Lappeenrannan teknillinen yliopisto
School of Business and Management

Sofware Development Skills

Pyry Santahuhta, 0545254

LEARNING DIARY, MOBILE MODULE

LEARNING DIARY

Introduction

19.11.2021

As I have already completed the front-end and full-stack modules of this course group, I already know how the course is structured, and what is expected of me. I started off by browsing through the course documentary. After that I created a git repository for the course and published it to GitHub. I have also completed the object-oriented programming course, which had a lot of mobile development specifically on android, so I think this course will be mainly recapping what I've already learned elsewhere. But that is important as well since repetition is the key to learning!

Part 1

19.11.2021

I also started part 1 of the exercises today. I opened Android Studio and created a project. I haven't written Java nor object-oriented programming since the first year, so this will be good practice. The tutorial started slowly, explaining all the parts of creating a new project and how the project is structured inside Android Studio, these things came easily to me. Nonetheless showing how setting global strings for common sentences or words or setting global colors for the theme was nice to remember.
Next up was doing the actual task, which was simple today. Adding two integer inputs together. Using constraintlayout is intuitive. Creating the components and putting them together worked nicely by just dragging them in and dragging the required constraints for them. For this I needed 4 components, two editText components to get the addable numbers and a button to add them together and a textview to hold the results. Testing the app using an Android Virtual Device or AVD was easy as my pixel3 emulator worked right off the bat.
Next up was doing the actual code for the program. All of this program's code will happen in the MainActivity, since there is only one screen needed. First, we need to find the

components to be assigned to variables. For this we use the findViewById method and give it the parameter of the id. The id’s are stored in a class that is called with R, so R.id.editText could be a correct parameter for example. Next, we set up an onClickListener, which is a method that is called every time the component is clicked. We set this for our add button. Inside this listener we will write our program. First off we get the values as integers from the editTexts with Integer.parseint(.getText().toString()). So we just get the value as a pure string and then parse it into an integer, after that is done for both of the editTexts, we just need to add them together. This result that comes is then set into the result textView. Here we need to be careful since the result is an integer and the setText function for the textView component requires a string. The fastest way to convert the integer to a string is to just add a blank to the integer like: + ””. After that is done, the app is complete, I tested it with my virtual device, and it worked as expected. The numbers I input are correctly added together.
Next the tutorial covered the danger of using an integer for setText and debugging it with Android Studio’s breakpoints. This was nice to learn since I never fully understood how to use breakpoints and I didn’t use them much before. Now I feel confident in my debugging with breakpoints.

Part 2

20.11.2021

Today I started part 2 of the tutorial. This tutorial’s app will be a quick app launcher that can open a second activity or a different app altogether. First the tutorial covered the meanings of activities, intents, intentServices and broadcastReceivers. I have used all these extensively in the past, so I already knew them. Next, I created a new project using an empty activity. Then I changed the app-name in the strings.xml. I also changed the theme’s colors a bit in the colors.xml. Next, I added the components needed for the app, this time I only needed two buttons to launch the different activities. I also constrained them. Then I created a new activity that the other button will launch, and set a clickListener on the button to create a new intent and start the second activity using the intent. I tested the button and it correctly went to the second activity. Then I needed to send data between

activities, and this happens by putting extra variables inside the intent used to start the new activity, and then receiving it by getting the intent sent and getting the extras from that. Then I used the data sent to set the textview's text to dynamically change the view depending on the intent sent.
Next up was creating the functionality of the second button. This button's goal is to take the phone to a fun cat bouncing website in a browser. First up I need to parse the Uri address for https://cat-bounce.com. Then I created an intent using Intent.ACTION_VIEW, which starts actions outside the application, for example opening an email application when used on a mailto or dialing some number using tel. Then I'll check if any apps were found to run the action. This is done using resolveActivity() and checking the apps on the phone using getPackageManager(). If apps were foumd that can do by action, I just start the activity and the phone will navigate to the required app and go to the site. Next up I tested the app and both buttons succeeded in their goal! Part 2 of the course done.

Part 3

20.11.2021

Continuing straight from part 2 I decided to complete part 3 as well today. This time the goal is to create a list application where a list of items is shown and making them clickable to see extra info on each one. Firstly of course I create a new project with an empty activity. The tutorial uses ListView to display the list which nowadays is deprecated, and I have always used ListView before, so for the sake of learning I will use the little more complicated RecyclerView, in which I will need to create my own adapter for it. I start off by adding the RecyclerView to my main page.
I will be creating a pasta list application. I will use three arrays, one for the pasta names, one for the numbers and one for the descriptions. I hardcoded the string arrays to the strings.xml for the purposes of this exercise. After that I created a layout for a single row of the list, it's simple since it just has one TextView but it's easily expandable to hold more information or functionality inside the list. Then I created another activity to hold pasta details when a user clicks on one of the entries in the list. In the detail activity I use three textview's for which I'll send the values later as intent extras. I also have a imageView that

gets an image of the pasta depending on the index of the pasta in the list. I then scale the img with a function I defined. Inside the function I use bitmapfactory to decode the image. Then I check if the width of the image is creater than the screen width, if it is I get the ratio of the difference and set the bitmapfactory's samplesize as the ratio and set the imagebitmap with the new scaled image using 700 pixels for the width and the height. Then I was off to create my RecyclerViewAdapter. I used these sources to help me create it since I haven't used recyclerView before:
https://developer.android.com/guide/topics/ui/layout/recyclerview and
https://stackoverflow.com/questions/40584424/simple-android-recyclerview-example I extend recyclerview.adapter to get the methods I need to implement and override them. First, I will define a string list that holds the data, a LayoutInflater which inflates the "recyclerview_row" layout I created and an itemClickListener for which I will next create the interface. The itemClickListener interface will just call another function onItemClick with the position of the clicked item from the list as a parameter. This way I can always define a different functionality for the click by overriding the onItemClick function when calling the adapter. Then I'll edit the constructor to set the required variables. I needed to convert my list to a List from an string array to use it easier. Then I set the layoutinflater's context, the list and the clickListener. I then overrid the onCreateViewHolder to inflate the rows as the list item views, I also sent the mClickListener I defined in the constructor, so the items are clickable. OnBindViewHolder is another override in which I set the textview inside the row layout as the current pasta corresponding to the position of the item view. Only overrides left were getItemCount, where I just returned the size of the datalist and onClick which is inside the ViewHolder class. The viewHolder class is sent to the onBindViewHolder so it needs to define the textView of the layout row and the onClickListener. That concludes the adapter, and next I'll just use it inside my MainActivity. First, I define my recyclerview and its adapter and the string arrays as variables. Next, I set the strings and the recyclerView. RecyclerView also needs a layoutmanager so I set that and my adapter. I send the pasta name list to the adapter so it will be showed in the rows, and I set the adapter to my recyclerView. Lastly, I'll just need to override the onItemClick inside the adapter. I will want to show a detail page for the pasta if it is clicked, so I create a new intent and put the title, pastanumber description and index as extras to the intent, I get all of them using the position sent by the adapter. Then I

start the activity. I tested the application and the list shows pasta names which are clickable, and on click they go to a new activity holding detailed information and an image for that pasta, so everything works as planned!

Project work

21.12.2021

For my project I decided to do an app like the pasta application of part 3. I will create a recipe application in which the user can create new recipes and add them to their list of recipes. Recipes can also be deleted, and they can have images.

I started off by creating a new project with an empty main activity. I'll start off by creating a class for a recipe, the class has four variables: title, ingredients, instructions, and image. For displaying the recipes, I will use a recyclerView, and next I created that. I created the required adapter and row layout very similarly to part 3, this time I did no conversions and did the whole app using the Arraylist format.

In the app I'll be passing data by sending it back and forth using intents. I won't use preset lists, so users can create their own lists. Next, I'll create the activity for creating new recipes, you can go to this adding activity by pressing a button on the main activity. This activity will need three entry fields for text, a button to add images from storage and a button to submit the recipe. After the layout I'll create the java class for the activity. First, I'll create a insertRecipe function, where I'll first make a JSON object from the entries off the form. Then I'll convert the json to a Recipe object using gson and add it to the recipes list. This insertRecipe function is called when the user presses the submit button, and after that a new intent to go back to the main page is created and the recipes list is added as extra. The image adding button still needs to be handled, so when the button is clicked, a new PICK action intent will be started. I'll use the android's image media as the location to pick. I'll also create a onActivityResult to check when the user has selected the image. So, if the user had selected an image, we set the variable for the image from the returned intent.

Next, I'll create the details activity. It works similarly to part 3's detail activity. I just send the data of the title, instructions, ingredients, and image of the recipe using intent and

visualize them using AndroidStudio’s views. Then I’ll add a delete button to the recycler row so that recipes can be deleted. The button is just an imagebutton with a delete icon from the android drawables folder. The functionality will be added into the adapter. There I set the onClickListener for the delete button when the view holder is binded, and then I’ll just remove the recipe corresponding to the index of the item. Then the recyclerview needs to be notified that an item was removed so I do that with notifyDataSetChanged().
For testing purposes, I needed to add some images to the android emulator’s storage, I succeeded in this by going into the image storage on the emulator, and just dragging and dropping my images to the emulator. My biggest troubles in the project were getting the delete button to choose the right index and getting an image from the phone’s storage. Lastly, I did some style changes by changing the primary, secondary and background colors and setting elements of the app to have those colors. Then the project was done! I learned mostly about using JSON combined with android studio, getting images from the storage, and using recyclerview.