

Analyse og Dokumentation af Database & OOP 2.HF

It & Data, Odense

UML værktøj

- **LucidChart** – download:

<https://lucidchart-diagrams-desktop.en.softonic.com/download>

Eksempler:

<https://www.lucidchart.com/pages/templates>

- **Draw IO** – download:

<https://www.microsoft.com/en-us/p/drawio-diagrams/9mvvszk43qqw?activetab=pivot:overviewtab>

Hurtig introduktion til Draw IO:

https://www.youtube.com/watch?v=cCMVEgmLazw&ab_channel=draw.io

Eksempel på hvordan et simpelt flow char tegnes i Draw IO:

https://www.youtube.com/watch?v=64MaQYyAN2w&ab_channel=Today'sTuts

Emne: Dokumentation

- Du skal dokumentere hele din opgave (dette er en forsmag på svendeprøven, hvor I skal skrive en proces- og en produktrapport).
- Start med at analysere, hvad der er behov for til at løse opgaven
 - Hvordan kan opgaven opdeles i mindre dele, der løses en efter en.
 - Se eventuelt "Hjælp til nedbrydning af ludo del 2"
- Dokumentation skal indeholde et UML klassediagram / domænemodel, hvor databasen indgår.
- Dokumentation skal være af teknisk natur og indeholde begrundelser for de valgte metoder, der indgår i løsningen.
- Du skal også skrive kommentarer i din kode, så det er tydeligt for en "kollega", hvad koden går ud på

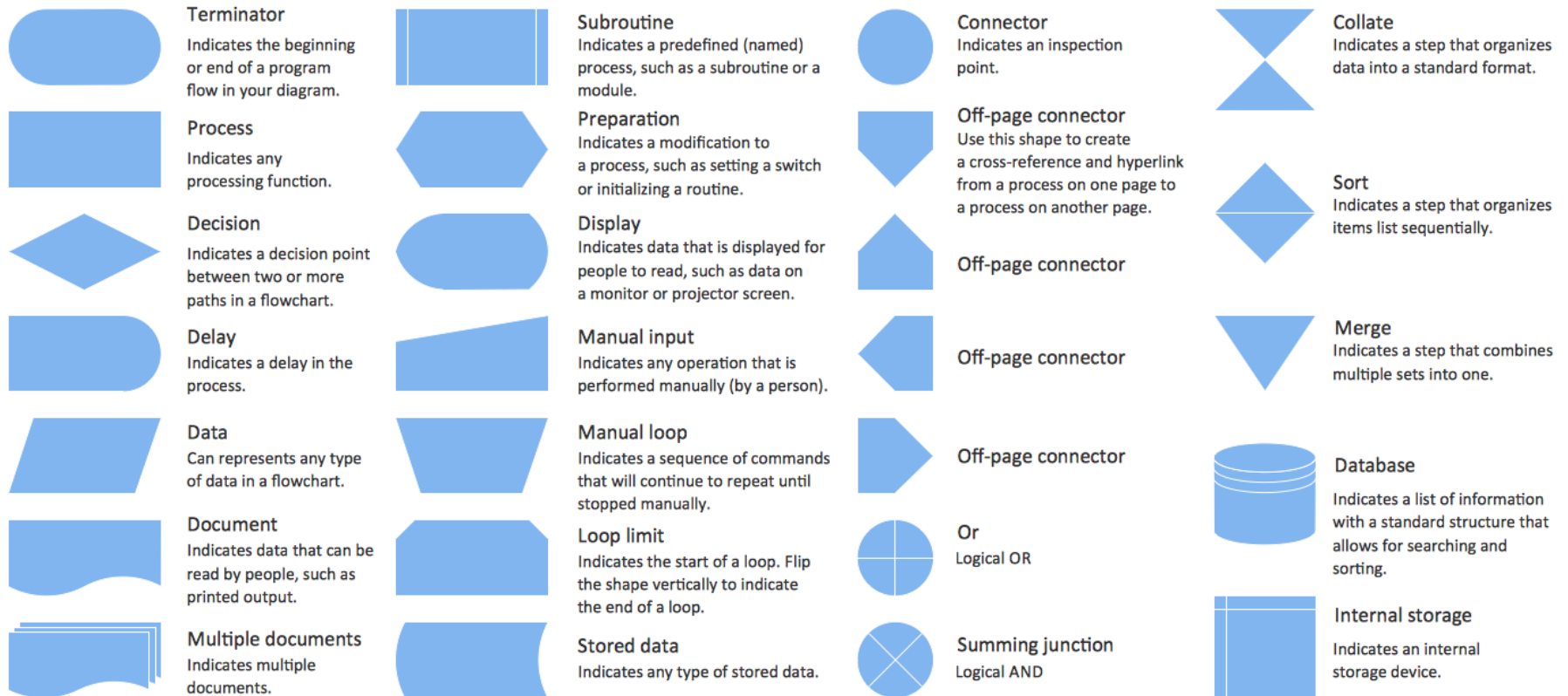
OOP: Dokumentation iflg. målpinde

- Dokumentation af grundlæggende viden om det valgte programmeringssprog.
- Redegøre for typer af collections som f.eks. `List<T>`
- Dokumentation af OOP koncept: Indkapsling, polymorfi og arv.
- Override versus overload af metoder
- Begrunde valg af virkefelter (access modifiers)
- UML klasse diagram
- Domænemodel ud fra best practice
- Redegøre for løs kobling / afhængige moduler
- Redegøre for grundlæggende problemstillinger med Thread Safety og Atomic State.
- Redegøre for anonyme og Lambda metoder

Note: Stammer fra H1

Dokumentation FlowChart

- OOP flow charts

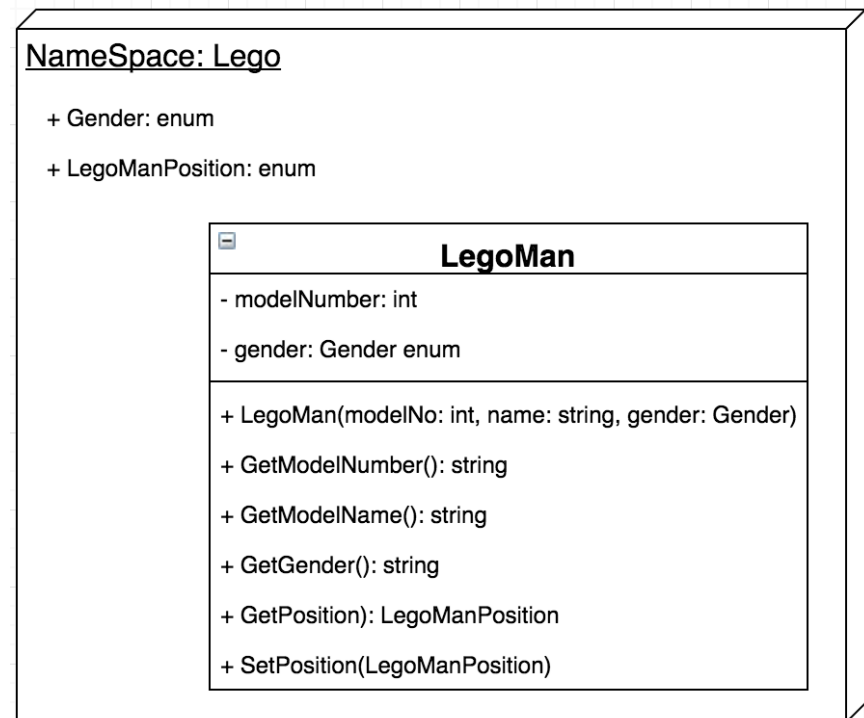


UML: Klasse diagram

- Konceptuelt klasse diagram
 - Vi kan starte med at modellere vores OOP design med et konceptuelt klasse diagram.



- Detaljeret klasse diagram



UML: Relationer i et klasse diagram

- **Association**

- Når et objekt, bruger et andet objekt.
- Ex. Når et objekt overføres (pares) til at andet objekt, som en parameter i en metode.

- **Aggregation**

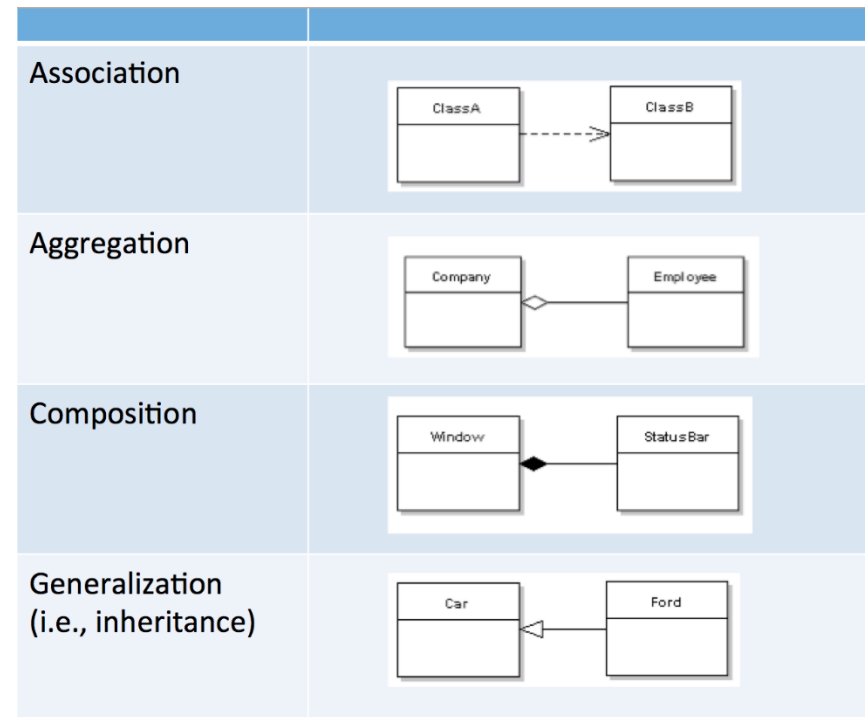
- Når et objekt tilhører et andet objekt.
- Ex. Når et objekt, der er overført (parsed) til et andet objekt, gemmes i en property.

- **Composition**

- Når et objekt er bundet til et andet objekt.
- Ex. Når et objekt instantieres i et andet objekt. Et Composite objekt, dør med dens "parent" objekt.

- **Generalization**

- Når et objekt er nedarvet fra et andet objekt.
- Ex. Nedarvning, implements interface.



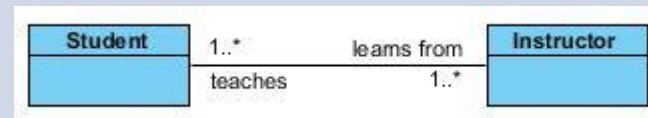
UML: Relationer i et klasse diagram

Association (Associeringsforhold)

En elev har flere lærer

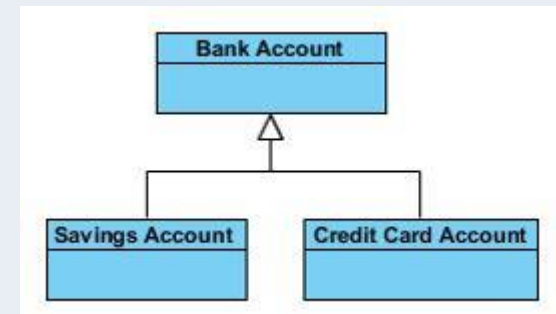
Hver lærer har en eller flere elever

Begge dele inkl. roller



Generalization (Generalisering/Arv)

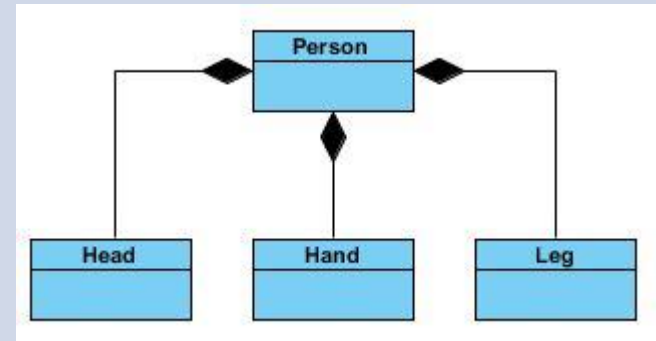
To typer konti: opsparing og kreditkort har fælles generelle egenskaber som f.eks. kontonummer, saldo osv. fra bank konto klassen – men de har også hver deres specielle egenskaber



UML: Relationer i et klasse diagram

Composition (Sammensætning)

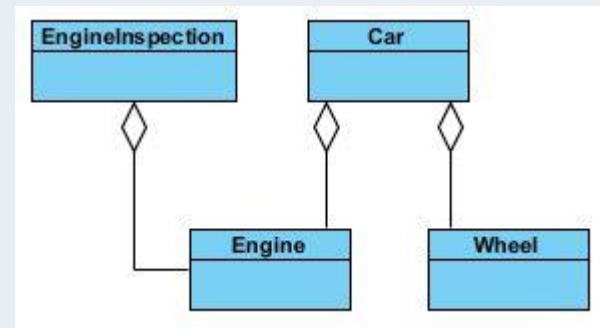
Hoved, hånd og ben kan ikke fungere uden person, der knytter det hele sammen.
Slettes person, forsvinder de andre også.






Aggregation (Aggregering)

Bil har brug for hjul og motor men ikke nødvendigvis et bestemt hjul eller motor.

Hjul og motor kan også anvendes af andre køretøjer (motorcykel, bus, lastbil). Et objekt af typen hjul eller motor kan sagtens have en betydning uden et bil objekt.



UML: Relationer i et klasse diagram

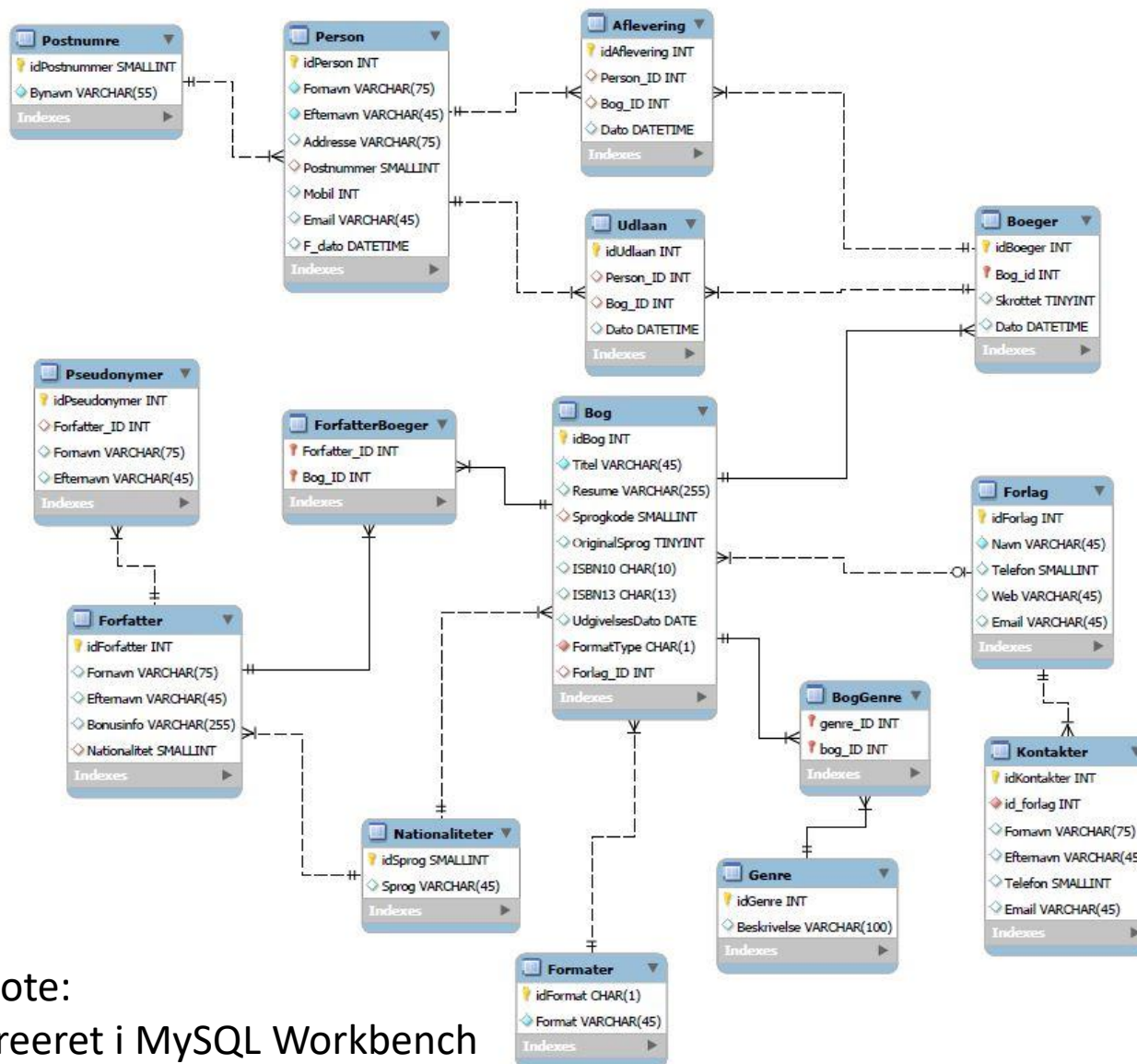
Association  (Associeringsforhold)	Aggregation  (Aggregering)	Composition  (Sammensætning)
Vises med en åben pil	Vises med et åbent diamant symbol	Vises med et udfyldt diamant symbol
Eksistere mellem flere klasser i UML.	Er en del af et associeringsforhold.	Er også en del af et associeringsforhold.
Kan være et forhold, der er en-til-en, en-til-mange eller mange-til-mange.	Kendetegnes ved et svagt forhold (associering) mellem klasserne.	Kendetegnes ved et stærkt forhold (associering) mellem klasserne.
Objekter er forbundet med hinanden.	Tilknyttede objekter er ikke afhængige af hinanden.	Objekter er afhængige af hinanden.
At slette et objekt kan muligvis påvirke et andet men ikke nødvendigvis.	At slette et objekt har ingen påvirkning på et andet	At slette et objekt betyder, at det andet også slettes.
Eks.: En lærer er associeret med flere elever	Eks.: En bil har brug for et hjul, men ikke nødvendigvis et bestemt hjul.	Eks.: En fil er placeret i en mappe. Hvis mappen slettes, forsvinder filen også.

Database: Dokumentation iflg. målpinde

Krav til indholdet af dokumentationen for databasen

- E/R diagram
- Dokumentere den valgte løsning ved hjælp af UML
 - Redegøre for konsistenskrav, referenceintegritet, relationer og andre constraints.
 - Normalisering af database
- Dokumentere performance-målinger

E/R diagram



Note:
Kreeret i MySQL Workbench

Datamodellering med UML

- CASE OPGAVE:**

Vi skal designe en database til et bibliotek.

Idéen er, at vi kan holde styr på bibliotekets bøger, samt udlån af bøger til bibliotekets kunder.

- Her er et udkast over de data vi ønsker i deres nuværende datamodel.

Laaner	Bog1	Bog2	Bog3	Dato
Søren Hansen, Sørenvej 14, 5000, Odense C	En fiaskos perfekte liv, Jan Magnussen, Forlaget Finsen	Tommy og Tanne, Tom Buk-Swienty, Gyldendag	NULL	22/01/2017
Søren Hansen, Sørenvej 14, 5000, Odense C	Djævelen i hullet, Leif Davidsen, Lindhart og Rindhof	NULL	NULL	23/01/2017
Peter Jensen, Petervej 2, 8000, Århus	En fiaskos perfekte liv, Jan Magnussen, Forlaget Finsen	NULL	NULL	02/02/2017
Klaus Sørensen, Klausvej 19, 5230, Odense M	Tommy og Tanne, Tom Buk-Swienty, Gyldendag	NULL	NULL	01/02/2017

- Hvilke **problematikker** er der med den nuværende datamodel?

Datamodellering med UML

- Når vi skal designe en database, kan vi anvende **datamodellering** til at nedbryde problemstillingen.

Datamodellering betyder; "det at illustrerer en **problemstilling** med **grafiske modeller**".

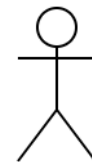
- Som et værktøj til **datamodellering**, kan vi anvende en **model** kaldet: **USE CASE**. USE CASEN bruges til at identificere de arbejdsopgaver/processer, der er forbundet med en problemstilling.

USE CASEN stammer fra **UML** (**U**nified **M**odelling **L**anguage), der generelt anvendes til modellering af Objekter og deres relationer. Anvendes ofte til Objektorienteret design i Objektorienteret programmering (OOP).

- For at overskueliggøre opgaven og finde ud af, hvilke data vi skal ende ud med, skal vi lave et **USE CASE diagram**.
- Vi starter med at identificeret nogle **aktører** (Actors). En aktør er i bund og grund en bruger af dataene vi skal ende ud med.

Q) Hvem vil have adgang til at manipulere med vores data?

A) – Bibliotekaren (og ikke andre i denne case).



Aktør

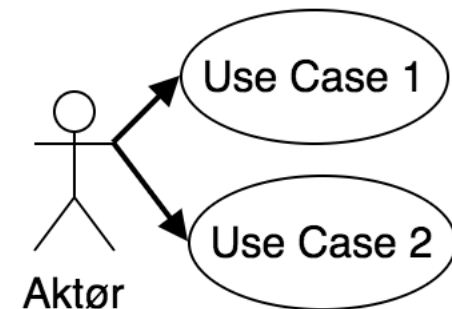
Datamodellering med UML

- Når vi har identificeret vores aktør(er), skal vi finde ud af, hvilke **USE CASES** vores aktør(er) vil gøre brug af.

Q) Hvad vil bibliotekaren typisk foretage sig med vores data?

A) Use cases:

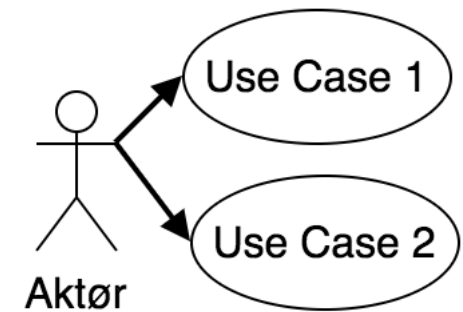
1. Opret bog
2. Rediger bog
3. Slet bog
4. Slå bøger op og se information
5. Oprette låner
6. Rediger låner
7. Slet låner
8. Udlån bøger til låner
9. Se udlånte bøger og låner
10. Modtag udlånte bøger fra låner



Hvad er **Inputdata**? og hvad er **Outputdata**?

Datamodellering med UML

- Når vi har identificeret vores **USE CASES**, skal vi finde ud af, hvilke USE CASES, der **inputdata** og hvilke der er **outputdata**.
- Inputdata:** Er data, der skal gemmes og som er unikt eller identificerbart.
- Outputdata:** Konstrueres ud fra eksisterende inputdata og skal dermed ikke inkluderes i vores datamodel.



Inputdata:

*Oprette/slette/redigere bøger
(3 Use Cases)*

*Oprette/slette/redigere låner
(3 Use Cases)*

Udlåne bøger til låner

Modtage udlånte bøger fra låner

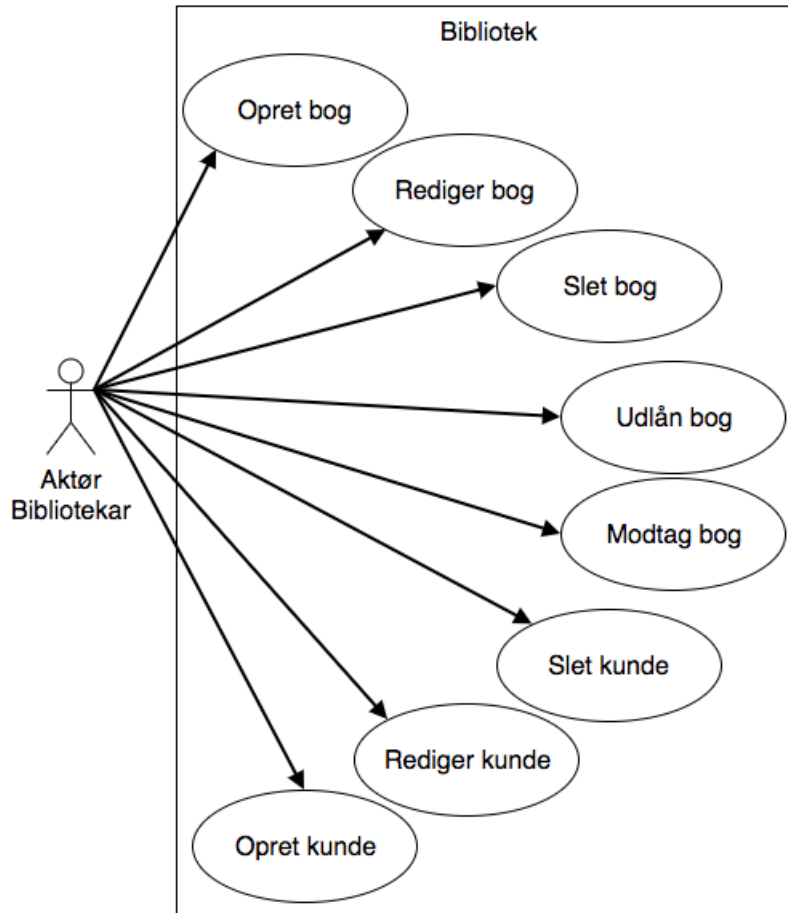
Outputdata:

Slå bøger op og se information

Se udlånte bøger og lånerdata

Datamodellering med UML

- Vi kan ud fra vores identificerede **inputdata**, lave et færdigt **USE CASE diagram**, som nedenfor.



Datamodellering med UML

- Nu, hvor vi har vores **inputdata**, kan vi begynde at inddele dem i en logisk sammenhæng (model).
- Til dette formål kan vi anvende endnu et **UML diagram**, kaldet **Class Diagram** (klassediagram).

Class
Attributes Properties Fields
Methods

(En klasse i Objektorienteret programmering, bruges som skabelon for objekter)

- Vi kan ud fra vores **inputdata**, udarbejde en **simpel** og **udetailjeret** datamodel. Først ved at identificere de oplagte "**objekter**" og deres **relationer**.
 - Låner
 - Bøger
 - Relation mellem bøger/låner

Inputdata:

Oprette/slette/redigere **bøger**

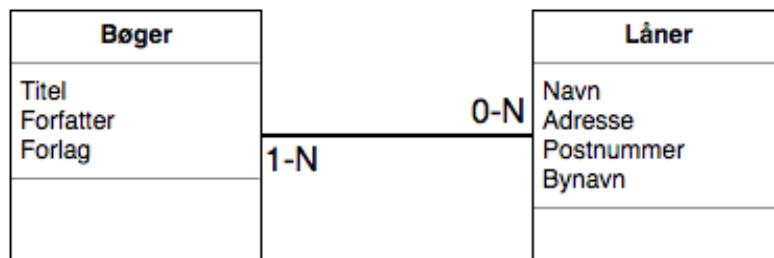
Oprette/slette/redigere **låner**

Udlåne **bøger** til **låner**

Modtage udlånte **bøger** fra **låner**

Datamodellering med UML

- I et simpelt **CLASS diagram**, kan vi nu overskueliggøre vores **objekter** og deres **relationer** som nedenfor.



- Når vi læser et **CLASS diagram**, læser vi relationerne fra begge sider.
 - **Bøger** kan have 0 til mange låner (**0-N**).
En bog kan godt være en bog, uden den er udlånt til en låner.
 - **Låner** kan have 1 til mange bøger (**1-N**).
En låner, kan ikke være en låner, hvis han ikke har lånt minimum 1 bog.

Datamodellering med UML

Oversættelse fra UML Class diagram til RDBSM database begreber

UML	RDBSM
Class	En tabel med en primærnøgle
Attributes/Properties/Fields	Kolonne i tabel med en passende datatype
Object (Instans af Class)	En datarække i en tabel
1 til mange relation	En fremmednøgle i en tabel, relateres til en specifik primærnøgle i en datarække i en anden tabel.
Mange til mange relation	Opret en ny tabel med 2 fremmednøgler. Fremmednøglerne relaterer til de 2 tabellers primærnøgle, der har en mange til mange relation.

Datamodellering med UML

- Det kan ofte være en fordel, at visualisere en GUI (Graphical User Interface) til sine USE CASES.
- Eks. USE CASEN: "Opret bog" kunne i en GUI, se således ud:

Opret ny bog:

Titel:

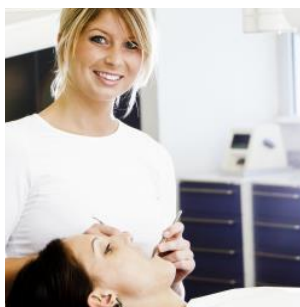
Forfatter: ▼

Forlag: ▼

- Vi kan se, at **forfatter** og **forlag**, skal vælges med en **dropdown-menu**. Derfor må hver dropdown-menu skulle fyldes med data, fra nogle tabeller. På denne måde har vi **identificeret** et behov for **2 nye tabeller** kaldet: forfatter og forlag.
- Lav selv flere GUI interfaces ud fra USE CASES, for at identificere flere tabeller.

Slut på Dokumentation.

Spørgsmål?



Campus M – Vi uddanner Danmarks dygtigste