Лабораторная по Уведомлениям

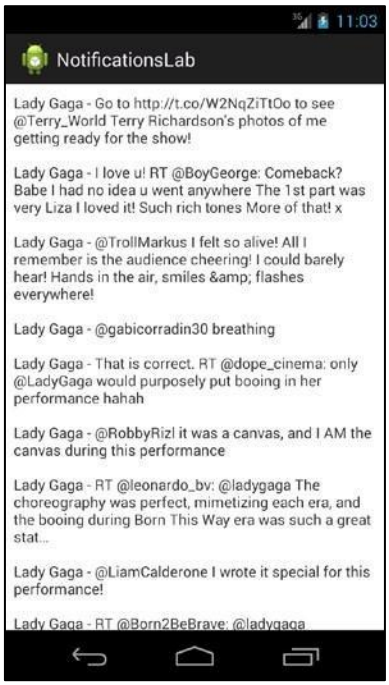
Уведомления и BroadcastReceiver

Цели:

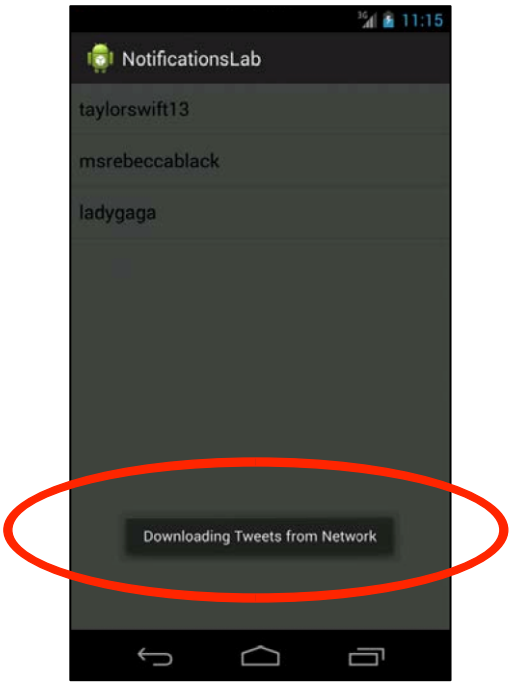
Лабораторная этой недели исследует Пользовательские Уведомления и Обработку широковещательных запросов. Выполнив эту Лабораторную, вы будете лучше понимать как создавать и отображать различные типы Пользовательских Уведомлений для того чтобы информировать пользователей о действиях вашего приложения. Вы также научитесь как отправлять широковещательные запросы и получать соответствующие Интенты.

Задание A:

Данная лабораторная отображает локально сохраненные данные Твиттера для набора друзей. Базовый интерфейс приведен ниже.

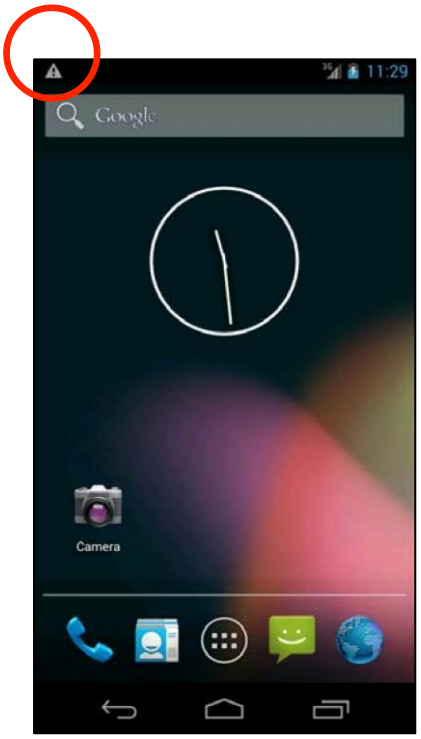


Когда MainActivity приложения запускается, оно проверяет, были ли скачаны данные ленты Твиттер в последние две минуты. Если нет, воспринимает имеющиеся данные за старые и запускает процесс скачивания свежих данных Твиттера. Для организации скачивания Activity использует AsyncTask , который загружает данные Twitter в отдельном Потоке. В тот момент, когда AsyncTask начинает скачивать данные Twitter, приложение создает и показывает Toast уведомляя пользователя о том, что началось скачивание. Сообщение тоста показано ниже:

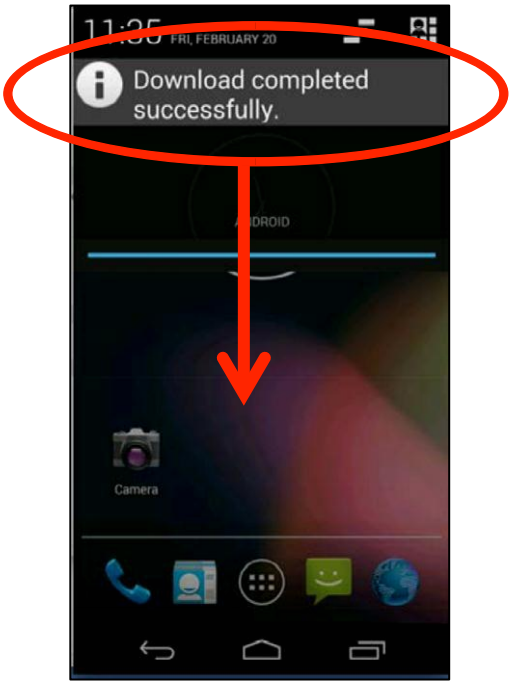


Процесс скачивания занимает значительную часть времени. Поэтому, может возникнуть такая ситуация, когда приложение было закрыто во время процесса скачивания. Если это произойдет, приложение использует область для Уведомлений чтобы информировать пользователя о том, что скачивание завершилось. Для этого, когда AsyncTask завершает скачивание данных твитовон посылает широковещательный запрос. Если MainActivity приложения не запущено и не отображается в данный момент на экране устройства, тогда AsyncTask создает Уведомление и помещает его в области Уведомлений. Однако, если MainActivity запущен и виден на экране, тогда AsyncTask не создает Уведомление. Вместо этого он отображает Всплывающее Toast сообщение, информируя пользователя, что скачивание завершилось.
Вдаваясь в конкретику, AsyncTask использует метод sendOrderedBroadcast() чтобы отправить широковещательный Intent когда скачивание завершилось. MainActivity динамически регистрирует BroadcastReceiver чтобы получить Интент, только когда MainActivity виден и находится на экране. Если BroadcastRecevier получает этот Intent, тогда он вернет специфичный код результата, который позволит AsyncTask узнать, что MainActivity видно на экране. Если этот код результата не передан обратно в AsyncTask, тогда AsyncTask подразумевает, что Activity не виден и не на экране и поэтому он оно отправит Уведомлению в Область Уведомлений.

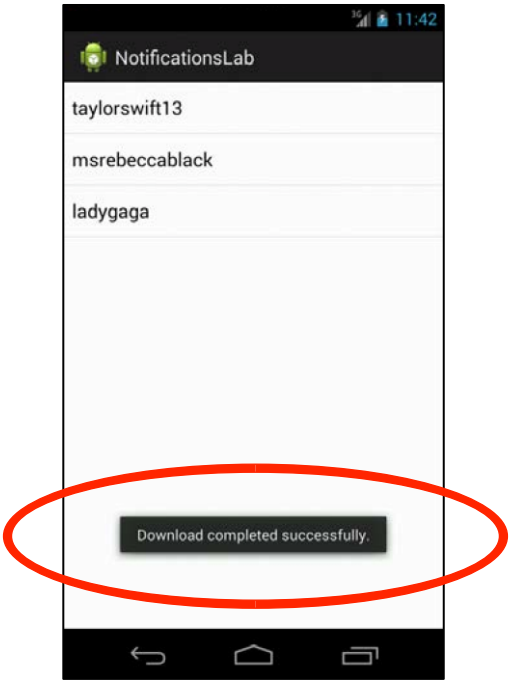
Если AsyncTask не отправляет Уведомление, пиктограмма появляется в Области уведомлений и пользовать может им воспользоваться.



Когда пользователь опускает выдвижную область уведомлений он видит индикацию того, что данные были успешно скачаны. Пример приведен ниже:



Если пользователь нажмет на Уведомление, MainActivity должно открыться вновь. И вновь, если MainActivity стартует более чем через две минуты после того как данные были скачаны, то MainActivity должно скачать данные снова. В противном случае, скачивание не производится. И, наконец, если скачивание завершается пока MainActivity видно на экране приложение отображает Всплывающее сообщение.



Архив приложения содержит видео демонстрацию работы приложения.

Заметки по реализации:

1. Скачайте файлы скелета приложения и проимпортируйте их в вашу IDE.
2. Реализуйте все найденные комментарии TODO в файлах MainActivity.java и DownloadTaskFragment.java
   a. Создаем BroadcastReceiver, который возвращает код результата (MainActivity.IS_ALIVE) чтобы проинформировать AsyncTask, что MainActivity активно и видно на экране, и следовательно, AsyncTask не должно отправлять Уведомление в Область Уведомлений.
   b. Регистрируем BroadcastReceiver в методе protected void onResume().
   c. Снимаем регистрацию в методе protected void onPause(). В DownloadTaskFragment.java.
   d. Реализуем логику для уведомления пользователя, что лента была скачана с использованием sendOrderedBroadcast(). Вам потребуется создать BroadcastReceiver чтобы получить результат данного широковещательного запроса. Если этот результат

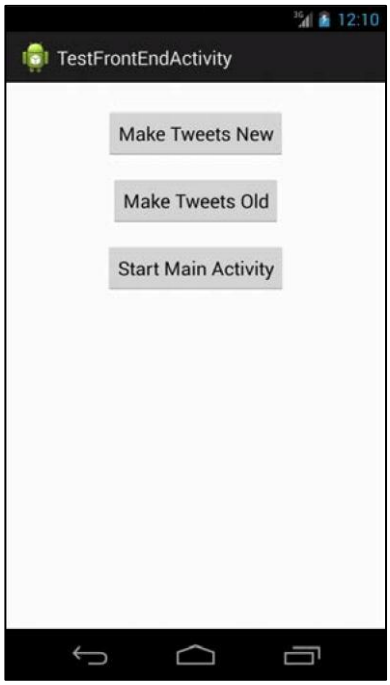
не MainActivity.IS_ALIVE , то этот BroadcastReceiver должен создать Уведомление для Области Уведомлений.

Тестирование:

Тест-кейсы для данной лабораторной находятся в модуле Lab6_NotificationsLabTest. Вы можете запустить их либо все разом, либо по одному с помощью контекстного меню по папке с модулем, выбрав Run As>Android Junit Test, или по одному, нажимая правую кнопку мышина отдельном классе тест-кейса и затем продолжая как раньше. Тесты используют Robotium. По мере реализации различных шагов Лабораторной, запускайте тест для проверки написанного кода, чтобы убедиться, что вы делаете прогресс в выполнении Лабораторной.

Предупреждения:

1. Эти тест-кейсы были протестированы на эмуляторе Galaxy Nexus AVD с API level 18. Чтобы ограничить возможные проблемы конфигурации, вам следует тестировать приложение на аналогичном AVD.
2. Эти тест-кейсы стартуют приложение, но не MainActivity, а другое Activity с именем TestFrontEndActivity. Если вы посмотрите в AndroidManifest.xml для данного приложения, вы увидите, что оба этих Экрана являются точками входа в приложение. Фактически, когда вы устанавливаете это приложение, появляются две иконки запуска. Данный подход позволяет нам модифицировать возраст уже скачанных данных Твиттер перед запуском MainActivity. Интерфейс данного Activity приведен ниже.



Как только все тесты проходят, можно загружать ваш проект.

Отправка

Чтобы убедиться в том, что ваше задание будет оценено корректно, уделите внимание следующим моментам:

1. Папка с вашим проектом должна быть в zip архиве NotificationsLabSubmit.zip.
2. Предварительно необходимо очистить папку с проектом от сгенерированных файлов class и dx. Это можно сделать с помощью ручного удаления папок out и/или build.

Если вы чувствуете, что можете сделать больше, то мы предлагаем следующие возможные пути улучшения приложения. Данное задание необязательное и не влияет на оценку.

Необязательное Задание B: Добавьте Alarm

Модифицируйте ваше приложение так, чтобы ваши данные Twitter всегда были свежими. Используйте Alarm для повторного скачивания данных Twitter каждые две минуты. В реальном приложение скачивание каждые две минуты будет, возможно избыточным. Вы можете поиграть с частотой скачивания. В добавок, если данные не будут изменяться, вся операция в целом бессмысленна, но несмотря на это дает возможность потренировать Alarm.