**Smart Home Automation System**

**Project Overview**

The **Smart Home Automation System** is a Java-based application that allows users to remotely control and monitor household devices such as lights, fans, air conditioners, and security systems. The system emphasizes **Object-Oriented Programming (OOP)** principles and Java-specific features to deliver a flexible, scalable, and maintainable framework for home automation.

Key features include device control, setting adjustments, scheduling of automation tasks, energy consumption tracking, and secure user authentication with role-based access control. The system supports two user roles: **Admin** and **Regular User**, each with distinct privileges.

**System Architecture**

The application is divided into six well-defined packages, each handling a specific responsibility:

1. **smarthome.auth** – User authentication and management
2. **smarthome.devices** – Implementations for controllable devices
3. **smarthome.automation** – Scheduling and automation logic
4. **smarthome.io** – File operations and data persistence
5. **smarthome.exceptions** – Custom exception handling
6. **smarthome.main** – Main application entry point and controller logic

**Core Components**

**1. User Management System**

* **User (Abstract Class)**: Contains basic properties (username, password) and methods like authenticate() and getUsername(). It includes an abstract method getRole() to enforce role-specific behavior.
* **Admin**: Inherits from User and has privileges to manage users and devices.
* **RegularUser**: Inherits from User and can control devices and schedule automations.
* **Security Class**: Contains utility methods, including a nested PasswordHandler class that ensures password strength (min. 8 characters, with at least one number).

**2. Device Control System**

* **ControllableDevice (Abstract Class)**: Base class with shared properties such as name, power consumption, and on/off state. Includes common methods like turnOn(), turnOff(), and getStatus().
* **Device Types**:
  + **Light**: Brightness control (1–100%)
  + **Fan**: Speed control (1–5)
  + **AC**: Temperature control (16–30°C)

Each device tracks its own energy usage based on operating time and power consumption rate.

**3. Automation System**

* **Automation (Interface)**: Defines methods for triggering device events.
* **ScheduledAutomation**: Manages timed device operations using a list of tasks.
* **ScheduledTask**: Represents individual tasks with a device reference, action, and execution time.
* **AutomationThread**: A background thread that continuously monitors and executes scheduled tasks.

**4. File I/O System**

* **FileHandler Class**:
  + saveUsers() and loadUsers() – Serializes/deserializes user data (users.dat)
  + logEvent() – Logs system events
  + readLogs() – Displays logged events
  + exportEnergyReportToCSV() – Exports energy usage to CSV format

**5. Exception Handling**

* **SmartHomeException**: Root custom exception class
  + **DeviceControlException**
  + **AutomationException**

These enable structured and informative error handling.

**System Workflow**

**Startup Process**

1. Main.main() launches the application.
2. A SmartHome instance is initialized.
3. Previously saved user data is loaded.
4. The AutomationThread begins running.
5. The main menu is displayed.

**User Login**

* **Admin**: Authenticated using username and password. Access to user and device management.
* **Regular User**: Authenticated using username and password. Access to control devices and schedule automation.

**Device Control**

1. User selects a device.
2. Device-specific controls are shown:
   * Turn ON/OFF for all devices
   * Adjust brightness (Light)
   * Adjust speed (Fan)
   * Adjust temperature (AC)
3. The system validates the action, applies changes, and updates energy consumption.

**Automation Scheduling**

1. User selects a device and specifies delay time and desired action.
2. A ScheduledTask is created.
3. AutomationThread executes the task at the scheduled time.

**Admin Features**

Admins can:

1. View, add, and remove users
2. View, add, and remove devices
3. View real-time energy consumption
4. View system logs
5. Save and logout

**Regular User Features**

Regular users can:

1. View and control devices
2. Schedule automation tasks
3. Logout

**Data Persistence**

* **User Data** is saved to users.dat when changes occur or on logout.
* **System Logs** record:
  + User logins/logouts
  + Device controls
  + Automation activity
  + System errors
* On restart, user data and logs are reloaded.

**Conclusion**

The **Smart Home Automation System** demonstrates a comprehensive implementation of Java's object-oriented features. Its modular structure, reusable components, and extensible framework make it a powerful base for developing real-world home automation solutions. Future enhancements may include support for more device types, mobile integration, or real-time monitoring dashboards.