Internet of Things Automated Home

Assignment 4 Seng 330, Fall 2018

Kai Kovacik kkovacik@gmail.com
Seth Tolkamp stolkamp@uvic.ca

Note: If the tests or UI Application Client.java exit with "java.lang.reflect.InvocationTargetException", comment out line 50 from Client.java. This line loads the style sheet that decorates the UI. We had trouble getting it to consistently load across platforms and we do not know if it is a local error or not. Additionally, note that the styles.css file is located in assn3-k-s\build\classes\java\main\ (\..\ wasn't cooperating).

How to Test

There are two ways for you to test the system.

1) Type 'gradle run'. This will run the UI Application Client.java and you can easily verify the functionality manually. The client is loaded with three predefined users, "kai" with pass "iak", "seth" with pass "htes" and guest with pass "tseug". Both "kai" and "seth" are admin accounts, whereas "guest" and any user that is added via the "New user" button, are basic accounts.
2) Type 'gradle test'. This will run both the UI acceptance tests and the non UI acceptance tests. The UI Acceptance Tests does extensive testing of the use cases. It adds devices, checks their functionality, adds users, assigns users to devices, etc. The non UI acceptance tests test the functionality of the devices only.

System Structure

This system was constructed with the ultimate goal of having a functioning system. It has been a huge learning experience but as we progressed onto AS4 from AS3 we refactored extensively and greatly improved the encapsulation of the system while reducing coupling.

The devices consist of a model and a view. In AS3 we used a controller for the camera device and the controller ended up duplicating code and implementing getters for getters in the camera model. This system does not require a controller as the control code is quite simple in the first place and separation of data storage and control just leads to more work and more code.

The organizer holds a collection of the views of each device. The Client is a Javafx driven tabular application with a tab for configuring the system, for users, and for each device. The Client instantiates the organizer and populates the tabs with the correct views. Each device tab refreshes based on the list of views in the Organizer and objects are added to the Organizer in the "Configuration Tab".

New Classes

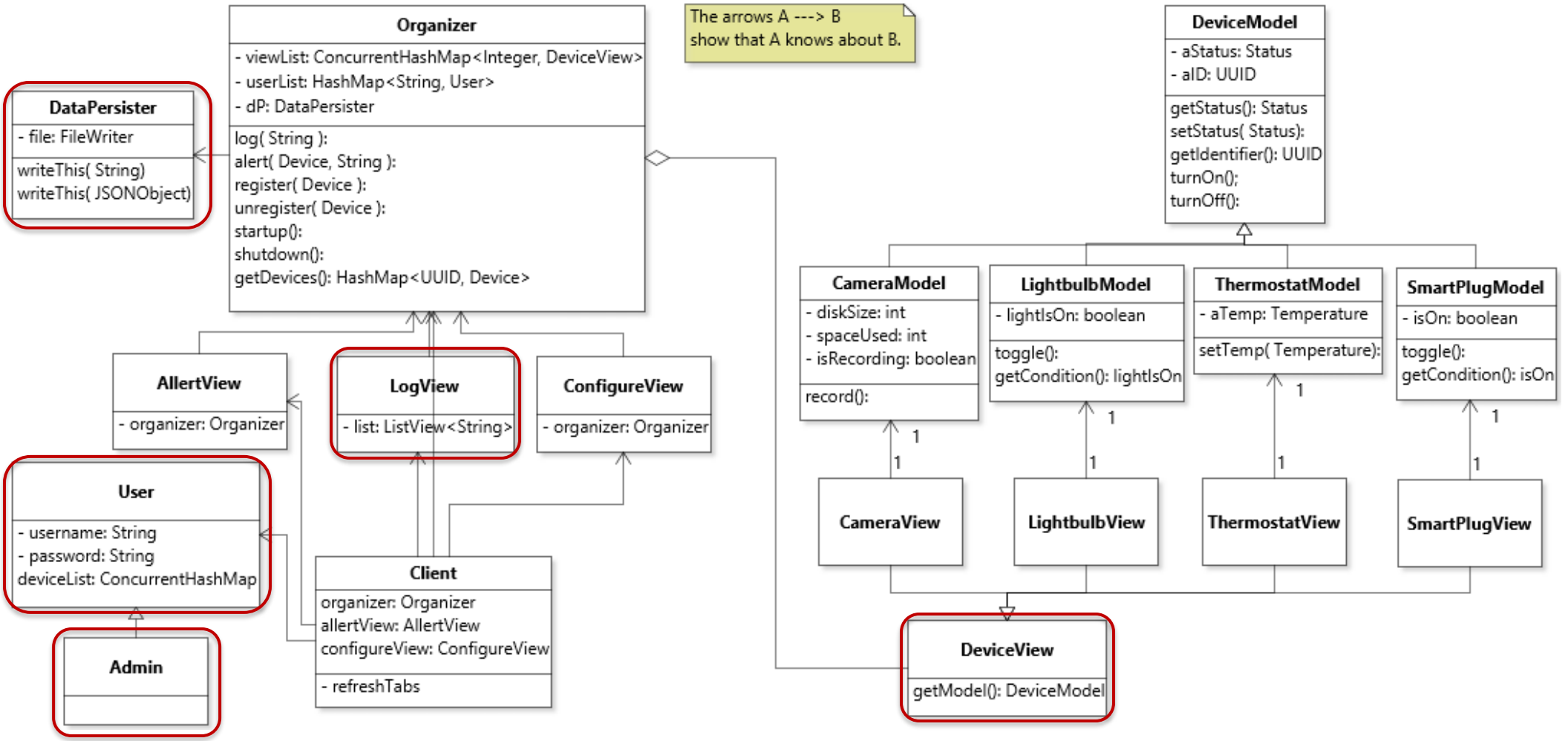
Seen below is the UML class diagram of the system. For clarity, it does not show all methods or attributes. The classes that have been added since AS3 are circled in red.

The important system alerts are displayed at the bottom of the Client. More general information that tracks all activity is displayed in the “Log View” which is a child of the “Configure Tab” where only admins can see it.

Admins and users where added for AS4 as well. Admins can always view all the information. When a new user registers, the active admins can assign them devices in the “Users” tab. By default, users cannot see any devices, they must all be added by an administrator.

The Data Persister class was added in order to save all the “Log View” logs into the file dataLog.txt. It writes all the activity that takes place in JSON format. When the client is shutdown and reopened, the “Log View” repopulates by reading from this file.

Although these are the only new classes, most classes underwent large modifications in order to go from passing half the acceptance tests to all of them and to improve the design.


Organizer
- viewList: ConcurrentHashMap<Integer, DeviceView>
- userList: HashMap<String, User>
- dP: DataPersister
log(String):
alert(Device, String):
register(Device):
unregister(Device):
startup():
shutdown():
getDevices(): HashMap<UUID, Device>
The arrows A ---> B
show that A knows about B.
DataPersister
- file: FileWriter
writeThis(String)
writeThis(JSONObject)
AllertView
- organizer: Organizer
LogView
- list: ListView<String>
ConfigureView
- organizer: Organizer
User
- username: String
- password: String
deviceList: ConcurrentHashMap
Admin
Client
organizer: Organizer
allertView: AllertView
configureView: ConfigureView
- refreshTabs
DeviceModel
- aStatus: Status
- aID: UUID
getStatus(): Status
setStatus(Status):
getIdentifier(): UUID
turnOn();
turnOff():
CameraModel
- diskSize: int
- spaceUsed: int
- isRecording: boolean
record():
LightbulbModel
- lightIsOn: boolean
toggle():
getCondition(): lightIsOn
ThermostatModel
- aTemp: Temperature
setTemp(Temperature):
SmartPlugModel
- isOn: boolean
toggle():
getCondition(): isOn
1
CameraView
LightbulbView
ThermostatView
SmartPlugView
DeviceView
getModel(): DeviceModel