Engineering Design for Embedded Systems: Assignment 2 (version 1)

Patrick Lam

Due Date: January 21, 2013, 11:59PM

Modification in v1. Please submit the `R.java` file along with your `MainActivity.java` file. This will let us compile your code.

Task. You are to write a timer Android app which takes two inputs: a delay n, in seconds, and a count c of the number of times to beep. Every n seconds, your app must cause the phone to beep, and the phone should beep c times.

Deliverables. Your solution for assignment 2 should look something like Figure 1. We will only mark assignments submitted here:

`https://ecesvn.uwaterloo.ca/courses/ece155/w13/students/[your-username]/a2`

Please hand in your `MainActivity.java` file, the `R.java` file in the `gen` subjectory, and the `A2.apk` file that results from "Android Tools > Export Unsigned Application Package." The TAs will (generously) mark solutions based on code quality and functionality.

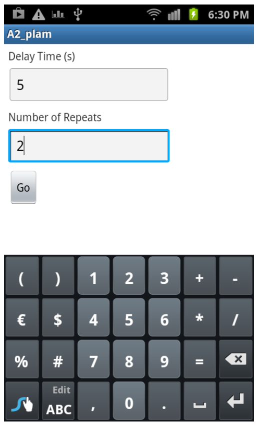

Figure 1: A complete assignment 2 implementation.

Assignments in this course are individual. You should be able to complete the assignment using just the Android emulator. **Tip:** Consult Lab 1 for more about Android programming.

Goals. After you complete this assignment, you'll be able to implement key features in basic Android applications, namely user-interface widgets. You will also have experience using `Handler`s and `Runnable` objects to schedule tasks to run in the future.

Creating the Widgets

The first step is to create the appropriate widgets to appear in your main Activity. Although it's fairly straightforward, we will show you how to do so in this week's tutorials.

Your solution should contain 5 widgets: 2 labels of type `TextView`, 2 numeric edit boxes of type `EditText`, and a "go" button of type `Button`. The `inputType` for the `EditText` boxes is `numberDecimal`. Create `@string` resources for the `TextView` and `Button` texts.

Respond to Button Clicks

We've talked about event handlers in lecture. In this assignment, you will need to add code to `onCreate()` which:

- retrieves the `Button` that you created using `findViewById`; and
- calls its `setOnClickListener()` method with an anonymous inner class. This class should contain the method `onClick(View v)`.

As a first step, put some logging code inside `onClick()`, such as `Log.d("a2", "click");` and test your code. Remove it afterwards.

Read Widget Contents

The next step is to read off the contents of the edit boxes. The following code will retrieve the contents of a text box whose id is `xyzzy`:

```
TextView tv = (TextView) findViewById(R.id.xyzzy);
String s = tv.getText().toString();
```

We'll use similar code in Lab 1 to replace the contents of a label. In this case, however, you want to read off the contents of the two edit boxes in the `onClick()` handler. You also want to convert the contents of the text box from `String`s to some more useful type. Log the converted contents to make sure that you're reading the contents properly.

Implement Alarms

Finally, you need to create a `Runnable` which counts down the number of alarms remaining and schedules an alarm for the appropriate time. Here's some hints:

- I recommend creating a new method `initializeAlarm()` which does the remainder of the work.
- This method should have a `final` local variable which contains the number of milliseconds to delay.
- You also need to create an anonymous inner class of type `Runnable` which implements the method `public void run()`. This method should decrement the number of remaining alarms, beep, and schedule the next alarm.

Here's some code to beep the phone:

```
final ToneGenerator tg = new ToneGenerator(AudioManager.STREAM_NOTIFICATION, 100);
tg.startTone(ToneGenerator.TONE_PROP_BEEP);
```

Recall that the lecture notes and this week's tutorial both explain how to schedule tasks for the future.

Error Handling

We may test your code on some invalid inputs. Your app should handle these cases appropriately: it should not loop infinitely, nor should it throw exceptions. It is OK to silently ignore error conditions.

Marking Scheme

We'll mark your submissions out of 10. Please be sure that your `apk` file matches the `MainActivity.java` file that you submit.

- Your code must compile; non-compiling submissions will get at most 3/10.
- When manually inspecting the code, we are looking for: 1) logical layout of the code, including good method boundaries and a reasonably-clear algorithm; 2) appropriate variable names.

(Note that I've allocated 0 points for comments. To be honest, there is nothing that needs commenting in this assignment, and making you write comments for this code is unrealistic and bad practice.)

Have fun!