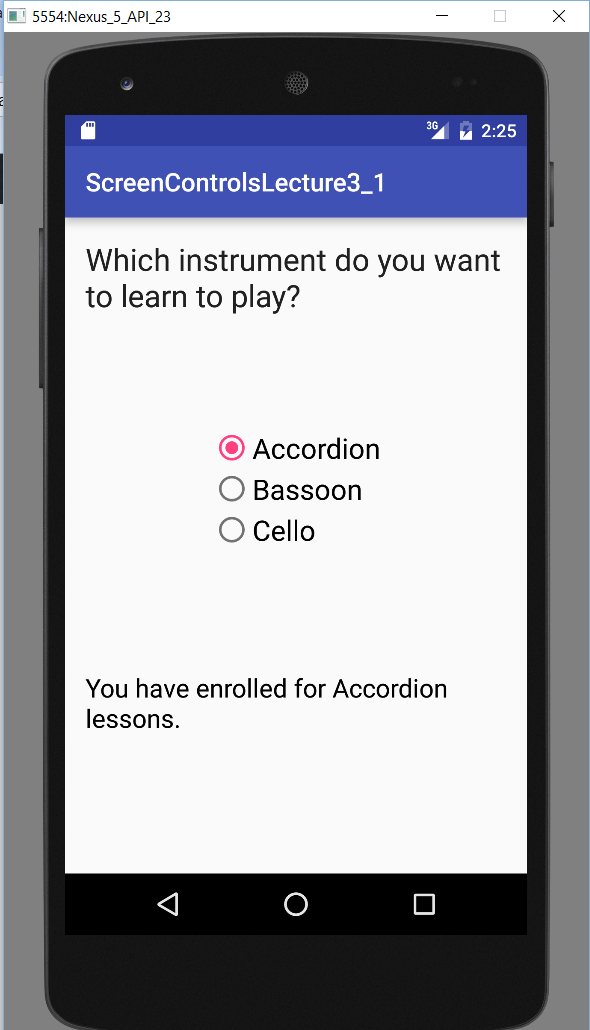
# IN721 2018 2.2: Complex Screen Controls

## Task 1: Improve the UI of the Music School app

In the lecture, we saw an app that let the user (pretend to) enrol in music lessons by selecting the desired instrument from a radio group, as shown.



As discussed, the UI here is flawed. When the user clicks on a radio button, s/he is enrolled without confirmation. Thus a simple clicking error could lead to enrolment in the wrong course. You are going to fix this problem. Build this app, but modify it so that the user ***first*** selects one of the radio buttons, ***then*** clicks on a normal button (technically called an ***action button***) to confirm the enrolment.

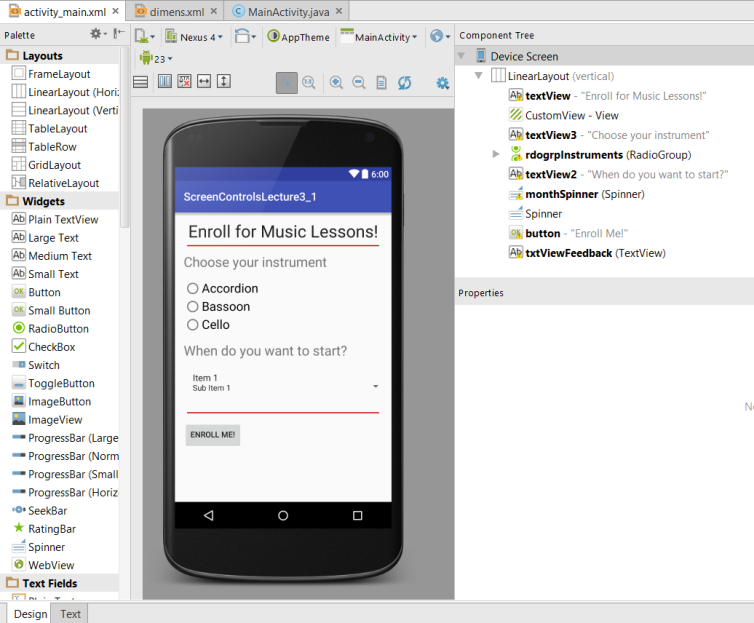
### Task 2: Use a Spinner for a Larger Set of Options.

Radio buttons are effective controls because users understand how they work without instructions (we all know those little circles mean "mutually exclusive set of buttons") and all the available options are clearly displayed on the screen. However, if the set of options you wish to offer is large, radio buttons can take up more screen space than is practical.

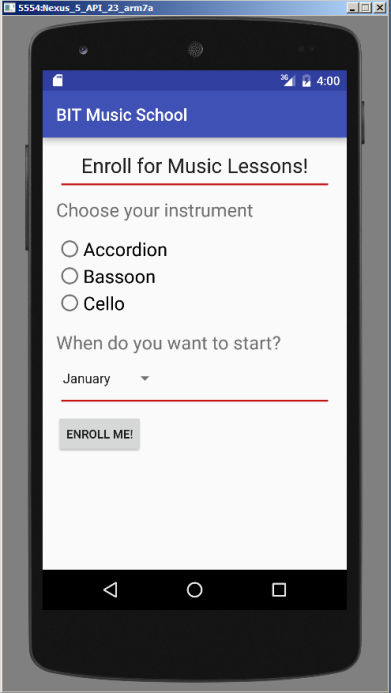
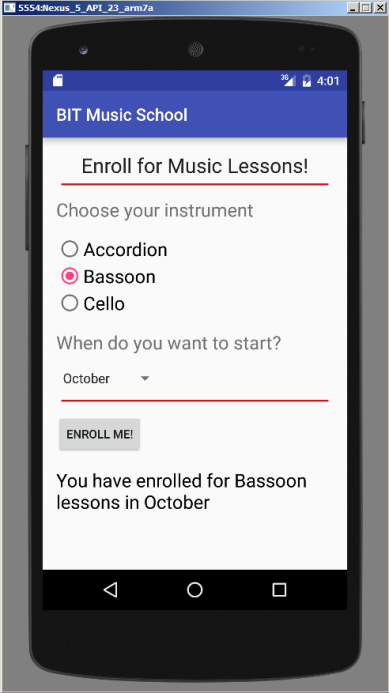
Imagine that your music school starts a new set of lessons each month. When a student enrols, s/he must specify both the instrument (accordion, bass or cello) and the month in which s/he wishes to start. With 12 options for month, radio buttons are not appropriate. Instead, you should use a Spinner to allow the month to be specified.

Modify the Music School app to let the user choose which month they wish to start lessons, via a Spinner control. As above, provide an action button for confirmation, and clear feedback of the result. Screenshots of my solution are shown on the next page. Some potentially useful features:

* This solution uses a LinearLayout. When all your controls are lined up vertically or horizontally, the LinearLayout can be easier to work with than the RelativeLayout.
* This solution uses the same margins and font size for the two instruction TextViews to give a consistent appearance. The values are defined in /res/values/dimens.xml to make modification easy. The colour for the text and horizontal lines is defined as a <color> resource in /res/values/colours.xml. Use resource constants this way in your app.
* The horizontal lines are <View> elements with android:layout\_height=*"2dp"* and android:background set to a colour. You will not find a plain <View> control in the Palette. How, therefore, can you add it to your screen layout?



In Android Studio

At Launch After selection and button click