Lecture 19 : Composite Design Pattern

Page No.
Date

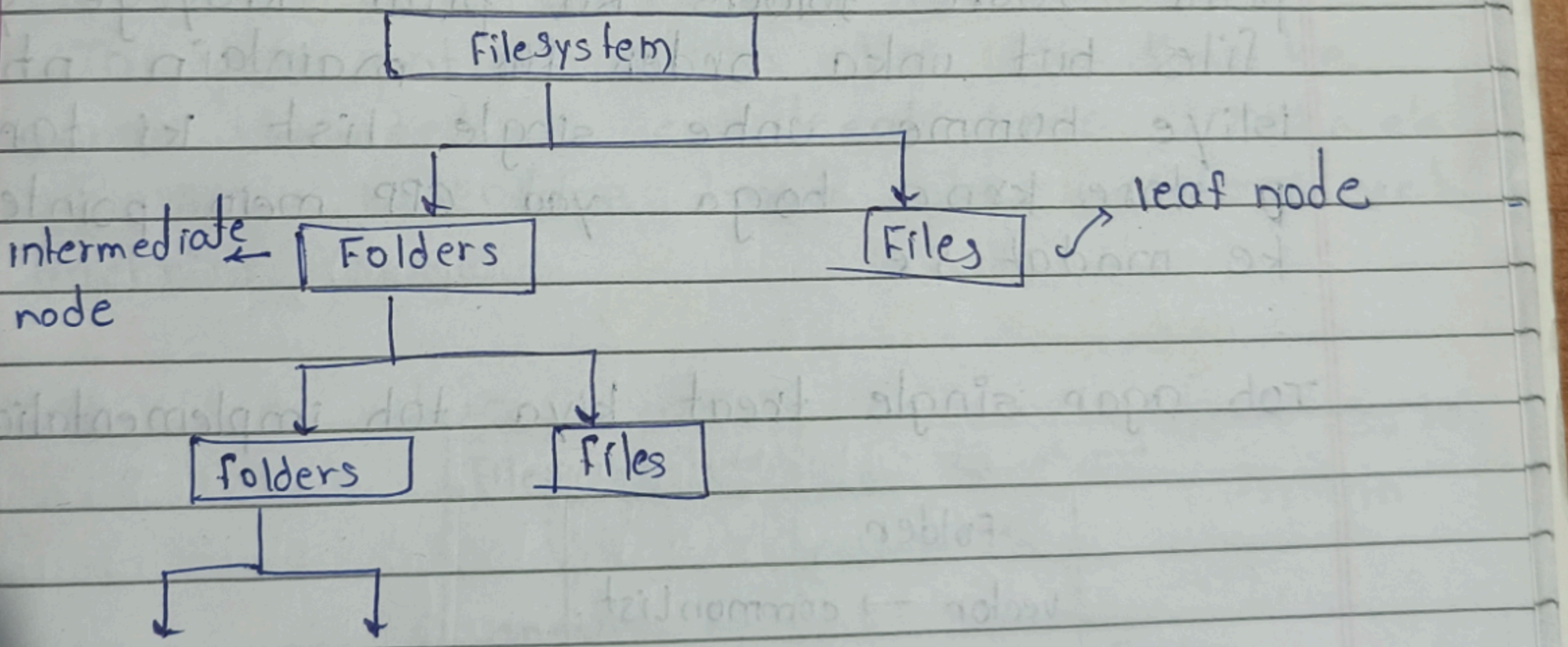
Introduction

- Abhi tak humne jitne bhi patterns dekhe woh kahi na kahi har problem mein use ho rhe the

- But composite design patter ek use case pr hi depend krta hai like uska need tbhi aayega jb aap koi hierarchy mein deal karoge

- koi bhi aise problems joh ki hierarchy form mein ho and uske pass doh tarah ke nodes hote hai - leaf and intermediate nodes.

Classic Problem : Designing File system



⇒ If composite Pattern did not existed !

Files
name;
size();
open();

(1..*)

Folder
name ;
vector < File > Files ;
vector < folder > Folder ;

Page No.
Date

- Ek folder ke pass ek se jyaada files and folders ho sakte hai

→ Ab implementation kyu hard hoga?

1. humme ls() command ko implement karna job list down karega usmein ke folders and files ko job ki uss directory mein present ho.

iske andar ke content nhi hoge display

2. openAll() → expanded form mein listing

- Ab humme openAll() ko implement krna hoga humare pass two list hai toh hum ek toh pehle saare folders ko print krayege yaa files but unka order fir maintain nhi hoga isliye humme unhe single list ki tarah store krna hoga yaa CPP mein pointer list ke madat se

Toh agar single treat kiya toh implementation-

Folder
vector → commonList
openAll()

commonList ← for(——) {

looping

if (element == file)
cout << element → name;
else
element → openAll(); → folder ko openAll() call hoga recursively
}

- so agar h
objects ki
aayenge ,
padega

- Aur sing
implement

What is
- Humare
hote hai

- To
same
commo
overri

UML

Fi

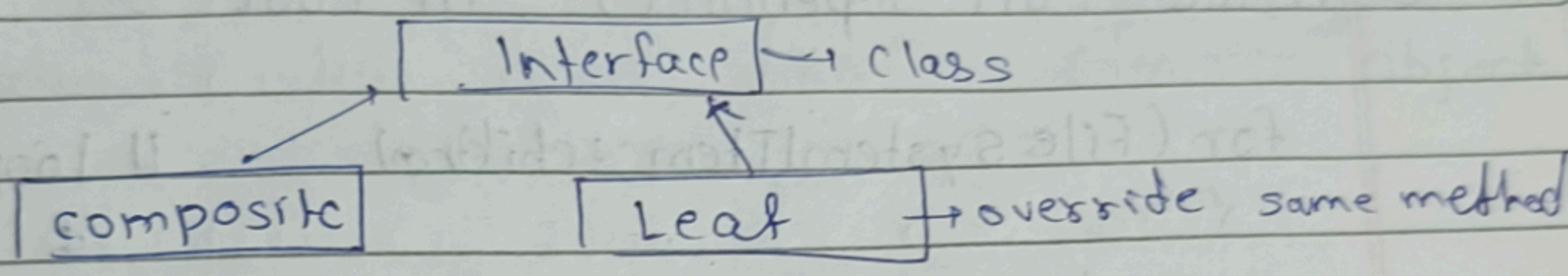
String
string
ls()
oper
get
get

- So agar humne file & folder ko different objects ki tarah treat kiya ⟶ bahoot problems aayenge, aur sequence bhi yaad karna padega

- Aur single object ki tarah treat kiya toh implementation like ls() ⟶ complicate hoga

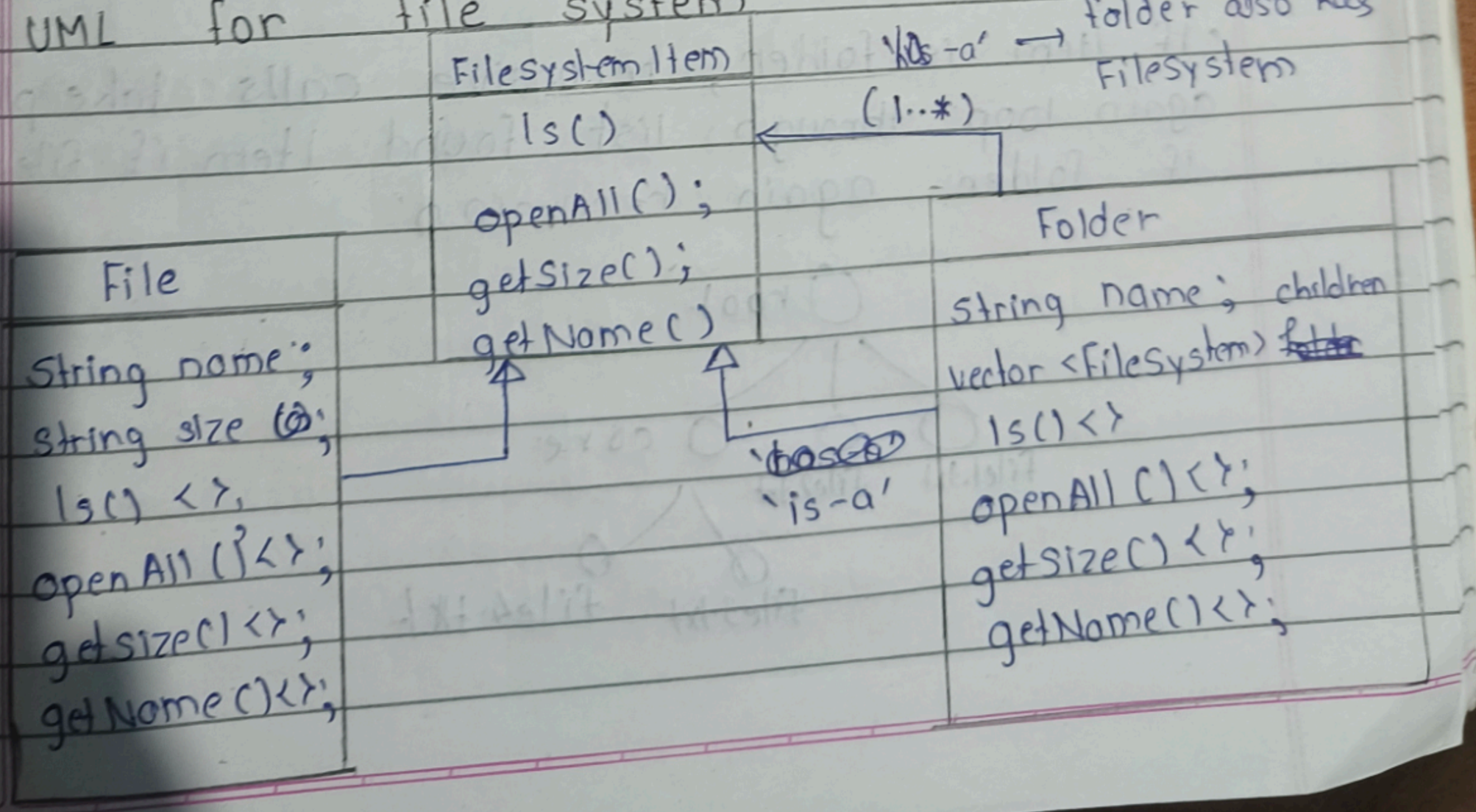
What is composite Pattern all about?

- Humare pass do (2) tarah ke elements hote hai - composite & leaf



- To yeh pattern kehta hai unn dono ke same tareeke se treat karo & unka ek common interface hoga jiske methods yeh override krege

UML for file system



Page No.
Date

◉ Hierarchy

(+) root — folder
- file1.txt
- file2.txt
- (+) core
  - file3.txt
  - file4.txt
- (+) user
  - file5.txt

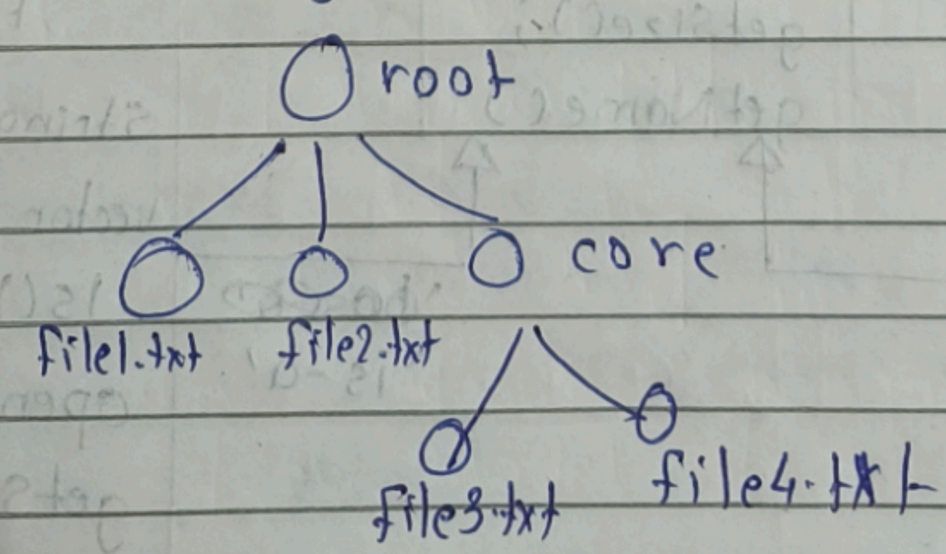
(+) denotes folder that can be expanded

◉ If we call openAll() in root —

```
for(FileSystemItem :children)        // loop through list & call files & folders in it
{
    cout<<Item -> openAll();
}
```

- If Item is file — openAll {
  getName();
  }

- If Item is folder, recursive calls take place. It again loop through list found Item if file → getName if folder again recursion



standar

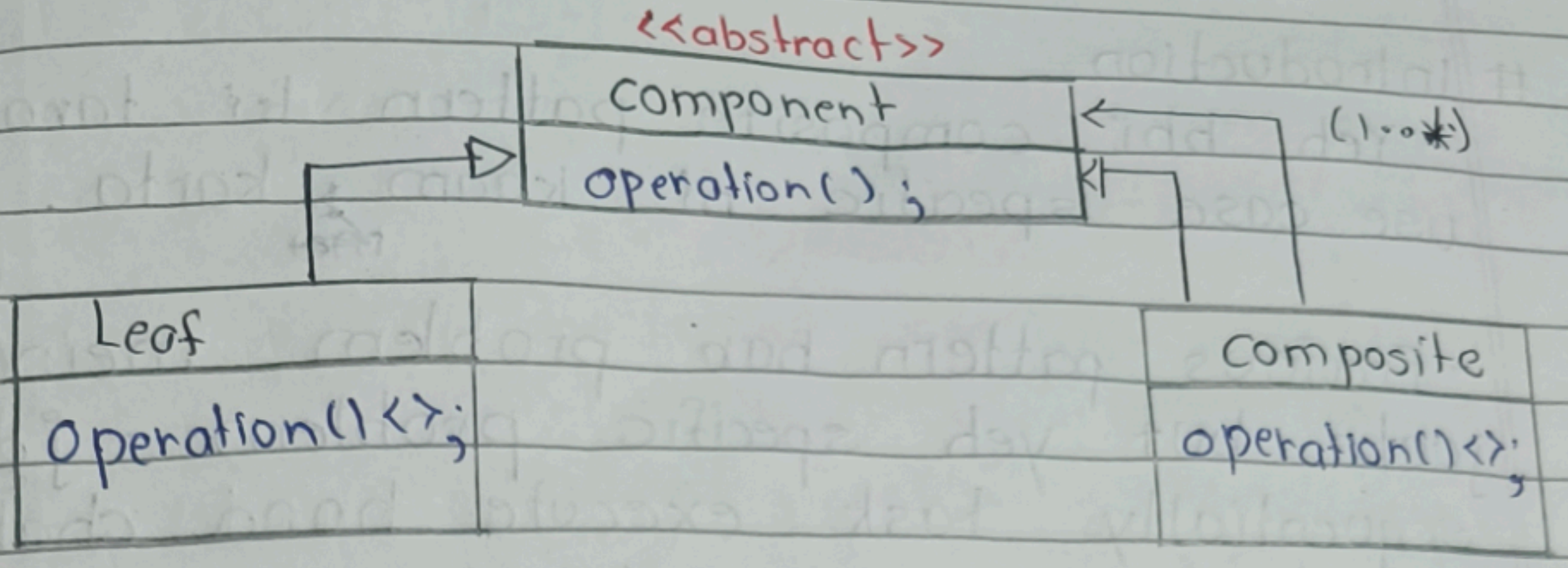
standa
Compo
tree
hiera
and

Real

Us
yo
Exc

①
②

Standard UML



Standard Defination

Composite pattern composes object into a tree like structure representing a part-whole hierarchy. It let client treat composition of object and individual object uniformly.

Real World Use Case

Useful in problem jaha tree like DS ho yaa hierarchy ho -

Example :

1. File System
2. Drop Down Menu