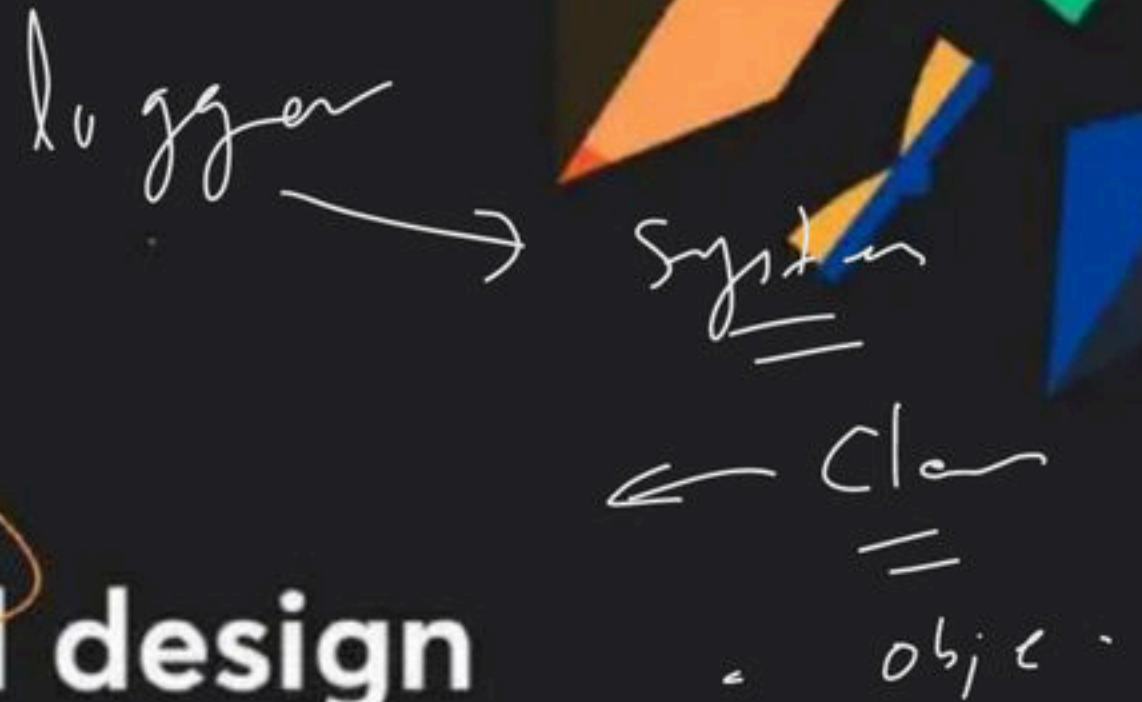
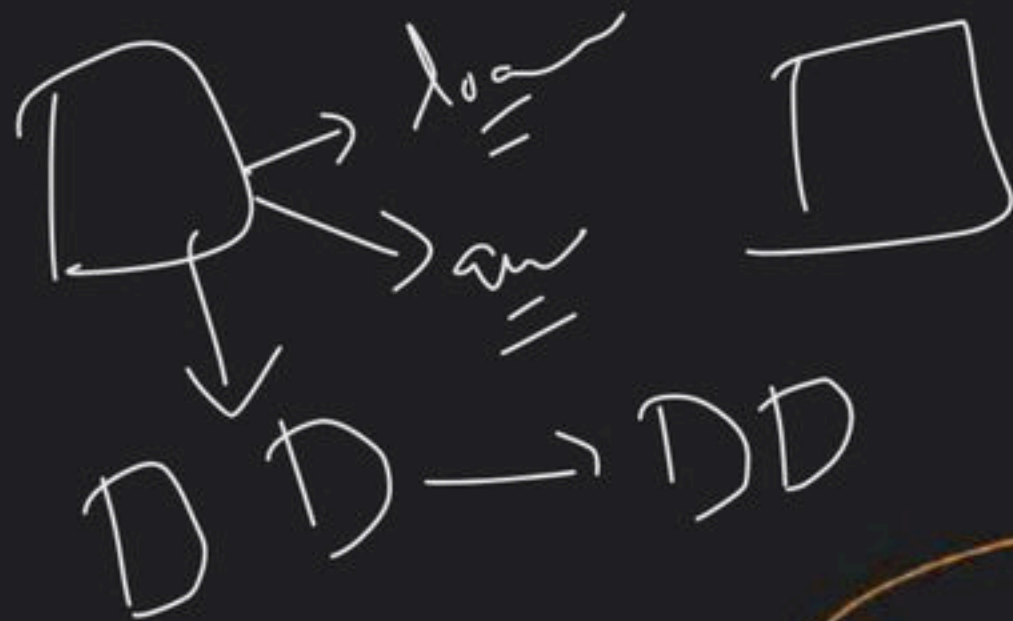




# Doubt Class with Lakshay

Special class

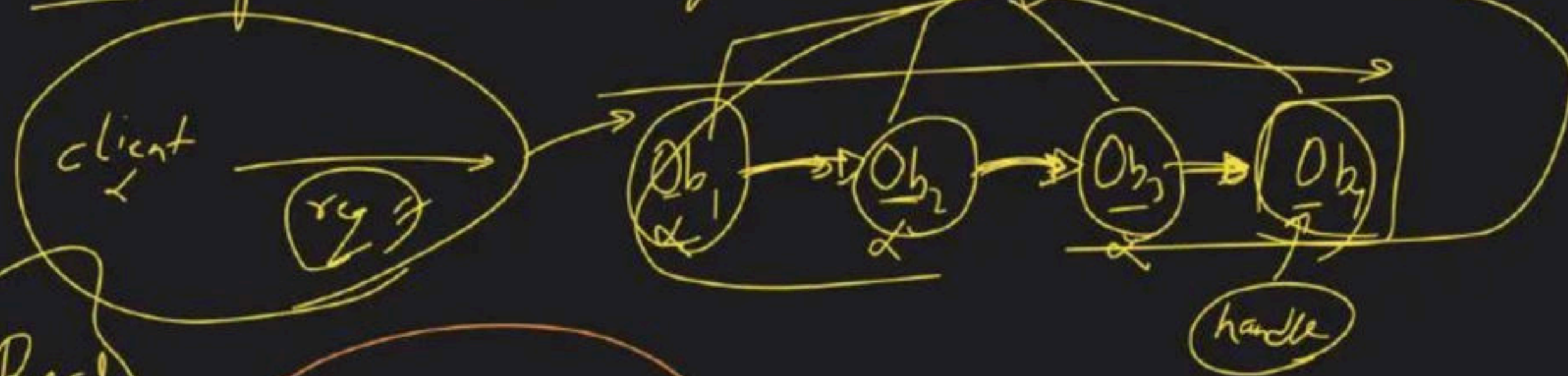


# Introduction to Behavioral design patterns

Special class

Object interaction  
& delegate responsibility

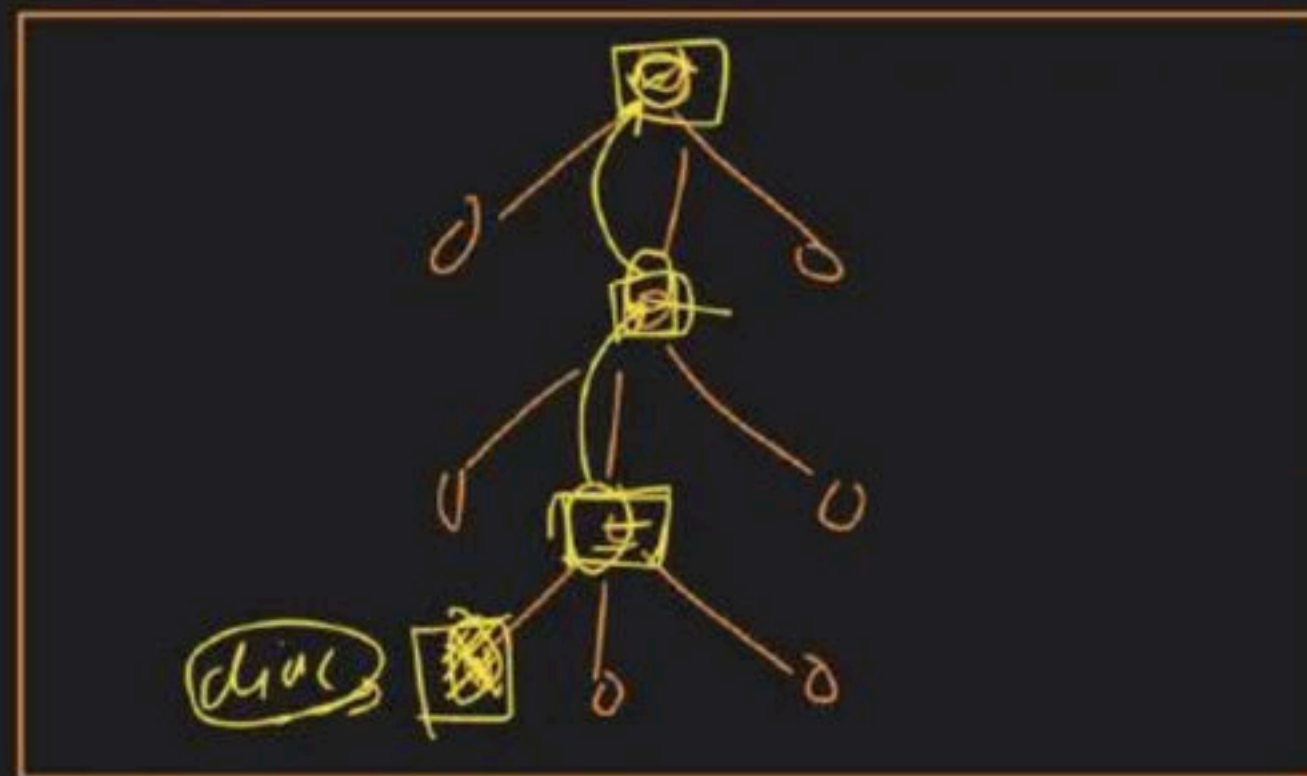
# ① Chain of Responsibility



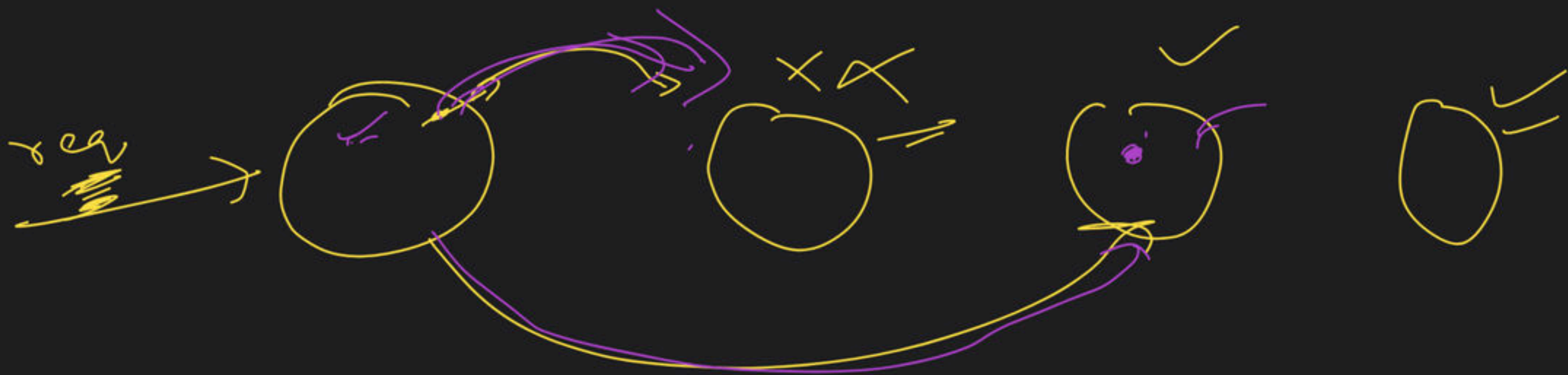
Real Life

Event Bubbling in DOM

Inf → Common interface  
→ Ob → successor object reference

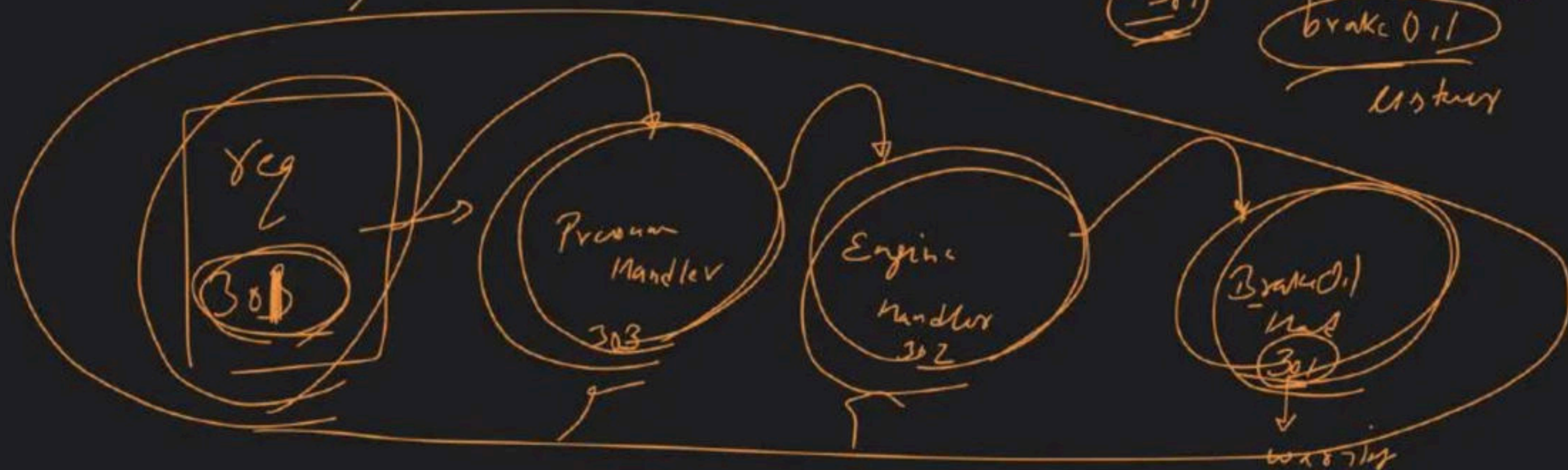




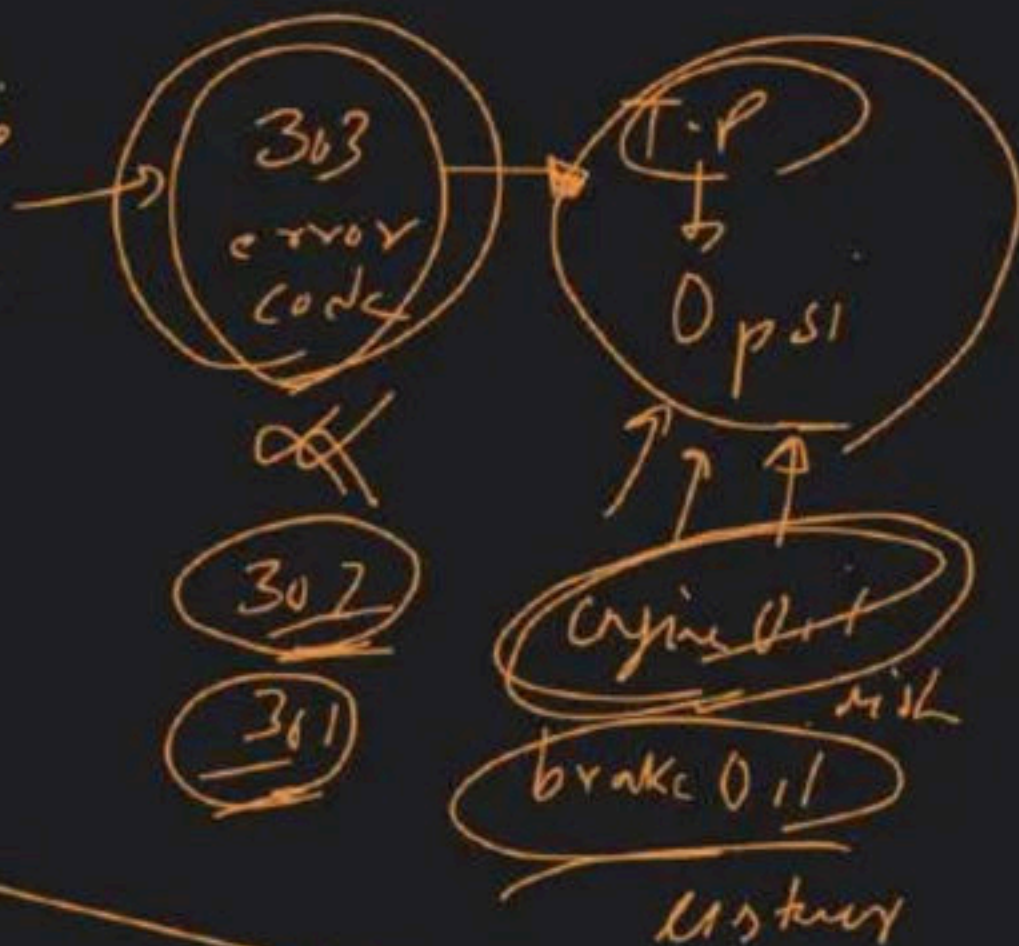


# Desi Example

client



Scoopi  
Type



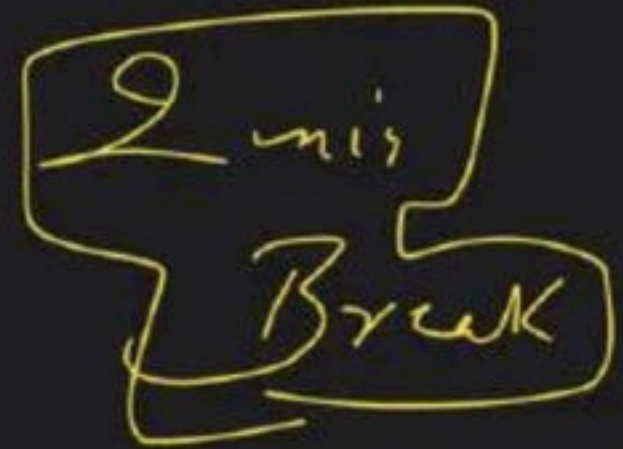
# → Command Pattern:-

focus

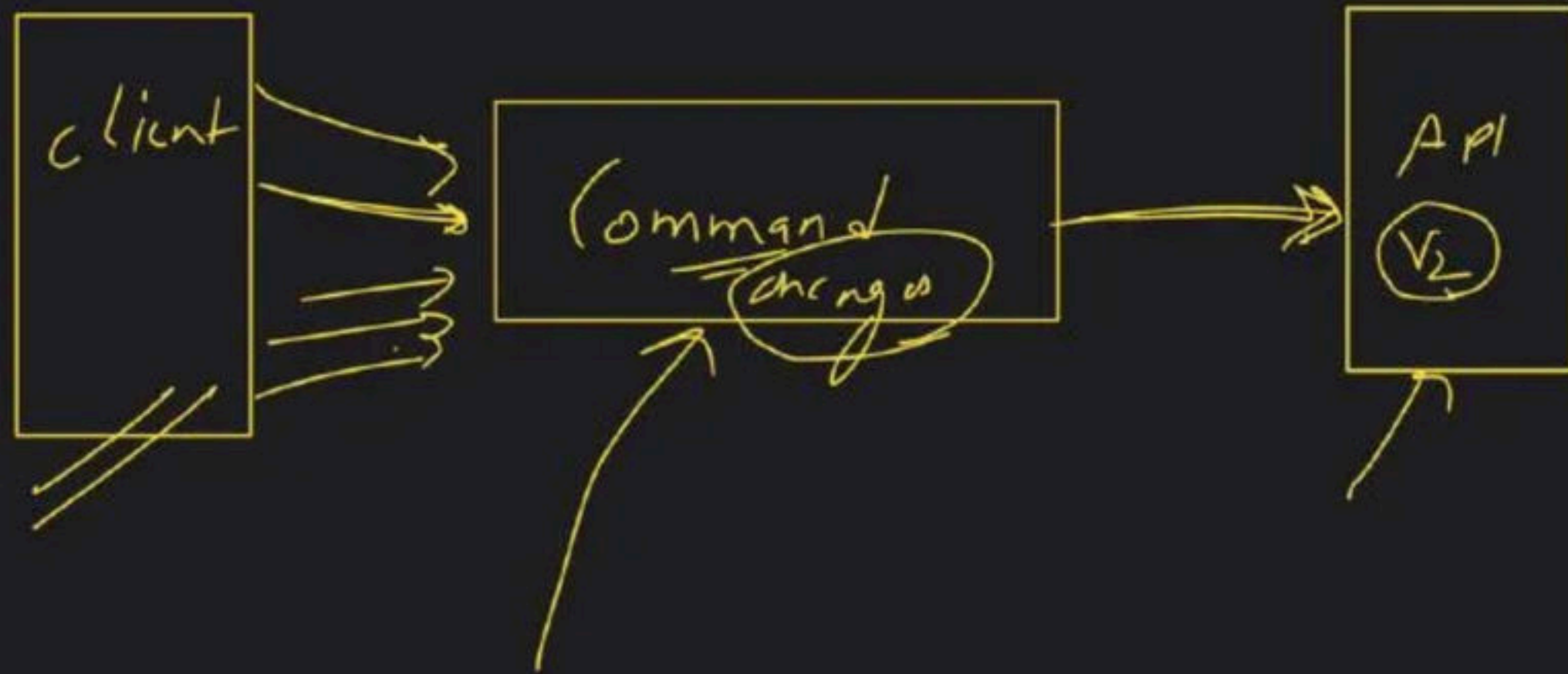
decoupling

object <sup>that</sup> requests

object that fulfills

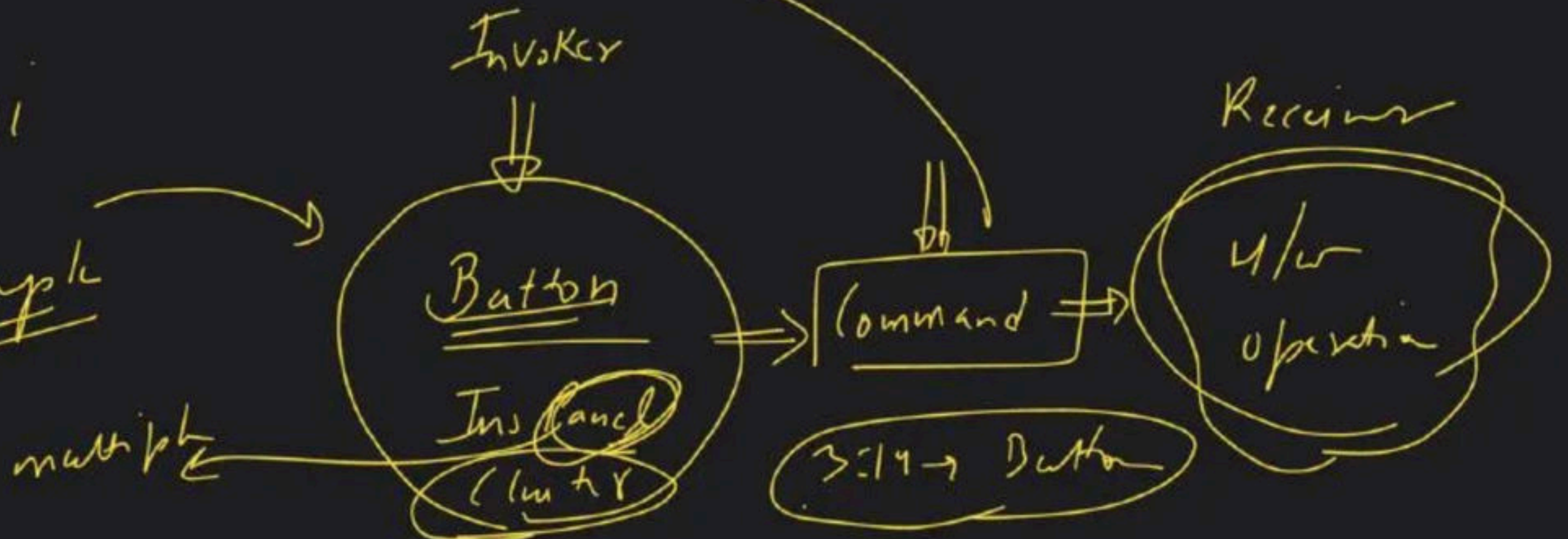








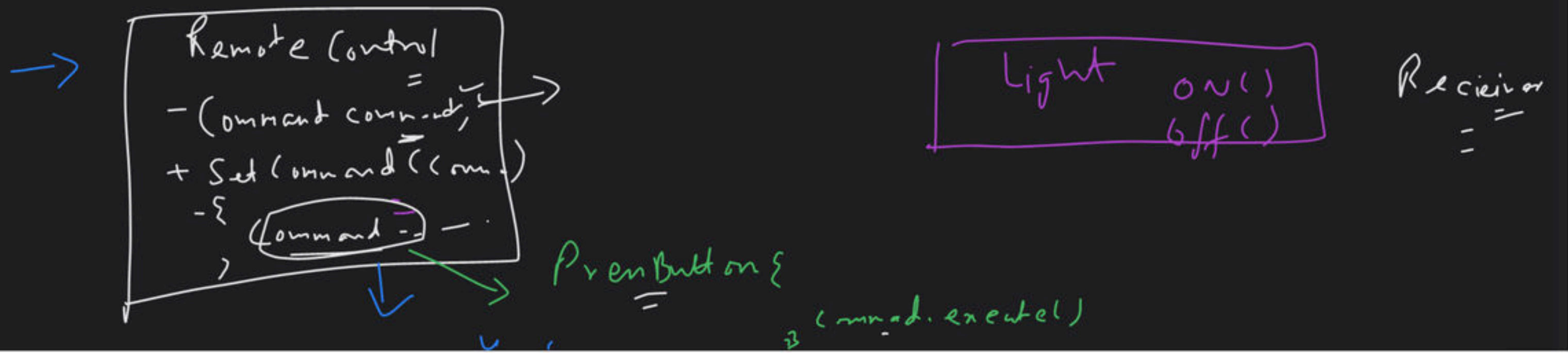
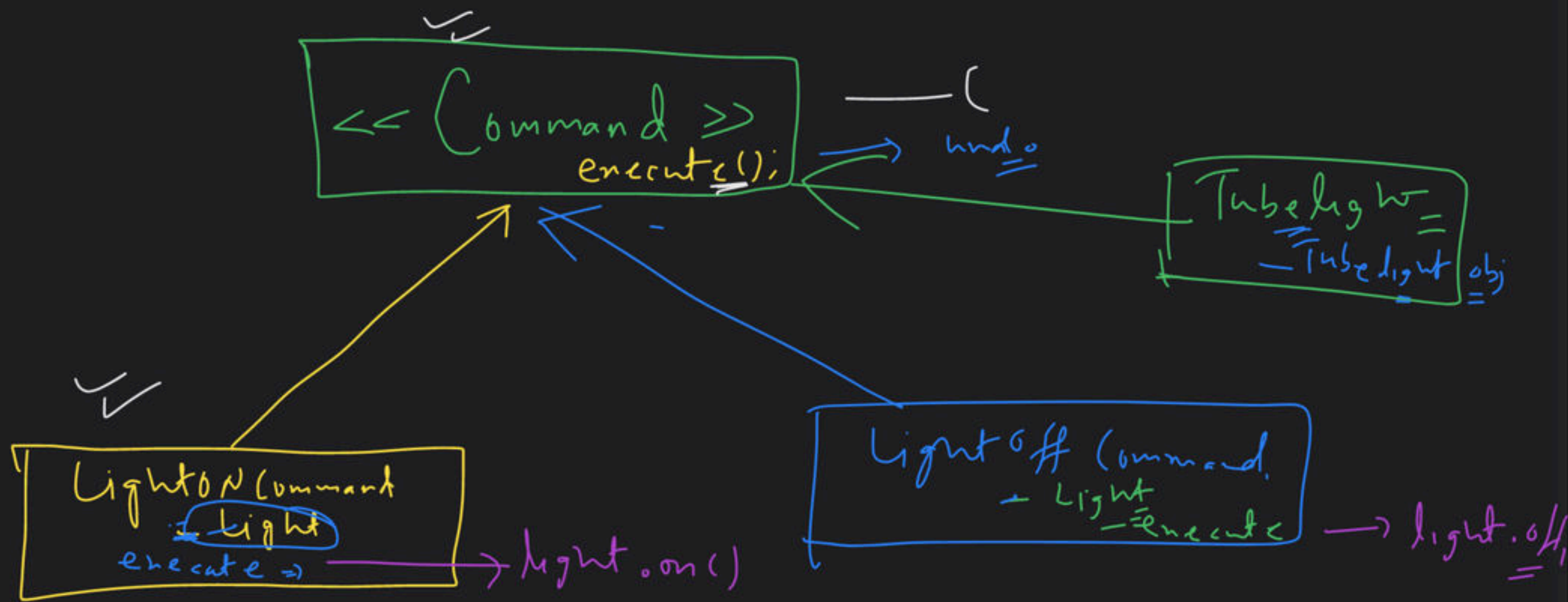
Desi  
Example





⇒ decouple the object that sends a req.  
from object that receives those req.





Client

{

light light = new light(); // Recu

ConcreteCommand //

Command light on = - - - - -;

// " light off = - - - - -;

RemoteControl r = new - - - ; // invoker

r.setCommand( light on );

r.pressButton ( );

}










































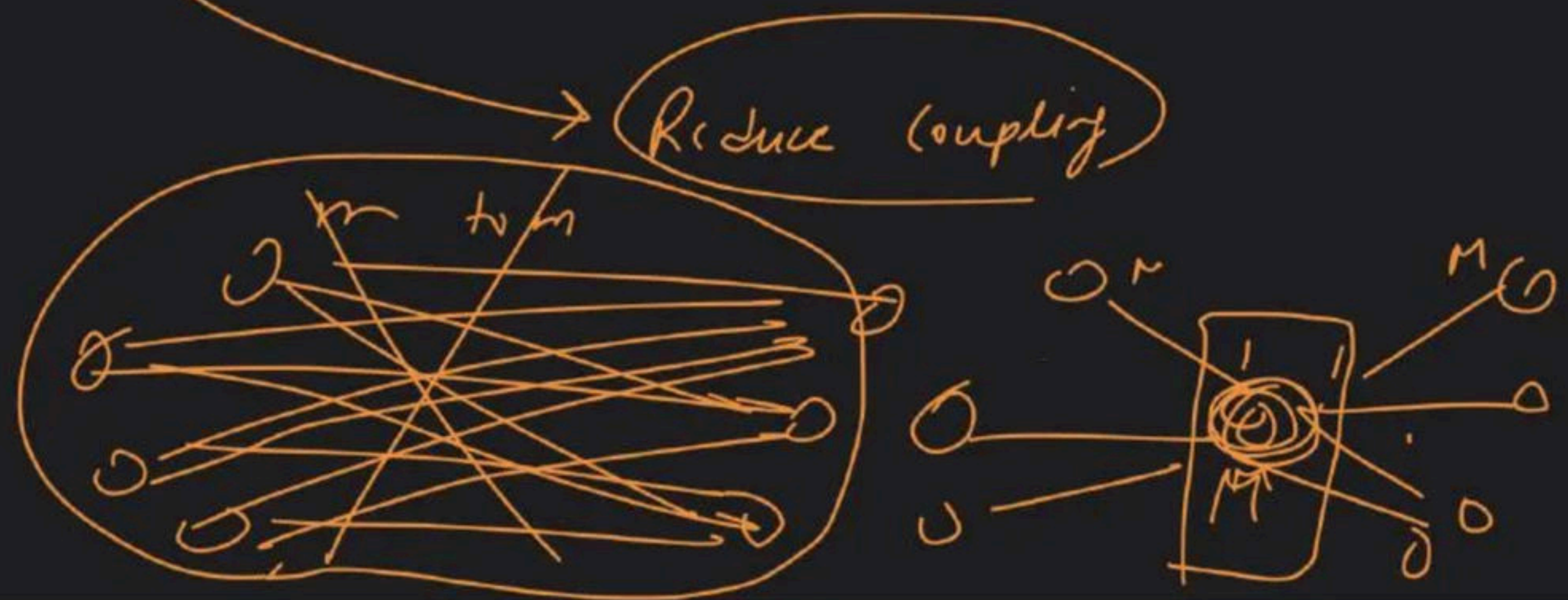
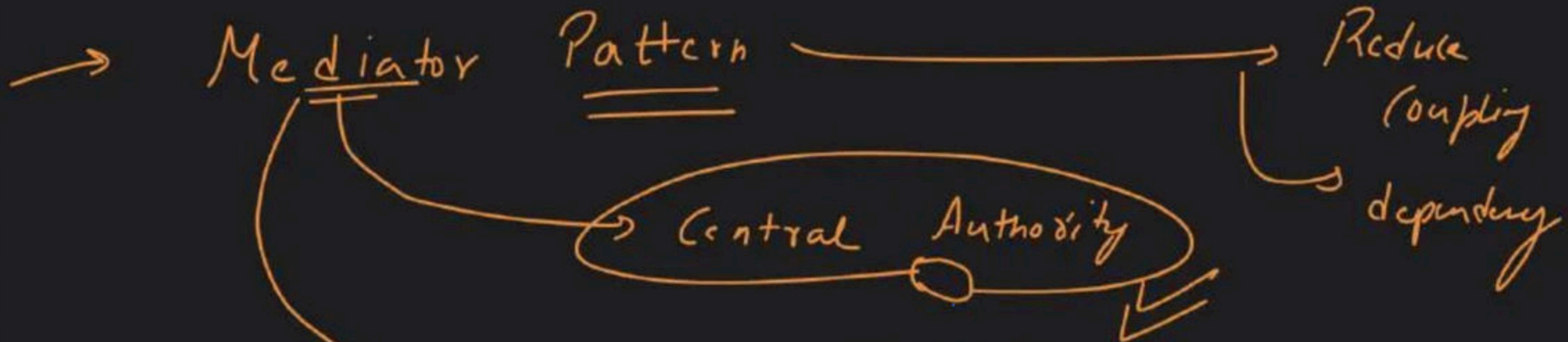




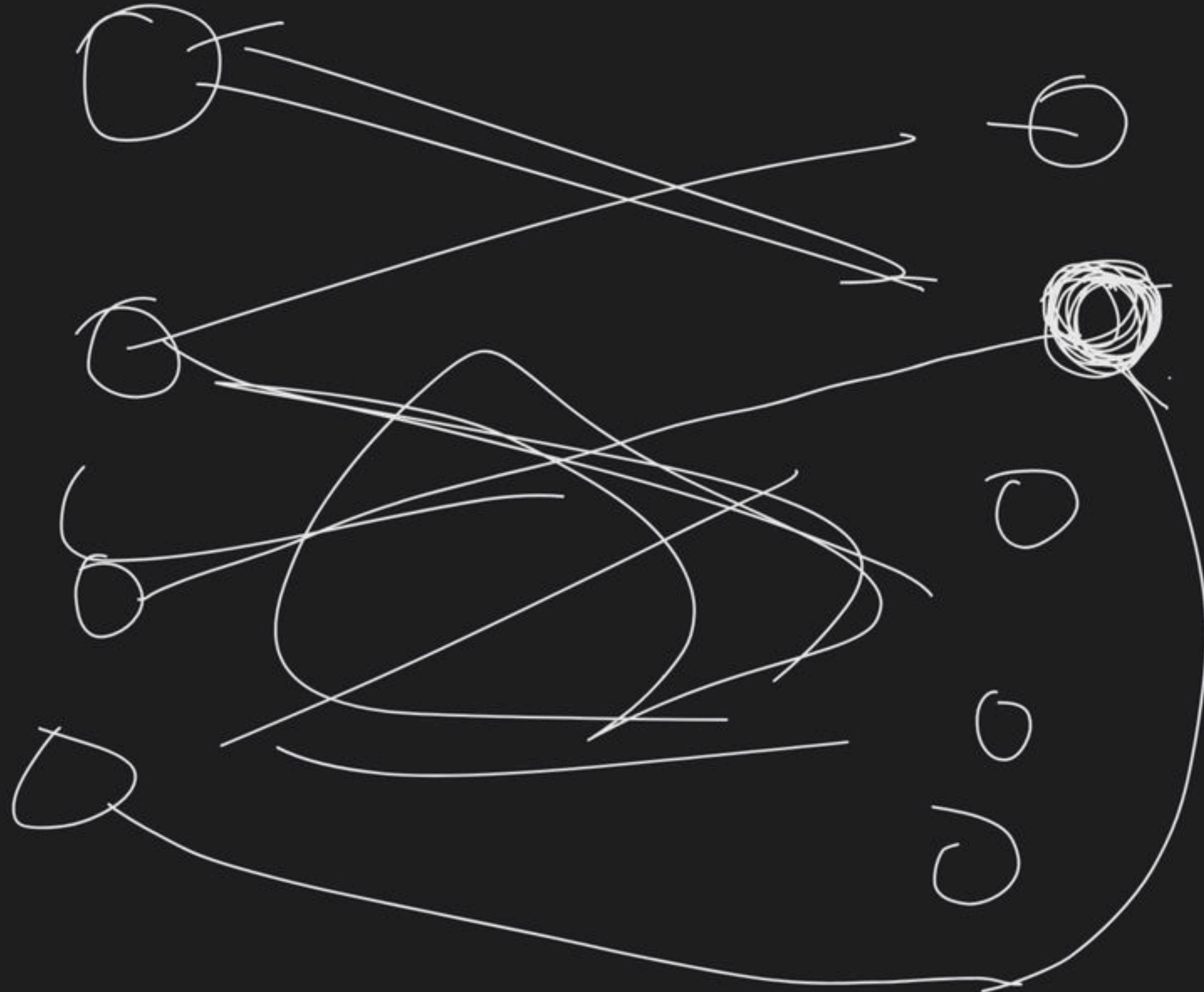


# Let's learn Observer & Mediator Pattern

Special class







⊕

Queue

Connection

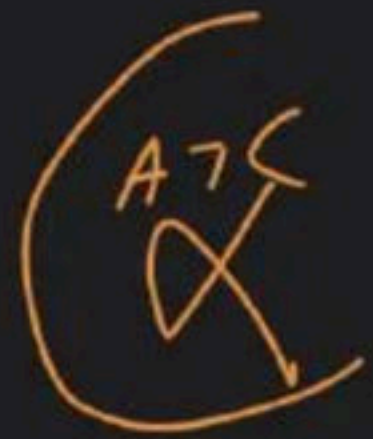
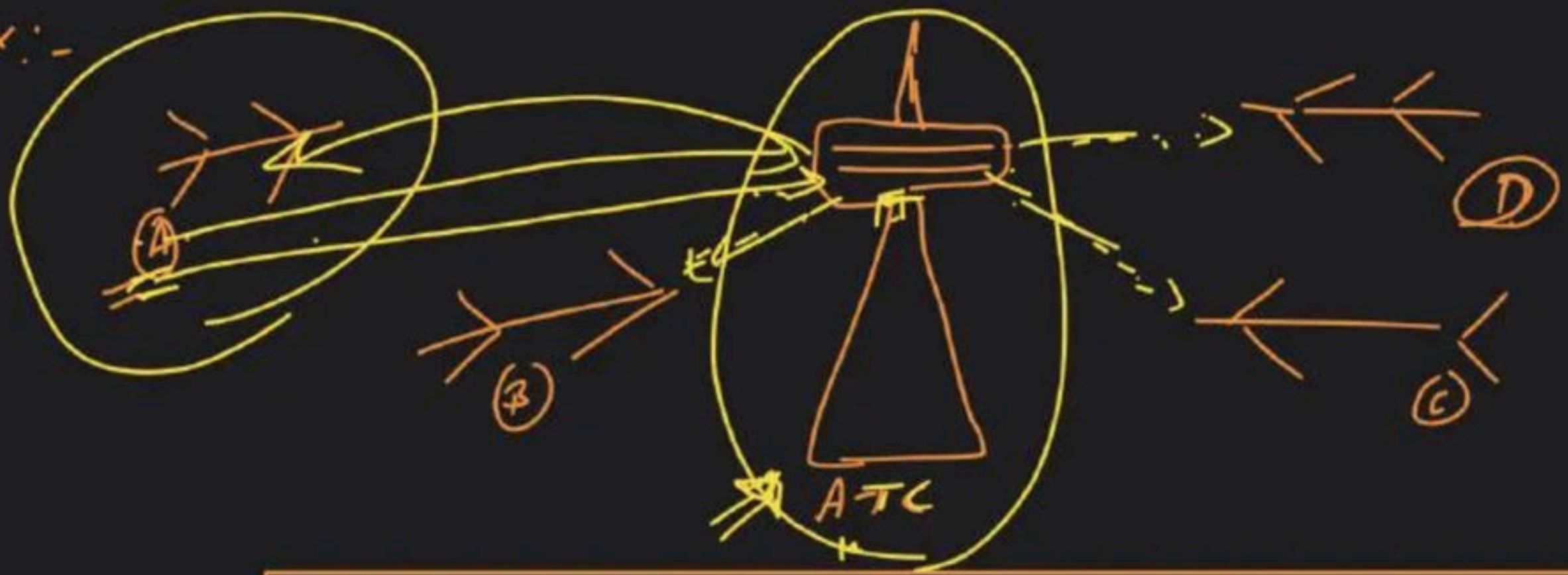
TCP/IP



File ptr

Link

Ex:-



Sign

Sync



I Aircraft

Indigo

Object

Plane 1 - - - Plane 4

AT C Tower = problem

perm()

land()

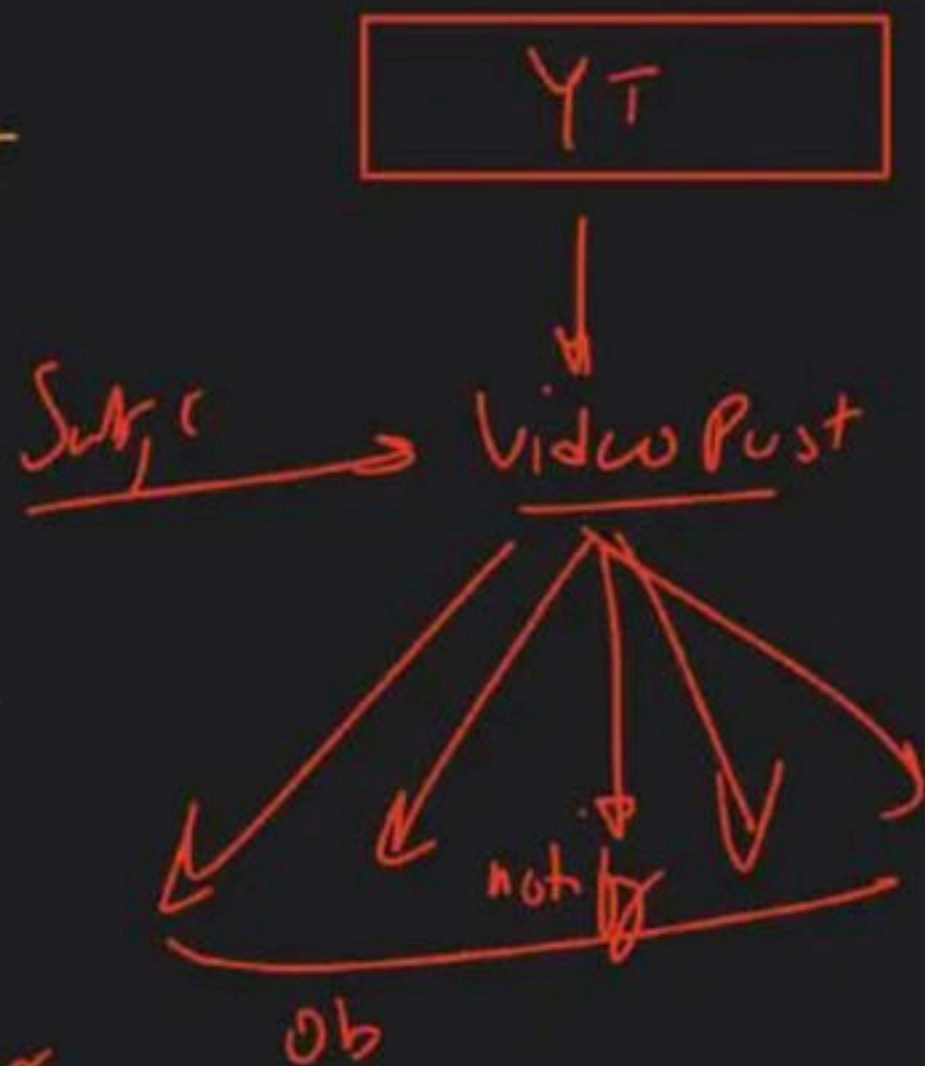
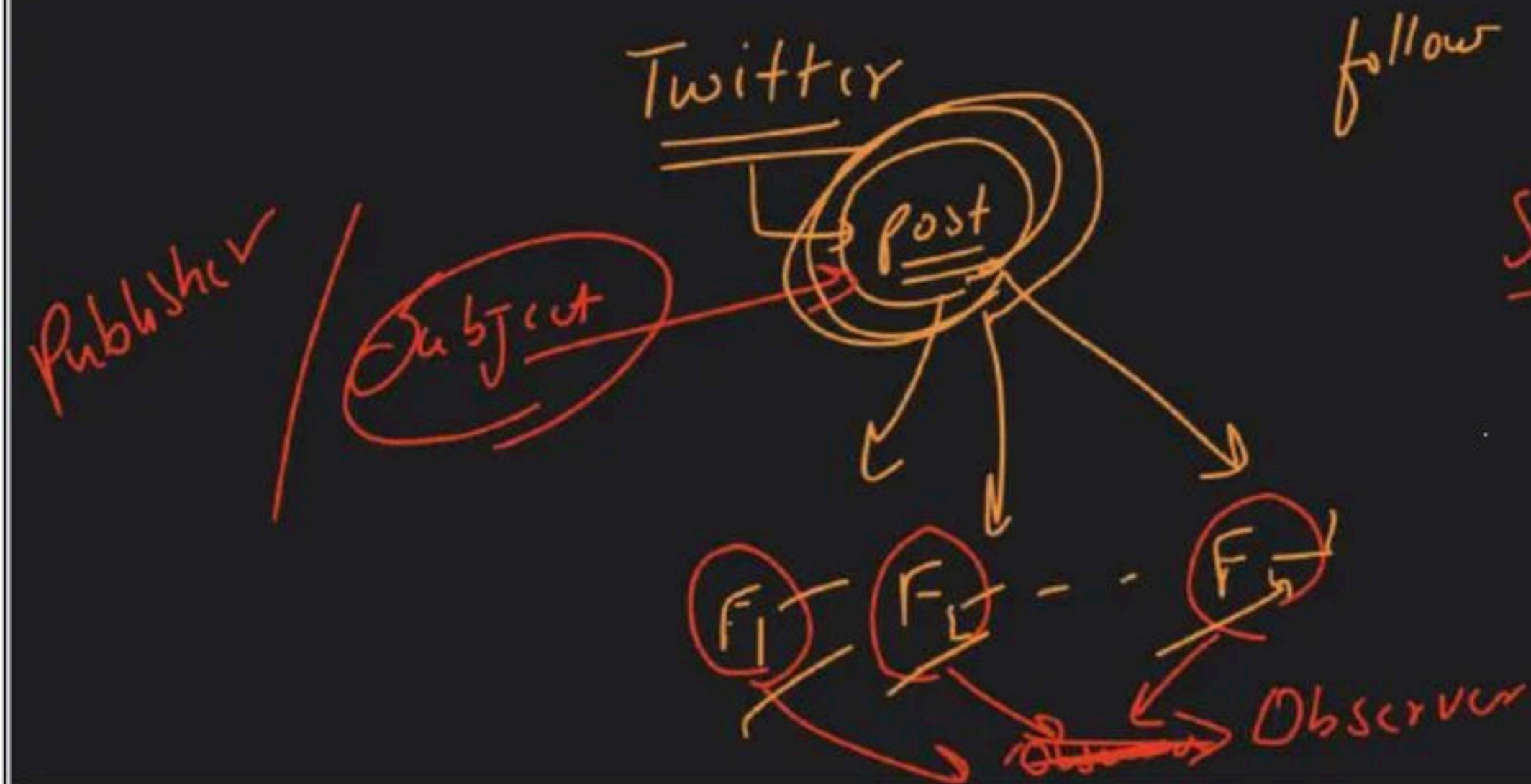
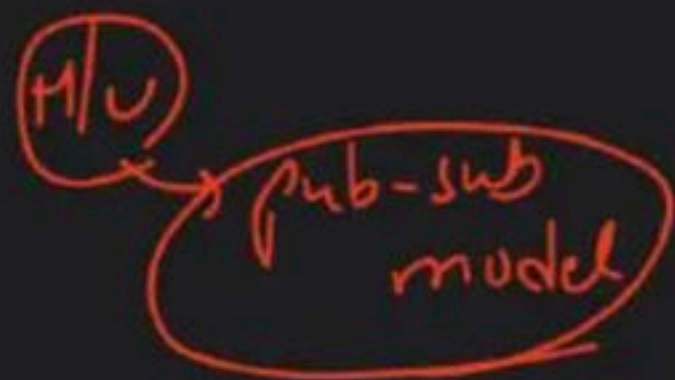
Landing Key

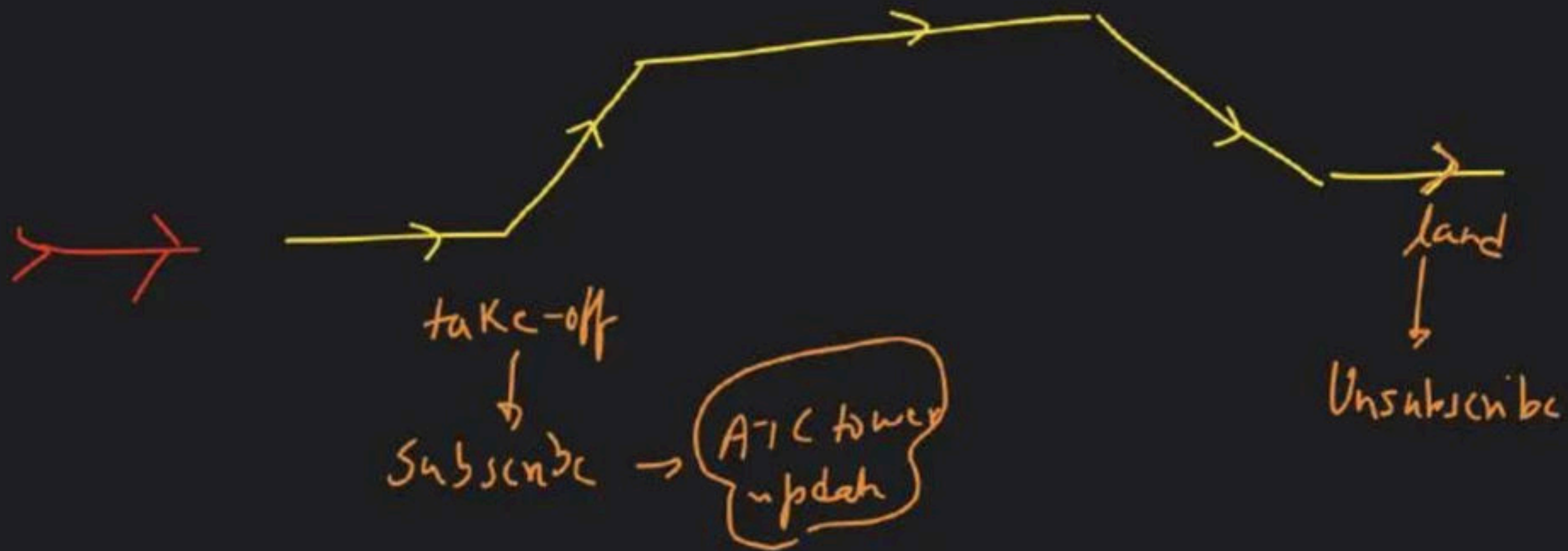
mediator

→

# Observer Pattern

Ex



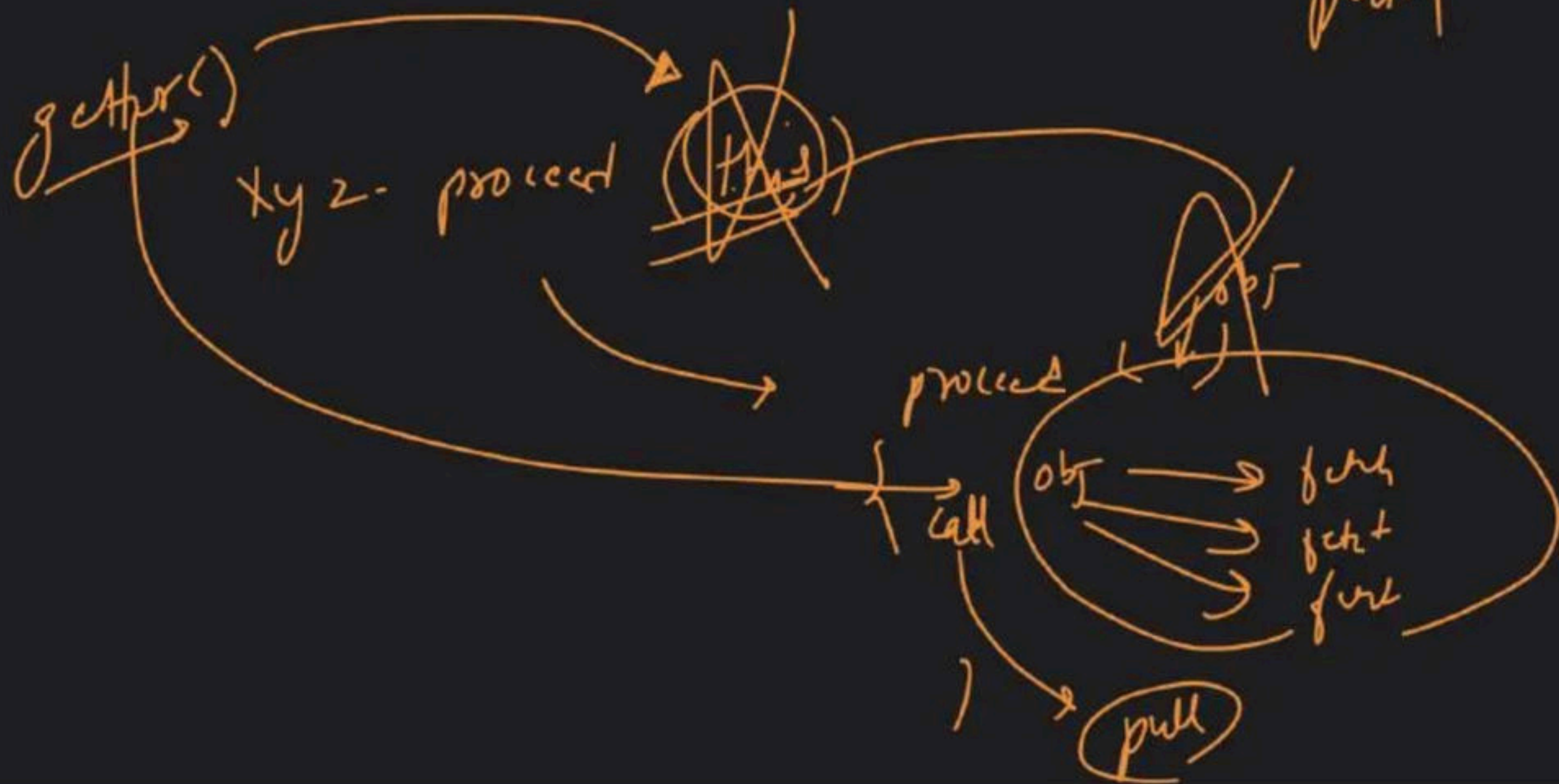




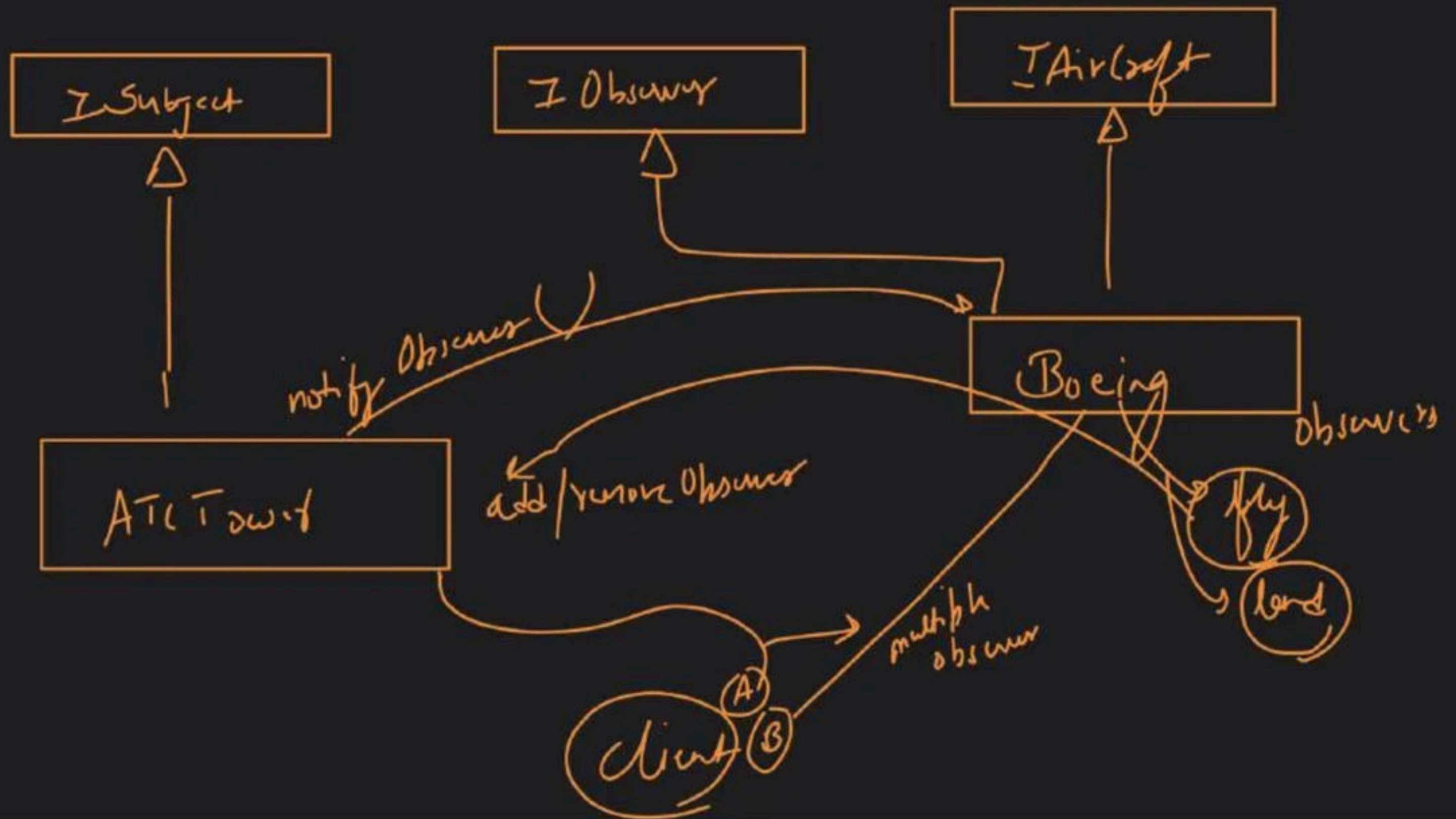
push

2 pull

push









































































# Let's learn Iterator & Visitor Pattern

Special class

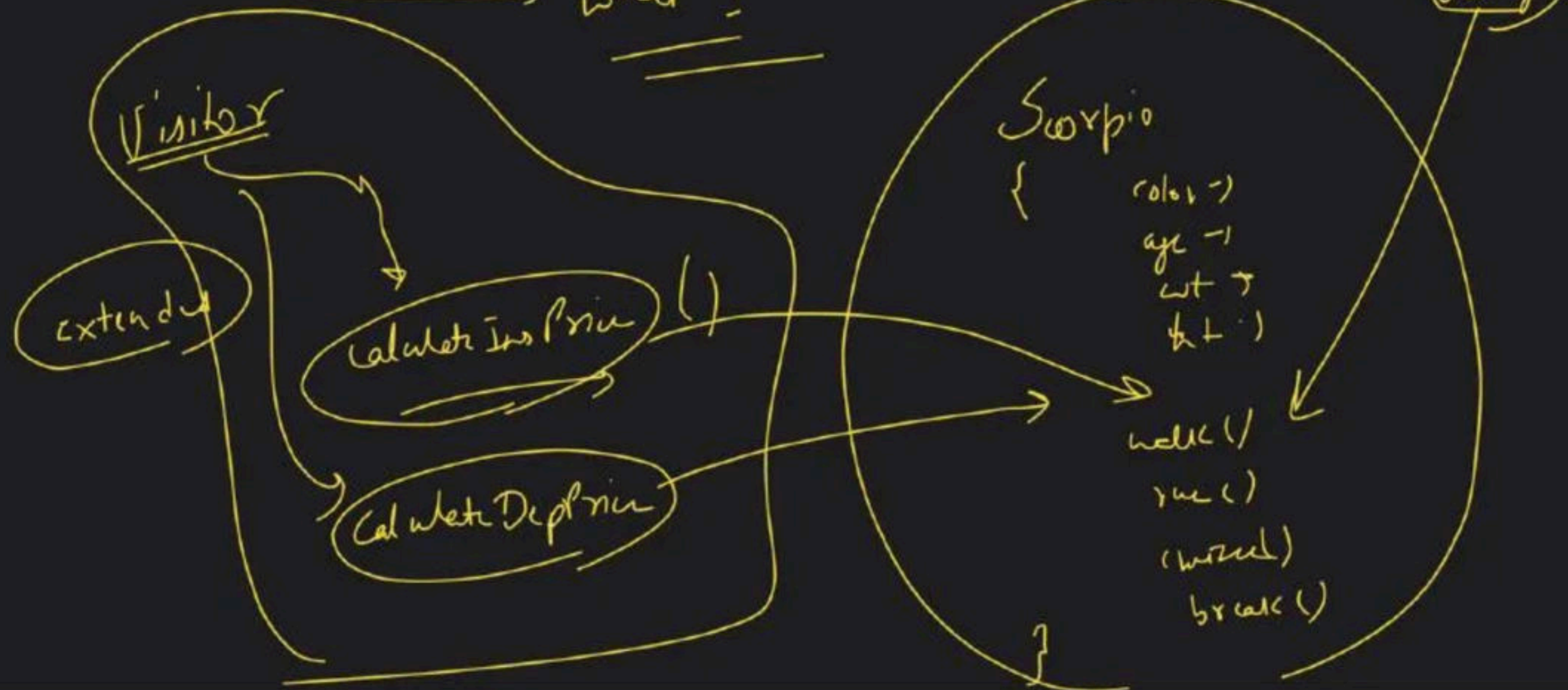
→ Iterator pattern:-  
perform repeatedly

Standard way  
to iterate over  
the collection  
w/o showing the  
underlying implem

container  
(Collection)  
~~vector <~~  
~~stack <~~  
~~Queue <~~  
LL <

→ Visitor Pattern → visitor

what?





Single



Container  
T

Airplane

direct

Traverse

ArrayList <> jets  
[ ] → heli  
LinkedList → cargo

1 Double Dispatch

2 min  
Break

Eng ✓

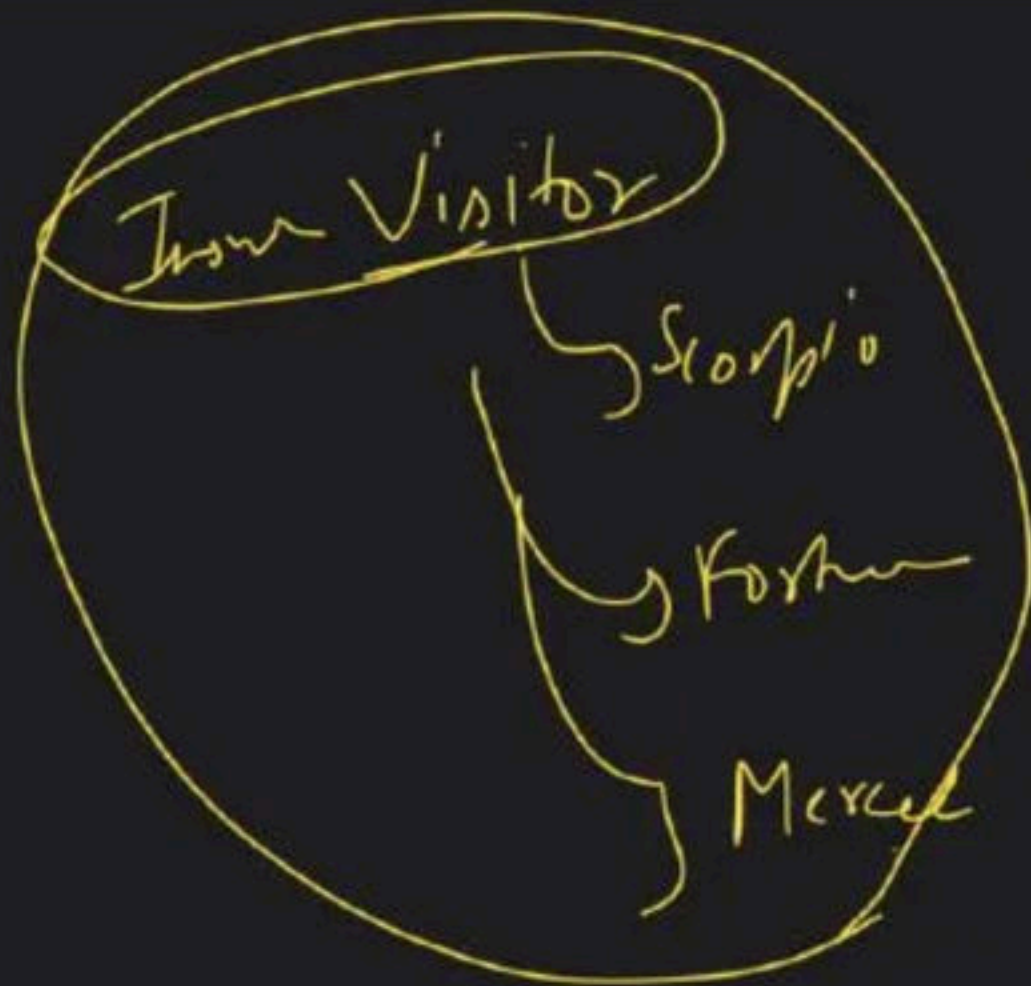
Phy ✓

Chem ✓

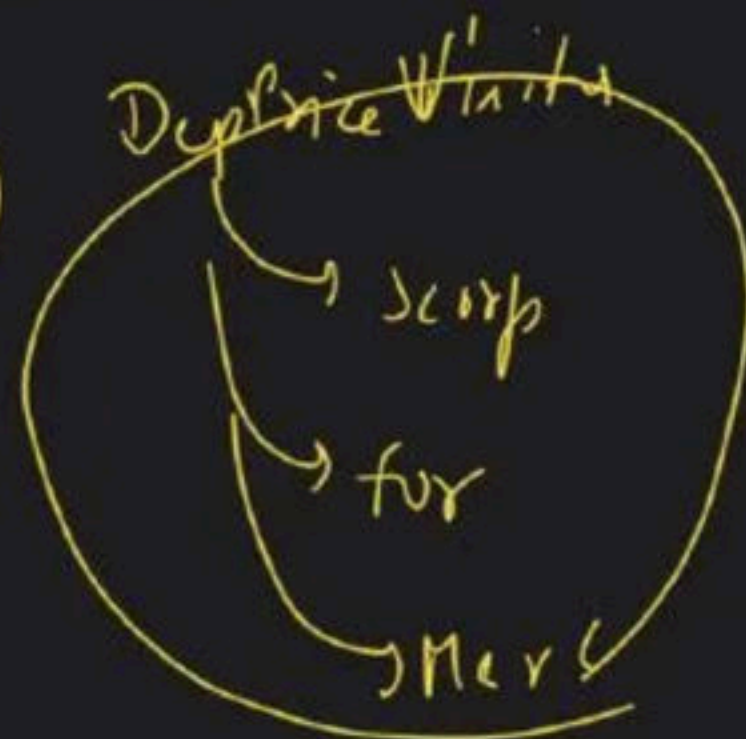
Math ✓

P.E.  
Sport

Bakwas

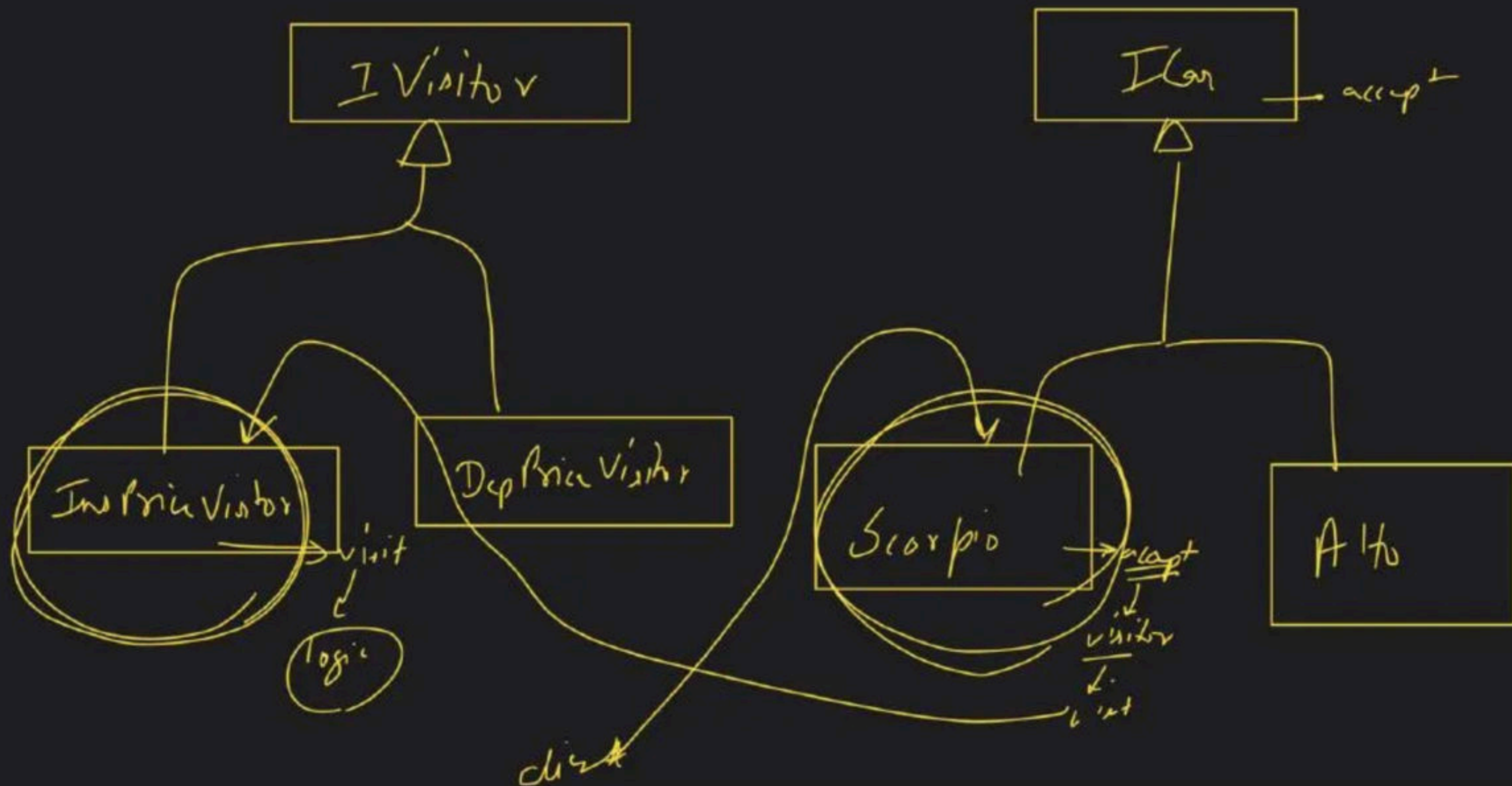


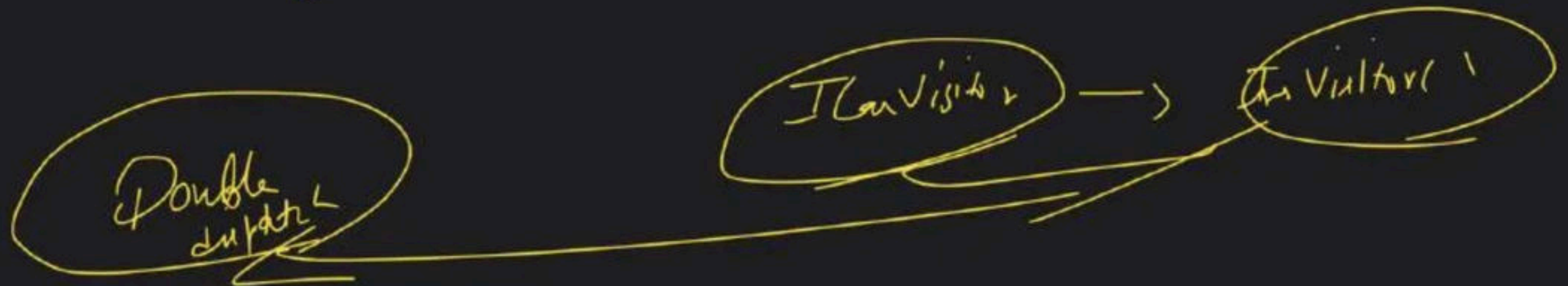
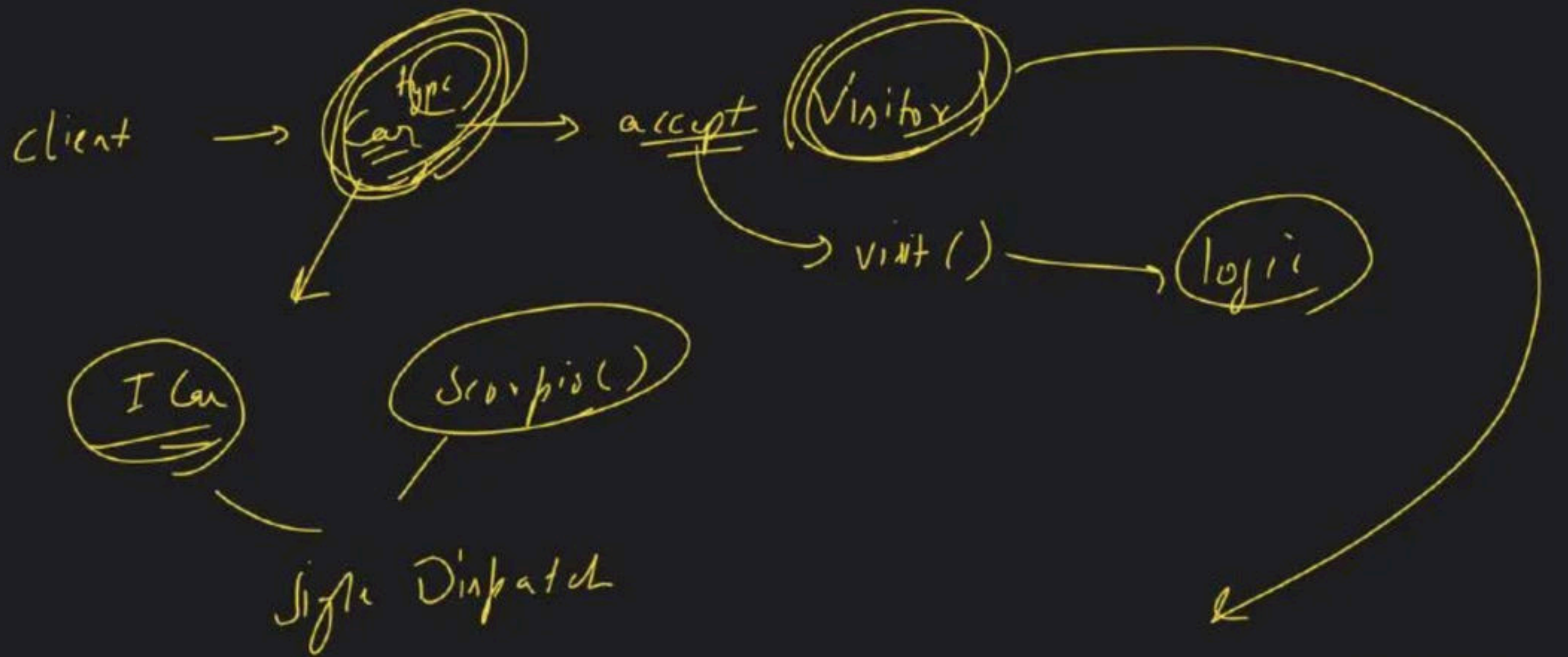
Scorpio

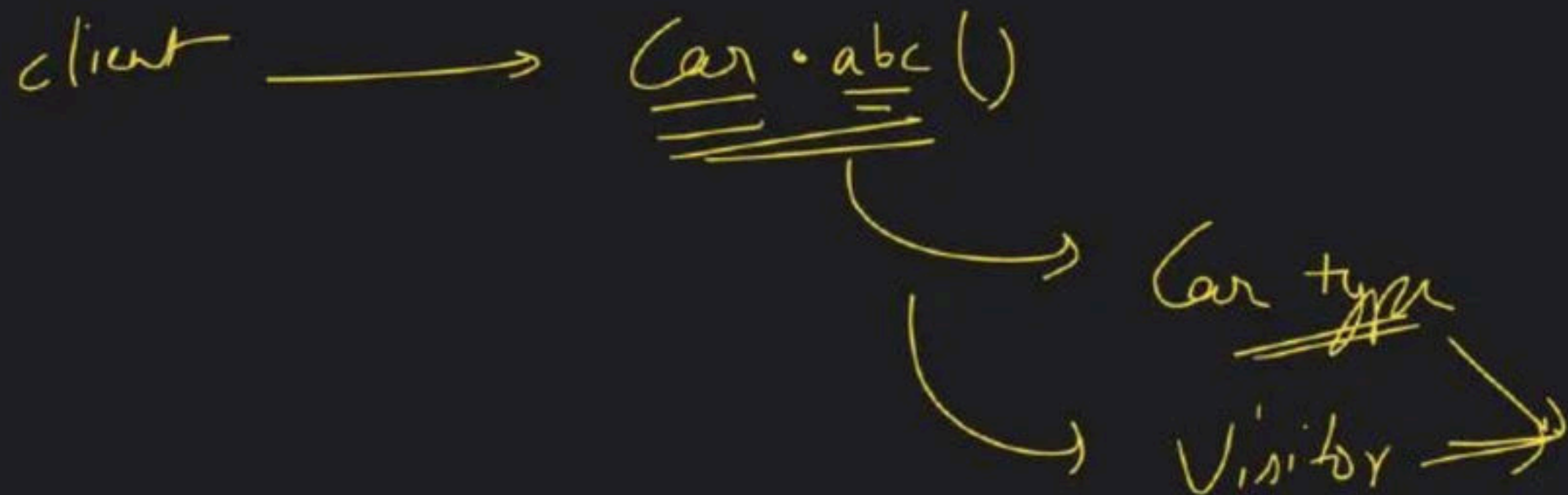




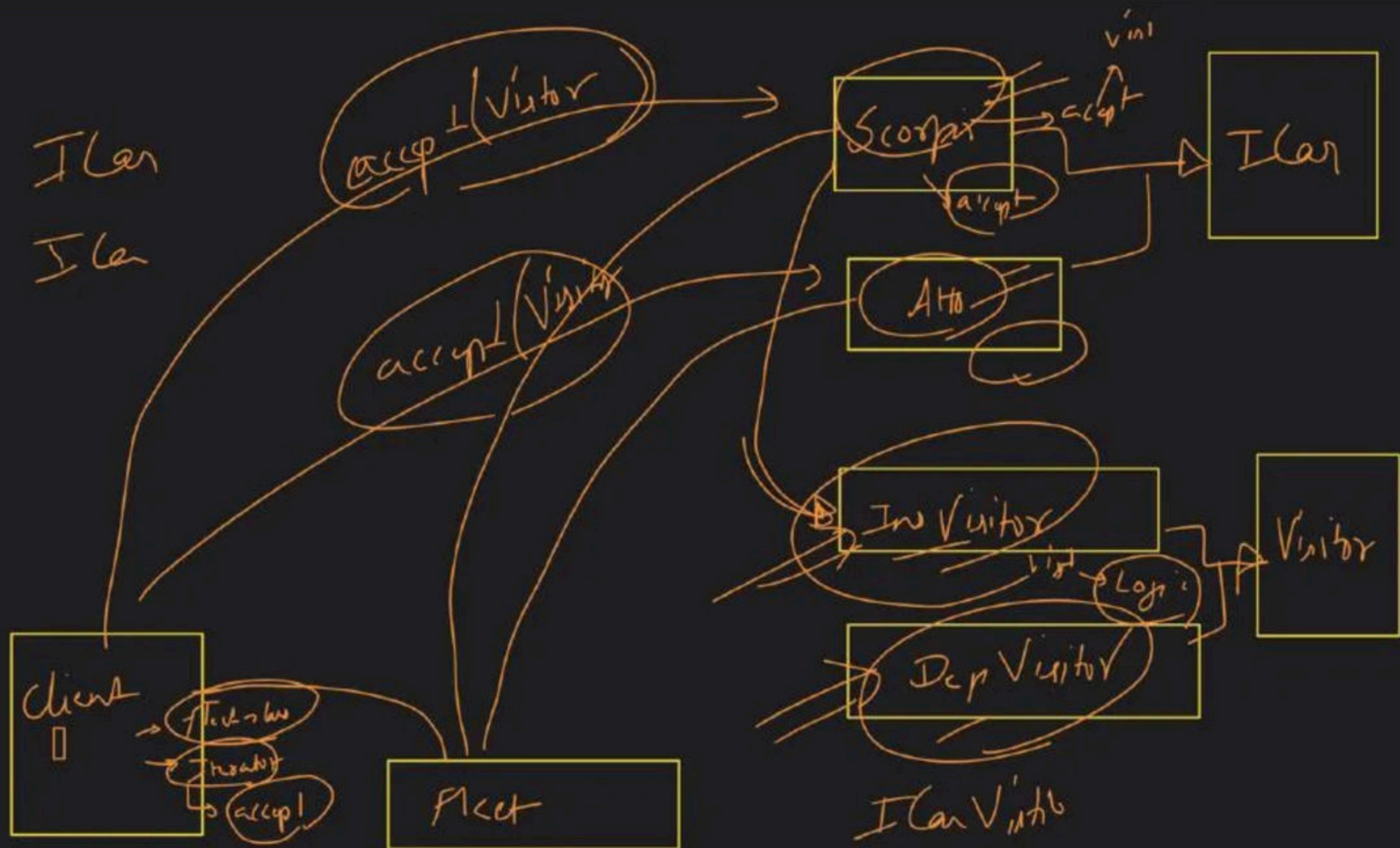












# ⑥ Visitor Pattern

Animal >>  
makeSound();  
Sleep();

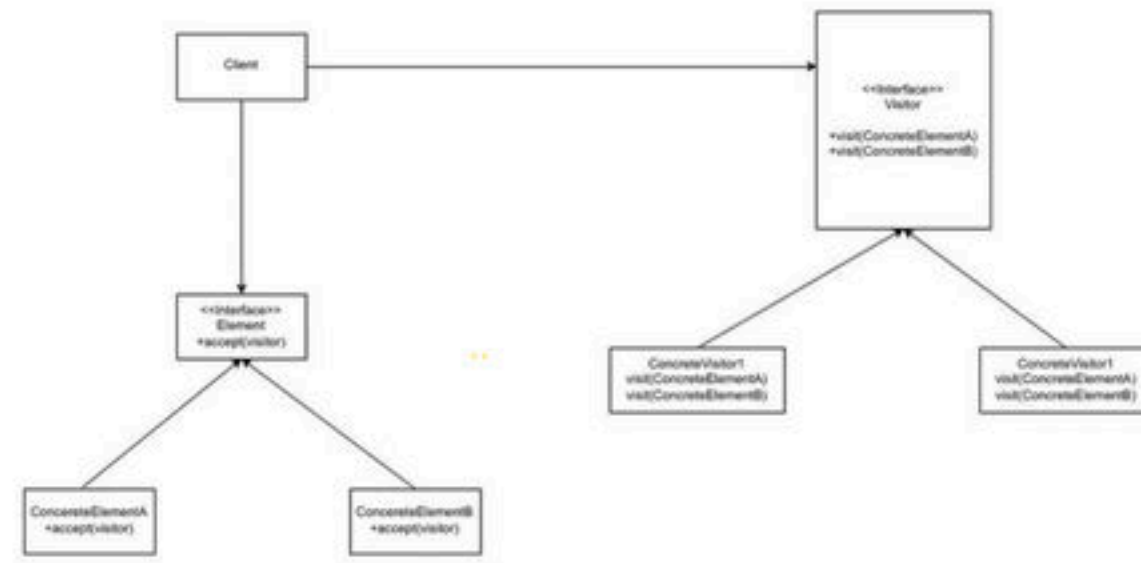
Dog

Cat

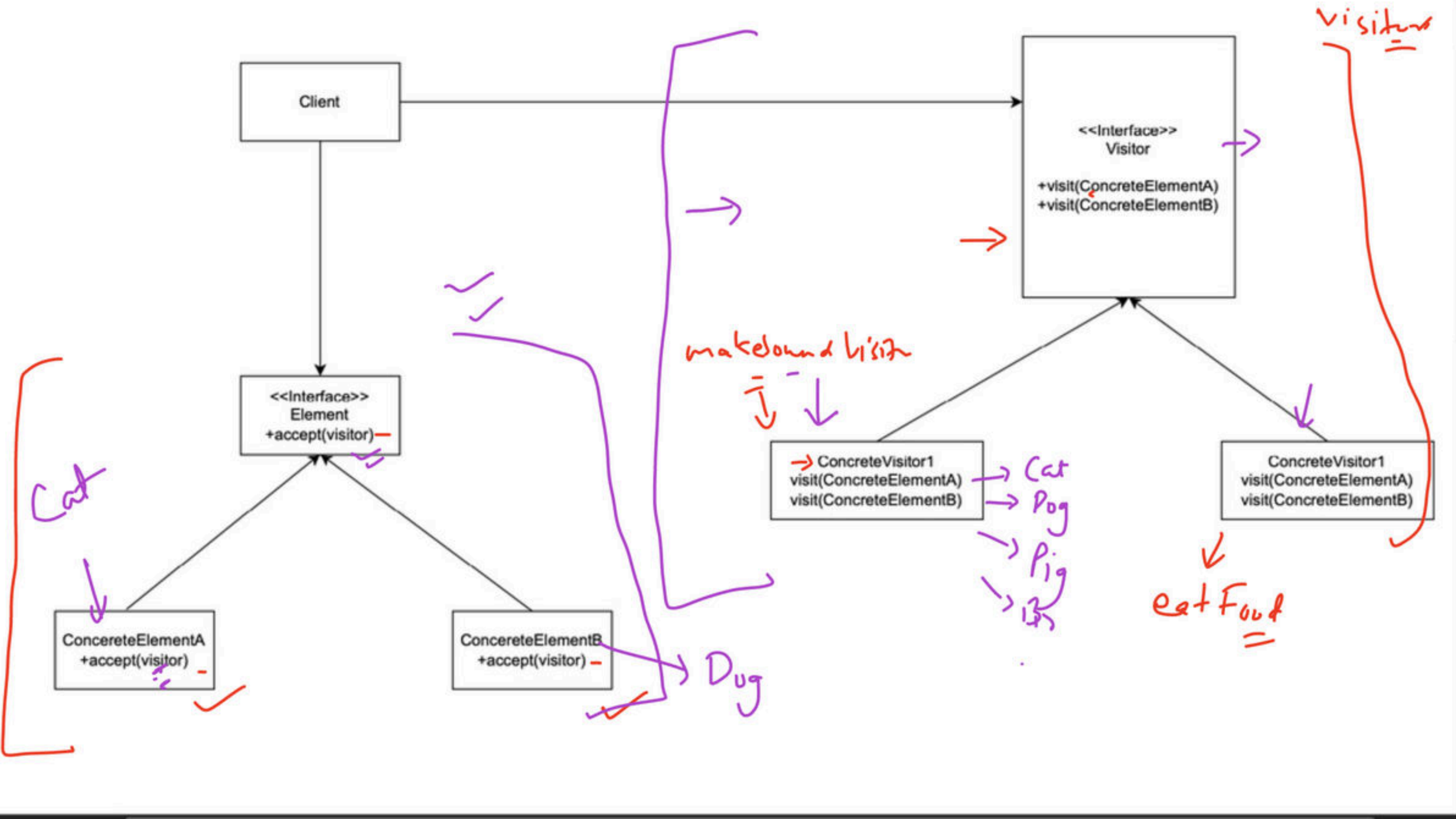
Pig











⇒ obj1 CreditCard

{	Bronze	Silver	Gold
	10%	20%	50%
Gas			
Food	2	3	-
Hotel	-	-	
Other	=	-	-

obj2  
afterType

CrediCard

→ Proccen has CB

P — Not CB

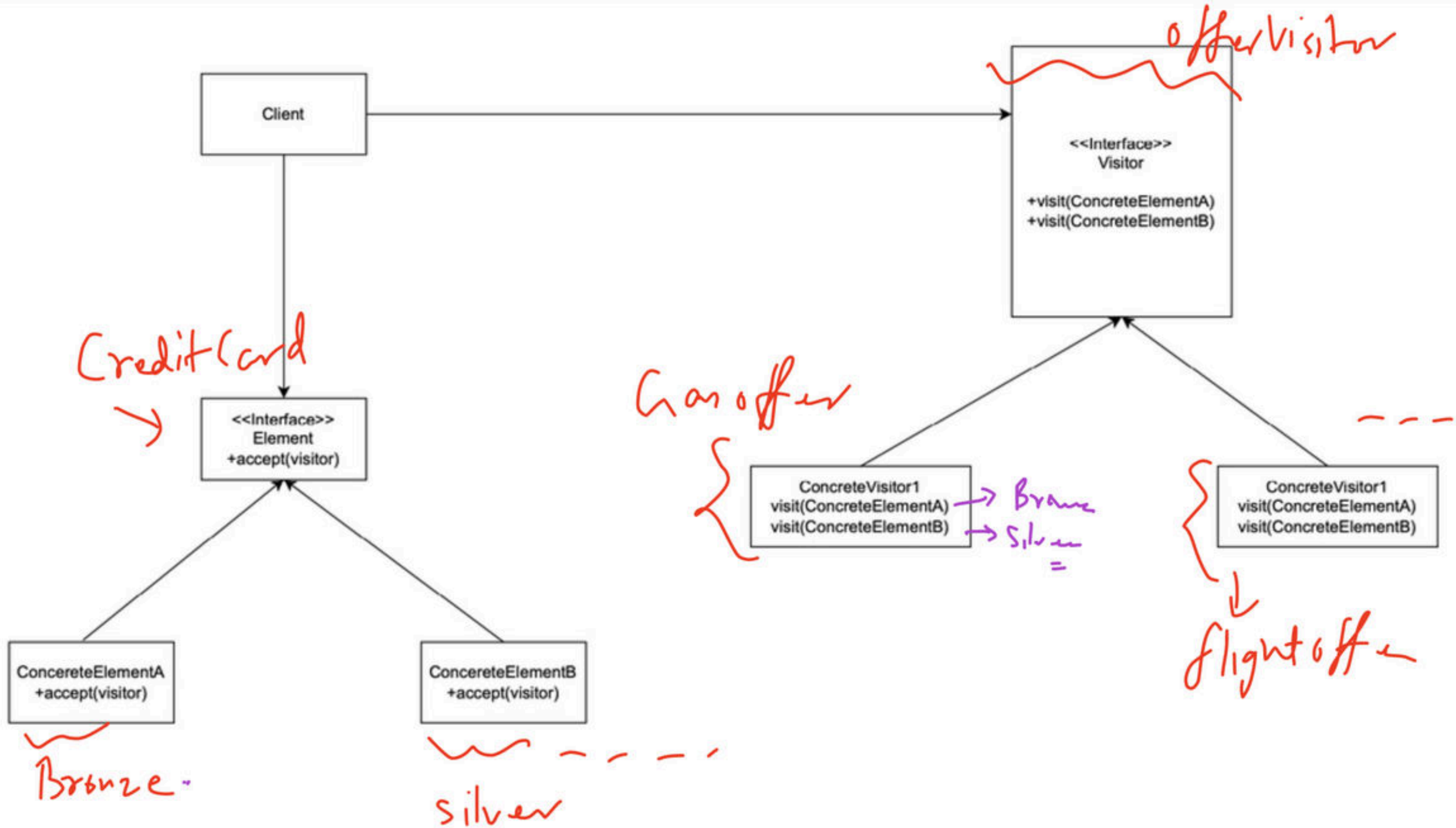
!

→ Bronze  
+ Guest  
→ Food

Silver  
↳ Food

Gold







Association  $\rightarrow$  Abstract Term  
=  $\rightarrow$   $\downarrow$   $\begin{matrix} 1:1 \\ \rightarrow 1:1 \end{matrix}$

Class Profen {  
List < student >;  
= } ✓✓



Aggregation

⇒

Class Library {

List < Student > S;

Library L =

}

new Lib ~ (List < Student > S);

Composition

→ hard age

→ Stronger Bond

Class Computer {

Process ✓ P ;

RAM → Ram ;

SSD → ssd ;

}

→ Ctrcon  
{ P = new Pm  
} R = new Ram

in(Computer c)













































