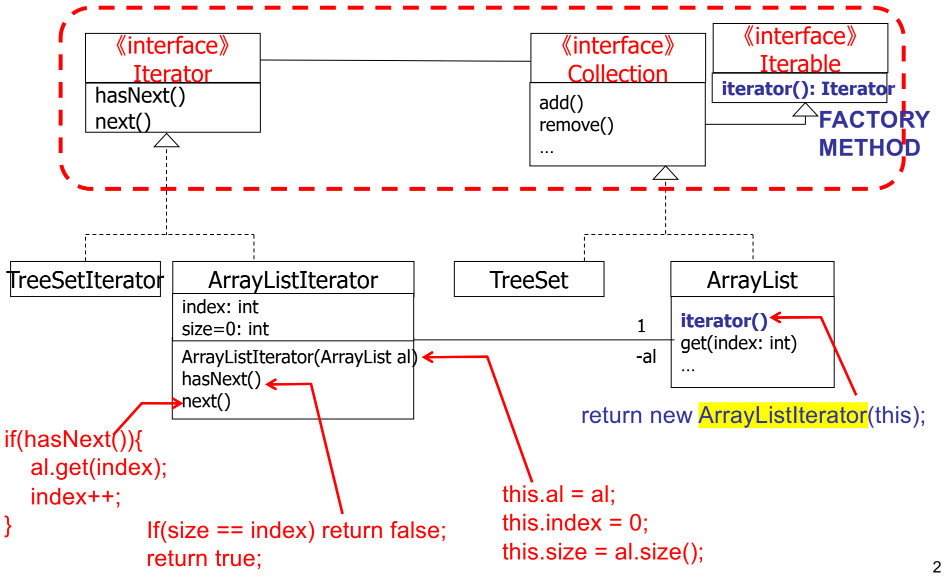
Recap

```
•  Stack<String> collection = new Stack<String>();
   ...
   java.util.Iterator<String> iterator = collection.iterator();
     // Get an iterator.
     // Iterator is an interface.
   while ( iterator.hasNext() ) {
        String o = iterator.next();
        System.out.print( o );}

•  ArrayList<Integer> collection = new ArrayList<Integer>();
   ...
   java.util.Iterator<Integer> iterator = collection.iterator();
   while ( iterator.hasNext() ) {
        Integer o = iterator.next();
        System.out.print( o ); }
```

iterator() is a Factory Method



return new ArrayListIterator(this);

```
if(hasNext()){
   al.get(index);
   index++;
}
```

If(size == index) return false;
return true;

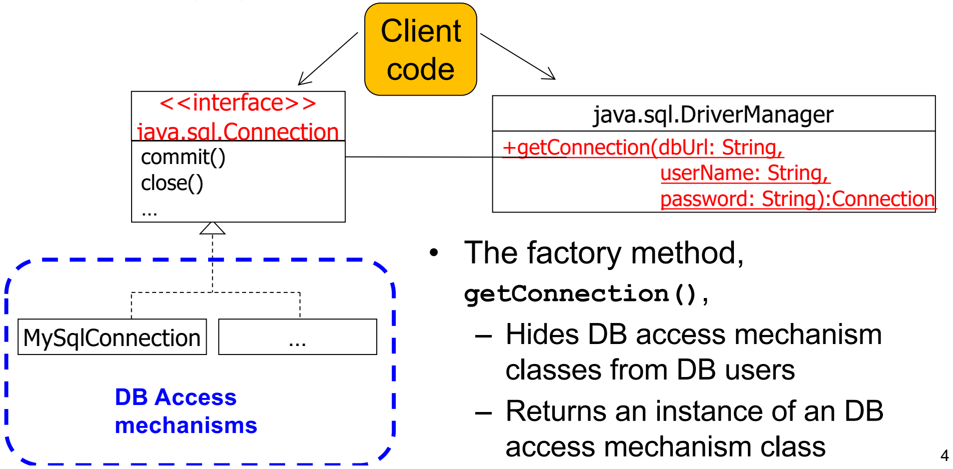
this.al = al;
this.index = 0;
this.size = al.size();

What's the Point?

- The factory method, **`iterator()`**,
  - Hides access mechanism classes from collection users
  - Returns an instance of an access mechanism class
    - e.g., ArrayListIterator

A Similar Example:
`DriverManager.getConnection()` in JDBC API

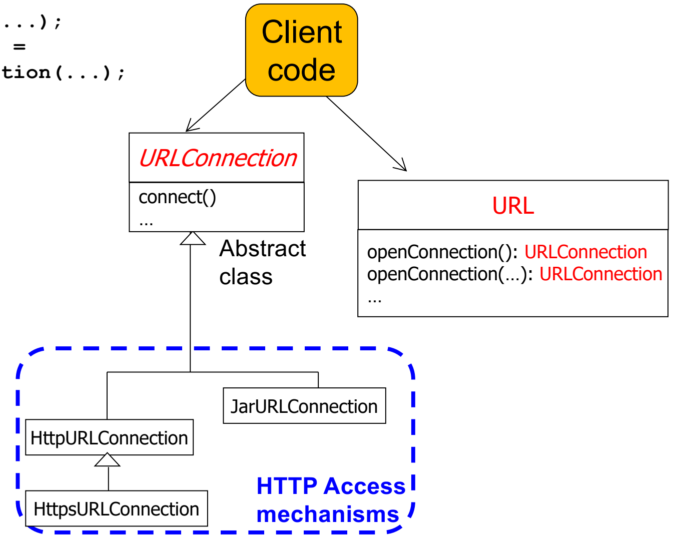
```
•  Connection conn =
     DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb",
                                 "johnDoe", "abcd");
   conn.commit(...);
```



- The factory method, **`getConnection()`**,
  - Hides DB access mechanism classes from DB users
  - Returns an instance of an DB access mechanism class

Another Example: URL and URLConnection in Java API

```
•  URL url = new URL(...);
   URLConnection conn =
       url.openConnection(...);
   conn.connect(...);
```

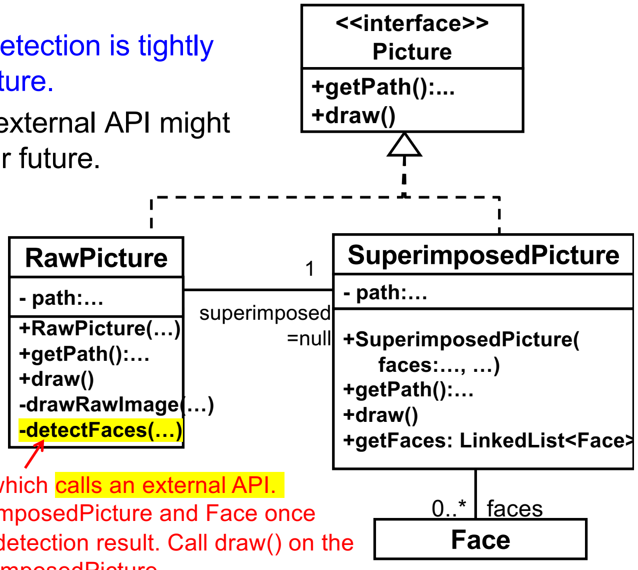


Misnamed?

- *Iterator* might have been misnamed
  - This design pattern's key rationale/benefit (i.e., hiding of access mechanisms) is not limited to the development of iterators.
- Alternative names
  - *Abstract access mechanism*?
  - *Pluggable driver*??
  - *Glue*???

Recap: Face Detection with *Proxy*

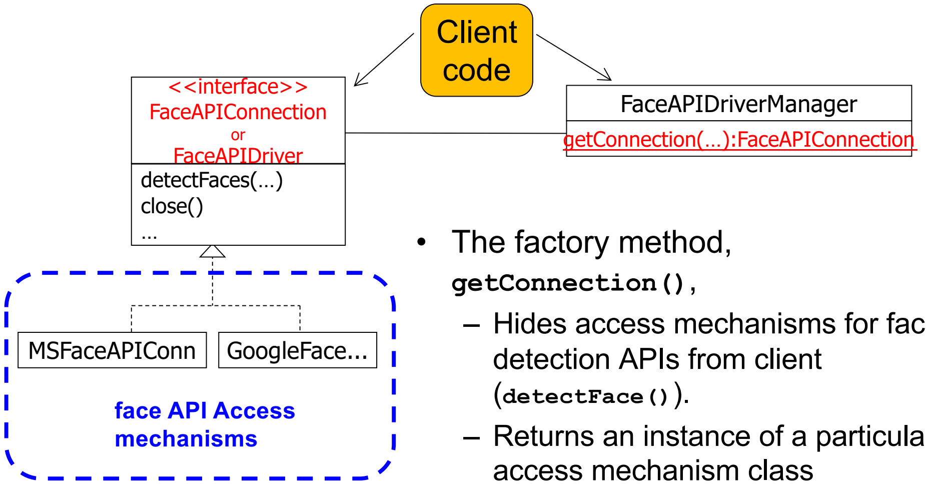
- An API call for face detection is tightly coupled with RawPicture.
  - The choice of an external API might change in the near future.



Create a thread, which calls an external API. Instantiate SuperimposedPicture and Face once the API returns a detection result. Call draw() on the instance of SuperimposedPicture.

- Have `detectFace()` obtain an access mechanism to a face detection API based on *Iterator*-inspired design.

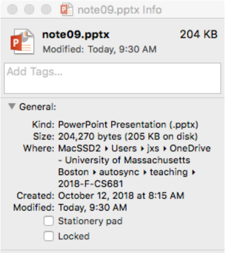
```
•  FaceAPIConnection conn =
     FaceAPIDriverManager.getConnection(...);
   conn.detectFaces(...);
```



- The factory method, `getConnection()`,
  - Hides access mechanisms for face detection APIs from client (`detectFace()`).
  - Returns an instance of a particular access mechanism class

Recap: Metadata Mgt in File Systems

- File system
  - Stores data as files in a structured way
  - Retrieves those data (files)
- Data to be stored
  - Data itself (file content)
  - Metadata (information about the data/file)
    - File name
    - Physical file location in a disk
    - Logical file location (i.e., file path)
    - File size
    - File owner
    - File creation time, last-modified time (the time that the file was last modified), last-backed-up time
    - Access permission


note09.pptx Info

note09.pptx 204 KB
Modified: Today, 9:30 AM

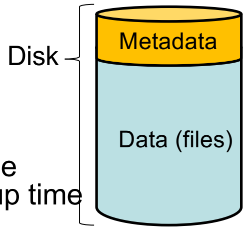
Add Tags...

General:

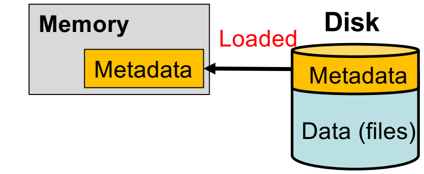
Kind: PowerPoint Presentation (.pptx)
Size: 204,270 bytes (205 KB on disk)
Where: MacSSD2 ▸ Users ▸ jxs ▸ OneDrive - University of Massachusetts Boston ▸ autosync ▸ teaching ▸ 2018-F-CS681
Created: October 12, 2018 at 8:15 AM
Modified: Today, 9:30 AM

□ Stationery pad

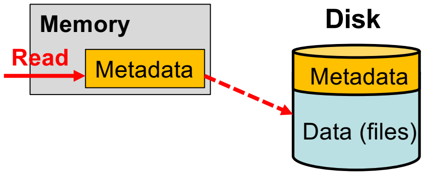
□ Locked

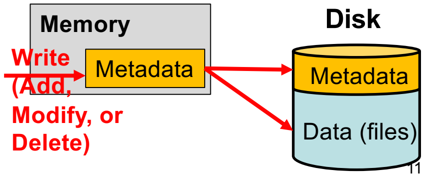


- During its bootup process, an OS loads metadata to the main memory.

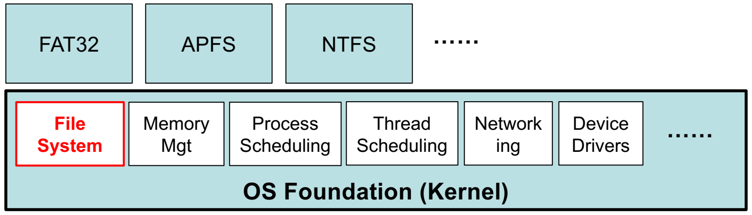


- Applications access (read and write) files through their metadata.
  - Read a file's metadata
  - Read a file's content
  - Add/store a new file
  - Modify a file's metadata and/or content.
  - Delete a file.

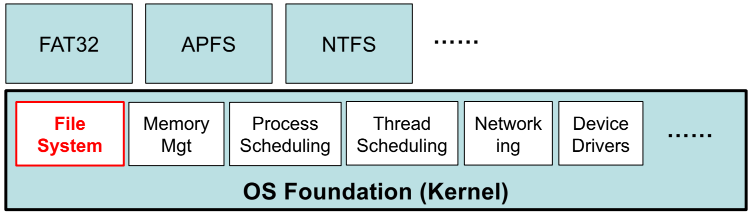




- An OS supports different types of file systems.
  - Mac: APFS, HFS+, HFS, NTFS, FAT32, NFS, etc.
  - Linux: ext4, ext3, ext2, FAT32, NTFS, XFS, etc.
  - Windows: NTFS, ReFS, exFAT, FAT32, etc.

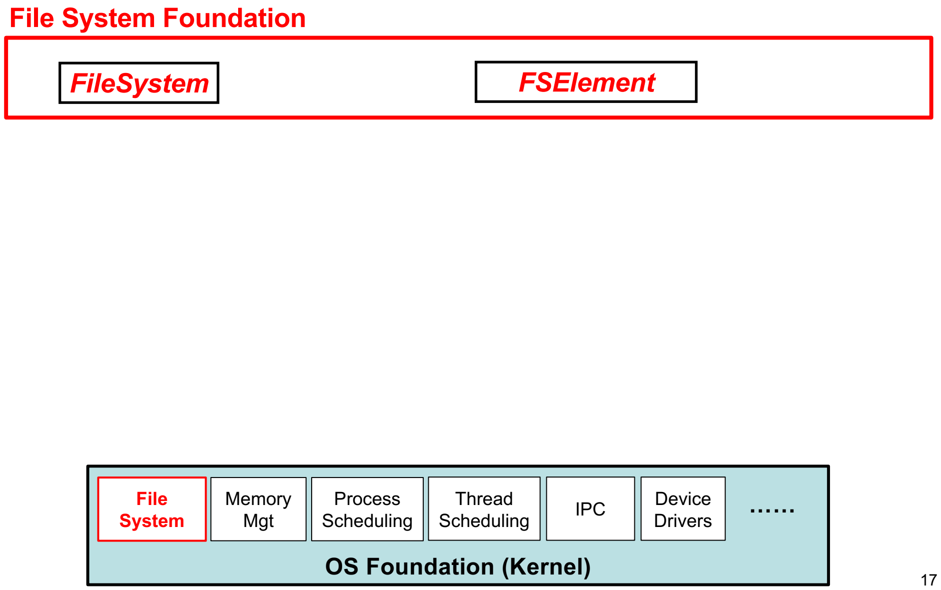


- File system foundation API
  - implements common metadata and common initialization procedure across different file systems.
    - so that the development of a file system can be quicker and more cost-effective.

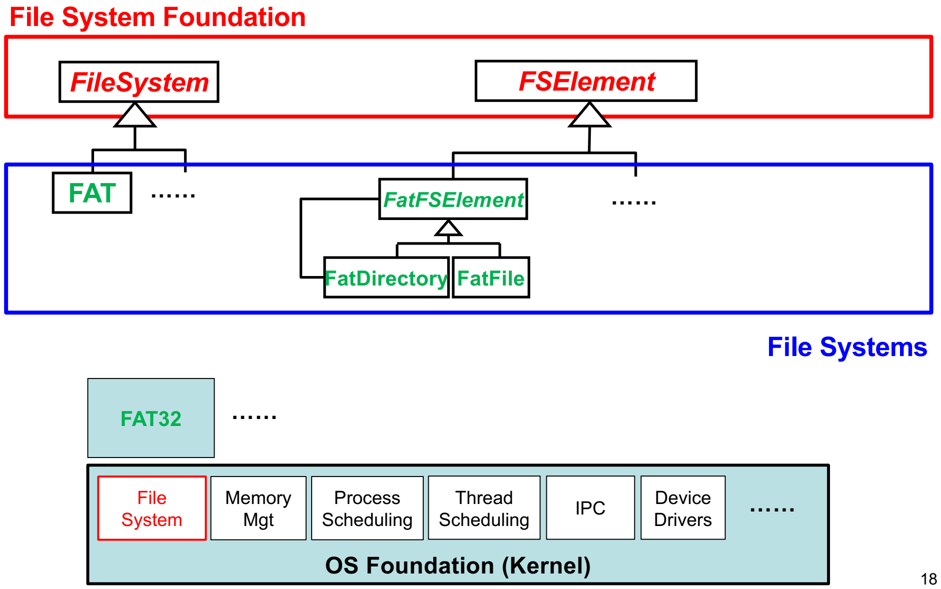


- Common metadata in FAT32, NTFS and APFS
  - Name, size and creation time
- FAT32
  - Name: up to 11 characters (8+3 format), case insensitive,
  - Multiple trees (drives)
  - No links allowed
- NTFS
  - Name: up to 255 chars, case sensitive
  - Extra metadata: Owner’s name, last-modified timestamp
  - Single tree
  - Links allowed
- APFS
  - Name: up to 255 chars, case sensitive
  - Extra metadata: Owner’s name, last-modified timestamp, checksum
  - Single tree
  - Links allowed

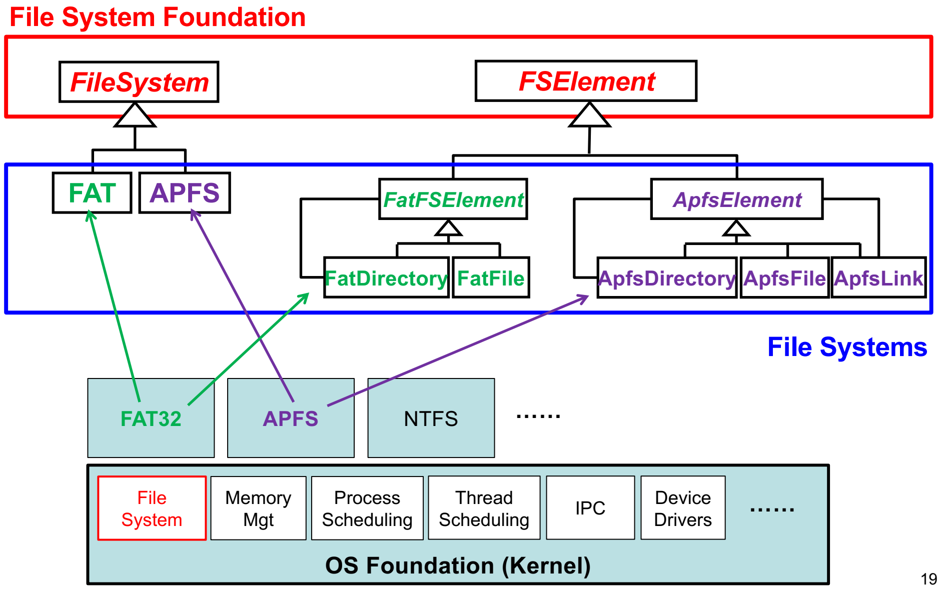
Decoupling the FS Foundation and Individual File Systems



Decoupling the FS Foundation and Individual File Systems



Decoupling the FS Foundation and Individual File Systems



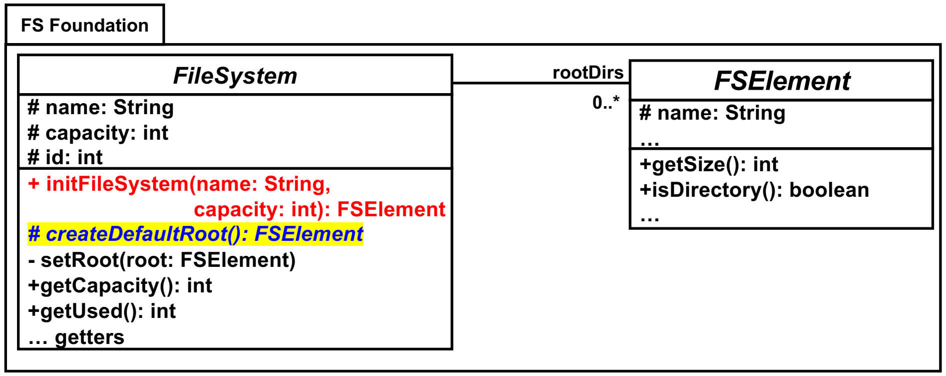
Common Procedure to Initialize a File System

- There is a *common procedure* to initialize a file system.
  - Create a file system
  - Initialize the file system by setting its metadata
    - e.g., FS's name, FS' capacity (total disk space), FS's unique ID
  - Create the default root directory
  - Initialize the default root directory by setting its metadata
    - e.g., name, size, creation time

- How can we implement the *common procedure* at the foundation layer (i.e., with `FileSystem` and `FSElement`) without knowing `FileSystem`'s and `FSElement`'s subclasses?

- *Factory Method* is well-applicable.

Solve this Design Issue with *Factory Method*

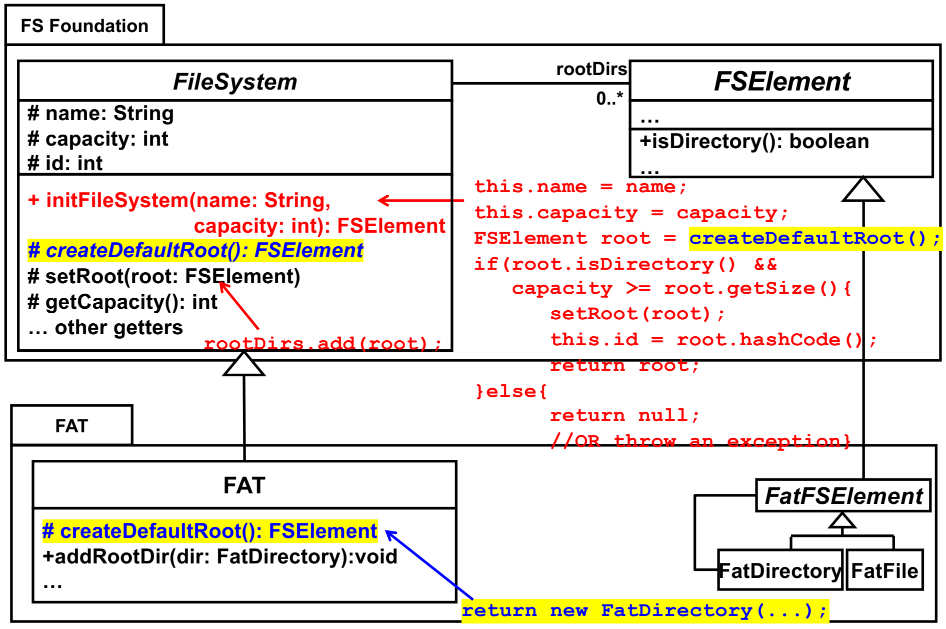


FS Foundation

FileSystem
# name: String
# capacity: int
# id: int
+ initFileSystem(name: String, capacity: int): FSElement
# createDefaultRoot(): FSElement
- setRoot(root: FSElement)
+getCapacity(): int
+getUsed(): int
… getters

rootDirs 0..*

FSElement
# name: String
…
+getSize(): int
+isDirectory(): boolean
…



FS Foundation

FileSystem
# name: String
# capacity: int
# id: int
+ initFileSystem(name: String, capacity: int): FSElement
# createDefaultRoot(): FSElement
# setRoot(root: FSElement)
# getCapacity(): int
… other getters

rootDirs 0..*

FSElement
…
+isDirectory(): boolean
…

```
this.name = name;
this.capacity = capacity;
FSElement root = createDefaultRoot();
if(root.isDirectory() &&
   capacity >= root.getSize(){
      setRoot(root);
      this.id = root.hashCode();
      return root;
}else{
      return null;
      //OR throw an exception}
```

```
rootDirs.add(root);
```

FAT

FAT
# createDefaultRoot(): FSElement
+addRootDir(dir: FatDirectory):void
…

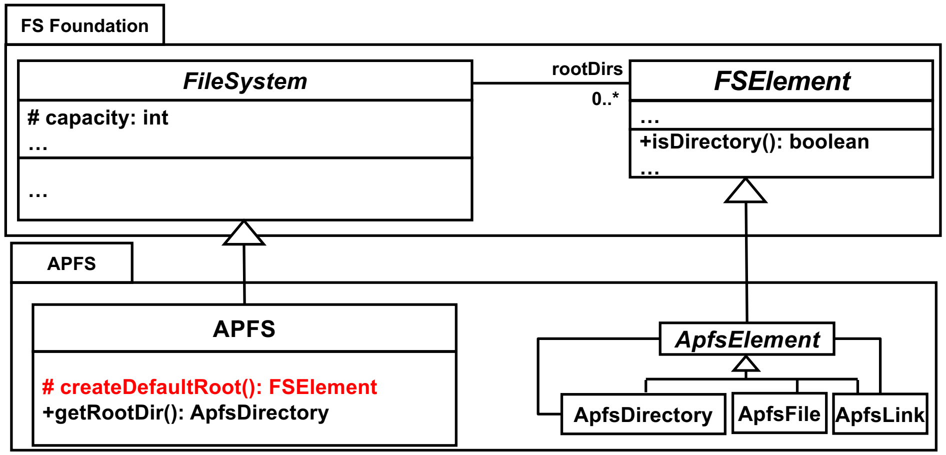
FatFSElement
FatDirectory
FatFile

```
return new FatDirectory(...);
```

```
FAT fat = new FAT(...);
fat.initFileSystem(...);
```

HW 8

- Implement APFS with *Factory Method*, *Composite* and *Proxy*.



FS Foundation

FileSystem
# capacity: int
…
…

rootDirs 0..*

FSElement
…
+isDirectory(): boolean
…

APFS

APFS
# createDefaultRoot(): FSElement
+getRootDir(): ApfsDirectory

ApfsElement
ApfsDirectory
ApfsFile
ApfsLink

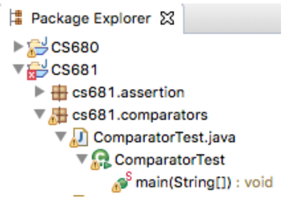
```
APFS apfs = new APFS(...);
apfs.initFileSystem(...);
```

- Revise FileSystem
  - With *Factory Method*
- Implement APFS as a subclass of FileSystem
  - [OPTIONAL] Implement it as a *Singleton* class.
- Separate the original FSElement to
  - Revised FSElement
    - Put all the data fields and methods that are common across different file systems; e.g., size, name, getName(), isDirectory(), etc. etc.
  - Add ApfsElement
    - Put getChildren(), etc.
    - Define extra metadata as its data fields
      - Owner's name and last-modified timestamp
- Rename Directory, File and Link to ApfsDirectory, ApfsFile and ApfsLink, respectively, and make any necessary changes.

- Deadline: Nov 14 (Thu)
- [OPTIONAL] Implement FAT as well.
  - FAT
    - Define FatFSElement, FatDirectory, FatFile
    - Name: up to 11 characters (8+3 format), case insensitive,
    - Multiple trees (drives)
    - No links allowed
  - APFS
    - Name: up to 255 chars, case sensitive
    - Extra metadata: Owner's name, last-modified timestamp
    - Single tree
    - Links allowed

Quiz

- Resource management in IDEs
  - Projects, packages, classes, data fields, methods

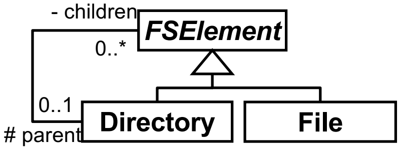


Package Explorer
- CS680
- CS681
  - cs681.assertion
  - cs681.comparators
    - ComparatorTest.java
      - ComparatorTest
        - main(String[]) : void

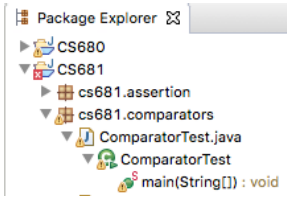
» Design the tree structure with *Composite*.

» Use at least 5 classes: Project, Package, Class, DataField, and Method

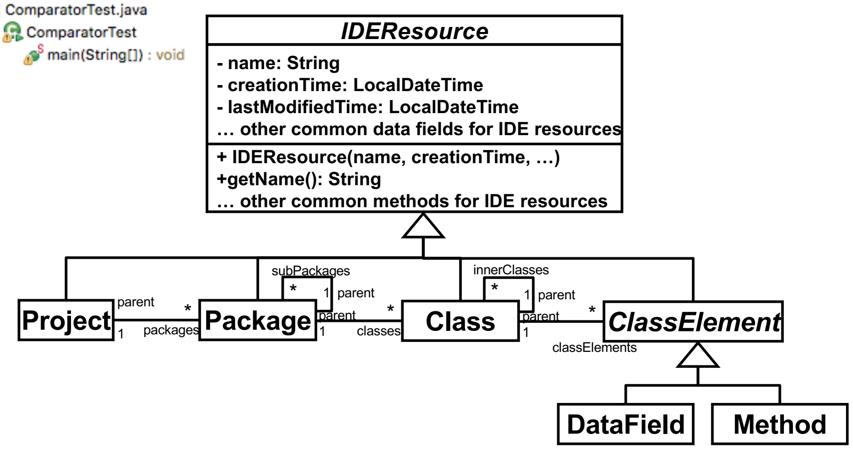
» c.f. file system example:



Example Solution 1



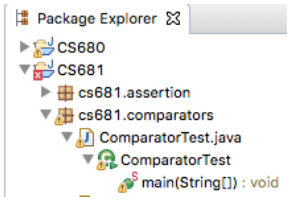
Package Explorer
- CS680
- CS681
  - cs681.assertion
  - cs681.comparators
    - ComparatorTest.java
      - ComparatorTest
        - main(String[]) : void



IDEResource

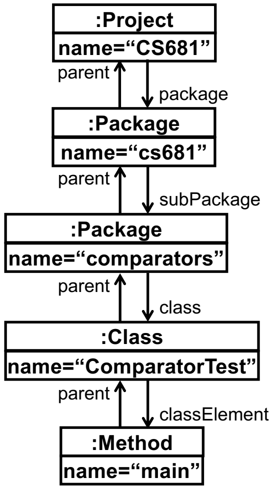
- name: String
- creationTime: LocalDateTime
- lastModifiedTime: LocalDateTime
… other common data fields for IDE resources

+ IDEResource(name, creationTime, …)
+getName(): String
… other common methods for IDE resources



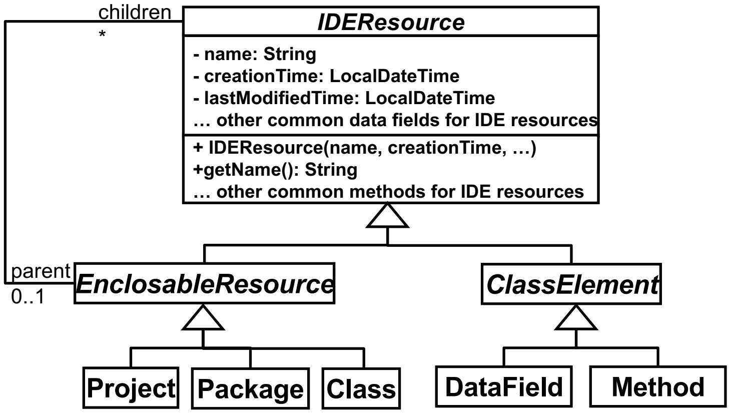
Package Explorer

- CS680
- CS681
  - cs681.assertion
  - cs681.comparators
    - ComparatorTest.java
      - ComparatorTest
        - main(String[]) : void

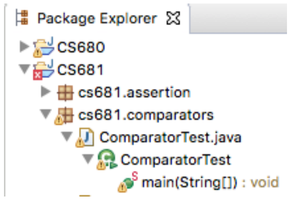


:Project
name=“CS681”
parent
package
:Package
name=“cs681”
parent
subPackage
:Package
name=“comparators”
parent
class
:Class
name=“ComparatorTest”
parent
classElement
:Method
name=“main”

Example Solution 2

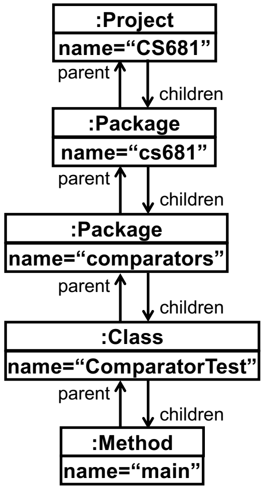


children
*
IDEResource
- name: String
- creationTime: LocalDateTime
- lastModifiedTime: LocalDateTime
… other common data fields for IDE resources
+ IDEResource(name, creationTime, …)
+getName(): String
… other common methods for IDE resources
parent
0..1
EnclosableResource
ClassElement
Project
Package
Class
DataField
Method



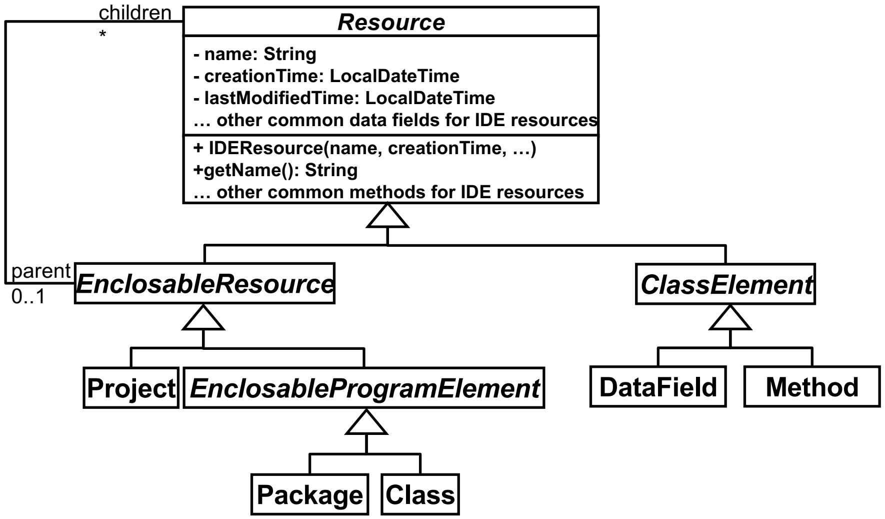
Package Explorer

- CS680
- CS681
  - cs681.assertion
  - cs681.comparators
    - ComparatorTest.java
      - ComparatorTest
        - main(String[]) : void

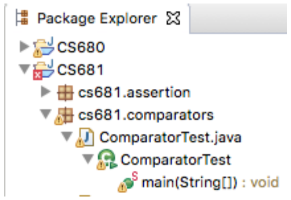


:Project
name=“CS681”
parent
children
:Package
name=“cs681”
parent
children
:Package
name=“comparators”
parent
children
:Class
name=“ComparatorTest”
parent
children
:Method
name=“main”

Example Solution 3

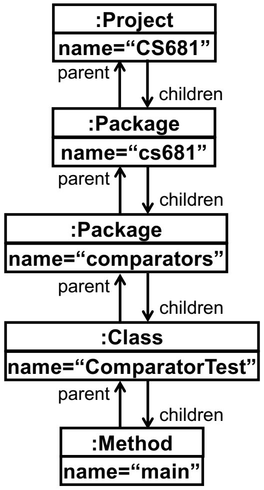


children
*
Resource
- name: String
- creationTime: LocalDateTime
- lastModifiedTime: LocalDateTime
… other common data fields for IDE resources
+ IDEResource(name, creationTime, …)
+getName(): String
… other common methods for IDE resources
parent
0..1
EnclosableResource
ClassElement
Project
EnclosableProgramElement
DataField
Method
Package
Class



Package Explorer

CS680
CS681
cs681.assertion
cs681.comparators
ComparatorTest.java
ComparatorTest
main(String[]) : void


:Project
name="CS681"
parent
children
:Package
name="cs681"
parent
children
:Package
name="comparators"
parent
children
:Class
name="ComparatorTest"
parent
children
:Method
name="main"