CMPT 381 Assignment 1: Widgets and Events in Android

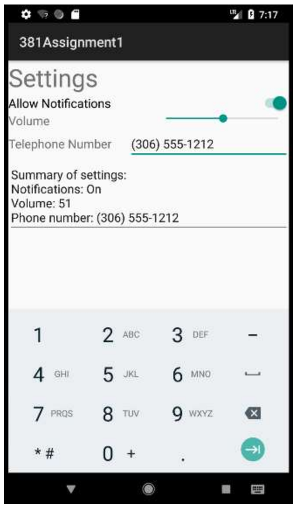
Due: Friday, January 18, 11:55pm

Overview

In Part 1, you will build a basic interface with Android widgets, and will write event handlers to deal with user interaction on those widgets. In Part 2, you will build a custom widget class for a "range slider" that can be used in your program. This assignment will demonstrate your ability to use widgets and widget APIs, to handle basic user-interface events at the application level, and to implement basic widget controls in a custom class.

Part 1: An interface with basic widgets and events

NOTE: when the Android documentation talks about "widgets" it often means "app widgets" (the small controls that users can put on their home screens). This is not the kind of widget that is involved in this assignment – in 381, we will use "widget" to refer to a visual interaction component in a UI toolkit.



Summary of settings:
Notifications: On
Volume: 51
Phone number: (306) 555-1212

Build a simple interface with the Android widgets TextView, EditText, Switch button, and SeekBar (see picture at right). Your system simulates a Settings page for a hypothetical app, and should include:

- A TextView at the top of the screen with the text "Settings" in large font.
- A Switch button with the text "Allow Notifications" that can be turned on or off.
- A TextView (with text "Volume") and a SeekBar that allows the user to set a value in the range 0-100.
- A TextView (with text "Telephone Number") and an EditText that allows the user to enter a phone number.
- A multiline TextView that displays a summary of the settings, as shown in the picture. The summary is updated immediately whenever the Switch, SeekBar, or EditText are manipulated by the user.
- The rows of widgets in the interface should be organized vertically using a LinearLayout container (with orientation set to vertical); when a row involves more than one widget, use another LinearLayout container.
- You can use the Android interface designer to build your UI (or you can build it in code).

Handling events for the widgets in the UI:

- Android widgets typically use Listener interfaces to receive events. You will need to listen for changes to the Switch, the SeekBar, and the phone number's EditText.
- For the SeekBar, use an OnSeekBarChangeListener.
- For the Switch, use an setOnCheckedChangeListener.
- For the EditText, use a TextChangedListener.
- How you wire together the widgets and the listeners is up to you (e.g., anonymous class, anonymous inner class, or implementing the listener interface in your MainActivity class)
- On any change, call a method that will create the summary string and put it into the summary TextView.

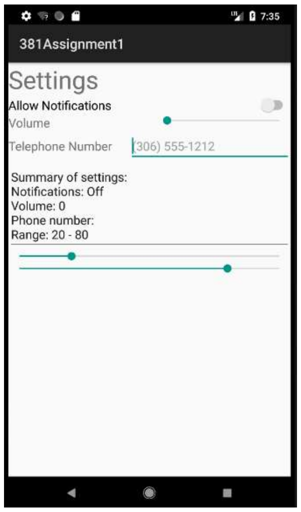
Resources:

- The API documentation for all of the Android classes listed above: https://developer.android.com/docs/
- Using Android Studio's visual editor: https://developer.android.com/studio/write/layout-editor

- Getting access to XML components from Java code: https://developer.android.com/guide/topics/ui/declaring-layout (see the example text "`Button myButton = (Button) findViewById(R.id.my_button);`")
- Android event handling overview: https://developer.android.com/guide/topics/ui/ui-events
- https://docs.oracle.com/javafx/2/ui_controls/slider.htm
- Tutorial for the JavaFX VBox: http://www.javafxtutorials.com/tutorials/hboxvbox/
- Tutorial for JavaFX event handling: https://docs.oracle.com/javafx/2/events/jfxpub-events.htm (note that this tutorial does not cover events for Slider widgets)

Part 2: Adding a custom widget

Build a custom widget called a RangeSeek, which contains two SeekBar widgets and uses them in combination to allows the user to set a range (e.g., 20-80 as in the picture at right). Add your RangeSeek widget to the interface you built for Part 1, and update the event handling so that the range from the RangeSeek widget is shown in the summary.



Settings
Allow Notifications
Volume
Telephone Number (306) 555-1212

Summary of settings:
Notifications: Off
Volume: 0
Phone number:
Range: 20 - 80

Your implementation should meet the following requirements:

- ***Appearance***. The widget shows two SeekBar widgets stacked vertically. The left handle of the widget is on the top, and the right handle is on the bottom.
- RangeSeek widgets always use a scale of 0-100, and always use MATCH_PARENT for their horizontal layout parameter, and WRAP_CONTENT for the vertical parameter.
- ***Interaction***. The user can click and drag the handles of the two SeekBar widgets to set the left and right ends of the range.
- The left handle cannot pass the right handle, and the right cannot pass the left.
- ***Creation of the widget***. RangeSeek's constructor should take four values:
  - A Context (for passing on in constructors of the superclass and the individual SeekBar widgets)
  - The int value of the initial left handle position
  - The int value of the initial right handle position
  - A MainActivity object for communicating event callbacks back to the parent
- ***Implementation***:
  - RangeSeek should extend class LinearLayout
  - You should create two SeekBar objects and add event handlers to listen for changes (within the RangeSeek class)
  - Whenever a change occurs, you should directly call method OnRangeSeekChanged() with the new left and right values. (You will have to write this method in your MainActivity class)
  - You may wish to add getLeftValue() and getRightValue() methods to the RangeSeek class.
- ***Demo program***. Add two RangeSeek widgets to the interface developed in part 1 (see example above).
  - NOTE: you may not simply build the functionality of a RangeSeek widget into your basic interface – you must create RangeSeek objects and add them to your layout container. If you do not build the RangeSeek as a widget class, you will not receive any marks for Part 2.
  - Whenever the range changes, immediately update the text summary in the main interface.

This assignment is to be done individually; each student will hand in an assignment.

What to hand in

- Android: a zip file of your Android Studio project folder. Note that you do not have to provide separate projects for part 1 and part 2 – you can just hand in the complete system built for part 2.
- A readme.txt file that indicates exactly what the marker needs to do to run your code. (Systems for 381 should never require the marker to install external libraries).

Where to hand in

Hand in your two files (one zip and one readme.txt) to the link on the course Moodle.

Evaluation

Marks will be given for producing a system that meets the requirements above, and compiles and runs without errors. Note that no late assignments will be allowed, and no extensions will be given, without medical reasons.