Android Application Development, COMP 10073

Mohawk College, Winter 2021

Layout Editor

Make sure you review the details for the Layout Editor on Android Studio web site.

https://developer.android.com/training/basics/firstapp/building-ui

https://developer.android.com/studio/write/layout-editor

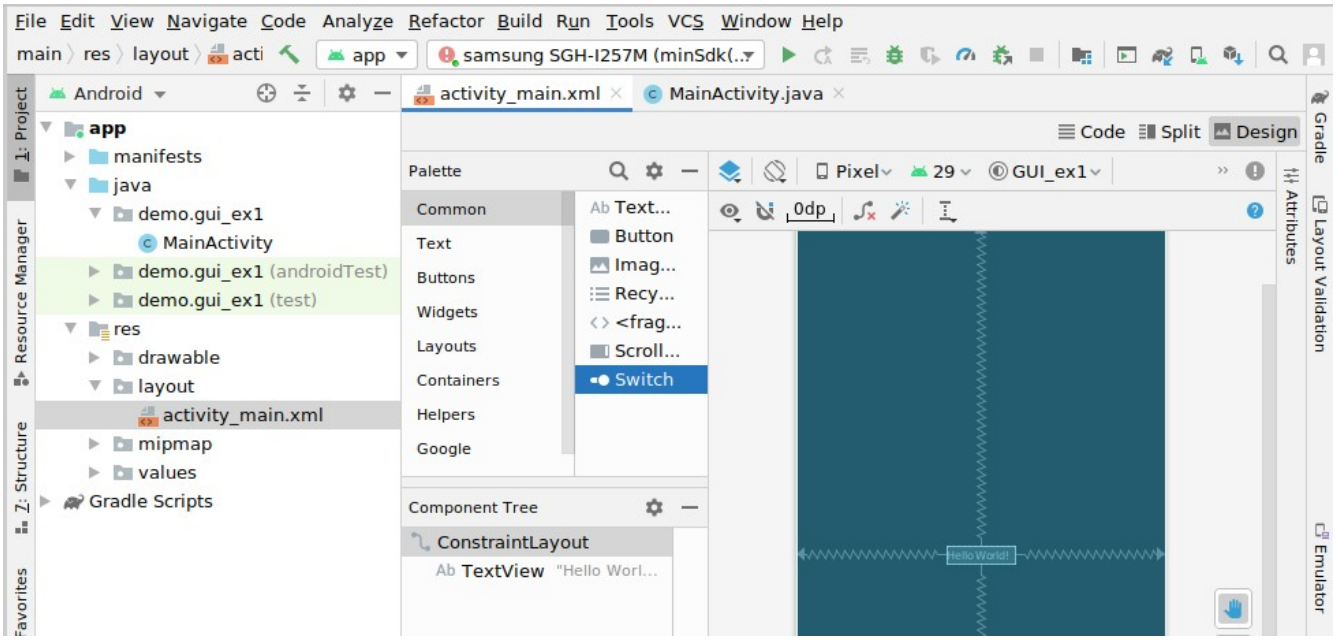
https://www.youtube.com/watch?v=yT22cqCGjQQ

Some important highlights will be mentioned here.

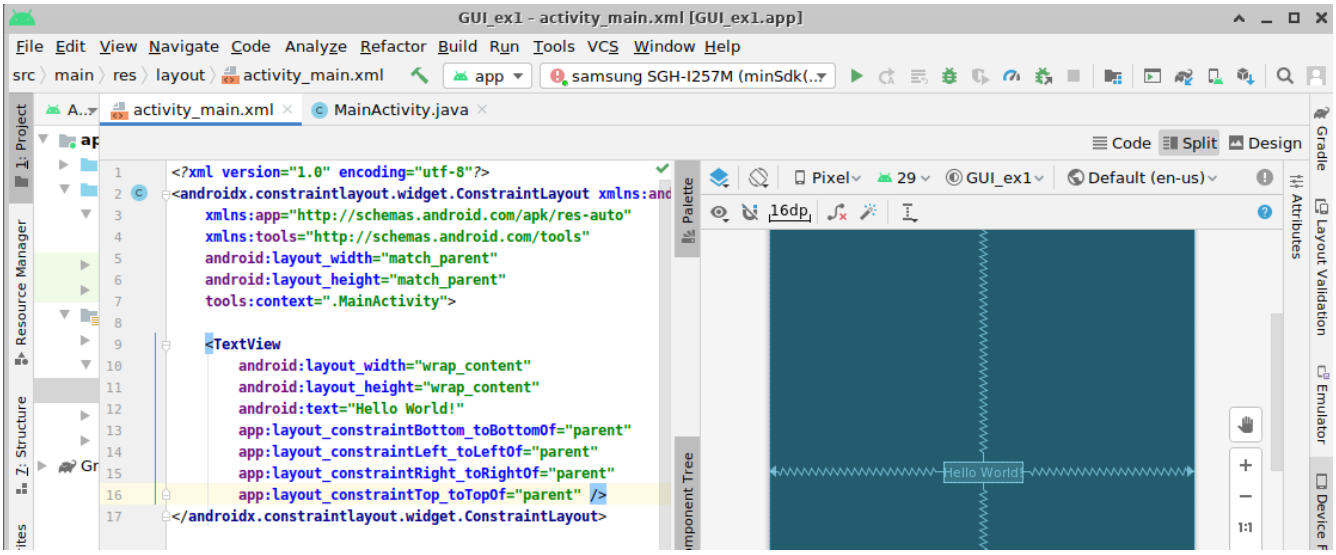
Using the instructions from the first lecture, create a project using the Empty Activity template and set it up for API 23: Android 6.0 (Marshmallow). Your Android Virtual Device (AVD) should also be configured as an Android 6.0 device. The Empty Activity template provides a collection of XML resources, configuration files, a MainActivity.java file, and some gradle build files.

Under resources click on the activity_main.xml file.

Use the "eye" tool to select "Show All Constraints". Use the blue box above the eye to select the blueprint view.



In the upper right corner of the editor, you are given the options code, split and design. Select split and the layout editor will display the blueprint and the XML.



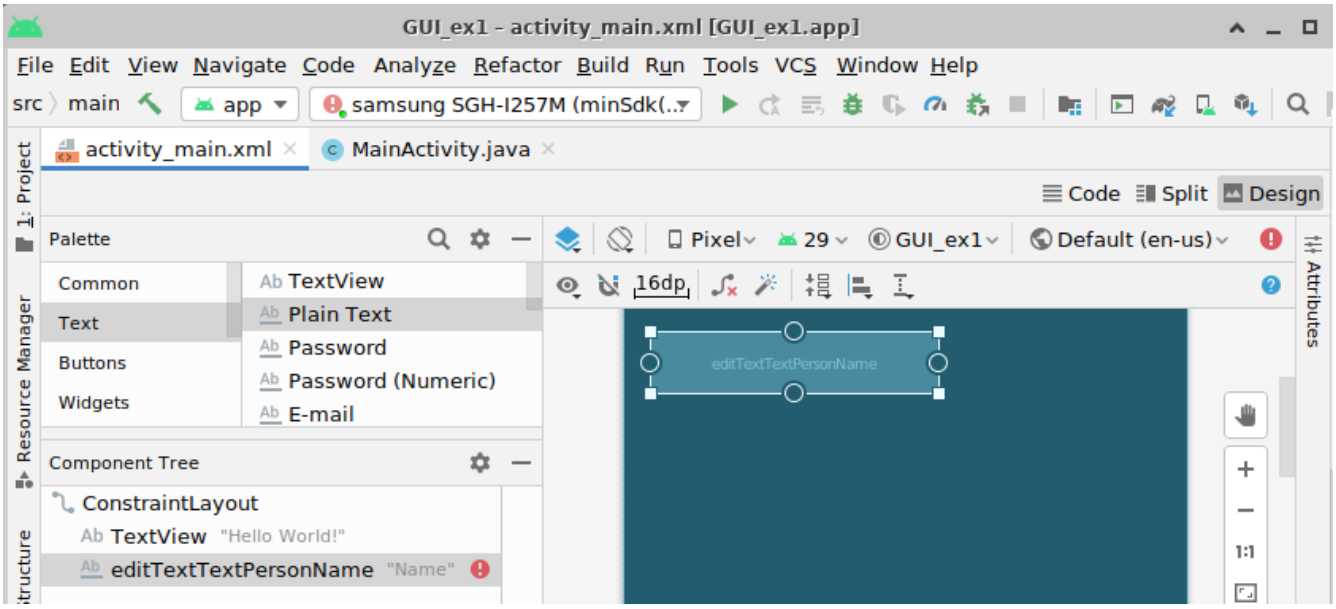
The following XML code comes with the Hello World Demo.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

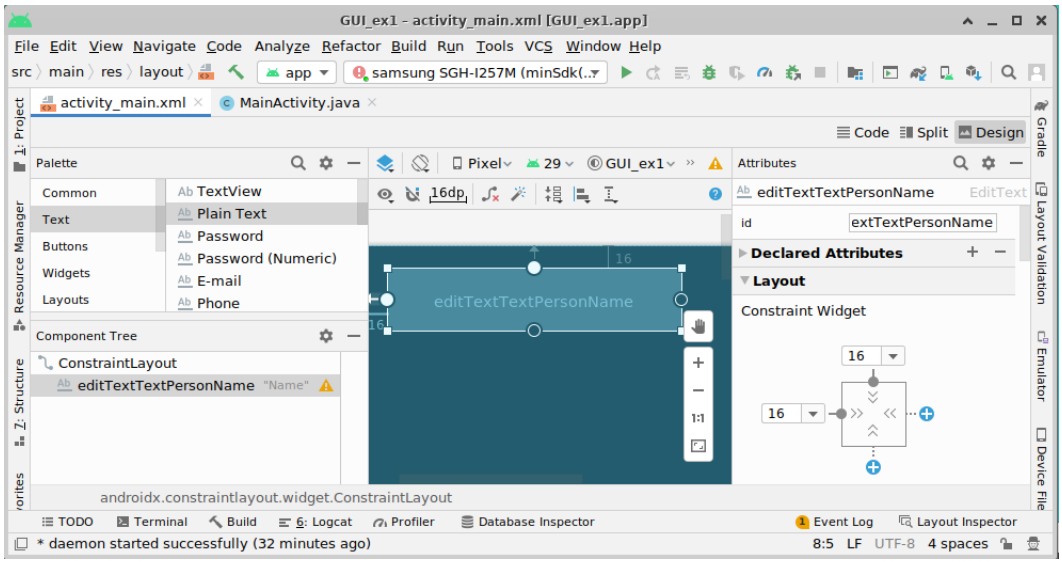
If you break the layout you can always revert to the original by replacing the old XML commands.

You can enter XML directly or copy and paste it from another source. Assignments will sometimes include a partial XML layout.

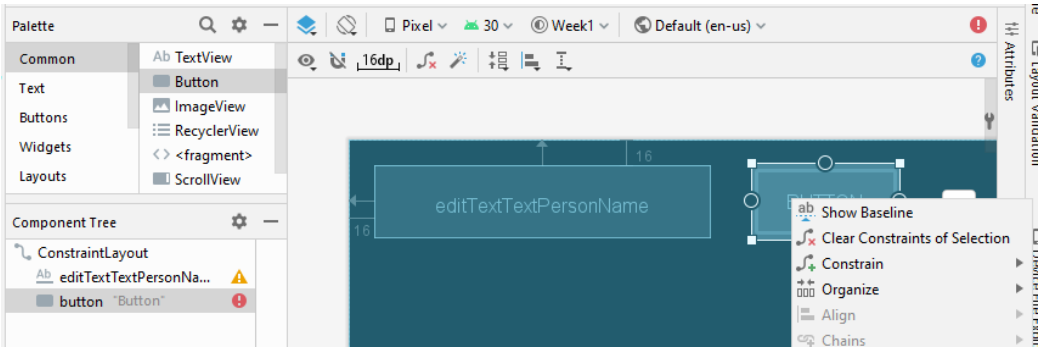
Delete the text view widget. Select the Text Pallette and choose the Plain Text widget and drag it to the design editor. The “Plain Text” widget is badly named, this widget is used for text entry, we will need it for the first few assignment.



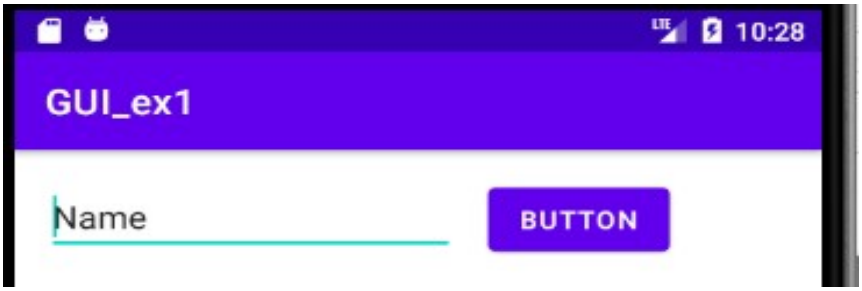
Initially it is unconstrained, add simple fixed constraints to the top and left sides. The constraint widget in the attributes tool will allow you to enter specific numbers if you like. This tool also shows the constraints of the widget size. The “>>” indicates that the widget width and height are set to wrap_content.



Add a button from the common palette, right click on the button and select show baseline. Use the button's baseline to connect it to the editText widget.



The baseline gives better horizontal alignment than attempting to anchor the button to the top of the parent since the widgets may not be the same size.



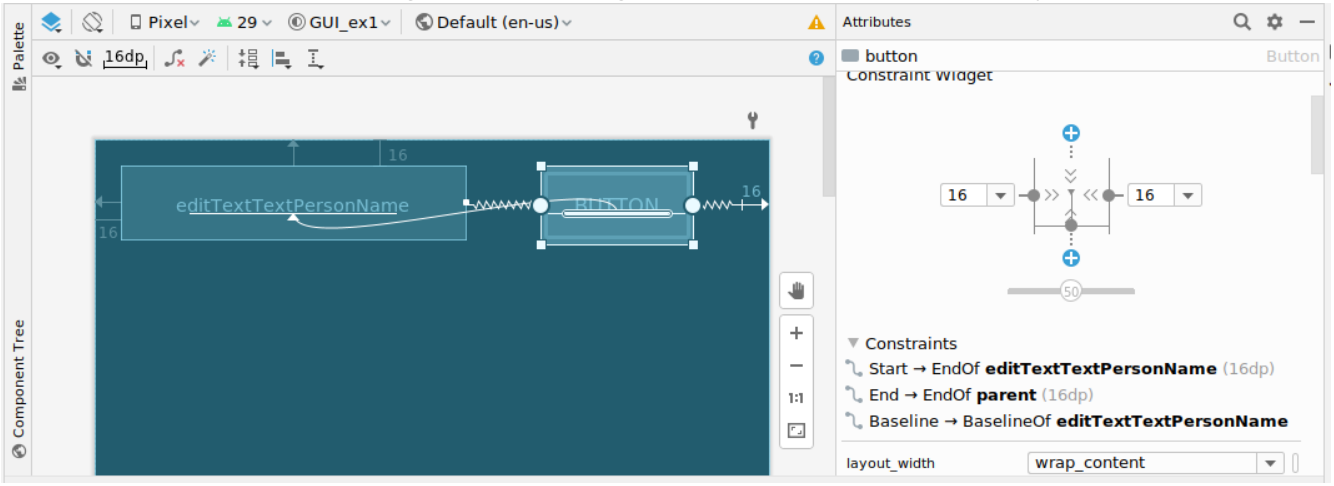

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/editTextTextPersonName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Name"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:text="Button"
        app:layout_constraintBaseline_toBaselineOf="@+id/editTextTextPersonName"
        app:layout_constraintStart_toEndOf="@+id/editTextTextPersonName" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Constraint Layouts

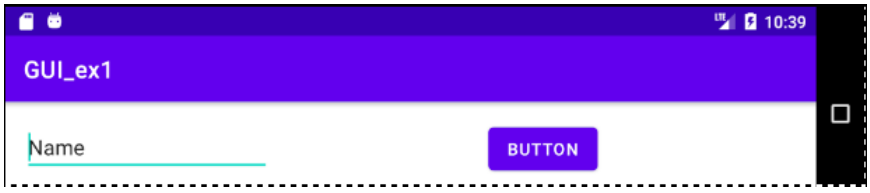
Some of the details from the following link are only relevant to older versions of Android studio. In particular the comments about including dependencies for the constraint layout to the project or converting a layout to a constraint layout are not relevant to us. See:

https://developer.android.com/training/constraint-layout

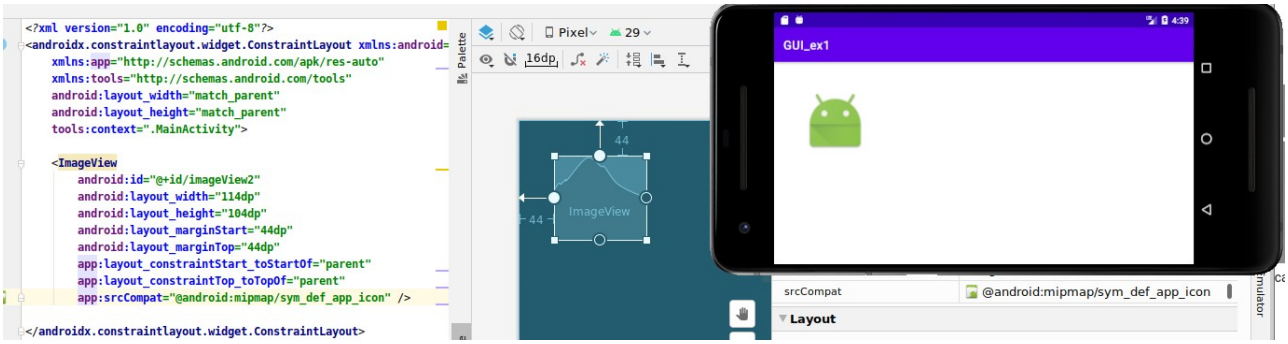
For our previous layout we can center the button by adding an opposing constraint to the left hand side of the button. This will have the effect of centering the button between the edit text widget and the right hand side of the activity.



The centering is noticable when the phone is rotated.



To use the ImageView widget select an example image from the resources directory, otherwise the image won't be available on the emulator. Set "app:srcCompat" to point to your image file. The "tools:srcCompat" attr sets the image for the editor, you should set both.



Aligning Widgets

https://developer.android.com/training/constraint-layout#position

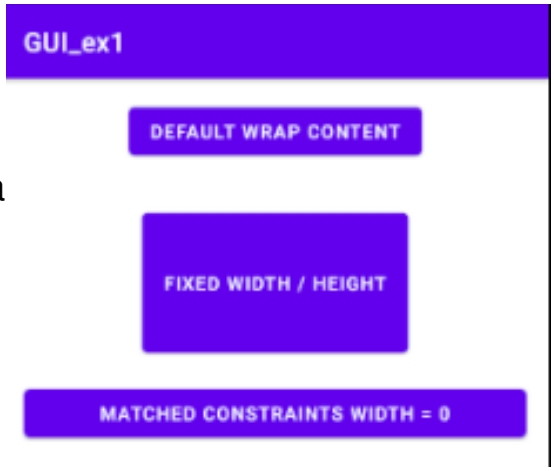
In this example the image is located a fixed distance from the top and left borders, the top button id defined relative to the top of the image, and the other buttons define their right hand edge in terms of the top button.


```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="114dp"
        android:layout_height="104dp"
        android:layout_marginStart="40dp"
        android:layout_marginTop="108dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@android:mipmap/sym_def_app_icon" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="52dp"
        android:text="Button"
        app:layout_constraintStart_toEndOf="@+id/imageView2"
        app:layout_constraintTop_toTopOf="@+id/imageView2" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:text="B2"
        app:layout_constraintEnd_toEndOf="@+id/button2"
        app:layout_constraintTop_toBottomOf="@+id/button2" />

    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:text="Button Number Three"
        app:layout_constraintEnd_toEndOf="@+id/button3"
        app:layout_constraintTop_toBottomOf="@+id/button3" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Adjust the View Size

https://developer.android.com/training/constraint-layout#adjust-the-view-size

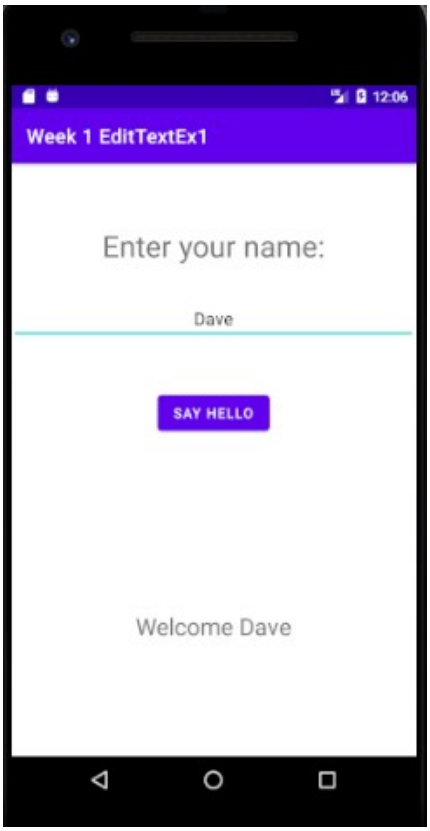
You can use the corner handles to resize a view, but this hard codes the size so the view will not resize for different content or screen sizes. To select a different sizing mode, click a view and open the Attributes window on the right side of the editor.




```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:text="Matched Constraints Width = 0"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button2" />

    <Button
        android:id="@+id/button2"
        android:layout_width="201dp"
        android:layout_height="117dp"
        android:layout_marginTop="32dp"
        android:text="Fixed Width / Height"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button3" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:layout_marginBottom="16dp"
        android:text="Default Wrap Content"
        app:layout_constraintBottom_toTopOf="@+id/button2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Week 1 - EditTextEx1

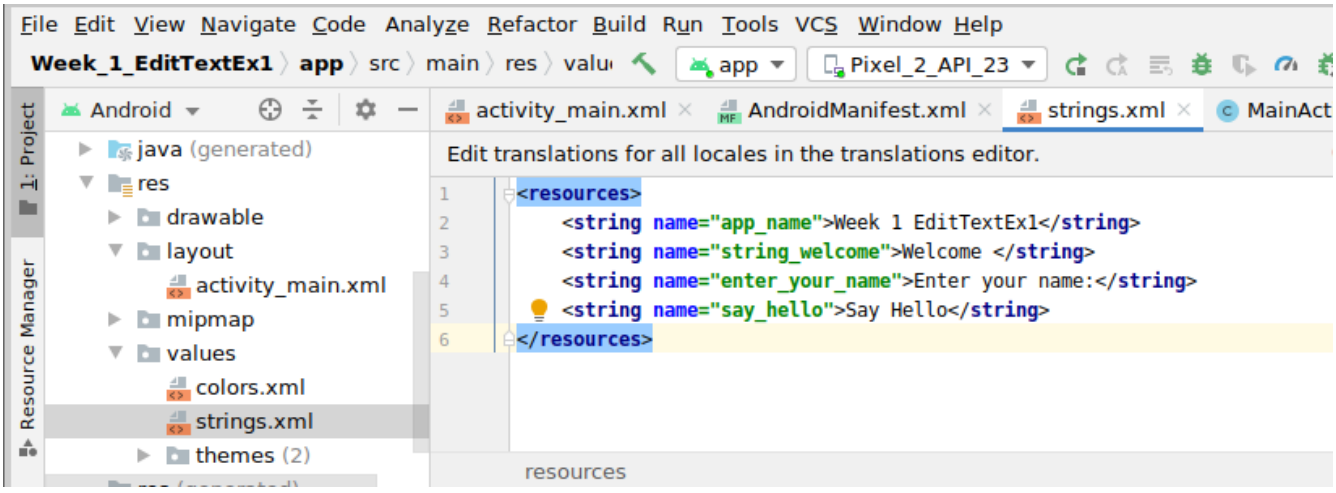
The following example can be used as a basis for the first lab, the code for the example will be posted with the lecture. The app asks you to enter your name, when you press the button it says “Welcome <name>”. We will develop an event handler and write text to the GUI. This example handles rotation in an interesting way.



Strings

We mentioned the string resource previously, see:

https://developer.android.com/guide/topics/resources/string-resource



Reference in Java via: R.string.string_name

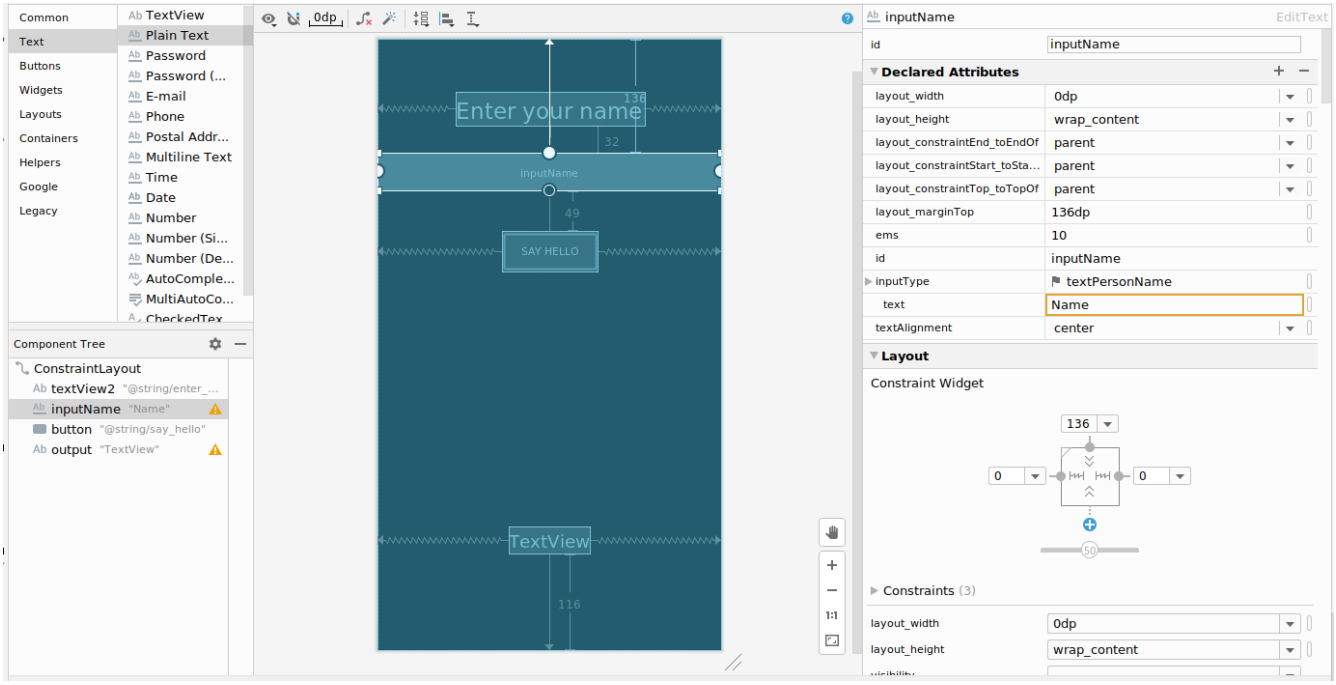
Reference in XML via: @string/string_name

Layout Edit Text Widget

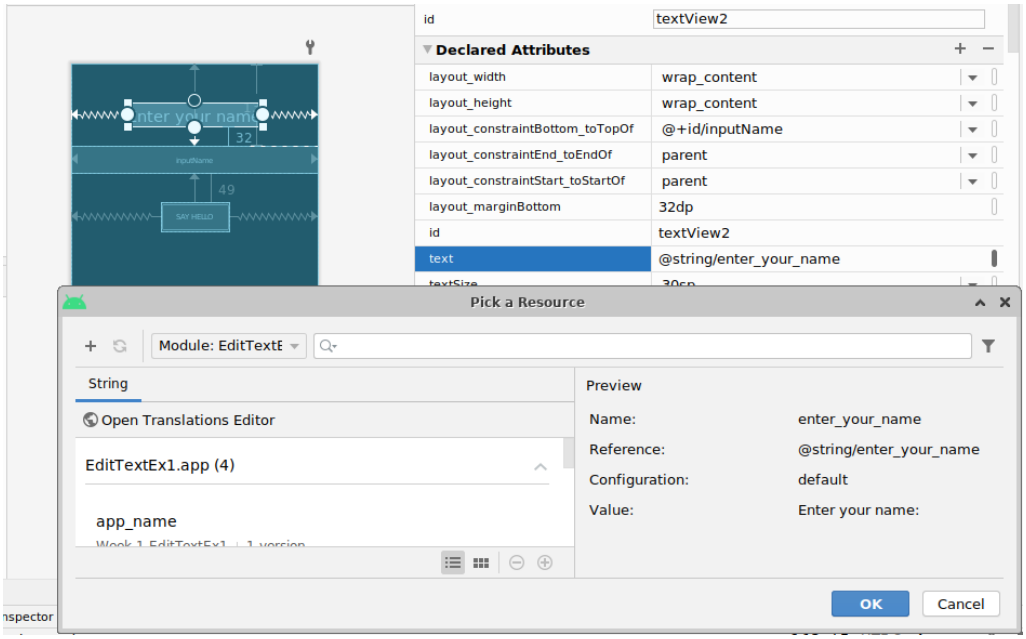
https://developer.android.com/reference/android/widget/EditText

The EditText widget is found on the text palette. Select “Plain Text” and drag it to the layout editor. Other widgets can limit the text allowed for entry.

Notice that the EditText widget fills the width of the activity. The TextView widget is anchored to the bottom rather than being anchored to the widget above it.



Use an XML reference for the fixed string values. Click the tab beside the value:



XML Layout

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="32dp"
        android:text="@string/enter_your_name"
        android:textSize="30sp"
        app:layout_constraintBottom_toTopOf="@+id/inputName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <EditText
        android:id="@+id/inputName"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="136dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Name"
        android:textAlignment="center"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="49dp"
        android:onClick="doStuff"
        android:text="@string/say_hello"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/inputName" />

    <TextView
        android:id="@+id/output"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="116dp"
        android:text="TextView"
        android:textAlignment="center"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

EditText and TextView and View Objects

https://developer.android.com/reference/android/view/View

The View object Represents the basic building block for user interface components. A View occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for widgets, which are used to create interactive UI components (buttons, text fields, etc.).

https://developer.android.com/reference/android/widget/TextView

The TextView object inherits all the methods of the View Object. It provides a basic element for display to the user.

https://developer.android.com/reference/android/widget/EditText

The EditText object inherits the methods of the TextView object. It provides an element that allows input.

findViewByID

https://developer.android.com/reference/android/view/View#findViewById(int)

To get the references for these objects we will use “findViewById” and provide a reference to our collection of resources. At the top of the attributes section in the editor you will find an “id”, for example, the id set for our TextView widget that we will use for output is specified in the XML file as:

```
<TextView
        android:id="@+id/output"
...
```

The following Java code declares a reference of type “TextView” and gets the value for the reference from the resource id “R.id.output” and then sets the value of the text to NULL.

```
TextView myOutput = findViewById(R.id.output);
myOutput.setText("");
```

N.B. findViewById will return a null if there is something wrong with the resource identifier.

We want to clear out the text when the activity starts, so add this code for both of the text view widgets.

getText()

https://developer.android.com/reference/android/widget/EditText#getText()

We need to write a handler for our button, call it “doStuff”. Use findViewByID() to look up the reference for the EditText widget. EditText has a method “getText()” which returns an object of type “EDITABLE”. We need to convert this to something we can use via “toString()”.

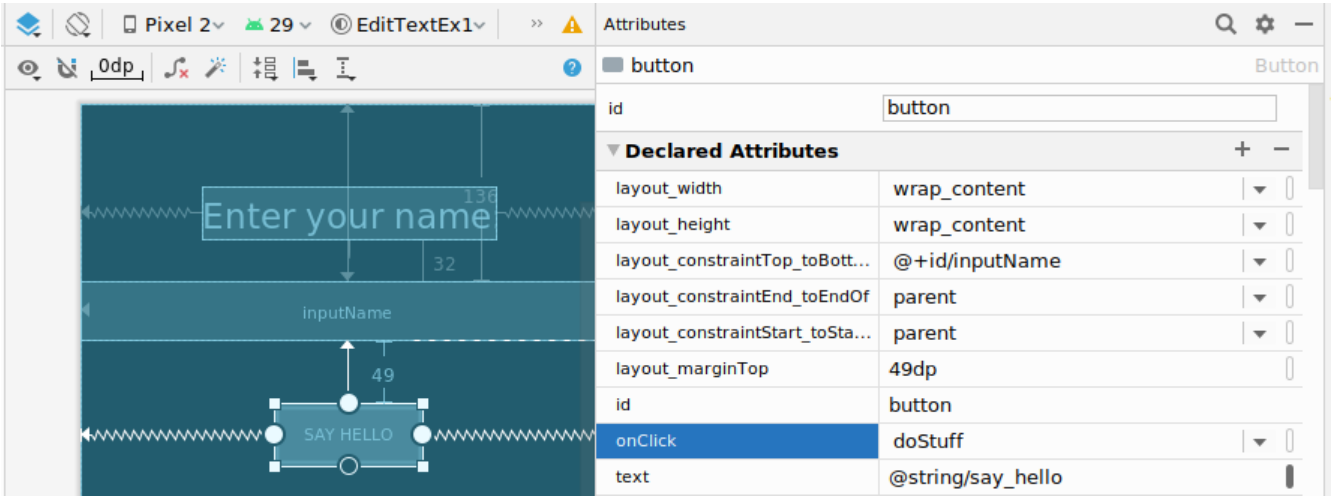
The full example code looks like this:

```
package mohawk.ca;

/**
 * Statement of Authorship goes here.
 *
 * Link to a video demonstrating your work goes here.
 */

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Initialize the fields to blanks
        EditText myInput = findViewById(R.id.inputName);
        myInput.setText("");

        TextView myOutput = findViewById(R.id.output);
        myOutput.setText("");

    }

    public void doStuff(View view) {

        // Read the Edit Field, get text string into variable
        EditText myInput = findViewById(R.id.inputName);
        String  name = myInput.getText().toString();
        name = getString(R.string.string_welcome) + " " + name;

        // Set the modified text string to the Text View on screen
        TextView myOutput = findViewById(R.id.output);
        myOutput.setText( name );
    }

}
```

Setting onClick

https://developer.android.com/guide/topics/ui/controls/button#HandlingEvents

Once you have written an appropriate button handler you will find the handler as a selectable option in the drop down menu for the "onClick" attribute. Your handler must return "void" and it must take a single argument of type "View".



These details should be enough to get you started with the first homework assignment, which includes a few modifications to this example.

Exporting Your Work

When you are done, create a .zip file of the project using the option under "Manage IDE Settings" --> "Export to Zip File..."