

# Lab – User Interface Classes

---

*Create a complex user interface with Android's user interface classes*

## Objectives:

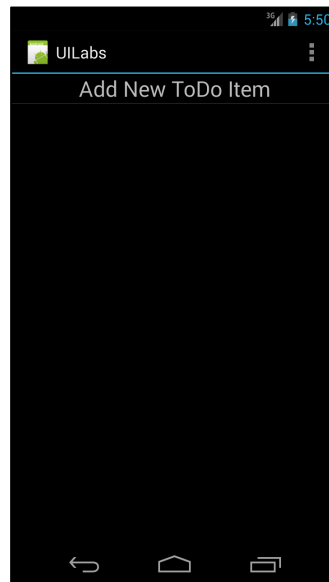
Familiarize yourself with Android's User Interface (UI) classes. Create a simple application that uses a variety of UI elements including: Buttons, TextViews and Checkboxes. You will also reinforce the knowledge you've gained in previous lessons by implementing a larger portion of the application from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

## Overview:

In this Lab, 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.”) You will design this application's user interface, including its layout and resource files. You will also implement a bit more of the application's features than you did in previous Labs. Do **NOT** modify any resource IDs or log messages contained in the skeleton layout files as this may break our test cases.

## Exercise A: The basic ToDo Manager

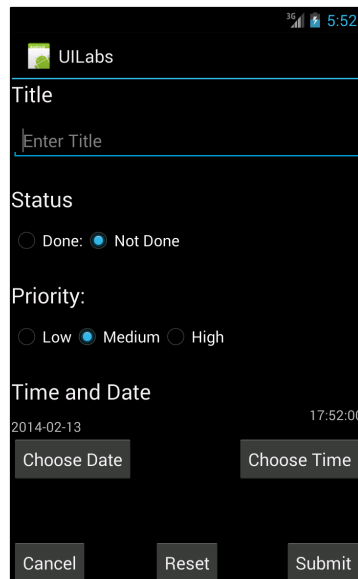
The main Activity for this application is called `ToDoManagerActivity`. When the Activity runs, but there are no previously saved ToDo Items, its initial UI will look something like this:



*Figure 1: Initial View*

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 AddToDoItemActivity that allows the user to create and save a new ToDo Item.



*Figure 2: Adding a New ToDo Item*

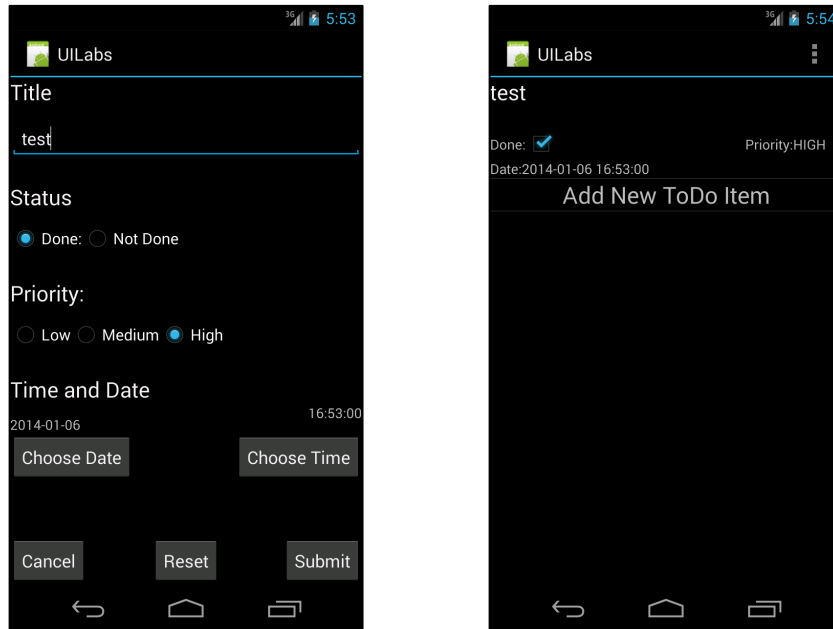
ToDo items have at least the following fields. Default values appear in bold:

- Title: A user-provided String. The default Title is the empty String ("").
- Status: one of {Done, **Not Done**}
- Priority: one of {Low, **Med**, 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.

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 below.



*Figure 3: (a) The user creates a new ToDo Item. (b) After submitting the new ToDo Item and the application returns to the main Activity, displaying the new ToDo Item.*

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 Not Done to Done.

## Implementation Notes:

1. Download the application skeleton files, for the UILabs project, from the assignment page and import them into your IDE.
2. Implement the project according to the specifications described above. To implement the Lab, look for comments in the skeleton files containing the String "//TODO." As with previous Labs, these comments contain hints as to what you need to do to complete the project. However, be aware that from here on out these comments will become increasingly less comprehensive, requiring you to make more decisions about how to implement the whole project.

## Testing

The test cases for this Lab are in the UILabsTest project. You can run the test cases either all at once, by right clicking the project folder and then selecting Run As>Android JUnit Test, or one at a time, by right clicking on an individual test case class (e.g., TestSubmit.java) and then continuing as before. The test classes are Robotium test cases. You will eventually have to run each test case, one at a time, capture log output, and submit the output to Coursera. These test cases are designed to drive your app through a set of steps, passing the test case is not a guarantee that your submission will be accepted. The TestSubmit test case should output exactly 4 Log messages. The TestReset test case should output exactly 6 Log messages. The TestCancel test case should output exactly 7 Log messages.

As you implement various steps of the Lab, run the test cases every so often to see if you are making progress toward completion of the Lab.

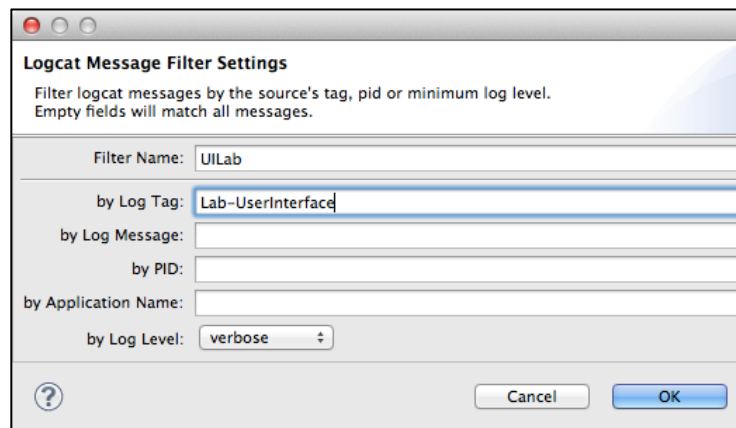
### Warnings:

1. These test cases have been tested on a Galaxy Nexus AVD emulator with API level 18. To limit configuration problems, you should test you app against a similar AVD. Also, when testing, make sure that your device is in Portrait mode when the test cases start running.
2. On startup, these test cases delete all existing ToDoItems.

Once you've passed all the test cases, submit your log information to the Coursera system for grading.

### Tips: Saving a LogCat filter.

1. In the LogCat View, press the green "+" sign to "add a new LogCat filter."
2. A dialog box will pop up. Type in the information shown in the screenshot below.
3. A saved filter called, "UI Lab" will appear in the LogCat View.



### Tips: Running your test cases and capturing your LogCat output for submission.

1. For each test case, clear the LogCat console by clicking on the "Clear Log" Button (the Button with the red x in the upper right of the LogCat View).
2. Then right click on an individual test case file in the Project Explorer. Run As -> Android JUnit Test.
3. When the test case finishes, if it's not already selected, click on the "UILab" filter in the left hand pane of the LogCat View.

4. Select everything within the LogCat View (Ctrl-A or Cmd-A) and press the "Export Selected Items To Text File" button (floppy disk icon) at the top right of the LogCat View.
5. Submit this file to the Coursera system.

If you got through Exercise A and submitted and passed the tests, and feel that you'd like to do more, here are some suggested additions. This is optional and ungraded, but if you do something cool, please consider posting some screenshots on the forums.

### Optional Exercise B: A Prettier ToDo Manager Application

Right now the ToDo manager is quite ugly. Try modifying the layout files to create more attractive and useable layouts. For example, play with the font size of the text, the amount of padding around the elements, background colors and more.

### Optional Exercise C: An improved ToDo Manager Application

Modify your application so that `ToDoItems` whose status is Not Done are displayed in the Main Activity with a different colored background than those whose status are Done. If you do this, then when the user toggles the Done checkbox, the background color should also change as appropriate. You might also consider changing the background color or adding a warning icon as the `ToDoItem`'s deadline get close. Finally, right now, the user can't modify the `ToDoItem`'s priority from the `ToDoItem` `ListView`. Modify this user interface so that it provides a drop down list, allowing users to select a different Priority.

### Optional Exercise D: A ToDo Manager Application You Might Actually Use

Modify your application so that if the user long presses a `ToDoItem` in the Main Activity's `ListView`, a dialog pops up, allowing the user to delete the selected `ToDoItem`. If you're really feeling adventurous place the app inside a `Tab`. 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.