Hands-On 1 - Design Patterns

February 2023

arnaud.nauwynck@gmail.com

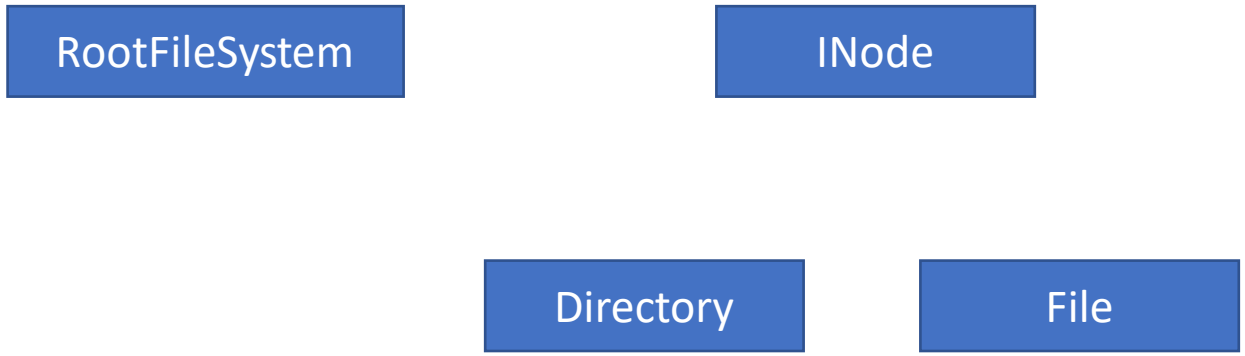
This document: https://github.com/Arnaud-Nauwynck/presentations
/tree/main/java/TP-design-patterns/handson-1-design-patterns.pptx

Outline

The goal of this Hands-On is to recognize and model as UML Classes many Design Patterns in the « FileSystem » (= File + Directories)

Composite,
Prototype,
Proxy,
Adapter,
Bridge,
Decorator,
Strategy,
Abstract Factory,
Command
Visitor,
…

Exercise 1 : complete UML links for classes diagram of a basic FileSystem



Directory: are composed of sub-directories and Files
File contains bytes
Both File and Directory are INodes
The root filesystem starts at « / » dir

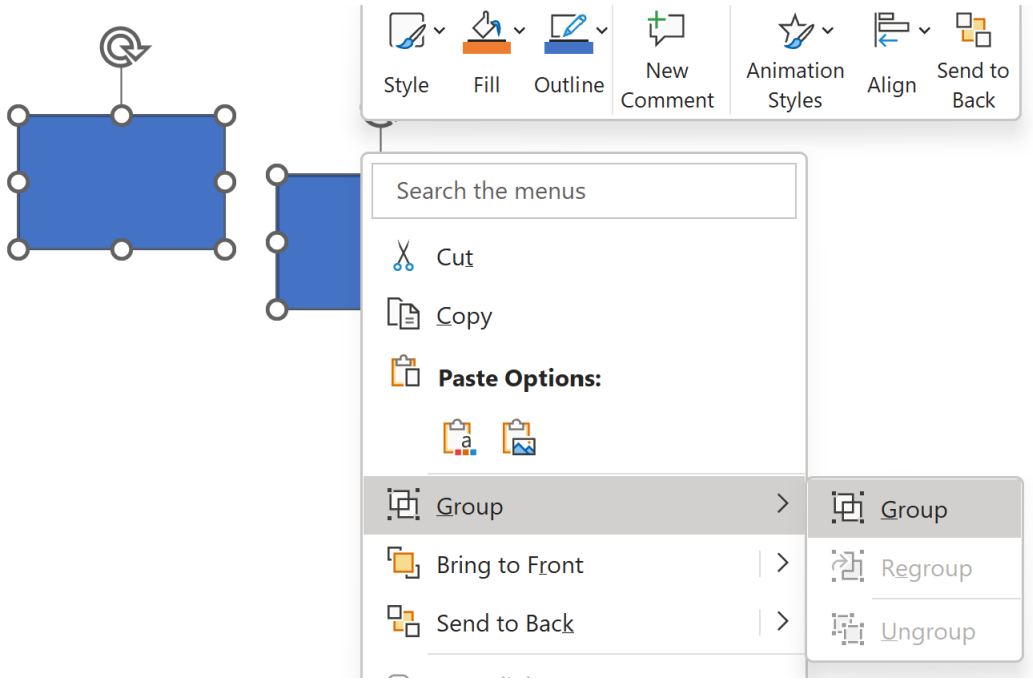
Exercise 2: Design pattern used for « Directory » ?

a/ Explain

b/ give the general structure of the « composite » Design-Pattern

c/ give other examples

Exercise 3 : Can you recognize same pattern in a graphic drawing app (PowerPoint)



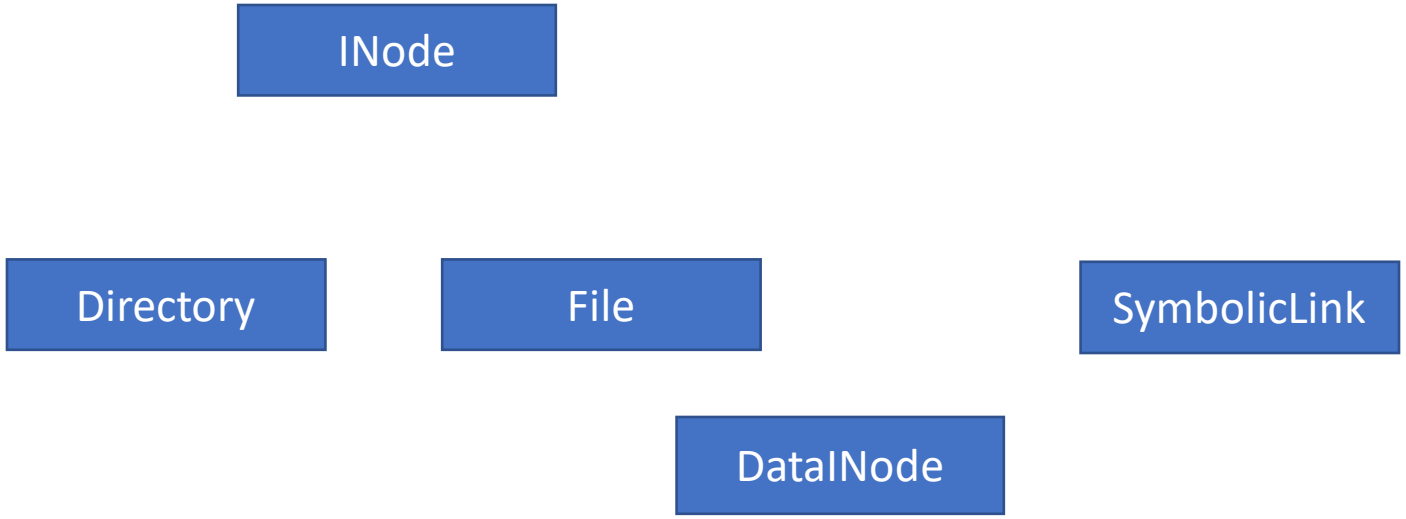
a/ give UML classes for a minimal graphic app supporting: Line, Rectangle, Text
b/ can you recursively « group » elements of type « group », rotate them ?

Exercise 4 : Copy&Paste .. Pattern=?

Every objects (File or Directory) can be copy&paste,
Therefore duplicating the full content of it

The file act as a « ?? Name of pattern ??» for creating another same file

Exercise 5 : complete more links



On Linux, you can create 2 types of links

$ ln a b → « hard link » : Inode counter is incremented for existing file « a », now also visible as « b »

… delete « a » => « b » still exists with content of previous file

$ ln –s a b → « symbolic link »

…. Delete « a » => « b » points to non-existing file name

Notice there is NO « Link » class, only « SymbolicLink » class … but there is a new class « DataINode »

Exercise 6 : explain design-pattern for « Symbolic Link » class

Exercise 7 : give other examples of « Proxy » design pattern in network

a/ DNS name vs Ipv4 address, Ipv6 address

b/ http proxy (cache)

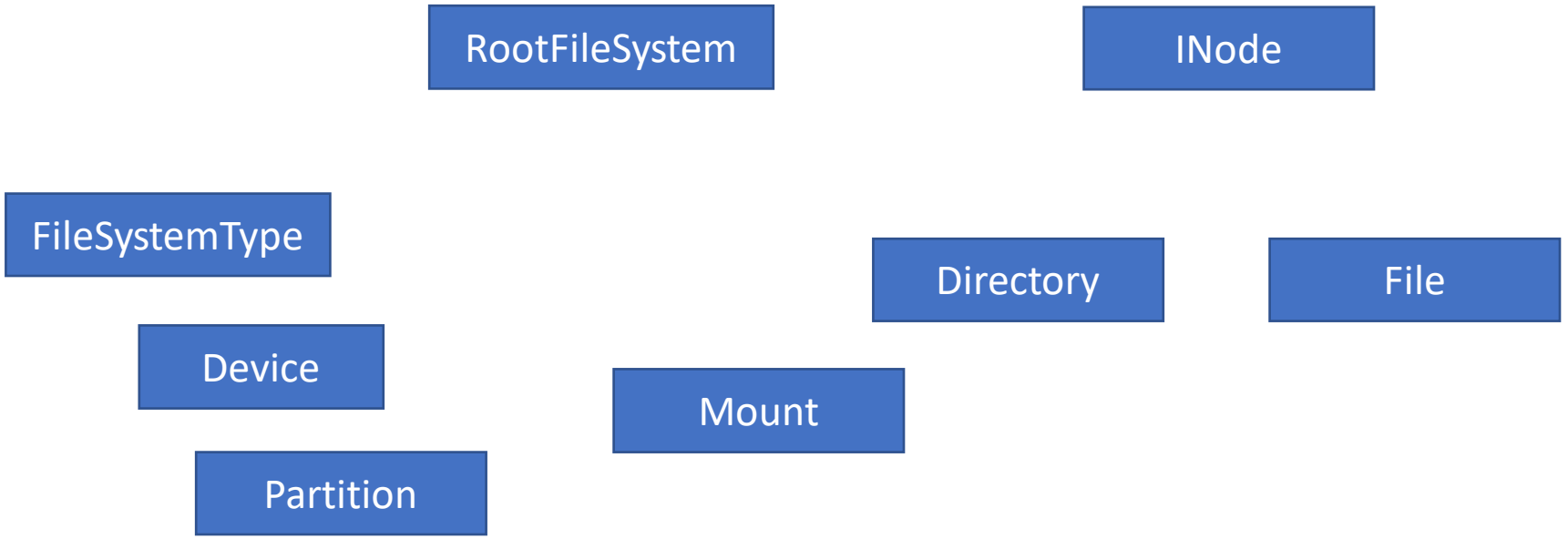
Model both examples as UML classes

Exercise 8 : explain design-pattern for « File» - « INodeData » class

Hint.. Bridge / Adapter / ..

AClass -> AImpl

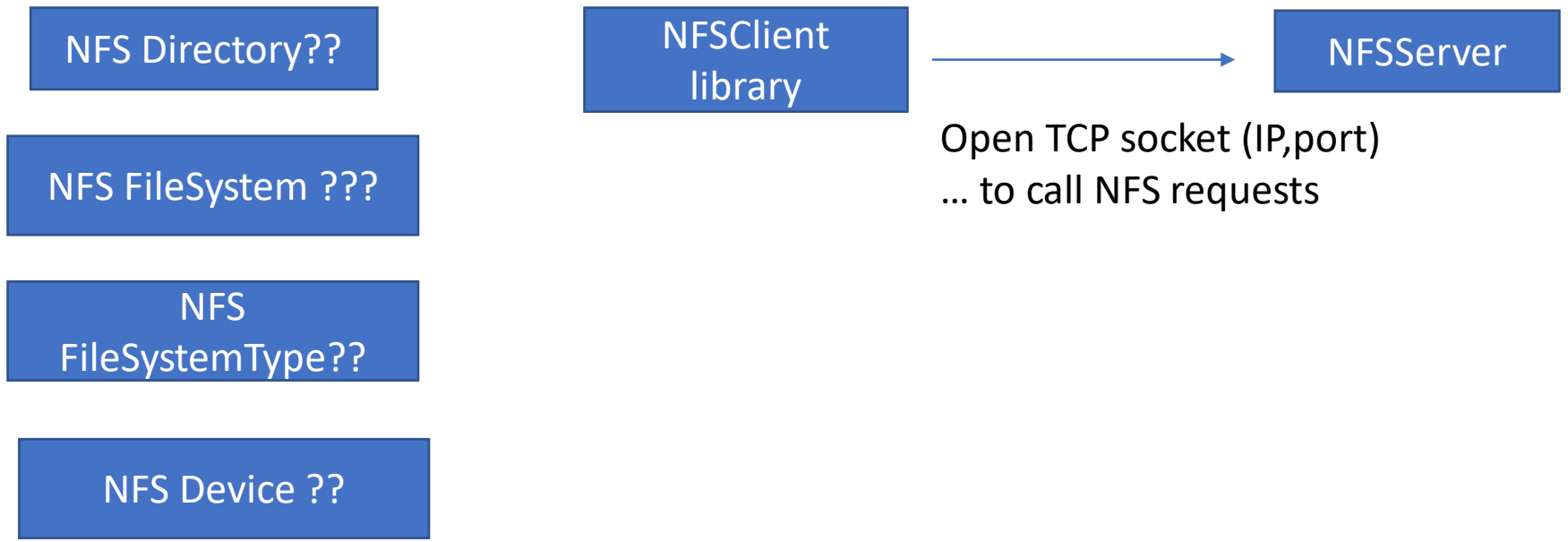
Exercise 9 : complete links … explain Pattern



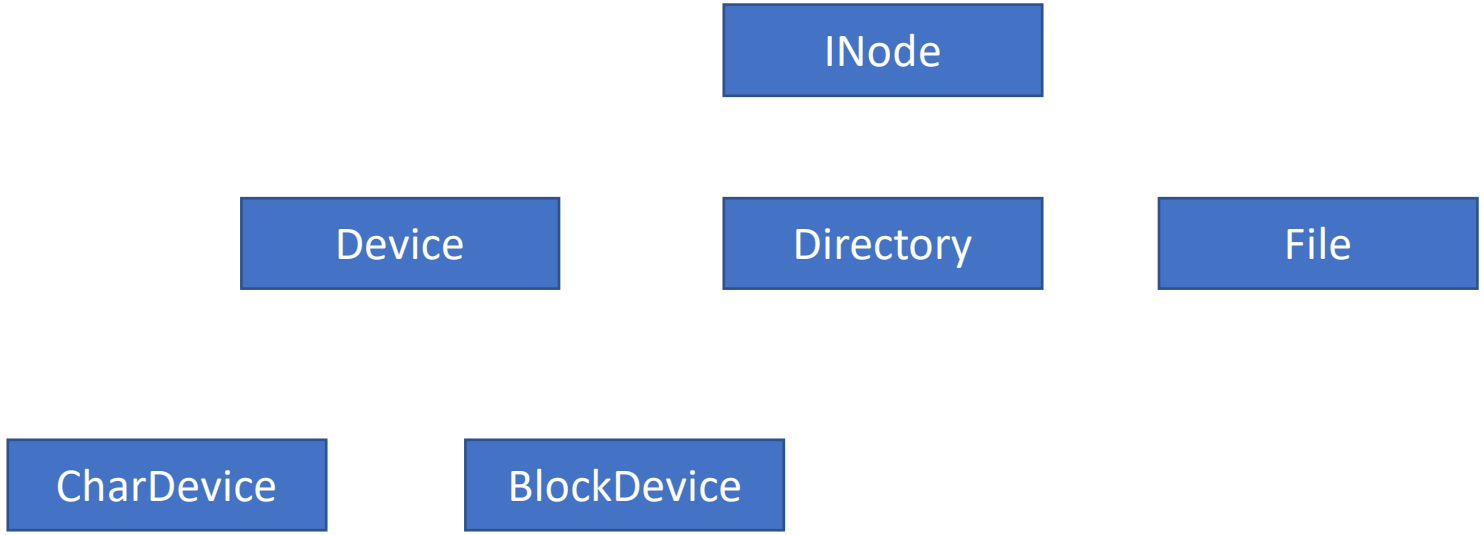
On unix, you can « mount » a partition of a disk(block device) as a directory ,
knowing its FileSystemType (ext3, betterfs, ntfs, ..)

```
$ su root
# mount –t ext2 /dev/hda1  /mnt/mydisk-partition1
# mount –t ext3 /dev/hda2  /mnt/mydisk-partition2
# mount      => show all current mounts
# cat /etc/fstab   => show all mounts at boot
```

Exercise 10 : model UML classes for unix « NFS » (Network File System)



Exercise 11 : linux « char device »



On linux, there are also « char device », in addition to « block device »
« char device » can get serial data, from a « mouse », a network interface, a keyboard, …
You can query linux mouse via « cat /dev/mouse » …

Model those classes

Exercise 12 : linux loopback device

Model UML class for loopback device

On linux, if you have a file (of 650Mo), with file format « iso9660 »
It can be seen as a « CD-ROM », and mounted

mkdir /mnt/iso
sudo mount -o loop /path/to/my-iso-image.iso /mnt/iso

Exercise 13 : linux « /proc » FileSystem

Model UML class for /proc special files

On linux, many system information can be read on virtual files of the « /proc » (and « /sys ») filesystems

$ ps
  => list processes
  … internally implemented as something equivalent to « ls /proc »

Getting details of process « $pid » can be obtained using
$ ls –l /proc/$pid/fd # list open files for process
$ cat /proc/$pid/xxx

Exercise 14 : Namespaces .. « Chroot »

Model as UML classes

On linux, you can « chroot » a process,
for security reasons, it is sandboxed, and can not get out of his « jail »

It is widely used for containers
…In fact, there is 1 « root » directory per « filesystem» namespace
There can be several namespaces

Exercise 15 : Namespaces .. Docker Containers support

Model as UML classes

A « process » points to several namespaces
When forking, a child process inherits all from its parent namespaces

Namespace for FileSystem
Namespace for IP,
Namespace for Processes
Namespace for Users
..

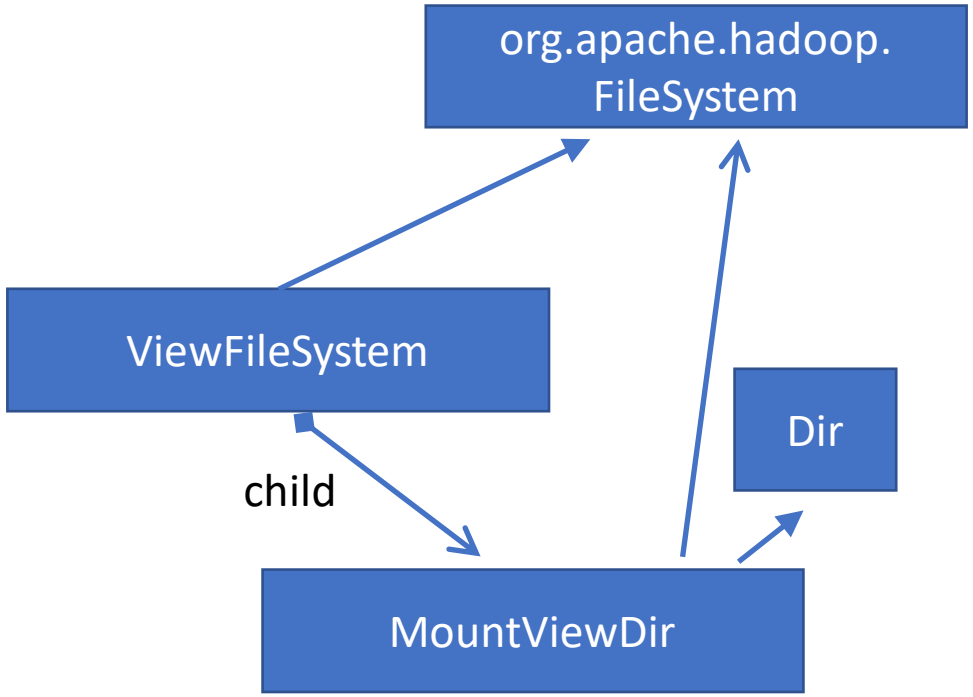
In kubernetes, all containers in the same pod share the same namespace..

Exercise 16 : File Explorer ClickHandler model as UML classes … recognize patterns « Decorator », « Chain Of Responsibility», « Strategy », else ?

When you double-click on a file
- with suffix « .exe » => it executes
- with suffix « .pdf », « .doc », « .pptx », « .html » …. => it opens an application
- with suffix « .tar », « .gz » => it proposes you to uncompress
- with suffix « .tar.gz » => it proposes you to uncompress, and then to extract in a new dir
- with suffix « .zip » => it proposes you to extract in a new dir

  zip files behave like a directory.. You can browse in it

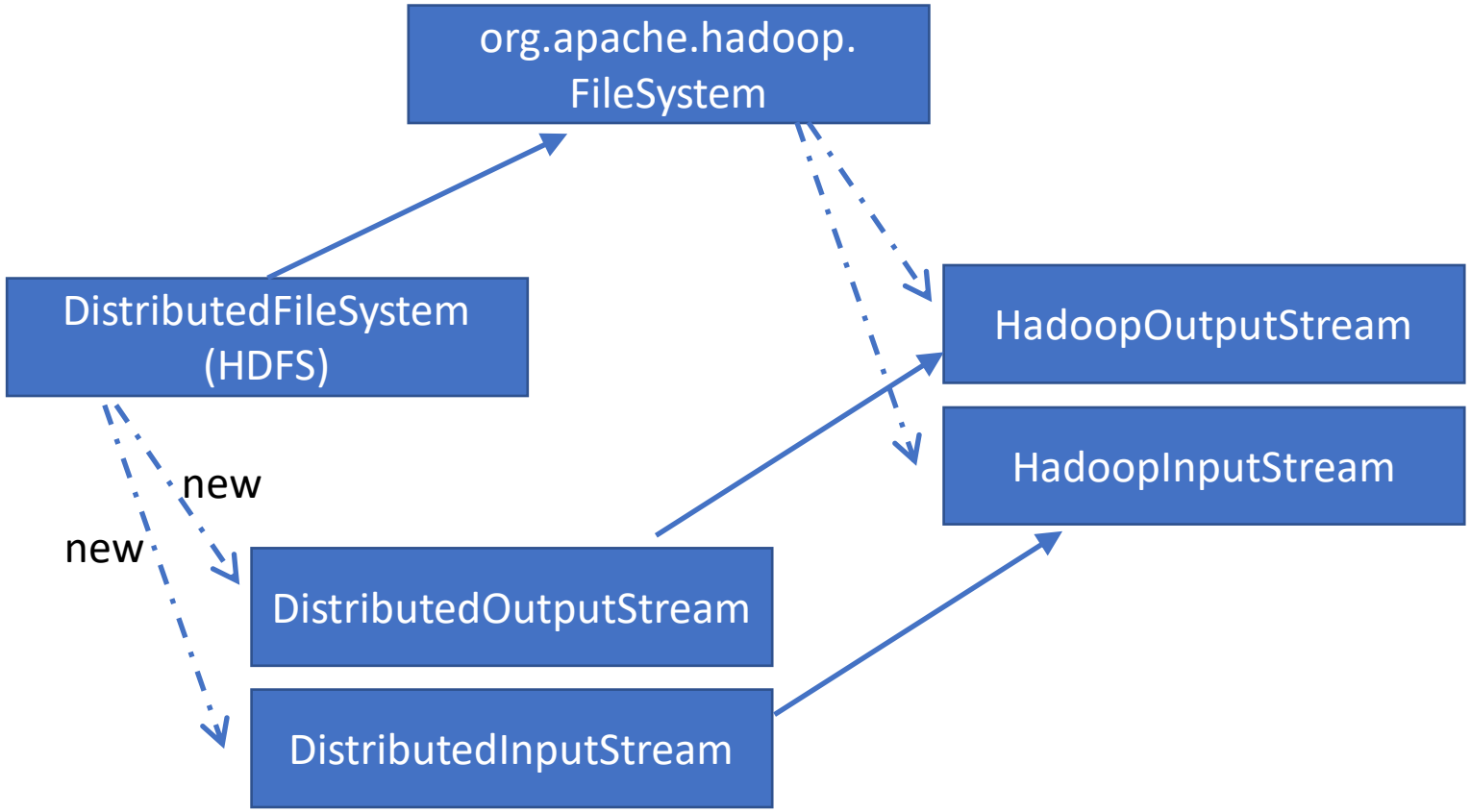
Exercise 17 : FileSystem class in Hadoop. which patterns do you recognize ?



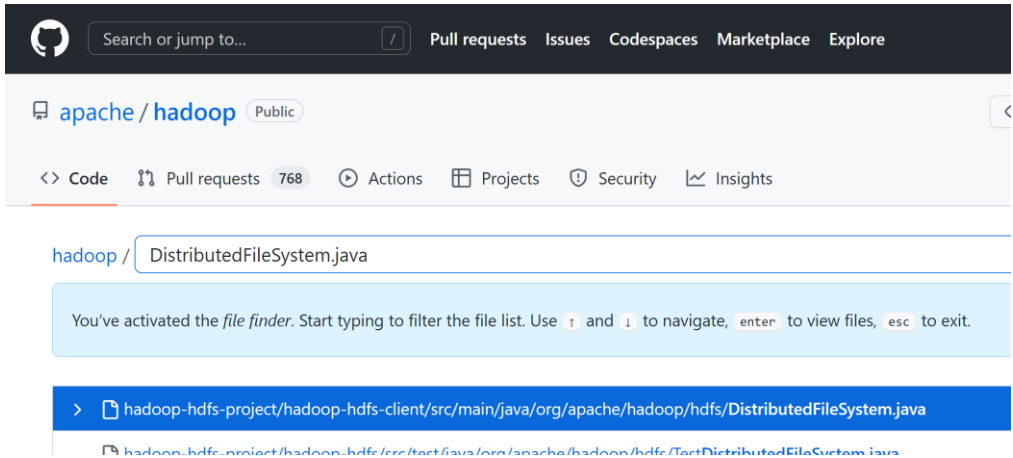
https://github.com/apache/hadoop/blob/trunk/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/fs/viewfs/**ViewFileSystem.java**

```
86  /**
87   * ViewFileSystem (extends the FileSystem interface) implements a client-side
88   * mount table. Its spec and implementation is identical to {@link ViewFs}.
89   */
90
91  @InterfaceAudience.Public
92  @InterfaceStability.Evolving /*Evolving for a release,to be changed to Stable */
93  public class ViewFileSystem extends FileSystem {
```

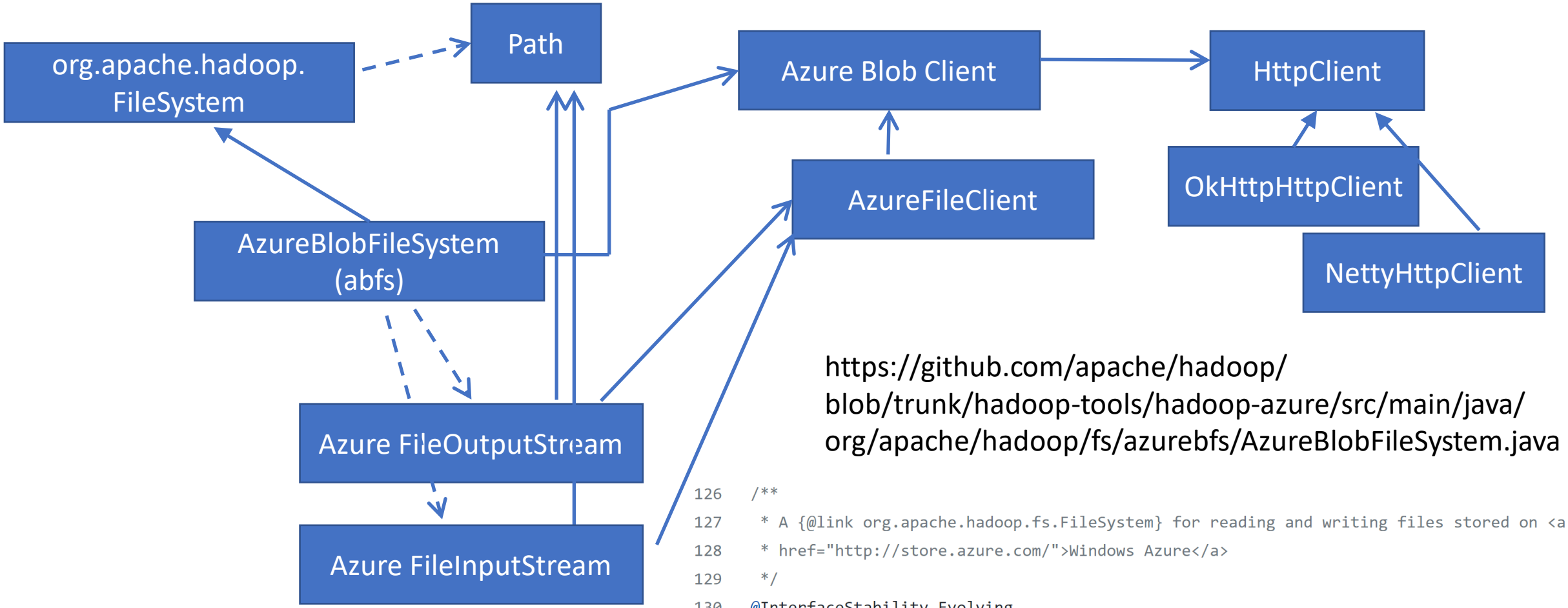
Exercise 18 : FileSystem class in Hadoop. which patterns do you recognize ?



https://github.com/apache/hadoop/



Exercise 19 : FileSystem class in Hadoop. which patterns do you recognize ?



https://github.com/apache/hadoop/blob/trunk/hadoop-tools/hadoop-azure/src/main/java/org/apache/hadoop/fs/azurebfs/AzureBlobFileSystem.java

```
126  /**
127   * A {@link org.apache.hadoop.fs.FileSystem} for reading and writing files stored on <a
128   * href="http://store.azure.com/">Windows Azure</a>
129   */
130  @InterfaceStability.Evolving
131  public class AzureBlobFileSystem extends FileSystem
```

Exercise …

Implements in Java

abstract class INodeDTO {.. }
class FileDTO extends INodeDTO { … }
class DirDTO extends INodeDTO { … }
Etc…

So as to describe some filesystem data using POJO classes
(to be serializable on Json, to get data over a http web page)

Exercise …

Implement the builder of DTO using real java.io.File
Implement a Factory
Implements the Visitor design pattern

Other ideas ?