CS385 Spring 2017 Homework 1

Lab 1.1 Obtaining Code

1. Ensure that the lab workstation is booted into OS X

2. Create a new directory on your desktop called Repositories.

3. Open a terminal session (Applications->Utilities->Terminal).

4. If you do not have a GitHub account, create one here: **https://github.com**

5. Ensure that there are no other default accounts entered into the OS X keychaing (keychain management instructions here: https://kb.wisc.edu/helpdesk/page.php?id=2197 . Search for the github.com entries and remove them).

6. Using the command line, change directory to the Repositories directory that you've just created.

7. Clone the repository located at https://github.com/SSU-CS370/Homework1-370.git

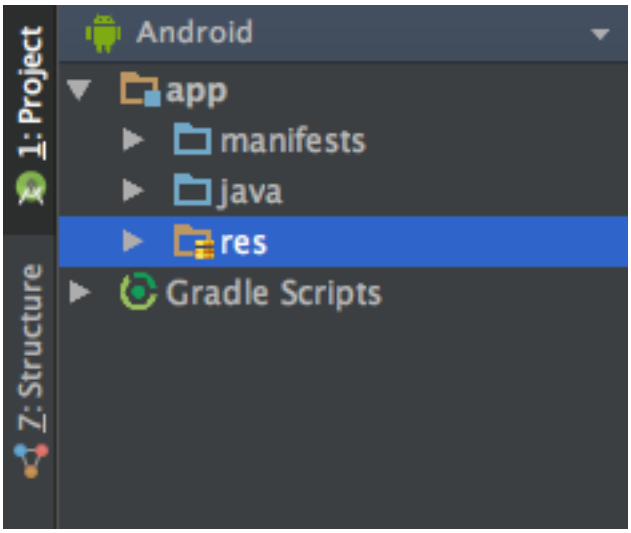
8. Branch the repository using a branch name of '**last-name + first-name + 370H1'**

9. Open Android Studio.

10. Open the Android project that you just cloned.

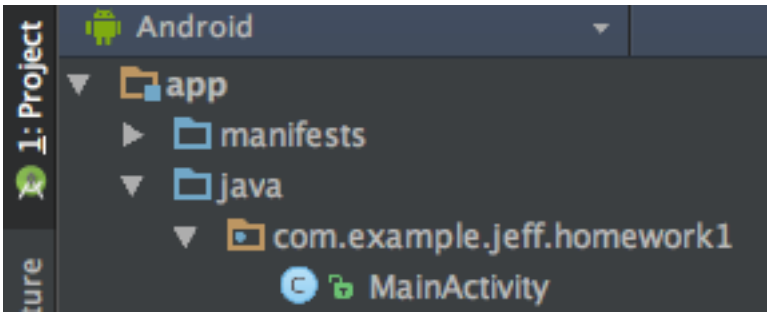
Lab 1.2: Android Presentation Layer Binding

1. On the left hand side of the Android Studio Instance, click the 'Project' tab and ensure that the 'Android' view is selected. This will present you with the proper file navigation layout for the lab.

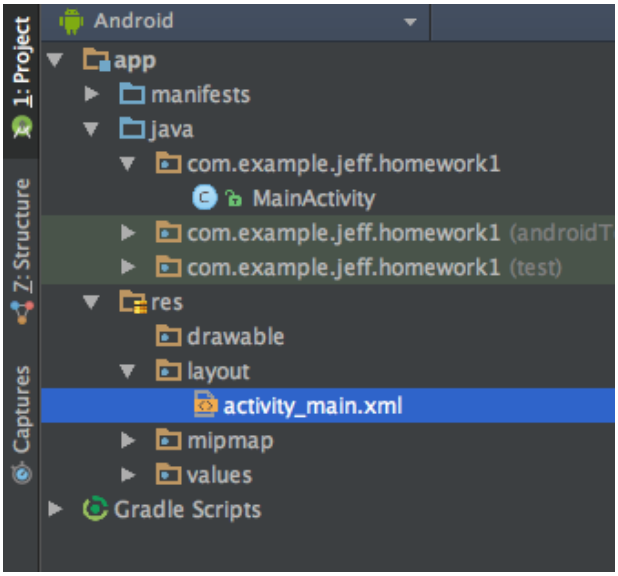


http://www.bocasteak.comhttp://www.bocasteak.com

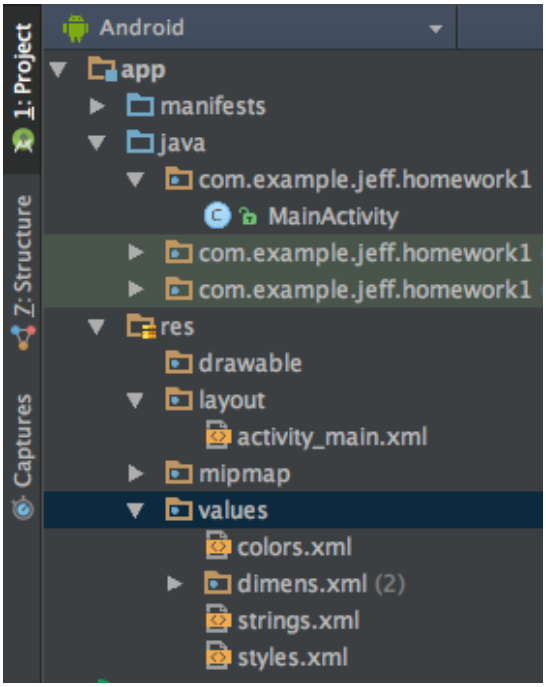
2. Expand app->com.example.jeff.homework1. You should see a file called MainActivity. Recall that Activities represent the code for the presentation layer in Android.



3. Expand app->res->layout. You should see a file called activity_main.xml. Recall that Layout.xml files represent the markup mechanism for Android apps, and that they are typically backed by an Activity.



4. Expand app->res->values. You should see a file called strings.xml. Recall that in Android applications, instead of declaring a String literal, color or numerical value in code or in markup, you can create a resource that may be referenced elsewhere in the project.



5. Open the activity_main.xml file. Ensure that you are in Text view mode and not Design view mode. There are tabs at the bottom of the file window that allow you to switch between these views.

6. Note that there are two elements in this layout file:
    1. **LinearLayout** (the root layout for the view that determines how child elements are arranged)
    2. **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 the element when referencing it from an Activity.

8. Also note that **TextView** text value is currently "Hello World!". This indicates that the value that will be displayed for that **TextView** is the string literal value shown.

9. Switch to Design view. Observe that there is a string ("Hello World! ") displayed on the view.

10. Switch back to text editor view.

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 the *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 **LinearLayout's** opening and closing tag. The **TextView** is now a child of the child **LinearLayout**.

13. Add an android:id value of "@+id/inner_layout" to the child **LinearLayout**.

14. Add an android:orientation value of "horizontal" to the **inner_layout** element

15. Add another **TextView** element as a child of the **inner_layout** and below the original **TextView**. And set it's *layout_width* and *Layout_height* values to "wrap_content". Provide it with an android:id of "@+id/name_text".

16. Below the **inner_layout LinearLayout**, add a **Button** element with an android:id of "@+id/name_button", and *layout_width* and *layout_height* values of "wrap_content". The **activity_main** root **LinearLayout** should now have two sibling children, **inner_layout** and **name_button**.

17. Open the strings.xml file. Note that an XML string element has been defined with the name **"app_name"** and a value of "Homework1"**.**

18. Add a new string element named **"hello_text"**. Add the value (without quotes) "Hello, ".

19. Add another new string element named "**name_text**". Add your name as the value.

20. Add one last string element named **"button_text".** Add the value (without quotes) "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 acquire it's 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 note that the string value for the **TextView** should reflect your updates.

24. Open the **MainActivity** file. Inside the class declaration, add two private variables:

    private Button nameButton;
    private TextView nameText;

    They should appear above the @Override for the onCreate method.

25. Farther down in the class is a method (another word for function) named '**onCreate'**. Note that, in this method, the **setContentView** method is invoked to bind the **activity_main** layout file to the **MainActivity** code.

26. Add a line break and then add the following two lines to the bottom of the '**onCreate'** method:

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

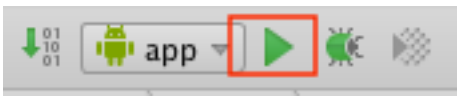
    These lines of code create a reference to the visual elements in the **activity_main** layout file.

27. Add another line break and add the following code block to the bottom of the **'onCreate'** method (type this in so that you can see how Android Studio auto-complete can help you):

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

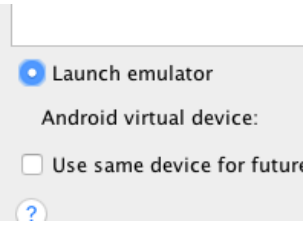
    This block of code reference the Button in the layout file and instructs it to change the value of the **TextView** text attribute as specified. This type of function is called an **event handler**. Note that we also use a value from the strings resource file to set the **TextView** value

28. Run the application using the 'Run' button at the top of the code view.



    This should present you with option to "Launch emulator". Select this option and click OK. Your app should run (eventually…the emulator is a bit slow). Once the app is

running, click the Button to see if your changes work.



29. If your app is running correctly, the app should replace the blank space after “Hello, “ with your name. If this is the case, open your terminal instance. From the Homework1 directory, execute the following commands sequentially:

    git add .
    git commit –m “working code complete”
    git push origin <your branch name>

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

    This will complete the homework.