

Higher Diploma in Computing, DT8265 Programming for Mobile and Smart Devices

Coursework 3 – User Interfaces

Release Date: 26th Oct 2017 (Wk7)

Submission Date: 26th Nov 2017 (Wk11 / 4wks)

Objectives:

Familiarize yourself with Android's User Interface (UI) classes. Create a simple application that uses a variety of UI elements including: Buttons, TextViews, EditTexts RadioButtons and Checkboxes. You will also reinforce the knowledge you've gained in the lab exercises by implementing a larger portion of the application code from scratch.

Description:

In this coursework, you will create a *ToDo Manager* application. The application creates and manages a list of *ToDo Items* (i.e. things that you need “to do.”). This application's user interface including its layout and resource files are already pre-designed. You will also implement a bit more of the application's features than you did in the previous Project. Do NOT modify any resource IDs or log messages contained in the skeleton layout files as this may break the test cases provided.

Base Requirements:

The main Activity for this application is called *ToDoManagerActivity*. When the Activity runs, but there are no previously saved *ToDo Items*, its initial UI will look something like this:

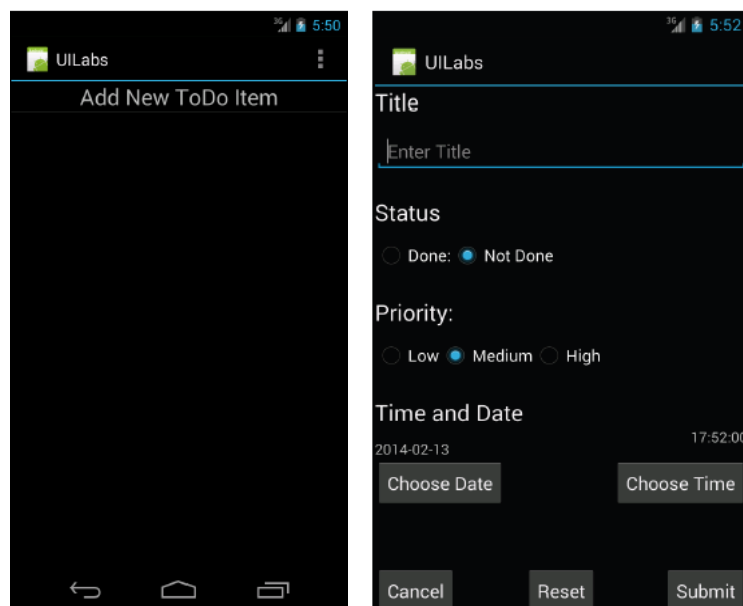


Figure 1: Initial View of *ToDoManagerActivity* UI and *AddToDoActivity* UI

This UI contains a single *ListView* that displays all existing *ToDo Items*. As shown above, the last row of the *ListView* always displays a special View, with the words, “Add New ToDo Item”. This last position within the *ListView* is known as the “footer.” You can add a View to the footer by using the *ListView*'s [addFooterView\(\)](#) method.

When the user clicks on the *ListView* footer, the application will start a new Activity called *AddToDoActivity* that allows the user to create and save a new *ToDo Item*.

ToDo items have the following fields. Default values (see *AddToDoActivity*) appear in bold:

- **Title**: A user-provided String. The default Title is the empty String ("").
- **Status**: One of {Done, **Not Done**}
- **Priority**: One of {Low, **Medium**, High}
- **Time & Date**: A deadline for completing this ToDo Item. The default deadline is 7 days from the current date and time.

This Activity's user interface includes the following buttons:

- **Cancel** – finish the Activity without creating a new ToDo Item.
- **Reset** – reset the fields of the ToDo Item to their default values and update the display to reflect this.
- **Submit** – create a new ToDo Item containing the user-entered / default data fields and return it to *ToDoManagerActivity*. When the application returns to *ToDoManagerActivity*, the new ToDo Item should appear in the ListView, just above the footerview.

For example, if the user creates and submits a new ToDo Item to an empty ToDo list, then once the application returns to the *ToDoManagerActivity*, its ListView will contain the new ToDo Item, as shown highlighted below.

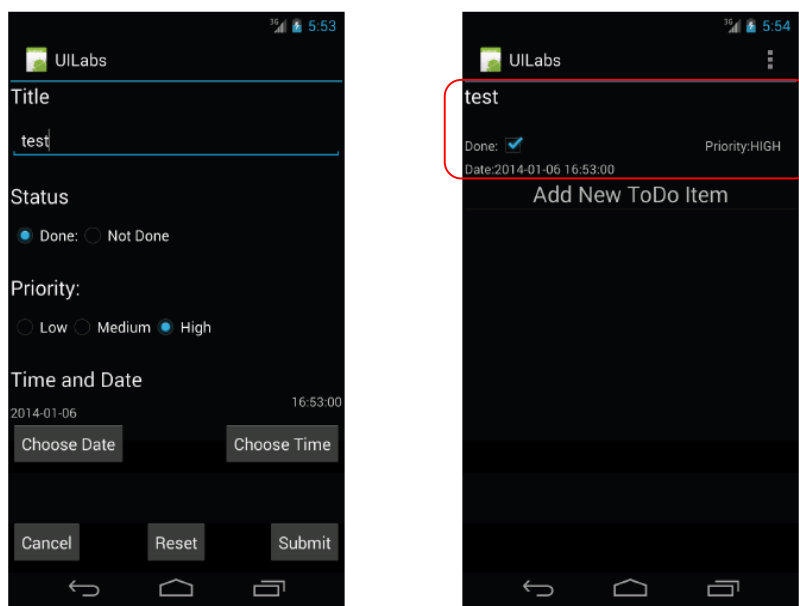


Figure 2: (a) The user creates a new ToDo Item and hits Submit. (b) After submitting the new ToDo Item, the application returns to the main Activity, displaying the new ToDo Item (shown highlighted).

Back in the Main Activity, the user will be able to toggle the Done checkbox to indicate that the ToDo Item's Status has changed, say from Done to Not Done but will not (currently) be able to modify any other fields.

Implementation:

1. Download the UILabs zip file from the webcourses Android Coursework folder, unzip it and open the Project UILabs directly in Android Studio IDE. This Project was created in a previous API level without backward compatibility features. Using the appropriate Project Template construct a new Project UILabs and incorporate the backward compatibility features needed, as per the labs. Use the *WeatherViewer* app as a comparison. Update the entry Activity *ToDoManagerActivity* to include a FAB and a ListView as per the *WeatherViewer* app.
2. Implement the necessary code within the source code in the UILabs project according to the specifications described above. Look for comments in the skeleton files containing the String "//TODO" or use the TODO tab to access them more easily. As in the previous Coursework, these comments contain hints as to what you need to do to complete the project.

UILabs Project java src:	TODO
ToDoManagerActivity.java	yes
AddToDoActivity.java	yes
ToDoListAdapter.java	yes
ToDoItem.java	no

Additional Functionality:

1. Modify your application so that when added, ToDoItems whose *Status* is *Not Done* are displayed in the Main Activity with a semi-transparent **MAGENTA** coloured background and those whose status are *Done* are displayed with a semi-transparent **CYAN** coloured background. If the user toggles the *Done* checkbox, the background color of the Item should also change accordingly.
2. Modify the app to change the title text color (or add a warning icon) as the *deadline* of the ToDo Item gets closer, say coloured **RED** from 1 day previous and with a push Notification, and an alarm from 5 mins before the *deadline*.
3. Currently the user cannot modify the ToDoItem's *Priority* via the ToDoItem ListView. Modify this widget so that it provides a drop down list, allowing users to select a different *Priority*. (Note this will effect a sort – see 5.)
4. Modify your application so that if the user long presses a ToDoItem in the Main Activity's ListView, a vibration triggers and an alertdialog pops up, allowing the user to delete the selected ToDoItem or cancel. Choosing delete should delete the ToDoItem from view as well as the pending Notification and alarm.
5. Add 2 Tab indicators to the app. Have one Tab display ToDoItems sorted by *deadline*, and another Tab for ToDoItems first sorted by *priority*, and within a particular priority further sorted by *deadline*. Provide screenshots in your submission to support sort examples.

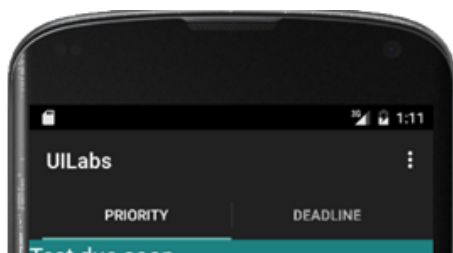
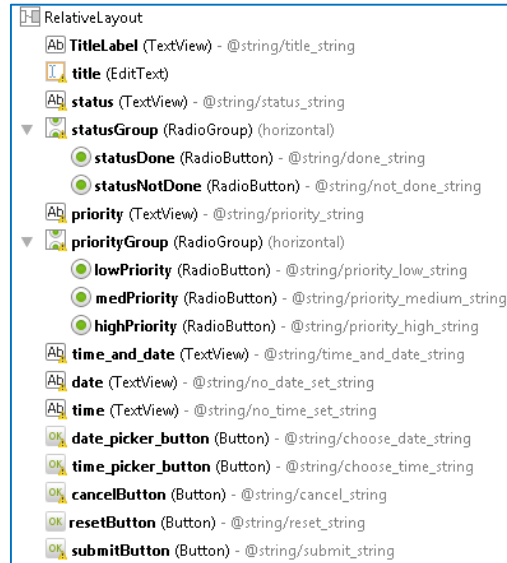
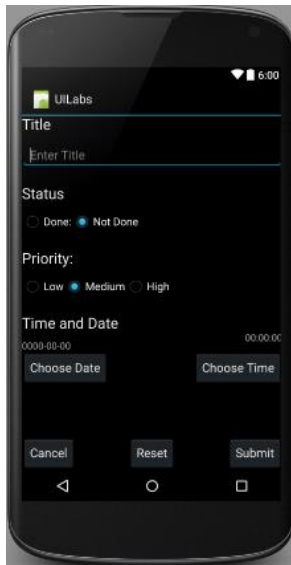


Figure 3: Sample Tab Design.

For each additional numbered piece of functionality completed, add an explanatory "//TODO" String and its number, within your Project code as well as in your submission document.

Consult Reference pdfs in webcourses and the android API online.

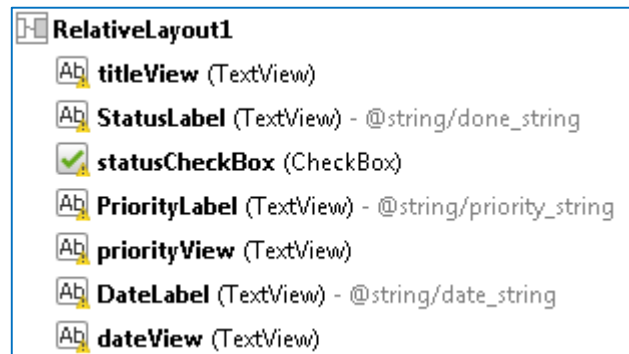
Layout Files:



add_todo.xml




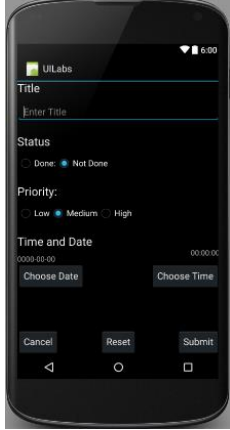

footer_view.xml (use a Floating Action Button to add a new ToDoItem & a ListView to display items)



todo_item.xml (list_item.xml in the WeatherViewer app)

Figure 4: The XML Layout Files employed in the App.

Application Flow:

1. ToDoManagerActivity.java	2. AddToDoActivity.java	3. ToDoListAdapter.java	4. ToDoItem.java
 <p>footer_view.xml</p> <p>* UI with a single ListView that displays all existing ToDo Items. * the last row of the ListView displayed a special View, with the words, “Add New ToDo Item” in the old Project. This last position within the ListView is known as the “footer.” You added a View to the footer by using the ListView’s addFooterView() method, but now handled by the FAB listener.</p> <p>*onCreate() mAdapter</p> <p>*onActivityResult() →ToDoItem(Intent)</p> <p>*onResume() → loadItems() → ToDoItem(Title, Priority, Status, Date)</p> <p>*onPause() → dump() *onCreateOptionsMenu() *onOptionsItemSelected()</p>	 <p>add_todo.xml</p> <ul style="list-style-type: none"> • Title: A user-provided String. The default Title is the empty String (“”). • Status: One of {Done, Not Done} • Priority: One of {Low, Medium, High} • Time & Date: A deadline for completing this ToDo Item. The default deadline is 7 days from the current date and time. <p>• Cancel – finish the Activity without creating a new ToDoItem.</p> <p>• Reset – reset the fields of the ToDoItem to their default values and update the display to reflect this.</p> <p>• Submit – create a new ToDoItem containing the user-entered / default data fields and return it to <i>ToDoManagerActivity</i>. * application returns to <i>ToDoManagerActivity</i>, the new ToDoItem should appear in the ListView, just above the footerview. → ToDoItem.packageIntent(Intent, ..)</p>	 <p>todo_item.xml</p> <p>*ToDoListAdapter() *add() *clear() *getCount() *getItem() *getItemId() *getView() *log()</p>	<p>1. ToDoItem(Intent) 2. packageIntent(Intent,...) 3. ToDoItem(Title, Priority, Status, Date)</p>

Testing:

These test cases are designed to drive your app through a set of steps.

test case 1: The *TestSubmit* test case should output exactly 4 Log messages.

test case 2: The *TestReset* test case should output exactly 6 Log messages.

test case 3: The *TestCancel* test case should output exactly 7 Log messages.

During testing, start the application from the launcher with your device in Portrait mode.

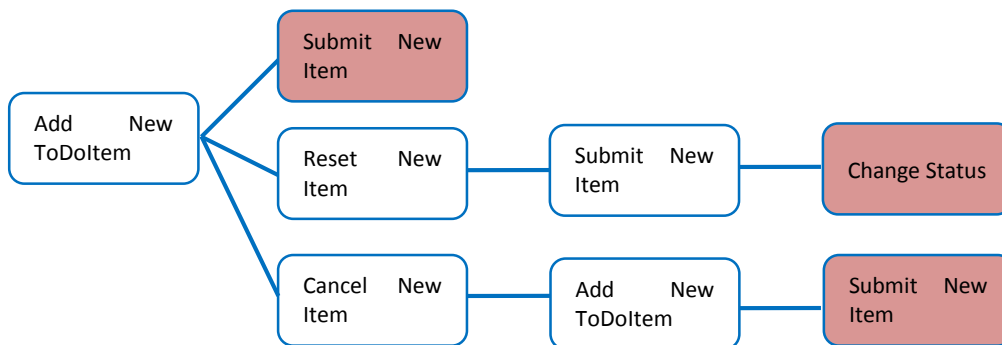


Figure 5: The 3 test scenarios to run.

Logcat Output:

Enable the Android Studio logcat view, if it isn't enabled already. To do this, select **View -> Tool Windows -> Android Monitor -> logcat**.

Connect your device or start your emulator. Make sure that you can see output in your logcat view. Change the Log level to **Info**. To the right of Log level, there is a search bar. In the bar enter the following string, without the quotes: "**Lab-UserInterface**". Make sure you enter this string exactly. This will filter the logcat output to only show Log messages whose tag field matches this string.

For each test case repeat the following steps:

Start with the App closed but its icon visible in your device. Towards the left of the logcat view, there is a "Clear logcat " button (a trash can icon). Press this button to clear the log. Now run your test as per the scenarios graphic above.

After the test finishes, Click anywhere in the logcat view, then select all (ctrl + A).

This will select all the logcat output from the test run. Click (ctrl + c) to copy, (ctrl + v) to paste into the Project Submission.

Submission:

A digital submission in a **.zip** file is to be uploaded via the webcourses module. It should contain:

1. an electronic copy of the UILabs Android project containing the added code.
2. A project report (see *CW3 User Interfaces Submission*).