Ilkka Jokela

Bank Application

30th April 2020

Project Documentation

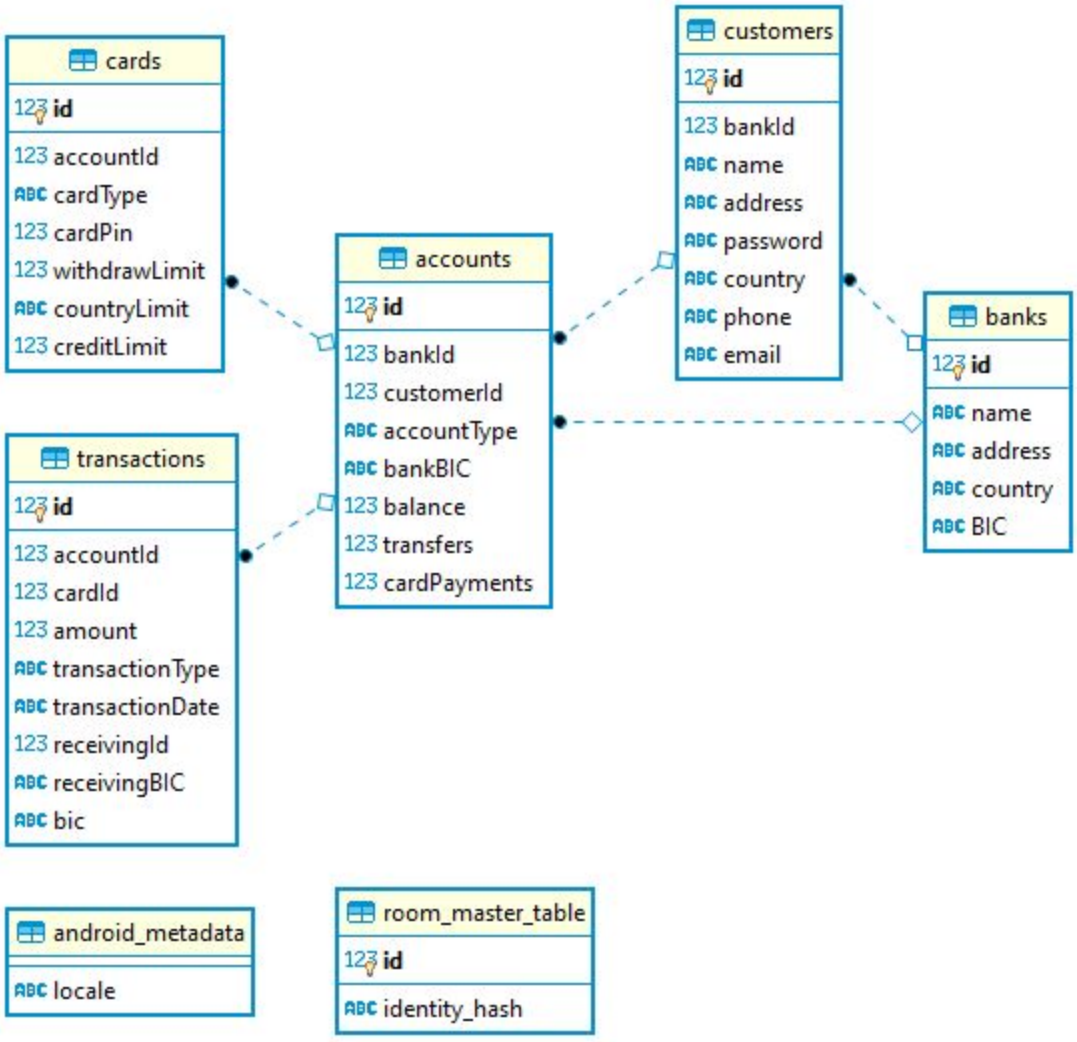
As for my LUT's object oriented programming course project I did a mobile bank application. The bank application is running on Android 8.0. The implementation was coded with Java programming language and with object oriented design. Android Studio was the choice of the IDE. Project documentation and source code is stored to Github repository https://github.com/IleJok/BankApp. The assignment and requirements can be found here. One of the requirements for this project was to create a design plan. Design plan is also stored in the same github repository https://github.com/IleJok/BankApp/blob/master/app/docs/Project%20design.pdf. Video presentation of the project is available on Youtube for now.

Bank application uses a relational database SQLite to store important information. Below you can see the Entity Relationship diagram (ERD). The ERD is created with DBeaver which is a free universal database manager. As you can see from the ERD, I have avoided many-to-many relationships due to use of Room persistence library. If you use many-to-many relationships with Room, you have to make associative entities and two extra classes for each many-to-many table. Due to the lack of many-to-many tables, transactions between accounts, transfers have to be fetched with both accountId and receiverId.

In my project I aimed to use as much as possible best practices and that is why I used architecture components recommended in https://codelabs.developers.google.com/codelabs/android-room-with-a-view/#0 as the basis of my application. The main idea of the architecture components is to separate UI and business logic so that you can build scalable applications. I have added the picture of the recommended architecture components below (Google). To be honest, I did not succeed totally with the recommended setup. All the pieces are there, but the implementation is not done totally by the book. The main issue with my implementation is that I allow queries on the main thread which is not recommended at all. I should have implemented a multithreaded solution with proper asynchronous reads and writes from the database. But to be fair to myself, this was my first Android application and this course was the first time I coded with Java.

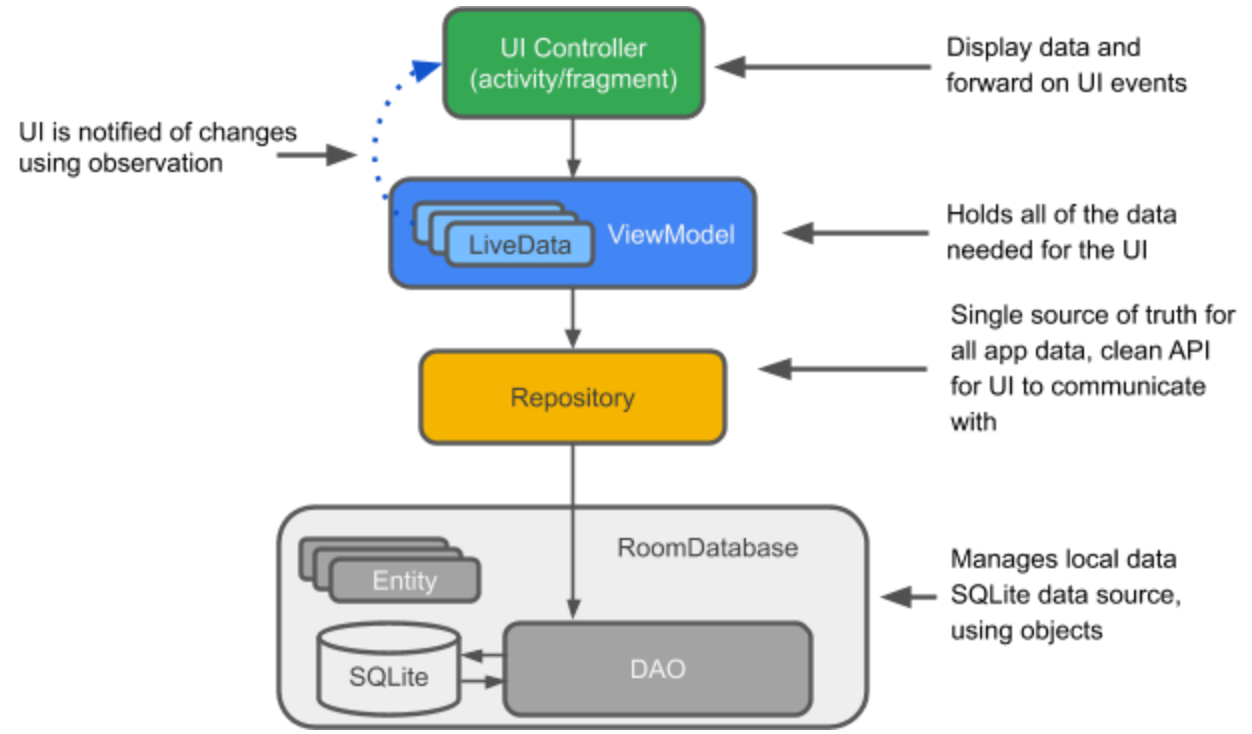
In addition to recommended architecture components I used conditional navigation in my application. This means that the user has to log in or register to application before the user is able to do stuff. More information about conditional navigation https://developer.android.com/guide/navigation/navigation-conditional. I started the implementation from the classes I had in design plan. From the beginning I had decided that I want to use a relational database in my application. From my professor I had learned that Android developers is good source for documentation and guides in addition to basic Stackoverflow, so there I went and found the Android Room with a View guide. Instead of creating many activities as in the guide, I used fragments as layouts for my different views. In the third picture I have all the different fragments represented in the navigation graph except register and password fragments which are under a nested navigation graph.

In the fourth picture I have an simplified UML class diagram of my project. I have listed all the class variables which are not stored in the database and therefore are not visible in the ERD. In the interest of clarity, I have left out all the getter and setter methods which are basic and provide no valuable information to the reader. I have also left out of the picture all the DAO classes which hold the SQL queries to db. Instead I have all the important methods of repository classes which provide access to DAOs. If I had used external data sources like open apis, then the access to those would also have been through repository classes. CSVWriter class is used to write the most important things to file like new accounts, transactions and cards.

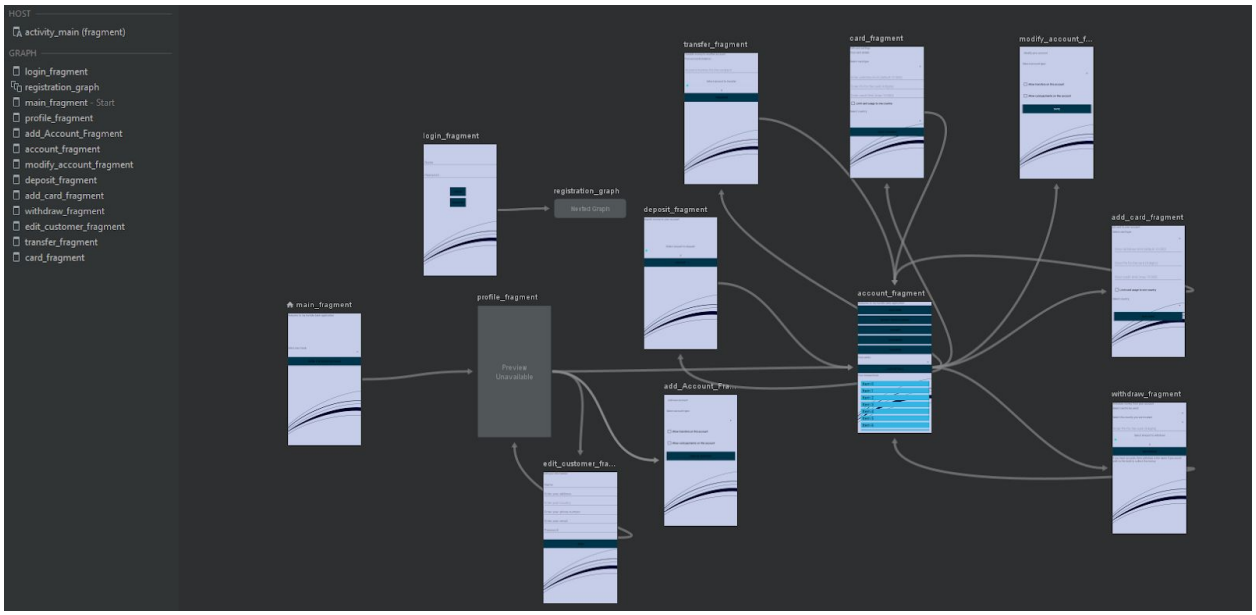


[1]

[1] Entity Relationship



2

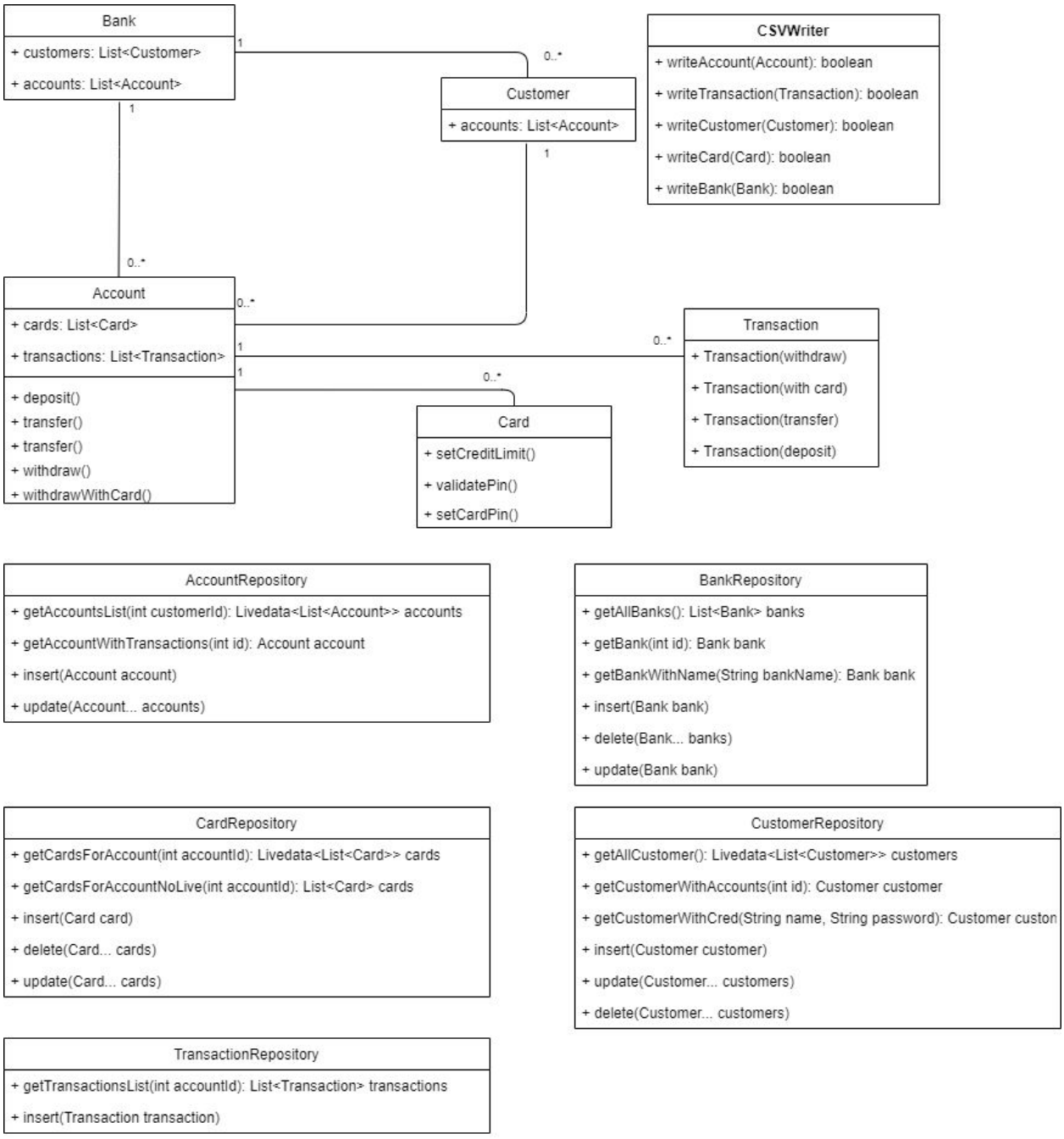


3

---

[2] Recommended architecture components. Source: https://codelabs.developers.google.com/codelabs/android-room-with-a-view/

[3] Picture of different fragments in the navigation graph.



4

[4] UML class diagram

SPECIFICATIONS AND FEATURES

There were several mandatory features which were required to be implemented in the application for passing the project assignment. Some of the requirements were universal for all of the projects and some were subject specific. These mandatory features give 13 points out of a maximum 40 points. Three points was also granted for the design plan if it was properly done. Here are the mandatory features listed (both the universal and bank application specific):

1. Use object-oriented programming (universal).
2. At least five different classes & objects in addition to GUI classes (universal).
3. Writing most important information to a file with XML, JSON or CSV data format (universal).
4. Users are able to edit their information (specific).
5. Users are able to create multiple accounts (specific).
6. Users are able to modify account features e.g. account type and whether you are able to transfer money to another account (specific).
7. Users are able to create cards (debit/credit) for their accounts (specific).
8. Users are able to view their account transactions like deposit, transfers, withdrawals etc (specific).
9. Users are able to add money to their account.

In addition to above listed mandatory features, there was incentive to add more features to the application to get more points for the project. Below is listed additional features which were implemented to the project and points that I want for the features:

1. Application uses well designed user interface components (2 points).
2. Database SQLite for storing information (5 points).
3. More users and users are stored into database (3 points)
4. Logging in to the application (3 points).
5. Extra features: Conditional navigation with navigation graph and nested navigation graph and use of Android developer's recommended app architecture (6 points).
6. Users are able to change rights and limits of cards (1 point).
7. Users are able to transfer money from user's own account to another (1 point).
8. Users are able to make money withdrawals or card payments (1 point).
9. Users are able to transfer money to an outside account and can transfer money between users (1 point).
10. Cards have country limits and the money withdrawals and payments can be tested in different countries (2 points).

11. System has multiple banks and each has their own users and accounts. Transactions, account information and payments need to show the bank's BIC code (3 points).

When I add up all the points from mandatory features (13 points), design plan (3 points) and additional features (25 points), I get a total of 41 points.

Reflections

Amount of work

One of the requirements for this documentation was to give an approximation of hours spent on the project. I started coding this project 3 weeks ago according to Github, and I have almost every day done at least several hours of coding, but I had one weekend when I did nothing on the project. If I count also the hours for the design plan and this documentation, I am sure that I have done approximately 90 - 115 hours of work for this project. As an adult student/open university student I consider that is quite a lot for one project, but the project was interesting and I hadn't done Android applications before. I also had quite a lot of time due to a temporary lay off from work caused by Covid-19.

Learning outcomes

I learned a lot from this course and this project. As I mentioned before, I had no prior experience with Java or Android before this course. I am familiar with C# which is quite close to Java at least what comes to syntax and object-oriented programming. What I learned most from the project was how to develop applications to Android. I learned how MVVM is done (or should be done) in Android mobile applications. Use of repositories is quite similar what I have been used to in regular Rest APIs with C#.

What I didn't totally learn is how asynchronous calls to database and multithreading are done in Android (also not in the scope of this course). I have to admit that I ran out of time and therefore allowed querying on the main thread. I learned that Room persistence library is a very useful addition to the application, because it enables querying livedata from the database. But at the same time I learned that many of the common relationships are not so easy to implement with Room. As I mentioned before, there should be one many-to-many table for money transfers in between customers. In that table I would store senders id, receivers id and the transaction id.

Looking back to this course and its different weekly modules, I can say that I have used knowledge from each module during this project and in my case, the objective of the course fulfilled. What I would continue with this project is to move the database from the device to cloud, add unit test and make calls to db asynchronous.