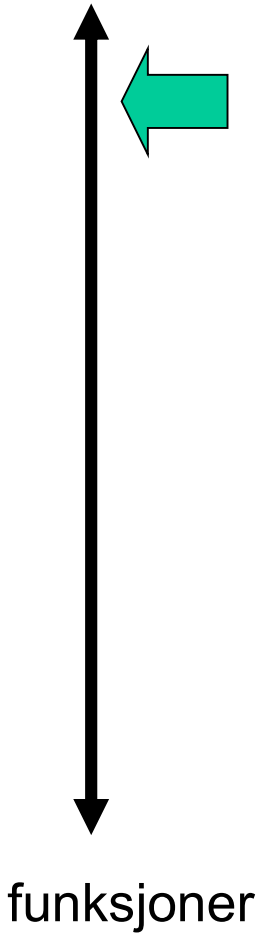
Datamodellering



- Objekter representerer ofte fenomener fra virkeligheten
- Disse fenomenene er ofte knyttet sammen i et nettverk av koblinger
  (ofte kalt relasjoner eller assosiasjoner)
- Det er ofte regler for hvilke (og hvor mange) objekter som kan kobles sammen
- Det er greit å ha oversikt over alt dette før en koder, derfor lager en *datamodeller* og tegner *diagrammer*

Eksempel: Person

- En *person* har **navn**, **e-post** osv.

klasse

Person

String navn
String e-post

instanser

#1: Person

navn = ”Hallvard”
e-post =”hal@idi.ntnu.no”

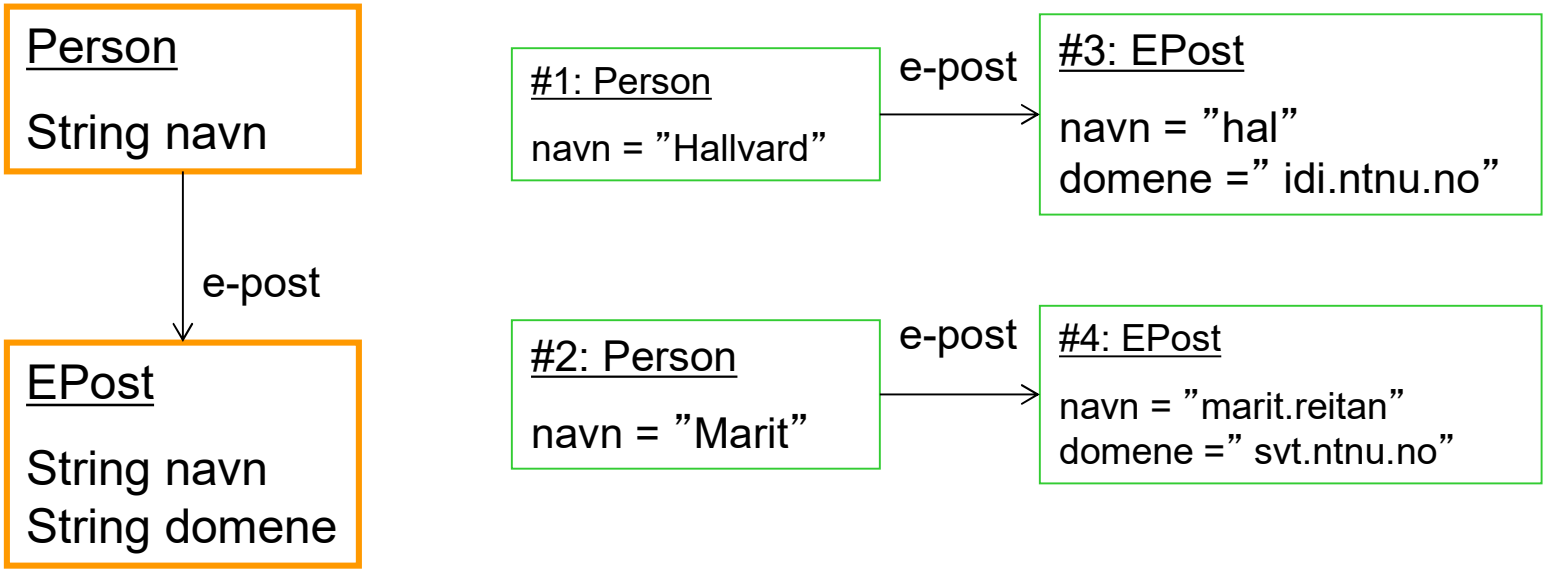
#2: Person

navn = ”Marit”
e-post = ”marit.reitan@svt.ntnu.no”

- Klassen beskriver det som er felles egenskaper i all instansene

Eksempel: Person

- En *person* har **navn**, **e-post** osv.
- Er e-post ”verdig” egen klasse?

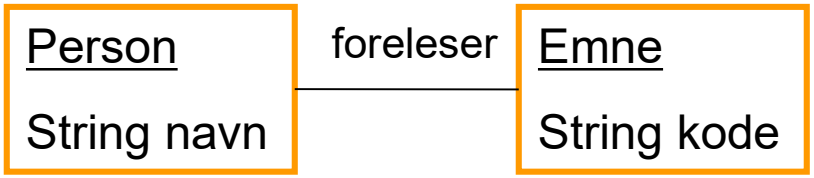


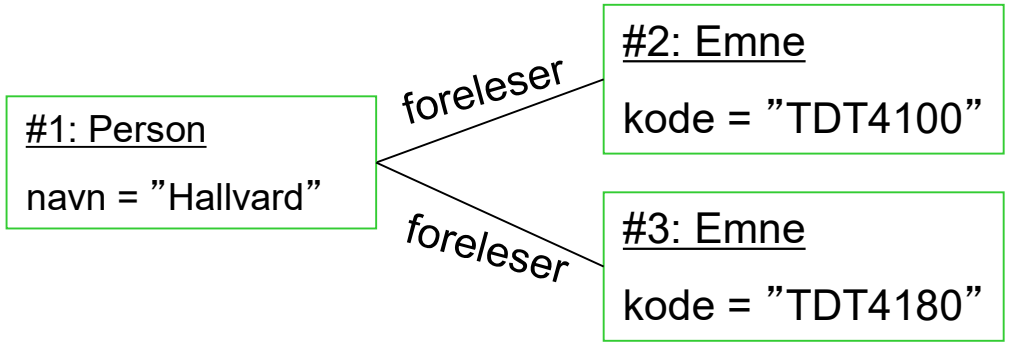
Tenk litt på Person, Car, Plate…

- Var det nødvendig å la Plate ha sin egen klasse?
- Hvor mye skal til for at noe skal fortjene sin egen klasse?
- Er det knyttet mot fysiske skiller, kompleksitet, bruk…

Eksempel: Person++

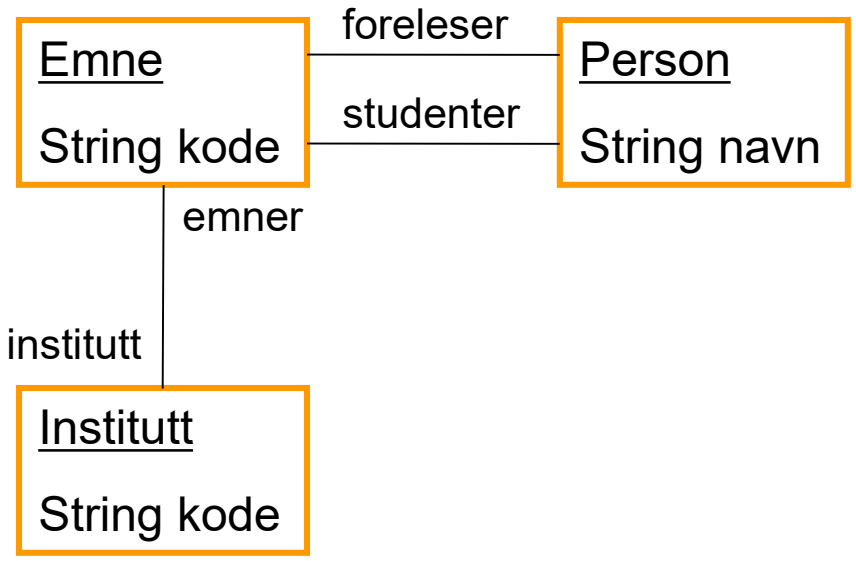
- …
- En person kan være *foreleser* i *emner*





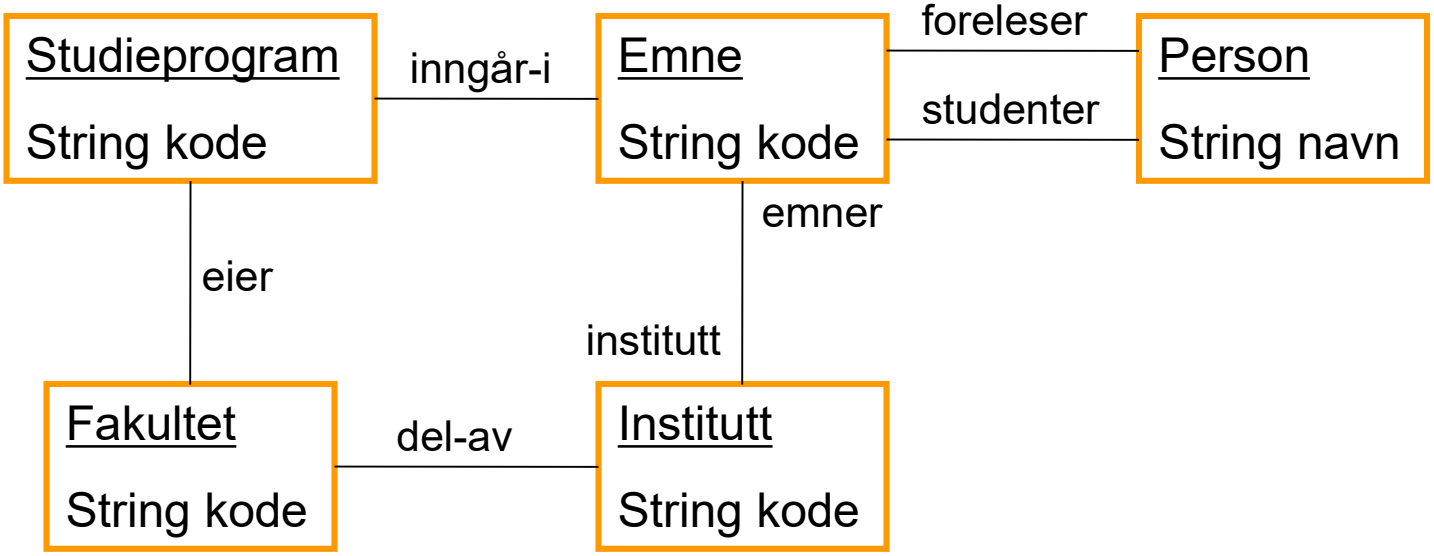
Eksempel: Person++

- …
- En person kan være *foreleser* i *emner*
- Emner tas av *studenter* og gis av *institutt*



Eksempel: Person++

- …
- En person kan være *foreleser* i *emner*
- Emner tas av *studenter* og gis av *institutt*
- Emner inngår i studieprogram, som eies av et fakultet…



Sosiale medier

- Hva slags data har vi her?
- Hvordan skal vi beskrive det, så vi kan lage koden for å representere dataene?

Facebook-begreper

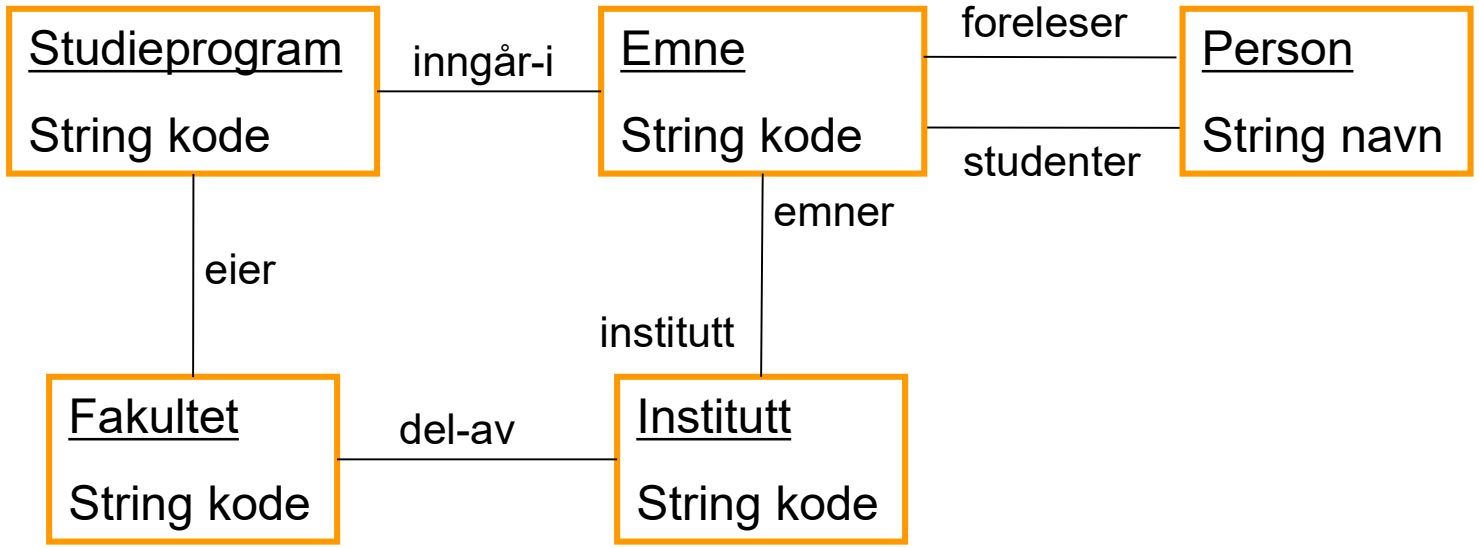
- Person
- Friend
- Group
- Wall
- Page
- Public
- Friend of friend
- Post
- Share
- Comment
- Message
- Like

Spørsmål om assosiasjoner

- Hvor mange assosiasjoner tillates?
  - **multiplisitet**:
    én til én (eller ingen) eller
    én til mange (bestemt antall eller flertall)?
- Er assosiasjoner to-veis?
  - **navigerbarhet**: skal begge ender vite om den andre?
  - **roller**: bruker en egne navn på hver retning?
- Impliserer assosiasjoner **inneholdt-i**-logikk
  - et objekt kan bare være direkte **inneholdt-i** ett annet objekt
  - når et objekt slettes, så slettes objekter som er **innholdt-i** det også

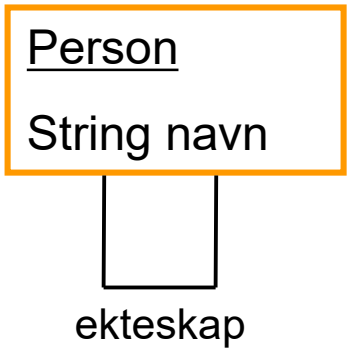
Spørsmål om assosiasjoner

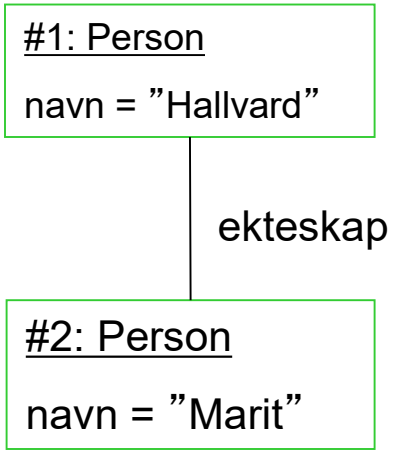
- **multiplisitet**: antall koblinger
- **navigerbarhet og roller**: retning og navn på kobling
- **aggregering/komposisjon**: eierskap



Assosiasjon innen klasse

- En *person* har **navn**, **e-post** osv.
- En person kan være *knyttet* til en annen person gjennom *ekteskap/partnerskap*





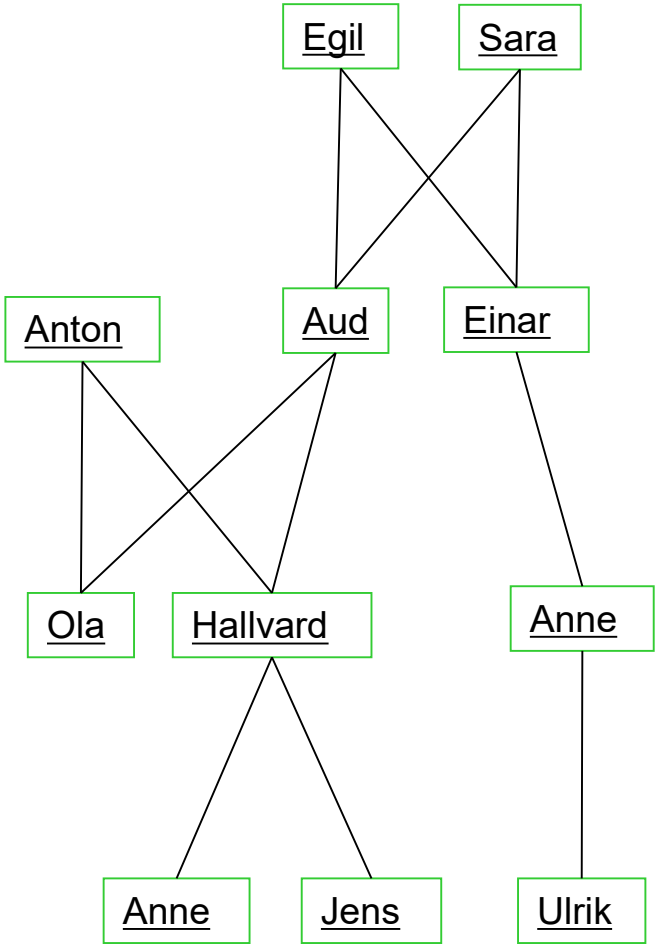
- En spesifikk person kan ikke være sin egen ektefelle/partner!

Ekstra spørsmål (beskrankninger/constraints)

- Refleksivitet: A -> A ($\forall x \in X : x \, R \, x$)
  - "ser lik ut", "er lik" for alle naturlige tall
  - anti-refleksiv: kobling til seg selv er ulovlig
    - A er ikke gift med seg selv.
- Symmetri: A -> B => B -> A
  - A gift med B
  - anti-symmetrisk: kobling tilbake er ulovlig
    - Armen er en del av meg, men jeg er ikke en del av armen
- Transitivitet: A -> B & B -> C => A -> C
  - Hånda er en del av armen, armen er en del av meg… hånda del av meg

Inverse og avledede assosiasjoner

- Familiebegreper:
  - **søsken**: barn av forelder
  - **besteforelder**: forelder til forelder
  - **tante/onkel**: søster/bror til forelder evt. deres ektemake
  - **niese/nevø**: sønn/datter av søsken
  - **kusine/fetter**: datter/sønn av onkel eller tante
  - **filleonkel/tante**: kusine/fetter til forelder
  - **tremenning**: barn av fille-onkel eller -tante evt. barnebarn av oldeforeldre (tre nivå opp til forforelder)



Person/familie-relasjoner

	Refleksiv	Symmetrisk	Transitiv
Søsken			
Halvsøsken			
Partnerskap			
Etterkommer			
Slektskap			
Venn			

Notasjon for assosiasjoner

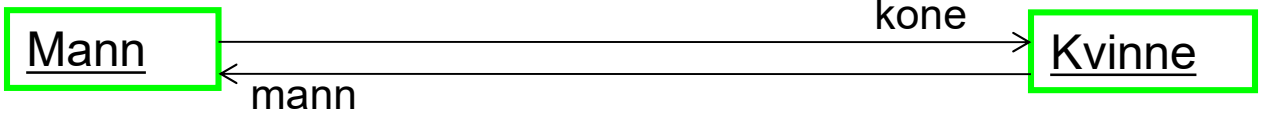
- **multiplisitet**: antall koblinger
- **navigerbarhet og roller**: retning og navn på kobling

- En instans av Klasse1 har minst **min2** og maks **max2** **rolle2**-koblinger til instanser av Klasse2.
- En instans av Klasse2 har minst **min1** og maks **max1** **rolle1**-koblinger til instanser av Klasse1.
- *Notasjon:*
  - *Når max er ubegrenset, så brukes **n** eller *****
  - *assosiasjonsnavnet utelates ofte*

Eksempel: 1-1

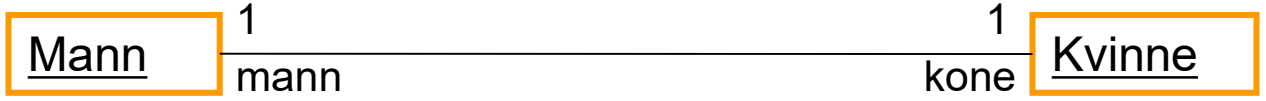
- En instans av Mann har minst **0** og maks **1 kone**-kobling til instanser av Kvinne *og motsatt*

gift mann og kvinne

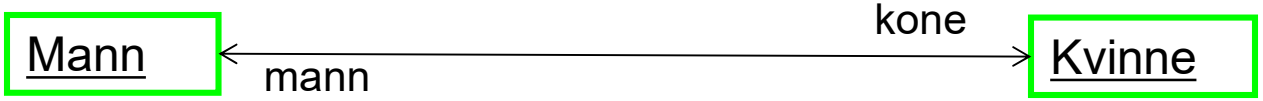


- Forenklet notasjon

*når **min** er 0, kan den utelates*

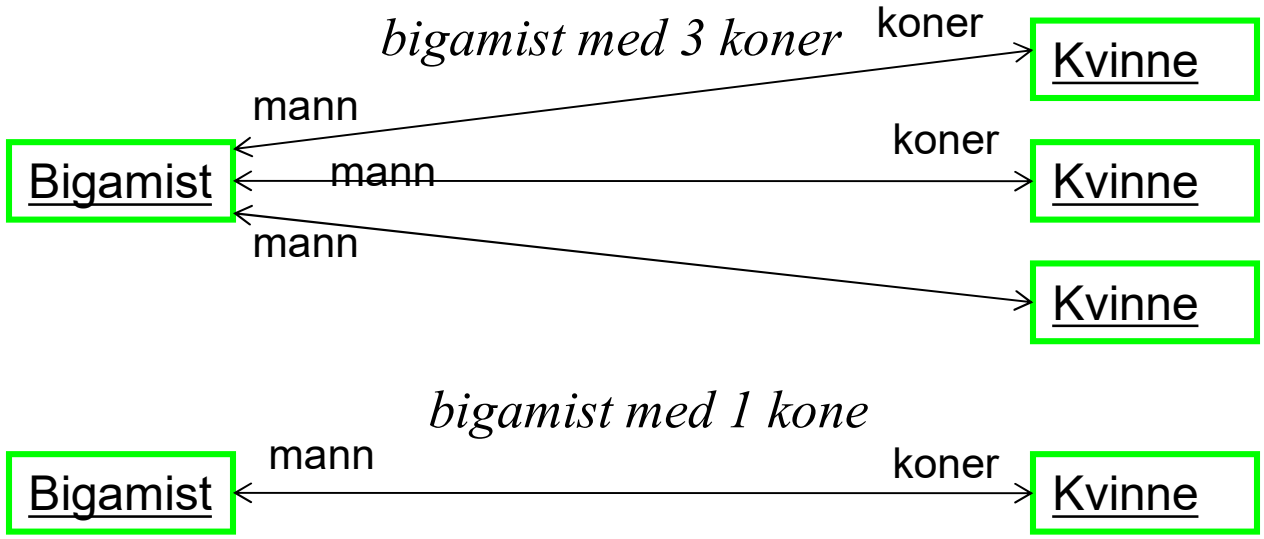


når koblingen går begge veier, så trengs bare én strek



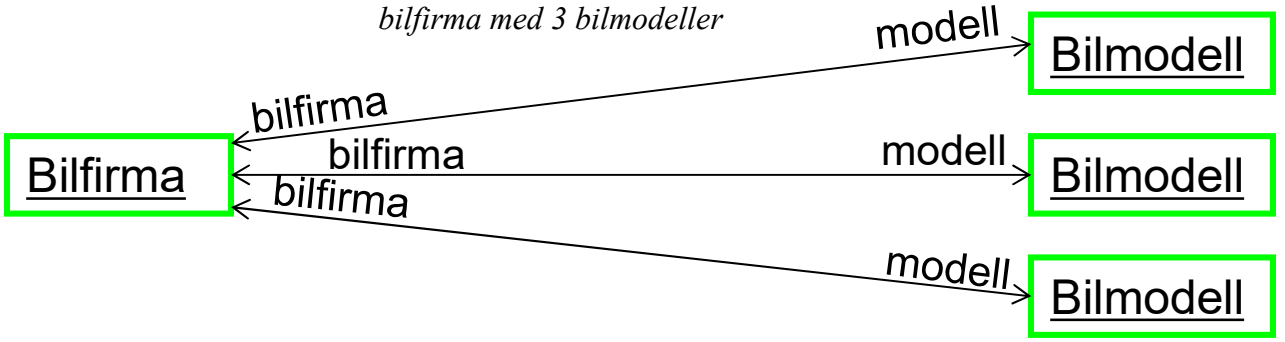
Eksempel: 1-n

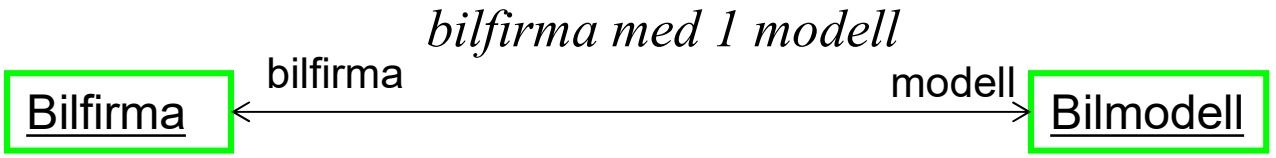
- En instans av Bigamist har (minst **0** og) ubegrenset antall **koner**-koblinger til instanser av Kvinne.
- En instans av Kvinne har (minst **0** og) og maks 1 **mann**-kobling til instanser av Bigamist.



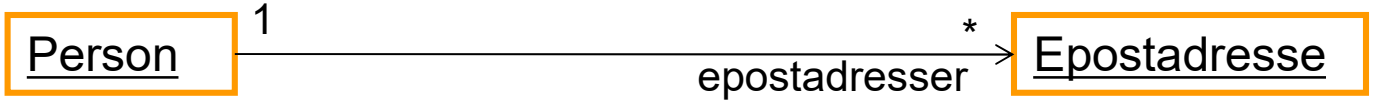
Eksempel: 1-n

- En instans av Bilfirma har (minst **0** og) ubegrenset antall **bilmodeller**-koblinger til instanser av Bilmodell.
- En instans av Bilmodell har (minst **0** og) og maks 1 **bilfirma**-kobling til instanser av Bilfirma.



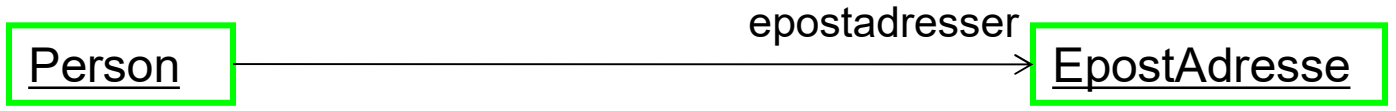


Eksempel: enveis 1-n

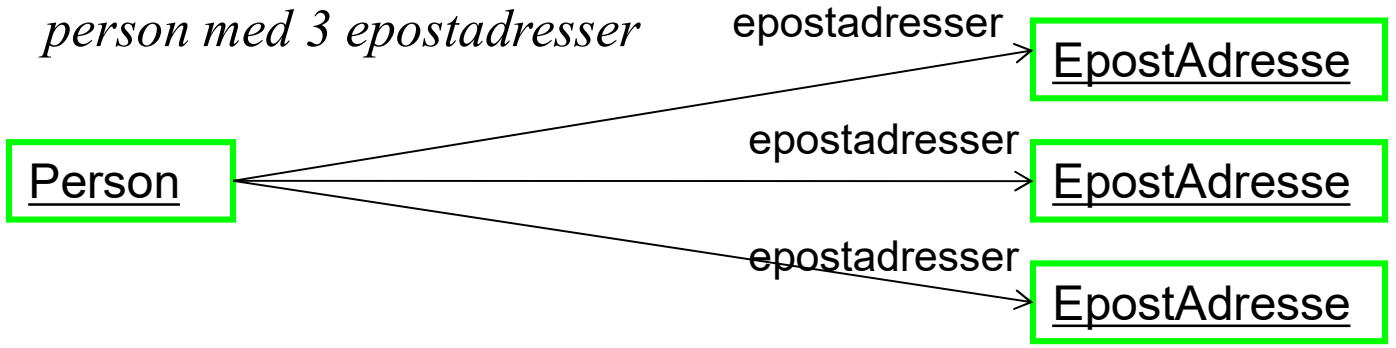


- En instans av Person har ubegrenset antall **ePostAdresser**-koblinger til instanser av Epostadresse.
- Spesialnotasjon: enveis-assosiasjoner tegnes med pil

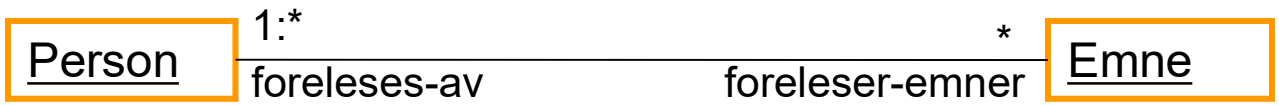
person med 1 epostadresse



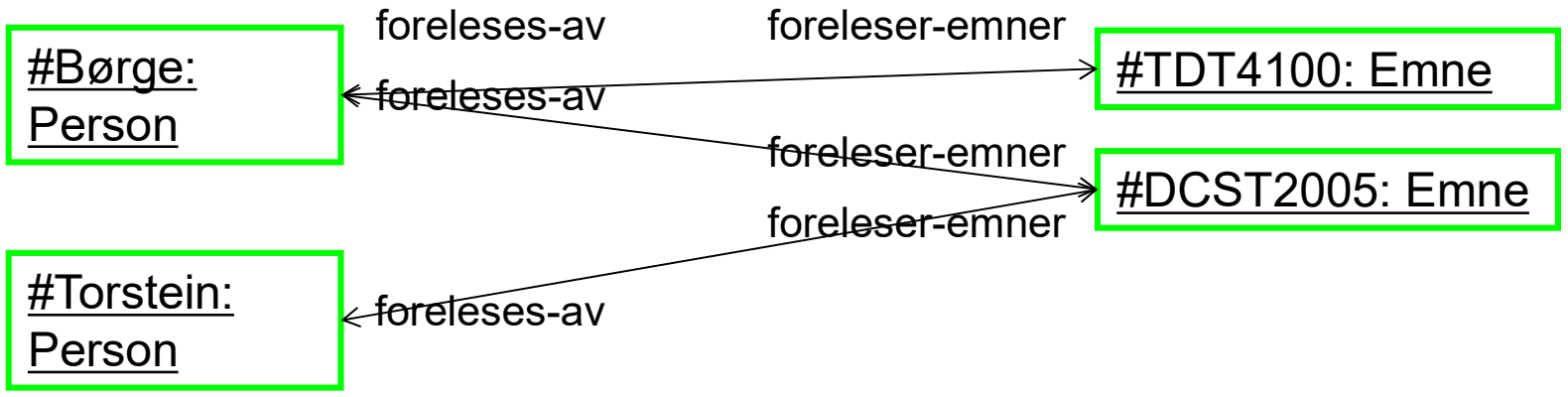
person med 3 epostadresser



Eksempel: n-n

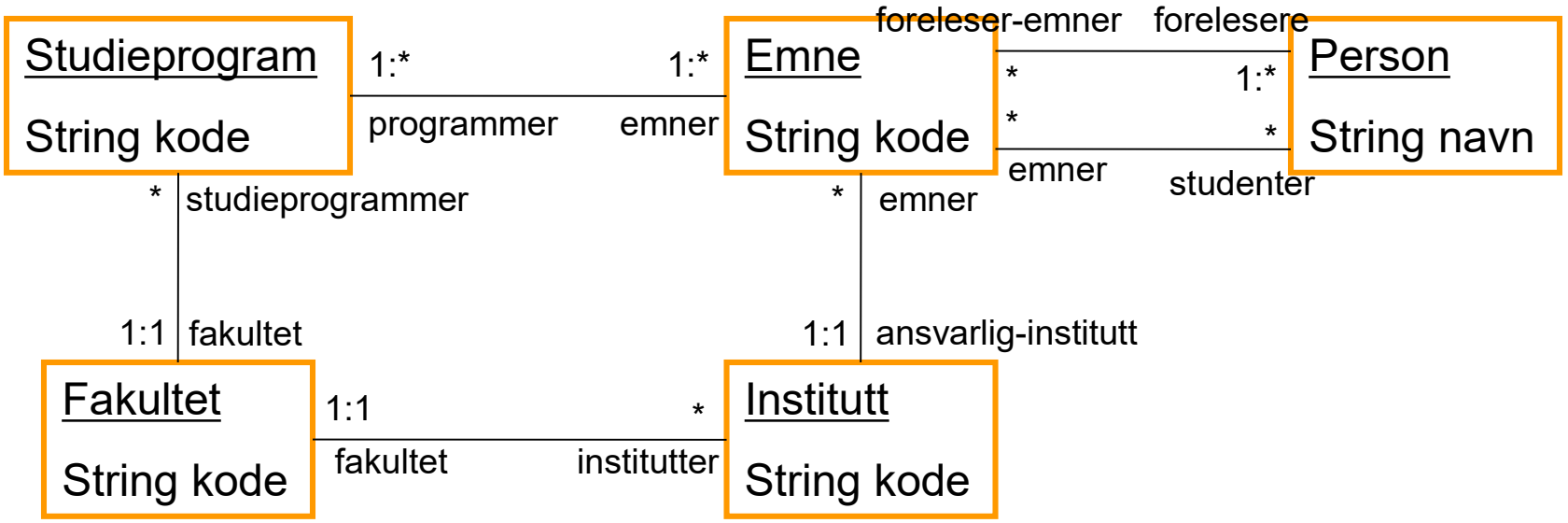


- En instans av Person har ubegrenset antall **foreleser-emner**-koblinger til instanser av Emne.
- En instans av Emne har minst **1** og ubegrenset antall **forelesere**-koblinger til instanser av Person.



Spørsmål om assosiasjoner

- **multiplisitet**: antall koblinger
- **navigerbarhet og roller**: retning og navn på kobling
- **aggregering/komposisjon**: eierskap
  - Aggregering: samling av noe – "har en"-relasjon



Assosiasjoner og koding

- Svarene på spørsmål om...
  - **multiplisitet**: antall koblinger
  - **navigerbarhet og roller**: retning og navn på kobling
  - **aggregering/komposisjon**: eierskap
  - andre assosiasjonsbeskrankninger
- ...styrer i stor grad hvordan klassen kodes
  - **type felt**, f.eks. enkeltverdi vs. List
  - **konstruktør** med eller uten argumenter for initielle verdier
  - **innkapsling**, f.eks. enkel getter vs. getCount og getElement
  - **validering** og håndtering av **konsistens**

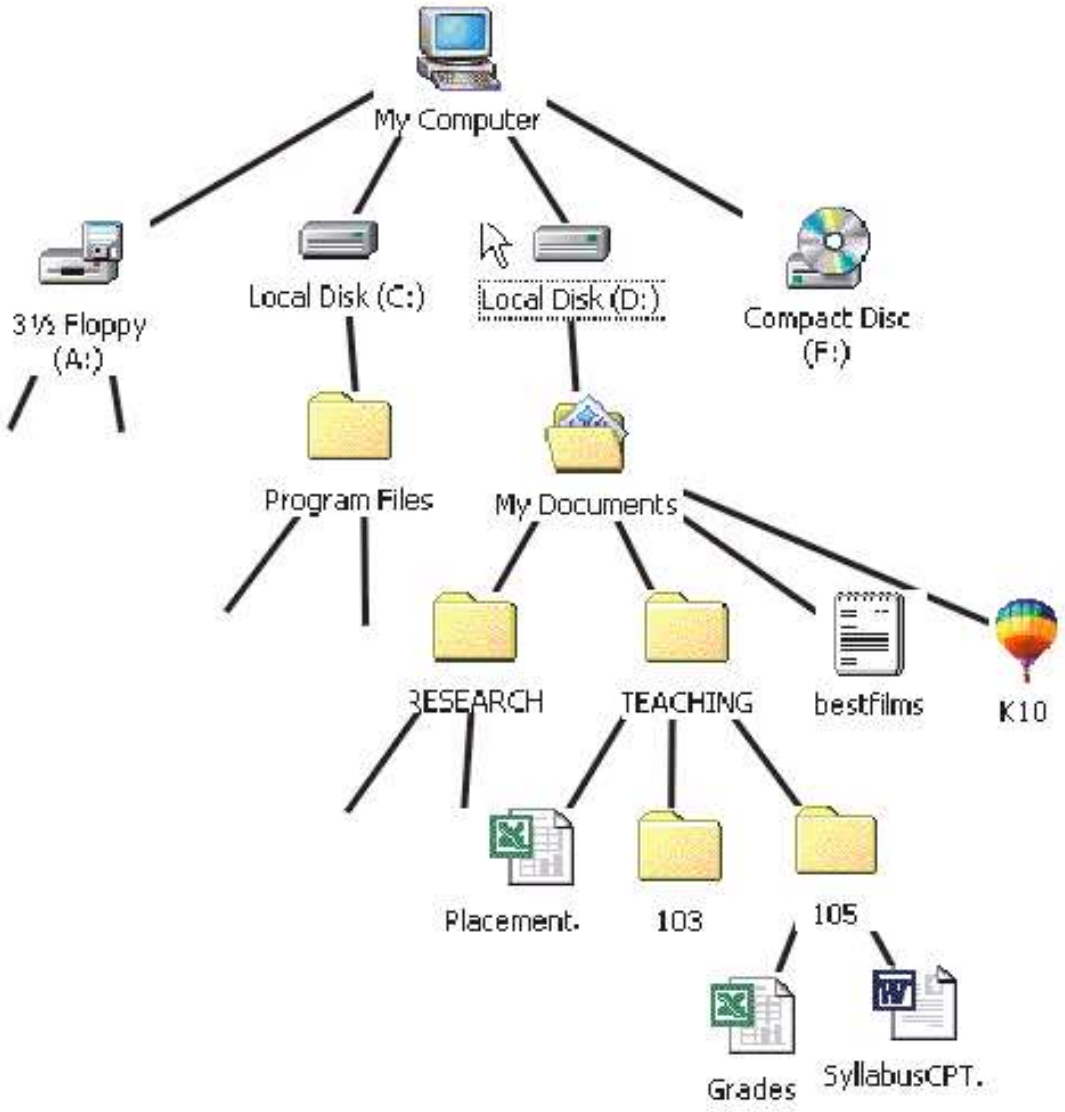
Slutt her du, Børge

- …det under tar du neste uke

Hierarkiske data

- En veldig vanlig form for assosiasjon
  - mappestruktur
  - organisasjonsstruktur
  - familietre
  - grafikk (HTML, JavaFX, OpenGL)
- To viktige aspekter
  - objekt kan kun være inneholdt i ett objekt
  - strukturen er ofte rekursiv, med ukjent antall nivåer
- Litt mer kinkig koding enn ellers…

Eksempel: Mappestruktur



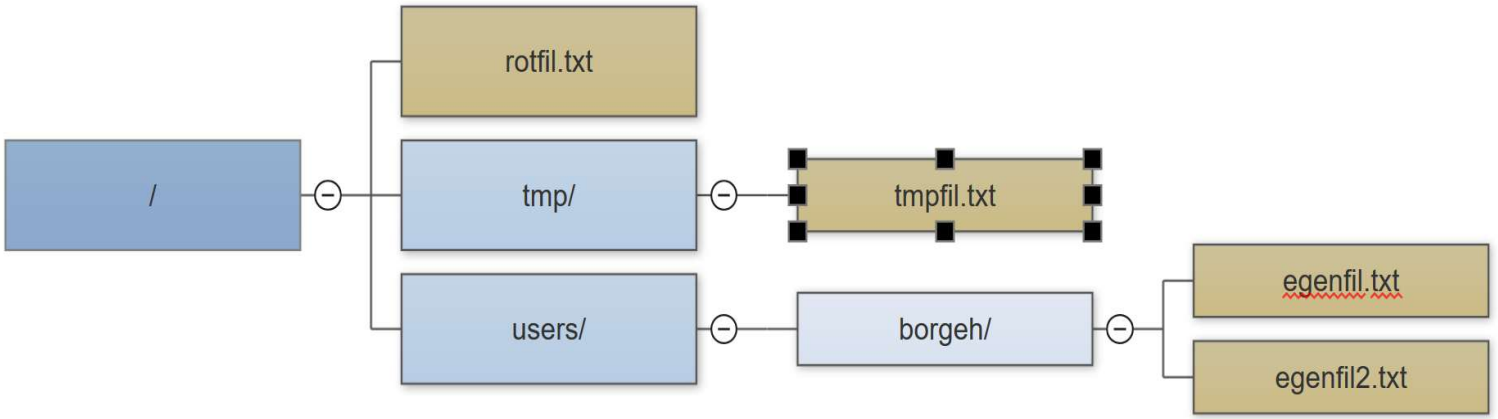
Mapper og filer

- Hvordan (data)modelleres dette?
- Hvilke kodingsvalg har vi?
- Er assosiasjonene
  - (anti)refleksive: A -> A
  - (anti)symmetriske: A -> B => B -> A
  - transitive: A -> B & B -> C => A -> C
- Hva har dette å si for validering?

Hvor var jeg

- Rotfolderen skal bare hete "/" når en spør om navnet dens
- Rotfolderen har ingen parentFolder, vi kan sette den til *null* og så huske dette når vi må.
- Du kan ikke lage en File uten å ha en *reell* Folder med som parameter i konstruktøren.
- Når en fil lages må parentFolder oppdateres med en peker til filen. Hvordan skal det gjøres!
- Alle Folder og File har en Folder parentFolder
- Alle Foldere har en liste med (sub)Foldere og en liste med File
- Man skal kunne flytte en File eller Folder til et annet sted: File.Move(Folder) og Folder.Move(Folder)
- Hva skjer hvis man forsøker å flytte folderen /users inni /users/borgeh?

/rotfil.txt
/tmp
/tmp/tmpfil.txt
/users
/users/borgeh
/users/borgeh/egenfil.txt
/users/borgeh/egenfil2.txt


rotfil.txt
/
tmp/
tmpfil.txt
users/
borgeh/
egenfil.txt
egenfil2.txt

En rask sak - utskrift

- Jeg vil skrive ut hele strukturen:
  - Folder.printContent()
  - Først skrive ut meg selv.
  - Først skrive ut filene i folderen, så kalle printContent på subfoldere. Rekursivt!

Move – hva er så vanskelig med det?

- Hva skjer om en flytter */users/* til */users/borgeh/div/*-folderen?
- Må en gjøre noe med parent til folderen en flytter noe fra?
  - Finnes det spesialtilfelle for dette? (Hint: /)
- Må en gjøre noe med folderen en flytter noe til?

Move – Mål

- Først sjekke om vi unngår sirkulær lenking ved en ny hjelpefunksjon
- Hvis det er lovlig, og vi ikke er på rota, fjern kobling fra tidlig getParent.
- Hvis tilfolder ikke er null, legg den til i ny folder.
- null som mål betyr at en kan fjerne en fil/folder fra treet.

Move – hjelpefunksjon - contains

- Hvis en skal flytte /users til /users/borgeh…
  - Sjekke om borgeh er lik users
  - Så sjekke om parentfolder (users) er lik users
  - Helt til en har kommet til null (rota)
  - Returner false hvis en kommer helt til rota uten å finne den same Folder som en ønsker å flytte, true hvis en finner at det er loop.

Hva med å finne alle filer med et filter?

- Jeg vil lage en metode som går igjennom hele treet, og finner alle filer som slutter på .txt!
- Folder.findAll(null,"txt") gjør susen.
- Folder.findAll("rotfil","txt") finner alle filer som heter *rotfil.txt*.
- Men hvordan går vi frem for å gjøre dette…

findAll - take I

- Sjekke filene inni mappen metoden kalles på først
  - Legge treff inni en samling av treff
- Deretter kalle findAll() i alle subfolderne
- Lage en collection som skal bli med hele veien oppover i treet

findAll() - take II

- matchName(String name, String base, String ext)
- Målet her er å lage en hjelpefunksjon som returnerer sant hvis "fil.txt" testes mot "fil" og "txt".
- Finne ut hvor punktum er plassert
- Lage nameBase og nameExt
- Returnere sammenlikning.