Assignment 01

First Steps

Prepare the Android development environment and create your first test app to check all involved components

Download the Android development IDE of your choice

Android currently offers two convenient ways to develop Apps for Android. Choose the IDE of your choice and set it up correctly, the ADT might be more suitable if you are already familiar with eclipse.

1) Android Development Tools (ADT) for Eclipse
   http://developer.android.com/sdk/index.html

2) Android Studio
   http://developer.android.com/sdk/installing/studio.html

Both environments come pre-bundled with the latest Android SDK and Emulator.

Create first app and run it on your device or in the emulator

Create a first sample App using the default values and get familiar with the structure of the IDE. Identify the IDE modules that let you inspect logs generated on the device (LogCat).
For running the App in the emulator, first create a new emulator configuration. Afterwards start the emulator and run the App in it.
If you have your own Android device, try running the App on it. You will probably have to install a driver on Windows machines; Linux and Mac should display your device as a startup target directly. It might also be necessary to bring your device into developer mode.

Note: Have a look at the Android tools reference for more information (http://developer.android.com/tools/workflow/index.html).

Pirate Slang Conversion App

As your first assignment, you will build an App that translates a given text into pirate slang and displays the result to the user. It makes use of standard design elements (ActionBar, NavigationDrawer) and uses a web service for the translation of the text.

Modify the main layout

The default activity layout should consist of two text elements (one for inserting the text to translate, the other for displaying the translation) and two buttons (one for clearing both text fields, one for submitting a new translation request). Together, all elements should take up the available content area.

Afterwards, wire all elements up to the activity in order to change the text programmatically and react to button events.

Create the translation service

For translating arbitrary text to pirate speech, we use the ARRPI API (http://isithackday.com/arrpi.php) that offers an easy to use interface for the conversion. The ARRPI accepts a `text` parameter in the requests and the response directly gives you the respective translation.

For setting up the GET query to this endpoint you are free to use the Http API of you choice. Android already comes with two options:

1) Apache HttpClient
2) HttpUrlConnection (favored since Android 3.0)

While those implementations are sufficient for the task, they are cumbersome to set up and to handle. Therefore I encourage you to use a third party wrapper library of your choice which encapsulate the lower level functionality and provide a convenient high level API. Possible solutions:

1) Retrofit (http://square.github.io/retrofit/)
2) RestTemplate (http://projects.spring.io/spring-android/)
3) Volley (https://developers.google.com/events/io/sessions/325304728)

Note: Some libraries already take care of dispatching the Http requests on a separate thread if you decide to use callbacks. So dependent on your choice, it might no be necessary to dispatch the requests yourself onto a different thread.

Wrap all the translation functionality in a new class in order to keep the activity clean. This will also allow you to reuse it in other places when needed.

Afterwards add the service to the Activity as follows:

1) Clicking the translate button will start a request with the text from the input text field
2) Once a response is received, it should be displayed in the respective text field

Provide a history Activity that displays recent translations

Now that we have the translation service up and running, it's time to add a history that displays the last translations (max 20, older ones will be removed).

Create a new activity and a respective layout, only containing a ListView used to display the latest entries. Each item in the list should show two lines, the upper one displaying the beginning of the original text, while the lower one displays the beginning of the translation.
In order to display the history, you have to come up with an implementation for storing past translations. Make sure that you only keep the last 20 requests in there.

Note: Android already provides build-in list item types with two rows and a ListAdapter that operates on arrays.

Add a NavigationDrawer to navigate do different Activities

In order to navigate to the history Activity, add a NavigationDrawer that displays two entries:

1) Pirate Translation
2) Pirate History

A click on the “Pirate History” is starting a new history activity that is displayed on top of the current one. Using the back button on his will bring you back to your translation Activity. A click on “Pirate Translation” does nothing (except closing the drawer).

Note: In an upcoming assignment, we’ll get into the world of fragments and clean up the UI workflow. For now the behavior using plain activities will be sufficient.

Master Students: Make the history persistent in a SQLite database

This applies to master students only!

Replace the current history storing functionality with an SQLite store. Save the history of translations in an SQLite database on the device and implement an access wrapper for the items.
The history Activity should then use this information source for displaying the recent translations (again, max 20) and also persist the history throughout App restarts.

Submission

Please zip up your complete Android project and hand it in via Uniworx. Projects that do not compile due to errors will not be accepted.