CS385-FA-H1 Instructions

This assignment is intended to provide a brief primer into how layout files work. The instructions on how to start the homework and turn it in via GitHub will have been covered in class, but are repeated here for convenience.

Getting Started

1. Ensure that you are using your own credentials for GitHub on whatever workstation you are using to complete the assignment.

2. Using the terminal/cmd, clone the empty Homework 1 repository from GitHub (addresses for HTTPS or SSH provided below).

   https://github.com/SSU-CS385/FA-2017-H1.git
   git@github.com:SSU-CS385/FA-2017-H1.git

3. A new directory labeled 'FA-2017-H1' should have been created. Using the terminal/cmd, change your working directory to this new directory

4. **IMPORTANT** – Create a new branch (git checkout –b '<branchname>') where '<branchname>' is 'yourlastname385H1'

5. Open Android Studio and create a new project.
   i. In the 'Application name' field, input '<Yourlastname>385FAH1'
   ii. In the 'Company domain' field, input 'ssu385.fa17.com
   iii. In the 'Project location' field, use the file browser to select the 'FA-2017-H1' directory that was created when you cloned the repository.
   iv. Click 'Next'
   v. Leave the 'Phone and Tablet' option checked
   vi. Set the 'Minimum SDK' to 'API 24: Android 7.0 (Nougat)'
   vii. Click 'Next'
   viii. Select 'Empty Activity
   ix. Click 'Next'
   x. In the 'Activity Name' field, input 'StubActivity'
   xi. Leave the 'Generate Layout File' and 'Backwards Compatibility (AppCompat)' options checked
   xii. Click 'Finish'

6. Once the new project has been created, In the toolbar, open the 'VCS' menu.
   i. Click the option to 'Enable Version Control Integration'

ii. In the dialog that appears, select the 'Git' option and click 'Finish'. The project has now been associated with your new branch in the repository.

First Commit and Push

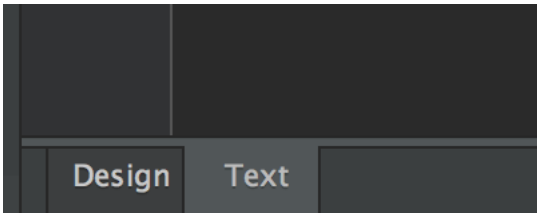
1. Now that new files have been added to the stage of the new branch, open the Terminal tab at the bottom of Android Studio.
2. This window provides you actual Terminal access and should show your project directory as the working directory (recall that your project directory is a sub-directory of the repository directory).
3. Type 'git status' in the terminal and execute. You should be presented with a number of file names highlighted in red. These are the files for your Android Studio project that have been added but not tracked.
4. Type 'git add .' in the terminal and execute to add all of these files to the tracking stage (meaning that they can be committed).
5. Type 'git status' in the terminal and execute again. Note that all of the file names are now green.
6. Type 'git commit –m "first commit" ' in the terminal and execute. All of your changes have now been saved on your new branch.
7. Type 'git push –u origin <branchname>' in the terminal and execute. Your changes have been pushed on your personal branch to the remote repository.

Layouts

1. Make sure that Android Studio has the 'Project' side tab active and that your browsing option is set to 'Android

2. In the directory structure, expand the ‘res’ package and then the ‘layout’ package beneath it. You should see a file named ‘activity_stub.xml’. This is the auto-generated layout file for the project.

3. Open the layout file. If you are not in ‘Design’ mode, switch to it using the tab at the bottom of the file display pane.



4. Note that the ‘Hello, World’ text is centered in the rendered layout.

5. Switch to the ‘Text’ view in the file display pane.

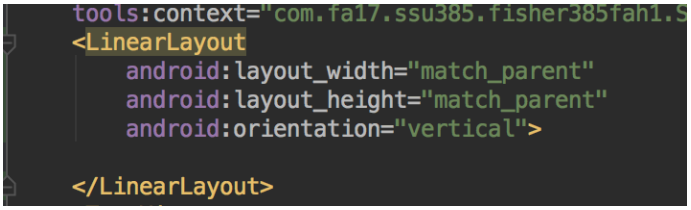
6. You should now see the contents of the layout file composed as XML. This should look familiar as we’ve briefly looked at this file in class. Note that the root layout for this file is a **ConstraintLayout**. We need to change this.
   i. Replace ‘android.support.constraint.ConstraintLayout’ with ‘FrameLayout’ taking care not to delete the xmlns declaration to the right.
   ii. Note that as we changed the Layout type in the opening tag, the closing tag changed to match. The root layout has now been changed to a **FrameLayout**.

7. Recall that the **FrameLayout** type is really only intended to have one child element. Currently, this directive is not being violated since there is only one child element (a **TextView**).

8. Above the **TextView** element, create a new **LinearLayout** element, with the following values:

   android:layout_width=”match_parent”
   android:layout_height=”match_parent”
   android:orientation=”vertical”

   It should look like the figure below:



9. Now that we have a second child element, we are violating the directive for **FrameLayout**. To resolve this, highlight the entirety of the **TextView** tag, cut it and then paste it between the open/close tags for the **LinearLayout**. The **FrameLayout** now has a single child again.

10. Switch to the 'Design' view in the file display pane. Note that the 'Hello, World' text is no aligned with the top left.

11. Switch back to the 'Text' view in the file display pane. Note that the **TextView** element has a variety of layout_constraint attributes defined which were intended to position it in the center of the rendered layout. However, now that we have removed it as a child of a **ConstraintLayout**, these attributes no longer apply. As a part of a vertically oriented **LinearLayout**, child elements are aligned in order, top to bottom.

12. In the layout file, add a new 'android.support.constraint.ConstraintLayout' child to the **LinearLayout** with the following attribute values:

    android:layout_width="match_parent"
    android:layout_height="match_parent"

13. Again, cut the **TextView** and paste it in as a child of the **ConstraintLayout**.

14. Switch back to the 'Design' view of the file display pane. Note that the 'Hello, World' text is once again centered. Now that the **TextView** is once again the child of a **ConstraintLayout**, its layout_constraint attributes once again apply.

15. Add a new **TextView** element as another child of the **LinearLayout**, after the **ConstraintLayout** closing tag, and assign it the following values:

    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello, Jeff"

    The **ConstraintLayout** and **TextView** are now siblings in the context of the parent layout.

16. Switch back to the 'Design' view of the file display pane. Note that the new **TextView** is not visible. Recall that we had set the layout_height of the **ConstraintLayout** sibling to a value of "match_parent". This instructs the **ConstraintLayout** to fill all available vertical space within the bounds of the **LinearLayout** parent, which leaves no room for the new **TextView** to be rendered.

17. Switch back to the 'Text' view of the file display pane. Change the layout_height value of the **ConstraintLayout** from "match_parent" to "wrap_content". If you switch back to the 'Text' option of the file display pane, you'll note that both **TextView** elements are both displayed, and that they are arranged in a linear manner with minimal space provided for either.

18. Add the following attribute and value to both the **ConstraintLayout** and its sibling **TextView**:

    android:layout_weight="1"

19. Switch to 'Design' view again. Note that the siblings now have equal amounts of vertical space allotted. The layout_weight attribute has directed the **LinearLayout** to render them according to this value. If the value of one of the siblings is set to "0", you will observe that this ratio of vertical space is altered so that the element with the larger value gets the majority of the vertical space.

20. Switch back to the 'Text' view. Add a **RelativeLayout** as a child of the **LinearLayout** with the following attributes:

    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"

21. Cut and paste the **ConstraintLayout** and it's **TextView** child so that it is a child of the new **RelatieLayout**.

22. Add a new **TextView** element as a child of the **RelativeLayout** with the following values:

    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello, Student"

23. Switching to the 'Design' view, note that the new element appears in line with its sibling. Switch back to the 'Text' view and add the following attribute to the **ConstraintLayout**:

    android:id="@+id/siblingConstraintLayout"

Also add the following attribute to the sibling **TextView**:

android:layout_below=”@id/siblingConstraintLayout”

24. Switch back to the ‘Design’ view. Note that the **TextView** sibling is now aligned below the **ConstraintLayout** sibling. Also note that, while have any number of sub-descendents of the **FrameLayout** it still technically has only one child.

    This completes Homework 1.

Turning Your Work In

1. In the Terminal window, type ‘git status’ and execute. You should see the files that you have modified highlighted in red (there should be only one, the ‘activity_stub.xml’).

2. In the Terminal window, type ‘git add .’ and execute.

3. In the Terminal window, type ‘git status’ and execute. The previously red files should now be green.

4. In the terminal window, type ‘git commit –m “completed homework” ‘ and execute.

5. In the terminal window, type ‘git push –u origin <branchname>’ and execute. Your work has now been saved on your branch in the remote repository.