CSE 437 Lab Assignment 6: Fragments and Broadcast Receivers

Part 1: **Fragments**

Introduction

A Fragment represents a behavior or a portion of user interface in a FragmentActivity. You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities. You can think of a fragment as a modular section of an activity, which has its own lifecycle, receives its own input events, and which you can add or remove while the activity is running (sort of like a "sub activity" that you can reuse in different activities).

Design Philosophy:

Android introduced fragments in Android 3.0 (API level 11), primarily to support more dynamic and flexible UI designs on large screens, such as tablets. Because a tablet's screen is much larger than that of a handset, there's more room to combine and interchange UI components. Fragments allow such designs without the need for you to manage complex changes to the view hierarchy. By dividing the layout of an activity into fragments, you become able to modify the activity's appearance at runtime and preserve those changes in a back stack that's managed by the activity. They are now widely available through the fragment support library.

For example, a news application can use one fragment to show a list of articles on the left and another fragment to display an article on the right—both fragments appear in one activity, side by side, and each fragment has its own set of lifecycle callback methods and handle their own user input events. Thus, instead of using one activity to select an article and another activity to read the article, the user can select an article and read it all within the same activity, as illustrated in the tablet layout in figure 1.

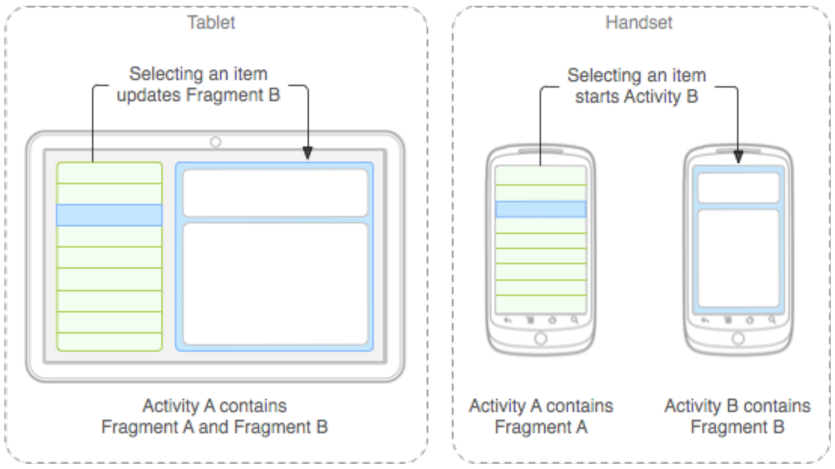

Tablet
Selecting an item updates Fragment B
Activity A contains Fragment A and Fragment B
Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B

Figure 1. An example of how two UI modules defined by fragments can be combined into one activity for a tablet design, but separated for a handset design.

You should design each fragment as a modular and reusable activity component. That is, because each fragment defines its own layout and its own behavior with its own lifecycle callbacks, you can include one fragment in multiple activities, so you should design for reuse and avoid directly manipulating one fragment from another fragment. This is especially important because a modular fragment allows you to change your fragment combinations for different screen sizes. When designing your application to support both tablets and handsets, you can reuse your fragments in different layout configurations to optimize the user experience based on the available screen space. For example, on a handset, it might be necessary to separate fragments to provide a single-pane UI when more than one cannot fit within the same activity.

Part 2: **Broadcast Receivers**

Android apps can send or receive broadcast messages from the Android system and other Android apps, similar to the publish-subscribe design pattern. These broadcasts are sent when an event of interest occurs. For example, the Android system sends broadcasts when various system events occur, such as when the system boots up or the device starts charging. Apps can also send custom broadcasts, for example, to notify other apps of something that they might be interested in.

Apps can register to receive specific broadcasts. When a broadcast is sent, the system automatically routes broadcasts to apps that have subscribed to receive that particular type of broadcast.

The system automatically sends broadcasts when various system events occur, such as when the system switches in and out of airplane mode. System broadcasts are sent to all apps that are subscribed to receive the event.

The broadcast message itself is wrapped in an Intent object whose action string identifies the event that occurred (for example android.intent.action.AIRPLANE_MODE). The intent may also include additional information bundled into its extra field. For example, the airplane mode intent includes a boolean extra that indicates whether or not Airplane Mode is on.

For a complete list of system broadcast actions, see the **BROADCAST_ACTIONS.TXT** file in the Android SDK. Each broadcast action has a constant field associated with it. For example, the value of the constant ACTION_AIRPLANE_MODE_CHANGED is android.intent.action.AIRPLANE_MODE. Documentation for each broadcast action is available in its associated constant field.

<u>Example: List of actions for Android-23</u>

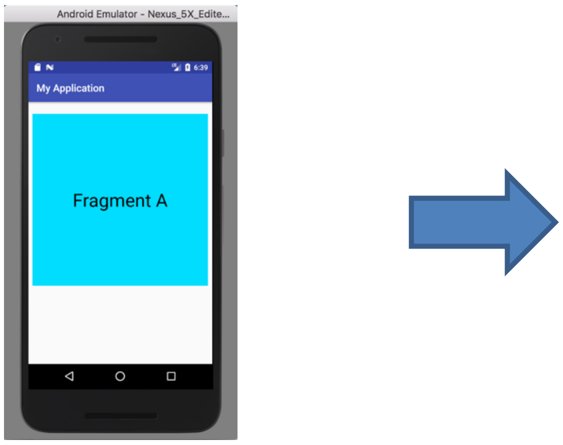
https://chromium.googlesource.com/android_tools/+/febed84a3a3cb7c2cb80d580d79c31e22e9643a5/sdk/platforms/android-23/data/broadcast_actions.txt

Instructions:

In this lab assignment, you'll use ***ACTION_TIME_TICK (****ACTION_TIME_TICK is a broadcast action that indicates that the current time has changed). This action is sent every minute by the system. Please be aware that you cannot receive this action through components declared in manifests, only by explicitly registering for it with Context.registerReceiver(). The constant value that should be use is: "android.intent.action.TIME_TICK"*

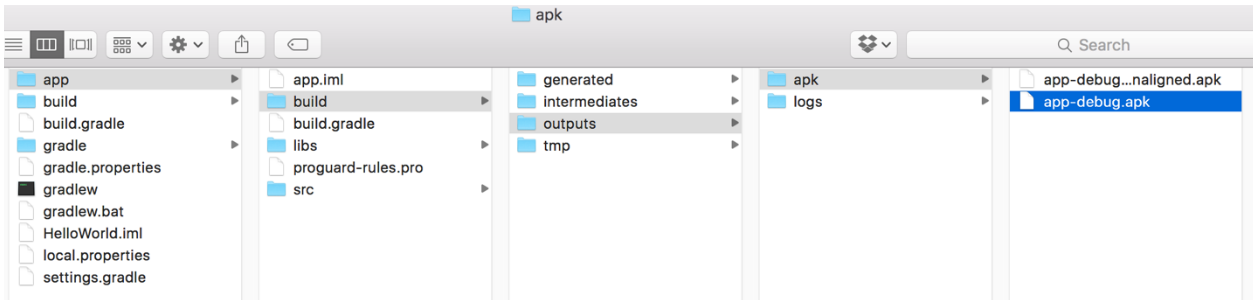
Develop an application that changes the layout of its main screen every minute as follows:

1. The application has one activity and two fragments – fragment A and fragment B
2. You are free to choose any layout design for the activity and fragments
3. *The layout of the main activity starts with Fragment A layout on the foreground*
4. *Every minute, the main activity replaces the current fragment with the second fragment through a fragment transaction. (the app should alternate between the fragments upon receiving the broadcast message)*
5. *The application uses a broadcast receiver to listen to ACTION_TIME_TICK*


Android Emulator - Nexus_5X_Edite...
6:39
My Application
Fragment A
Android Emulator - Nexus_5X_Edite...
6:40
My Application
Fragment B

Submission Instructions

1. Open a "Windows Explorer" window (Finder on Mac) and navigate to the project source code location



2. Copy the app-debug.apk file to your desktop. Rename the file to <lastname_firstname_lab06.apk> and send it to **pete.ragheb@gmail.com with the subject line: CSE437_Lab6**

3. Print a hard copy of the activity_main.xml files and the java file(s). Write your name and class number on the paper and submit it to the lab instructor.

4. Demonstrate your app on a real device and show it to the lab instructor during the lab time for marking and feedback.

5. Keep a copy of the project workspace for your record. The lab instructor might ask you to submit it during the marking process.