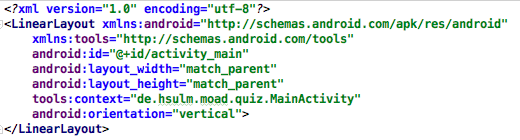
## Exercise 1.1: Set up your development tools

Set up your development tools and implement the first example from the lecture!

## Exercise 1.2: Create the UI for a small quiz game

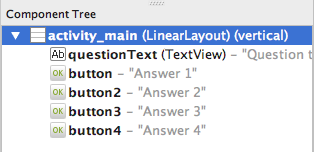
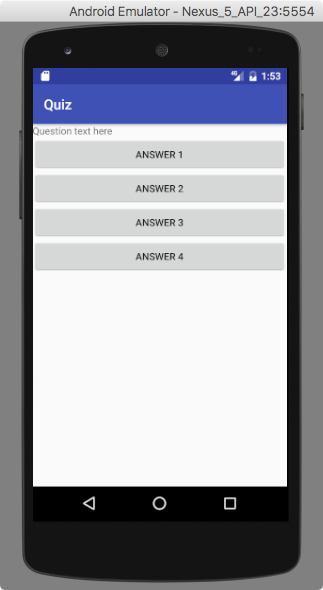
In this exercise you will create a very small quiz game. We start with a simple layout:

* Create a new project and open you layout *(activity\_main.xml)*.
* In the *Text* view change the root element to *LinearLayout* and set an attribute *android:orientation=“vertical“.* The result should look similar to this:



* In the *Designer* view add a *TextView* for the question text and four *Button*s for the possible answers. You can also directly drag elements into the *Component Tree* if you want. The controls will automatically be arranged as a vertical list.
* Set values for initial text and an id for each of your UI elements.
* Test your layout by starting the app with the emulator or a real device.

The result can look like this (running app and component tree):



## Exercise 1.3: Add questions to your quiz game

In this exercise you will add a list of questions to your quiz, select a random one and show it in the UI from 1.2.

* Set up a class Question that can store a quiz question, four possible answers and the number of the correct one. Also provide a constructor and necessary methods to access the data.
* Set up a class QuestionRepository that manages a set of questions. The class should manage the questions in an internal ArrayList<Question> and have a method   
  Question randomQuestion() that returns a random question from the list (obviously…).

The questions themselves can be hard-coded in the source code. Don’t lose time thinking of quiz questions, you can find plenty on the web[[1]](#footnote-1)!

* In your *Activity* initialize a QuestionRepository as an attribute and in the onCreate() method get a random question, find the TextView and the Buttons and set their texts accordingly.

## Exercise 1.4: Add quiz functionality to your game

Now you will actually implement the game play:

The user can select one of the answers offered on the buttons. If the user selected a wrong button, it will be disabled (greyed out) and she/he can try another option. Once the right answer was selected, a new question is drawn from the QuizRepository, shown in the UI and all buttons reenabled.

Remarks:

* Set onClick-methods for the buttons. For now, it is easiest to define a different method for each button.

However, as an advanced approach you can also use the same method for every button and distinguish with the *View*-parameter the callback method is supplied with.

* You can disable (grey out) a Button (actually any UI element) with a call to its method setEnabled(false) and reenable it with setEnabled(true).

1. f.e. see <https://www.gamefaqs.com/gba/919785-who-wants-to-be-a-millionaire-2nd-edition/faqs/40044> [↑](#footnote-ref-1)