Notifications Lab

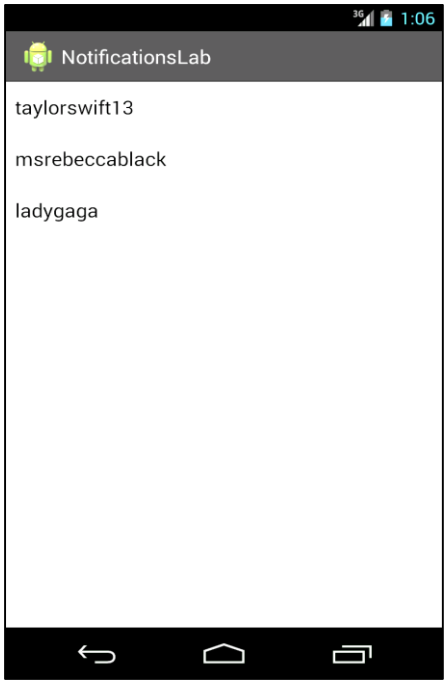
Notifications and BroadcastReceivers

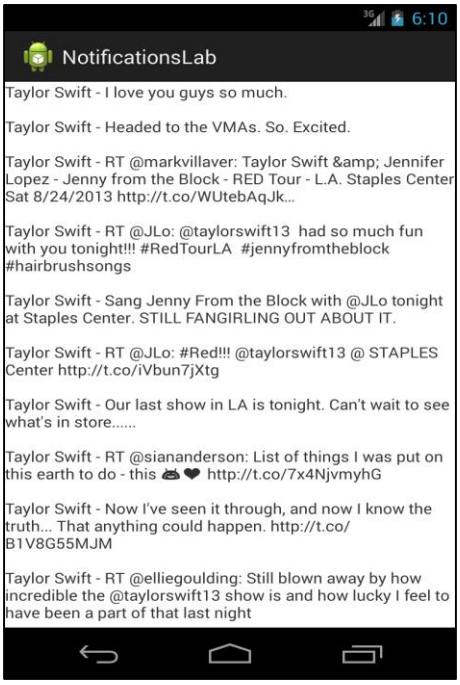
Objectives:

This week's lab aims to give you a better understanding of User Notifications and Broadcast Receivers. Upon completion of this lab, you should have a better understanding of how to create and display different types of User Notifications so that you can inform a user of an application's actions. You will also learn how to broadcast and receive Intents.

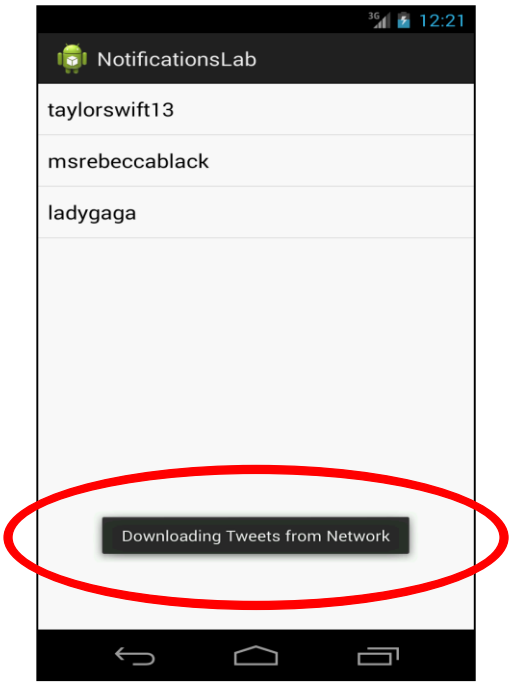
Exercise A:

This lab is a modified version of the Fragments Lab in which you displayed locally stored Twitter data. Thus, its interfaces and code are quite similar to those of the previous lab. For grading purposes, we will only be testing the phone version (not the tablet version).





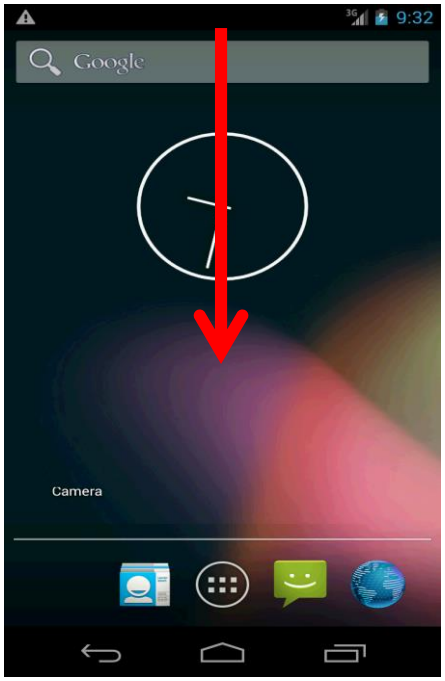
When the application's MainActivity begins running, it will determine whether any Tweet data has been downloaded within the last two minutes. If not, it will pretend to download Twitter data. This "downloading" step will retrieve fixed data from raw data files included in the project. To do this, the application will construct and execute an AsyncTask, which will load the Twitter data off of the main Thread. As the AsyncTask begins to download the Twitter data, the application should create a Toast message to alert users that the download is starting. An example is shown below:

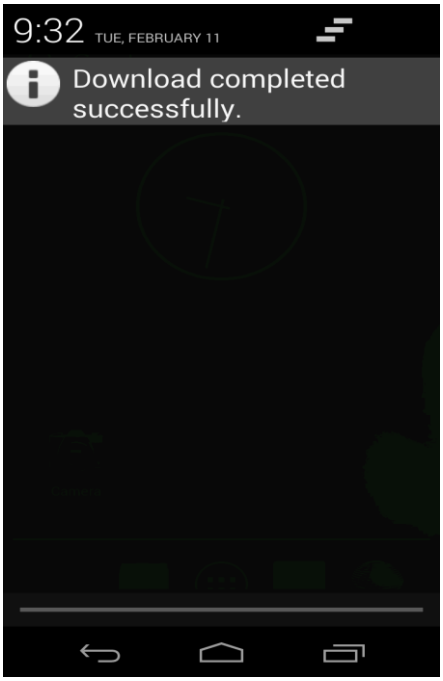


The download process will take some time, during which the user might even exit the application. If that happens, the application still should be ready to inform the user that the data has been downloaded. To do this, when the AsyncTask finishes downloading the tweet data, it will broadcast an Intent. If the application's MainActivity is not running and in the foreground when this Intent is broadcast, then the AsyncTask will create a Notification Area Notification and place it in the Notification area. However, if the MainActivity is running and in the foreground, then the AsyncTask will not create this Notification.

To do this, the AsyncTask will use the sendOrderedBroadcast() method to broadcast an Intent once downloading has finished. It will need to pass its own BroadcastReceiver into this call so that it can receive the result of the broadcast. If a BroadcastReceiver in the MainActivity receives this Intent, then it will return a specific result code. This way the AsyncTask knows that the MainActivity is in the foreground. If this result code does not arrive back to the AsyncTask, then the AsyncTask will assume that the Activity is not active and therefore it will send the User Notification.

If the AsyncTask does send the Notification, an icon will appear in the Notification Area and the user will need to examine it. When the user pulls down on the Notification drawer, he or she will see an indication of whether or not the data was successfully downloaded. An example is shown below:





If the user clicks on this Notification's View, the MainActivity should re-open. Again, if the MainActivity starts more than two minutes after the data was downloaded, then MainActivity should download the data again. Otherwise, it should not.

The following screencast shows the app in operation.

Implementation Notes:

1. Download the application skeleton files and import them into your IDE.
2. Implement the following classes and methods
   In MainActivity.java.
   a. Modify the **private void ensureData()** method. Create a BroadcastReceiver that returns a result code (MainActivity.IS_ALIVE) which will inform the AsyncTask that the MainActivity's active and in the foreground, and therefore, the AsyncTask should not send the user notification.
   b. Register the broadcast receiver in the **protected void onResume()** method.
   c. Unregister the broadcast receiver in the **protected void onPause()** method.

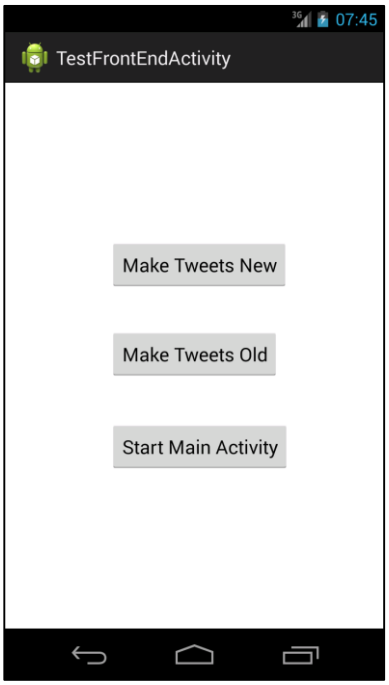
   In DownloadTask.java.
   d. Modify the **private void notify(boolean success)** method to notify that the feed has been loaded using sendOrderedBroadcast(). You will need to create a BroadcastReceiver to receive the result of this broadcast. If that result is not MainActivity.IS_ALIVE then this BroadcastReceiver should create a Notification Area Notification. More information about the sendOrderedBroadcast() method can be found at: http://developer.android.com/reference/android/content/Context.html

Testing:

The test cases for this Lab are in the NotificationLabTest project, which is located in the NotificationsLab/SourceFiles/TestCases directory. You can run the test cases either all at once, by right clicking the project folder and then selecting Run As>Android Junit Test, or one at a time, by right clicking on an individual test case class (e.g., X.java) and then continuing as before. The test classes are Robotium test cases. As you implement various steps of the Lab, run the test cases every so often to see if you are making progress toward completion of the Lab.

Warnings:

1. These test cases have been tested on a Galaxy Nexus AVD emulator with API level 18. To limit configuration problems, you should test you app against a similar AVD.
2. These test cases will start the application at an Activity called, TestFrontEndActivity. This allows us to modify the age of any already downloaded Tweet data before starting the MainActivity. The interface for this activity is shown below.



Once you've passed all the test cases, submit your project to Coursera.

Submission

To submit your work you will need to submit the NotificationsLab project files we've asked you to modify. These files should be stored in specific directories as described below and then compressed in a zip file. Then you will submit this zip file to the Coursera system. The automatic grading system will test your submission and give you feedback. This process may take some time, especially if many students are submitting at the same time. To make sure your submission is correctly graded, pay attention to the following aspects:

1. Your project files must be compressed in a zip file named NotificationsLab.zip.

2. When decompressed, your submission should contain one top-level directory named NotificationsLabSubmit. Inside that directory there should be one directory named NotificationsLab containing the following files: MainActivity.java and DownloadTask.java.
3. Each new submission will overwrite any previous submissions.

If you get through Exercise A and submitted and passed the tests, and feel that you'd like to do more, here are some suggested additions. This is optional and ungraded.

Optional Exercise B: Add Alarms

Modify your application so that your Tweet data is always fresh. Use an Alarm to download the Tweet data every two minutes. **Note:** In a real application, downloading every two minutes would probably be excessive. You can play with the download frequency. In addition, since our data will never change, the whole operation is somewhat pointless, but it give you a chance to create Alarms.