Notifications Lab

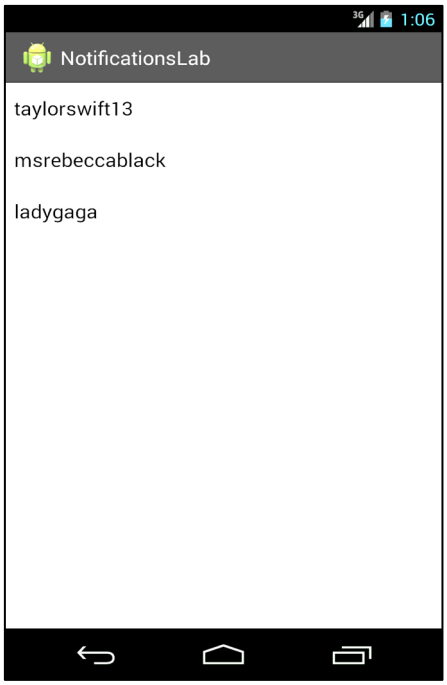
Notifications, BroadcastReceivers, AsyncTasks and Networking

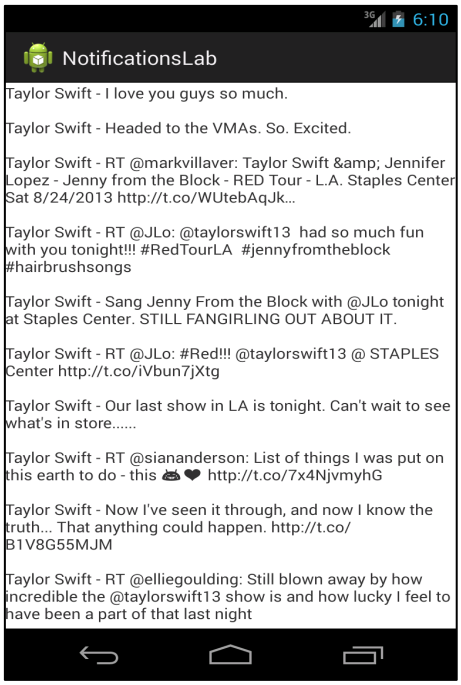
Objectives:

This week's lab is aimed at giving you a better understanding of User Notifications, AsyncTasks, Broadcast Receivers, and Networking. 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 should also gain familiarity with AsyncTasks and how they are used to perform slower operations, like downloading data from the Internet, in a non-UI Thread. Finally, you will learn how to broadcast and receive Intents.

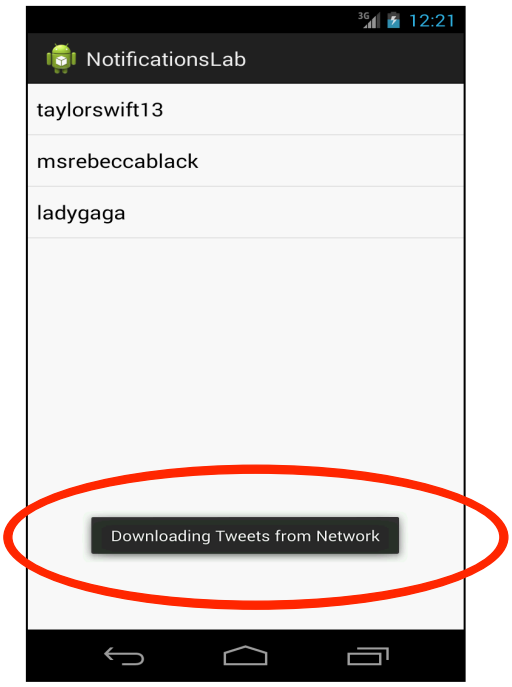
Exercise A:

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).





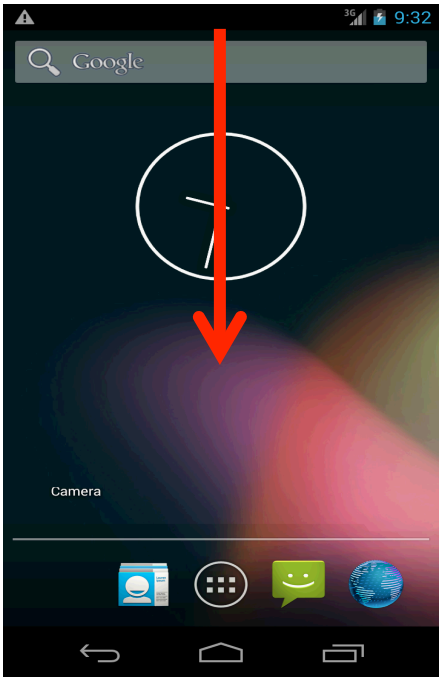
When the application's MainActivity begins running, it will determine whether any Tweet data has been downloaded within the last two minutes. If not, it will download Twitter data from the network. This “downloading” step will retrieve fixed data files from a Coursera server, not a live stream from a Twitter server. To do this, the application will construct and execute an AsyncTask, which will download 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:

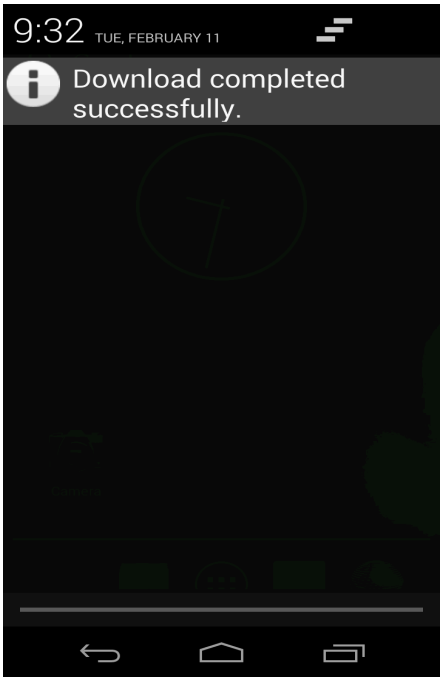


The download process can take a while, during which time the user might even exit the application. If that happens, the application still should be ready to inform the user that the data has been downloaded. To do this, when the AsyncTask finishes downloading the tweet data, it will broadcast an Intent. 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.

Alternative for Students without a Stable Network Connection

If you do not have a stable network connection or have a slow network connection, you can simply read the data from a local file. To do this, open the DownloaderTask.java file. In there you will find a boolean variable named HAS_INTERNET_CONNECTION. Change the value of this variable to "false."

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 (RESULT_OK) 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.

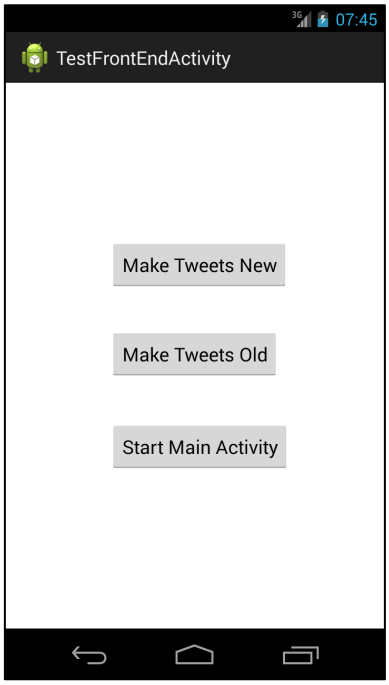
   In DownloadTask.java.
   d. Modify the **private void notify(boolean success)** method to notify that the feed has been loaded using sendOrderedBroadcast(). You will need to create a BroadcastReceiver to receive the result of this broadcast. If that result is not RESULT_OK, this BroadcastRecevier should 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/NotificationsLabTest.zip file. 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. You will eventually have to run each test case, one at a time, capture log output, and submit the output to Coursera. These test cases are designed to drive your app through a set of steps, passing the test case is not a guarantee that your submission will be accepted. The NewFeedTest test case should output exactly 1 Log message. The OldFeedNoNotificationTest test case should output exactly 7 Log messages. The OldFeedWithNotificationTest test case should output exactly 7 Log messages. 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:

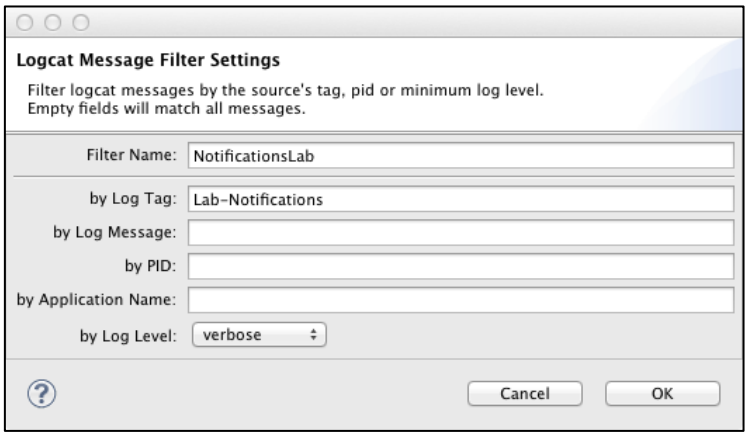
1. These test cases have been tested on a Galaxy Nexus AVD emulator with API level 18. To limit configuration problems, you should test you app against a similar AVD.
2. These test cases will start the application at an Activity called, TestFrontEndActivity. This allows us to modify the age of any already 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 log information to the Coursera system for grading.

Tips: Saving a LogCat filter.

1. In the LogCat View, press the green "**+**" sign to "add a new LogCat filter."
2. A dialog box will pop up. Type in the information shown in the screenshot below.
3. A saved filter called, "NotificationsLab" will appear in the LogCat View.



Tips: Running your test cases and capturing your LogCat output for submission.

1. For each test case, clear the LogCat console by clicking on the "Clear Log" Button (the Button with the red x in the upper right of the LogCat View).
2. Then right click on an individual test case file in the Project Explorer. Run As -> Android JUnit Test.
3. When the test case finishes, if it's not already selected, click on the " NotificationsLab" filter in the left hand pane of the LogCat View.
4. Select everything within the LogCat View (Ctrl-A or Cmd-A) and press the "Export Selected Items To Text File" button (floppy disk icon) at the top right of the LogCat View.
5. Submit this file to the Coursera system.

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, so the whole operation is somewhat pointless, but it give you a chance to create Alarms.

Optional Exercise C: Use Real Tweet data

For advanced students only. If you have access to twitter.com, consider downloading live tweets to your application. You can find out more information about the Twitter APIs at https://dev.twitter.com/docs