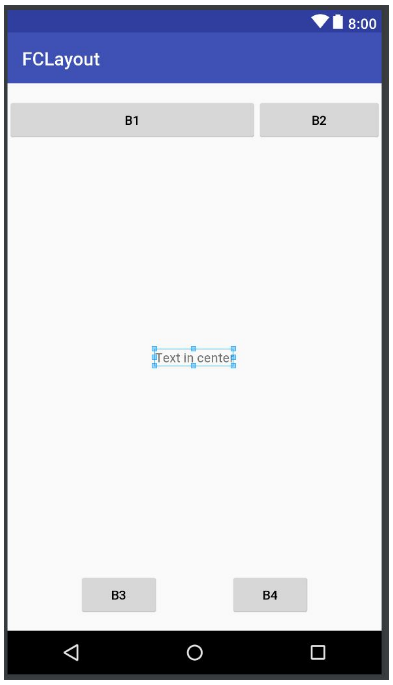
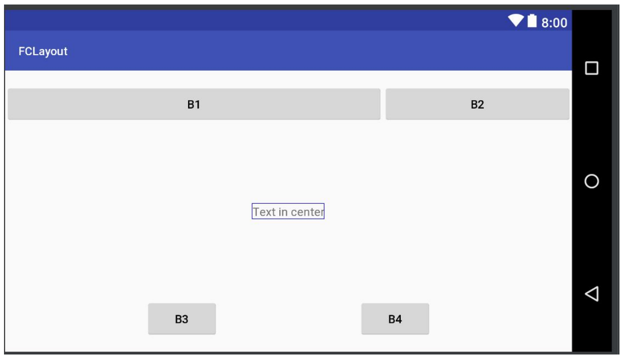
Android programming: Flipped classroom 0: Layout

Recall the rules for flipped classrooms: do the work alone or with a partner that is unique for the semester. You can find the github link on Ed Discussion and via github classroom. `https://classroom.github.com/classrooms`

Only one student needs to submit this code, but both students may do so if they wish. This assignment should be submitted through github (sorry, partners can't share the repository). Include a README at the top directory level that contains both student's EIDs.

The purpose of this exercise is to lay out four buttons (named B1, B2, B3, and B4) and one text view. You will do two independent layouts that look (almost) the same. One uses a Linear Layout (and possibly nested Linear Layouts), the other uses a Constraint Layout (no nesting needed). The layouts for Constraint and Linear should be pretty much identical. Here is a visual depiction of your goal. Also see the (short) video, accessible via the course web page.





Above are portrait and landscape screenshots of our ideal layout, with the two top buttons 16dp from the top, aligned with each other, with the left button (B1) $\frac{2}{3}$ of the horizontal real estate relative to the parent (root window), and the right (B2) the remaining $\frac{1}{3}$. The buttons can touch. The text (which should be the string, "Text in center") is in the middle of the entire screen. The other two buttons (B3 and B4) are 16dp from the bottom and equidistant from each other and from the edges of the phone. They are wrap_content in horizontal size.

Your goal is to get as close as you can to this layout, **without** using fixed-sized directives. The one exception is the 16dp margin, which you are allowed to specify for TWO view objects (buttons or layouts). Another exception is the size 0dp, because that often does not act like a fixed width. Please remember, if you are specifying widths using explicit numbers like

160dp, then you are doing it wrong. However, explicit numbers for weights and biases are fine because those values express a ratio.

We suggest writing the layout while using the live preview feature of Android Studio. In both layouts, the TextView should say, “Text in center” and the buttons should display their name.

We do not provide either layout file, though we do provide the files and the first line which is the xml header. You should be able to start typing XML in the layout editor and get intelligent completion.

Your layout should work if there is a toolbar present or not. In my solution, there is no toolbar in the preview mode, but the style uses a toolbar, so there is one present when the app executes. However, the correctness properties are identical whether there is a toolbar or not. For example, the two top buttons should be 16dp “from the top” where the top is either the bottom of the toolbar, or the top of the screen.

AI Challenge. Create a new XML layout file called ai_challenge.xml. Get an AI (or if you like, do it yourself) to generate 3 copies of a telephone number layout with a 3x3 box with numbers 1-9 and then a zero underneath in the middle. Please no * or button. One telephone layout on top, and two underneath, all centered in the layout. In an XML comment in ai_challenge.xml, please write the prompt that got you the result and what AI you used. You can also add other XML files.

This challenge will be worth some points, but not many so only do it if you think it is fun or if it is easy.

Files of interest

1. **activity_main.xml** Please use a ConstraintLayout.

   The design view helps when working with constraint layouts. You might want to look at chains, which you can define with the GUI and modify the XML text. You will not need any nested layouts.

2. **activity_main_linear.xml** Use only LinearLayouts, including nested LinearLayouts. Remember, layout_weight is your friend, as is gravity (position within an object (e.g., a button within a LinearLayout)), and layout_gravity (position of an object in its parent).

   The nesting on the bottom layout gets a little deep. That makes it tricky, so if you are getting too frustrated, just some number of invisible views or buttons (android:visibility=“invisible”). But if you like puzzles, know that you can do it with only LinearLayouts and Buttons and without invisible buttons/views. It takes multiple LinearLayouts! And it does not recreate the exact button layout for B3 and B4, but it is very close.

Please check your layout in the design view and also run it in the emulator.