Exercises:
Android Programming Basics

If you can do problems 1-4, you are in good shape. The last two problems preview features that we will explain in more detail in the next lecture on widget event handling.

1. Make an app that contains two Buttons (with labels "Push Me" and "Click Me") and a TextView (with text "This is a Test"). Use the XML-based approach, and you can hard-code the label of the buttons and the text of the TextView inside main.xml (i.e., you do not need to use strings.xml at all yet). Use the android:text attribute in both cases, but it is easiest to use the visual editor first, then edit main.xml later. Nothing needs to happen when you press the buttons.

2. Test your app on the emulator.

3. If you have an Android phone or tablet, test your app on it.

4. Update your app so that the Button labels and TextView text are taken from strings.xml.

5. Give your buttons some behavior. Here are some options for the button behaviors:

- Make them pop up Toasts (copy the code from one of the SayHello apps from the lecture, which is in your Eclipse projects)
- Have them change the foreground color of the Button that was clicked. Choose at random among Color.RED, Color.BLUE, Color.YELLOW, etc. To change the color of the Button, call setTextColor on the Button that is passed to the event handler. However, note that although Button has a setTextColor method, View (the parent class of Button) does not. So, you have to cast the View to Button before calling setTextColor.
- Have them change the text of the Button to "I was clicked *n* times". Use two instance variables for the counts.

  Use either the pure-XML way or the hybrid way of assigning the event handler.

  Test on the emulator and, if you have one, a real phone or tablet.

6. Make a new app that is similar to the old one, but this time, when you press a button, it should change the color of the TextView instead of the color of the button that was pressed. You can assign the event handler either in XML or in Java, but you'll need to at least partially use the hybrid approach because you need an explicit Java reference to the TextView that was defined in the XML file.