CMSC436: Fall 2013 – Week 4 Lab

Objectives:

Familiarize yourself with Android Permission and with the Fragment class. Create simple applications using different Permissions and Fragments.

Once you've completed this lab you should understand: Android Permissions, how to define and enforce Permissions for your own apps as well as use Permissions from other apps, and the Fragment class and its lifecycle. You also should know how to interact with Fragments hosted in an Activity.

Overview:

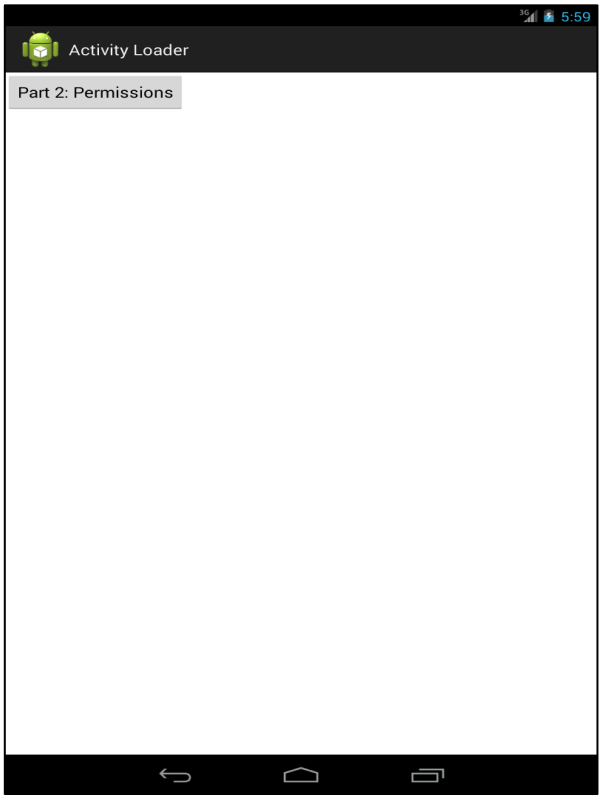
This lab has two parts: one that focuses on Permissions and one that focuses on Fragments.

Part 1: Permissions

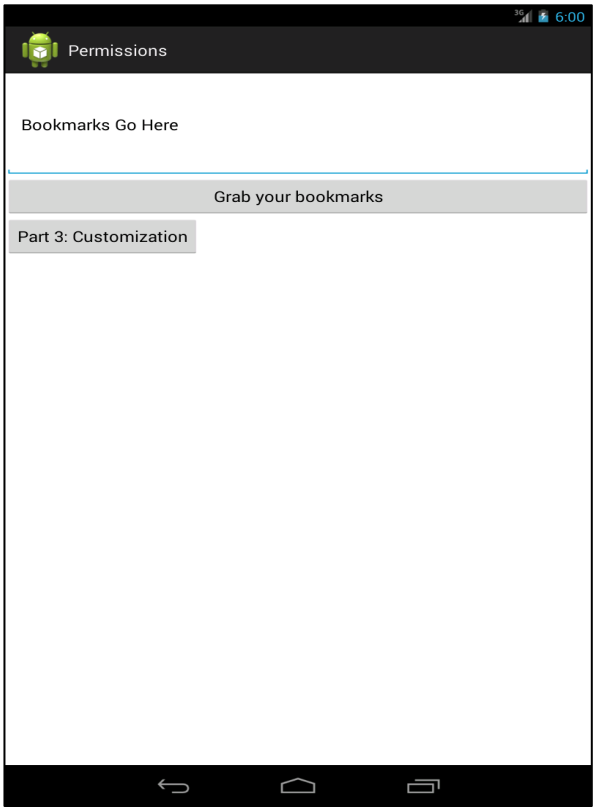
Exercise A: Permissions

This exercise will focus on using Permissions when loading protected content.

The main Activity, called "Activity Loader" displays a button labeled "Part2: Permissions".



When the user clicks this Button, the application should start a new Activity called "Permissions"

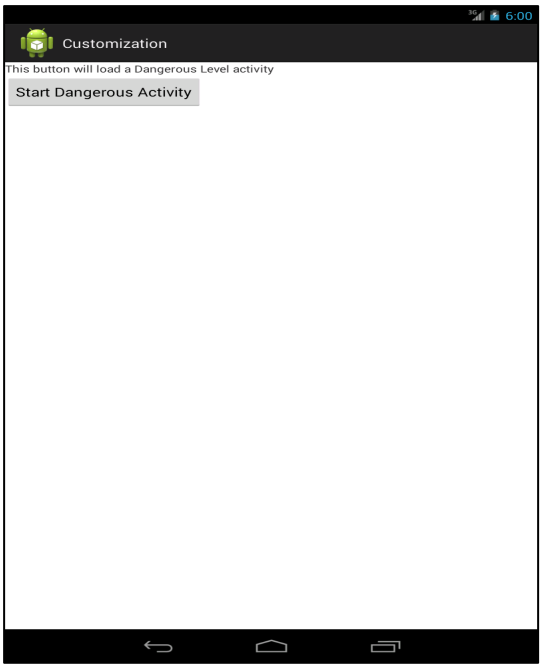


This activity presents a Button labeled, “Grab your bookmarks.” When the user presses this Button, the application should retrieve all of your Browser bookmarks and should display them in the TextView (shown above) that currently displays the words “Bookmarks Go Here”. The Browser Bookmarks are stored in a ContentProvider. We haven’t discussed ContentProviders in detail yet, so the application skeleton includes the code needed to query this ContentProvider.

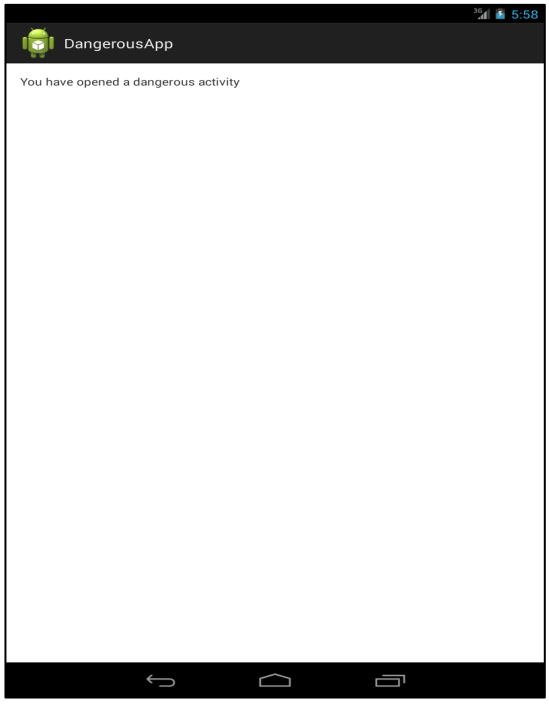
Exercise B: Custom Permissions

This part of the lab will focus on using permissions to access another one of your apps**.**

When you click on the button called “Part3: Customization” from Exercise A, another activity called “Customization” with the user interface as below should occur:



Clicking on “Start Dangerous Activity” should open a new application called “DangerousApp” which simply says “You have opened a dangerous activity”.



The DangerousApp will require you to create a new Android application project. In it, you will create a new permission called “DANGEROUS_ACTIVITY”, a dangerous level permission. You will need to create an intent filter that accepts an implicit intent for the permission.

Implementation Notes:

1. Download the application skeleton files from the following URL and import it into your IDE.

2. For exercise 1, implement the following classes and methods

In ActivityLoader.java.

   a. Implement the public void startPermissionsActivity (View view) method.
      // This function launches “Permissions” activity.

In Permissions.java.

   b. Implement the public void startCustomizationActivity(View view) method.
      // This function launches “Customization” activity.

In AndroidManifest.xml

   c. You need to add the appropriate permission for accessing the Browser bookmarks. See http://developer.android.com/reference/android/Manifest.permission.html for more information.
   d. You need to add the appropriate permission for accessing the dangerous app.

3. For Exercise 2, implement the following classes and methods

In Customization.java.

   a. Implement the public void run(View view) method.
      // This function launches “Dangerous” activity from “DangerousApp” application.

Create the “DangerousApp” application (you are not provided with this app’s skeleton) as described above.

   b. This application should define and enforce a new permission called “DANGEROUS_ACTIVITY”, a dangerous level permission.
   c. You will also need to create an Intent Filter so that the main Activity of this application accepts an implicit Intent.

Deliverables:

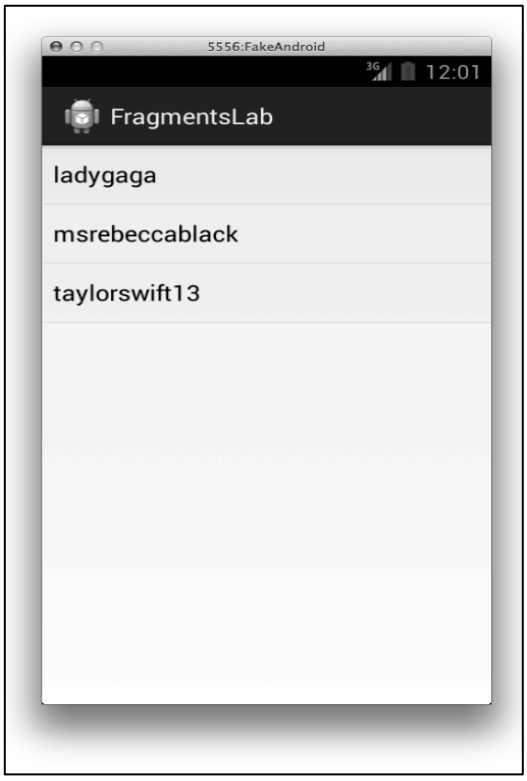
Submit your project via the submit server. Your submission should include:

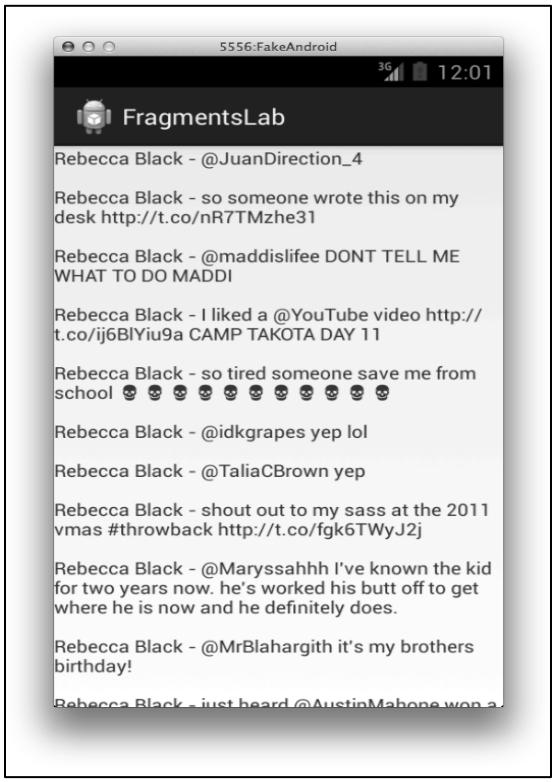
a. A zip file containing the source code project that implements both Exercise A and Exercise B.

Part2: Fragment

The purpose of this activity is to familiarize yourself with fragments and interacting with fragments through an Activity. This application will present multiple Fragments in different layouts depending on the device's screen size. These Fragments will display the names of several celebrities and will allow the user to see several Twitter tweets from these people. For this lab, the number and identity of the celebrities will be fixed.

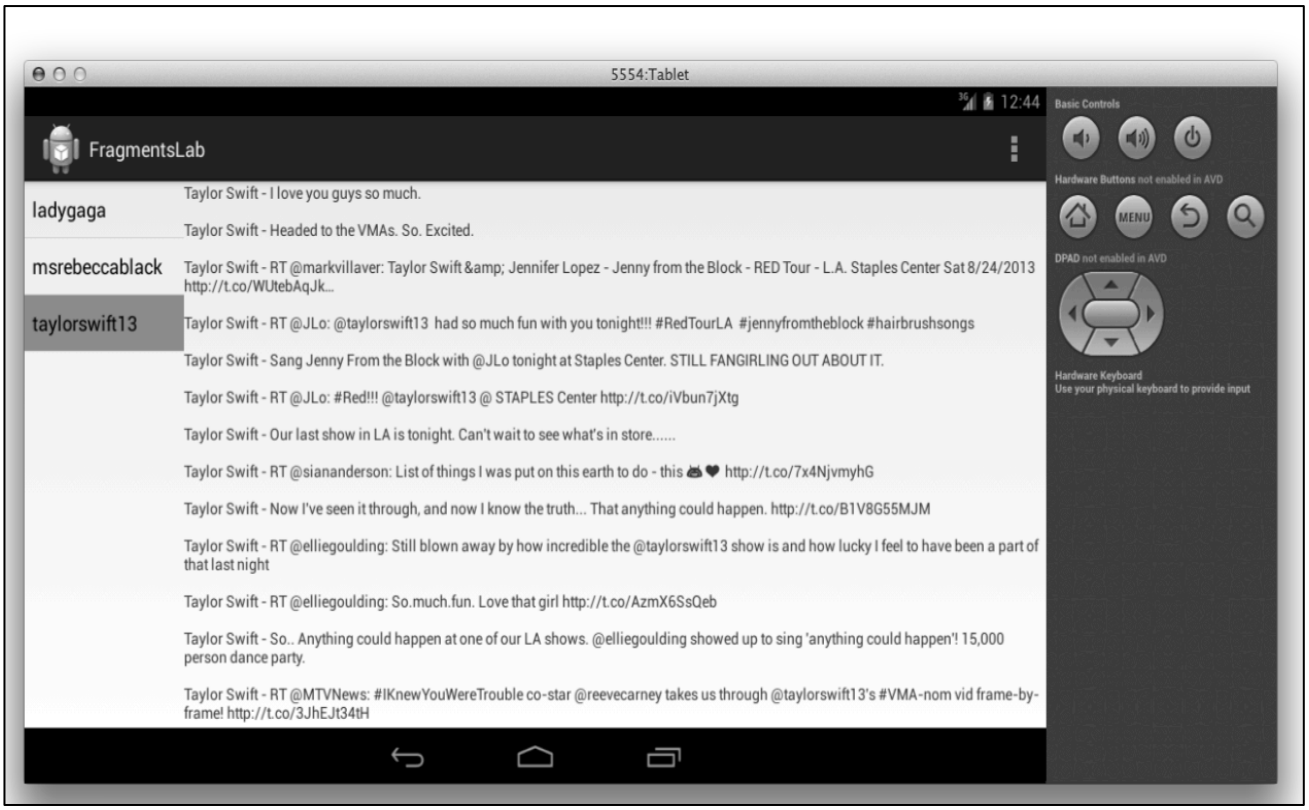
Below is the user interface of this Part's application when running on a phone:





You will be creating two fragments. The one on the left is a List Fragment. When the user selects a name from this list, the tweets by this person will be displayed in a second Fragment.

When running on a larger screen tablet, the application will present a different user interface:



When running on a tablet, the application will display both fragments at the same time. You will implement the application layouts and any necessary interaction between the Fragments and Activity. You can find more information here:

http://developer.android.com/training/basics/fragments/index.html

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 using a live Twitter feed. That is, we are storing tweet data on your device and accessing it from there.

Implementation Notes:

This exercise is very similar to the application presented in the lecture video (you can download the source code of that application from the class repository). In addition, there are many different ways to implement this application. Therefore, in this part you will have considerable flexibility to design your code and layouts. However, we still provide you with a skeleton of the application with its necessary layout and resource files.

In addition the skeleton has two methods for getting and updating the Twitter feeds.

1. **private** JSONArray getFeed(**int** id)

This function receives an id representing the person whose Twitter feed you want to get. It returns his or her feed.

For example, to get feed of *ladygaga,* you should call:

getFeed(**IDS[0]**)

where IDS is an integer array of interested people. It is declared in the provided code as follows:

int[] IDS = {R.raw.ladygaga, R.raw.rebeccablack, R.raw.taylorswift};

2. **private** String procFeed(JSONArray feed)

This function processes the ouput from the getFeed() method and returns a string representing the feed. As an example, you can get the string formatted Twitter feed of *ladygaga*, by calling:

procFeed(getFeed(IDS[0])

Again, we will discuss the inner workings of these files when we get to Networking.

Deliverables: - your source code project

Extras:

For those who are interested, read more about accessing Twitter data through the Twitter API here:

https://dev.twitter.com/docs/api/1/get/statuses/user_timeline