Lecture 15 : Command Design Pattern

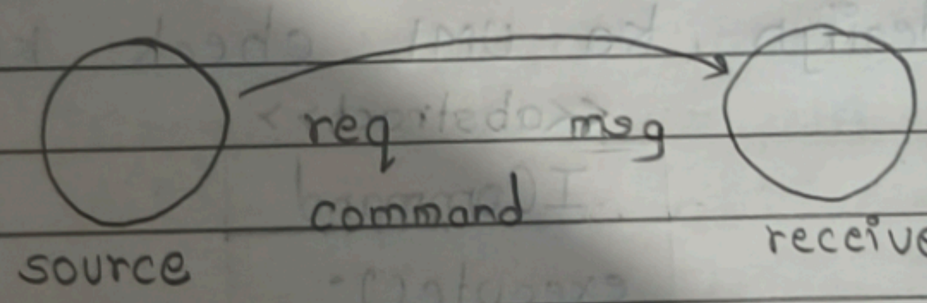
Page No.
Date

Introduction

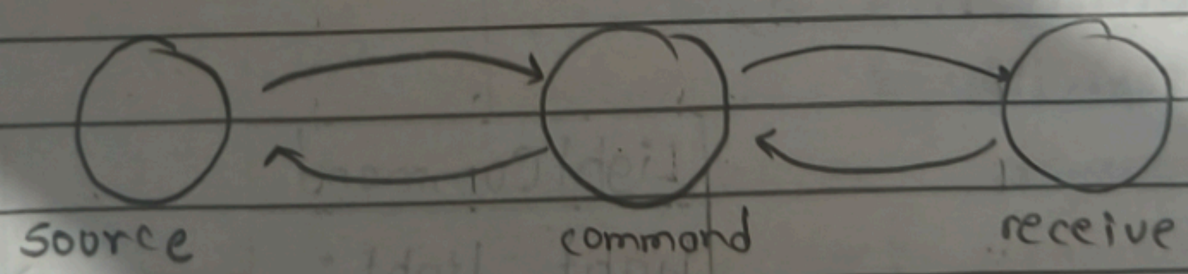
→ This is the most used pattern from very start of our computer life (spoiler : Undo operations supported). It is the most simple pattern.

→ Suppose we have to send message / req / command from source to receiver, this can be done by calling receiver's method

→ Ab mein joh message / req / command bhej rahi thi agar usse as a object treat kar diya jaaye toh ; ab ~~wah~~ source → command (obj) ko bolega mere liye kuch karke laa (→ command → receiver ko bolega krne ko



converting to obj

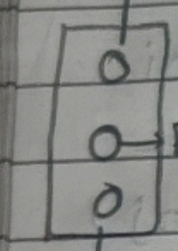


→ Why we made it object? Agar aisa soch rhe ho ki agar OOPS hai toh object hona X

real reason : taaki source & receiver loosely coupled ho and hum log runtime par receiver ke task change kr ske.

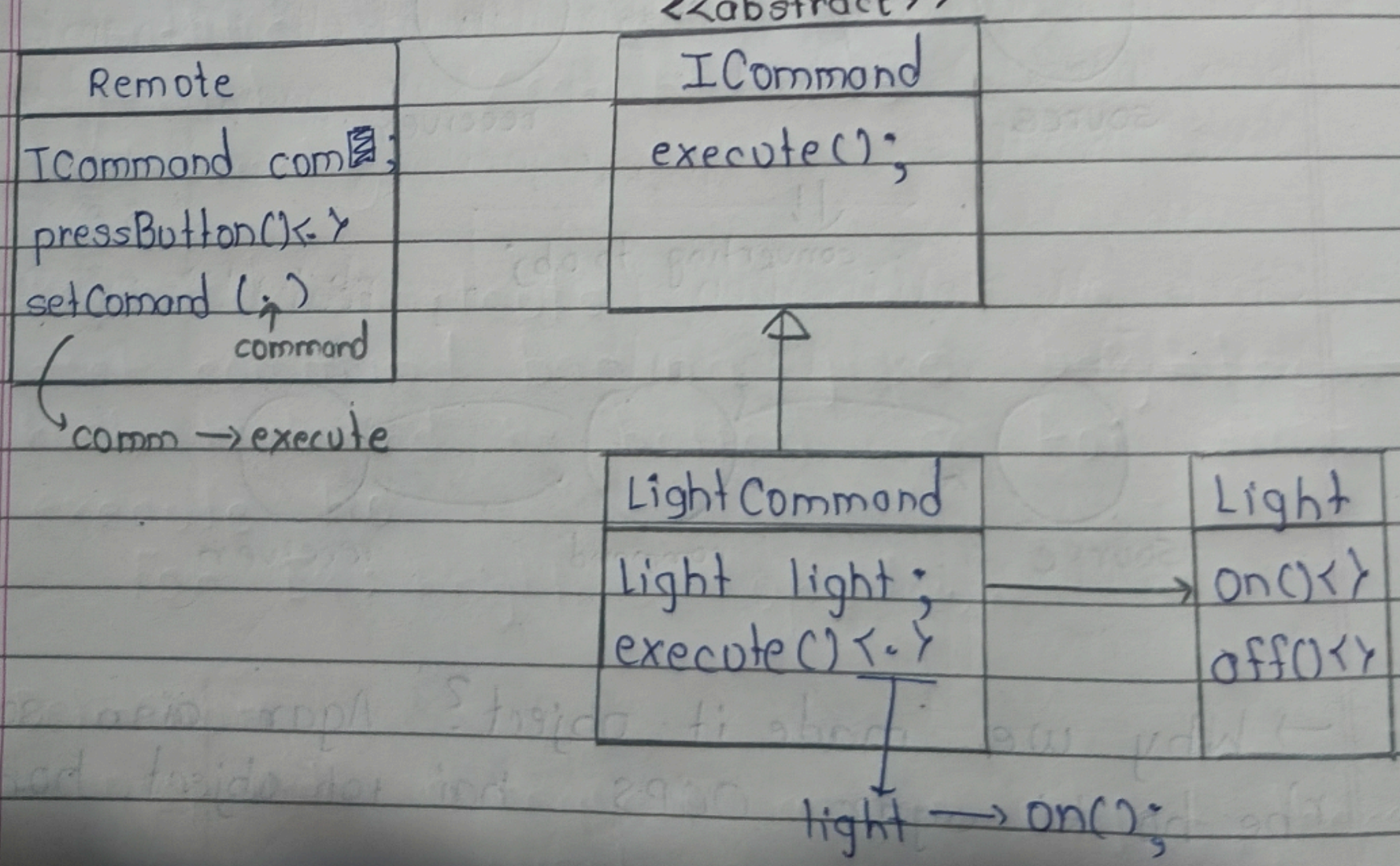
Classic Example of Command Design Pattern :
Smart Home automation

- Smart Home as name suggests hum humare applications ko control kar sakte hai using app/remote buttons,



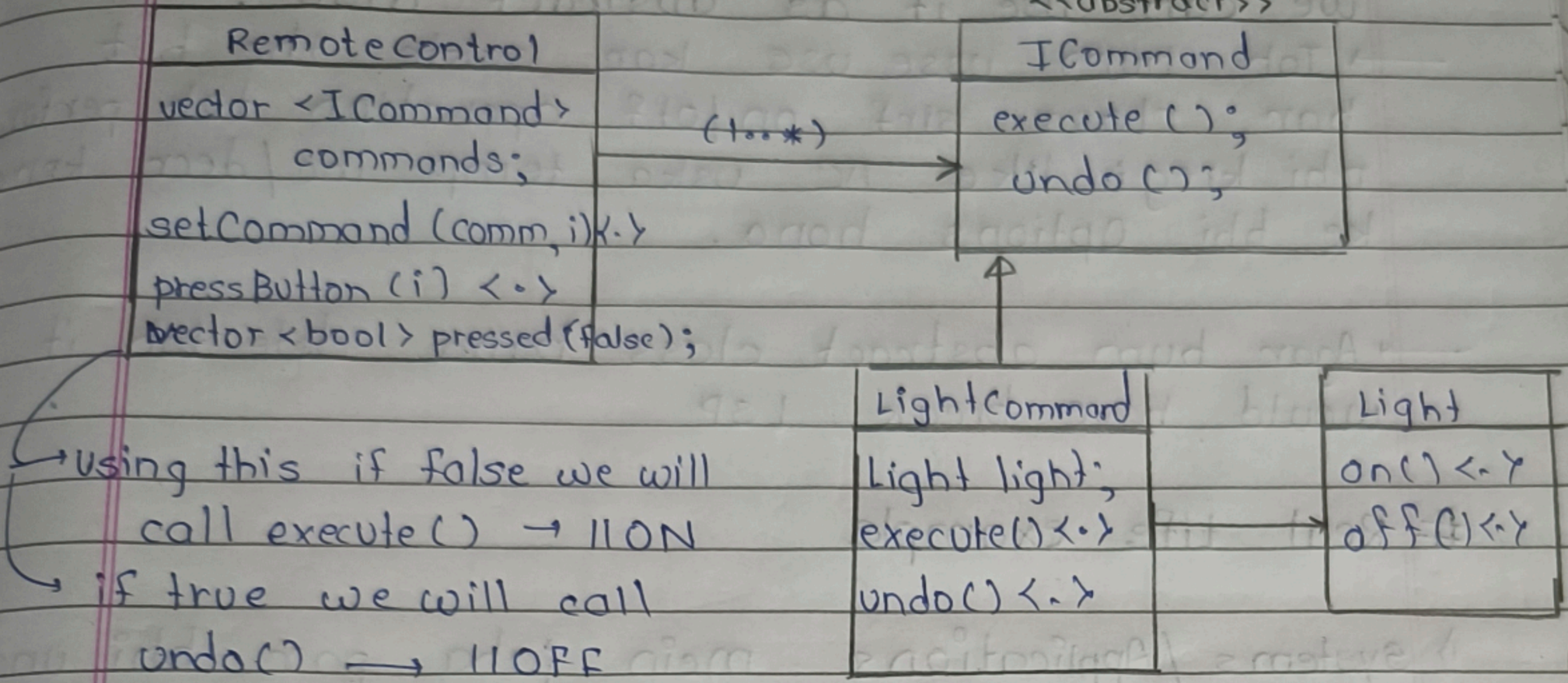
- But ek normal method is button1 ko assign krdo light ko; button2 ko assign krdo fan, button3 to AC and so on. Par ab humara remote and functionality tightly coupled ho gaye infuture agar fan ke jagah mujhe refrigerator operate karana hai toh i need to change remote class ka code → breaks OCP

- Ab optimal design ka UML check karte -



This is for single button → ab incase mujhe fan hota toh I will create concrete class of FanCommand and Fan class & call its function.

Optimal UML Diagram with Undo feature



↳ using this if false we will call execute() → ||ON
↳ if true we will call undo() → ||OFF

Doubts about Pattern

1] Why ICommand ~~has~~ do not have 'has-a' relation with Light

→ Ab hum waha bhi bana skte but we should follow class intent approach yaane ~~jo~~ class ka kya kaam hai uss according relations.

→ Ab yeh job ICommand class hai it do not requires has-a relation kyuki uska intent bss execute & undo karna ~~agar~~ uske pass reference bhi nahi hai kuch execute karne ka.

→ Agar hum usi ko reference dede toh humari class tightly couple ho jayegi & break kregi OCP ko coz har button ke liye alag woh class banani padegi joh khudke methods ko undo/execute kare. Uska kaam bss task krna hai joh class use inherit kregi woh decide kregi ki task konse appliance par hona chahiye.

Date

2] Humne light class ko concrete kyu banaya we can use it as abstract also ?

→ Toh hum usse use kar sakte the but har class sirf on/off operation toh perform nhi kregi like AC uska temp inc/decr krne ke bhi options hona.

→ Agar hum abstract class bana dete toh it would have break LSP

Real life example

1) Systems / Applications mein jaha par bhi undo feature ki need ho

2) Text Editor, Photoshop
   - Italic
   - Bold
   - formating

   } can be undo

3) Keyboard shortcuts
→ ek command banana jisse ~~brightness~~ kuch keyboard ke button se koi task execute ho
   For Ex: brightness up/down

Definition
↳ Encapsulate a request as an object, thereby letting you parametrize clients with different request, queue or log request and support undoable operations.

Standard UML