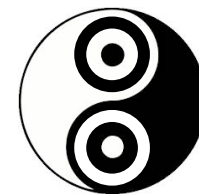


Assosiasjoner og koding

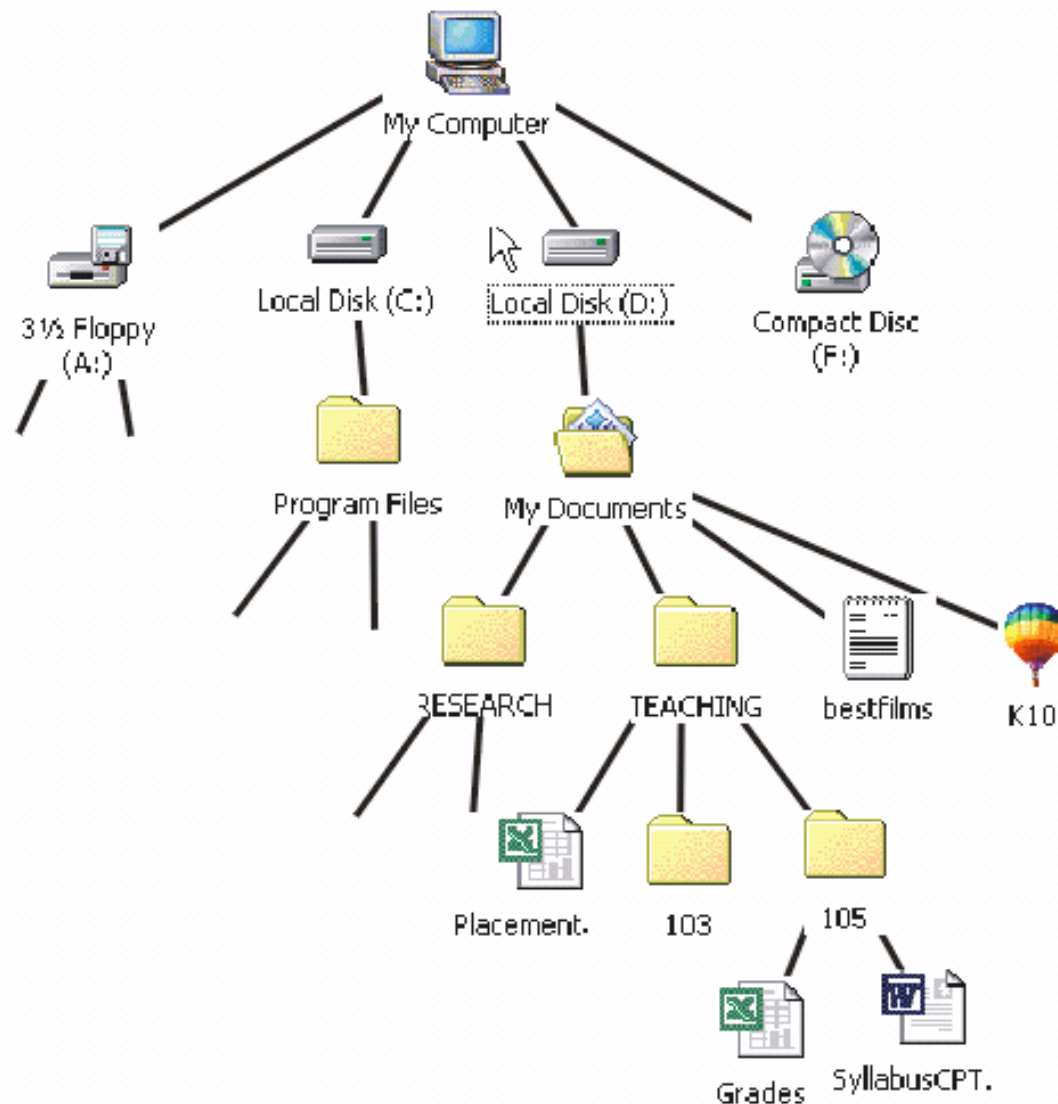


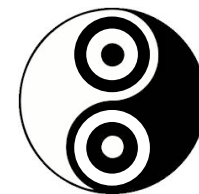
- Svarene på spørsmål om...
 - **multiplisitet**: antall koblinger
 - **navigerbarhet og roller**: retning og navn på kobling
 - **aggregering/komposisjon**: eierskap
 - andre assosiasjonsbeskrankninger
- ...styrer i stor grad hvordan klassen kodes
 - **type felt**, f.eks. enkeltverdi vs. List
 - **konstruktør**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

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: $A \rightarrow A$
 - (anti)symmetriske: $A \rightarrow B \Rightarrow B \rightarrow A$
 - transitive: $A \rightarrow B \ \& \ B \rightarrow C \Rightarrow A \rightarrow C$
- Hva har dette å si for validering?

Egenskaper til fil-hiarkiet

- Rotfolderen skal bare hete "/" når en spør om navnet dens
- Alle Folder og File har en Folder parentFolder, bortsett fra. Rotfolderen har ingen parentFolder. Vi setter denne til *null* og så huske dette når vi må. Vi skal senere se andre løsninger på dette.
- Hvis en Folder har assosiasjon til alle sine barn, så er parent en avledet assosiasjon. Likevel så lagrer vi denne for å gjøre navigering enklere.
- Du kan ikke lage en File eller Folder uten å ha en *reell* Folder hvor den være. Vi skal faktisk be Folder om å lage ny File eller Folder objekt, som så returneres.
- Når en File eller Folder lages må parentFolder oppdateres med en peker til det nye objektet. Faktisk så *eier* Folder alt den inneholder

```

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

```

Egenskaper til fil-hiarkiet

- 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/dok?

/rotfil.txt

/tmp

/tmp/tmpfil.txt

/users

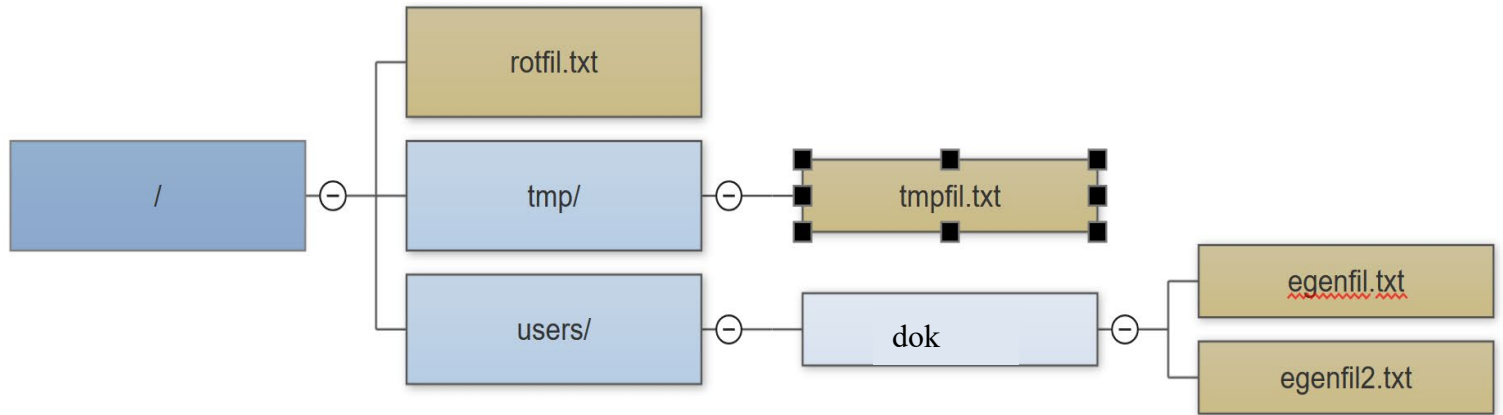
/users/dok

/users/dok/egenfil.txt

/users/dok/egenfil2.txt

I vår implementasjon

- Alle Foldere har en List med (sub)Foldere og en liste med File



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/dok/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/dok...
 - Sjekke om dok 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.