Assignment 02

Pirate Slang Conversion App

After creating our own pirate translation app in Assignment 01 we will now start to improve it: First we will include an online pirate translation database to make our translations more accurate. Then we will save past translations so users can look at them again later.

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 translation (this means that our own translation must be applied to the response of the database).

For setting up the GET query to communicate with the ARRPI, you are free to use the Http API of your 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 may be somewhat cumbersome to set up and handle. Therefore, I encourage you to use a third party wrapper library of your choice which encapsulates the lower level functionality and provide a convenient high level API. Possible solutions include:

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 not be necessary to dispatch the requests onto a different thread yourself.

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 and it has also been processed by your own translator it should be displayed in the second text field (as in assignment 01)

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 three lines: The first one should show a preview of the original text, the second one a preview of the translated text, and the last line should show time and date when the user requested the translation.

Make sure that lines have different text sizes or other varying attributes which make the entry more readable. 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.

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 database. 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 retain 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.