Name :Sinam Ametewee

ID: 97202026

Date: 20th April, 2025

Course: Software Engineering

Smart Home Controller

Overview

The Smart Home Controller is a Java-based application that simulates the control and automation of various smart devices such as lights, alarms, locks, and cameras within a household. To ensure modularity, reusability, and maintainability, this project integrates 3 core software design patterns which are the Observer, Factory, Strategy design patterns

Design Patterns Used and Why

Observer Pattern

The Observer Pattern was chosen to make it easy to differentiate between the states of a device and allow users to monitor the current mode or status of their smart devices. For example, users can see whether the lighting in a room is turned ON or OFF. Whenever a device's state changes, all registered log components are notified immediately, simulating a real-time update on the system's interface or logs. Implemented in: UI, Log, and Component interface

Factory Pattern

The Factory Pattern enables the dynamic creation of various smart devices such as lights, alarms, locks, and cameras. This made it easier to scale the system and allowed users to access a broad range of device types from the same interface without hardcoding each one individually. Implemented in: DeviceChoice, Lighting, Cameras, Alarms, Locks

Strategy Pattern

The Strategy Pattern was implemented to support different automation modes, where various sets of behaviors could be applied to groups of devices. For example: Night Mode turns off all devices. Vacation Mode ensures that security devices are ON while others are OFF. Manual Mode turns on all devices manually. This pattern separates automation logic from the core system, improving flexibility and making it easier to add or modify automation behaviors in the future. Implemented in: ControlStrategy, NightMode, VacationMode, ManualControl, AutomationController.

Key Classes and Their Roles

UI – Maintains and updates the log of device state changes via the Observer pattern.

DeviceChoice – Dynamically creates device instances based on user requests.

AutomationController – Applies different control strategies (modes) to a list of devices.

SmartHomeController – Singleton class responsible for centralized device control.

Challenges Faced

A notable challenge was understanding how to properly integrate the various design pattern concepts during the early stages of development. In particular, distinguishing between the Observer and Strategy patterns required extra effort. The Observer pattern is centered around notifying dependent components about state changes, whereas the Strategy pattern is about selecting and executing different behaviors. Overcoming these challenges helped reinforce my understanding of object-oriented design and strengthened my implementation skills.