**Smart Home Automation:**

**1. Introduction**

Behavioral design patterns focus on how objects interact and communicate with each other. They define **responsibility, communication, and algorithms** in a system. In this assignment, I have implemented a **Smart Home Automation System** in Python that demonstrates **eight behavioral design patterns**. The system simulates devices like lights and fans, user notifications, scheduling, alerts, and a central smart hub.

**2. System Overview**

The Smart Home Automation System automates tasks such as:

* Switching devices ON/OFF
* Undoing/Redoing device commands
* Managing device states (e.g., fan speed)
* Scheduling devices based on strategies
* Saving/restoring device configurations
* Parsing user commands in natural language
* Handling alerts with escalation
* Centralized communication between devices

This system is designed to be **modular, extensible, and easy to understand**, thanks to the use of **behavioral design patterns**.

**3. Implemented Behavioral Patterns**

**Observer Pattern**

* **Purpose:** One-to-many notification mechanism.
* **Usage:** When a device’s state changes (e.g., Light ON/OFF), all registered user apps are notified automatically.
* **Example in system:** Device notifies UserApp.

**Command Pattern**

* **Purpose:** Encapsulate requests as objects to support undo/redo.
* **Usage:** Turn ON/OFF commands for devices can be undone or redone.
* **Example in system:** TurnOnCommand, TurnOffCommand, and CommandHistory.

**State Pattern**

* **Purpose:** Change behavior based on internal state.
* **Usage:** Fan cycles through states (OFF → LOW → MEDIUM → HIGH → OFF).
* **Example in system:** OffState, LowState, MediumState, HighState.

**Strategy Pattern**

* **Purpose:** Define a family of algorithms and make them interchangeable.
* **Usage:** Device scheduling can follow different strategies (Timed, Sensor-based).
* **Example in system:** TimedStrategy, SensorStrategy.

**Mediator Pattern**

* **Purpose:** Reduce direct dependencies between objects by using a mediator.
* **Usage:** Smart Home Hub sends commands to devices, avoiding direct device-to-device communication.
* **Example in system:** SmartHomeHub.

**Memento Pattern**

* **Purpose:** Capture and restore an object’s state.
* **Usage:** Save and restore device configurations (e.g., light state).
* **Example in system:** DeviceMemento, DeviceCaretaker.

**Interpreter Pattern**

* **Purpose:** Interpret and execute language/grammar-based expressions.
* **Usage:** Parse user commands like “turn on light” or “toggle light.”
* **Example in system:** CommandInterpreter.

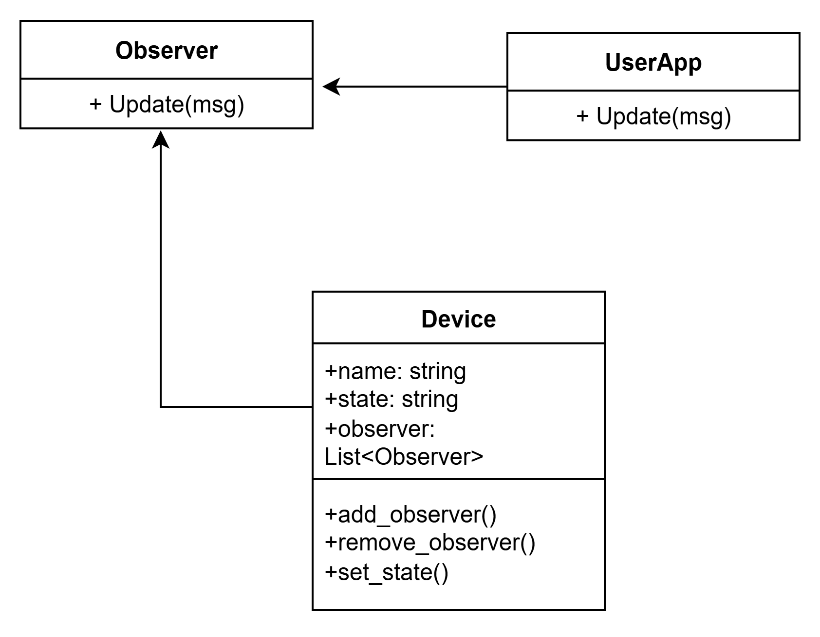
**Chain of Responsibility Pattern**

* **Purpose:** Pass requests along a chain until handled.
* **Usage:** Alerts escalate (Motion → Alarm → Police).
* **Example in system:** MotionHandler, AlarmHandler, PoliceHandler.

**4. UML Diagrams**

**4.1 Class Diagram**

**Observer Pattern:**

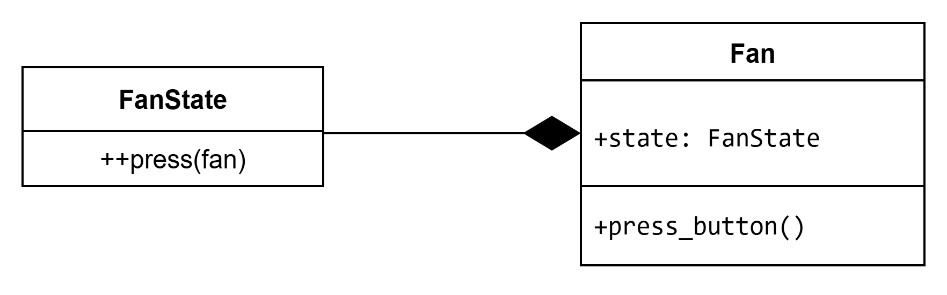


**Command Pattern:**

A black screen with white text

AI-generated content may be incorrect.

**State Pattern:**



**Strategy pattern:**

A black background with a black square

AI-generated content may be incorrect.

**Mediator Pattern:**



**Memento Pattern:**

A black background with a black square

AI-generated content may be incorrect.

**Interpreter Pattern:**

A black background with white text

AI-generated content may be incorrect.

**Chain of responsibility Pattern:**

A black and white rectangular object with text

AI-generated content may be incorrect.

**4.2 Sequence Diagrams**

**Observer Pattern (Light state change)**

A white rectangular object with a black background

AI-generated content may be incorrect.

**Command Pattern (Undo/Redo for Light)**

A black background with white rectangles

AI-generated content may be incorrect.

**State Pattern (Fan button press):**

A white rectangular object with a black background

AI-generated content may be incorrect.

**5. Backend Logic:**

This section presents the Python implementation of the Smart Home Automation System. The program is **console-based with hardcoded values** to focus on demonstrating behavioral design patterns.

**Observer pattern:**

A screen shot of a computer program

AI-generated content may be incorrect.

Output:

A white background with black text

AI-generated content may be incorrect.

**Command Pattern:**

A screen shot of a computer program

AI-generated content may be incorrect.

Output:

A black text on a white background

AI-generated content may be incorrect.

**State patten:**

A screen shot of a computer program

AI-generated content may be incorrect.

Output:

A screenshot of a computer program

AI-generated content may be incorrect.

**Strategy pattern:**

A computer screen shot of a program code

AI-generated content may be incorrect.

Output:

A black text on a white background

AI-generated content may be incorrect.

**Mediator pattern:**

A screen shot of a computer program

AI-generated content may be incorrect.

Output:

A black text on a white background

AI-generated content may be incorrect.

**Memento pattern:**

A screen shot of a computer program

AI-generated content may be incorrect.

Output:

A black text on a white background

AI-generated content may be incorrect.

**Interpreter pattern:**

A computer screen shot of a program code

AI-generated content may be incorrect.

Output:

A screenshot of a computer

AI-generated content may be incorrect.

**Chain of responsiblity pattern:**

A screen shot of a computer program

AI-generated content may be incorrect.

Output:

A screen shot of a computer

AI-generated content may be incorrect.

**6. Conclusion:**

The Smart Home Automation System demonstrates how behavioral design patterns can make software more modular, reusable, and extensible. Each pattern solves a specific problem related to object behavior and communication.

This single integrated system successfully applies eight behavioral patterns in a real-world context, showing their practical use in software design.