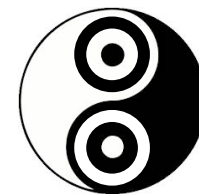


Assosiasjoner og koding

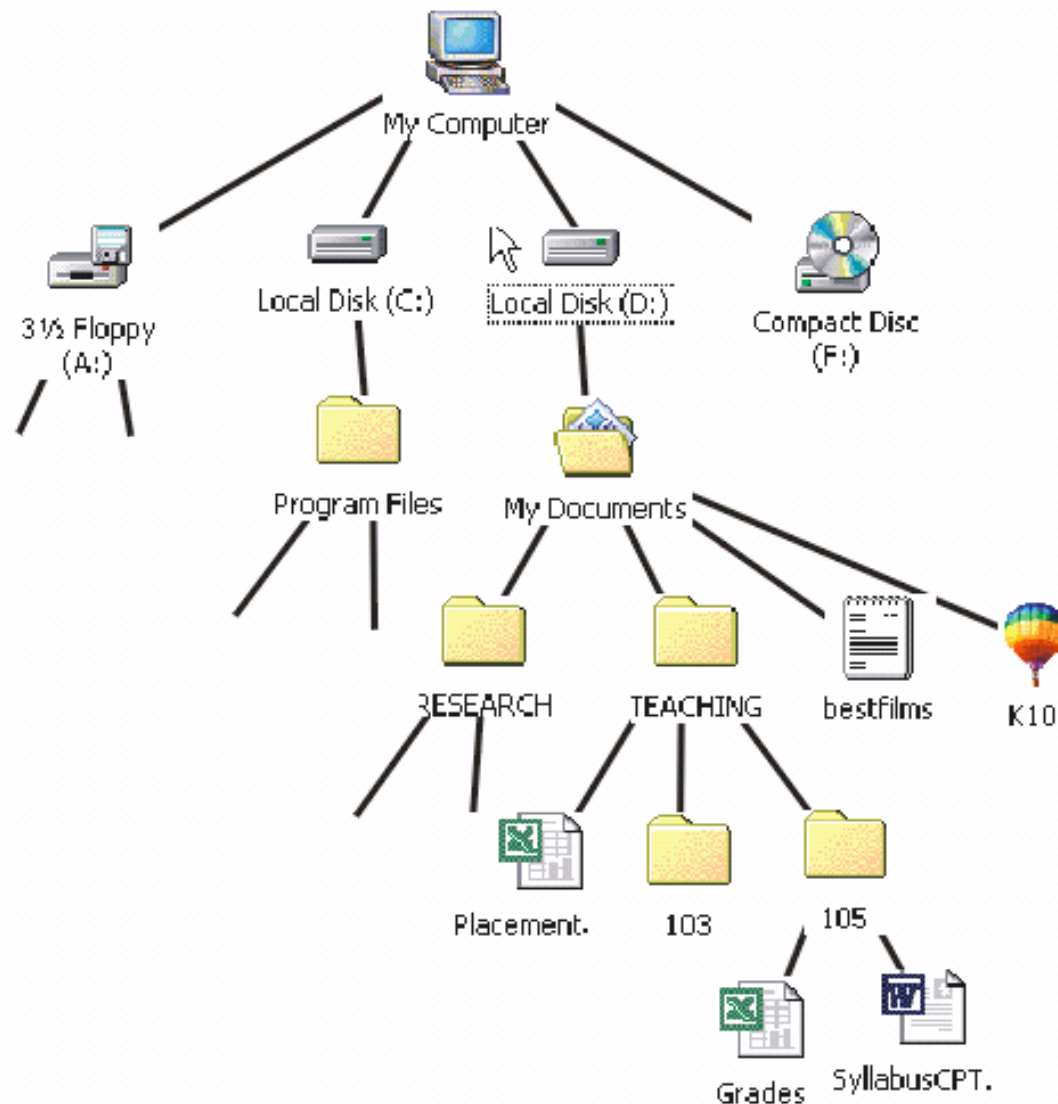


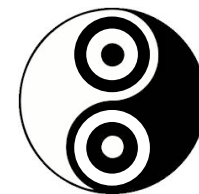
- Svarene på spørsmål om...
 - **multiplisitet**: antall koblinger
 - **navigerbarhet og roller**: retning og navn på kobling
 - **aggregering/komposisjon**: eierskap
 - andre assosiasjonsbeskrankninger
- ...styrer i stor grad hvordan klassen kodes
 - **type felt**, f.eks. enkeltverdi vs. List
 - **konstruktør** 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 *er* 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

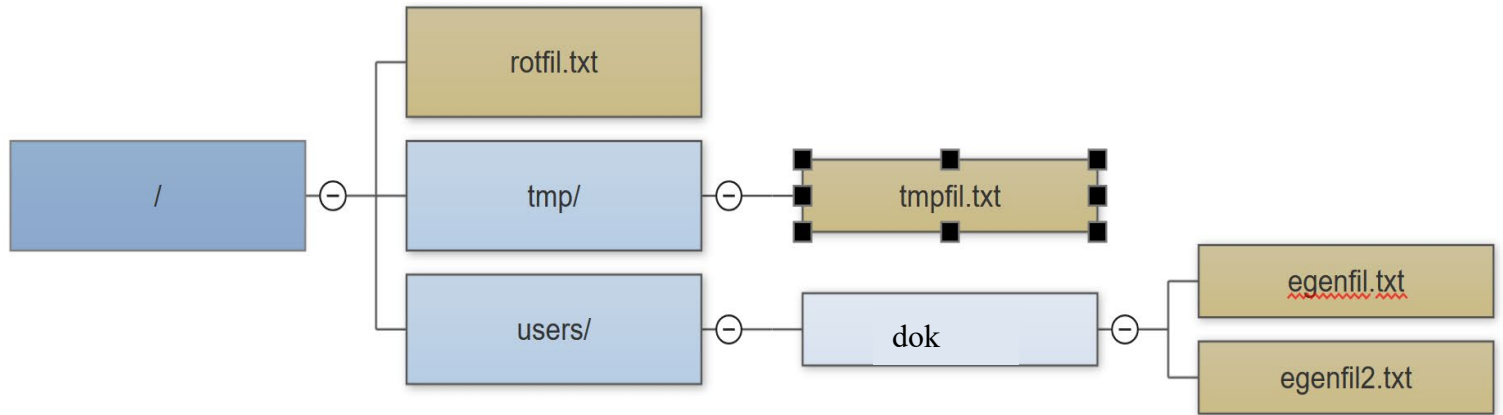
/users/dok

/users/dok/egenfil.txt

/users/dok/egenfil2.txt

I vår implementasjon

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



Denne uka (uke 6)

- Fullføre Folder-eksempel
 - Hva vi har gjort, hva var planen
 - Bruke diagrammer til å analysere
 - Bruke debugger til å analysere
 - Rydde opp
 - Legge til litt manglende funksjonalitet:
 - toString(path), printe hiarkiet,
- Andre eksempler på assosiasjoner
- Motivere og begynne med inheritance (arv) og interface (grensesnitt)

En avklaring om søskenrelasjonen

- Hvis en ikke er sin egen søsken (ikke refleksiv), så kan ikke den være både symmetrisk og transitiv.
- Vi må legge til en betingelse til transitiviteten: (underforstått, disse gjelder for alle valg av A,B og C)

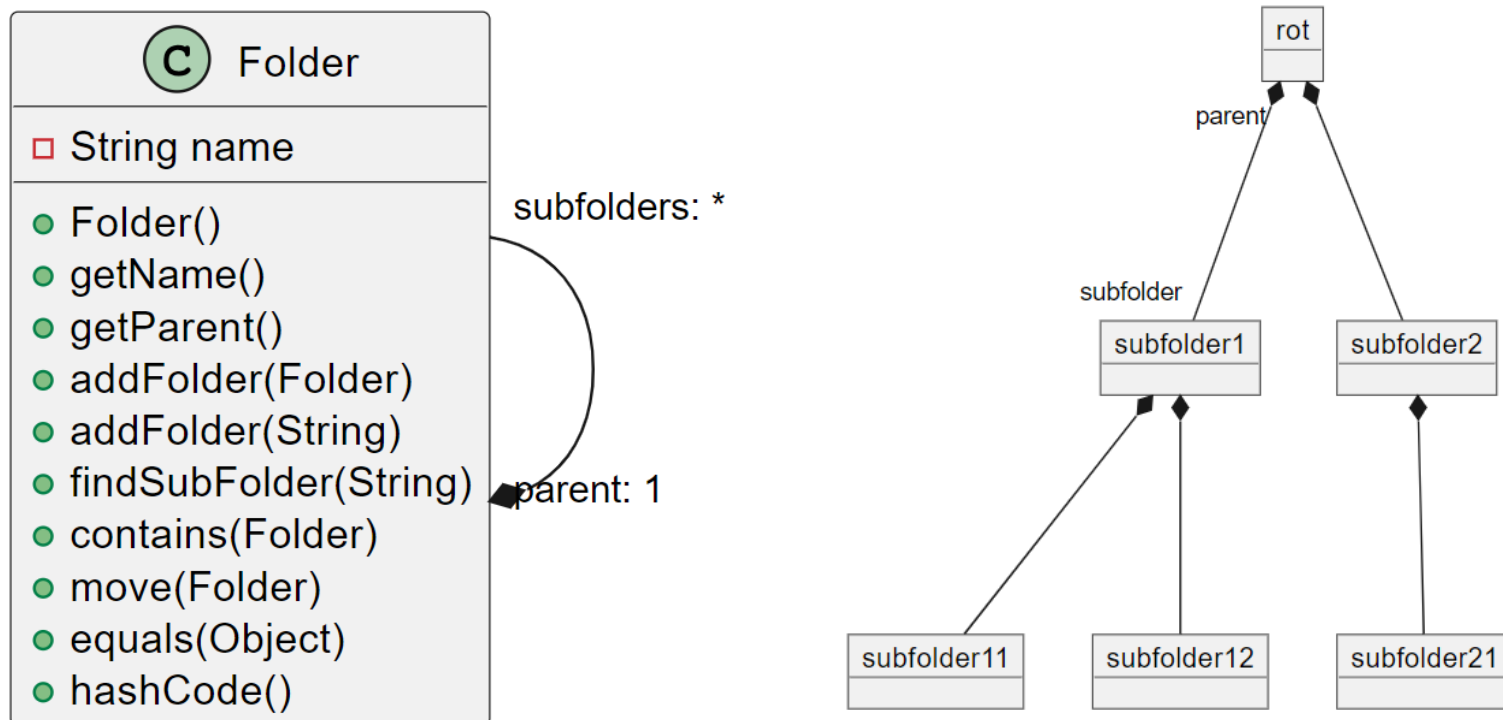
$$A \rightarrow A$$

$$\text{Hvis } A \rightarrow B, \text{ så er } B \rightarrow A$$

$$\text{Hvis } A \rightarrow B, B \rightarrow C \text{ og } A \neq C, \text{ så er } A \rightarrow C$$

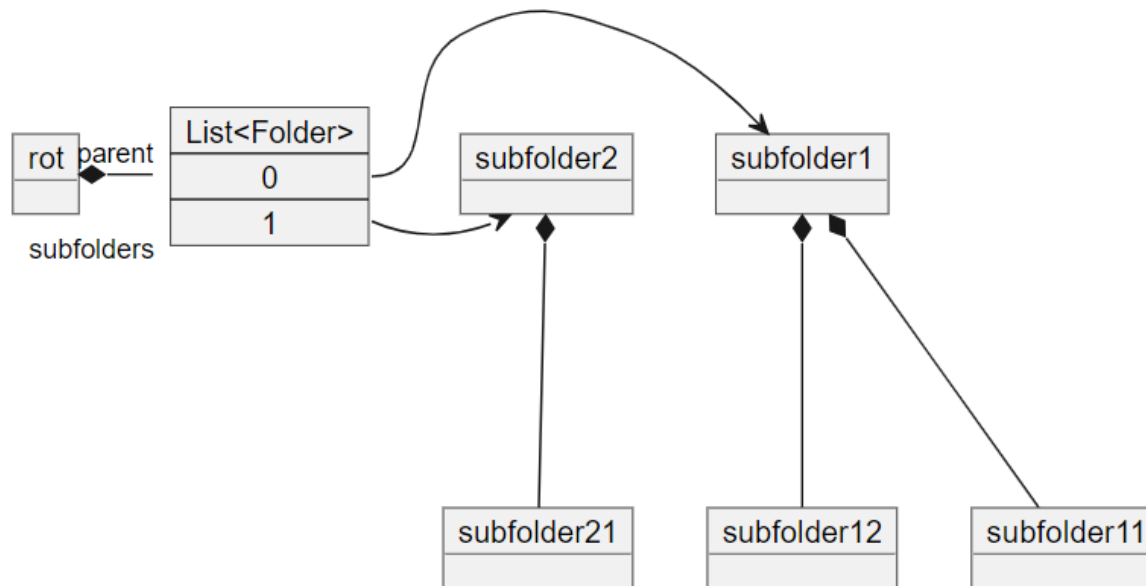
Dette er da ikke en *ekvivalensrelasjon*, men ikke langt unna.

Folder så langt: Klasse og object-diagram



Subfolders er lagret som en Collection /liste, det har vi ikke tegnet inn her

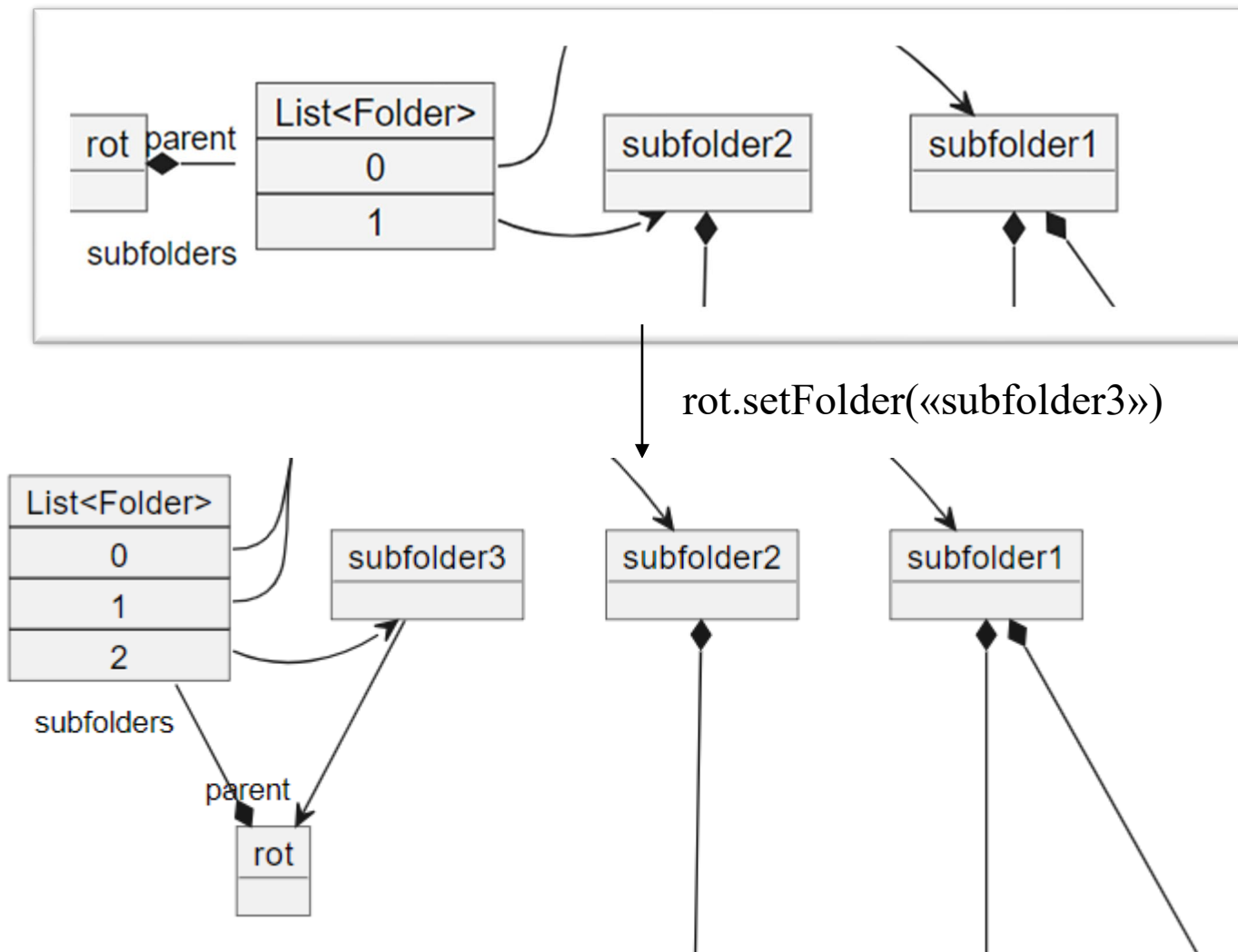
Litt mer detalj



Her viser vi at det er et objekt i rot, som er en liste, som implementerer relasjonen til subfoldrene.

Hva skjer når vi lager ny subfolder?

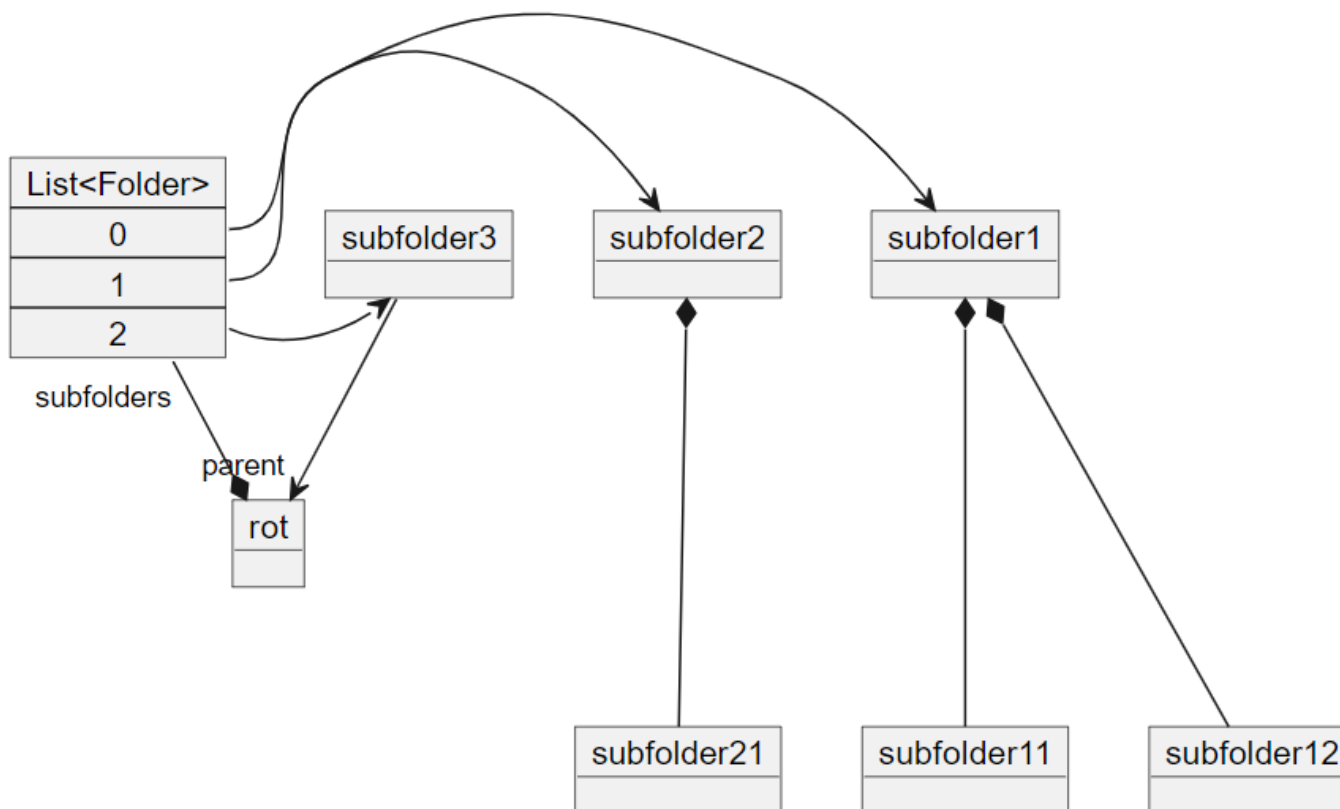
Objekt-tilstandsdiagram



Hva skjer når vi lager ny subfolder?

Objekt diagram etter at vi har laget

(NB! Vi viser bare List delen for rot. De andre folderene har sine egne.)

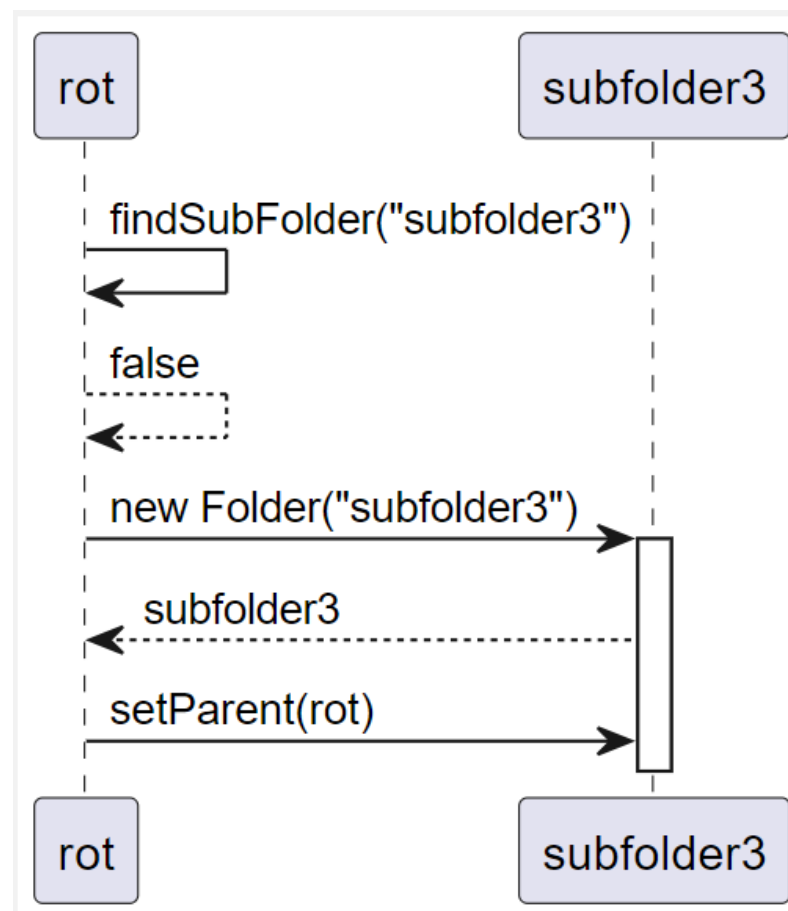


Hva skjer når vi lager ny subfolder?

Sekvensdiagram

Stemmer dette med koden vi har?

La oss sjekke med debuggeren...



Litt refaktorering...

- Vi kan godt returnere subfolder når vi lager subfolder. Da slipper vi å finne den på nytt, og kan jobbe videre...

Hva kan skje med addFolder(Folder)?

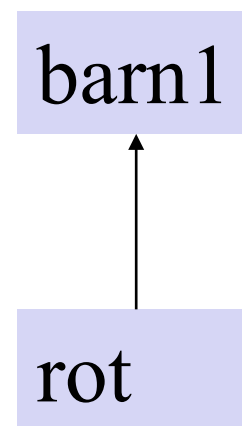
Å kunne opprette Folder-objekter fritt og legg dem til som subfoldere kan ødelegge gyldig tilstand.

Vi kan lett legge til en Folder som inneholder en høyere opp.

Pga. rekkefølgen vi gjør det fanger ikke testen vår dette.

```
Folder root = new Folder("/");
Folder sf1 = new Folder("sf1");
sf1.addFolder(root);
```

```
//Dette bør vi nå ikke få lov til:
root.addFolder(sf1);
```



Innkapsling av Folder

En løsning: Innkapsle
`addFolder(Folder)` ved å gjøre
den *private*.

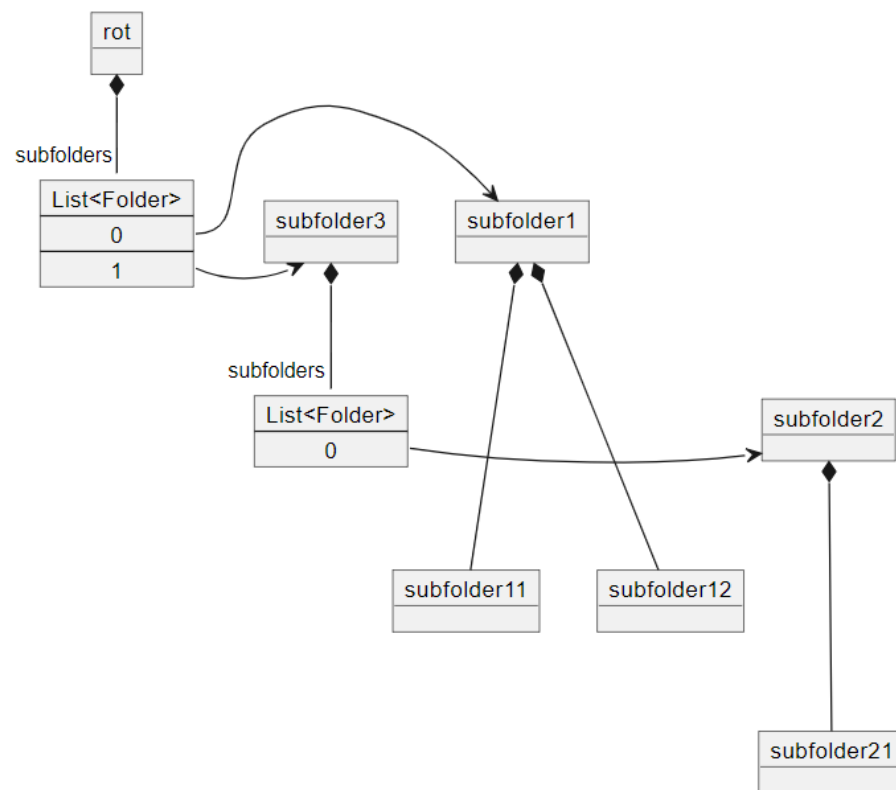
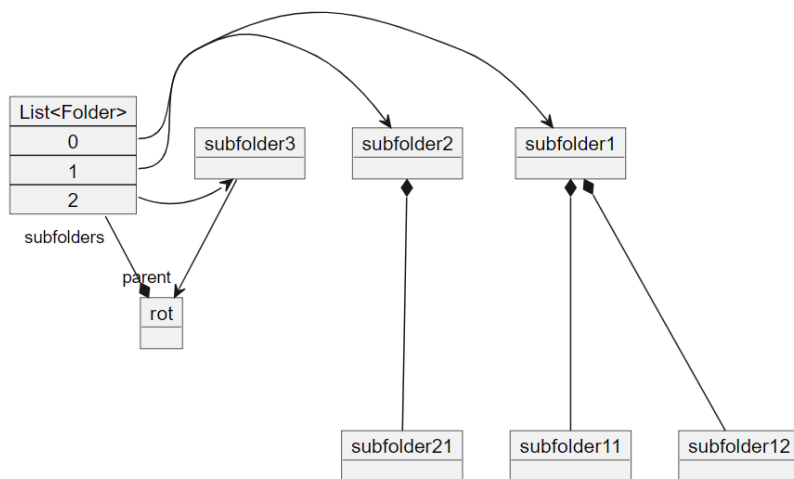
Grensesnittet til `Folder` tillater
bare da at subfoldere opprettes i
konteksten av en eksisterende
foldere.

En “klient” kan da bare lage
rotobjekter.

Hva skjer når vi flytter subfoldere?

Ved å ha kontroll på legge til subfoldere, så skal det holde å sjekke at vi ikke flytter en folder til seg selv eller en subfolder av seg selv.

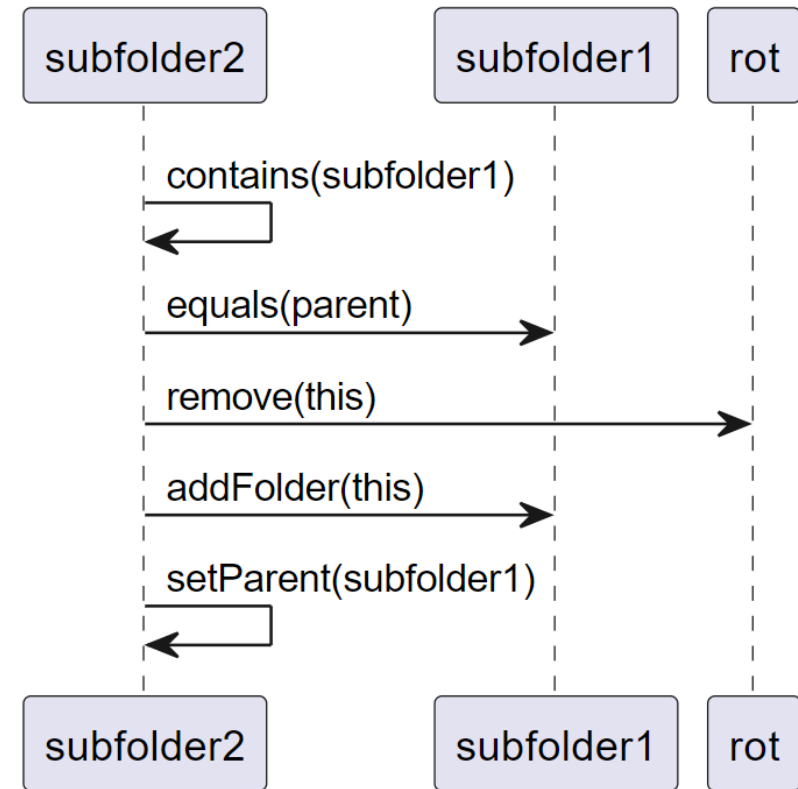
Flytting av folder til parent er lovlig, men krever ingen handling.



Hva skjer når vi flytter subfoldere?

Dette er koden vi har, stemmer den?

```
public boolean move(Folder toOther) {
    if (contains(toOther) ||
        toOther.equals(parent)) {
        return false;
    }
    // Selve flyttingen:
    parent.subfolders.remove(this);
    //potensielt problem: todo: Undersøk
    toOther.addFolder(this);
    //todo: Sjekk at ikke allerede har
    subfolder med samme navn.
    return true;
}
```



Hva skjer når vi flytter subfoldere?

Hvis vi ser på `addFolder(Folder)`, så ser vi at vi må legge til sjekk på at navn ikke finnes fra før.

La oss sjekke i debugger.

Litt refaktorering...

- `contains(Folder)` er litt tvetydig navn. Et mer beskrivende navn for `contains(Folder)` kan f.eks. være `isPredecessorOf`
- Kanskje bedre å utløse unntak hvis det ikke er lov å flytte. La bruker bestemme hvordan feil håndteres.

Bygge sti (path)

- Vi går oppover parent-relasjonen, rekursivt.
 - Stien er implisitt i relasjonen, og ikke eksplisitt lagret
 - Vi returnerer en tekststreng som representerer stien.
 - Dette kan også være toString().

Skrive ut hele strukturen

- 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!
 - Legge til innrykk?

Mer rydding i Folder-klassen

Vi har ikke brukt equals- og hashCode metodene. Disse kan vi droppe I denne versjonen.

Vi har gitt rotmappen navnet “/”, men ser at det blir en liten inkonsistens. Kanskje tom streng I stedet?

Nå har vi nesten glemt filene som skal være i mappene...

- Vi legger til en ny List i hver mappe, som peker til fil-objekter OurFile
- For enkelthets skyld inneholder OurFile bare String som selve data
- For å gjøre håndtering enklere, så har vi også en referanse til parent.

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.

Relasjoner og rettede grafer

Assosiasjoner /koblinger mellom objekter kan vi se på som (matematiske) relasjoner

Flere eksempler på relasjoner

- (To-veis) lenkede lister

- Mappestrukturen vår er et rotet tre.
- Det er et spesialtilfelle at et rettet tre (DAG = Directed acyclic graph)
- Som er et spesialtilfelle av en rettet graf.
- Lenket liste er også ettype rettet graf.

Hva kan være et passende grensesnitt?

Hva må vi gjøre for å ivareta gyldig tilstand?