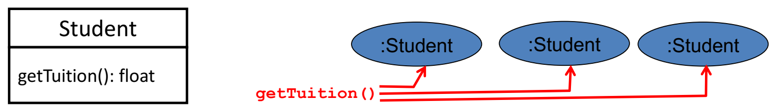
Visitor Design Pattern

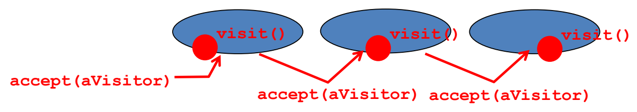
Visitor Design Pattern

- Intent
  - Separate (or decouple) a set of objects and the operations to be performed on those objects.

- In a traditional/normal design, if an operation is performed on some objects, it is defined (as a method) in a class(es) for those objects.

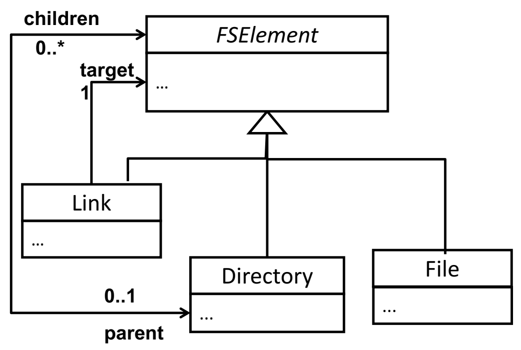


- With *Visitor*, the operation is defined in a visitor.

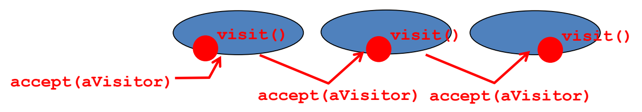


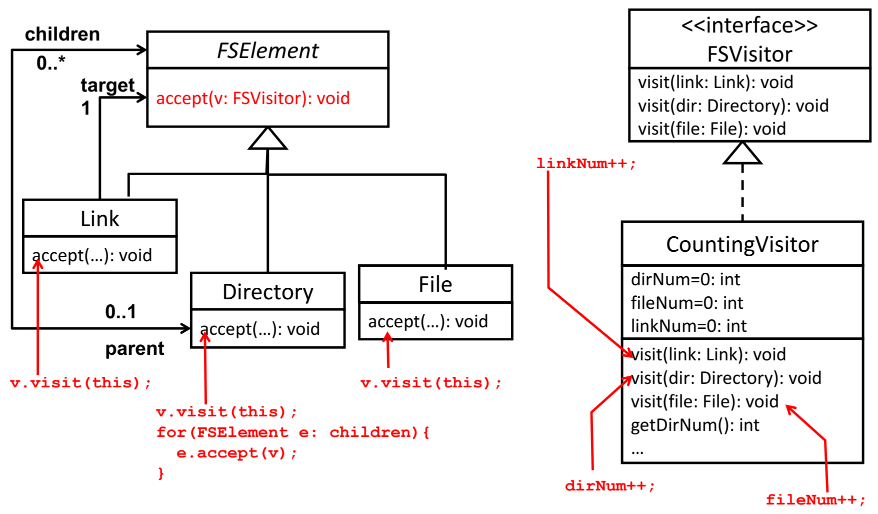
File System Examples (1/3)

- Counting the number of directories, the number of files and the number links in a file system



- With *Visitor*, an operation to count FS elements is defined in a visitor.





```
v.visit(this);
```

```
v.visit(this);
for(FSElement e: children){
  e.accept(v);
}
```

```
v.visit(this);
```

```
linkNum++;
```

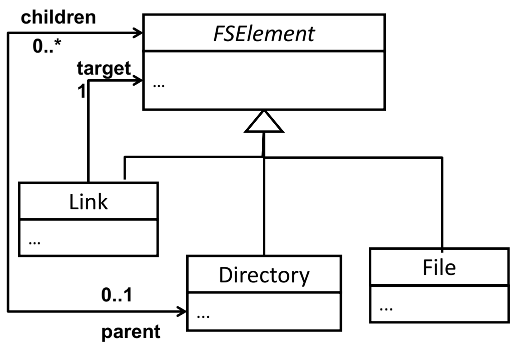
```
dirNum++;
```

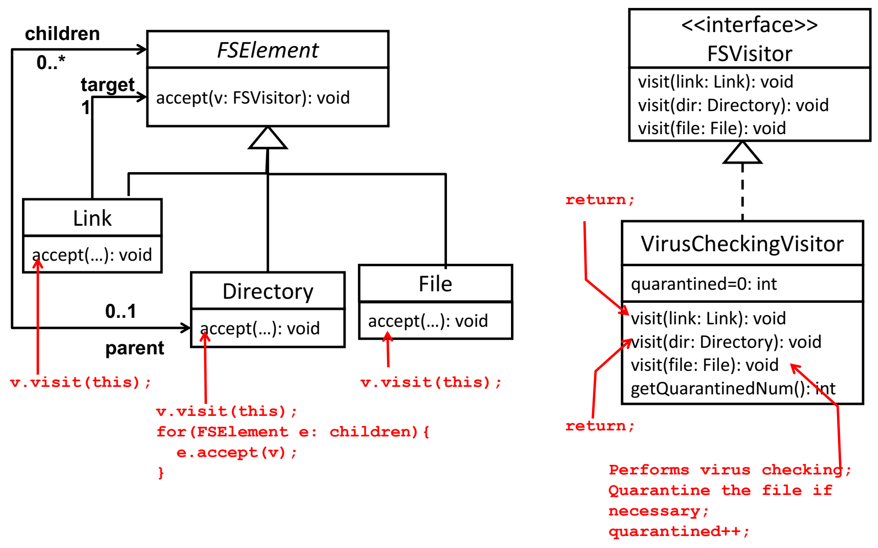
```
fileNum++;
```

```
CountingVisitor visitor = new CountingVisitor();
rootDir.accept( visitor );
visitor.getDirNum(); visitor.getFileNum(); visitor.getLinkNum();
```

File System Examples (2/3)

- Virus checking for each file
- File indexing





```
v.visit(this);
```

```
v.visit(this);
for(FSElement e: children){
  e.accept(v);
}
```

```
v.visit(this);
```

```
return;
```

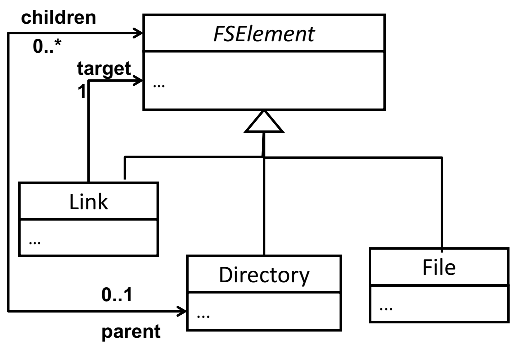
```
return;
```

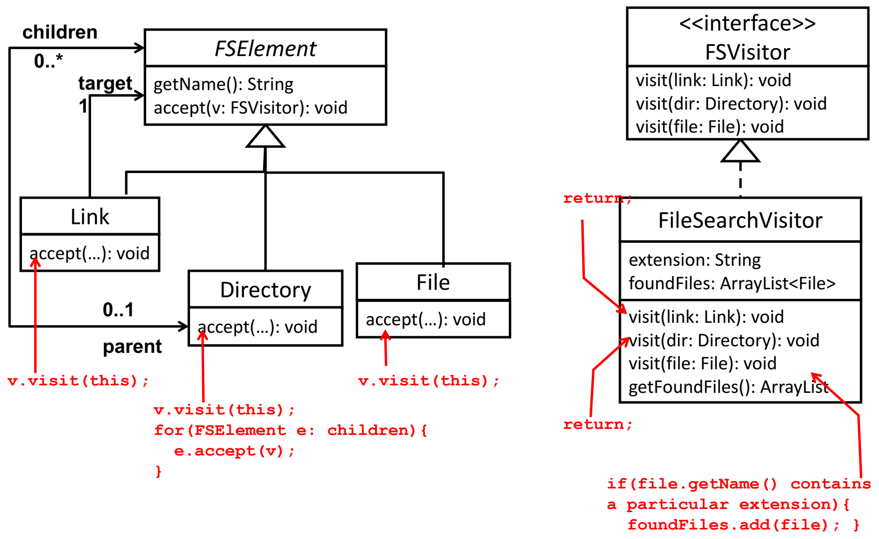
```
Performs virus checking;
Quarantine the file if
necessary;
quarantined++;
```

```
VirusCheckingVisitor visitor = new VirusCheckingVisitor();
rootDir.accept( visitor );
visitor.getQuarantinedNum();
```

File System Examples (3/3)

- File search
  - Searching/identifying files that have a particular extension
    - e.g., *.txt, *.jpg





```
v.visit(this);
```

```
v.visit(this);
for(FSElement e: children){
  e.accept(v);
}
```

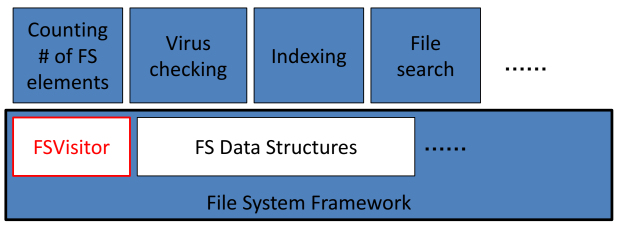
```
v.visit(this);
```

```
if(file.getName() contains
a particular extension){
  foundFiles.add(file); }
```

```
FileSearchVisitor visitor = new FileSearchVisitor(".txt");
rootDir.accept( visitor );
visitor.getFoundFiles().size();
```

What's the Point?

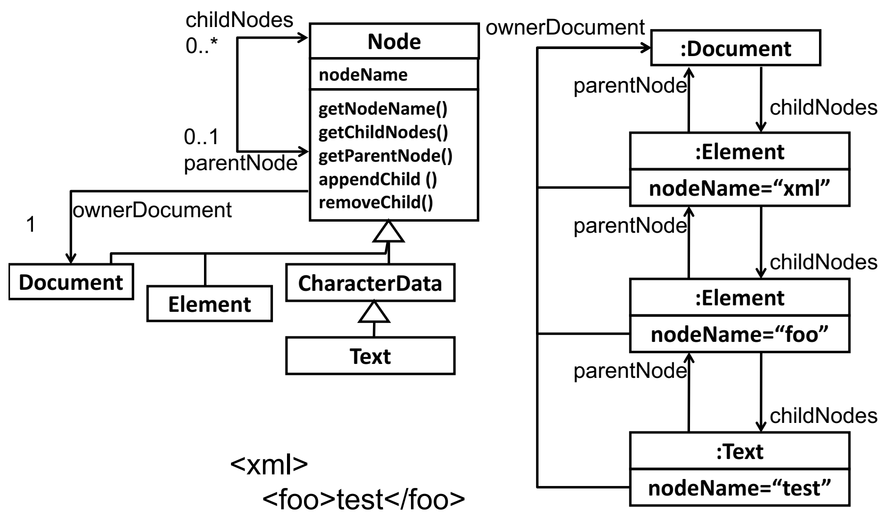
- Separating foundation data structures and the operations performed on those data structures.
  - It is easy to add, modify and remove operations.
  - Data structures can remain intact.



HW 10

- Implement FSVisitor and three visitor classes.
- Due: December 24 (Sun) midnight
  - c.f. Dec 16 to 22: Final exam period

Another Example with DOM

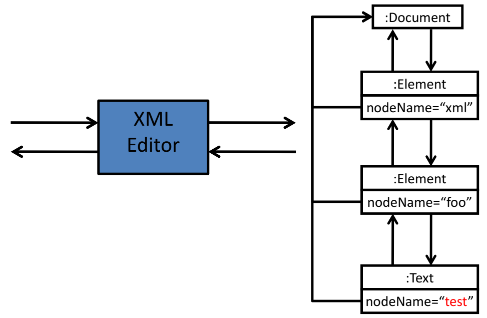


```
<xml>
   <foo>test</foo>
</xml>
```

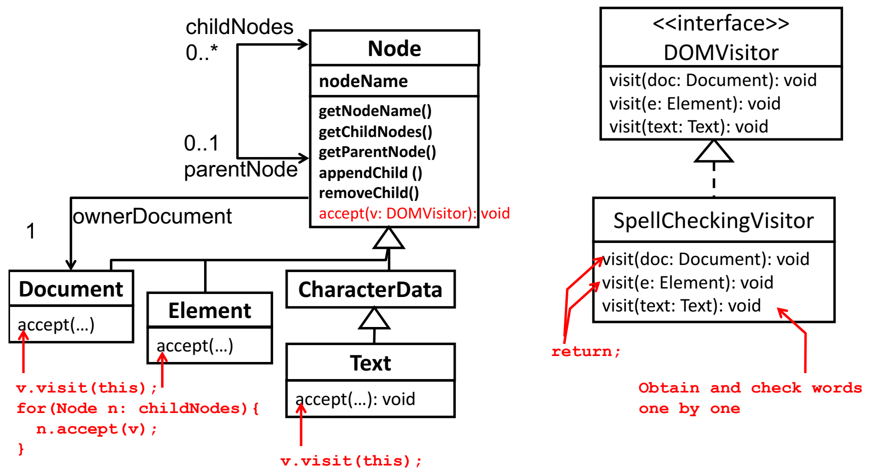
Spelling Checker in an XML Editor

- Imagine an XML editor that
  - Reads/imports an XML file, parses it and build its in-memory representation in DOM
  - Allows the user to check the spelling of each word in "Text" elements.

```
<xml>
   <foo>test</foo>
</xml>
```



Spelling Checker as a Visitor

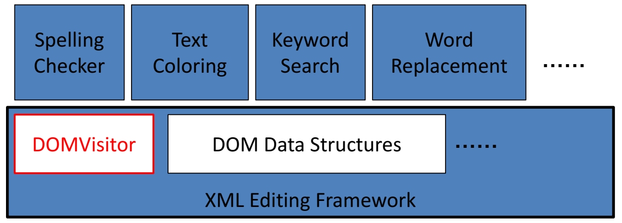


```
v.visit(this);
for(Node n: childNodes){
  n.accept(v);
}
```

```
v.visit(this);
```

Other Potential Visitors

- Many other visitors can be defined.
  - Any features/operations that are applied to a set of objects.



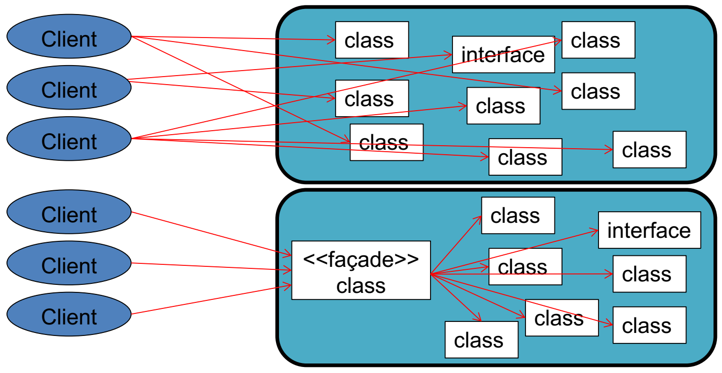
Applicability of *Visitor*

- *Visitor* can be applied to any collection of objects, not limited to *Composite*-based tree structures.
  - Set, list, graph, etc.

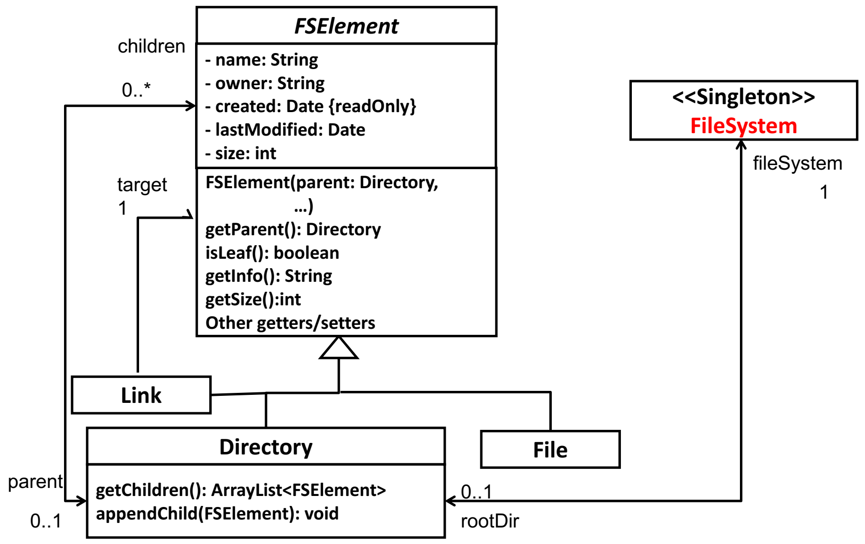
Façade Design Pattern

Façade

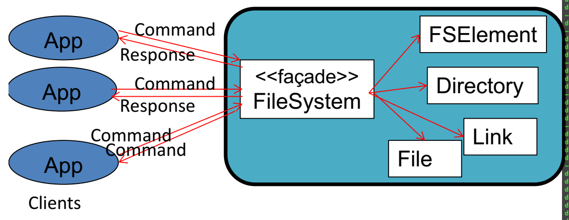
- Intent
  - Provide a unified interface (or primary point of contact) to a set of data structures in a system.
  - Define a higher-level interface that makes those data structures easier to use for clients.

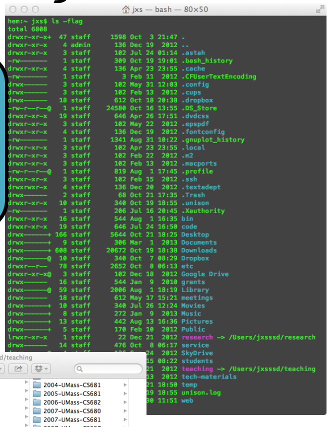


File System

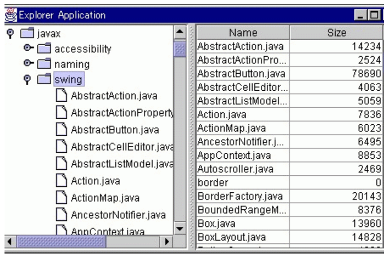


FileSystem as *Façade*



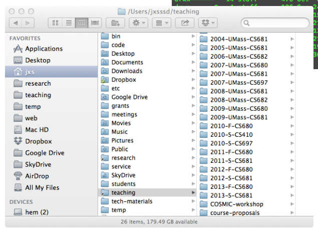

jxs — bash — 80×50

```
hem:~ jxs$ ls -flag
total 6808
drwxr-xr-x+  47 staff     1598 Oct  3 21:47 .
drwxr-xr-x    4 admin      136 Dec 19  2012 ..
drwxr-xr-x    3 staff      102 Jul 24 01:14 .astah
-rw-------    1 staff      309 Oct 19 19:01 .bash_history
drwxr-xr-x    4 staff      136 Apr 23 23:55 .cache
-rw-------    1 staff        3 Feb 11  2012 .CFUserTextEncoding
drwx------    3 staff      102 May 31 12:03 .config
drwx------    3 staff      102 Feb 13  2012 .cups
drwx------   18 staff      612 Oct 18 20:38 .dropbox
-rw-r--r--@   1 staff    24580 Oct 16 13:55 .DS_Store
drwxr-xr-x   19 staff      646 Apr 26 17:51 .dvdcss
drwxr-xr-x    3 staff      102 May 22  2012 .epspdf
drwxr-xr-x    4 staff      136 Dec 19  2012 .fontconfig
-rw-------    1 staff     1341 Aug 31 10:22 .gnuplot_history
drwxr-xr-x    3 staff      102 Apr 23 23:55 .local
drwxr-xr-x    3 staff      102 Feb 22  2012 .m2
drwxr-xr-x    3 staff      102 Feb 13  2012 .macports
-rw-r--r--@   1 staff      819 Aug  1 17:45 .profile
drwxr-xr-x    3 staff      102 Feb 15  2012 .ssh
drwxrwxr-x    4 staff      136 Dec 20  2012 .textadept
drwx------    2 staff       68 Oct 21 17:35 .Trash
drwxr-xr-x   10 staff      340 Oct 19 18:55 .unison
-rw-------    1 staff      206 Jul 16 20:45 .Xauthority
drwxr-xr-x   16 staff      544 Aug  1 16:35 bin
drwxr-xr-x   19 staff      646 Jul 24 16:50 code
drwx------+ 166 staff     5644 Oct 21 18:25 Desktop
drwx------+   9 staff      306 Mar  1  2013 Documents
drwx------+ 608 staff    20672 Oct 19 18:38 Downloads
drwx------@  10 staff      340 Oct  7 08:29 Dropbox
drwxr--r--   78 staff     2652 Oct  8 06:13 etc
drwxr-xr-x@   3 staff      102 Dec 18  2012 Google Drive
drwx------   16 staff      544 Jan  9  2010 grants
drwx------@  59 staff     2006 Aug  1 18:19 Library
drwx------   18 staff      612 May 17 15:21 meetings
drwx------+  10 staff      340 Jul 26 12:24 Movies
drwx------+   8 staff      272 Jan  9  2013 Music
drwx------+  13 staff      442 Aug 13 16:36 Pictures
drwx------+   5 staff      170 Feb 10  2012 Public
lrwxr-xr-x    1 staff       22 Dec 21  2012 research -> /Users/jxsssd/research
drwx------   14 staff      476 Oct  8 06:17 service
                                 24  2012 SkyDrive
                                 15 08:22 students
                                 21  2012 teaching -> /Users/jxsssd/teaching
                                 13  2012 tech-materials
                                 21 18:50 temp
                                 19 18:55 unison.log
                                 30 11:51 web
```


Explorer Application

- javax
  - accessibility
  - naming
  - swing
    - AbstractAction.java
    - AbstractActionProperty
    - AbstractButton.java
    - AbstractCellEditor.java
    - AbstractListModel.java
    - Action.java
    - ActionMap.java
    - AncestorNotifier.java
    - AppContext.java

Name	Size
AbstractAction.java	14234
AbstractActionPro...	2524
AbstractButton.java	78690
AbstractCellEditor...	4063
AbstractListModel...	5059
Action.java	7836
ActionMap.java	6023
AncestorNotifier.j...	6495
AppContext.java	8853
Autoscroller.java	2469
border	0
BorderFactory.java	20143
BoundedRangeM...	8376
Box.java	13960
BoxLayout.java	14828


/Users/jxsssd/teaching

FAVORITES: Applications, Desktop, jxs, research, teaching, temp, web, Mac HD, Dropbox, Google Drive, SkyDrive, AirDrop, All My Files

DEVICES: hem (2)

bin, code, Desktop, Documents, Downloads, Dropbox, etc, Google Drive, grants, meetings, Movies, Music, Pictures, Public, research, service, SkyDrive, students, teaching, tech-materials, temp

2004-UMass-CS681, 2005-UMass-CS681, 2006-UMass-CS682, 2007-UMass-CS680, 2007-UMass-CS681, 2007-UMass-CS697, 2008-UMass-CS681, 2008-UMass-CS682, 2009-UMass-CS680, 2009-UMass-CS681, 2010-F-CS680, 2010-S-CS410, 2010-S-CS697, 2011-F-CS680, 2011-S-CS681, 2012-F-CS680, 2012-S-CS681, 2013-F-CS680, 2013-S-CS681, COSMIC-workshop, course-proposals

26 items, 179.49 GB available

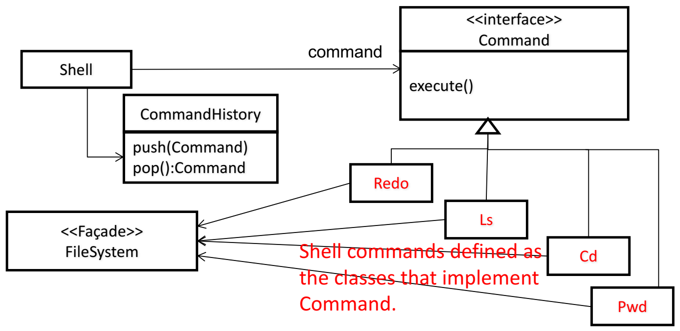
HW 11

- Implement a shell for your FS elements.
  - NOT GUI shell, but CUI (character UI) shell just like a Unix/Windows terminal.
- Implement individual shell commands with *Command.*
- Implement FileSystem as *Façade.*
- Implement a "pluggable" soring feature with Comparator (*Strategy*).

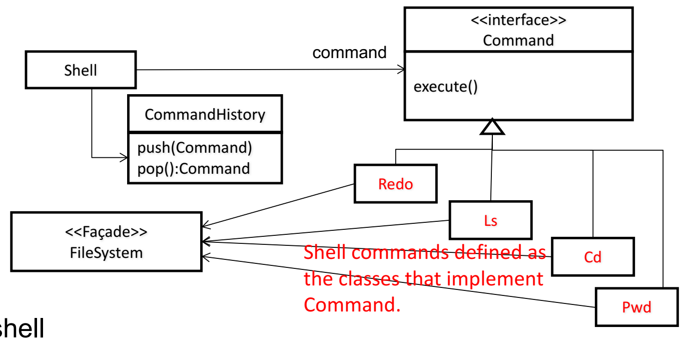
- Shell accepts the following commands:
  - pwd
    - Print the current working directory.
  - cd <dir name>
    - Change the current directory to the specified directory. Accept a relative (not absolute) directory name. Accept ".." (move to the parent directory of the current directory.)
  - cd
    - Change the current directory to the root directory.
  - ls
    - Print the name of every file, directory and link in the current directory.
  - dir
    - Print the information (i.e., kind, name, size and owner) of every file, directory and link in the current directory.
  - dir <dir/file name>
    - Print the specified directory's/file's information. Accept relative (not absolute) directory name. Accept ".."
  - mkdir <dir name>
    - Make the specified directory in the current directory.
  - rmdir <dir name>
    - Remove the specified directory in the current directory.
  - ln <target (real) dir/file> <link (alias) dir/file>
    - Make a link
  - mv <dir/file> <destination dir>
    - Move a directory/file to the detonation directory
  - cp <dir/file> <destination dir>
    - Copy a directory/file to the destination directory
  - chown
    - Change the owner of a file/directory
  - history
    - Print a sequence of previously-executed commands.
  - redo
    - Redo the most recently-executed command.
  - sort
    - Sort directories and files in the current directory

Designing FS Commands with *Command*

- Why *Command*?
  - There exist several (potentially many) clients/apps for a command.
  - Each command has relevant arguments/options.
  - New commands are often added.
  - Existing commands are often modified/updated.
  - Need to record/log command history.
    - "history" command, "up" arrow

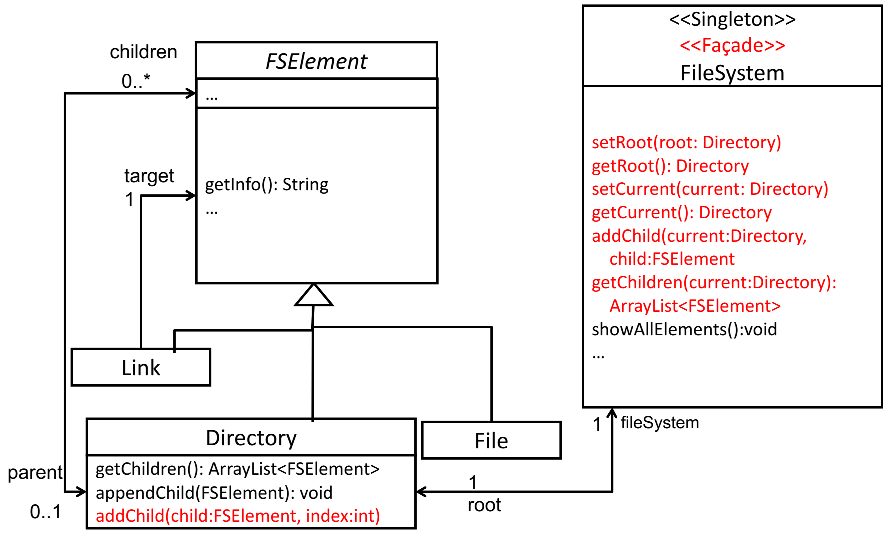


Designing FileSystem as *Façade*



- The shell
  - Receives a command (e.g. "cd" command) from the user,
  - Creates an instance of a corresponding Command class (e.g. Cd), and
  - Calls execute() on the instance.
    - execute() calls a method(s) of FileSystem.
  - Interacts with FileSystem through execute().
    - That is, FileSystem serves as Façade.

Example (not Complete) Methods in FileSystem



children
0..*
FSElement
...
target
1
getInfo(): String
...
Link
parent
0..1
Directory
getChildren(): ArrayList<FSElement>
appendChild(FSElement): void
addChild(child:FSElement, index:int)
File
1
root
1
fileSystem
<<Singleton>>
<<Façade>>
FileSystem
setRoot(root: Directory)
getRoot(): Directory
setCurrent(current: Directory)
getCurrent(): Directory
addChild(current:Directory, child:FSElement
getChildren(current:Directory): ArrayList<FSElement>
showAllElements():void
...

An Example Interaction among User, Shell and FileSystem

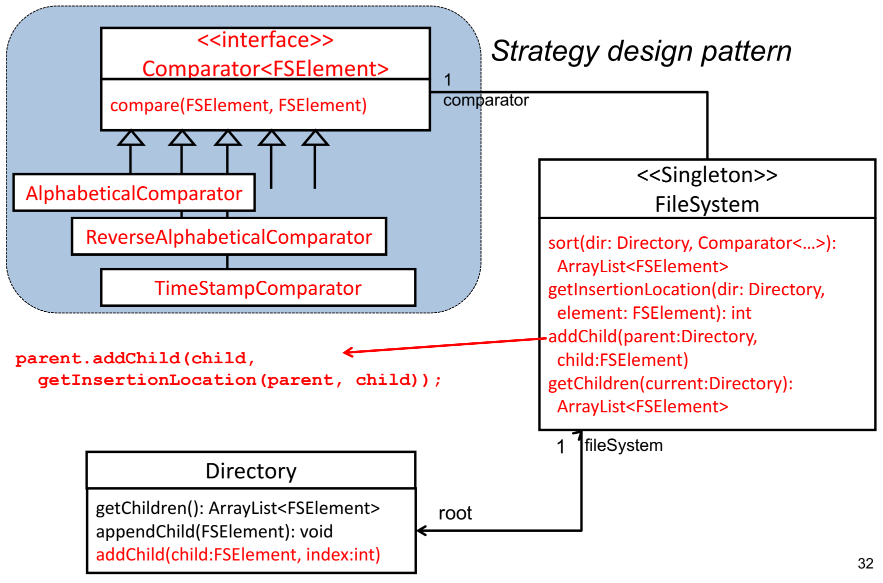
- The shell
  - prints out a prompt like ">",
  - lets the user enter a command and parses it,
    - Assume the user enters "cd …" as a command.
  - Creates an instance of Cd, and
  - Calls execute() on the instance.
- execute()
  - implements the logic of a command by calling a method(s) of FileSystem, and
    - execute() of the Cd class
      - Checks if the destination directory exists by calling getChildren(), etc. and moves to the destination by calling setCurrent().
      - calls setCurrent(getRoot()) if "cd" command has no parameters.
  - returns any output message to Shell.

Sorting FS Elements

- Example soring policies
  - Alphabetical
  - Reverse alphabetical
  - Timestamp-based (e.g. "last-modified date"-based)
  - Element kind based (e.g. directories listed first followed by files and links, file type based)

- It is not a good idea to hardcode sorting logic in Directory.
  - Whenever a new sorting policy is required, you need to modify Directory.
- Better idea: Make Directory open-ended for various sorting policies (i.e., make Directory loosely-coupled from sorting policies)
  - Allow each FS user to select a sorting policy dynamically
  - Allow FS developers to add new sorting policies in a maintainable manner.
    - Have them add extra code (classes) rather than modify Directory.
- Solution: Use *Strategy* (Comparator).

Soring FS Elements with Comparator


Strategy design pattern
<<interface>>
Comparator<FSElement>
compare(FSElement, FSElement)
AlphabeticalComparator
ReverseAlphabeticalComparator
TimeStampComparator
1
comparator
<<Singleton>>
FileSystem
sort(dir: Directory, Comparator<...>): ArrayList<FSElement>
getInsertionLocation(dir: Directory, element: FSElement): int
addChild(parent:Directory, child:FSElement)
getChildren(current:Directory): ArrayList<FSElement>
1
fileSystem
Directory
getChildren(): ArrayList<FSElement>
appendChild(FSElement): void
addChild(child:FSElement, index:int)
root

```
parent.addChild(child,
  getInsertionLocation(parent, child));
```

- addChild() always follows the default (alphabetical) sorting policy.
  - Directory always retains alphabetically-sorted FS elements.
  - getChildren() returns alphabetically-sorted elements.
- sort(Directory, Comparator<FSElement>) re-sorts FS elements based on a custom (non-default) sorting policy, which is indicated by the second parameter, and returns re-sorted elements.
  - Directory does not have to retain the re-sorted elements.
  - Implement at least one custom sorting policy (e.g., timestamp-based)

Extra Commands to be Implemented

- Support extra command that are associated with your visitors.
  - c.f. HW 9
  - e.g., count, viruscheck, search
  - Add extra methods in FileSystem to create and run visitors.
- All previous HW solutions for file system development must be integrated into a single code base.
- Unit tests are required for all classes.

- Due: December 24 (Sun) midnight
  - c.f. Dec 16 to 22: Final exam period