Non-graded Exercise: Hello World++

Attempt 1

In Progress

NEXT UP: Submit Assignment

Add Comment

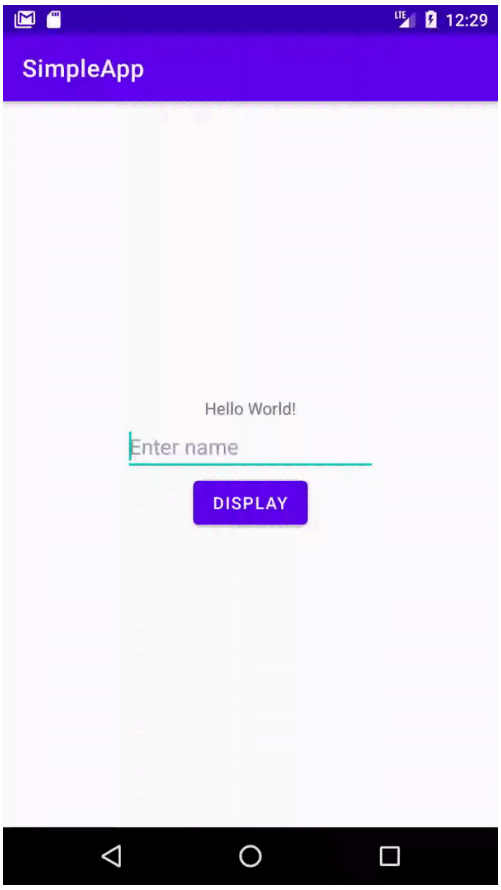
Unlimited Attempts Allowed

Details

Note: The following activity has code written in Java. This isn't a big issue, but please note that you can either write the project as is or use this as a practice for writing code in Kotlin. If this is discussed in class, Kotlin equivalents should be discussed by your instructor.

Description

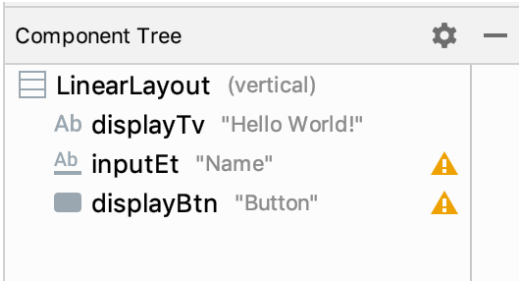
We haven't discussed any specifics in terms of Android Development, but let's try coming up with a simple app that looks like the following:



General Steps

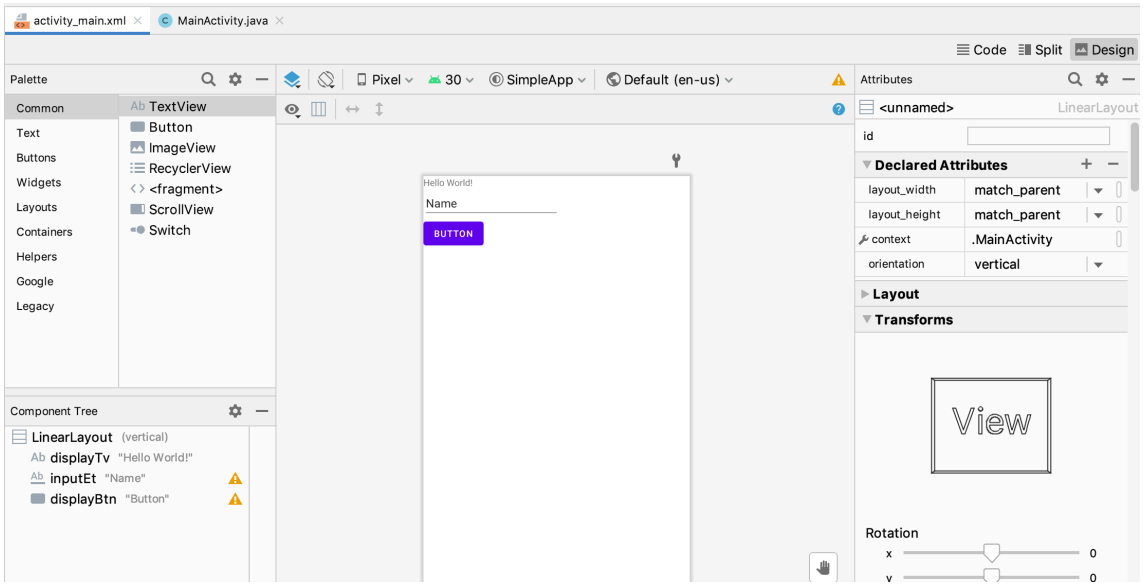
1. Create a new project using the Empty Activity template
2. In the Project Pane, head to res > layout and double-click on activity_main.xml to open the file
   - This is your MainActivity's layout (UI) file
   - In the Component Pane, perform the following:
     - Right-click on the ConstraintLayout and click "*Convert view...*". Kindly select LinearLayout.
     - Right-click again on the LinearLayout, hover over LinearLayout, and select "*Convert orientation to vertical*". We'll want our UI elements to appear vertically instead of horizontally.
3. Next, in the [View] Platte Pane, find and drag to the Component Pane the following: (1) Plane Text (EditText) and (2) Button

- Make sure the Component Pane looks like the following:



  - Rename the Views accordingly.
    - To do so, click on the View to bring up the View's attributes (see the Attribute Pane). Change the "*id*" to change its name.
- If the Plain Text and Button views span the entire screen, click on each View and identify the attribute "*layout_weight*" in the Attributes Pane. Remove the value.

4. Your layout file should look like the following:



- Don't worry so much about it looking ugly right now. The important thing right now is that we have the UI in place. We'll fix the alignment after.

5. From the Project Pane, find the MainActivity.java file and double-click on it.
   1. Add the following attributes to the class:

```
private EditText inputEt;
private Button displayBtn;
private TextView displayTv;
```

   2. Add the following code within the onCreate method:

```
this.inputEt = findViewById(R.id.inputEt);
this.displayBtn = findViewById(R.id.displayBtn);
this.displayTv = findViewById(R.id.displayTv);
```

   3. Add the following code after the previously added code in the onCreate method:

```
this.displayBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        displayTv.setText("Hello, " + inputEt.getText().toString());
    }
})
```

   - Tip: Type this out slowly and allow Android Studio to auto-generate what code can be generated.

6. To tie up loose ends in terms of the UI, proceed back to the activity_main.xml layout file
   - Click on the LinearLayout and find the attribute "*gravity*". Use the search bar to find it easily. Find the value center and check it so that the value is true.
   - Click on inputEt and remove the value in the attribute "*Text*". Find the attribute "hint" and enter the following: "Enter a name"

Animo Space Suppor

- Click on displayBtn and change the attribute "*Text*" to "Display"

Non-guided Tasks

From the base app you made, make a couple of changes:

- Changing the font size
- Modify the color of the button.
- Add spacing so that the UI elements aren't so close together
  - Clue: explore Spacing Views, margins, or padding
- Revert to a ConstraintLayout and explore constraining all views properly
  - This will be further explained when we formally tackle Views
- Add a button and give it logic such that clicking on it will display a Toast feedback that greets the user with the input name
- Add an image
  - If this is too boring, then (1) place another image in the project, (2) add logic such that when you click the image, it swaps to the second image you placed in the project. This might be a little confusing at first. :)
- Add a log.d() method call in any listener supplying it with the proper parameters. For example:
  - ```
    private static final String TAG = "MainActivity"
    ...
    log.d(TAG, "Log call here!")
    ```
  - Then observe the log call in the Logcat
- Use the debugger of Android Studio to stop execution at a specific point in the code.
  - Observe how the variables are kept track, as well as how the debugger is used

Submission Details

If you plan to submit your project here, please submit a zip file of your project. **BUT WAIT**! A zip file in this case does not mean zipping your entire folder. In order to export an Android Studio Project as a zip file, from the IDE, go to File -> Export -> Export to Zip file. Select the appropriate directory to save the zip file. Please note that this is different from compressing the entire project folder. Performing the indicated steps results in a slightly more compressed version of the project -- lighter file, easier to upload. There are some components that do need to be transferred over and it makes it a much longer process (*for me*) to download. haha.

Choose a submission type

Upload | ARC | More

Choose a file to upload

File permitted: ZIP
    ```

AnimoSpace Support

or

Canvas Files

(https://dlsu.instructure.com/courses/214805/modules/items/5850434)

Assignment

(https://dlsu.instructure.com/courses/214805/modules/items/585(