Documentation MQTT Android App

Project EI

Griesser, Gruber, Hinz, Sollacher

June 23, 2021

Contents

List of Figures

1 Overview

1.1 Data Storage

To save all information related to the added devices a textfile is used. After the file has been read, the data is temporarily stored in an array. Whenever the data set is changed, the file is updated. There are two different accesses:

1. Write:
   - AddDeviceActivity adds a line whenever a device is added
   - if a device is deleted in the CommunicationActivity, the corresponding line is removed from the file.
2. Read
   - HomeMainActivity reads in the file to initialize the recyclerview with data

1.2 Logic Schematic

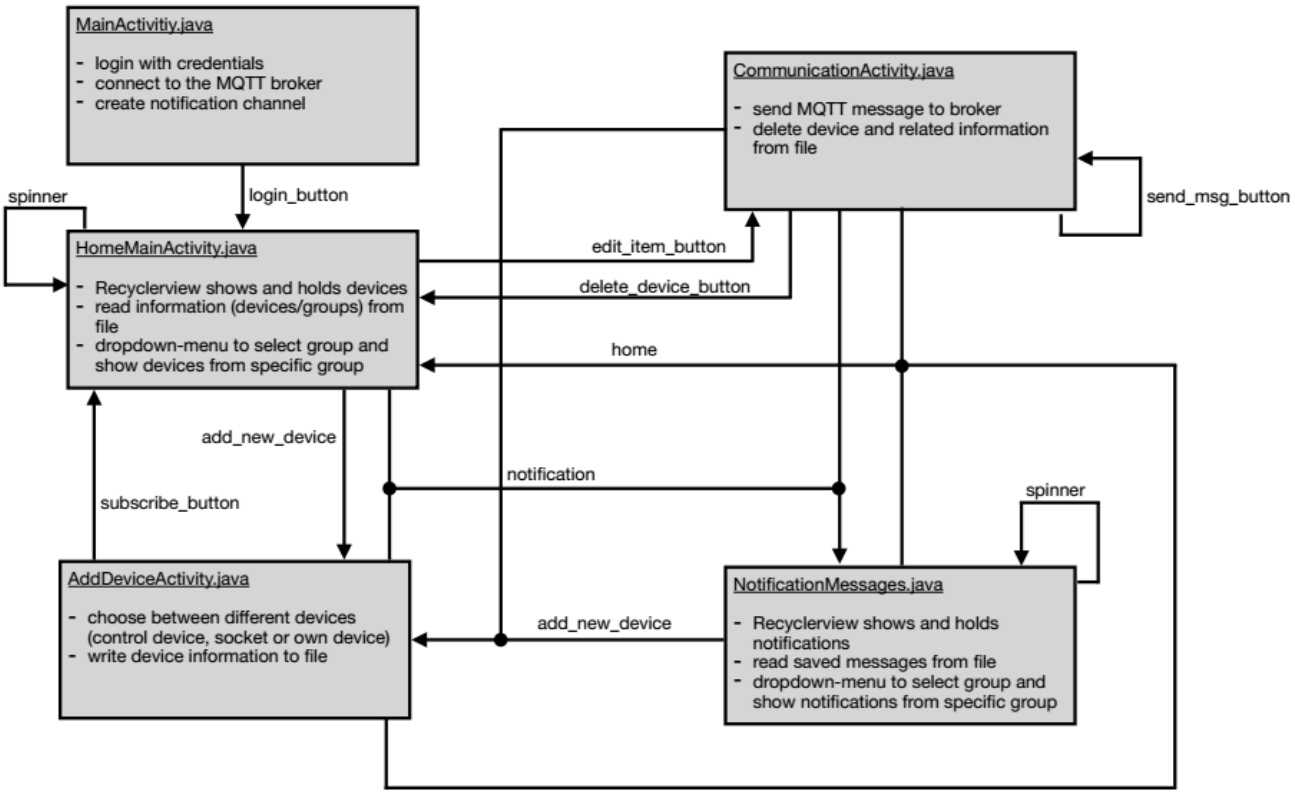

Figure 1: Logic Schematic of the App

2 UML Diagram of the whole App

Here you can see a UML diagram of the whole app. Unfortunately, individual parts had to be removed for the sake of clarity. Each component is explained and broken down in more detail on the following pages.

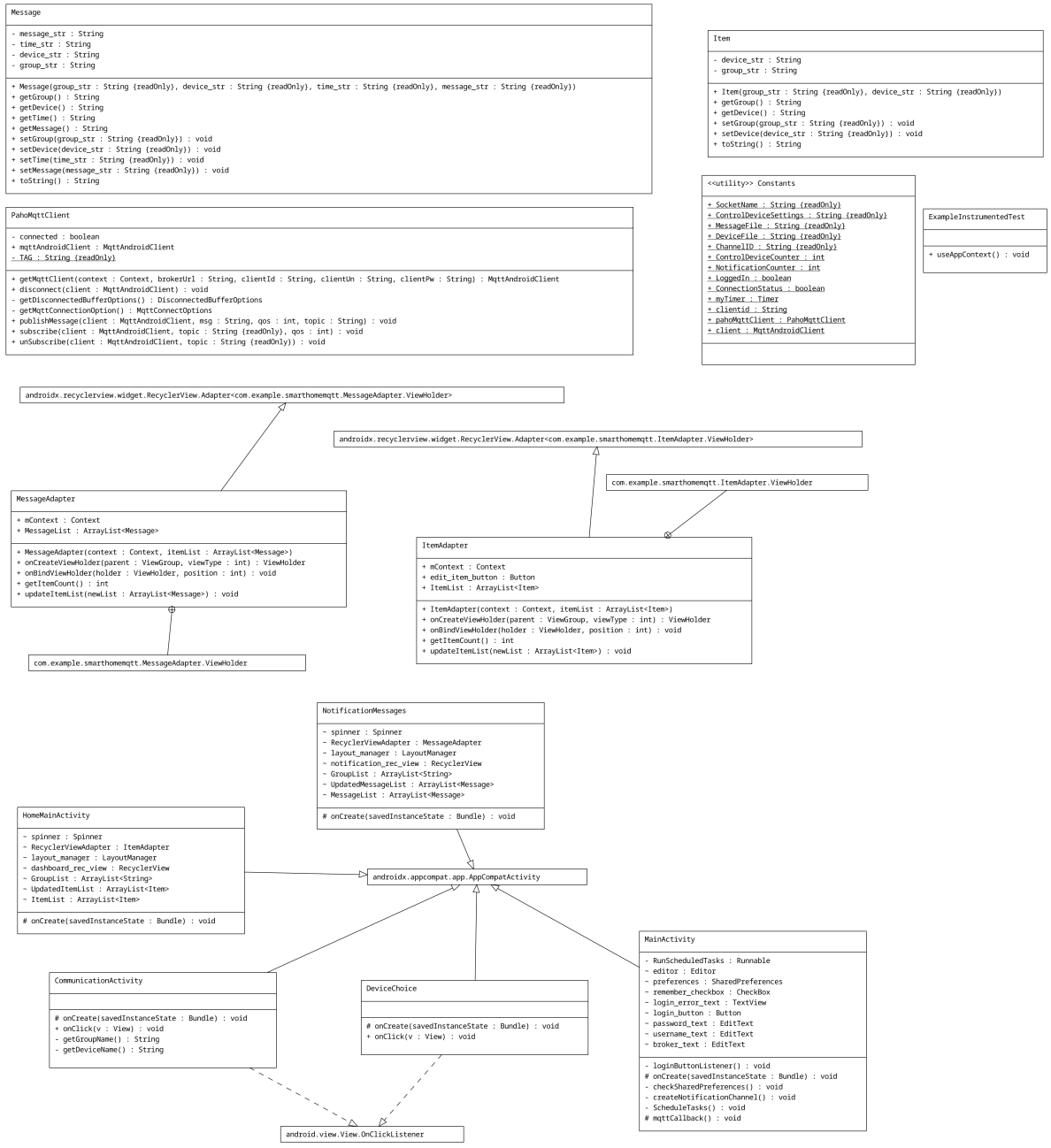

Figure 2: UML Diagramm of the whole App

3 Classes

Here you will find an overview of all files and classes in this project. You can find the most important files here:

app/src/main/java/com/example/smarthomemqtt

Classes and Files:

- MainActivity
- HomeMainActivity
- Constants
- DeviceChoice
- AddDeviceActivity
- AddControlDevice
- AddSocket
- CommunicationActivity
- Item
- ItemAdapter
- Message
- MessageAdapter
- NotificationMessages
- PahoMQTTClient

3.1 MainActivity

The MainActivity is the login screen of our app. Here, the connection to the MQTT broker is initialised and all constants are saved. In addition, a notification channel is set up on which the receipt of a message appears later. The MQTT callback method is responsible for receiving messages. In addition, it also takes care of disconnections. It runs continuously in the background. In the future, however, an additional background thread should be considered. Because as soon as the app is completely closed, the connection to the server ends and no more messages can be received.

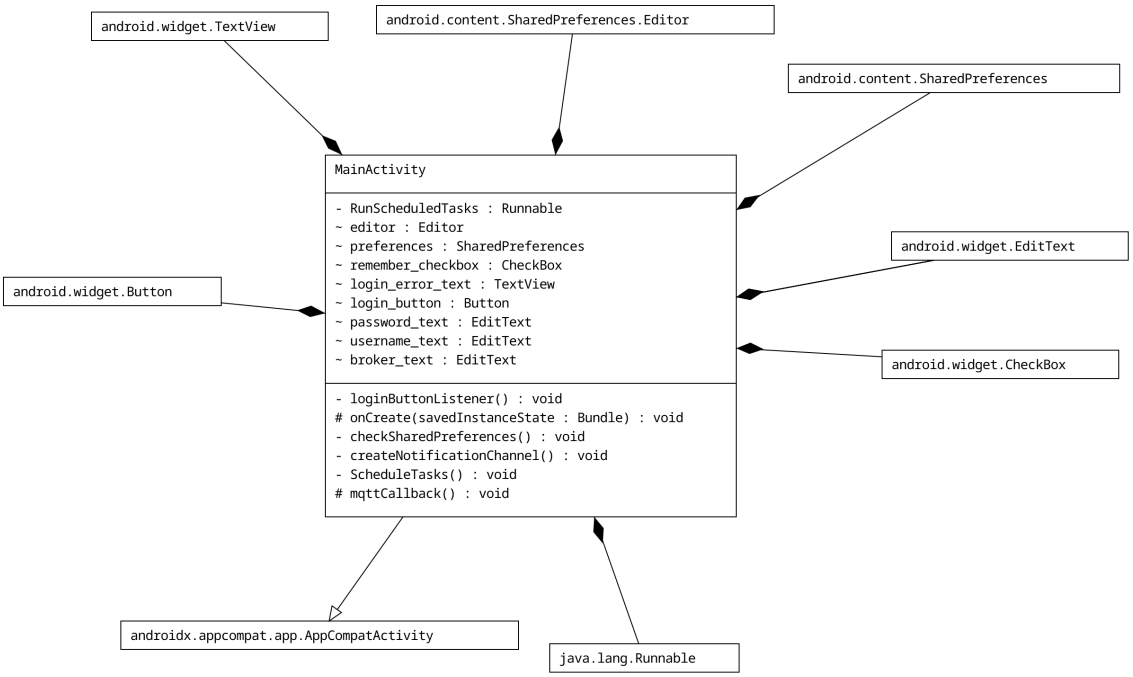

Figure 3: UML Diagram of the MainActivity

3.2 HomeMainActivity

The HomeMainActivity is the dashboard of our app. Here the user can see every single device that he or she has saved and can therefore also operate. The drop-down menu at the top of the screen can also be used to sort by groups. The button edit takes the user to the CommunicationActivity. The structure of the individual items is regulated with a recyclerview. The individual elements are loaded into an array via a text field. The layout of the page is the home_main.xml layout. The individual items are loaded into it with the layout dashboard_item.xml.

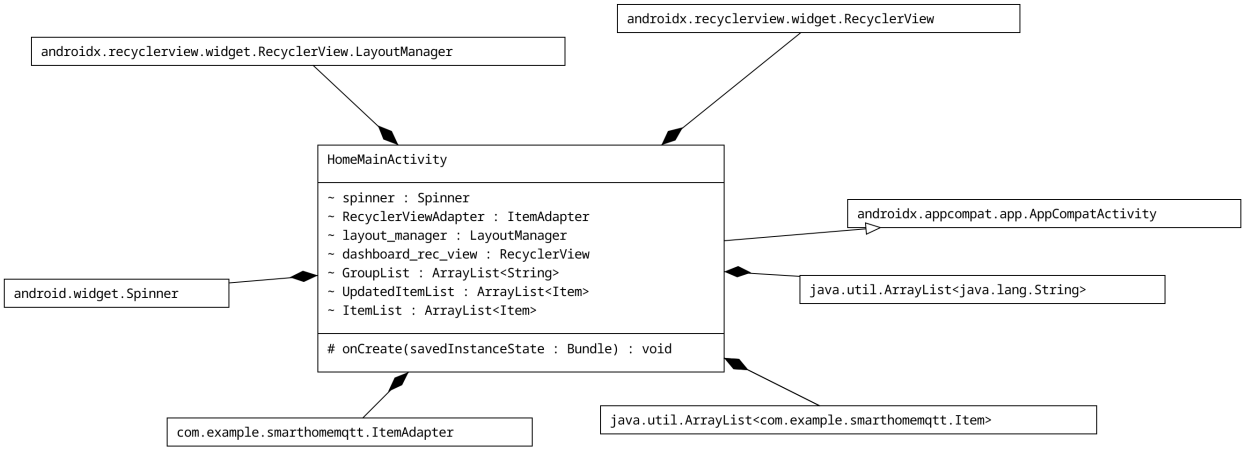

Figure 4: UML Diagram of the HomeMainActivity

3.3 Constants

The Constants class is globally readable and writable via the app. All important constants can be found here. The majority are for the PahoMQTTClient, which must always connect to the same server with the same ID.

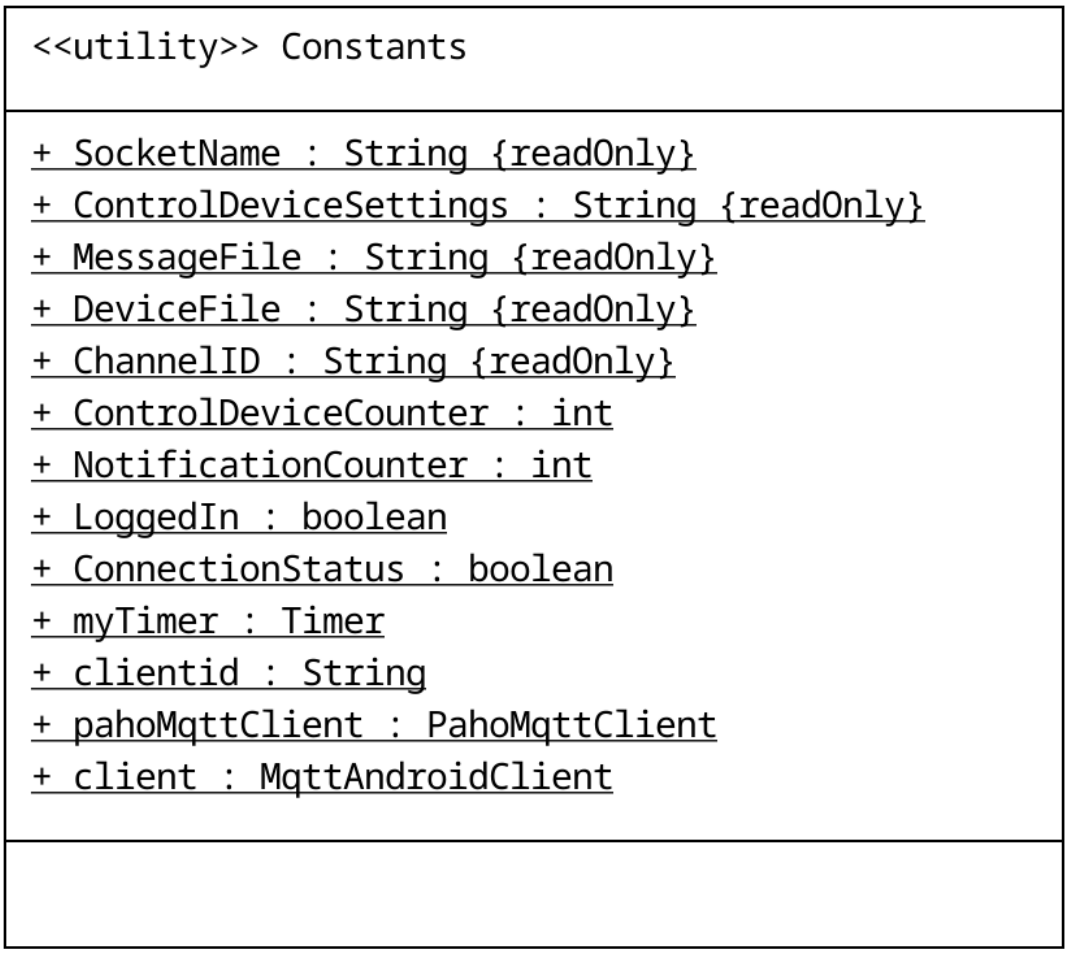

Figure 5: UML Diagram of the Constants

3.4 DeviceChoice

The DeviceChoice class is the parent class before the actual adding of the special item. Here the user chooses which item to add. A choice can be made between these devices

- Control Device
- Socket in a Control Device
- Own Device (With subscribe option)

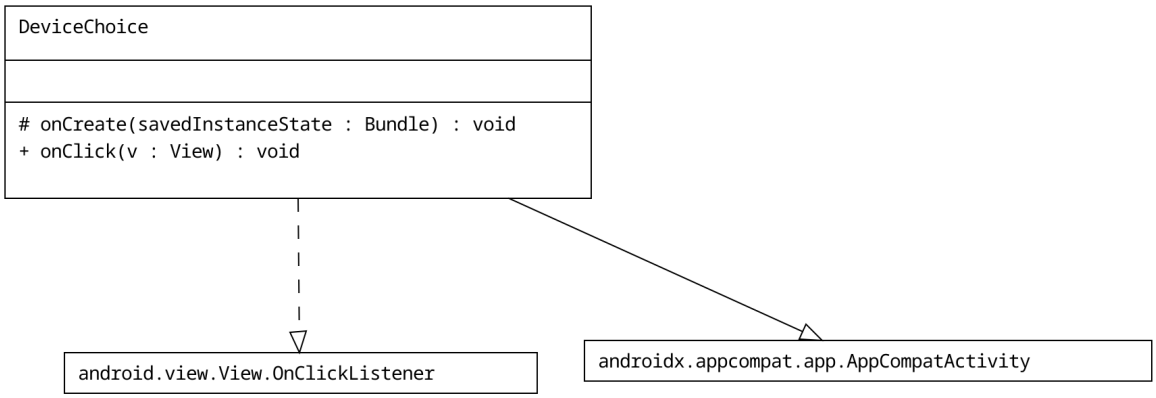

Figure 6: UML Diagram of the DeviceChoice

3.5 AddControlDevice

The AddControlDevice class adds a new control device. It sends a string with the name of the new control device to the topic: control-device/settings.
The control device takes its new name from this string. This makes it feel addressed whenever a string is sent to this group. In addition, the four pins that are present on every control device are added. After a successful save, the user is informed via a toast that the save was successful. Afterwards, the user is automatically redirected to the dashboard.

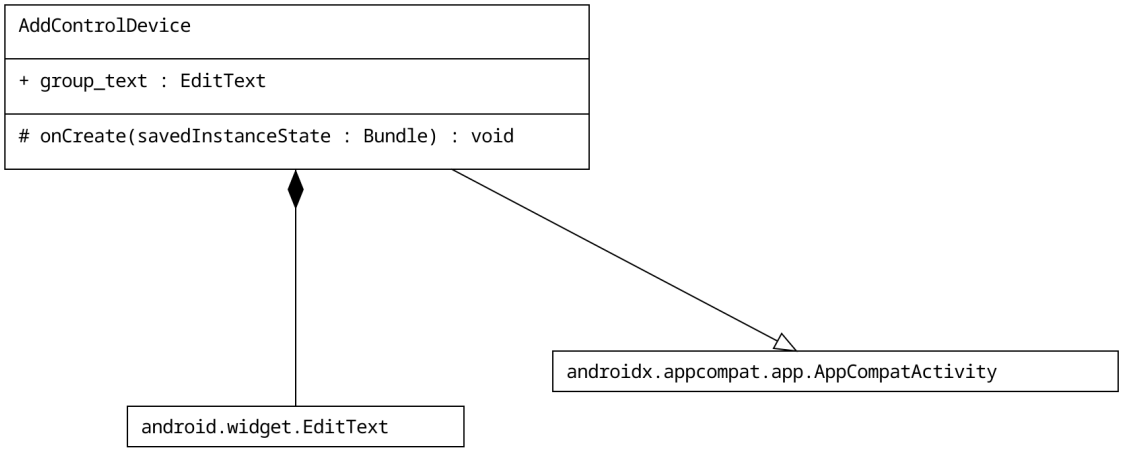

Figure 7: UML Diagram of the AddControlDevice

3.6 AddDeviceActivity

The AddDeviceActivity class adds a general device. This can be addressed via strings. The user must inform himself about the manufacturer's specifications and observe them. When saving, the device can also be subscribed. This allows the user to receive push notifications. This is important for a motion detector or a temperature sensor, for example.

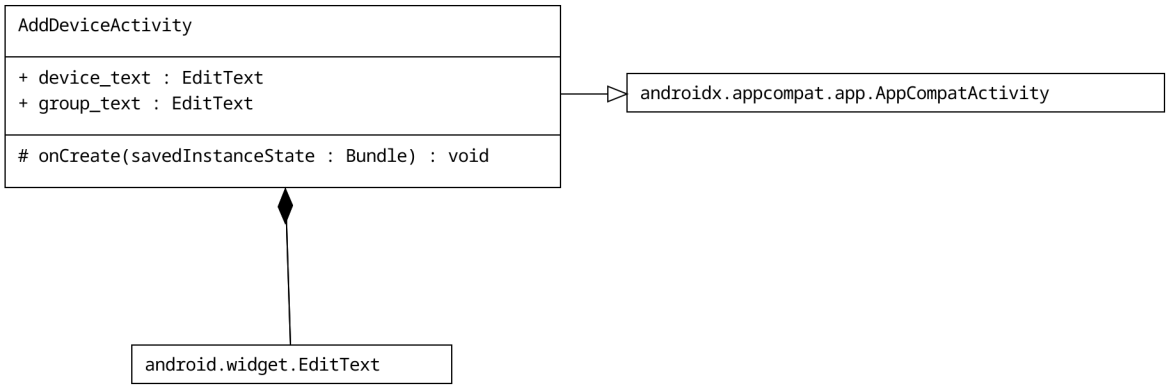

Figure 8: UML Diagram of the AddDeviceActivity

3.7 AddSocket

The AddSocket class adds a 433 MHz socket to a previously set up Control Device. For this, the group of the control device must be specified in the app. However, the control device only adds the socket when it receives the message "on". The socket must be in pairing mode at this time.

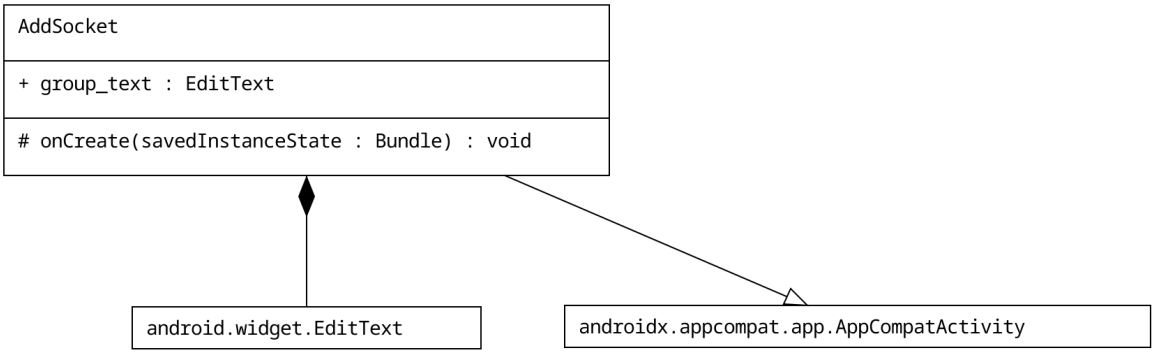

Figure 9: UML Diagram of the AddSocket

3.8 CommunicationActivity

The CommunicationActivity class handles the sending of MQTT messages. The user can enter his message here and send it to the device. The class gets the topic to which it must publish via the private methods getGroupName and getDeviceName.

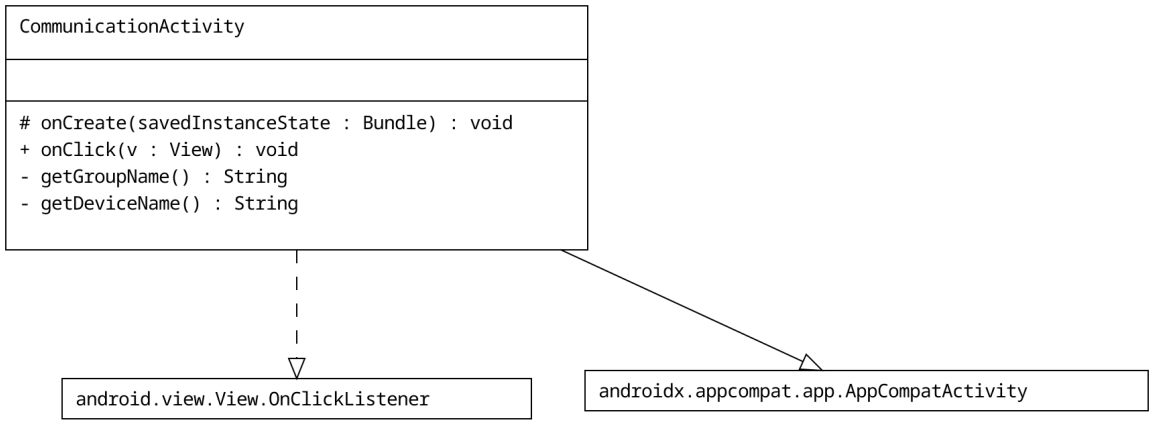

Figure 10: UML Diagram of the CommunicationActivity

3.9 Item and ItemAdapter

3.9.1 Item

The individual devices are stored in the Item class. The method has the attributes Group and Name. These have setter and getter methods respectively. The individual devices can thus be stored in an array of individual items.

3.9.2 ItemAdapter

The ItemAdapter class contains an array of the individual Items objects. This ItemAdapter is used to initialise the RecyclerView and the respective Onclick methods.

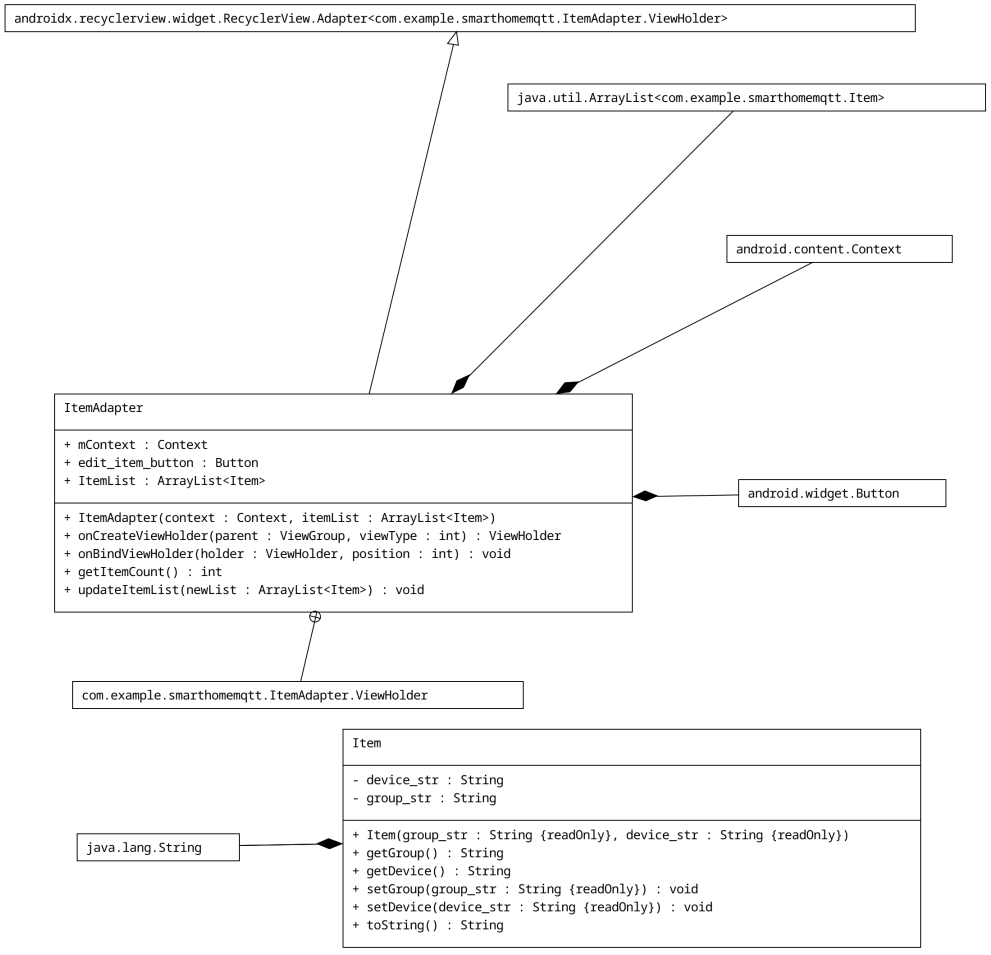

Figure 11: UML Diagram of the Item and the ItemAdapter

3.10 Message and MessageAdapter

3.10.1 Message

The class Message is the equivalent of the class Item. It contains the group name, the device name and additionally the message and a timestamp. The individual attributes have both getter and setter methods.

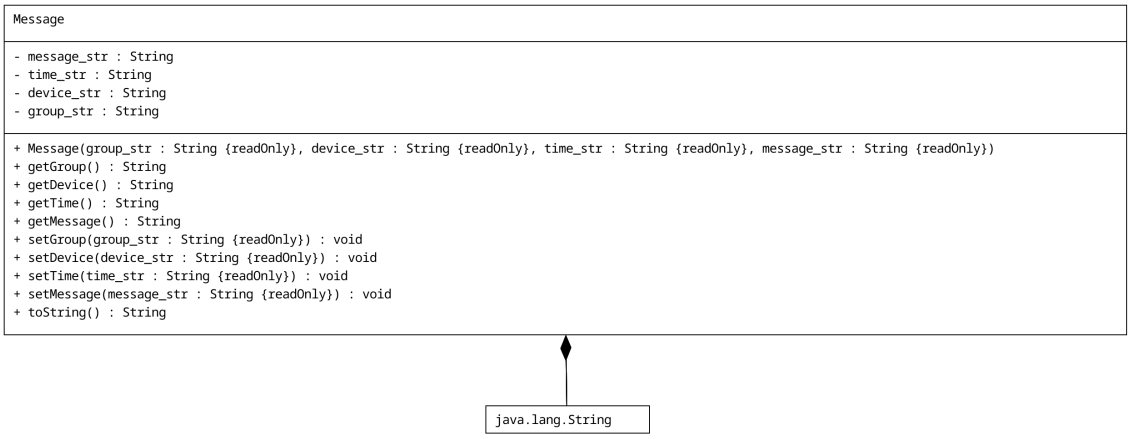

Figure 12: UML Diagram of the Message

3.10.2 MessageAdapter

The MessageAdapter class contains an array of the individual Message objects. This MessageAdapter is used to initialise the RecyclerView.

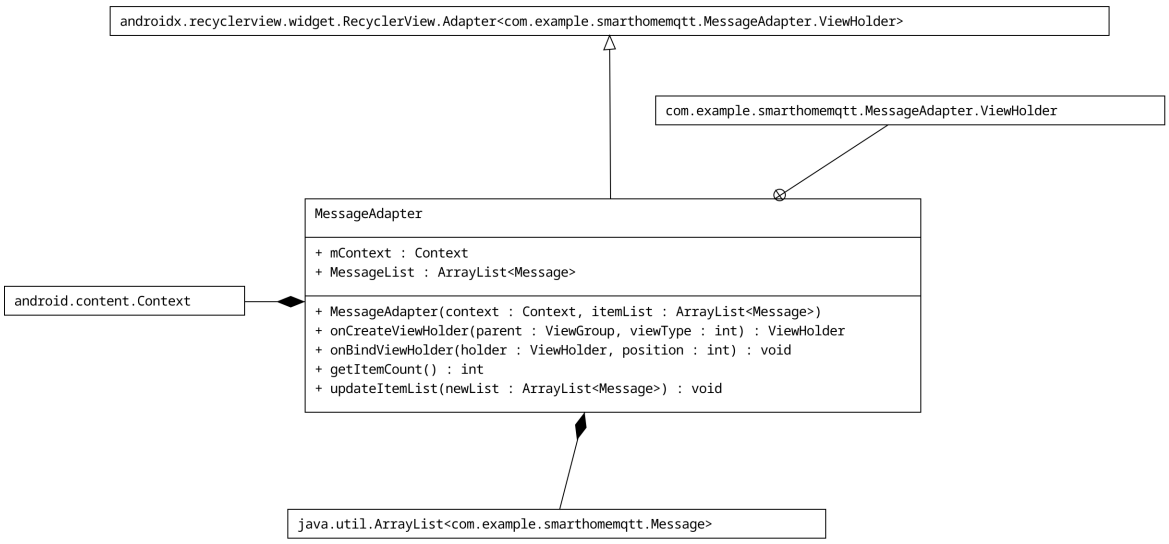

Figure 13: UML Diagram of the MessageAdapter

3.11 NotificationMessages

The NotificationMessages class displays the individual messages received. They are rendered into the layout with a RecyclerView as in the Dashboard. In addition, the respective group can be selected via a drop-down menu at the top of the display.

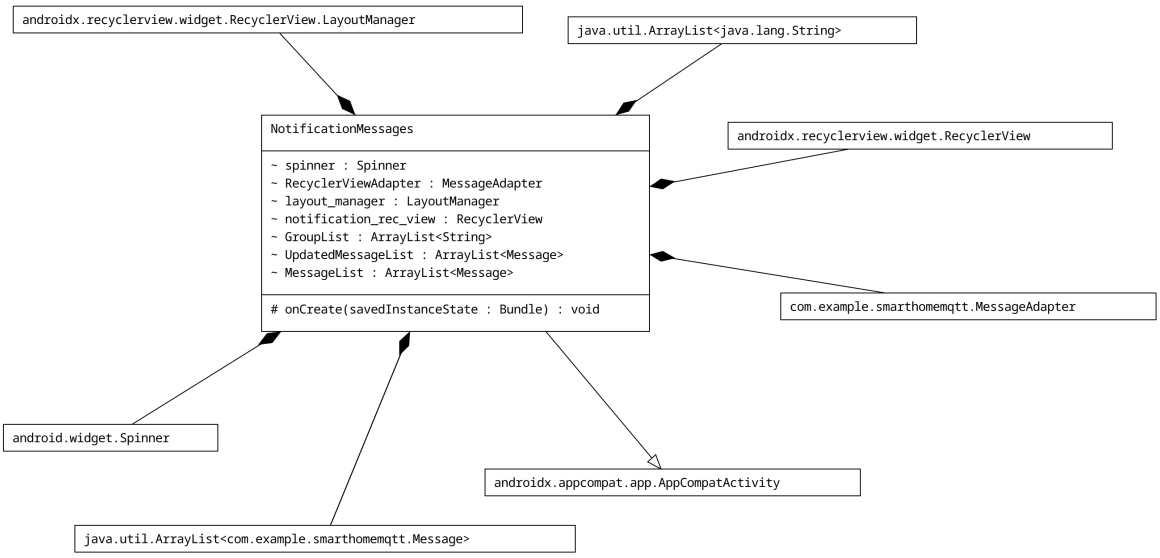

Figure 14: UML Diagram of the NotificationMessages

3.12 PahoMQTTClient

The class PahoMQTTClient represents a kind of wrapper for the Java Paho MQTT library. It facilitates the call of these functions and provides some fallback options.

Basically, the following functions are offered:

- Connect to MQTT Broker
- Disconnect from MQTT Broker
- Publish a Message to MQTT Broker
- Subscribe to a specific Topic on MQTT Broker
- Unsubscribe from a specific Topic on MQTT Broker

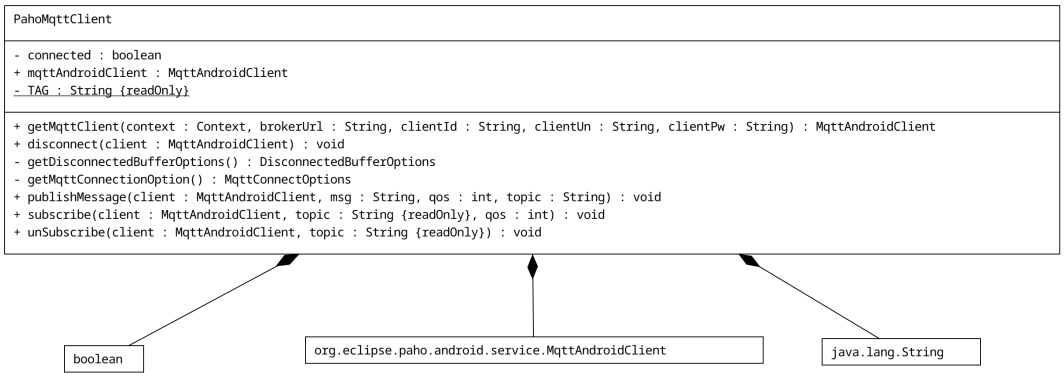

Figure 15: UML Diagram of the PahoMQTTClient