CMSC 436 Lab 10

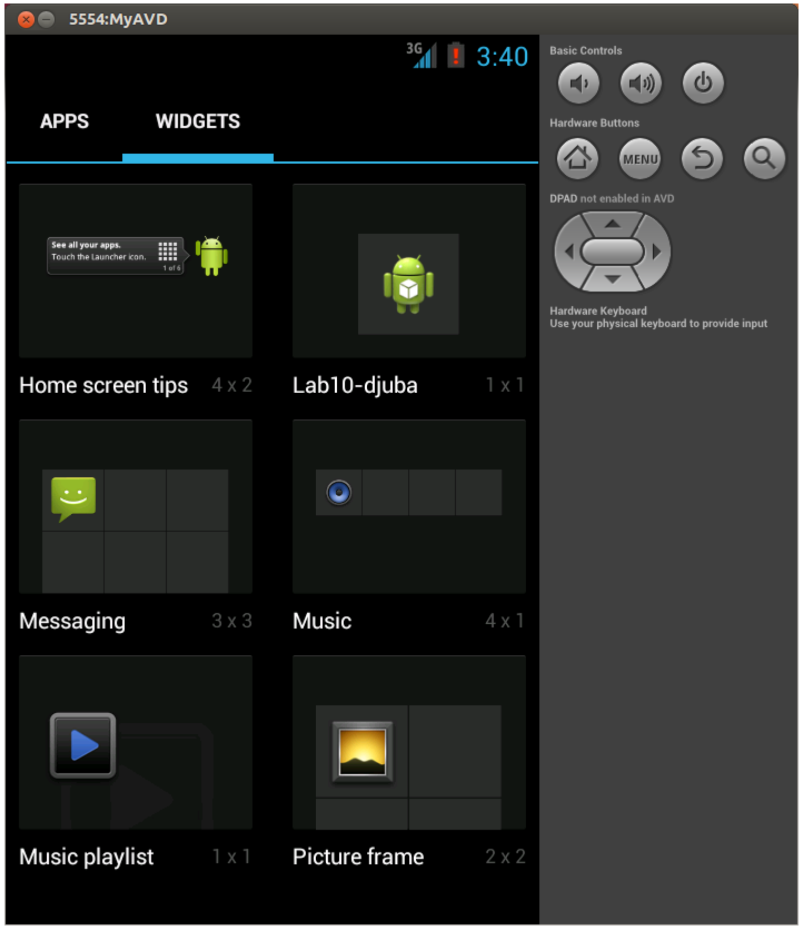
App Widgets
and
Supporting Different Devices

Overview

- For this lab you will create an App Widget that uses a Configuration Activity
- You will also localize the widget to support different languages and have it display the current SDK version
- The information needed to do this lab can be found on the Android developer site at
  http://developer.android.com/guide/topics/appwidgets/
  http://developer.android.com/training/basics/supporting-devices/languages.html
  http://developer.android.com/training/basics/supporting-devices/platforms.html

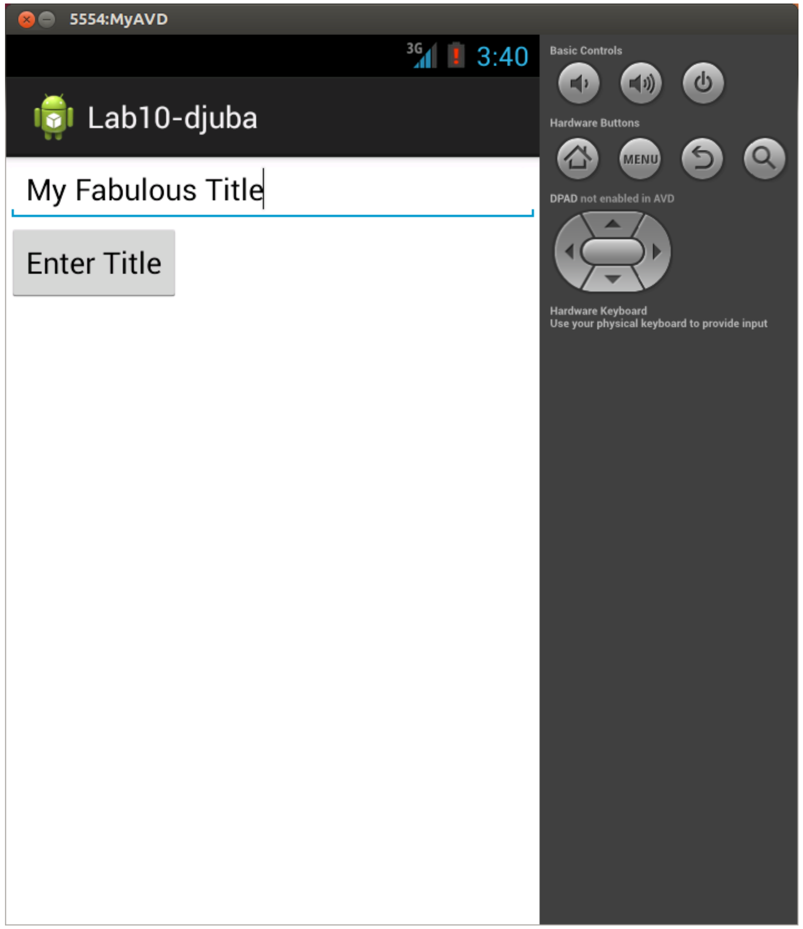
Interface

- The Lab10 widget should appear in the device's list of widgets
- Long-clicking on the widget should allow you to add it to the home screen



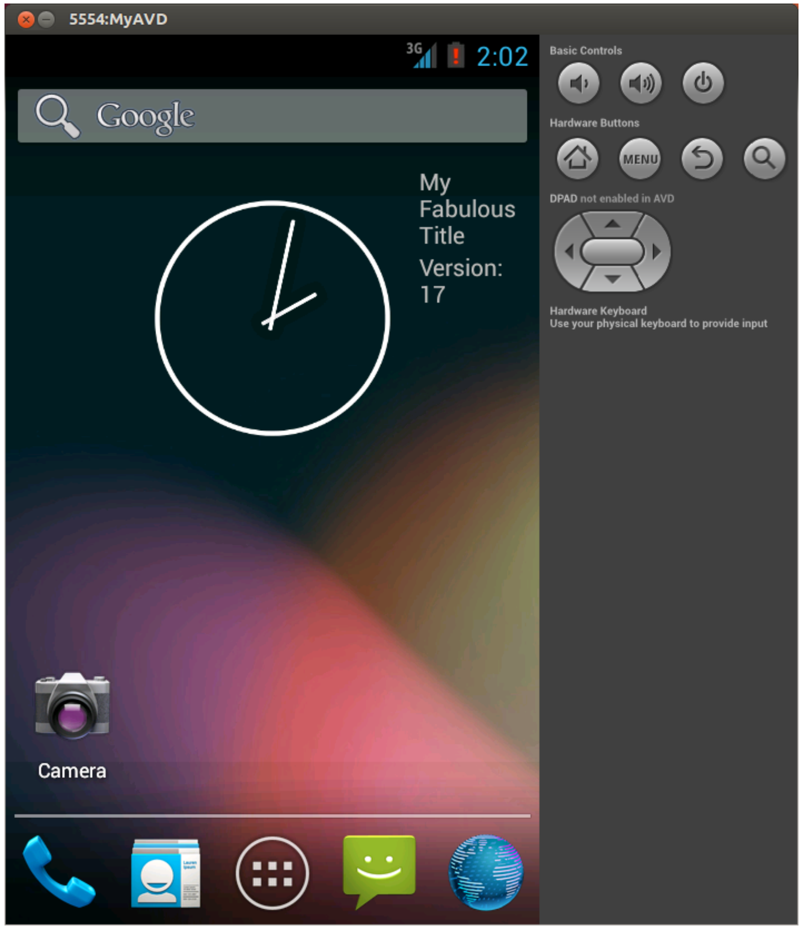
Interface

- When the widget is placed, the configuration Activity should be launched
- You should be able to enter a title in an EditText and finish the Activity by pushing the “Enter Title” Button



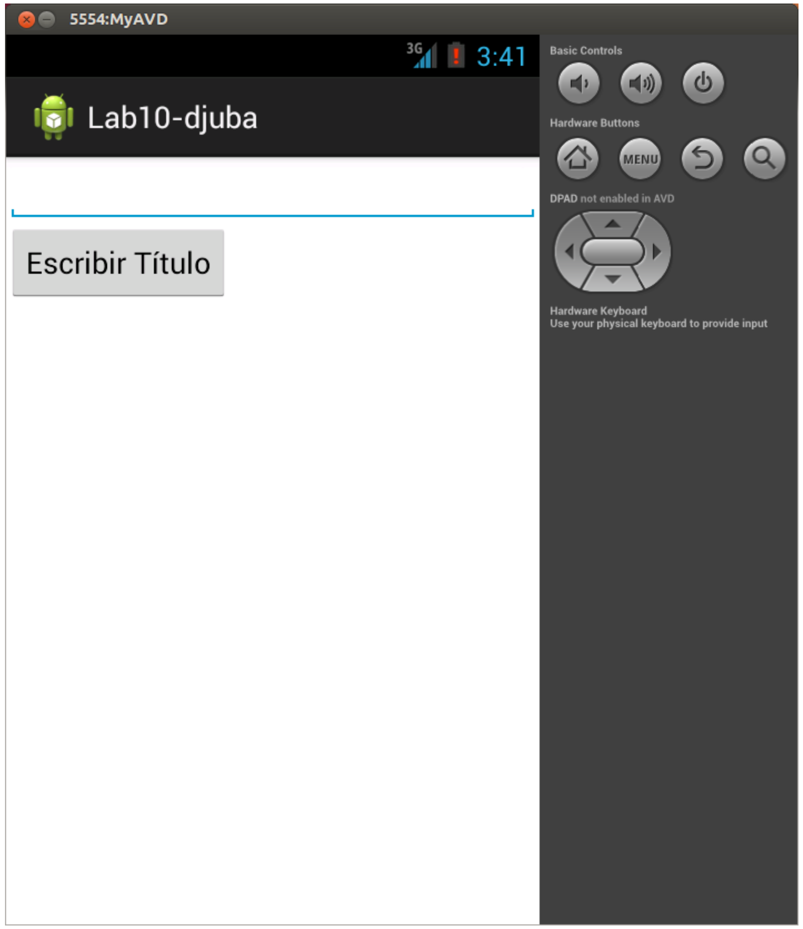
Interface

- The widget should have a TextView showing the title you entered and another TextView showing the current SDK version integer



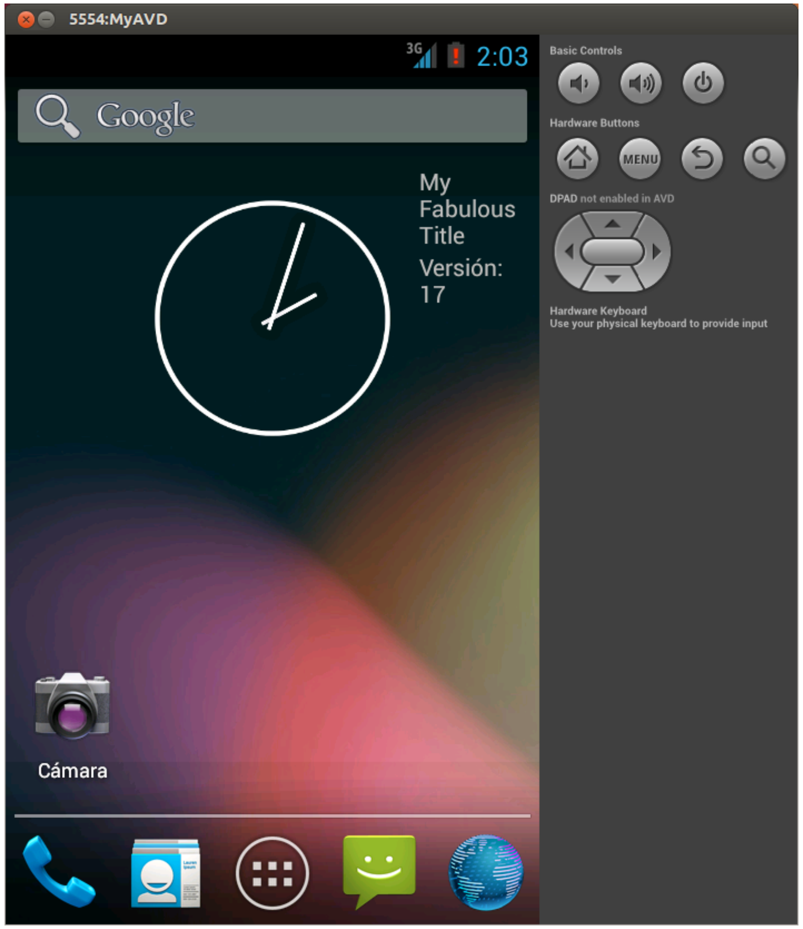
Interface

- If the locale is changed to Spanish then the text on the enter button should be appropriately adjusted



Interface

- The version text should also be adjusted for the Spanish locale



Lab Name

- To make our grading easier, you should name the app Lab10-<directory_id>, for example Lab10-djuba
- You can enter the name when you create the project in Eclipse, or change it later by editing the “app_name” string in strings.xml

App Widget

- The Android developer site contains the full documentation on how to make an App Widget that uses a Configuration Activity
- In the following slides I will summarize the basic steps

App Widget

- Create the App Widget Provider class
  - You can think of this as the actual widget
  - This should extend AppWidgetProvider, which extends BroadcastReceiver- so, it is not an Activity
- Create an xml layout for the widget
- Register the App Widget Provider as a <receiver> in the AndroidManifest.xml
- Create the AppWidgetProviderInfo metadata xml file in res/xml

App Widget

- Create the Configuration Activity
  - Since you don't want to be able to launch this Activity directly, it should not have the intent filter with action MAIN and category LAUNCHER in the manifest that is added by default- it will instead need an intent filter with action APPWIDGET_CONFIGURE
- Create an xml layout for the Configuration Activity

App Widget

- In the AppWidgetProviderInfo you should use the example settings from the Android developer site with the following changes
  - minWidth and minHeight should both be 36dp
  - There should be no previewImage
  - widgetCategory should be home_screen
  - There should be no initialKeyguardLayout
  - File and class names should match what you use

App Widget

- Since this lab does not include a main Activity, no Activity will launch when you run the project in Eclipse- this will instead simply install the apk on the device
- You will then have to add the widget manually to the home screen

App Widget

- Because the App Widget Provider is not an Activity, you cannot inflate the layout xml with setContentView or look up Views using findViewById as you normally would (both of these are methods of the Activity class)
- Instead, you generate a RemoteViews object containing the layout, which is then sent to the widget
- To access and modify Views in the RemoteViews object you can use methods such as setTextViewText

SDK Version

- It is sometimes necessary to create different code paths for different versions of the Android SDK
- You may see an example of this in the upcoming lab on Network Operations
- You can get an int representing the current SDK version by calling a static method of the Build class

Localization

- The res directory can contain different versions of resources to use in different situations
  - In lab 3 you created different layouts to use in portrait and landscape orientation in res/layout and res/layout-land
  - In lab 9 you added an icon to res/drawable which was used for all resolutions- versions for specific resolutions could have been included in res/drawable-hdpi, etc.

Localization

- For this lab you will create an alternate set of strings to use if the locale is set to “es”
- The default set of strings (in English) should be stored in res/values/strings.xml, while Spanish strings should be in res/values-es/strings.xml
- For the text on the Button you should specify the string to use in the xml layout
- For the version text in the TextView you should get the correct string programatically

Localization

- I got the Spanish translations from Google Translate- if you are actually fluent in Spanish, feel free to use a corrected version
- One way to change the locale is to launch the Custom Locale app
- You do not need to support the situation where the locale changes while your widget is running