Lab - Fragments

Objectives:

Familiarize yourself with the Fragment class. Create a simple application that uses Fragments to produce a two-pane or single-pane user interface depending on the current device's screen size.

After completing this Lab you should better understand the Fragment class and its lifecycle, and how Fragments interact with the Activities that host them.

Exercise: Fragments

In this exercise you will create an application that uses Fragments to display simulated[1]Twitter Tweets. This application will present multiple Fragments arranged in different layouts depending on the device's screen size. One of these Fragments, called the FriendsFragment, will display the names of several celebrities. If the user selects a name from this Fragment, he or she will see several simulated Twitter Tweets from these people, appearing in a second Fragment, called the FeedFragment. For this Lab, the number and identity of the celebrities will be fixed (once you learn more about User Interface classes, Networking, and Services, you may want to extend this application into a more general Twitter feed reader).

The graphics below depict the application's user interface when running on a typical small screen device (e.g., a smartphone). This layout will be called the single-pane layout:
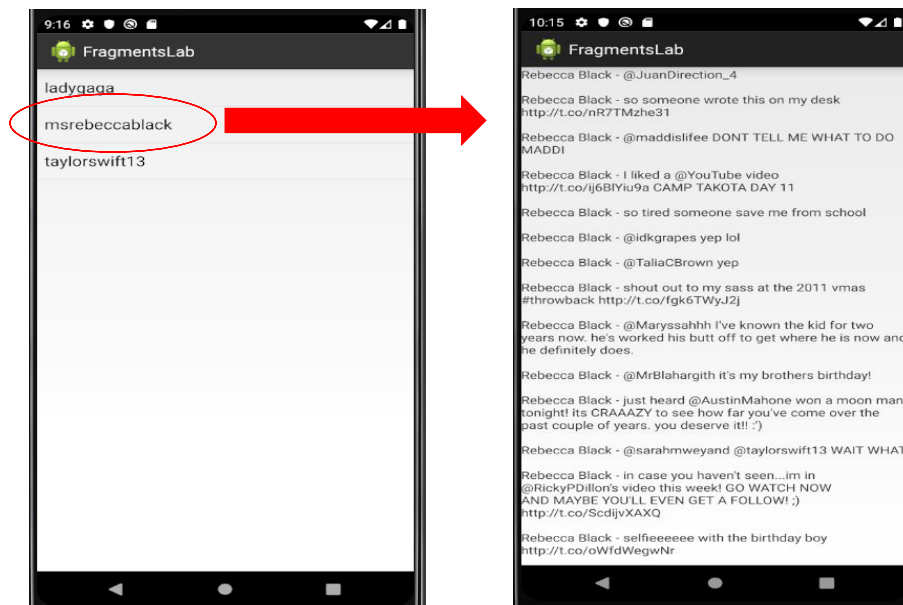

Figure 1: When user clicks on a celebrity's name, they see Twitter tweets from that celebrity.

---

[1] The Twitter API requires a network connection and authentication in order for your application to retrieve Twitter data. Because we have not yet discussed networking, this application will only simulate a live Twitter feed.

To implement this user interface, you will implement two Fragments; one called FriendsFragment and the other called FeedFragment. The FriendsFragment, displayed on the left of the figure above, is a subclass of ListFragment. If the user selects a celebrity name from this ListFragment, then the Tweets from that person will be displayed in the FeedFragment. The FeedFragment displays a single TextView, wrapped in a ScrollView, containing all the tweets for a single celebrity. If the user hits the back button when the FeedFragment is visible, the application should return to the previous View in which the FriendsFragment was visible.

When running on a larger-screen tablet, however, the application will present a different user interface, as shown below. This layout is called the two-pane layout.
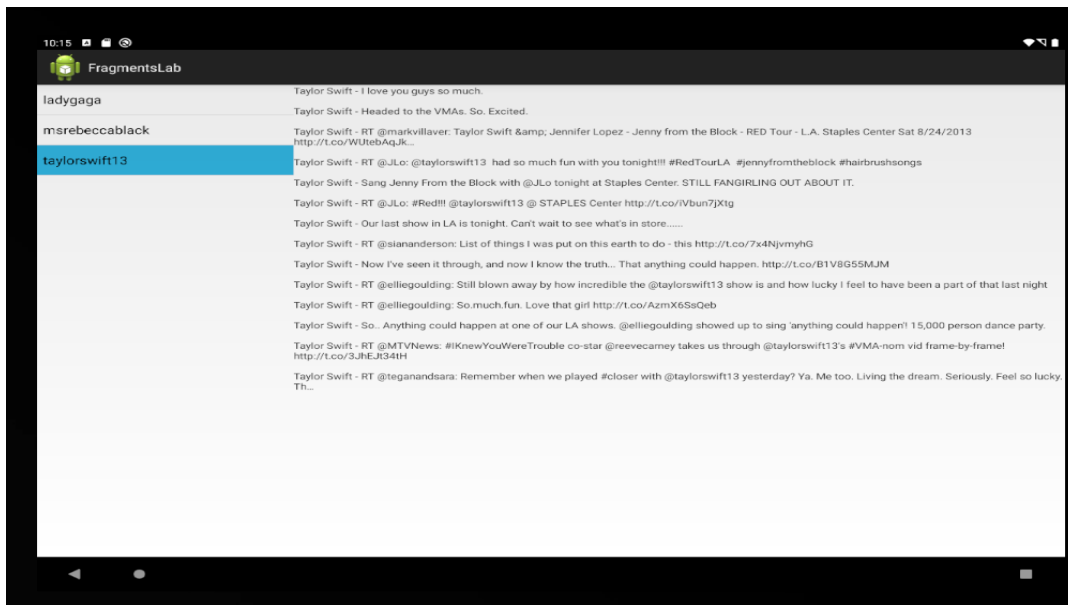

Figure 2: A two-pane user interface used when the device is a Tablet.

In this case, the application displays both Fragments at the same time. You will implement the code that manages the application layouts. **Note**: You are not going to create two different applications. You are going to create a single application that works whether the particular device it's running on is small or large. Look at the documentation (http://developer.android.com/guide/practices/tablets-and-handsets.html) for more information about multi-pane user interfaces here.

See the FragmentsLabPhone.mp4 screencast to see the app in action on a phone. See the FragmentsLabTablet.mp4 screencast to see the app in action on a tablet.

Implementation Notes:

This exercise is quite similar to the applications discussed in the lecture (as always, download and examine the source code for the example applications from the class source code repository). We have provided you with an application skeleton, including all necessary layout and resource files. Don't change any resource IDs in these files as that might break the Lab's test cases.

You will need to modify two areas within the Lab's source code. Both of these areas are marked with a comment containing the word "TODO." Both are in the MainActivity.kt file. To do this, you will need

to examine and understand the resource IDs contained in the main_activity.xml files that are in res/layout and res/layout-large directories.

1. In MainActivity.kt, find "TODO 1." Add the source code needed to add the FriendsFragment to the two-pane layout.
2. In MainActivity.kt, find "TODO 2." Add the source code to replace the FeedFragment displayed in the single-pane layout.

Testing and Submission:

The test cases for this Lab are inside the tests directory under course (androidTest) -> labs -> fragmentslab. TabletTest.kt should be run on a Tablet (we used a Pixel C with an API 29). You can run the test cases by right clicking on the tests folder and selecting "Run Tests in …" or individually by right clicking on an individual test case class and then continuing as before. The test classes are Robotium test cases.

To submit your work you will need to commit your solution to your repo on GitLab by running the following command: git push origin master. Always verify by navigating to your repo on gitlab that your code changes are there.
Note: you must commit your changes locally before you can push them to your remote origin repo.

Warnings:

1. These test cases have been run on the emulator using a Galaxy Pixel 3 AVD with API level 29 with 1536 MB of RAM (for the phone test) and a Pixel C at API level 29 with 1536 MB of RAM (for the tablet test). To limit configuration problems, you should test your app against similar AVDs.

Extras:

This assignment may be too easy for those with strong programming backgrounds. If you fall into that category, here are some suggestions for more challenging enhancements you can make to the application.

1. Add the necessary code so that the application maintains its state (which Friend and/or Feed is selected) after a reconfiguration.

2. Instrument the application to monitor the lifecycle callbacks for both the Activity and the Fragment classes.

3. Take a deeper look at the issue of supporting multiple screens. We handled this by defining two different main_activity.xml files, one for large devices and one for smaller devices. Read through the documentation (http://developer.android.com/guide/practices/screens_support.html) on Supporting Multiple Screens. After reading this over, go back and look at your application. Could you come up with a way to have layouts that are different for small devices in landscape mode, small devices in portrait mode, and a large device (two-pane layout)?