CS 370 Lab 1: Android Basics

Create GitHub Account

1. If you do not have one already, create a GitHub account (https://github.com)
2. Once you have an account, see me to have it added to the class organization

Obtain Code

1. Ensure that the lab workstation is booted into OS X
2. Create a new folder on your desktop called Repositories
3. Open a terminal session (Applications -> Utilities -> Terminal)
    - This is just a terminal window! You **don't** need to log in to blue
4. Ensure there are no other default accounts in the OSX keychain
    a. Keychain management instructions: https://kb.wisc.edu/helpdesk/page.php?id=2197
    b. Search for any *github.com* entries and remove them
5. Using the command line, change directory to the Repositories folder you just created
6. Clone the repository located at: ***git@github.com:SSU-CS370-F18/Android-Lab-1***
7. Change directory to the Android-Lab-1 folder you just cloned
8. Branch the repository using a branch name of ***lastname+firstname+370H1***
9. Open Android Studio.
10. Open the Android project that you just cloned.

Title Part 2

1. On the left-hand side of the Android Studio instance, click the 'Project' tab and ensure that the 'Android' view is selected. This gives you proper file navigation layout for the lab.
2. Expand ***app - java - ssu.hollant.homework1***. You should see a file called *MainActivity.java*. Activities represent the code for the presentation layer in Android.
3. Expand ***app – res – layout***. You should see a file called *activity_main.xml*. Layout files represent the markup mechanism for Android apps, and are typically backed by an Activity.
4. Expand ***app – res – values***. You should see a file called *strings.xml*. In Android applications, instead of declaring a String literal, color, or numerical value in code or in markup, you can create a resource here that will be referenced elsewhere in the project.

5. Open the *acitivity_main.xml* file. Make sure you are in Text view mode, not Design. There are tabs at the bottom of the screen that will let you switch between modes.
6. Note that there are two elements in this layout file:
    a. **Linear Layout**: the root layout for the view that determines how child elements are arranged
    b. **TextView**: an element for displaying text
7. Note that the LinearLayout has an id value: *android:id="@+id/activity_main"*. These id values are used to identify an element when referencing it from an Activity.
8. Note the TextView's text value is currently "Hello World!". This is the value that will be shown when the TextView is displayed on the screen.
9. Switch to Design view. Notice there is a string (*Hello World!*) displayed on the view
10. Switch back to Text editor mode.
11. Inside of the first LinearLayout but before the TextView, add another LinearLayout element (make sure it has an opening and closing tag). It will prompt you to set the *android:layout_width* and *android_layout_height*; set them to *match_parent* and *wrap_content* respectively. There should now be a root LinearLayout with two sibling elements, a LinearLayout and a TextView
12. Highlight and cut the entire TextView element. Paste it between the inner LinearLayout's opening and closing tags. The TextView is now a child element of the inner LinearLayout.
13. Add an *android:id* value of "@+id/inner_layout" to the inner LinearLayout.
14. Add an *android:orientation* value of *"horizontal"* to the *inner_layout* element
    a. Also give the outer LinearLayout a *"vertical"* orientation
15. Add another TextView element as a child of the *inner_layout* and below the original TextView. Set its *layout_width* and *layout_height* values to *wrap_content*. Provide it with an *android:id* of "@+id/name_text". Do not provide it an *android:text* attribute – we will set its text elsewhere.
16. Below the *inner_layout* LinearLayout, add a Button element with an *android:id* value of "@+id/name_button" and *layout_width* and *layout_height* values of *wrap_content*. The *activity_main* root LinearLayout now has two direct children: *inner_layout* and *name_button*
17. Open the strings.xml file. Notice that an XML string element is already defined with the name "app_name" and a value of "Homework1"

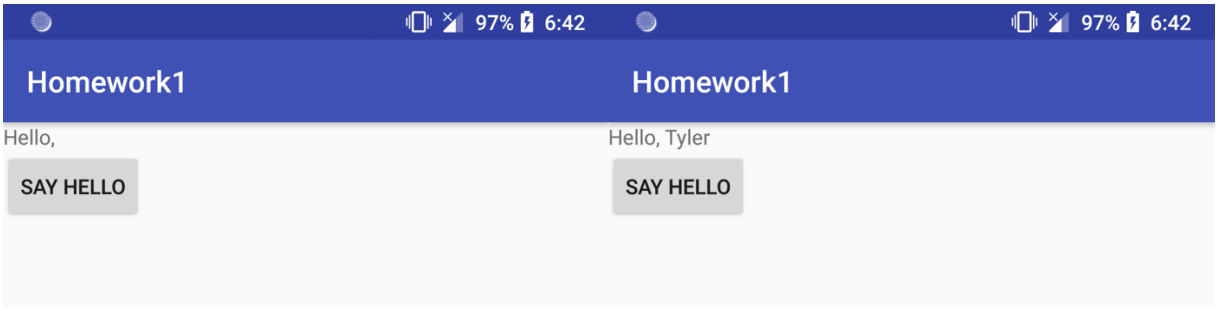
18. Add a new string element named “hello_text”. Add the value “Hello, ” (without quotes).
19. Add another new string element named “name_text”. Add your name as the value.
20. Add another new string element named “button_text”. Add the value “Say Hello”
21. Move back to the *activity_main* file. Add an *android:text* attribute to the Button element and set the value to “@string/button_text”. The button will now get its text from the strings.xml resource file.
22. Do the same for the first TextView child of *inner_layout*, but assign it the *hello_text* string resource value.
23. Switch to Design view and verify that the string values for the TextView and Button match your updates
24. Open the MainActivity.java file. Inside the class declaration, add two private variables:
    a. private Button nameButton;
    b. private TextView nameText;

    They should go above the @Override annotation of the onCreate method.
25. Further down the class is a method/function named *onCreate*. Note that in this function the *setContentView* method is invoked to bind the *activity_main* layout file to the *MainActivity* code
26. Add a line break after *setContentView* and add the following lines to the bottom of the *onCreate* method:
    a. nameText = (TextView)findViewById(R.id.name_text)
    b. nameButton = (Button)findViewById(R.id.name_button)
27. Add another line break and add the following code block to the bottom of the *onCreate* method (Android Studio’s auto-complete should help you with this):

```
nameButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
                nameText.setText(R.string.name_text);
        }
});
```

This block of code references the Button in the layout file and instructs it to change the value of the TextView's text attribute as specified. This type of function is called an event handler. Note also that we are using a value from the strings.xml resource file.

28. Run the application using the Run button in the top toolbar of Android Studio. This should present you with the option to "Launch Emulator". Select it and click OK. An Android emulator should pop up (eventually) and your app will run.
29. Click the button and see if your changes worked!



30. If your app is running correctly, the app should replace the blank space after "Hello, " with your own name. If not, revisit the above steps and make sure you've followed the instructions.
31. When your app runs properly, execute the following commands to upload your changes to your branch on GitHub:
    a. git add .
    b. git commit -m "working code complete"
    c. git push origin yourbranchname

    Your code branch is now saved and committed to the git repository.

This completes Lab 1.