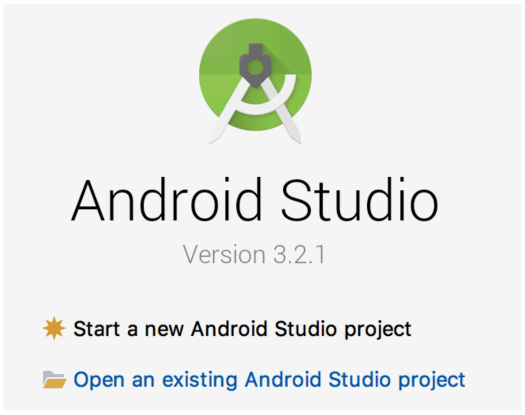
CS 370 Lab 1: Android Basics

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

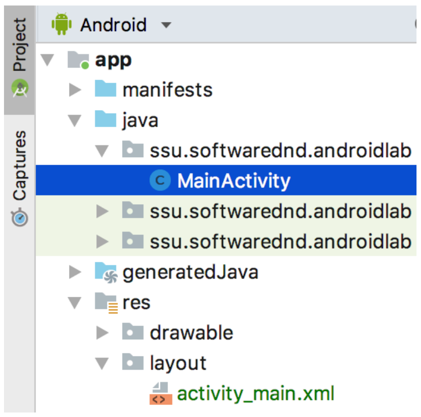
Obtain Code

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 on your local machine! ***Do not 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: **git clone https://github.com/ssu-cs370-s19/AndroidLab1.git**
7. Change directory to the AndroidLab1 folder you just cloned
8. **Create a new branch** in the git repository called **LastnameFirstname**
9. Open Android Studio.
10. Open the Android project that you just cloned.


Android Studio
Version 3.2.1
Start a new Android Studio project
Open an existing Android Studio project

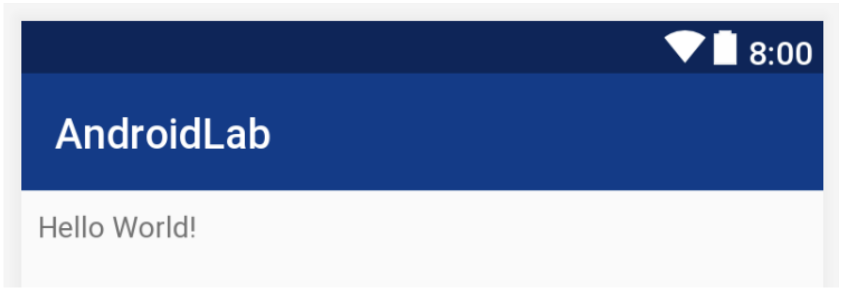
Android Basics

1. On the top left-hand side of the Android Studio window, open the 'Project' side tab and ensure that the 'Android' option is selected from the drop-down menu. This will make it easy to find the relevant code and resource files.



2. Explore the file tree. You should see:
   a. *MainActivity.java* (contains all Java code)
   b. *activity_main.xml* (the layout for main activity)
   c. a *values* folder (reference files for other parts of the app – Strings, colors, ints, etc.)

3. Open activity_main.xml, and switch to Text mode, not Design (tabs at bottom of the panel).
4. Note that there are two elements in this layout file:
   a. **LinearLayout**: root layout for the view. determines how child elements are arranged
   b. **TextView**: an element for displaying text
5. Look at the structure of the **LinearLayout**:
   a. `<LinearLayout` : the opening tag. several attributes come next:
   b. `android:id="@+id/activity_main_layout"`: id values are used to identify an element when referencing it from an Activity.
   c. `android:orientation="vertical"`: orientation controls which direction **LinearLayout** places the elements it contains. vertical is down the page, horizontal across.
   d. `</LinearLayout>` : the closing tag. marks the end of the **LinearLayout**
6. Note that the **TextView** has a text attribute: `android:text="Hello World!"`
   This is the value that will be shown when the **TextView** is displayed on the screen.
7. Switch to Design mode (tab at the bottom). You should see text (*Hello World!*) displayed.



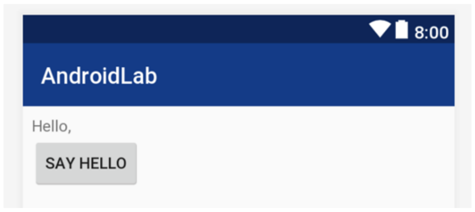
8. Switch back to Text editor mode.

9. 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 width to match_parent and height to wrap_content. There should now be a root **LinearLayout** with two sibling elements: an inner **LinearLayout**, and a **TextView**.
10. Add an android:id value of "@+id/inner_layout" to the inner **LinearLayout**.
11. Add an android:orientation value of horizontal to the *inner_layout* **LinearLayout**.

12. Move the **TextView** element into the inner **LinearLayout**. (cut/paste, or highlight and drag). It will go between the inner **LinearLayout**'s opening and closing tags.
    The **TextView** is now a child element of the inner **LinearLayout**.
13. Add another **TextView** element as a child of the *inner_layout* and below the original **TextView**. Set its layout_width and layout_height values both to wrap_content. Provide it with an android:id of "@+id/name_text". Do not provide it an android:text attribute – we will set this text value from code instead.
14. Outside and below the *inner_layout* **LinearLayout**, add a new **Button** element with an android:id value of "@+id/name_button" and layout_width and layout_height values of wrap_content. The *activity_main_layout* LinearLayout now has two children: *inner_layout* and *name_button*.

15. Open the strings.xml file (res/values/strings.xml). Notice that a string element is already defined here, with the name "app_name" and value "AndroidLab1".
    ```
    <string name="app_name">AndroidLab1</string>
    ```
16. Add a new string element named "hello_text". Add the value "Hello, ".
    a. The quotes in this one are necessary to keep the extra space at the end of the string
17. Add a new string element named "name_text". Add *yourname* as the value.
18. Add a new string element named "button_text". Add the value Say Hello!
19. Add an android:text attribute to the **Button** element and set value to "@string/button_text". The button will get its text from the string resource we added to strings.xml.
20. Change the first **TextView**'s android:text value from "Hello World!" to @string/hello_text.

21. Switch to Design view and verify that the appearance of the TextView and Button match your updates. Your name should not appear on the screen yet.



22. Open MainActivity.java. Inside the class declaration, add two private member variables:

```
private Button nameButton;
private TextView nameText;
```

They should go above onCreate, and below the class declaration.

23. Now look at the method named onCreate. Remember that the setContentView method is used to bind the *activity_main* layout file to the *MainActivity* code
24. After setContentView, add the following lines to the onCreate method:

```
nameText = findViewById(R.id.name_text);
nameButton = findViewById(R.id.name_button);
```

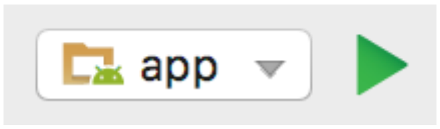
These get the TextView and Button from your layout and assigns it to your Java variables.

25. 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):

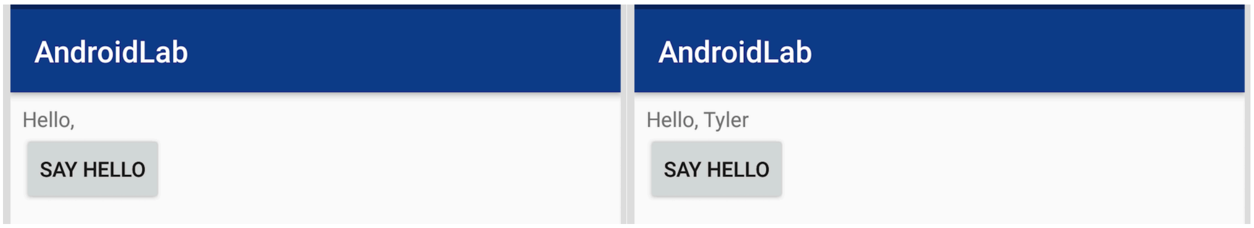
```
nameButton.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      nameText.setText(R.string.name_text);
   }
});
```

This block of code references the **Button** in the layout file and instructs it to change the value of the **TextView**'s text when the Button is ~~clicked on~~ tapped.
This type of function is called an event handler.
Notice also that we are using a value from the strings.xml resource file – although we could also have used a string literal here.

26. Run the application using the green Run button in the top toolbar of Android Studio. This should present you with a Launch Emulator dialog. Select one and click OK. An Android emulator should pop up (eventually) and your app will then install and open.



27. Click the button and see if your changes worked!



28. 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 "lab 1: working code complete"
    c. git push origin *yourbranchname*

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

This completes Lab 1.