Smart Home Automation System

Introduction

The Smart Home Automation System is an innovative IoT project that provides homeowners with comprehensive control and monitoring capabilities for their home appliances and security systems. Through a user-friendly interface, users can remotely manage various aspects of their homes, including lighting, thermostat settings, and security surveillance. This system leverages object-oriented programming principles in Java to enhance convenience, energy efficiency, and home security.
Our implementation focuses on creating a flexible, extensible system that allows users to control devices remotely, schedule automation tasks, and monitor energy consumption while ensuring secure access through proper authentication and authorization mechanisms.

System Architecture

The Smart Home Automation System is built using a modular architecture with the following key components:

Core Components

1. Device Management
    - Abstract base class for all devices with common functionality
    - Specialized device implementations (lights, thermostats, security devices)
    - Energy consumption monitoring for compatible devices
2. User Management
    - Role-based access control (Admin vs. Regular users)
    - Authentication system
    - Permission-based actions
3. Automation Rules Engine
    - Time-based automation (e.g., turn on lights at specific times)
    - Event-based automation (e.g., motion detection triggers lights)
    - Scheduled automation with day-of-week support
4. Security Layer
    - User authentication
    - Access token management
    - Encryption services

5. User Interface
    - Command-line interface for direct system control
    - Graphical user interface using Java Swing

Package Structure

The system is organized into the following packages:

- `com.smarthome` - Main system components
- `com.smarthome.devices` - Device-related classes
- `com.smarthome.users` - User management
- `com.smarthome.automation` - Automation rules
- `com.smarthome.security` - Security features
- `com.smarthome.ui` - User interfaces
- `com.smarthome.utils` - Utility classes

Key Features

Device Control and Monitoring

- Remote control of devices (on/off, settings adjustment)
- Real-time status monitoring
- Energy consumption tracking
- Support for different device types:
    - Lights with brightness control
    - Thermostats with temperature settings
    - Security devices with arming capabilities

Automation Capabilities

- Scheduled operations based on time
- Event-triggered actions
- Condition-based automation
- Multiple devices can be controlled by a single automation rule

User Management

- Different access levels (Admin and Regular users)
- Secure authentication

- User-specific permissions and preferences

Security Features

- Password protection
- Token-based authentication
- Secure communication
- Access control for sensitive operations

Persistence

- Configuration saving and loading
- Device state persistence
- User settings storage

Implementation Details

OOP Concepts Utilized

1. Inheritance
    - Hierarchical inheritance for devices and users
    - Base classes with common functionality
2. Polymorphism
    - Device-specific behaviors through method overriding
    - Runtime type determination for appropriate actions
3. Encapsulation
    - Private fields with public getters/setters
    - Information hiding for security
4. Abstraction
    - Abstract classes for devices and users
    - Interfaces for common behaviors
5. Interfaces
    - `EnergyMonitored` for energy-tracking devices
    - `AutomationRule` for different types of automation

Design Patterns

- Observer Pattern - For device status monitoring
- Command Pattern - For automation rule execution

- Factory Pattern - For creating different types of devices
- Singleton Pattern - For system-wide components

Smart Home Automation System: Detailed Class Explanation

Based on our Smart Home Automation System implementation, here's a comprehensive explanation of all the classes used in the project:

Core System Classes

SmartHomeSystem

The central class that manages the entire system. It coordinates all components and provides the main functionality.

- Maintains lists of devices, users, and automation rules
- Handles user authentication
- Manages device operations
- Processes automation rules
- Implements configuration persistence through serialization
- Contains a nested `DeviceMonitor` class for multithreading
- Provides both CLI and programmatic interfaces

DeviceMonitor (Inner Class)

A nested class within SmartHomeSystem that implements Runnable for multithreading.

- Continuously monitors device statuses
- Triggers automation rules when conditions are met
- Tracks energy consumption for compatible devices

Device Package Classes

Device (Abstract Class)

The base class for all smart devices in the system.

- Provides common attributes like ID, name, and status
- Implements basic on/off functionality
- Defines abstract methods for device-specific behavior
- Implements Serializable for persistence

EnergyMonitored (Interface)

Interface for devices that can track energy consumption.

- Defines methods for energy consumption tracking
- Allows for consistent energy monitoring across different device types

LightDevice

Implementation for smart lights.

- Extends Device and implements EnergyMonitored
- Adds brightness control functionality
- Tracks energy usage based on brightness and usage time
- Provides overloaded constructors for different initialization options

ThermostatDevice

Implementation for smart thermostats.

- Extends Device and implements EnergyMonitored
- Manages temperature settings
- Tracks energy usage based on temperature differential
- Provides methods for temperature adjustment

SecurityDevice

Implementation for security-related devices.

- Extends Device
- Contains an inner enum for device types (CAMERA, MOTION_SENSOR, ALARM)
- Implements arming/disarming functionality
- Provides alert triggering capabilities

User Package Classes

User (Abstract Class)

Base class for all system users.

- Manages authentication credentials
- Provides basic user information
- Defines abstract permission checking
- Implements Serializable for persistence

AdminUser

Implementation for administrative users.

- Extends User
- Has full system permissions
- Can add/remove devices and users
- Can create automation rules

RegularUser

Implementation for standard users.

- Extends User
- Has limited permissions (device control, status viewing)
- Cannot modify system configuration

Automation Package Classes

AutomationRule (Interface)

Interface for all automation rules in the system.

- Defines methods for rule triggering and execution
- Implements Serializable for persistence

TimeBasedRule

Implementation for time-triggered automation.

- Implements AutomationRule
- Executes actions at specific times

- Uses Java's time API for scheduling

EventBasedRule

Implementation for event-triggered automation.

- Implements AutomationRule
- Executes actions when specific events occur
- Provides event detection logic

ScheduledRule

Implementation for day-specific scheduled automation.

- Implements AutomationRule
- Extends time-based functionality with day-of-week support
- Allows for more complex scheduling patterns

Security Package Classes

PermissionManager

Manages permissions for different users or devices in the smart home system.

- Access Token Management
- Secure Mode Control
- Encryption Operations (Static Nested Class: Encryptor)

SecurityLogger

Provides a utility for logging system events, errors, and informational messages.

- Log Messages to Console and File
- Log Levels:
  Supports different logging levels: `DEBUG`, `INFO`, `WARNING`, and `ERROR`

- Exception Logging:
  Automatically logs detailed information about exceptions, including stack traces, for easier debugging.
- Configurable Logging:
  Allows setting a minimum log level to filter out unnecessary log messages.

SecurityManager

Manages security operations within the smart home system.

- User Access Tokens:
  Generates unique access tokens for users to securely identify and authenticate them.
- Token Validation
- Token Revocation
- Secure Mode Management
- Basic Encryption and Decryption

UI Package Classes

SmartHomeGUI

Graphical user interface for the system.

- The app is built with **Java Swing** (`JFrame`, `JPanel`, `JButton`, etc.).

Constructor: `SmartHomeGUI(SmartHomeSystem system)`

- Initializes the GUI frame (window).
- Adds two main screens into `mainPanel`:
  - **Login Panel** (`loginPanel`)
  - **Device Control Panel** (`devicePanel`)
- Shows the **Login screen** first.
- Adds a **window listener** that stops (`system.stop()`) the backend when the app closes.

Login Panel: `createLoginPanel()`

- Simple login form:
  - **Username field**
  - **Password field**
  - **Login Button**

- When login button clicked:
  - It **authenticates** user through `system.authenticateUser(username, password)`.
  - If successful:
    - Saves the logged-in user to `currentUser`.
    - Updates and switches to **device control screen**.
  - If failed:
    - Shows an **error message**.

Device Panel: `updateDevicePanel()`

- After login, shows:
  - A **Welcome message** with user's name.
  - A **Logout button**.
  - A **scrollable list of devices** (lights, thermostat, security devices).
- Creates a list of **demo devices** inside `updateDevicePanel()` itself (this could later be dynamic from system backend).
- For each device, calls `createDeviceControlPanel(device)` to create its control UI.

Device Control Panels: `createDeviceControlPanel(Device device)`

Each device gets:

- A **Status Label** ("ON" or "OFF").
- **Turn ON** and **Turn OFF** buttons.

  Then **additional controls** depending on device type:

Device Type	Additional Controls
`LightDevice`	- Brightness slider (0-100%)
`ThermostatDevice`	- Temperature control with + and - buttons
`SecurityDevice`	- Arm and Disarm buttons

All device controls **update** the device's state immediately when buttons or sliders are used.

Main Method: `main(String[] args)`

- Creates a `SmartHomeSystem` object.
- Starts the backend system.
- Launches the GUI (`SmartHomeGUI`) on the Swing Event Dispatch Thread (`SwingUtilities.invokeLater`).

Utils Package Classes

Logger

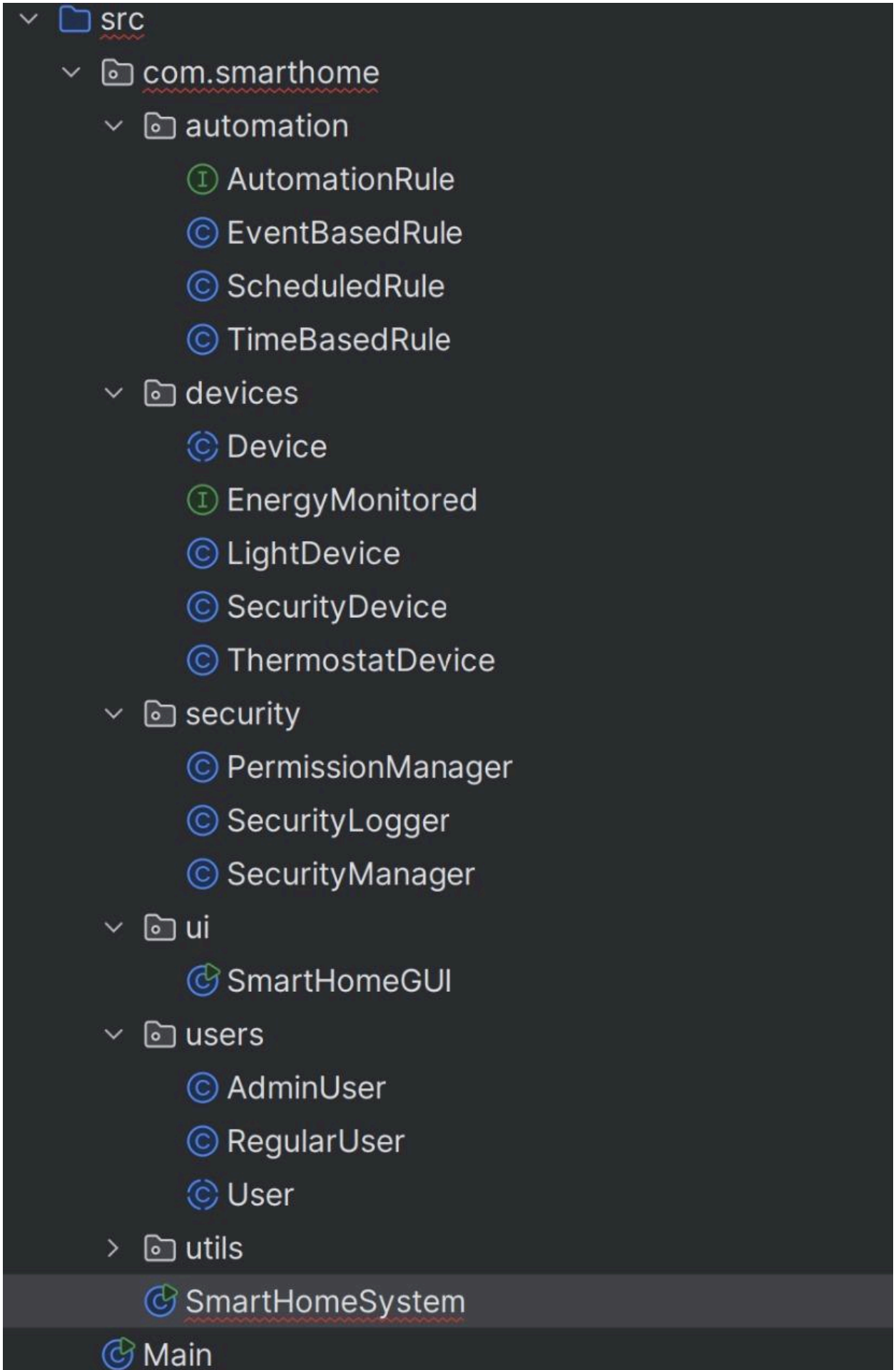
Utility class for system logging.

- Implements different log levels (DEBUG, INFO, WARNING, ERROR)
- Writes logs to both console and file
- Provides specialized methods for different log types
- Handles file I/O with proper exception management

Technical Requirements Implemented

- Overloaded Methods - Multiple methods with the same name but different parameters
- Overloaded Constructors - Different initialization options
- Vararg Overloading - Methods accepting variable numbers of arguments
- Nested Classes - Inner classes for related functionality
- Abstract Classes - Base classes with some implementation
- Interfaces - Contracts for implementation
- Hierarchical Inheritance - Class hierarchies
- Multiple Inheritance (through interfaces) - Classes implementing multiple interfaces
- Exception Handling - Proper error management
- **File I/O Operations

This comprehensive class structure creates a flexible, extensible system that fulfills all the requirements of a modern smart home automation solution while demonstrating advanced Java programming concepts.


src
com.smarthome
automation
AutomationRule
EventBasedRule
ScheduledRule
TimeBasedRule
devices
Device
EnergyMonitored
LightDevice
SecurityDevice
ThermostatDevice
security
PermissionManager
SecurityLogger
SecurityManager
ui
SmartHomeGUI
users
AdminUser
RegularUser
User
utils
SmartHomeSystem
Main

RUN COMMANDS

```
admin> help
Available commands:
  devices - List all devices
  control <deviceId> - Control a device
  rules - List all automation rules
  logout - Log out
  energylogs - View energy consumption logs
  exit - Exit the system
```