iii) Elevator System

i) Requirments

a) How many lifts should we consider [n]

b) Lift Dispatch Algorithm.

i) Odd/Even → floors
ii) fixed floors
iii) Minimum Seek time [nearest to person]

ii) objects

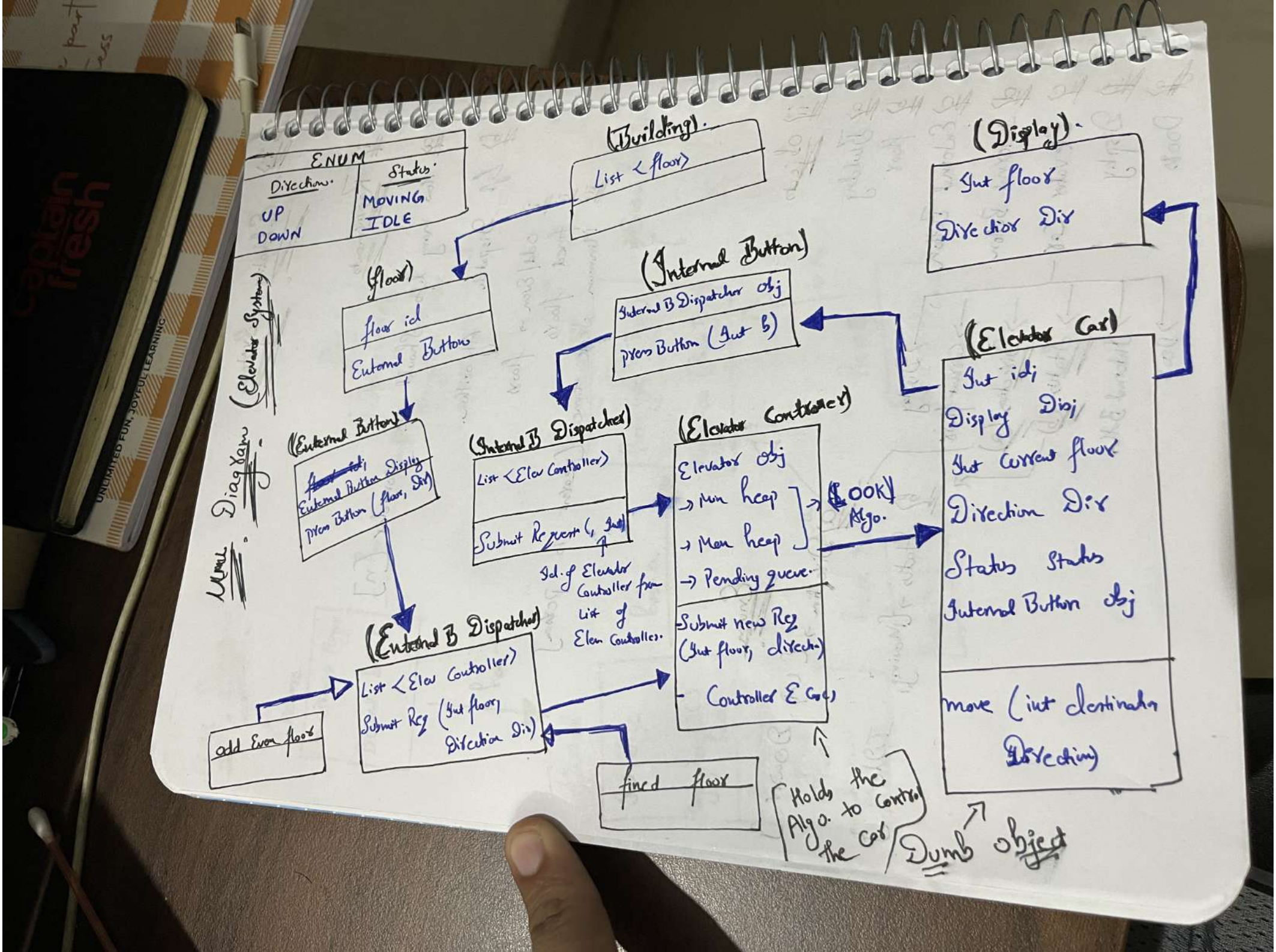
a) Building
b) floor
c) External Button
d) Internal Button
e) Elevator Car
f) Display
g) Doors

Enum

a) Direction → { Up, Down }

b) Status → { Moving, Idle }

Display
Current floor
Direction (a)
Status (b)
Internal Button
Door


ENUM
Direction
UP
DOWN
Status
MOVING
IDLE
(Building)
List <floor>
(Display)
Int floor
Direction Dir
(Floor)
floor id
External Button
(Internal Button)
Internal B Dispatcher obj
press Button (Int b)
(Elevator Car)
Int id;
Display Disj
Int current floor
Direction Dir
Status Status
Internal Button obj
move (int destination
Direction)
(External Button)
External Button Display
press Button (floor, Dir)
(Internal B Dispatcher)
List <Elev Controller>
Submit Request (, Int)
(Elevator Controller)
Elevator obj
→ Min heap
→ Max heap
→ Pending queue.
Submit new Req
(Int floor, direction)
Controller E Code
LOOK Algo.
Id. of Elevator Controller from List of Elev Controllers.
(External B Dispatcher)
List <Elev Controller>
Submit Req (Int floor,
Direction Dir)
add Even floor
find floor
Holds the Algo. to control the car
Dumb object
(Elevator System)
Diagram.
Uml.

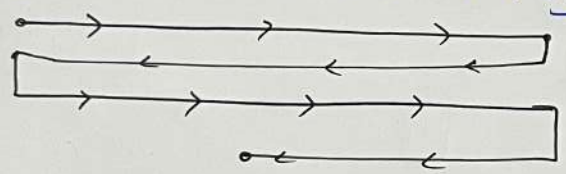
iv) Algorithm

a) SCAN

→ In this it fulfills All the request that comes in the same direction, > current floor & goes till the End of the same direction floor before changing the Direction

→ used but not optimal

[0 1 2 3 4 5 6 7 8.]

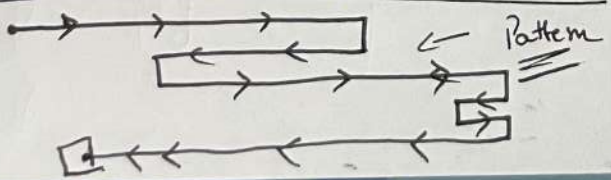


This kind of pattern

b) Look

→ In this, it is basically a upgrade upon the SCAN Algo., as, rather than going till the end, to change the direction, it look, if upon going in the same direction, if there exist an request then fulfill, else, change the direction.

[0, 1, 2, 3, 4, 5, 6, 7, 8]



Pattern

Min heap → UP direction

Max heap → DOWN direction

Pending queue → To save req. out of scope for same direction.