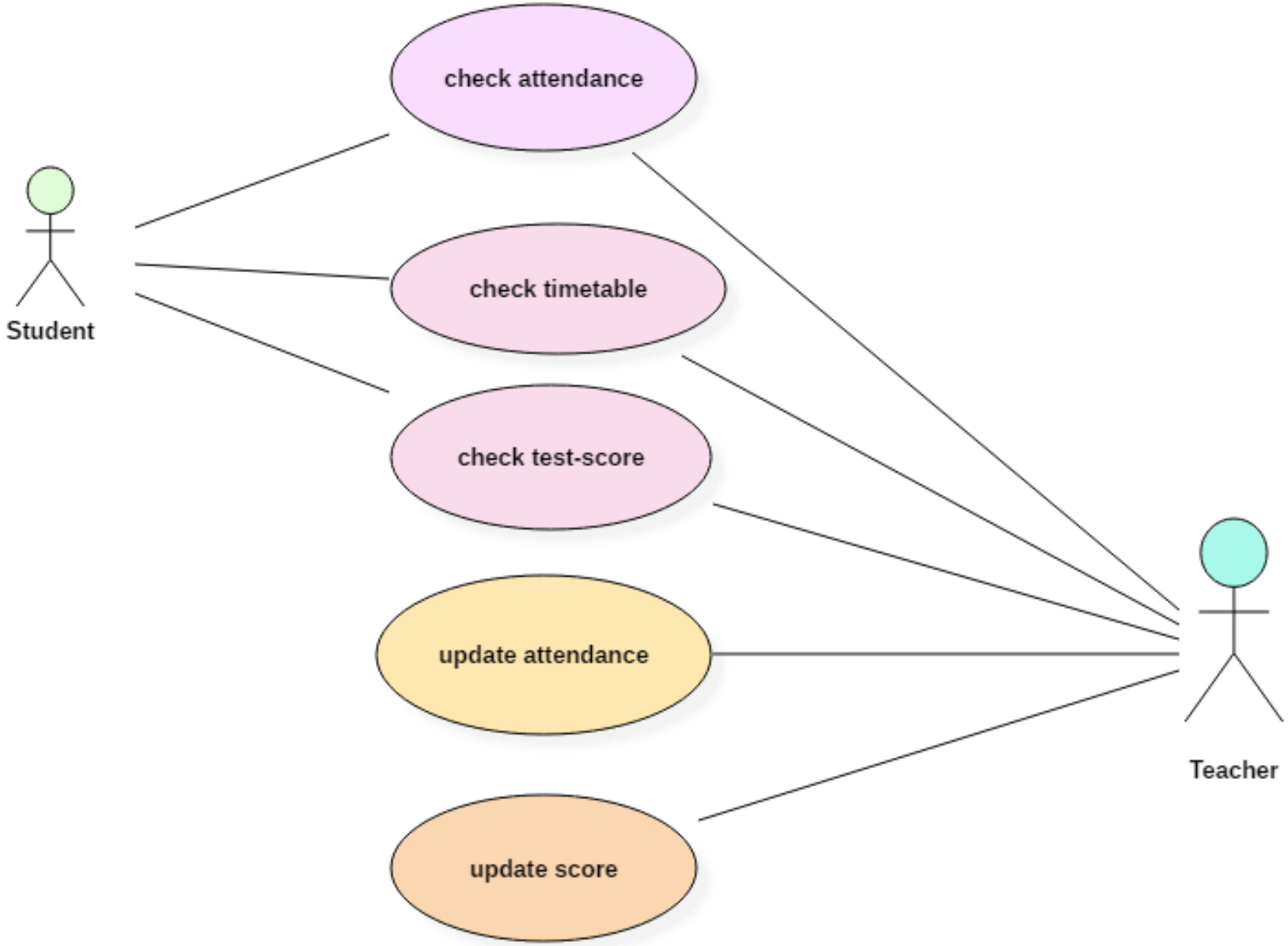
Use Case Diagram – Exempel 2



https://www.it.uu.se/edu/course/homepage/oopjava/vt09/notes/f08-uml.html

Use Case Diagram

Exempel 3



https://www.guru99.com/use-case-diagrams-example.html

Övning i små grupper

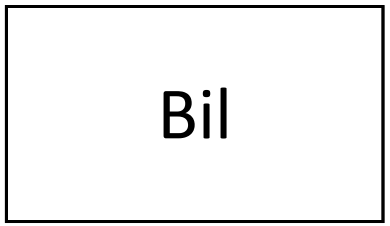
- **Rita ett användningsfallsdiagram (Use Case Diagram) i LucidChart**
- Följande är kända.
  - **Aktörer**
    - Kund
    - Lagerarbetare
    - Transportör
  - **Användningsfall**
    - Gör en beställning
    - Plocka varor
    - Paketera varor
    - Lasta varor
    - Transportera varor

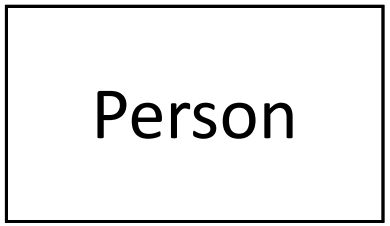
Att rita klasser och objekt

- Under OOAD är det vanligt att kommunicera om objekt och klasser i ett program.
- **Klassdiagram** representerar den objektorienterade vyn av ett system. Det visar upp systemets klasser, deras attribut, metoder och relationen mellan klasserna.
- **Objektdiagram** visar ett antal objekt och dess länkar till varandra.
  Ett objektdiagram exemplifierar alltså ett klassdiagram i ett visst ögonblick.

Klassdiagram (Class Diagram)

- En klass i ett klassdiagram representeras med en rektangel som är indelad i tre fack:
  1. I den översta rutan står klassnamnet, centrerat med stor första bokstav.
  2. Mittenfacket innehåller klassens attribut.
  3. Nedersta facket innehåller klassens metoder.
- OBS!
  Del 2 och 3 är inte obligatoriska!
  Ett klassdiagram kan vara en liten ruta med enbart klassnamnet.



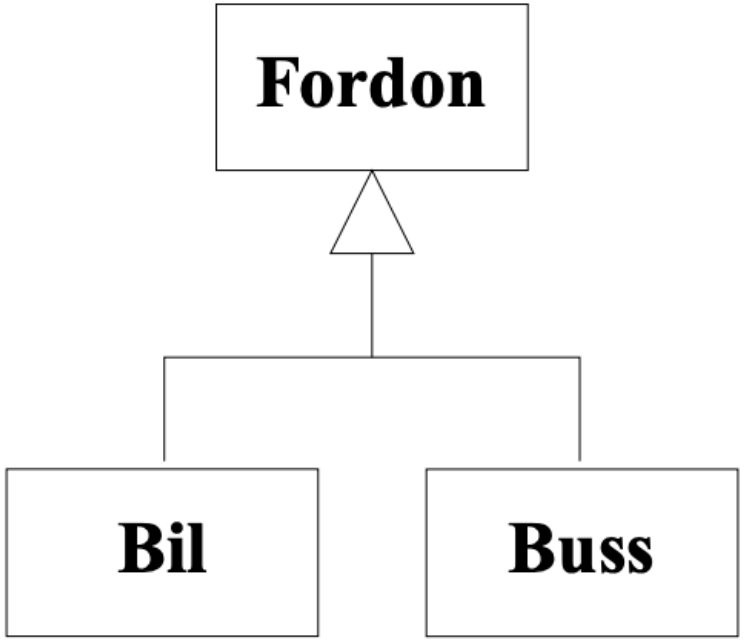


Relationer

- Ett klassdiagram beskriver olika typer av objekt i ett system och relationer mellan dessa.
- I princip kan man särskilja två slags relationer
  1. **Arv** innebär att en typ är en specialisering av en annan typ, till exempel att en bil är ett specialiserat fordon, vilket också exempelvis en buss är.
  2. **Association** representerar någon form av relation mellan instanser av klasser, till exempel att en person kan äga en bil eller att en bil har en motor.

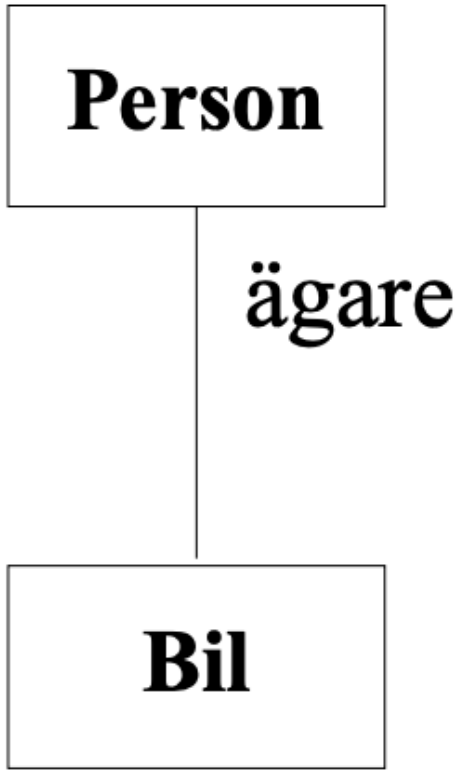
Arv – Exempel

- Arv ritas med en triangelformad symbol vid superklassen och en enkel linje till subklassen.
- Arv är en **är-relation** (en bil **är** fordon).



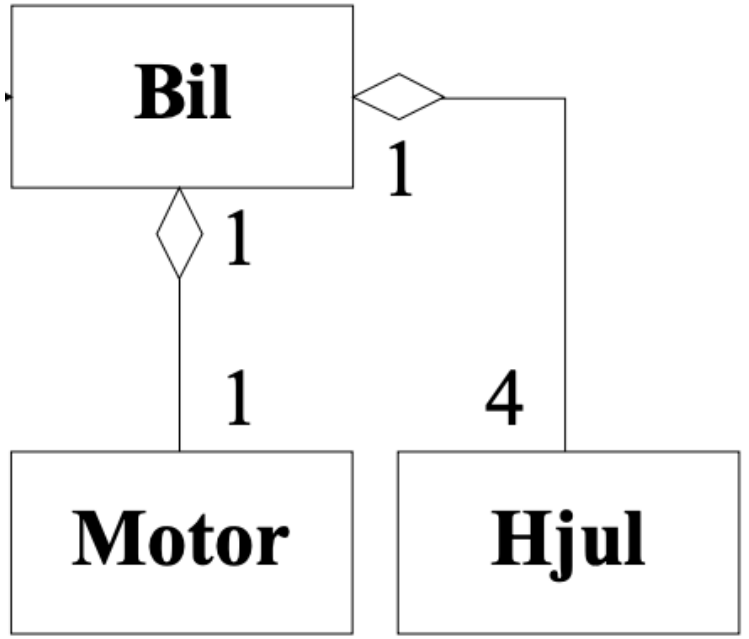
Association – Exempel

- En association ritas som en enkel linje mellan relaterade klasser.
- Association, ibland kallad **känner-till-relation**, kan användas för att beskriva alla former av relationer som ej är arv.



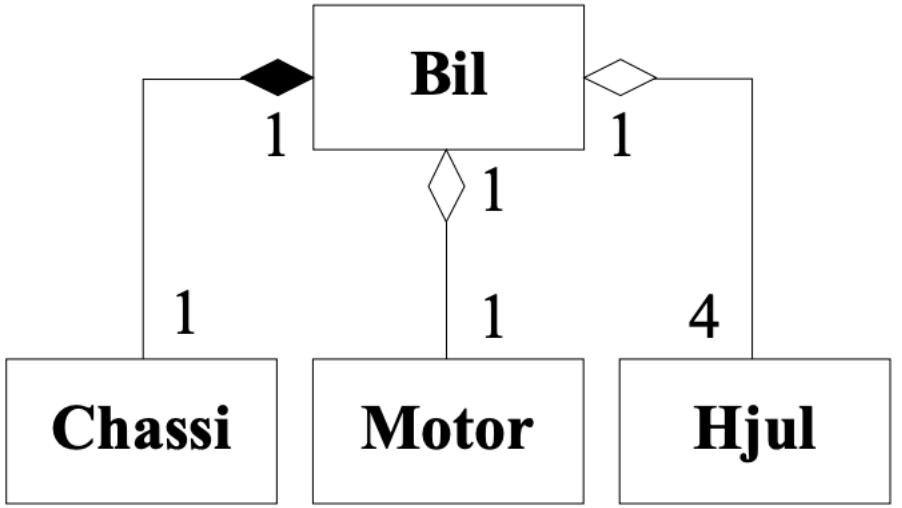
Aggregering (Aggregation)

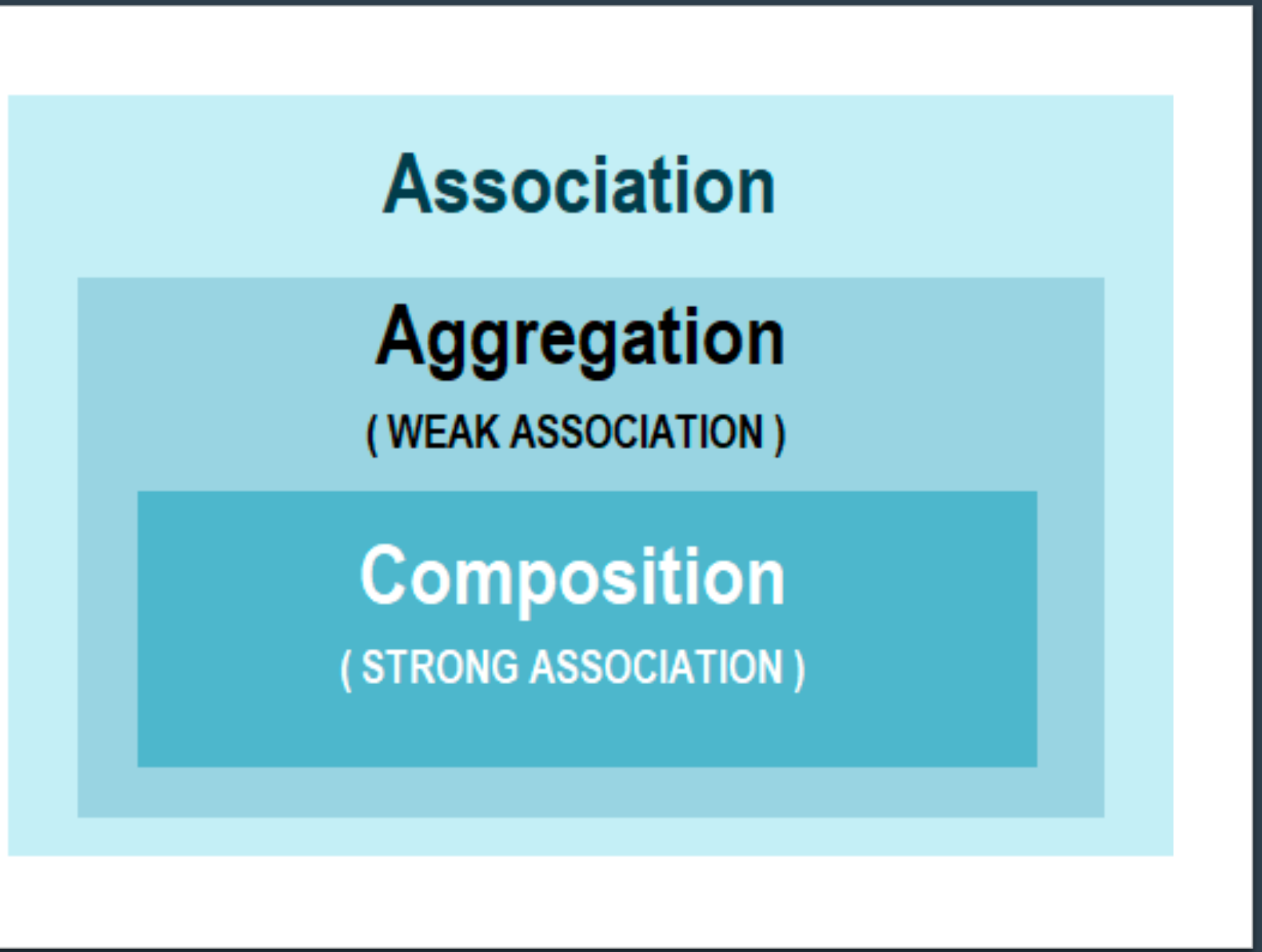
- En klass byggs ofta upp med hjälp av ett antal andra klasser.
- Om en klass består av en/flera andra klasser är detta ett **aggregatförhållande**.
- Aggregering kallas också **har-relation** eller **del-av-relation**.
- Ett exempel på aggregering är att en bil kan bestå av (har) fyra hjul och en motor.
- Aggregering i UML betecknas med en ofylld romb.



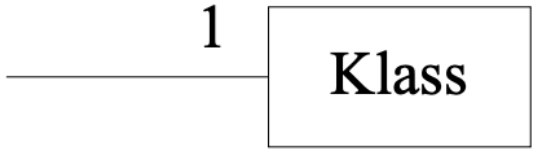
Komposition (Composition)

- Komposition är en starkare form av aggregering, där ett delobjekt endast kan ingå i en helhet och att delobjektet normalt endast existerar medan denna helhet existerar.
- Man kan anse att chassit är starkt förknippat med bilens existens; då bilen skrotas försvinner chassit. Motor och hjul, däremot, kan bytas ut.
- Komposition i UML betecknas med en fylld romb.

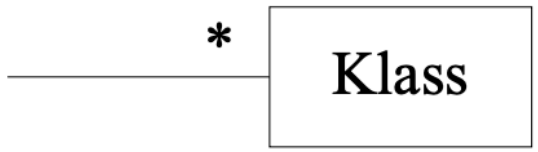



Association
Aggregation
(WEAK ASSOCIATION)
Composition
(STRONG ASSOCIATION)

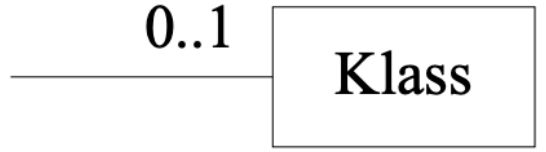
Multiplicitet (beteckna antal)



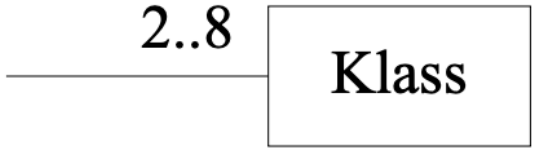
exakt ett



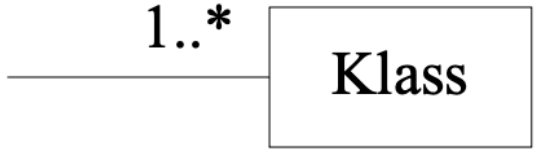
godtyckligt många, även inget



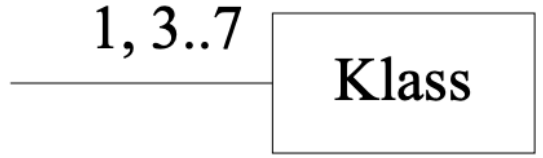
inget eller ett



mellan två och åtta



ett eller flera



ett eller mellan 3 och 7

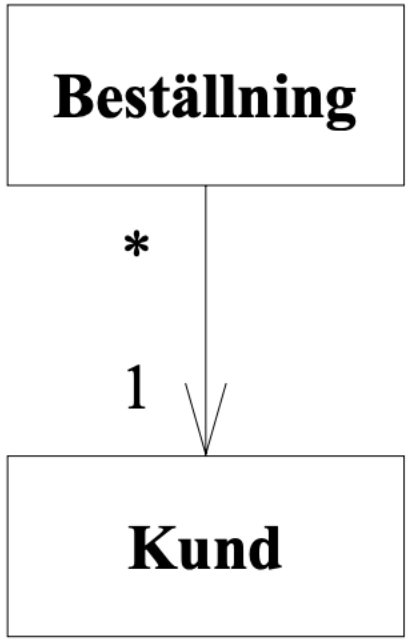
Många-till-många relationer

```java
public class Teacher {
    private String name;
    private List<Student> students;
    // getter and setter methods
}
public class Student {
    private String name;
    private List<Teacher> teachers;
    // getter and setter methods
}
```

https://www.javaguides.net/2018/08/association-in-java-with-example.html

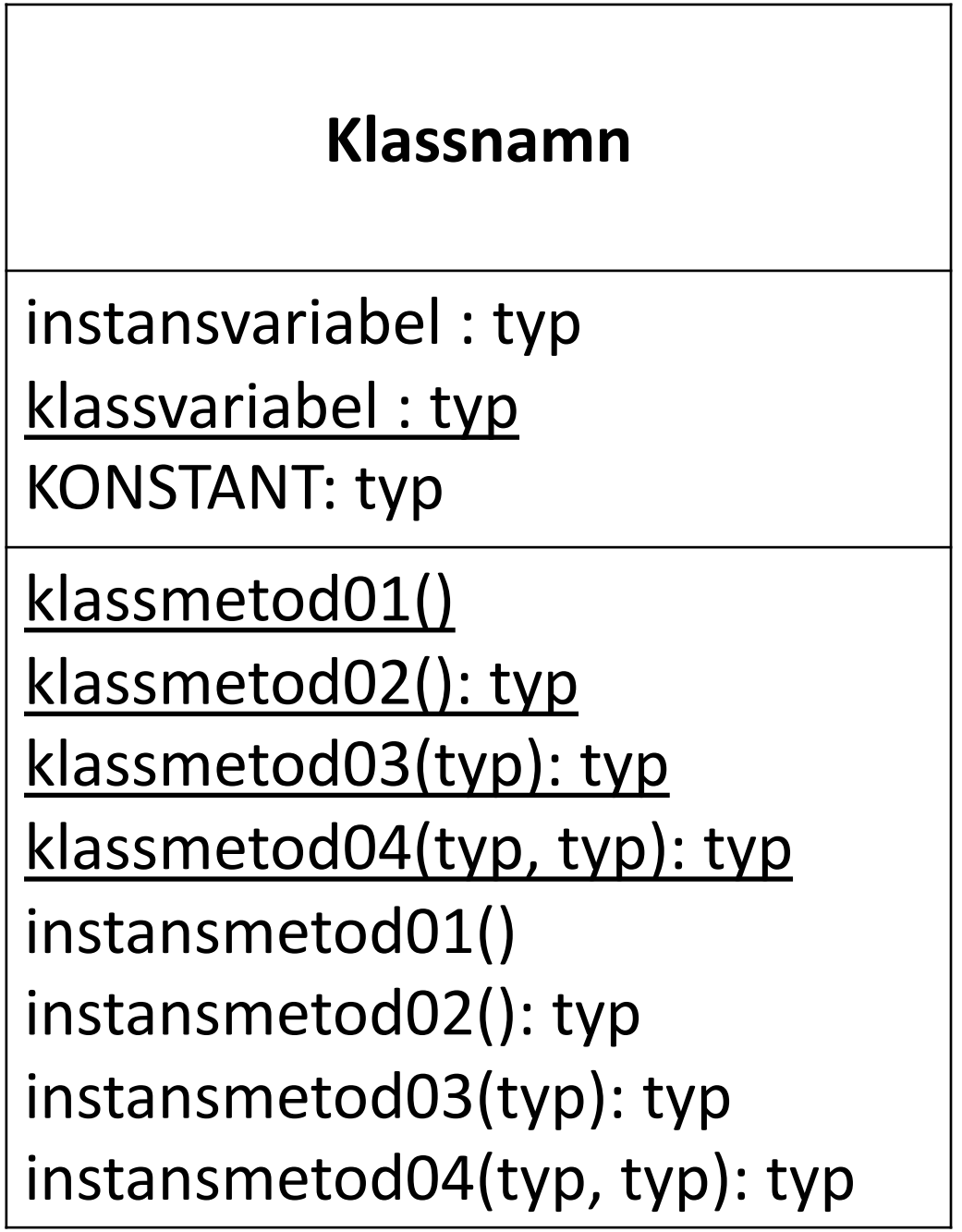
Navigerbarhet (navigability)

- En association kan vara riktad, vilket kallas navigerbarhet (navigability).
- **Riktning anger på vilken sida av associationen som ansvaret finns.**
- I en beställning av en produkt anges vilken kund som gjort en beställning. Man kan tänka sig att en beställning har ansvar för att upplysa om vilken kund som gjort beställningen.



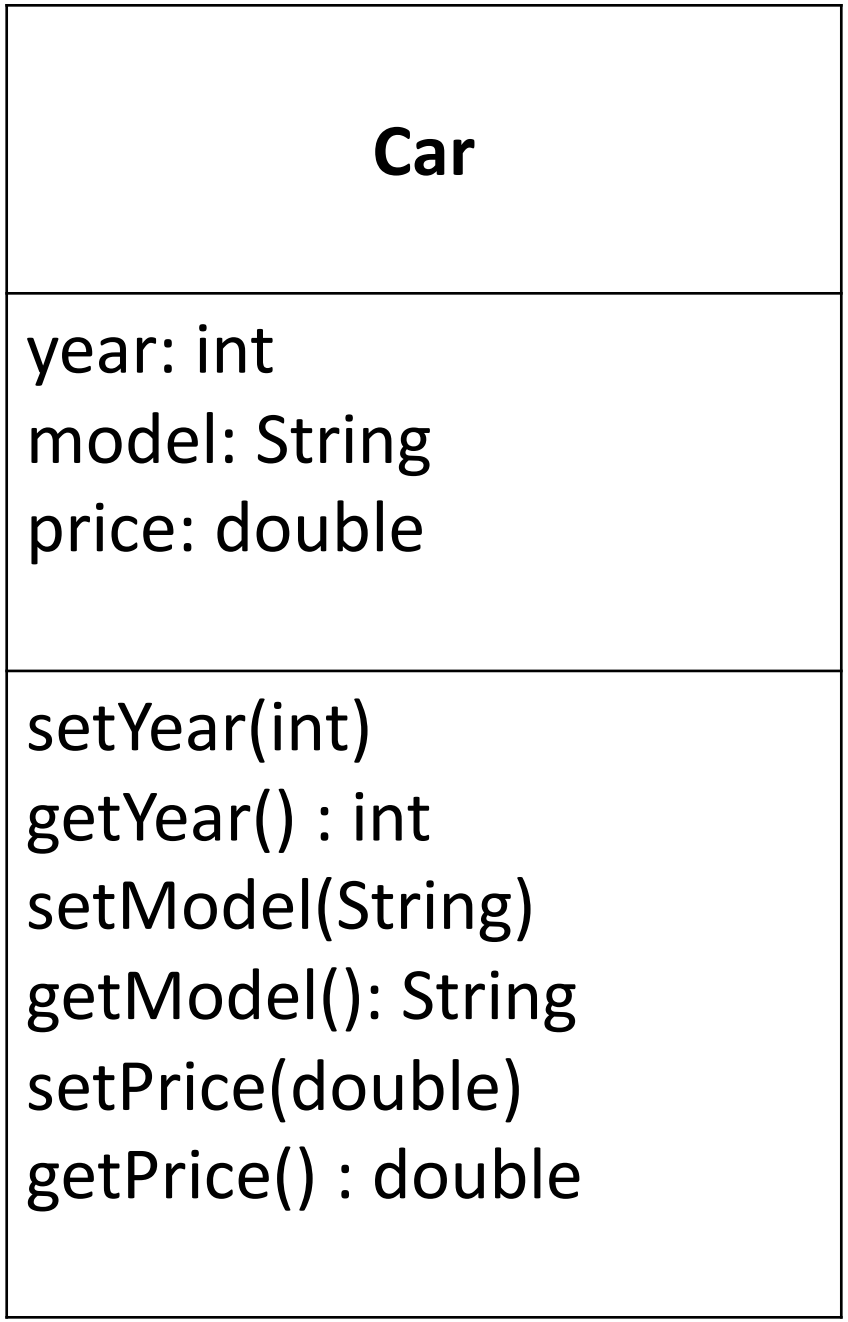
UML – Klassdiagram

- Rektangeln har tre sektioner
  1. Klassens namn
  2. Egenskaper/attribut t.ex.
     - Instansvariabler
     - Klassvariabler (static)
     - KONSTANTER
  3. Operationer (metoder)
     - Klassmetoder (static)
     - Instansmetoder



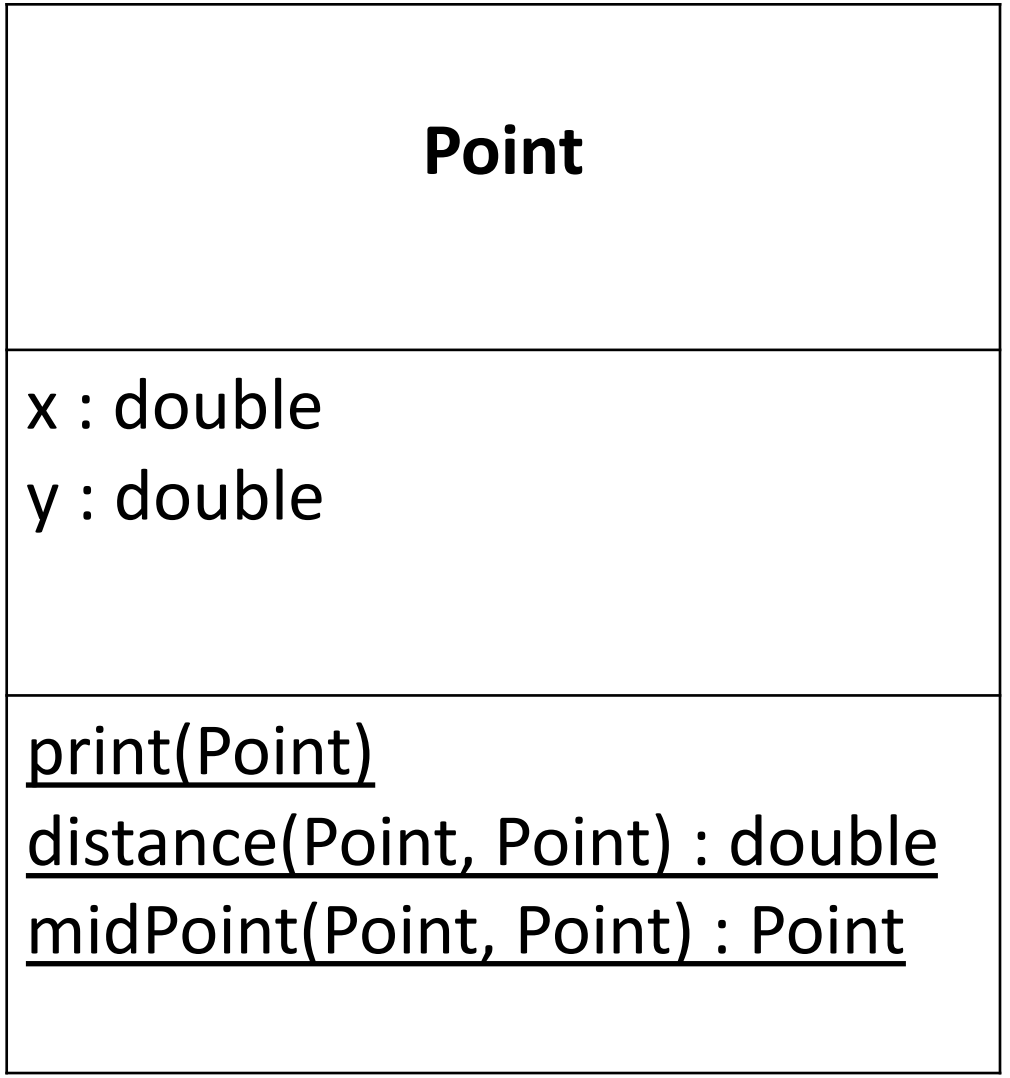
Class Diagram
Övning 1

- Implementera diagrammet (Skapa en klass i IntelliJ IDEA)

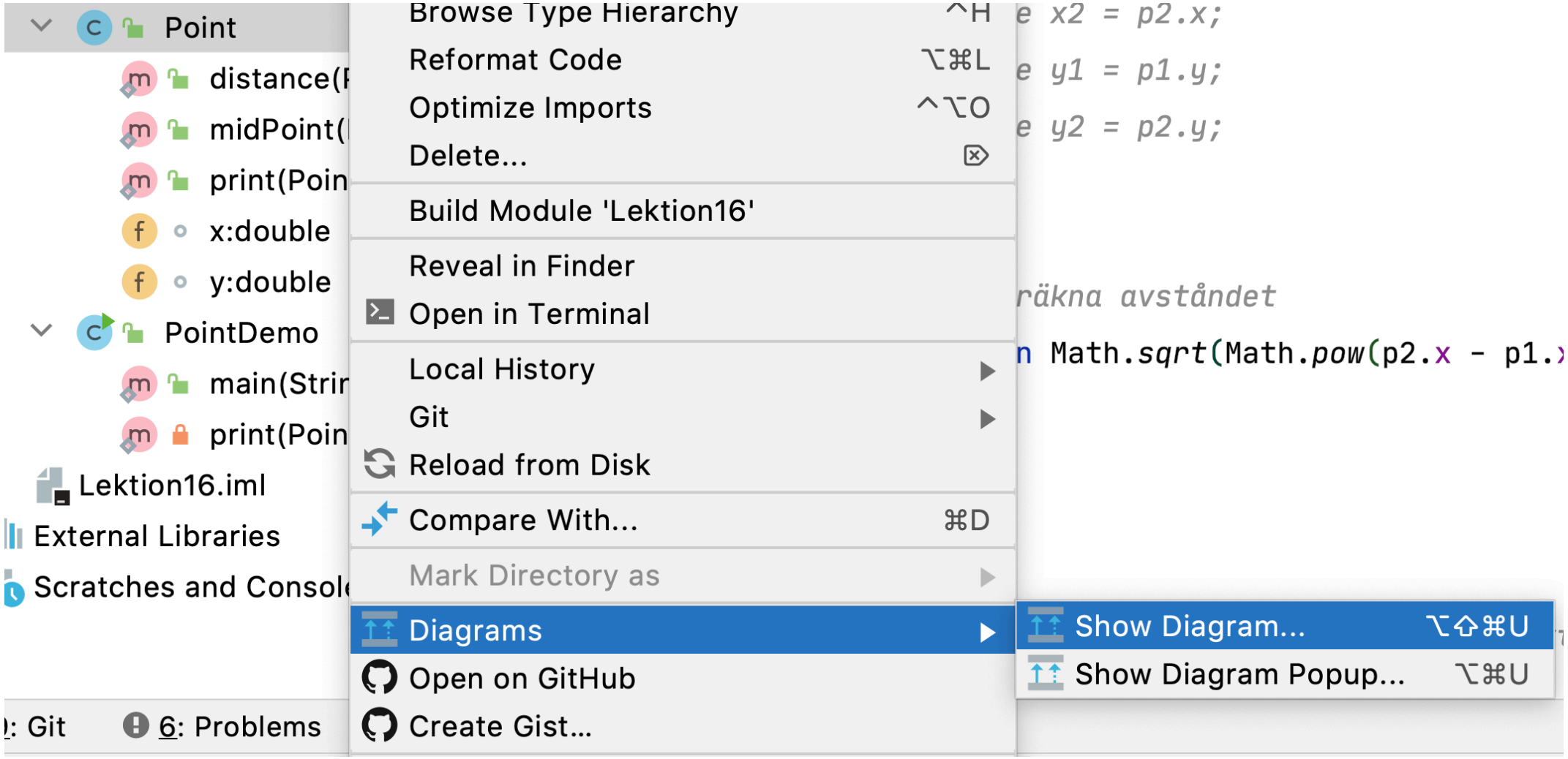


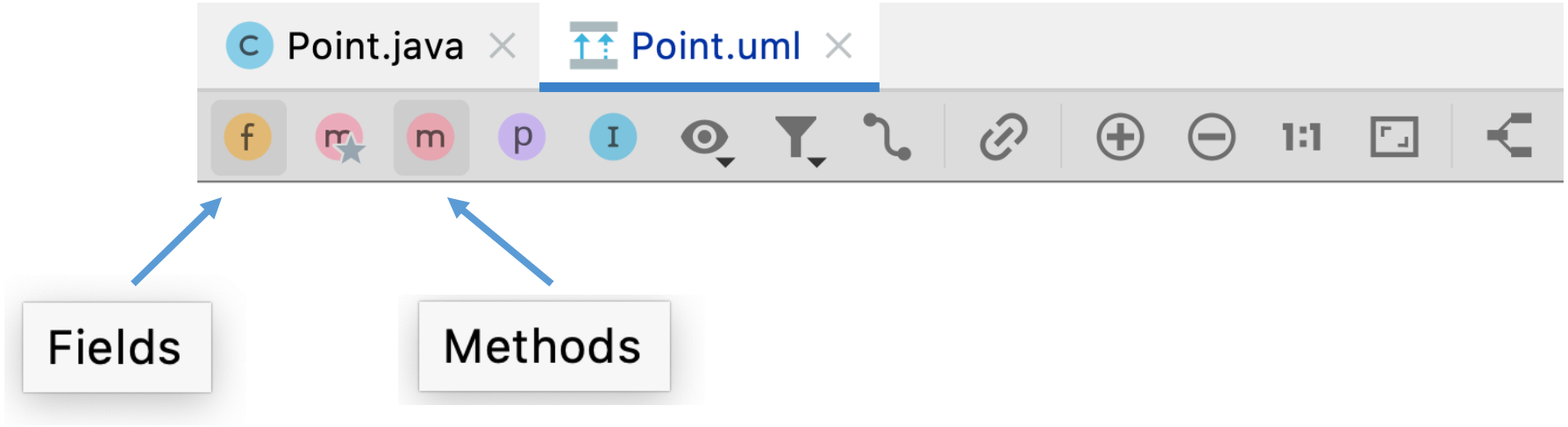
Class Diagram Övning 2

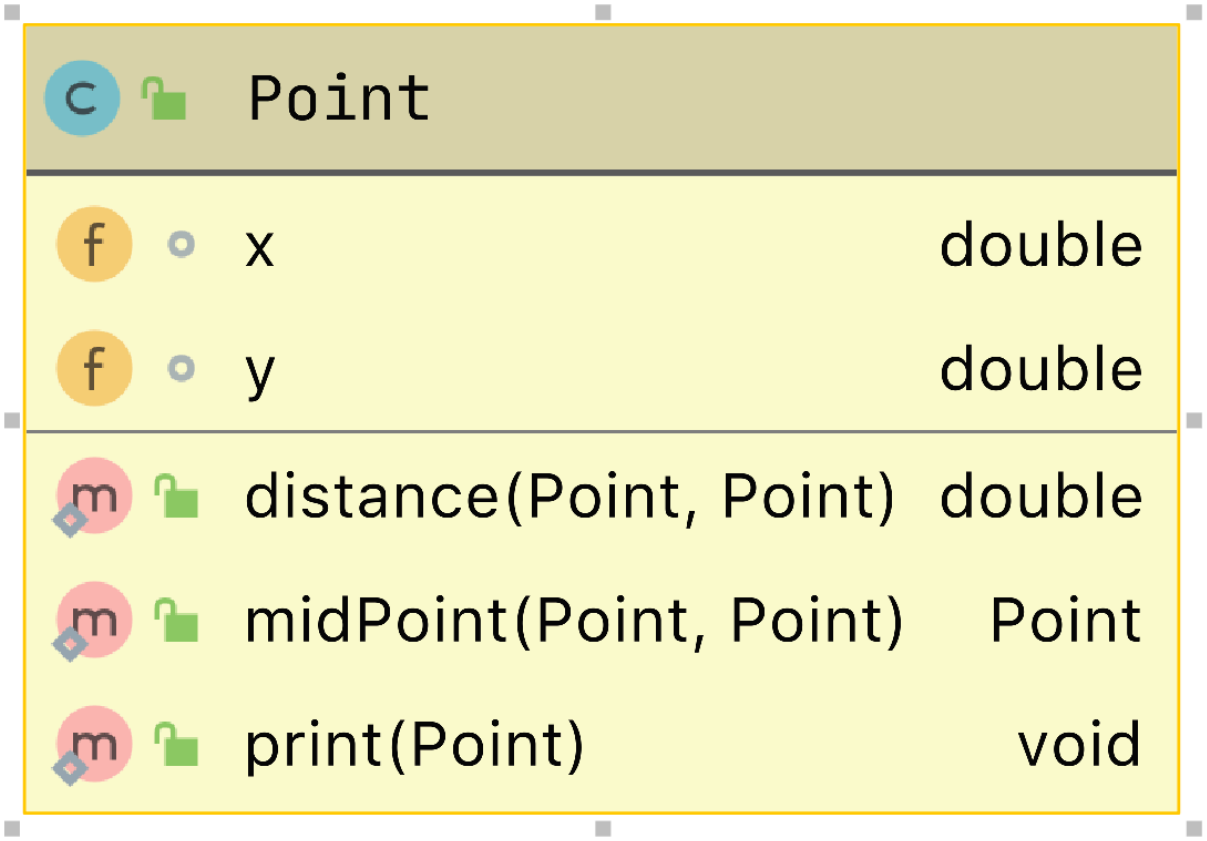
- Implementera diagrammet (Skapa en klass i IntelliJ IDEA)



Generera klassdiagram i IntelliJ IDEA Ultimate

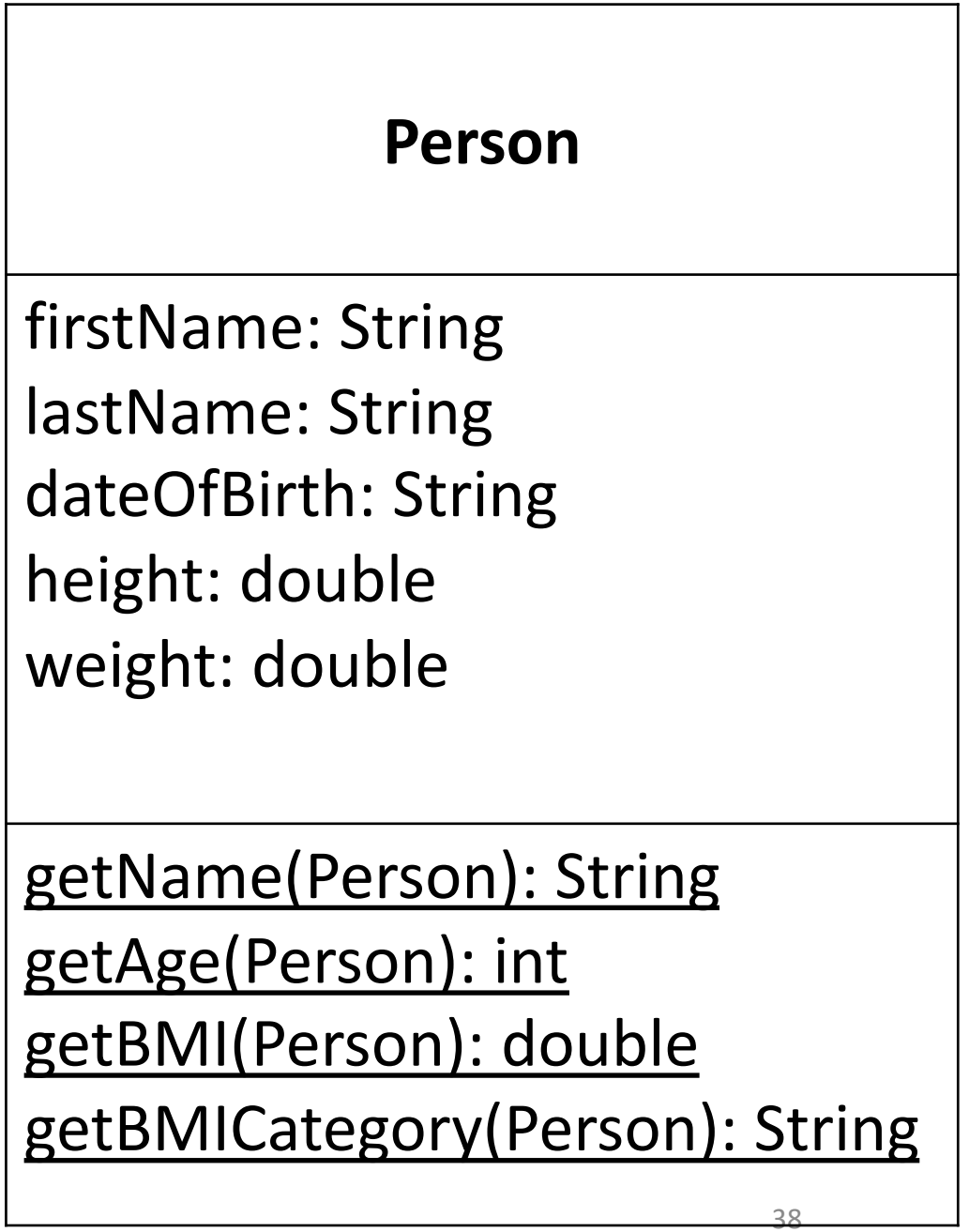



Point.java
Point.uml
Fields
Methods


Point
x double
y double
distance(Point, Point) double
midPoint(Point, Point) Point
print(Point) void

Övning 3

1. Skapa ett Javaprogram som innehåller en klass enligt klassdiagrammet här till höger.
2. Skapa några olika objekt/instanser av klassen.
3. Skriv ut följande info om varje objekt i terminalen.
    1. Namn: (hela namnet)
    2. Ålder: (enbart år t.ex. 45 år)
    3. BMI: (kroppsmasseindex t.ex. 23)
    4. Viktklass: (t.ex. Normalvikt)



Övning 4

1. Skapa ett Javaprogram som innehåller en klass enligt klassdiagrammet här till höger.
2. Skapa några olika objekt/instanser av klassen.
3. Skriv ut följande info om varje objekt i terminalen.
    1. Namn: (hela namnet)
    2. Ålder: (enbart år t.ex. 45 år)
    3. BMI: (kroppsmasseindex t.ex. 23)
    4. Viktklass: (t.ex. Normalvikt)

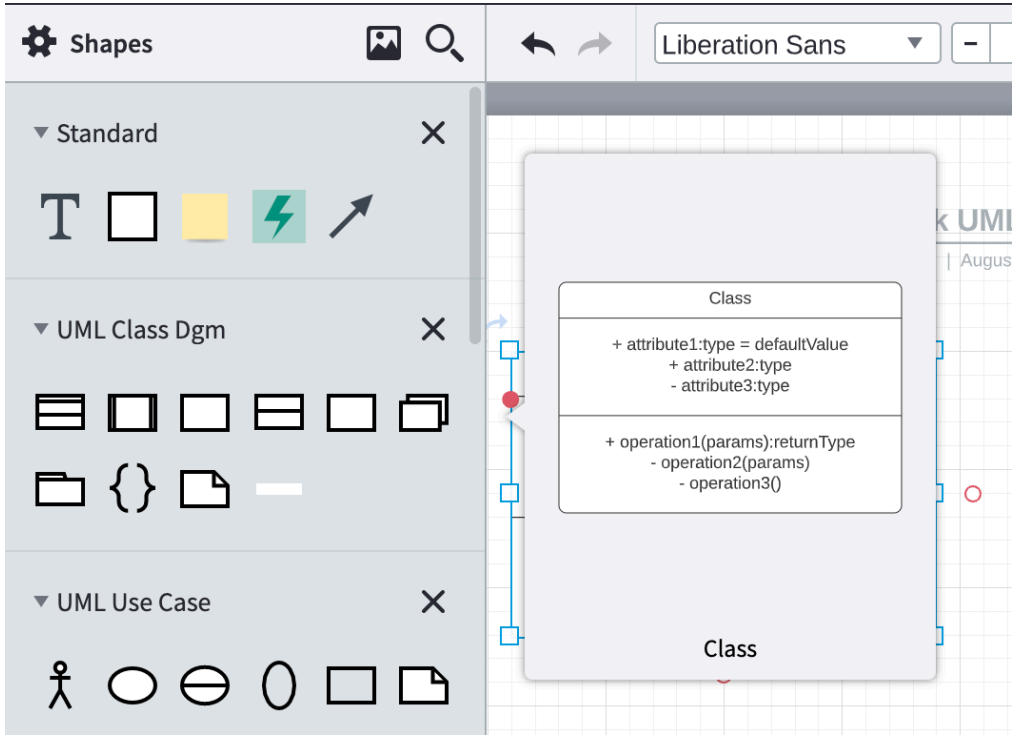
Person
firstName: String lastName: String dateOfBirth: String height: double weight: double
getName(): String getAge(): int getBMI(): double getBMICategory(): String

Övning 5

- **Diskutera i små grupper**
  - Vad är skillnaden mellan övning 3 och 4?
  - Vilken design är bättre? Varför?

Laboration 1 (arbeta i små grupper)
Genom denna laboration får ni möjlighet att lära er…
Modellera objekt och rita klassdiagram

1. Skapa klasser som beskriver följande objekt
    1. Hiss
    2. Kort i kortlek
    3. Produkt (i en webbutik)
2. Rita klassdiagram i Lucidchart.
3. Implementera nödvändig kod i intelliJ IDEA.
4. Redovisa och diskutera resultatet i små grupper.



Summering av dagens lektion

Mer om OOAD

- **Systemdesign/Arkitektur**
- **Modellering**: Introduktion till UML
  - Modelleringsverktyg
- Användningsfall (Use Case)
- Användningsfallsdiagram (Use Case Diagram)
- Klassdiagram (Class Diagram)

- **Reflektioner kring dagens lektion**
  - Vad tar du med dig från dagens lektion?
  - Finns det något som var extra svårt att förstå?
  - Finns det något som vi behöver repetera?
  - Hur upplevde du dagens arbetsmetoder?

NACKADEMIN

Framåtblick inför nästa lektion

- **Projektarbete (Kravanalys och Systemdesign)**
- Projektarbetet startar!
- Grupper bildas.

NACKADEMIN