Matthew Lonis

Jeff Whitmer

CSCI-A 290

February 23, 2017

***MINI 3***

**Topic 1: Fragments/FragmentActivity**

**Topic Information**

My first topic is called fragments. Fragments are going to be used a ton in my final project. To implement a navigation drawer or a master/detail flow, where the user is navigating through pages, each page will be a fragment.

**3 URLs**

1. <https://developer.android.com/guide/components/fragments.html>
2. <https://developer.android.com/training/basics/fragments/index.html>
3. <https://developer.android.com/training/basics/fragments/creating.html>

**Precis of Sources**

Fragments are in many ways “sub activities” that exist inside of other activities. These fragments are activities that correspond to their own layouts, input events, and activities, much like a regular activity. The difference is these can be instanced whenever or thrown away whenever allowing them to be reused or take up only certain parts of the screen. A fragment has a lifecycle just like a regular activity and is even defined in similar ways to a regular activity class. Fragments can be found in the android.support.v4.app.Fragment import.

Fragments can be organized using a FragmentManager which will have its own stack to hold fragments that are added or removed in the main activity of an app. With regards to my final project, I will need to use this to implement a working back button to call to the last fragment added to the fragment manager stack. Fragments allow for a developer to make a dynamic user interface. Fragments can be configured in so many ways there is nothing left but a developer’s imagination to get what they want done.

**Summary of Topic**

Fragments are little activities that can be called into a main activity. They can have their own interface, set by content xml, and even have their own methods and take input from the user. In the case of my app, one of the first fragments the user will encounter is my home page fragment. This fragment is an example of how a fragment can demonstrate a robust layout. My home page fragment includes a scroll view, linear layout, 4 text views, and an image view, all contained inside a mere fragment! Whenever a user moves to a different page a new fragment takes its place. The fragments can encompass the space defined in my main content file. I used a frame layout for this effect.

**JIT 3**

To build dynamic UI’s, implement navigation drawer activity, or master/detail flow activity, one must understand fragments.

**Topic 2: Navigation Drawer Activity**

**Topic Information**

My second topic is called navigation drawer activity. This is the primary layout for my final project app. It is essentially a slide out menu with options so the user has a clean and neat menu to traverse the app.

**3 URLs**

1. <https://developer.android.com/training/implementing-navigation/nav-drawer.html>
2. <http://stackoverflow.com/questions/32944798/switch-between-fragments-with-onnavigationitemselected-in-new-navigation-drawer>
3. <http://stackoverflow.com/questions/20448323/how-to-get-items-in-navigation-drawer-to-change-view>

**Precis of Sources**

To further define what a navigation drawer is I refer to Android Developer:

“The navigation drawer is a panel that displays the app’s main navigation options on the left edge of the screen. It is hidden most of the time, but is revealed when the user swipes a finger from the left edge of the screen or, while at the top level of the app, the user touches the app icon in the action bar.”

The key aspects of a navigation drawer to remember to program are to create the drawer layout, initialize the drawer list on creation, handle navigation click events, and listen for open and close events. Now, if one simply uses the template for navigation drawer, the developer needs to only worry about navigation click events and customizing the drawer layout.

The most important aspect of the navigation drawer is handling navigation click events. In this case, switching between fragments. In order to do this the developer needs to have a fragment class they want to switch to. Then, the developer needs to get a FragmentTransaction using .getSupportFragmentManager().beginTransaction(); This code will be stored in a variable, let’s call it ft for the rest of this precis. After the developer gets ft, they need to invoke the .replace() method which takes a content frame and a fragment. The content frame is usually defined in the main content of the app as a frame layout. Here’s an example: ft.replace(R.id.content\_frame, fragment); After that the developer needs to call ft.commit(); (sort of like GitHub) and they are done. The hardest part here is making sure your Fragments are correctly implemented.

**Summary of Topic**

The navigation drawer is my personal favorite means of navigating within an app. The most obvious thing to prioritize is establishing working code to switch fragments on navigation item clicks. The tricky part here is there are at least 3 fragment types that I know of and each have their corresponding FragmentManager and FragmentTransaction. For example, there is Fragment in android.support.v4.app.Fragment, There is a fragment in android.support.v7.appcompatActiviy (which is a subclass of Fragment), and then there is android.app.Fragment. So, making sure the developer is consistent is the most important thing to do here. I find working with android.support.v4.app.Fragment to be the easiest and most reliable (plus it targets low APIs as well).

**JIT 3**

Navigation Drawers are one of the best means on navigation, but require a robust understanding of Fragments, FragmentManager, and FragmentTransaction just to even get the basic functions to operate.

**Topic 3: Master/Detail Flow Activity**

**Topic Information**

My final topic is called master/detail flow. This is essentially a table view, but when you click on the items in the table they open a “detail” page for the user to get more information about the item they selected.

**3 URLs**

1. <https://developer.android.com/training/implementing-navigation/descendant.html>
2. <https://developer.android.com/studio/projects/templates.html>
3. <http://www.techotopia.com/index.php/An_Android_Master/Detail_Flow_Tutorial>

**Precis of Sources**

A master/detail flow consists of two screens: a master screen and a detail screen. The master screen is a list of items that belong to a collection. The detail screen shows the detailed information about an item that was selected on the master screen. Android Developers calls this a form of descendant navigation. Android developers describe the problem of multiple screen sizes about master/detail flows. Basically, for handhelds, instead of using fragments and transactions, activities will be started using intents. On a tablet however, both the master and the detail page can be on the screen at the same time so the detail page is changed using fragments. Luckily for the purposes of my project, I will only be addressing handheld devices.

Luckily for modern developers, android studio comes with templates for the master/detail flow. To do this, the developer needs to right click on the folder they want to add the component in, select new, and scroll all the way down to the bottom. There the developer will find some activity names with green little androids next to them. To get to the activity templates, hover your mouse over the one that says “Activity” and you’ll see a big list of default activities that are shown when you are first creating a new android project.

**Summary of Topic**

Master/Detail Flows are great for grouping a collection of items that are all related and providing easy navigation to view more detail about each item. Some common implementations of this that we use every day are our text messaging apps. Whether you use iPhone or Android, the text message app uses a master detail flow to show a collection of conversations, but when the user clicks on one they are taken to the “detail” page of that conversation to view more. This same feature is also implemented in many email apps. I plan on using this to group all of my courses in a master page and for each have a corresponding detail page including course info and projects I worked on in that course.

**JIT 3**

Master/Detail Flows are a great way to organize a collection of similar items and allow the user to view more information about them with clean navigation. Just remember to account for what screen size your application is going to be deployed on.