

Notifications Lab

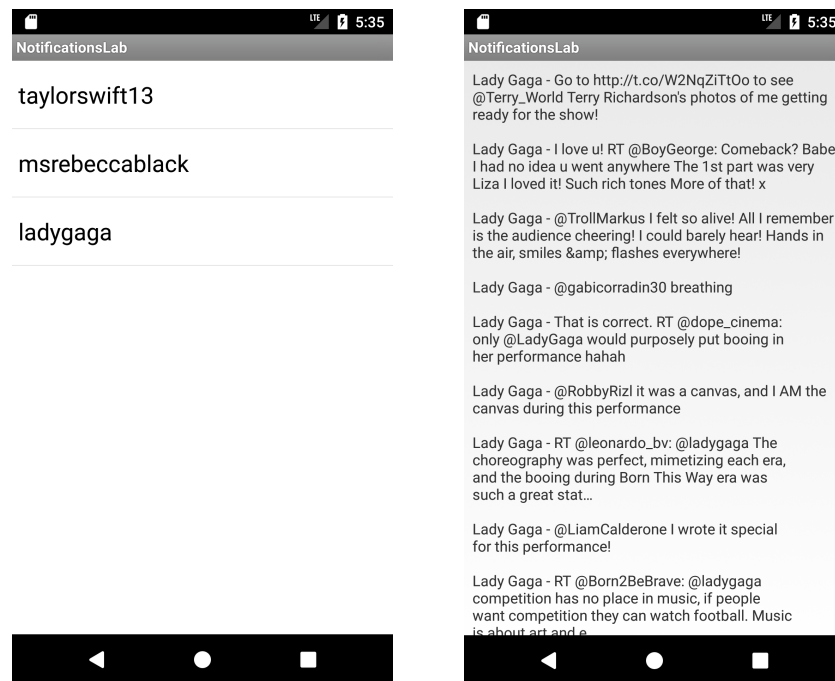
Notifications and BroadcastReceivers

Objectives:

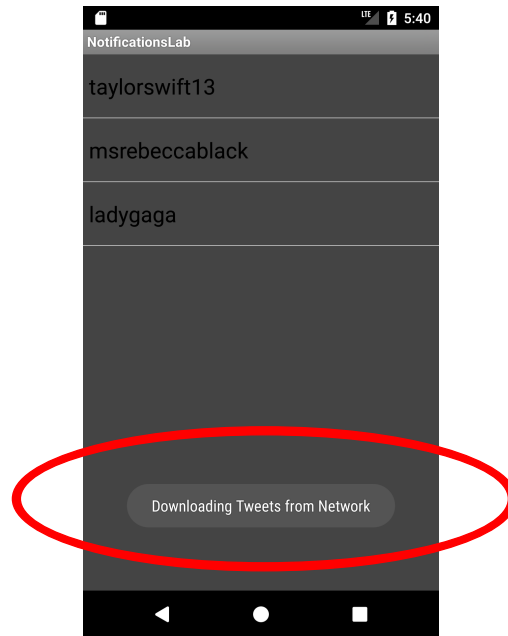
This week's Lab explores User Notifications and Broadcast Receivers. After finishing this Lab, you will have a better understanding of how to create and display different types of User Notifications in order to inform users of your application's actions. You will also learn how to broadcast and receive Intents.

Exercise A:

This Lab involves an app that displays locally stored Twitter data from a set of friends. The basic interface appears below.



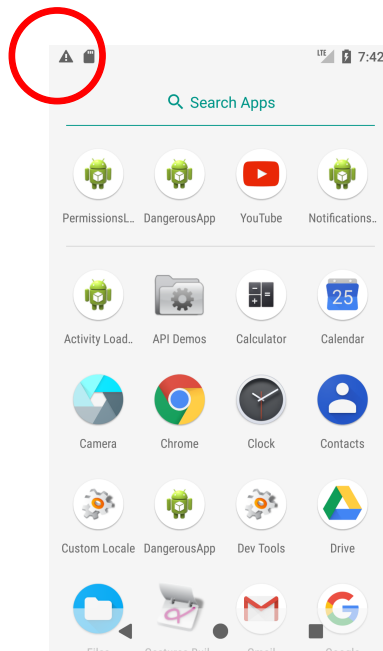
When the application's MainActivity begins running, it determines whether any Tweet data has been downloaded within the last two minutes. If not, it considers any existing data to be stale and therefore downloads fresh Twitter data. To do the “downloading” step the Activity causes an AsyncTask to execute, which will loads the Twitter data off of the main Thread. As the AsyncTask begins to download the Twitter data, the application creates and displays a Toast message, alerting the users that the download is starting. The Toast message is shown below:



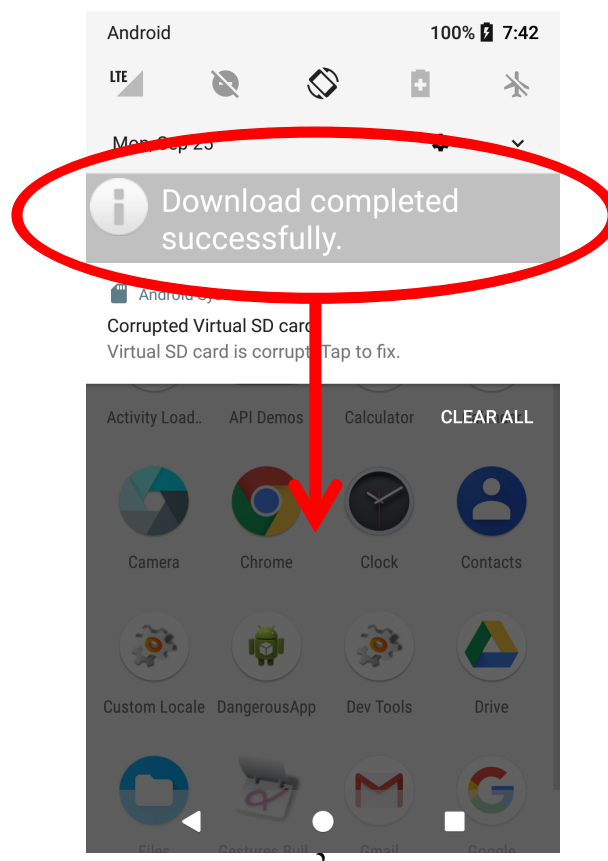
The download process takes a noticeable amount of time. Therefore, it's possible that the user might exit the application while the download is in progress. If that happens, the application uses a Notification Area Notification to inform the user once the download finishes. To do this, when the `AsyncTask` finishes downloading the tweet data, it broadcasts an `Intent`. If the application's `MainActivity` is not running and in the foreground when the `Intent` is broadcast, then the `AsyncTask` creates a Notification Area Notification and places it in the Notification area. However, if the `MainActivity` is running and in the foreground, then the `AsyncTask` does not create this Notification. Instead, it displays a Toast message informing the user that the download is finished.

Specifically, the `AsyncTask` uses the `sendOrderedBroadcast()` method to broadcast an `Intent` once downloading has finished. It also passes its own `BroadcastReceiver` into this call so that it can receive the result of the broadcast. The `MainActivity` dynamically registers a `BroadcastReceiver` to receive this `Intent`, only when the `MainActivity` is visible and in the foreground. If the `BroadcastReceiver` receives this `Intent`, then it will return a specific result code, which lets the `AsyncTask` know that the `MainActivity` is currently visible and in the foreground. If this result code does not arrive back to the `AsyncTask`, then the `AsyncTask` assumes that the `Activity` is not visible and in the foreground and therefore it will send the Notification Area Notification.

If the AsyncTask does send the Notification Area notification, an icon appears in the Notification Area and the user can then examine the notification.

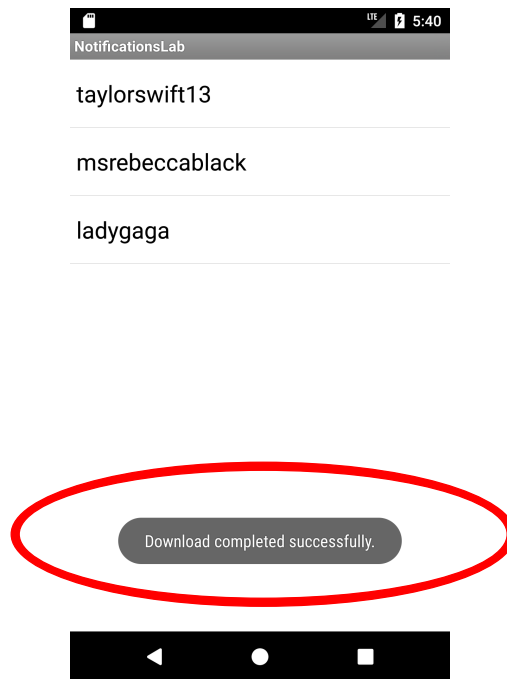


When the user pulls down on the Notification drawer, he or she will see an indication of whether or not the data was successfully downloaded. An example is shown below:



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.

Finally, if the download finishes while the MainActivity is visible and in the foreground, the app displays a Toast message to that effect.



Implementation Notes:

1. Checkout a local copy of the new branch Lab5_Notifications from the upstream repo. See the Gitbook tutorial if you need any help.
2. Implement the TODO comments found in the MainActivity.java and DownloadTaskFragment.java files

In MainActivity.java.

- a. Create a BroadcastReceiver that returns a result code (MainActivity.IS_ALIVE) to inform the AsyncTask that the MainActivity's active and in the foreground, and therefore, the AsyncTask should not send the Notification Area Notification.
- b. Register the broadcast receiver in the protected void onResume() method.
- c. Unregister the broadcast receiver in the protected void onPause() method.

In DownloadTaskFragment.java.

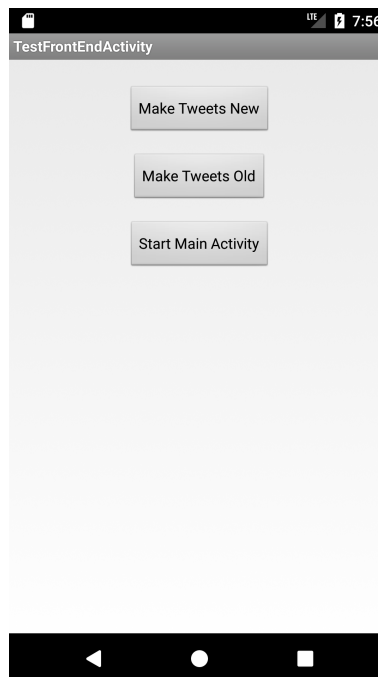
- d. Implement the logic to notify the user that the feed has been downloaded using sendOrderedBroadcast(). You will need to create a BroadcastReceiver to receive the result of this broadcast. If that result is not MainActivity.IS_ALIVE then this BroadcastReceiver should create a Notification Area Notification and create a NotificationChannel to support API level 26.

Testing:

The test cases for this Lab are in the Lab5_Notifications project. You can run the test cases either all at once, by right clicking the test package and then selecting Run 'Tests in 'course.lab...', or one at a time, by right clicking on an individual test case class 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 Pixel XL with API level 26. To limit configuration problems, you should test your app against a similar AVD.
2. These test cases will start the application, not at the MainActivity, but at another Activity called, TestFrontEndActivity. If you look into the AndroidManifest.xml file for this application, you'll see that both of these Activities are main entry points for the app. In fact, when you install this app, two icons will appear in the launcher. This approach 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 GitLab.

Submission

To submit your work for grading, commit all of your changes locally and then push to your origin repo on gitlab. See the gitbook tutorial if you need assistance with this.