Assignment 02

Group Phase Preparation

In the next lecture, we'll get started with the group project. Therefore, you can already get started forming teams of 4 and filling out this form http://goo.gl/xdZSKA afterwards.

Pirate Slang Conversion App Enhancements

In your last assignment, you build a simple app that provided the ability to translate arbitrary text into pirate slang. In this assignment, we will build up on those results and add additional functionality to it.
This includes adding notifications, the addition of a horizontal and vertical layout, as well as converting the plain Activity layouts into ones that use Fragments.

Convert history Activity to make use of Fragments

Until now, the Activities all use their own layout and the functionality is tied to the Activity itself. In order to provided different behavior based on different screen orientations or sizes, its now up to you to convert the existing history Activity to a more generic implementation using a dedicated Fragment for displaying the history list and its items.
In the end, you will still end up with the same activity, but rather than adding the list view to the layout, the list view fragment is added to the Activities layout.

It will probably take you a minute or two to realize what this switch means for reusability but we will come back later to this point and build up on the advancements you just implemented. The Android developer pages provide a good introduction into the use of Fragments (https://developer.android.com/training/basics/fragments/index.html, http://developer.android.com/guide/components/fragments.html)

Post a Notification once a translation was made

Back in the translate Activity, we'll add functionality to post Notifications once a translation was successful. In order to do this, create an appropriate notification layout (http://developer.android.com/design/patterns/notifications.html, http://developer.android.com/guide/topics/ui/notifiers/notifications.html) and post one each time the translation service successfully translated a text. The notification does not have to provide additional functionality like expanding or interaction (but it can ☺), but should display the latest input and output text (style and length is up to you).

Add a new Activity and Fragment displaying the complete text of a history item

For now, the app only displayed the last translation history in a list format, probably cutting most of the actual content. Therefore, we are now adding a new history detail view that displays a single item that was previously selected in the list. This should be based on a new Fragment displaying the original text and the translation (same as with the translate Activity, both taking up half of the scree), encapsulate in its own Activity.

The behavior should be as follows:

1. A click on a list item in the history list Fragment should notify the underlying Activity that an item was selected using a dedicated callback (e.g. create your own interface).
2. The underlying activity receives this callback and starts the newly created history details Activity.
3. In order for the details view Activity to know which translation it should display, you will have to provide either a reference (ID) or the complete translation in the starting Intent.
4. The back button on the new activity behaves as expected and brings you back to the history list

The Android resources provide a nice introduction to the Fragment communication: http://developer.android.com/training/basics/fragments/communicating.html.

Change the History display based on the current screen orientation

Now that we have the history functionality up and running it is time to make use of the fragments we just created for both views (list and details).
Normally you would reuse fragments and their layout position based on the type of device the app runs on. The Android developer portal provides a good overview of the general idea: http://developer.android.com/training/basics/fragments/index.html

In order to keep it targeted on mobile phones, we do not change the layout based on device type, but rather depending on the screen orientation (landscape vs. portrait).

Note: You should make use of the different layout loading mechanisms provided by the Android framework, but you will also have to add you own logic to the activities to provide the expected behavior.
In portrait mode, the history should not change behavior or layout:

- One Activity displays the history list fragment and reacts to clicks via the callback.
- The second Activity is started once an item is clicked, displaying the history details fragment.
- Using the back button on the details Activity brings you back to the history list.

In landscape mode, we will make use of a combined layout, displaying the list and details Fragment side-by-side:

- As with the example on the developer portal (tablet vs. mobile), the history Activity should display the list fragment on the left side while simultaneously displaying the history details on the right side of the screen.
- Once entering the history Activity, the list Fragment displays the history, while the details Fragment either shows a placeholder or no text at all.
- When a history item is selected in the list Fragment, the details Fragment will update and display the complete translation. (Make use of the callback for the list Fragment you implemented earlier).
- Now there is no need to start an additional Activity compared to the portrait mode, as we display all information directly in one view.

Master Students: Implement the swipe view pattern for the history details Activity

This applies to master students only!

In order to make navigating the translation history more elegant, implement the swipe view pattern (http://developer.android.com/design/patterns/swipe-views.html) for the history details Activity. This way, you will be able to swipe between the history items without having to go back to the list view.
Note that this behavior should only be present in portrait mode, as the landscape mode already displays the history list and the actual history item side-by-side.

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.