State Design Pattern

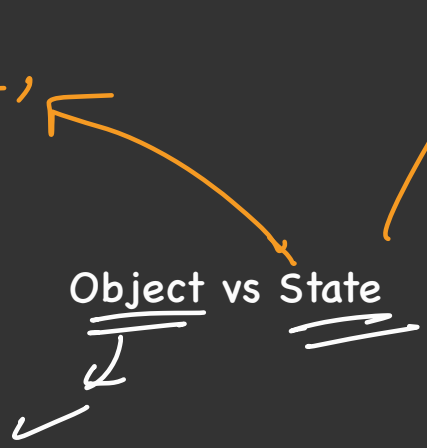
Employee.

Bike.

↳ mV: n, model, cc

↳ mF: start > br, <,

↳ State: onroad,
offroad,
repairs



Student

↳ mV: id, name

↳ mF: attendSession(), respectOrgVal(),
completeAssig(), code()

↳ States: ill, healthy, medical leave,

Problem Statement:

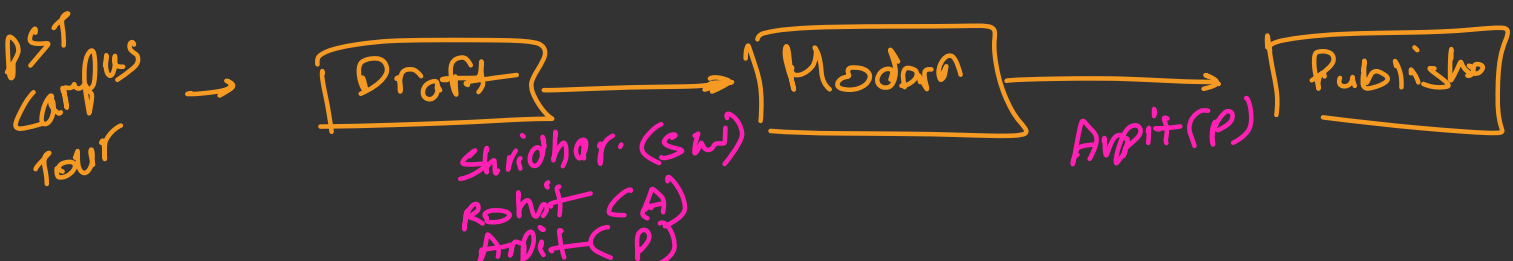
You are building a collaborative movie scripting tool where multiple users interact with a movie script. A script goes through multiple states during its lifecycle:

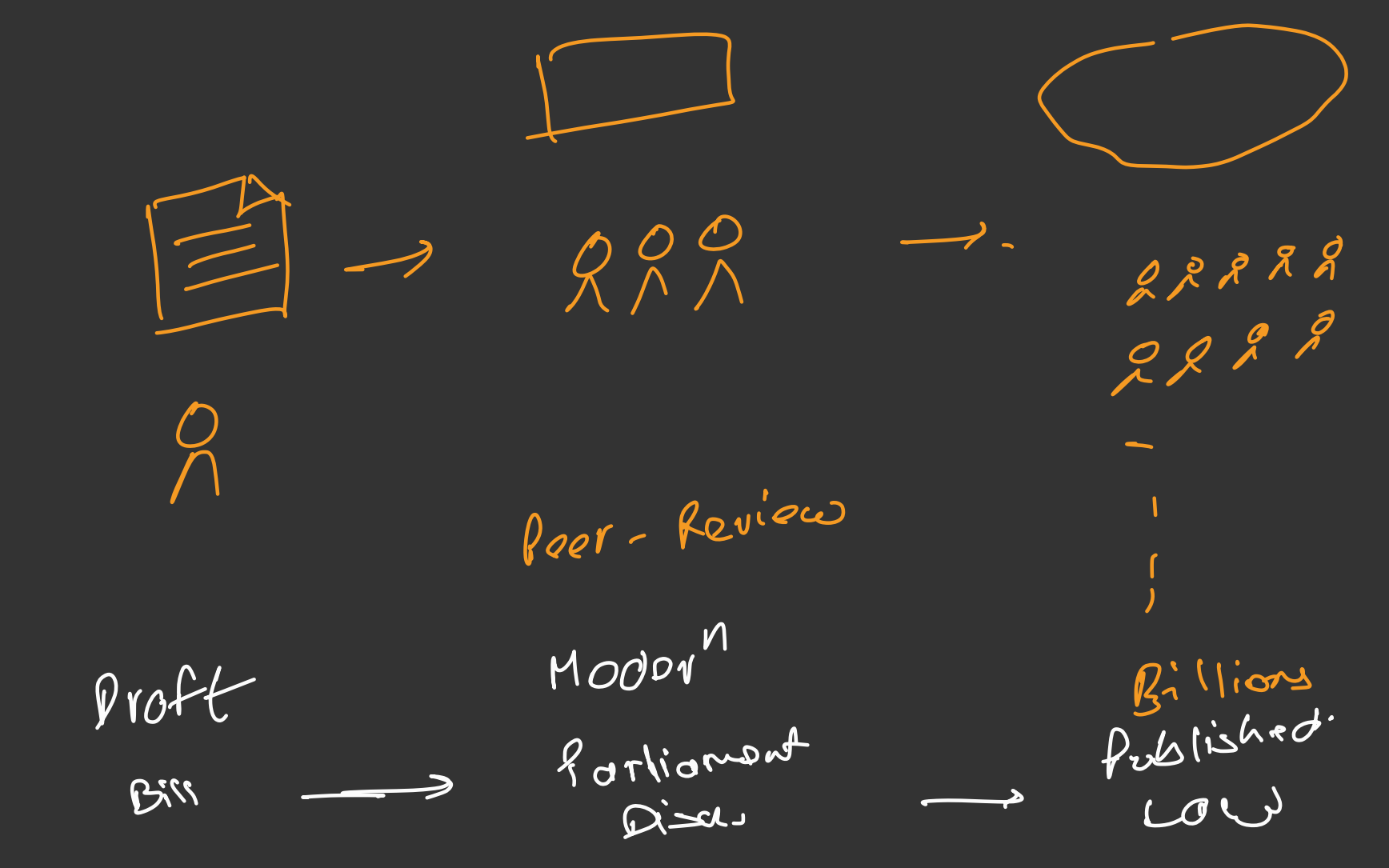
Draft

Moderation

Published

Different roles have different powers in this workflow: scriptwriter: Can create and send scripts to Moderation. actor: Can send scripts to Moderation, but cannot publish. producer: Can publish a script that's in Moderation.




Peer - Review
Draft
Bill
Modern
Parliament
Billions
Published
Law

what-ifs

Problem

State! you wrote logic using conditional checks based on user roles and documen

It led to tightly coupled logic.

Every new role or state meant modifying existing code. [OCP violated]

You needed a cleaner and extensible approach.

Goal

Use the State Design Pattern toi Represent each script state as an object (Draftstate, ModerationState, PublishState)

Delegate behavior changes (publish()) to the current state

Allow seamless transitions between states without hardcoded conditionals.

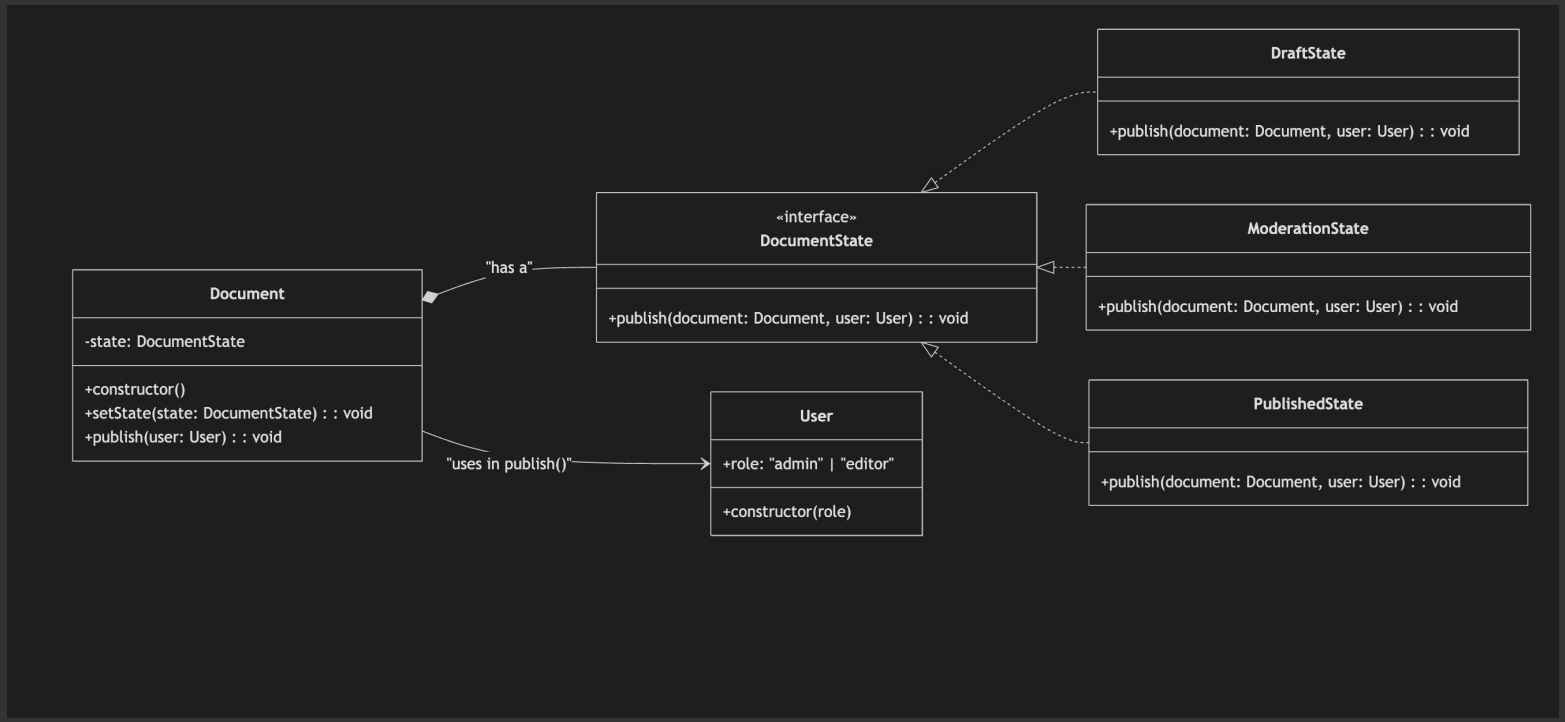

<< Docu >>


< DState.>


Draft

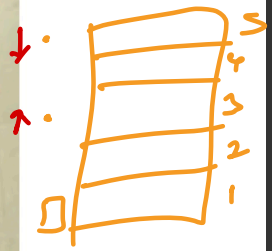

Modern

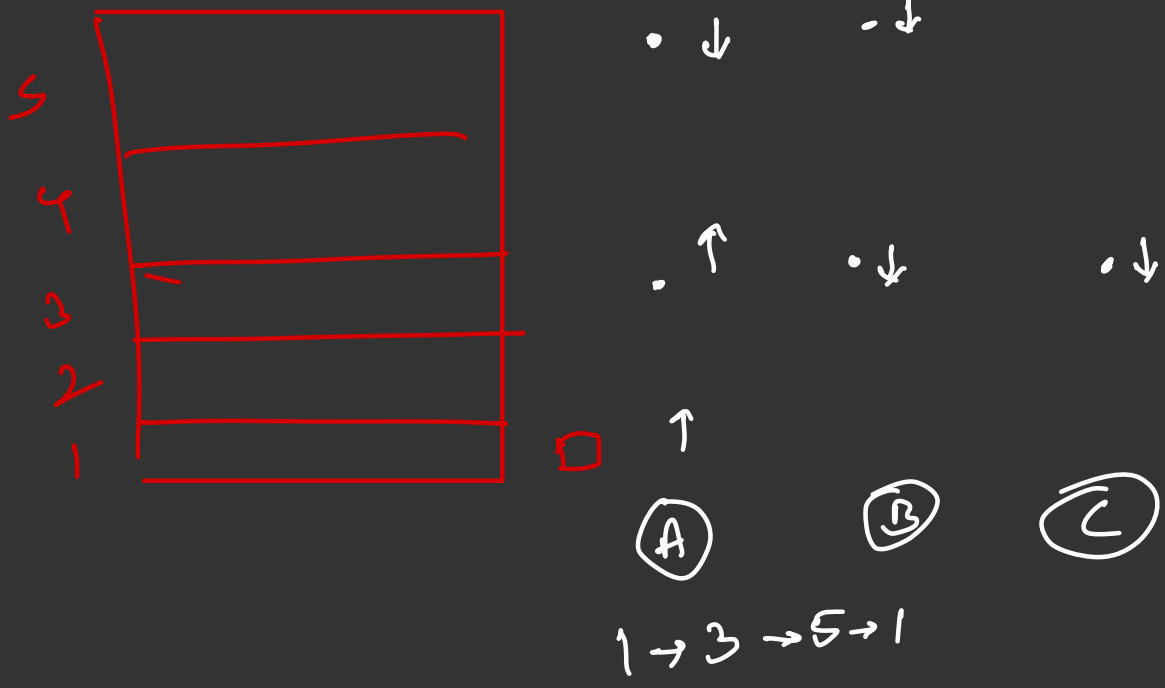

Publis


DraftState
+publish(document: Document, user: User) : : void
«interface»
DocumentState
+publish(document: Document, user: User) : : void
ModerationState
+publish(document: Document, user: User) : : void
Document
-state: DocumentState
+constructor()
+setState(state: DocumentState) : : void
+publish(user: User) : : void
"has a"
"uses in publish()"
User
+role: "admin" | "editor"
+constructor(role)
PublishedState
+publish(document: Document, user: User) : : void

Elevator System Design

Elevator System Design




5
4
3
2
1
A
B
C
1 → 3 → 5 → 1

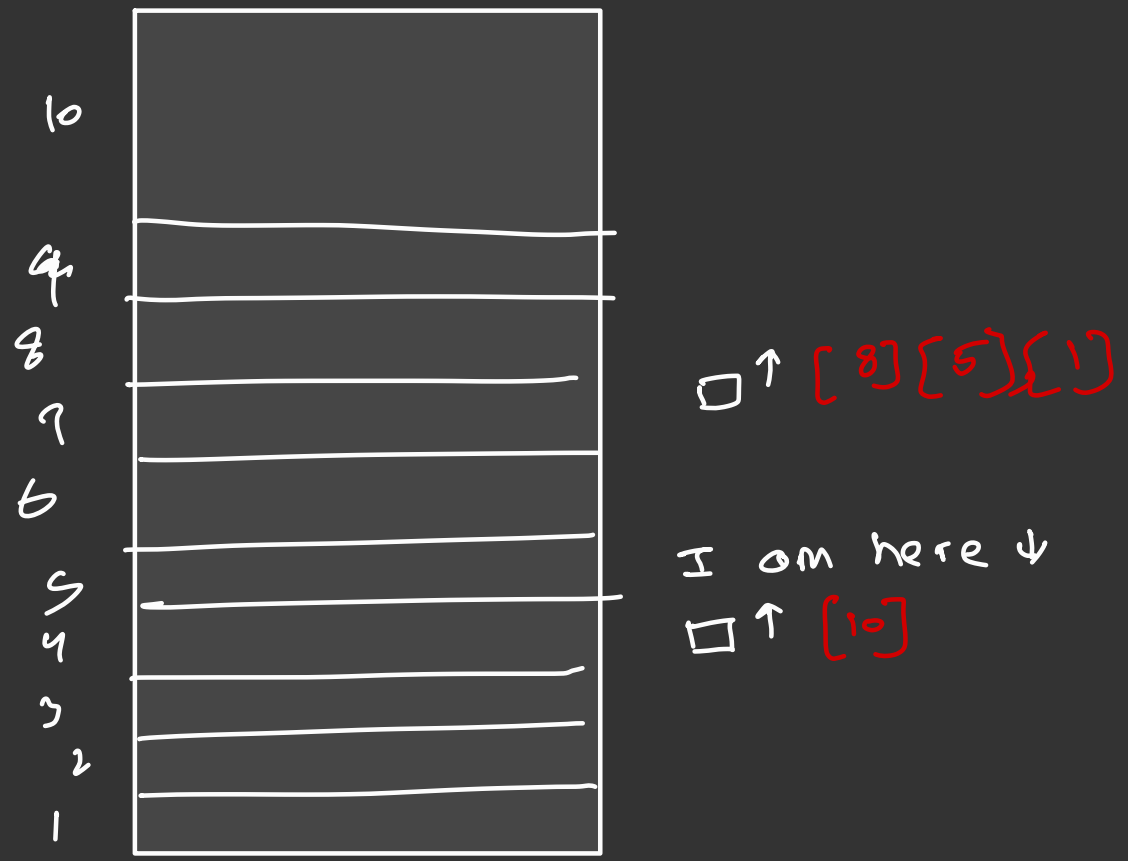
Requi

→ Floors

→ Elevators

→ Capacity-

→ Algo


10
9
8
7
6
5
4
3
2
1
[8][5][1]
I am here
[10]

Elevator System Design

What happens when you press the up button on an elevator from the 4th floor, while it's already moving down from the 10th floor?

Elevator System Design

"Would you consider all these variables during the elevator system design process?"

Popular Elevator Companies :
- Otis
- Schindler
- Mitsubishi

Gen2® Prime

Budget-friendly lift for residential buildings.

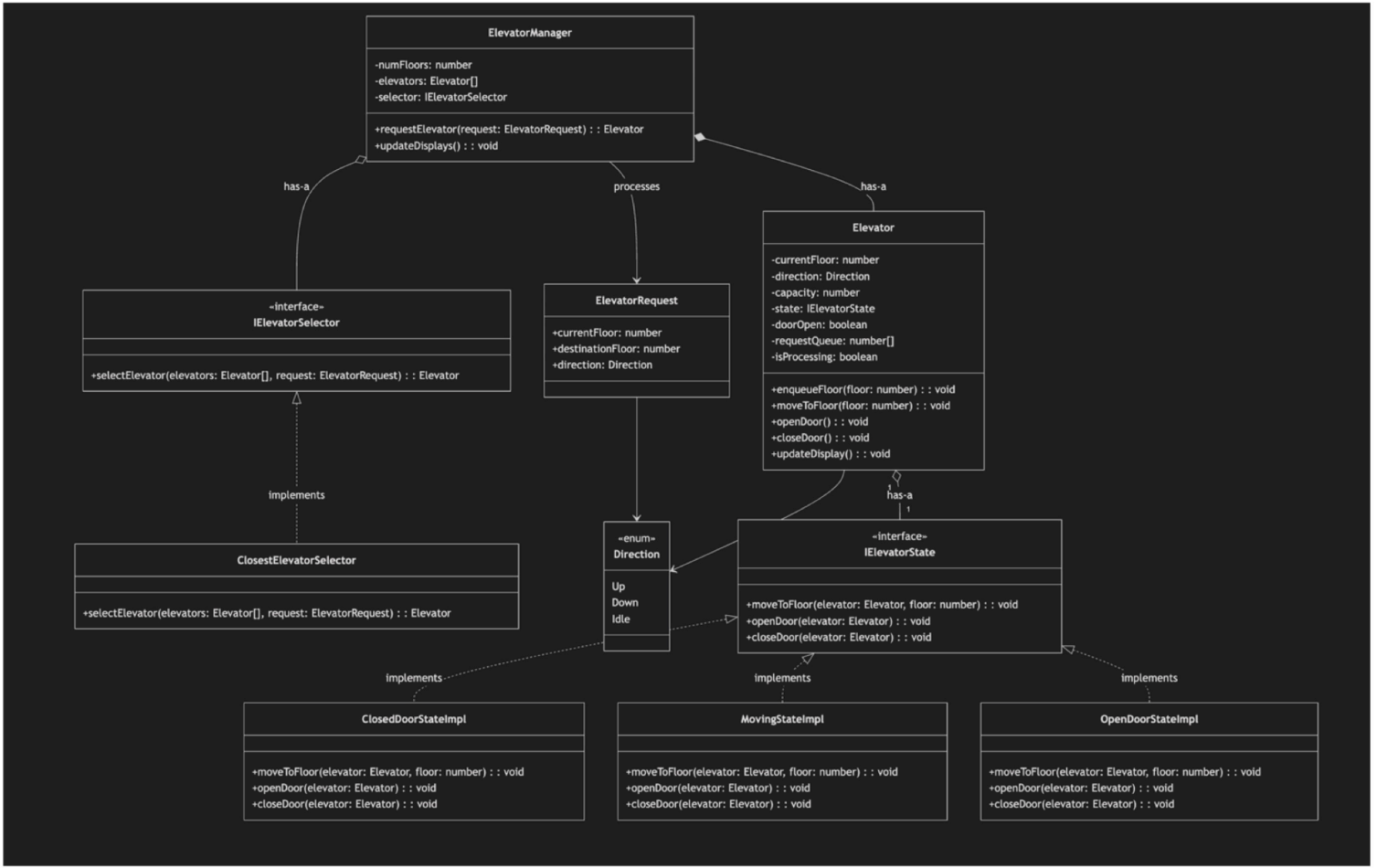
BOOK NOW

Gen3™ Nova

Designed for low- to mid-rise residential and commercial buildings.

BOOK NOW

	Gen2® Prime	Gen3™ Nova
Price	From ₹818,021	From ₹979,037
Smooth Landings	✔	✔
Energy savings up to 75%	✔	✔
Sleep mode	✔	✔
Zero lubrication	✔	✔
Smart IoT technology		✔
Big data analytics		✔
Advanced monitoring		✔
Seamless elevator calling		✔
Max Rise	20m	60m
Passenger capacity	up to 408 kg (5-6 p)	up to 1020 kg (>13 p)
Top speed	up to 0.7 m/s	1.0 m/s
Max stops	6	21
Available aesthetics	8	20
Hall fixtures	TBC	Flat and flush
Handrail finishes	Flat only	Stainless steel
Handrail arrangement	Rear or none	Left or Right Hand / Rear
Car operating panels	1	3 variations
Entrance door openings	2	3


ElevatorManager
-numFloors: number
-elevators: Elevator[]
-selector: IElevatorSelector
+requestElevator(request: ElevatorRequest) : : Elevator
+updateDisplays() : : void
has-a
processes
has-a
«interface»
IElevatorSelector
+selectElevator(elevators: Elevator[], request: ElevatorRequest) : : Elevator
ElevatorRequest
+currentFloor: number
+destinationFloor: number
+direction: Direction
Elevator
-currentFloor: number
-direction: Direction
-capacity: number
-state: IElevatorState
-doorOpen: boolean
-requestQueue: number[]
-isProcessing: boolean
+enqueueFloor(floor: number) : : void
+moveToFloor(floor: number) : : void
+openDoor() : : void
+closeDoor() : : void
+updateDisplay() : : void
implements
has-a
1
1
«enum»
Direction
Up
Down
Idle
«interface»
IElevatorState
+moveToFloor(elevator: Elevator, floor: number) : : void
+openDoor(elevator: Elevator) : : void
+closeDoor(elevator: Elevator) : : void
ClosestElevatorSelector
+selectElevator(elevators: Elevator[], request: ElevatorRequest) : : Elevator
implements
implements
implements
ClosedDoorStateImpl
+moveToFloor(elevator: Elevator, floor: number) : : void
+openDoor(elevator: Elevator) : : void
+closeDoor(elevator: Elevator) : : void
MovingStateImpl
+moveToFloor(elevator: Elevator, floor: number) : : void
+openDoor(elevator: Elevator) : : void
+closeDoor(elevator: Elevator) : : void
OpenDoorStateImpl
+moveToFloor(elevator: Elevator, floor: number) : : void
+openDoor(elevator: Elevator) : : void
+closeDoor(elevator: Elevator) : : void