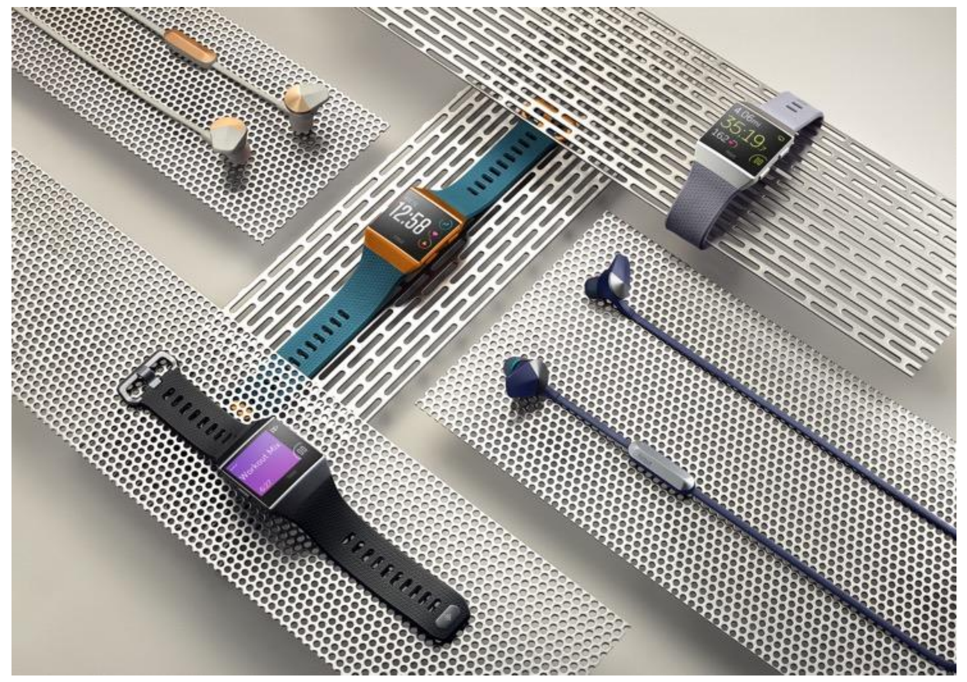


Android: Data Management

11.07.2019

fitbit

Objectives

- Use Shared Preferences to store app settings
- Store data locally in database using Room library
  - timestamp series collected from Nordic Thingy about air pollution data
- Use background threads using Runnables and Handlers

Recap last sessions

- In the previous three sessions you have:
  - Created Activities
  - Used UI elements such as Buttons, TextViews and RecyclerViews
  - Used permissions to enable for Bluetooth and Network communication
  - Interacted with a public API using Retrofit for HTTP operations
  - Used Intents and BroadcastReceivers to communicate between Android components
  - Used Nordic thingylib to connect and receive data from the Thingy board
  - Learned basic concepts related to BLE: services, characteristics, states
- Code on github: https://github.com/Fitbit/FitbitSummerSchool2019/

Data storage in Android

- Shared pref
- Internal storage
- External storage
- Databases

SharedPreferences

- `SharedPreferences` APIs allow you to read and write persistent key-value pairs of primitive data types: booleans, floats, ints, longs, and strings.

SQL

SQL is a standard language for storing, manipulating and retrieving data in databases.

- `SELECT * FROM Customers;`
- `DELETE FROM Customers WHERE CustomerName='Quentin Tarantino';`

Databases in Android

- Android provides full support for SQLite databases.
- Database operations can’t be run on the UI thread
- Also we should have only one instance linking to the database
- Any database you create is accessible only by your app. However, instead of using SQLite APIs directly, we recommend that you create and interact with your databases with the Room persistence library

Room

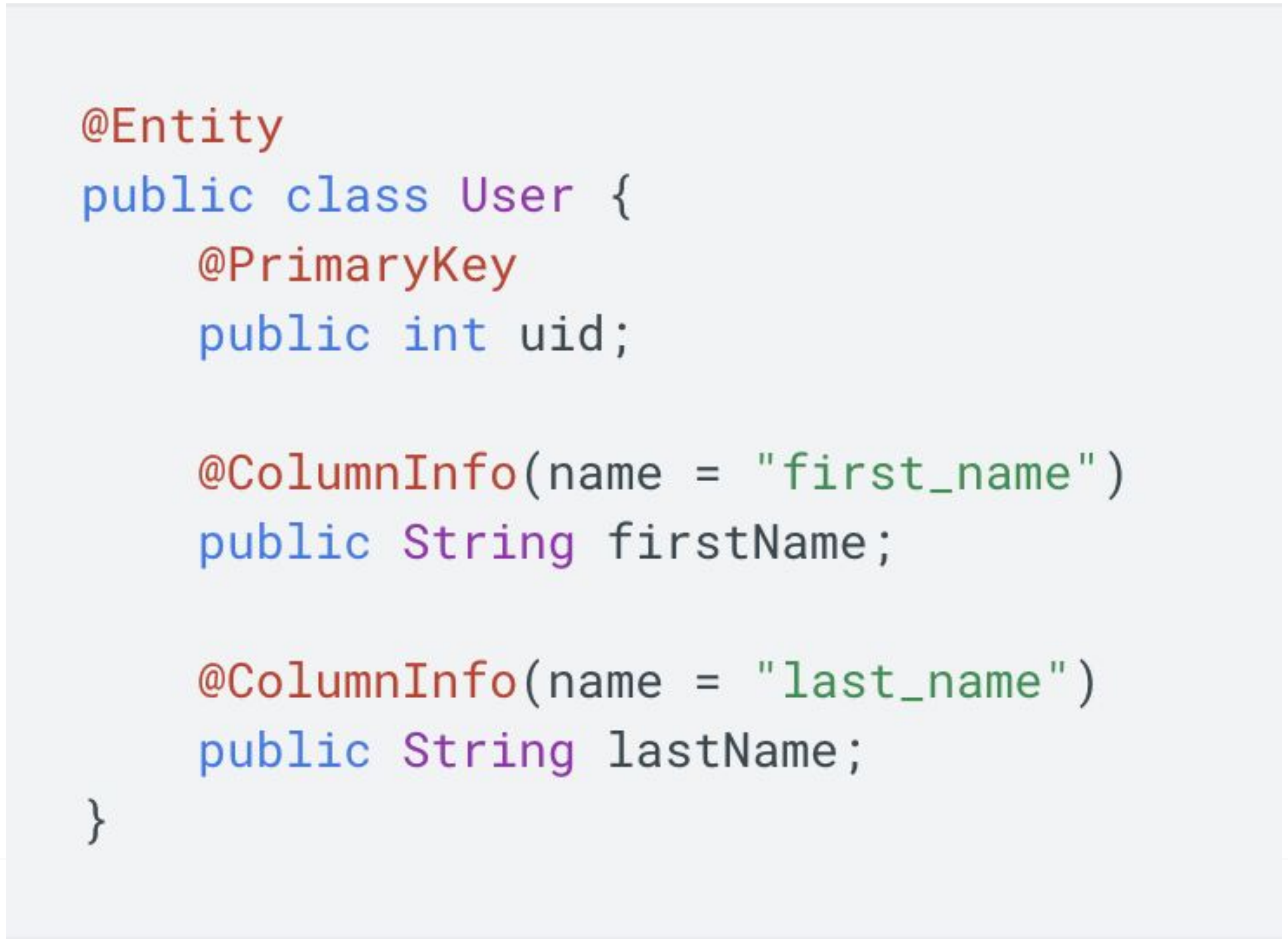
Why use Room library

- compile-time verification of raw SQL queries.
- as your schema changes, you don't need to update the affected SQL queries manually.
- less boilerplate code to convert between SQL queries and Java data objects.

Room

There are 3 major components in Room:

1. **Entity:** Represents a table within the database.



```
@Entity
public class User {
    @PrimaryKey
    public int uid;

    @ColumnInfo(name = "first_name")
    public String firstName;

    @ColumnInfo(name = "last_name")
    public String lastName;
}
```

Room

2. **DAO:** Contains the methods used for accessing the database.

```
@Dao
public interface UserDao {
    @Query("SELECT * FROM user")
    List<User> getAll();

    @Query("SELECT * FROM user WHERE uid IN (:userIds)")
    List<User> loadAllByIds(int[] userIds);

    @Query("SELECT * FROM user WHERE first_name LIKE :first AND " +
           "last_name LIKE :last LIMIT 1")
    User findByName(String first, String last);

    @Insert
    void insertAll(User... users);

    @Delete
    void delete(User user);
}
```

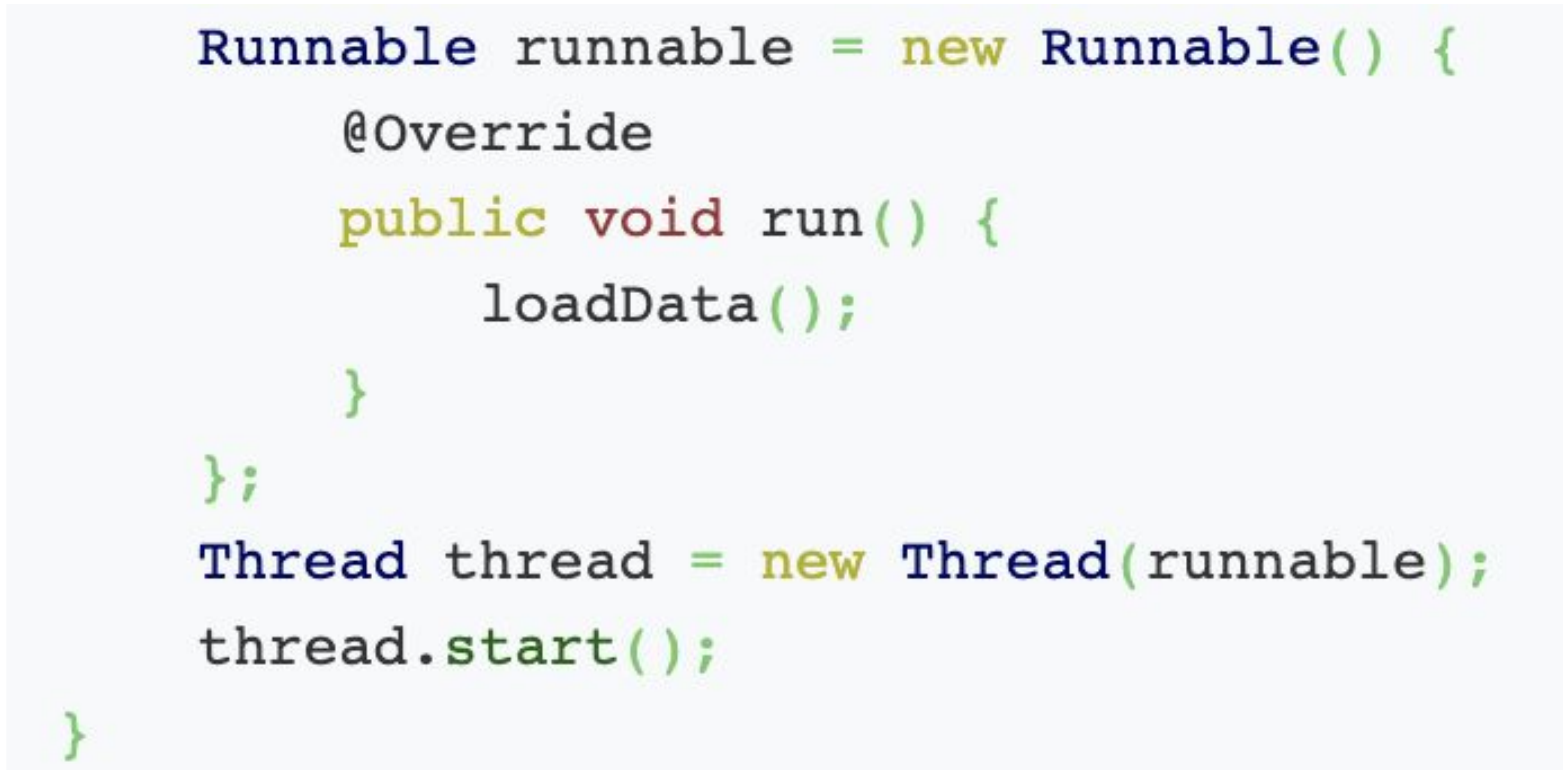
Room

3. **Database:** Contains the database holder and serves as the main access point for the underlying connection to your app's persisted, relational data.

```
@Database(entities = {User.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {
    public abstract UserDao userDao();
}
```

Threading

To run something on a different thread we can create a Thread and pass it a Runnable



```
    Runnable runnable = new Runnable() {
        @Override
        public void run() {
            loadData();
        }
    };
    Thread thread = new Thread(runnable);
    thread.start();
}
```

Threading

Using handlers we can schedule a thread

```
HandlerThread thread = new HandlerThread("I handle things");
thread.start();
writeHandler = new Handler(thread.getLooper());

writeHandler.postDelayed(new Runnable() {
    @Override
    public void run() {
        try {
            saveDataToDB();
        } finally {
            writeHandler.postDelayed(this, interval);
        }
    }
}, interval);
```

This session's App

- Extend the communication prototype for the ThingyBoard implemented in the last session
- New Features and flows:
  - Receive air pollution data from the board
    - Subscribe to notifications from the AirQuality characteristic! Todo adriana need to verify which is it!
  - Store data in the database
  - Settings Activity to set the some interval, batch frequency etc
  - Alerts (aka notification) when air pollution is bad, based on the setting to enable it - cod pe wiki