

TDT4100 uke 6

- Foldere og filer
- Et mer komplekst scenario, med masse godsaker!
- I mappa 'kode' under uke6 er det kode dere kan ta som utgangspunkt. Jeg vil ikke legge inn endringer her, så dere vil ikke få trøbbel med git.

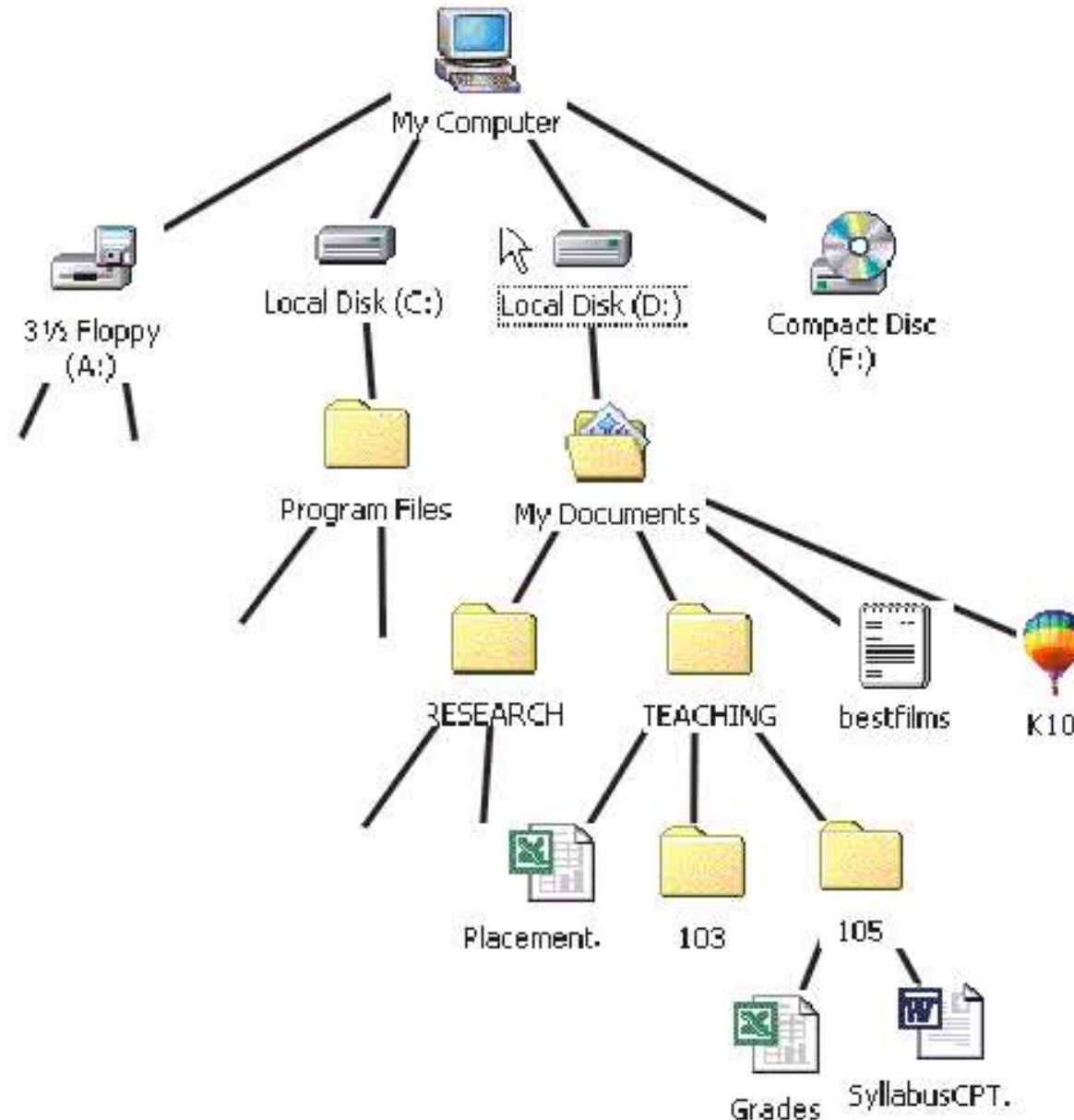
Dagens/ukens mål

- Et mer avansert eksempel
 - Endring én plass må medføre endringer et annet sted
 - Enkelte endringer ulovlige, men hvordan sjekke!
 - Ulike algoritmer for å løse samme problem
 - Objekter av type File må kunne
 - Rekursjon
 - Vi forholder oss for første gang til at File og Folder begge er av type Object (i findFirst(), sannsynligvis neste gang)

Hierarkiske data

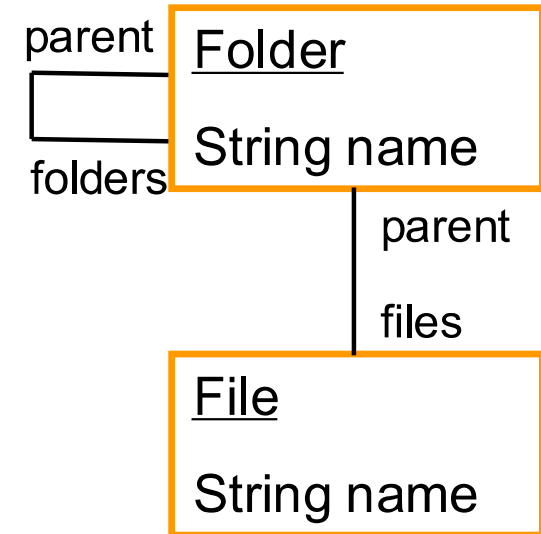
- En veldig vanlig form for assosiasjon
 - mappestruktur
 - organisasjonsstruktur
 - familietre
 - grafikk (HTML, JavaFX, OpenGL)
- To viktige aspekter
 - objekt kan kun være inneholdt i ett objekt
 - strukturen er ofte rekursiv, med ukjent antall nivåer
- Litt mer kinkig koding enn ellers...

Eksempel: Mappestruktur



Eksempel: Mappestruktur

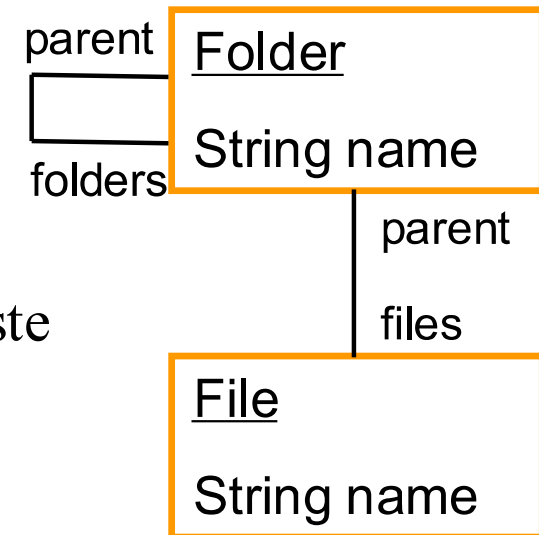
- Struktur
 - Mapper kan inneholder andre mapper
 - Rot-mappa har ingen parent
 - Nederst i hierarkiet har vi filer
- Assosiasjoner og felt
 - Folder:
 - Folder parent
 - List<Folder> folders
 - List<File> files
 - File
 - Folder parent



Eksempel: Mappestruktur

- Konstruktører

- tar parent som argument
- File må ha en parent
- File trenger tilgang til Folder sin files-liste



- Metoder

- toString(): gir path til fil
- move: for å legge til/flytte/fjerne mapper og filer
- contains: sjekker om en fil eller mappe er inni en mappe, uansett nivå
- findFirst og findAll: finn første/alle fil(er) med spesifikt navn (eller mønster)

Opprettelse av elementer

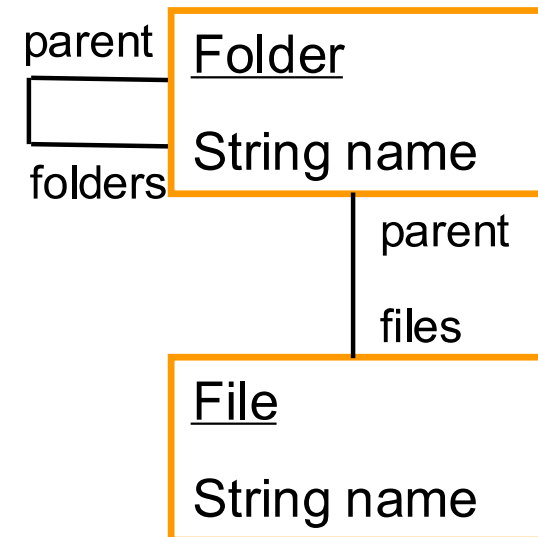
- Konstruktør tar inn navn og parent
 - parent kan være null for mapper, så en kan opprette rot-mappa
 - må legge element til en evt. parent
- Problem
 - Folder sin konstruktør kan legge nytt element til folders-lista
 - File sin konstruktør må be sin parent om å legge seg til files-lista

«Kinkige» metoder

- `File.toString()` og `Folder.toString()` – returnere stien fra rota til this-objektet (File eller Folder), som en sammensetning av navnene med / mellom
- `Folder.contains(Folder)` – sier om mappe-argumentet er inneholdt i this-objektet, uavhengig av antall mapper i mellom, brukes til å unngå sirkularitet
- `Folder.findFirst` og `Folder.findAll` – finner første eller alle mapper og filer med et gitt navn (evt. base og endelse)

Eksempel: Mappestruktur

- `String toString()`
 - bygge opp en stor String, med segment fra hvert nivå
- `Folder.toString()`
 - ta parent sin `toString()` og legg til eget navn + “/” bak
- `File.toString()`
 - ta parent sin `toString()` og legg til eget navn bak



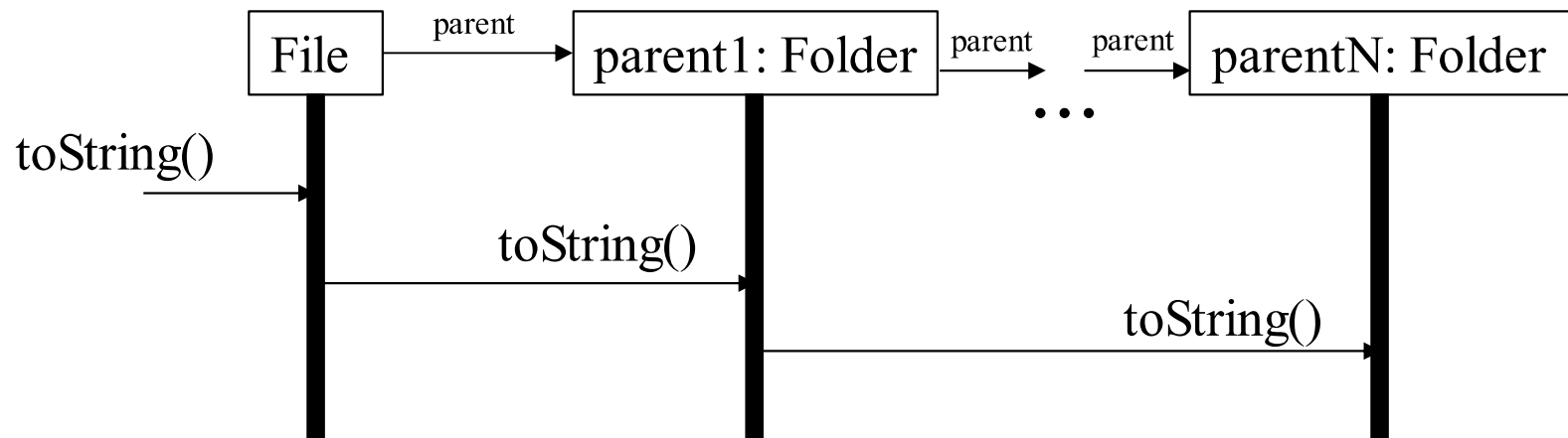
toString()

- Iterativ løsning

- bygg opp en **String** bakenifra, ved å flytte en peker oppover i hierarkiet og hele tiden legge til mapp-navn og «/» foran

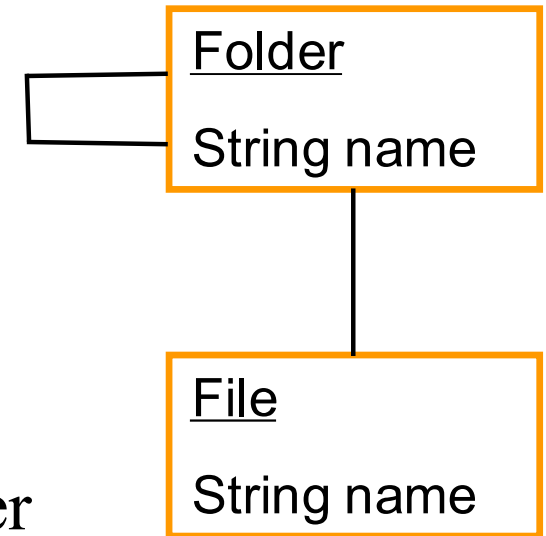
- Rekursiv løsning

- be om **toString()** fra mappa **this**-objektet er inneholdt i (hvis ikke **this**-objektet er rota) og legg til «/» og **this**-objektets eget navn



Eksempel: Mappestruktur

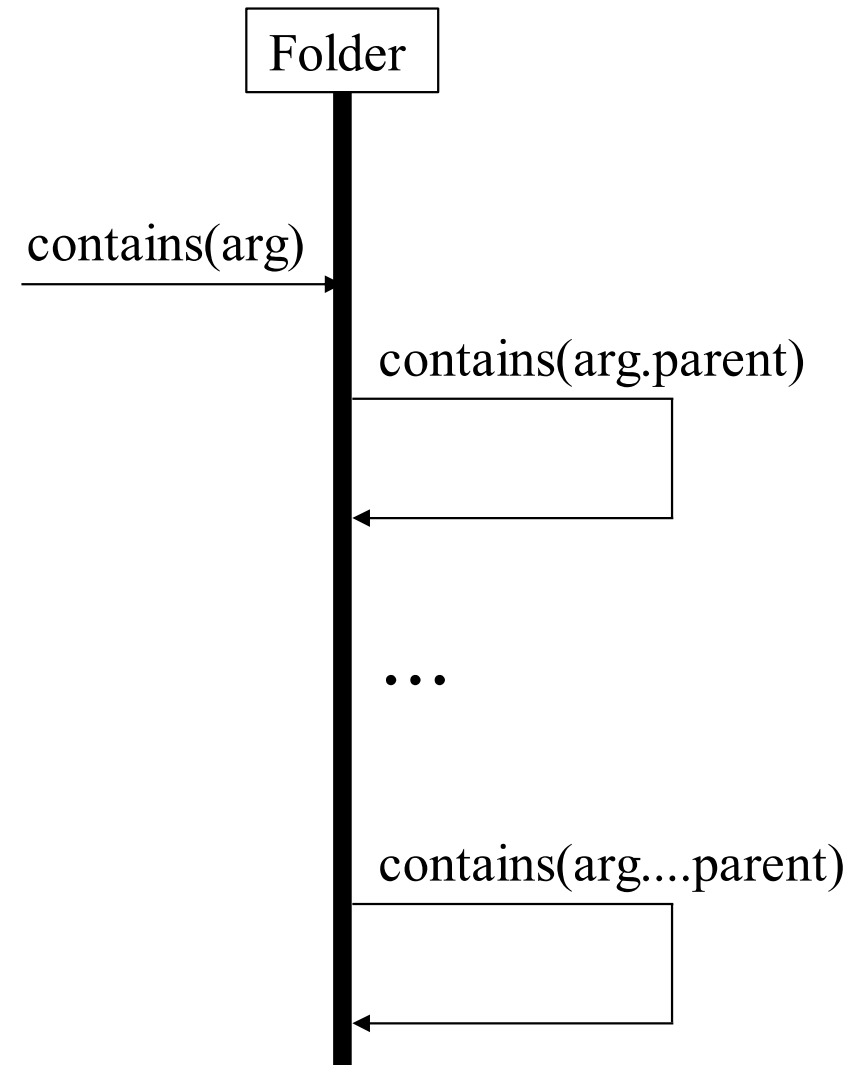
- `void move(Folder target)`
 - tre varianter i ett: add/move/remove
 - må unngå at fil eller mappe er i to mapper samtidig
 - må sikre konsistens mellom koblinger begge veier



- Se <https://www.ntnu.no/wiki/display/tdt4100/Koding+av+1-n-assosiasjoner>

Folder.contains(Folder)

- Iterativ løsning
 - naviger fra argumentet og oppover i hierarkiet inntil en kommer til rota
 - hvis en treffer på **this**-objektet underveis så returneres **true**, ellers blir resultatet **false**
- Rekursiv løsning
 - returner **true** hvis argumentet er samme objekt som **this**
 - hvis argumentet er rota, returner **false**, ellers kall rekursivt med argumentets **parent**-mappe



findFirst og findAll

- Iterativ løsning
 - unaturlig, en må søke nedover og i bredden og trenger en stack
- Rekursiv løsning
 - en må gå gjennom under-mappene og filen og sjekke navnene
 - hvis en må søke videre, så kall rekursivt på under-mappene
 - **findFirst** – returner så snart en finner én match
 - **findAll** – samle opp alle matchende mapper og filer
 - trenger en hjelpemetode for å samle resultatet i en og samme List
 - **findAll(String, List<Object>)** // Object, siden en har både File og Folder

Folder.findFirst

```

public File findFirst(String base, String ext) {
    // search among files
    for (File file : files) {
        String fileName = file.getName();
        if (matchesFileName(base, ext, fileName)) {
            return file;
        }
    }
    // recurse into sub-folders
    for (Folder folder : folders) {
        File file = folder.findFirst(base, ext);
        if (file != null) {
            return file;
        }
    }
    return null;
}

// helper method for matching file names
static boolean matchesFileName(String base, String ext, String fileName) {
    return (base == null || fileName.startsWith(base + ".")) &&
        (ext == null || fileName.endsWith("." + ext));
}

```

Folder.findAll

```

public List<File> findAll(String base, String ext) {
    List<File> result = new ArrayList<>();
    findAll(base, ext, result);
    return result;
}

private void findAll(String base, String ext, List<File> result) {
    // add all files matching base and ext
    for (File file : files) {
        String fileName = file.getName();
        if (matchesFileName(base, ext, fileName)) {
            result.add(file);
        }
    }
    // recurse into sub-folders
    for (Folder folder : folders) {
        folder.findAllHelper(base, ext, result);
    }
}

```