LUT

School of Engineering Science

Software Development Skills

Jani Heinikoski, 0541122

LEARNING DIARY, MOBILE MODULE

**LEARNING DIARY**

03.11.2020

I read the course overview and the general course information sections from the course Moodle page. I learned that this course is intended to give me an edge in the industry by providing tools to create unique projects which reflect my current skills as a software developer.

Furthermore, I learned that the main environment used in this course is Android Studio which I have used for approximately 300 hours before this course. I found out that the version control system used in this course is Git which I am already familiar and comfortable with. I made a new public Git repository to GitHub and added this document to it.

After briefly checking out the course module task-list and watching the first video, I realized that there is not going to be much new for me in the “beginner level” videos. This learning diary might thus be a little empty regarding the task modules but when I get to the course project, many entries in this diary are to be expected. I will be sure to come up with new challenges and take this course as a chance to expand my skillset regarding native Android mobile development.

04.11.2020

While working on the Introduction module I learned that the EditText object method getText will not return a null value in any case. Instead, it returns an empty string if the String inside of the widget is empty. Another concept I learned during the Introduction module was theming and styling while following the Material design guidelines (documented in <https://material.io/>). Themes and styles in native Android development are complex concepts due to the levels of indirection (compared to e.g. CSS). I added support for a light (default) theme and a dark theme using Android developers documentation (available: <https://developer.android.com/guide/topics/ui/look-and-feel/themes>, referred 04.11.2020).

While watching the Core Elements -module’s video, I learned that IntentServices handle Intents (I have created my previous applications mainly as a single Activity using fragments which is the best practice nowadays).

After playing around couple of hours with animations, selectors, animated-selectors, shapes, lists etc. trying to create an animation for a custom button by using only XML, I stumbled upon a nice thread in Stackoverflow <https://stackoverflow.com/questions/35400318/custom-button-animation-in-android>. It was extremely hard to find from the Android developer’s documentation how to properly animate a View (specifically a button) using only XML. I have used sort of a bubble gum approach when animating buttons in the past (using mixed XML and Java code). This is the first solution which allowed to fairly simply attach a StateListAnimator to a button.

05.11.2020

As I watched the third module video, I did not really learn anything new. I have previously used RecyclerView instead of ListView in my projects since it is the best practice for displaying large and/or dynamic (set size not yet known) datasets. ListView is much easier to implement and has less overhead so it is acceptable to use it in this case since we only have an (ahead-of-time) known small dataset to display. Furthermore, I only use SVG -vector based image assets as my image assets which can automatically scale to fit a container while maintaining same quality (no fear for app crashing or need to write own scale-down functions).

As I created the base project for the third module (initial commit to module 3), I encountered an error I have not seen before:

* “AAPT: error: failed writing to ‘~\GitHub\sds\_mobile\modules\ListViewDemo\app\build\intermediates\runtime\_symbol\_list\debug\R.txt': The data is invalid. (13).”

I did not find any solid fix for the problem by Googling it, so I tried to downgrade my Gradle -plugin version from 4.1.0 to 4.0.2 which solved the problem.

While working on the module I learned how to use Dialogs by fiddling around with them for a while. Because it was extremely inefficient to change the Activity to display a picture about the clicked item, I used a dialog to display the clicked item.

14.12.2020

I started working on the course project by first brainstorming ideas. I ended up with choosing a “coinline speed test” type of game (seen in a Finnish television game show called *“*[*Speden Spelit*](https://en.wikipedia.org/wiki/Speden_Spelit)*”*) which uses SQLite database (with room) to store high scores. I haven’t used Room before, so I read Android developer’s documentation about it and learned that it’s basically an abstraction layer for SQLite’s own API (<https://developer.android.com/training/data-storage/room>). It takes care a lot of tedious work one would otherwise have to do and is worth adding the dependency.

When I started working on the preliminary GUI, I had to revisit Android Developers documentation about fragment’s lifecycles (sources can be found in the source code headers). I had forgotten when ViewModels are supposed to be initialized (onActivityCreated) and when should View Binding be used (onViewCreated).

15.12.2020

I started doing GUI-driven development (create a raw GUI and navigation before actual logic is added). I learned how to customize a ProgressBar -component (see drawable -package’s game\_progress\_bar.xml).

Due to material shape theming I struggled for a while, wondering why my button’s background seems to be locked as the accent color even though I defined it a custom drawable as the background. After an intensive googling session, I found out that you need to define the following attribute to disable this theming (shape theming is enabled on API level 21+ due to material theming):

I stumbled upon a strange bug where my custom Progress Bar wouldn’t appear on API level 23 (tested on two different AVDs and four different API levels). I was able to fix it by removing some “?android:attr/” references from my game\_progress\_bar.xml (see commit “hotfix for api 23” from today). Also, after two hours of Googling, videos, docs and playing around, I could not get the GridLayout to work on API level 21 (see commented section in fragment\_speed\_test.xml), so I decided to use two nested LinearLayouts to create a two-by-two button “grid layout”.

17.12.2020

I wanted to support screen rotation in my application which led me to find an interesting bug. If I navigated to “speed test mode” and rotated the screen and clicked the back button on my custom action bar, the application crashed. Although my bug did not directly derive from Activity State Loss, I learned that trying to commit a fragment transaction after Activity’s onSaveInstance -lifecycle phase, an IllegalStateException is thrown (Lockwood A., 2013).

My bug stemmed from the fact that my main menu fragment set onClickListeners which used IFragmentOwner -interface as a callback system to change to a new fragment from another fragment. When a new fragment (e.g., speed test fragment) replaced the main menu and the Activity was recreated, the back button was trying to reference the destroyed Activity before recreation. I solved the bug by implementing a SharedViewModel between the Activity and Viewmodels. Now the main menu fragment observes if the Activity is recreated, and it is updated (even if in the background) and updates the Activity’s IFragmentOwner reference to the recreated one in the onClickListeners.

Although I could have denied the user from rotating the screen, it would not have been the correct solution. Android might force the recreation at any given time without any warning especially if system resources are running low. This would have led to the same bug (which would have been even harder to detect).

Sources:

* Lockwood A., 2013. Fragment Transactions & Activity State Loss. [Website]. Referred 17.12.2020. Available: <https://www.androiddesignpatterns.com/2013/08/fragment-transaction-commit-state-loss.html>