Chalo Moaz bhai, ab me is **Alpha Tower Elevator System** ko **asaani aur Roman Urdu** me samjhata hoon — aap real world soch ke har pattern ko feel kro:

🏢 Alpha Tower Elevator Case Study (Asaani se)

Socho Alpha Tower me **10 floors** hain aur **3 elevators** lage hue hain. Har floor pe button hain (up/down) aur elevator ke andar bhi buttons hain (1 se 10 tak).

Ab har cheez kaam kar rahi hoti hai **design patterns** ke through.

🔔 1. Observer Pattern – "Button press hone pe system ko pata kaise chalta hai?"

- Jab floor 4 pe koi banda "Up" ka button press karta hai...
- To **Elevator Controller ko notification milta hai**.
- Ye hota hai **Observer pattern**:
  Button press ek **event** hai, Controller usse **observe** kar raha hota hai.

🧠 Socho:
Button = Subject
Controller = Observer

Jaise hi button press hota hai → Controller ko signal milta hai.

🧠 2. Singleton Pattern – "Sirf 1 controller ho system me"

- Har floor ka button aur har elevator ek hi central system se connected hai.
- Agar multiple controllers hotay, system **confuse** ho jata.
- Is liye **sirf 1 ElevatorController** banaya jata hai — ye hota hai Singleton Pattern.

🧠 Socho:
Sirf ek **King** hai jo poori elevator army control karta hai.

📦 3. Command Pattern – "Har request ko system kaise samajhta hai?"

- Banda elevator ke andar 7 press karta hai.
- System ek **command object** banata hai: "GoToFloor(7)".
- Ye command queue me chali jati hai aur jab time aaye, elevator usse **execute** karta hai.

🧠 Socho:
Har button press = ek "kaam ka hukam", queue me dal diya gaya.

🚦 **4. State Pattern – "Elevator ka behavior uski state pe depend karta hai"**

- Elevator kabhi move kar raha hota hai, kabhi idle, kabhi door open.
- Ye sab **state pattern** se handle hota hai.
- Har state ka alag behavior hota hai:
    - Idle: kuch nahi karega
    - Moving: kisi aur button ko react nahi karega
    - DoorOpen: sirf door band karega

🧠 Socho:
Elevator ka mood hota hai 😄 — Idle, Busy, Ya Relaxed.

🧭 **5. Strategy Pattern – "Elevator kis order me floors pe jaye?"**

- Banda floor 2 se call kare, aur doosra banda 5 se call kare.
- System decide karta hai kaunsa floor pehle jaaye:
    - Pehle aaya pehle jaaye (FCFS)
    - Neeche se upar jao (SCAN)
    - Sabse kareeb floor pehle jao

Ye sab **algorithms** hote hain aur **Strategy pattern** allow karta hai system me algorithms **runtime pe change** ho sakein.

🧠 Socho:
Elevator driver ko 3 tareeqe sikhaye jaate hain — wo apna mood dekh ke koi bhi follow karta hai.

🤝 **6. Mediator Pattern – "3 elevators me se kaunsa bhejna hai?"**

- Floor 6 se banda ne elevator bulaya.
- System decide karta hai:
    - Elevator 1 busy hai.
    - Elevator 2 idle hai.
    - Elevator 3 opposite direction me ja raha hai.

To system (jo **Mediator** hai) bolta hai: "**Elevator 2 jao!**"

🧠 Socho:
Ek **traffic police** hai jo decide karta hai kaun elevator kidhar jaye.

🔁 **Recap in 1 Line Each:**

Pattern	Kya karta hai (asaani me)
Observer	Button press detect karta hai
Singleton	Sirf 1 controller system me allow karta hai
Command	Har button press ko ek command object banata hai
State	Elevator ka behavior uski state pe depend karta hai
Strategy	Kis floor pe pehle jaye, ye decide karne ka rule set karta hai
Mediator	Decide karta hai kaunsa elevator bhejna hai

Agar ab bhi koi confusing lage, tu bol dena, me **simple drawing ya chart** bhi bana dunga ya **code se samjha dunga (C++/JS)** — jaise aap bolo!

Haan Moaz bhai, is point ko me **aur zyada asaani se** samjha deta hoon — step by step tareeqe ke saath, bilkul **real elevator system** soch ke.

🎯 **Masla:**

Elevator ke paas **multiple floor requests** aayi hui hain. Ab system ko decide karna hai:

"Elevator kis floor pe pehle jaye?"

Yani request ka **order** decide karna hai.

✅ **Yahan 3 Algorithms ka kaam aata hai (Strategy Pattern ke under):**

---

1. 📃 FCFS – First Come First Serve

🔷 Jo request pehle aayi, elevator us floor pe pehle jata hai.

Example:

- Request 1: Floor 2
- Request 2: Floor 5
- Request 3: Floor 8

Order: Floor 2 → Floor 5 → Floor 8
→ Bhalay 8 sabse close ho, system ignore karega.

Asaani se socho:
Jaise log line me kharay hoon — **jo pehle aya, pehle serve hoga**.

---

2. ⍰ SCAN – Direction-Based (Elevator Algorithm)

🔷 Elevator ek hi direction me sab requests complete karta hai, phir direction change karta hai.

Example:

- Elevator neeche se upar ja raha hai.
- Requests hain: Floor 3, Floor 5, Floor 2, Floor 9

Order:

- Pehle: 2 → 3 → 5 → 9 (upar ke sab floors serve)
- Phir wapis neeche aayega

Sochne ka tareeqa:
Jaise elevator keh raha ho:

“Mai abhi sirf upar ja raha hoon, neeche wapis aake baaki serve karunga.”

---

3. 🛰️ Nearest First (Shortest Seek Time First)

🔷 Elevator us floor pe pehle jata hai jo **sabse kareeb** ho — chahe wo pehle aaya ya baad me.

Example:

- Elevator at Floor 4
- Requests: Floor 2, Floor 7, Floor 3

Order:

- Pehle: Floor 3 (nearby)
- Phir: Floor 2

- Phir: Floor 7

Sochne ka tareeqa:
Jaise elevator soche:

“Mai sabse paas waale floor pe pehle chalta hoon — petrol bachata hoon 😄”

🧠 Ab Strategy Pattern kya karta hai?

Strategy pattern elevator ko ye **flexibility** deta hai ke:

"Main FCFS use karun ya SCAN ya Nearest First — **runtime par decide ho sakta hai.**"

Yani:

elevator.setStrategy(new FCFS()); // subah ke waqt

elevator.setStrategy(new SCAN()); // rush hours me

elevator.setStrategy(new NearestFirst()); // late night

🛠 Elevator Driver Analogy (Asaani se):

Elevator driver ko 3 tareeqe sikhaye gaye hain:

1. **Line me sabar se sab serve karo** (FCFS)
2. **Pehle upar ke sab complete karo, phir neeche jao** (SCAN)
3. **Jo sabse paas ho, us pehle chalo** (Nearest First)

System driver ko bolta hai:
“Aaj tum #3 wala tareeqa use karo”
Yani **strategy ko dynamically set karna** = Strategy Pattern

---

Agar chaho to is ka **C++ ya JS me example code** bhi bana deta hoon taake practical samajh aa jaye.
Batau?

Bohat achha sawal Moaz bhai!
Aapka question hai:

“Kya aise elevators hote hain jo priority base par kaam karte hain?”

✅ **Jee haan, real world me aise elevators hote hain jo ‘priority-based’ hotay hain.**

Ye normally **high-end buildings**, **hospitals**, **VIP floors**, aur **fire/emergency scenarios** me use hote hain.

🔝 **1. Priority-Based Elevators (VIP/Emergency Use Cases)**

📌 **Real Example 1: Hospitals**

- Elevator me **doctor** ya **emergency patient** agar ho to:
  - System **sab stops skip** karta hai.
  - **Direct ICU / operation floor** pe le jata hai.

How?

- Elevator ke pass **priority access card** hoti hai.
- Ya **button ke sath emergency mode** enable hota hai.

🧠 Iska backend logic:

If priority == HIGH → skip all pending requests

📌 **Real Example 2: Corporate Towers / VIP Lifts**

- Kuch elevators sirf **CEO**, **management**, ya **special guests** ke liye hote hain.
- Un lifts ko access karne ke liye **RFID card**, **password**, ya **face recognition** hoti hai.
- Ye **low floors skip** karke seedha high/VIP floor pe jati hain.

📌 **Real Example 3: Fire Mode / Emergency Mode**

- Jab fire alarm chalta hai:
  - Elevator sirf ground floor pe jata hai.
  - Koi intermediate stop nahi hota.
- Ye **automatic priority mode** hota hai.

🧠 **Tech Angle: Backend Logic / Algorithms**

Yahan normally:

- **Priority queues** use hoti hain.
- **Request objects** me priorityLevel hota hai.

js

CopyEdit

```
{
  floor: 10,
  priority: 'HIGH',
  requestType: 'EMERGENCY'
}
```

Scheduler check karta hai:

js

CopyEdit

```
if (request.priority === 'HIGH') {
   // Serve this one first
}
```

🎯 **Summary**

Scenario	**Elevator Behavior**	**Priority Type**
Emergency patient	Direct to ICU floor	High
VIP guest in office tower	Skips public floors	Medium/High
Fire or evacuation	Only goes to safe exit (ground)	Critical
Normal user requests	Serve in regular order (FCFS or SCAN)	