Lecture 38 : Visitor Design pattern

Introduction

1) Easy pattern help to use strategy pattern efficiently (will see it later).

2) Let's understand why we need visitor design pattern. – suppose there is class A and with some methods. as requirement increases then methods also increases. It leads to $\underset{\downarrow\text{Break}}{\text{OCP}}$ principle and SRP.

UML Diagram for Filesystem Example:-

1) Assume we have Application which handle multiple type of document - textfile, image, video.

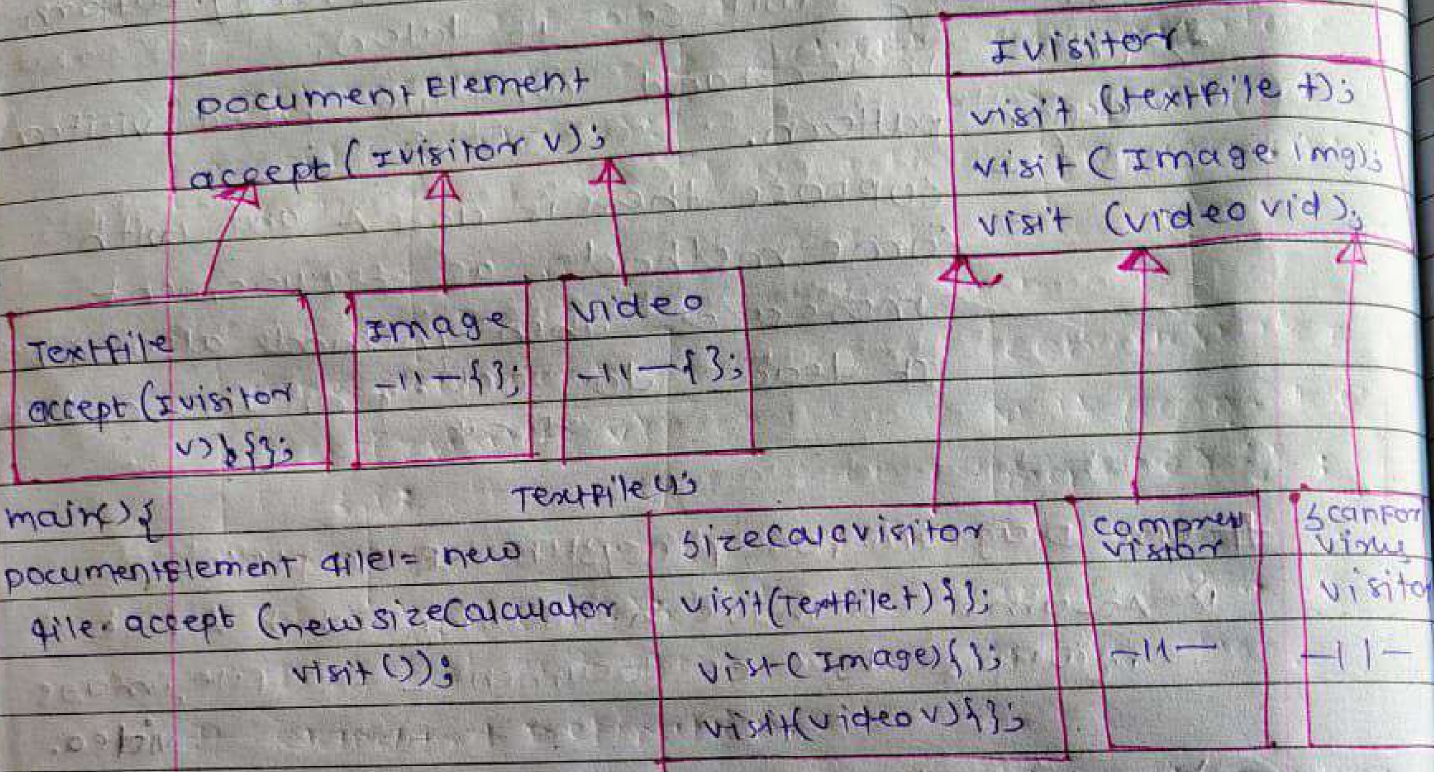
2) we create parent class having some concrete class - textfile, image, video.

3) Now we have requirement to calculatesize(), then compress(), scan for virus().

4) Now our req^n are increasing leads to break OCP and SRP principle. to learn these all patterns the main rule is to extract to changes.

5) Now to implement visitor pattern we need to figure out which part of our application is not changing and in this let's, let's assume type of document file not change. ie:- textfile, video, image will be some or rarely we will add more.

⑦ But we are frequently changing our methods. Now let's see our final UML if we need used visitor pattern.



```
main(){
DocumentElement file1= new TextFile();
file.accept(new SizeCalculator
                visit());
```

strategy and visitor:- Unknown friendship

- They both solve a most important problem of tight coupled application
- Now when we learn strategy dp we learnt it using example of robot where No. of operations same like - fly(), talk() just their behaviour is different.

* Now in visitor pattern agar same example robot ko le toh uske no. of behaviours same nhi honge like new behaviour jaise swim() aa jayega to hum visitor pattern lagayenge but how it will behave will be same

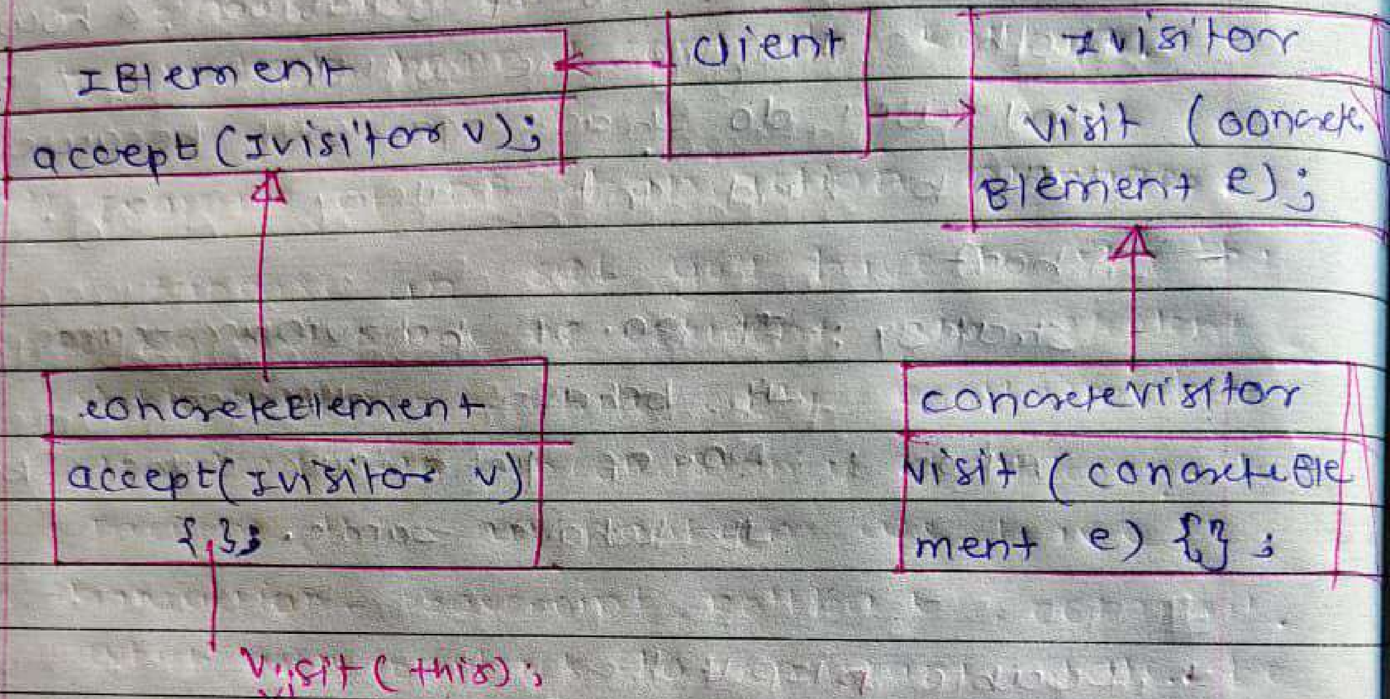
- If we add new method in strategy then it will break ocp.
- But incase of No-of behaviours & how it behave both is changing. we will use strategy pattern. and will do trade off with because, visitor apply will be hard and strategy is easy.
- Inshort
  strategy :- If no. of behaviour same but how it behaves different.
  visitor :- No. of behaviour changes but how it behaves same.

* Double Dispatcher :

- In our example we create file class create kiya using TextFile class so then call size(acvisitor ka reference diya tha let's call it ref1);
- Now to textfile accept mein ek reference ref2 legi and apna reference bhi pass kregi & size (acvisitor (TextFile)) ka reference lega and decide krega un teeno mein se knosa method ~~chalna~~ chalna chahiye

- This is double dispatcher
  ↳ deciding which method to execute based on type of two object involved in method call.

standard UML



Standard Defination.

- Allows to add New operation to existing classes without, changing these structure seprates operation from, the Object it operates.

Real world use case.

1) compiler
2) Game Rendering Engine.