Name: _____

ROSE-HULMAN INSTITUTE OF TECHNOLOGY **CSSE490 Android Application Development**

Lab 2: Layouts and Samples

The purpose of this lab is to learn about the different Layout classes and play around with some sample programs that come with Android. Layouts are subclasses of **View->ViewGroup** and end in the word Layout (Linear**Layout,** Relative**Layout**, etc). We'll have you work the tutorials at **developer.android.com** associated with Layouts.

_____ Part A: Reading

_____ Part B: Linear Layout

_____ Part C: Relative Layout

_____ Part D: WebView

_____ Part E: Frame Layout

_____ Part F: Run a Sample

In order to check off this lab, open your emulator or device and the instructor or assistant will check off each part. **Note:** To run them on the emulator or your phone you don't even have to have Eclipse open. That is the fastest (and preferred) way to do all checkoffs. It takes too long to load each one from Eclipse. Just launch each app so it has been recently run. Then do a long press on the home button to quickly jump between apps.

Part A. Reading

Read about Layouts here:

http://developer.android.com/guide/topics/ui/declaring-layout.html

This information is important when you are getting started with Android. Finish reading after you finish Common Layouts (we'll study adapters later). Answer these questions based on your reading. Please write your answers in another document (.docx, .pdf, or .txt are fine).

You load the layout resource (the xml) from your application code by calling _____, passing it the reference to your layout resource.

The plus sign in @+id/ means that this is _____

The previous question referred to **R.java**. Go to any app you've created and open up the gen (generated) folder to find R.java. Note that it has a number of nested classes containing constants. What is the value of R.layout.activity_main? _____

Name as many other (besides layouts) familiar elements you see in R.java.
_____

Note that sometimes if you have an error in your xml, it will prohibit R.java from being generated, which can cause all sorts of problems with the constants not being defined. So look for errors in the xml first if get such errors.

You can declare UI elements in XML or in code. There are clear advantages to creating them in XML (remember the MVC paradigm?). Think a bit and give an example when you should declare one in code:

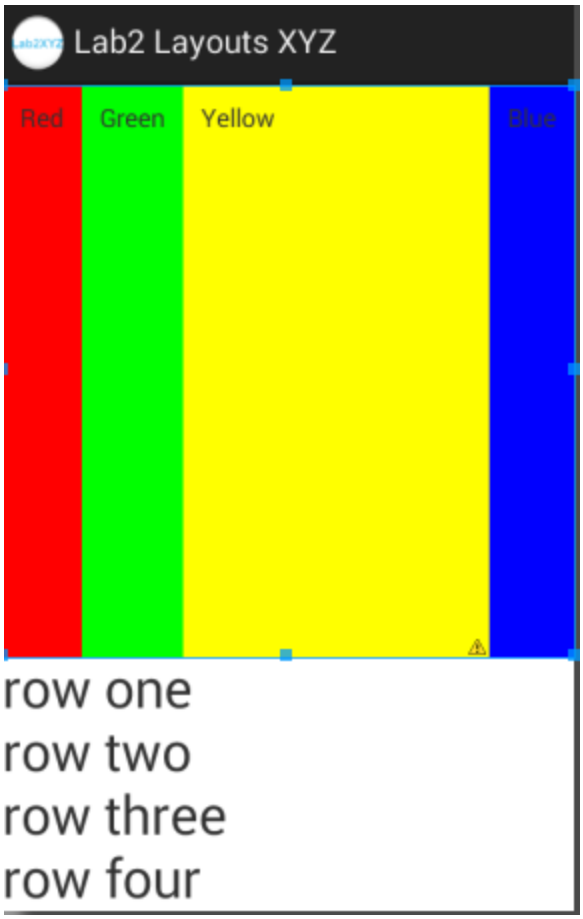
_____

We will have you do the exercises on the first three layouts under Common Layouts: **Linear Layout**, **RelativeLayout**, and **WebView**.

Part B. Linear Layout

Follow the link under Common layouts to read about LinearLayout. Pay particular attention to the layout_weight attribute.

Create a Lab2b (your initials) project. In it, modify activity_main.xml to make this layout:



Note that you will need to nest a horizontal layout inside a vertical one. Make colored columns have a weight of 1 within the vertical layout and the yellow column have a weight of 1 within the horizontal layout. The other columns can just wrap their content; no weight needed. Add a small amount of padding around the text to the vertical columns as well, as shown. Note that yellow is full red+green (so, #FF0).

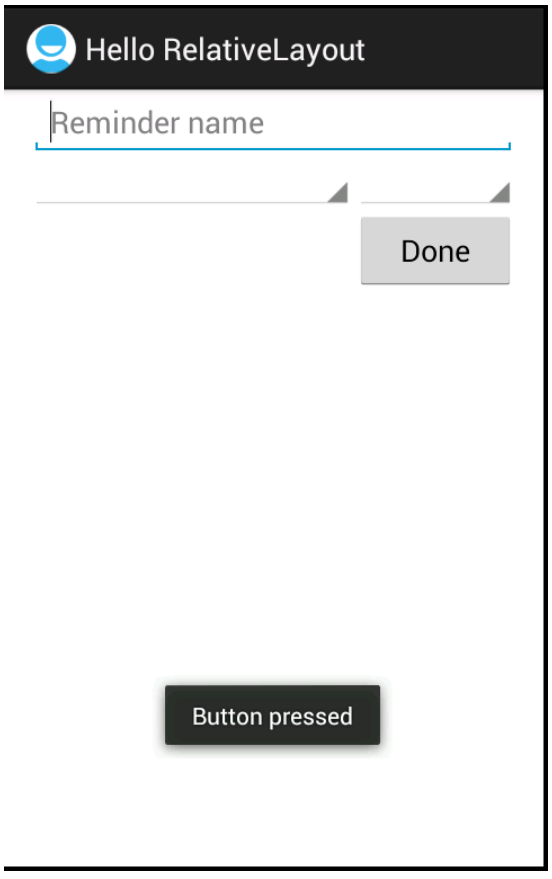
Part C. Relative Layout

Create a Lab2c (your initials) project. In it, modify activity_main.xml to make this layout.

We work with RelativeLayouts all the time, so don't need much more practice. Read about them and create the same layout they give as a demo. Feel free to copy and paste their .xml as long as you understand it. Don't worry about populating the spinners with dates and times; we'll learn how to do this later in the course. For this part, we also want you to display a message to the screen when the user presses the done button. You can create a quick and simple transient message by using a Toast notification:

http://developer.android.com/guide/topics/ui/notifiers/toasts.html

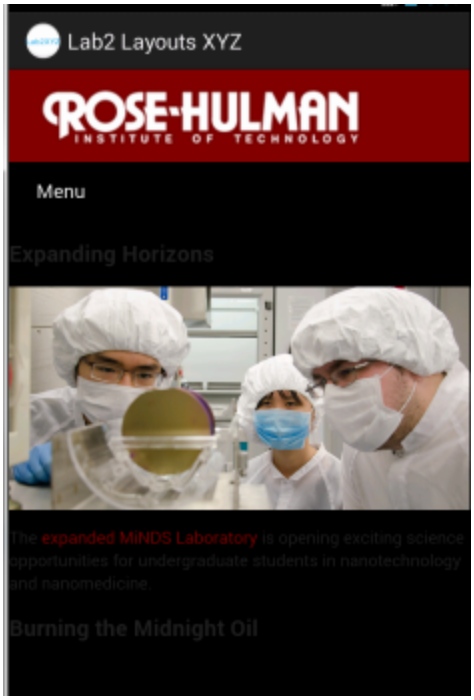
For example,
Toast.makeText(MainActivity.this, "Button pressed", Toast.LENGTH_LONG).show();



Part D. WebView

Create a Lab2d (your initials) project.

Create the WebView in the demo. In your java code, use the url website of your choice. No need to write additional code. You will find that many institutions' websites (like Rose-Hulman's and MIT's) aren't optimized for mobile devices.



Note this is your first time you've actually had to change the AndroidManifest file. You will need to do so many times throughout the course. (Also, this explains why if you didn't do enough of the WebView tutorial, you will get a Page Not Found error in the view. :))

Above is a screenshot from the emulator - note that it opens the page within the app. If you run your app on a device, it may open the web page in an external browser! For this tutorial, that is OK.

If you want to control this behavior, you can override **shouldOverrideUrlLoading()** in a custom WebClient. Here's a quick and dirty line of code that uses an anonymous client to force the app to always show its pages inside the WebView:

```
myWebView.setWebViewClient(new WebViewClient() {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
```

```
            return false;
        }
    });
```

If you want yet more control, keep reading the WebView tutorial page (http://developer.android.com/guide/webapps/webview.html). Details are at the bottom.

Part E: Frame Layout

Create a Lab2e (your initials) project. In it, modify activity_main.xml to make this layout

There is no HelloFrameLayout tutorial from Google. Perhaps FrameLayout is too simple to warrant being included in the API Guides? However, for the sake of completeness, you have:

- FrameLayout
- LinearLayout
- RelativeLayout
- TableLayout
- AbsoluteLayout (Deprecated)

It'd be nice to have a small tutorial using all four current Layout classes. First, practice using help at http://developer.android.com/index.html. Search for FrameLayout, then use the page for the top result to answer the questions:

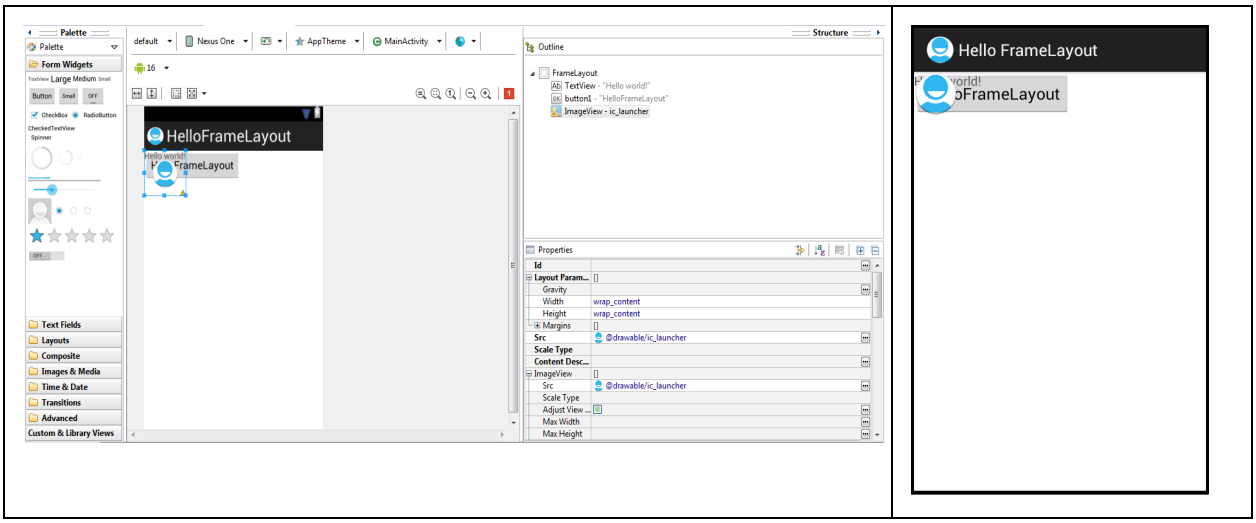
FrameLayout is designed to block out an area on the screen to display a single item. You can add multiple children to a FrameLayout, and control their position within the FrameLayout by assigning _____ to each child.

Children are drawn in a stack, with the _____

The size of the frame layout is _____

As you can see a FrameLayout is a very simple layout. It's only useful for very simple layering tasks, but those are important too. Let's do a very short tutorial with a frame layout.

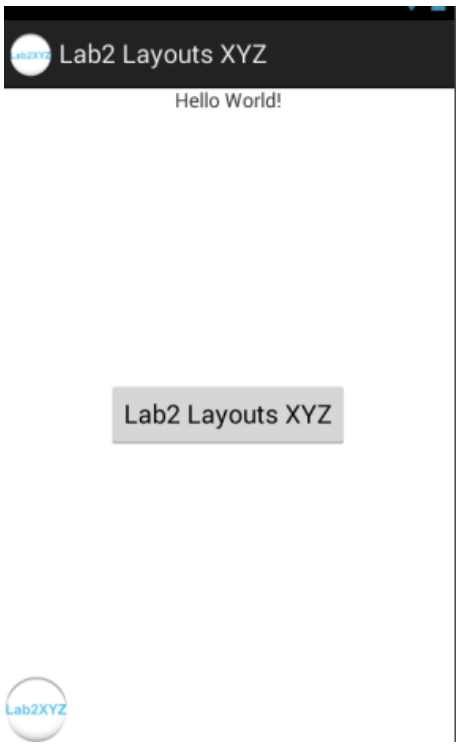
In your Lab2b project, create a new layout xml file called activity_framelayout.xml. In it, make a FrameLayout, and within FrameLayout, add a TextView that says "Hello World", a Button with the text on the button set to the @string/app_name, and an ImageView and with the src property (source) set to @drawable/ic_launcher. The order you put things into the frame determines which is on top. For example, try to make your layout look like this:



Obviously that isn't very useful since they all overlap, but you now understand how FrameLayout works. Let's make it look a bit better by setting the layout_gravity properties of the three child views. Set them as follows:

TextView	layout_gravity = center_horizontal
ImageView	layout_gravity = center
Button	layout_gravity = left|bottom

Hopefully now it looks something like this:



So you can see that FrameLayout is very simple, but combined with the child view layout_gravity property it can be used to perform simple, but very useful layout tasks. To check off this part just show your instructor this final version.

Part F: Run a Sample

The Android SDK includes a variety of sample code

http://developer.android.com/tools/samples/index.html

To run a sample, create a new Project... > Android > Android Sample Project. Select your build target and choose a sample. There are tons!

You basically have two options here:

1. Use something from Android 4.4 that requires google play services, like cuteanimals - this will require you to import a library.

2. Use something from an earlier version, like Wiktionary. This will require you to load another SDK if you only have 4.4 downloaded.

Neither is too difficult to do, but does require a bit more explanation (below).

I spent way too much time on the Wiktionary (use the menu option at the bottom right to get random words)

The goal of the samples is to look for something you want to make in YOUR apps and then look at the code in the sample to get ideas for how to modify it. I just want to make sure you know the samples are there available as a resource.

Run any sample you like and demo it for us.

Using Google Play Services (option 1):

1. From Eclipse, do File > import > Android > Existing Android Code Into Workspace. Browse to google-play-services_lib FOLDER (wherever you stored the Android sdk, then : android-sdk/extras/google/google_play_services/libproject).

In the sample project, right click Properties > Android, Add … and select the project you just imported. Then OK.

From top answer at

http://stackoverflow.com/questions/19843784/google-play-services-library-update-and-missing-symbol-integer-google-play-serv

Downloading another SDK (option 2):

Choose Eclipse > Window > Android SDK Manager. In the window that pops up, select an older one. Any one will do.

Turn in

You have finished Lab 2. Well done! Hopefully you understand Layouts and Samples better.

For any ones that are static (LinearLayout, FrameLayout), you can show them either on a device (your phone, tablet, or emulator), or in Eclipse in the graphical layout editor, whichever is easier for you. For the others, please show them on a device.