

MOBILE DEVELOPMENT

# Basic Layout Development 1

# Outline

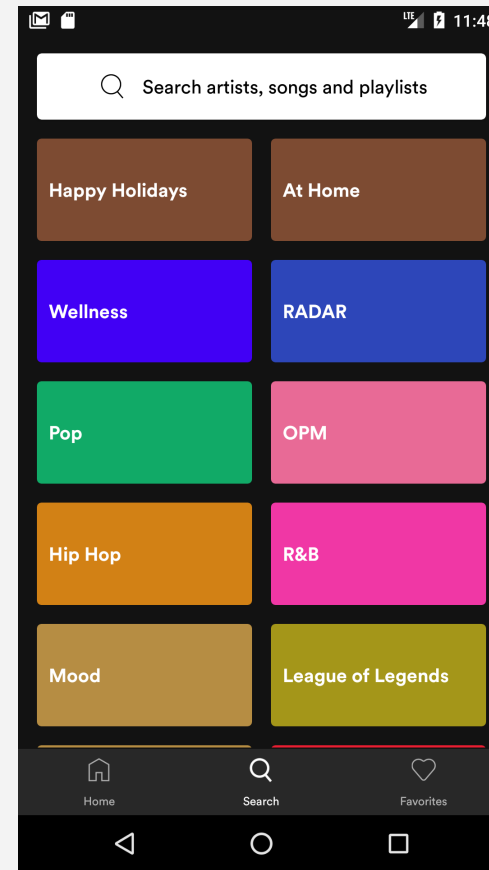
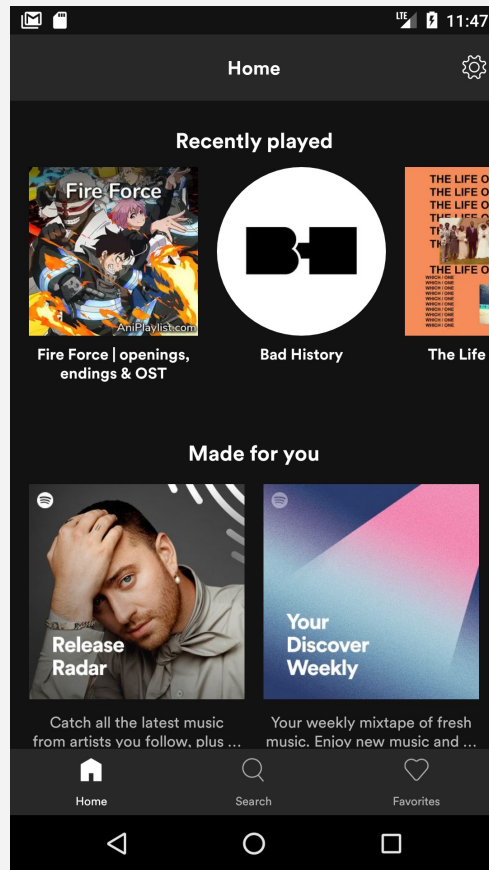
- Android Basics: User Interface
  - Resources
  - Views (UI elements)
  - ViewGroups / Layouts

# Take a moment to think:

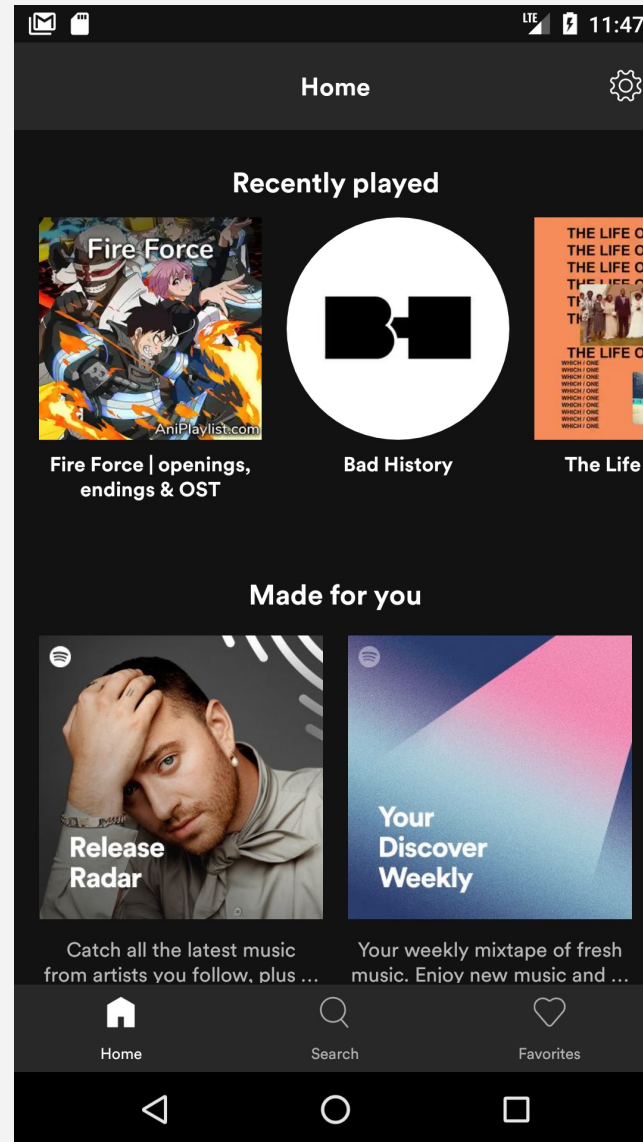
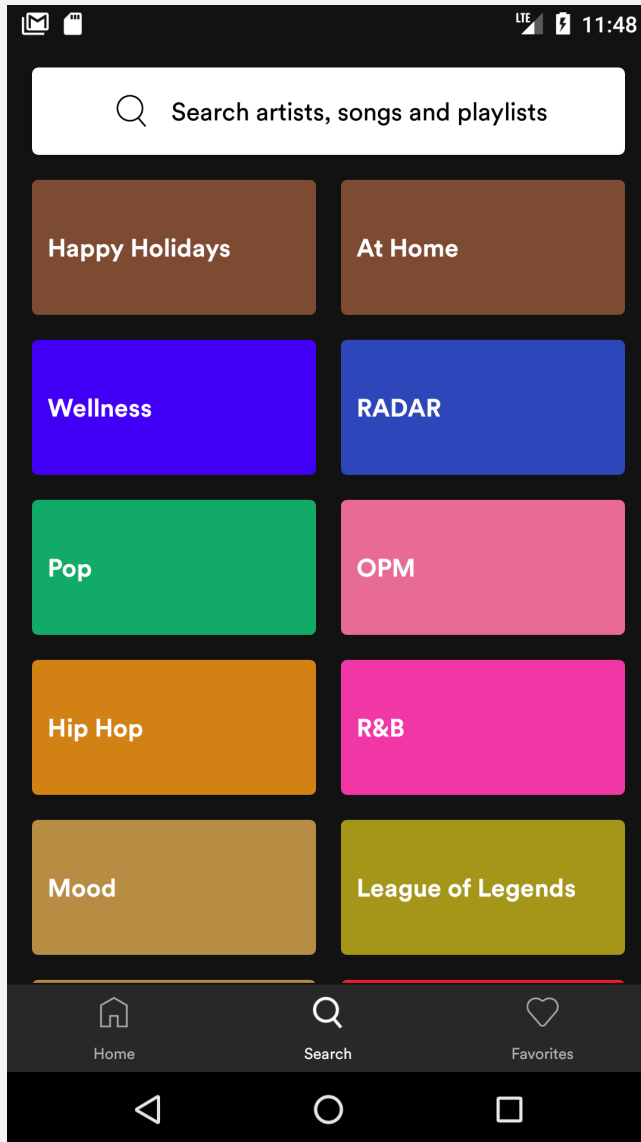
What **app(s)** do you **commonly** use?

# Motivation: Think of their UI

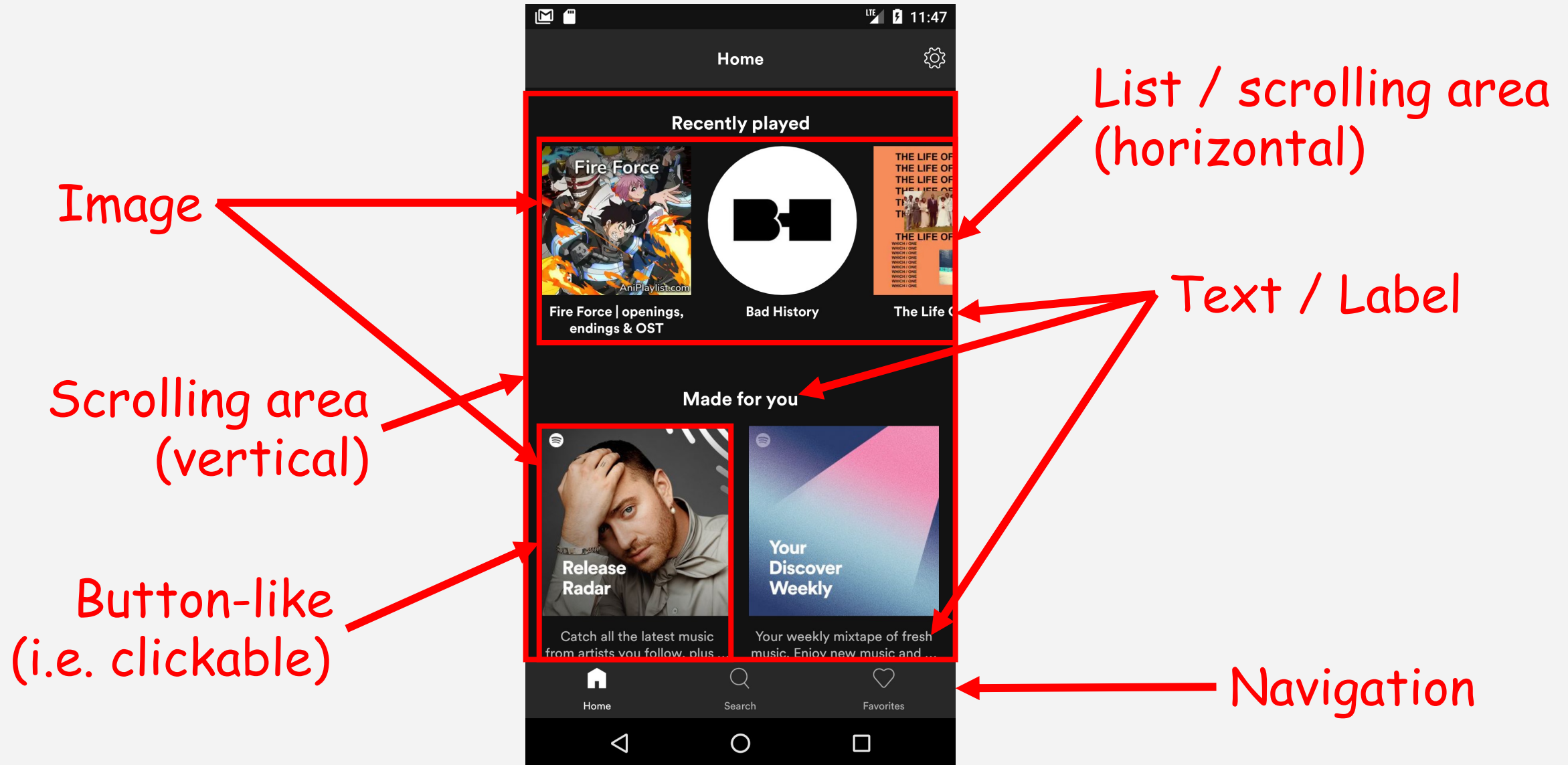
- We can learn a lot the **user-facing** aspect just by looking at our commonly used apps



# What UI elements can you identify?



# What UI elements can you identify?



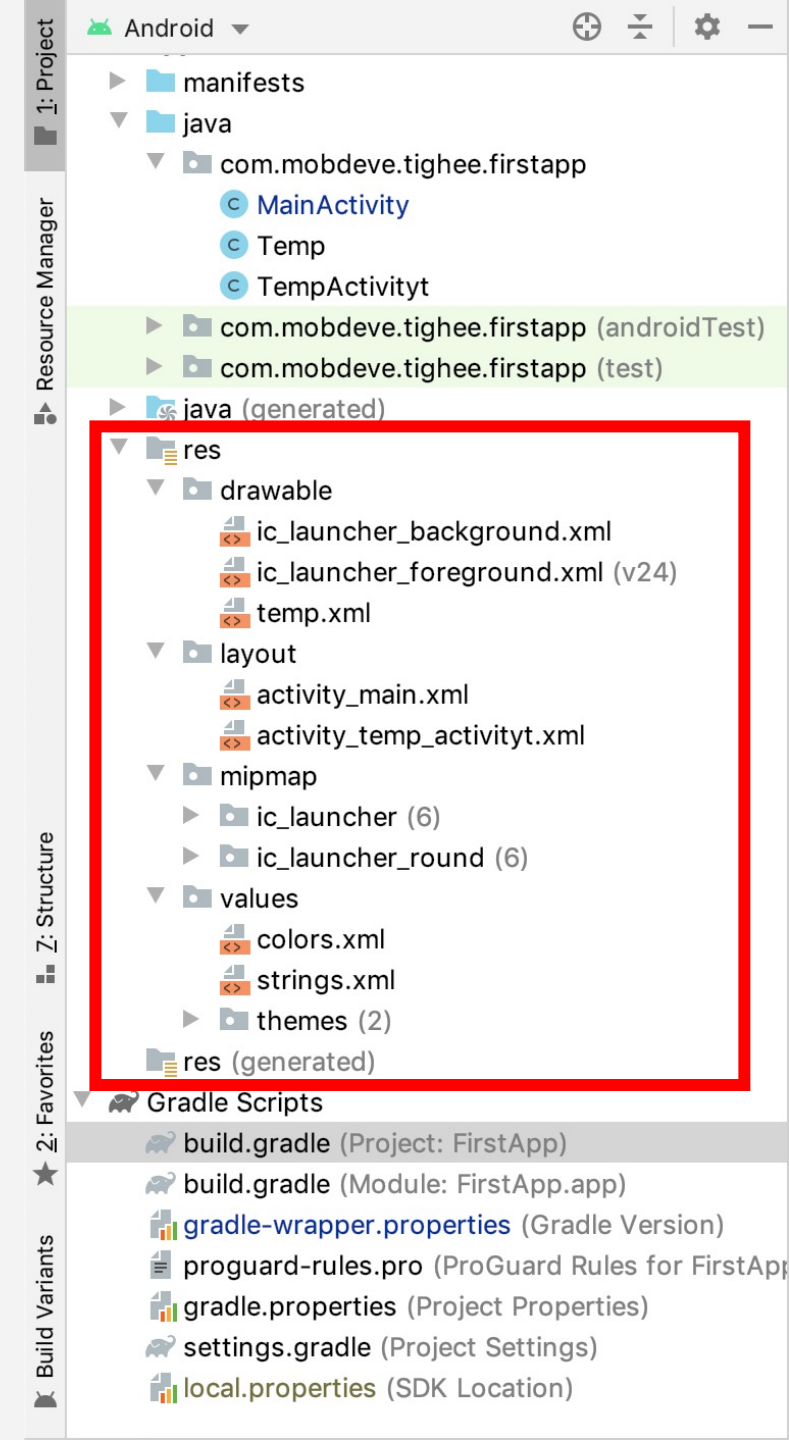
# Motivation

- What are **UI elements** with respect to Android?
- How do we **create** UI elements?



# What is a **resource**?

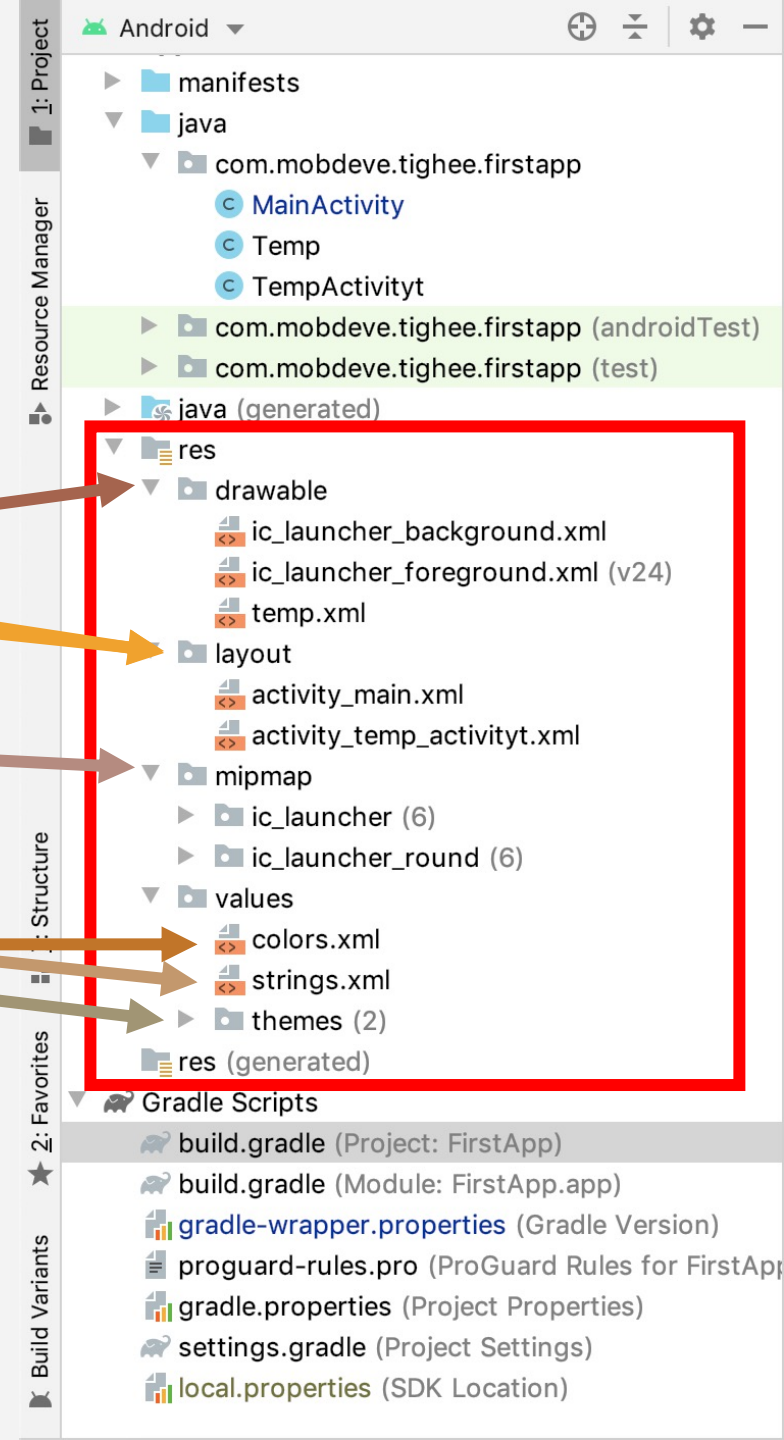
- Elements that are **external** to or separate from the app's behavior (i.e. code)
- Consist of
  - Images
  - XML files
- Recall: Found in the res folder





# What should be a resource?

- Layouts
- Controls (buttons, etc.)
- Images
- Strings
- Style and theming information



# UI components are written in XML files

The image illustrates the relationship between XML code and the resulting Android UI. The XML code on the left defines the layout structure, while the visual preview on the right shows how it renders on a device.

**XML Code:**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingLeft="16sp"
    android:paddingTop="32sp"
    android:paddingRight="16sp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/instructions_tv"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Please enter your details:"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/firstname_etv"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Enter first name"
        android:inputType="textPersonName" />

    <EditText
        android:id="@+id/lastname_etv"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Enter last name"
        android:inputType="textPersonName" />

    <Button
        android:id="@+id/add_btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ADD" />

</LinearLayout>
```

**Visual Preview:**

The visual preview shows the rendered UI. A red box highlights the entire form area. The text "Please enter your details:" is bold. Below it are two text input fields with hints "Enter first name" and "Enter last name". At the bottom right is a purple button labeled "ADD".

Red arrows indicate the mapping from the XML code to the visual elements:

- From `<TextView android:text="Please enter your details:" />` to the bold text "Please enter your details:".
- From `<EditText android:hint="Enter first name" />` to the first input field.
- From `<EditText android:hint="Enter last name" />` to the second input field.
- From `<Button android:text="ADD" />` to the "ADD" button.

# Views

- **View** is the **superclass** for visual interface elements
  - I.e. Elements you see onscreen
- Common Views include:
  - TextViews
  - EditTexts [input]
  - ImageViews
  - Buttons
- All behave the same, but subclasses have “extra” features



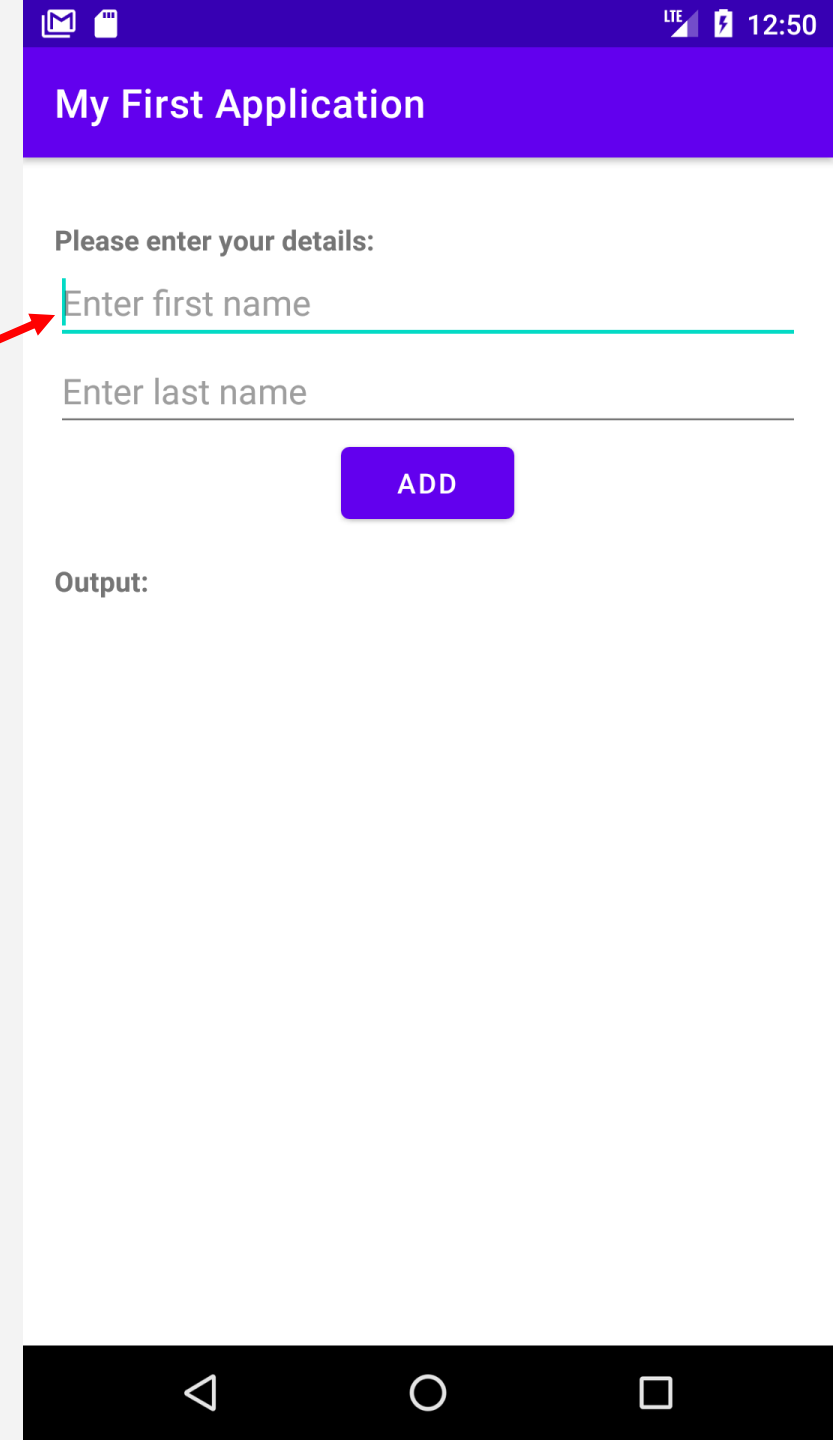
These are commonly called Widgets

But these are different from app widgets!

# Views

- All Views have...
  - Width and height
  - Padding
  - OnClick [listener] Not just buttons!
- As for subclasses with “extra” features...
  - EditTexts allow for hints
  - ImageViews don't have text

Hints disappear unlike  
the text attribute



# View Properties

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingLeft="16sp"
    android:paddingTop="32sp"
    android:paddingRight="16sp"
    tools:context=".MainActivity">
```

```
<TextView
    android:id="@+id/instructions_tv"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/instructions_tv"
    android:textStyle="bold" />
```

```
<EditText
    android:id="@+id/firstname_etv"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/first_edit_text"
    android:inputType="textPersonName" />
```

- Just like any object, **Views** have **attributes / properties**
- Common attributes you'll frequently modify:
  - id     **notice all ids start with @+id/**
  - layout\_width
  - layout\_height

# View Properties – Height / Width

- Both `layout_height` and `layout_width` are used to specify the size of a View with respect to a ViewGroup / layout
- Typically:
  - A specific value (e.g. 12dp)
  - `wrap_content` – wrap dimension based on content of View
  - `match_parent` – dimension matches that of the parent View

# View Properties – Height / Width

The screenshot displays the Android Studio interface with the XML editor on the left and the visual preview on the right. The XML code defines a `LinearLayout` containing three `TextView` elements. Red arrows point from specific XML attributes to callout boxes that explain their effect on the UI layout.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">

  <TextView
    android:id="@+id/first_example_tv"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:text="@string/random_text" />

  <TextView
    android:id="@+id/second_example_tv"
    android:layout_width="wrap_content"
    android:layout_height="100dp"
    android:text="@string/random_text" />

  <TextView
    android:id="@+id/third_example_tv"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/random_text" />

</LinearLayout>
```

**first\_example\_tv...**  
width is strictly 100dp, forcing the text to wrap the text that would exceed 100dp

**second\_example\_tv...**  
height is forced to 100dp, creating space below

**third\_example\_tv...**  
width is forced to match the width of parent -- LinearLayout

# View Properties – Units

- **dp** is a “**d**ensity-independent **p**ixel”
  - On a 160-dpi (dots-per-inch) screen, 1 dp equals 1px (pixel)
  - As dpi increases, the number of pixels per dp increases
- **px** is an actual screen **p**ixel
- **sp** is a “**s**cale-independent **p**ixel”
  - Like dp but scaled by the system’s font preference
  - 1sp will cover more dp in a device set at larger font
- **pt** is 1/72 of an inch of the physical screen

Q: Which unit should normally be used?

A:  
sp and dp are good. sp usually for text, dp used for when you don’t want the views be to be resized  
px and pt are allowed, but are discouraged



How about images?

# Before hand...

- Let's understand that there are two types of image assets:
  - **Drawables**
    - Contains graphics (PNG, JPEG, etc.)
    - Can be created through File > New > Image Asset or by dragging/saving images into the project's drawable folder
  - **Mipmap**
    - Launcher icon files in different resolutions
    - Can be created through File > New > Image Asset

There are also multiple drawable / mipmap folders that map image assets to a phone's resolution

If an image is too high-res for a device, an error will be thrown and the app will crash ☹

# Adding Image Files to ImageViews

The screenshot illustrates the process of adding image files to ImageViews in an Android application. The interface is divided into several panels:

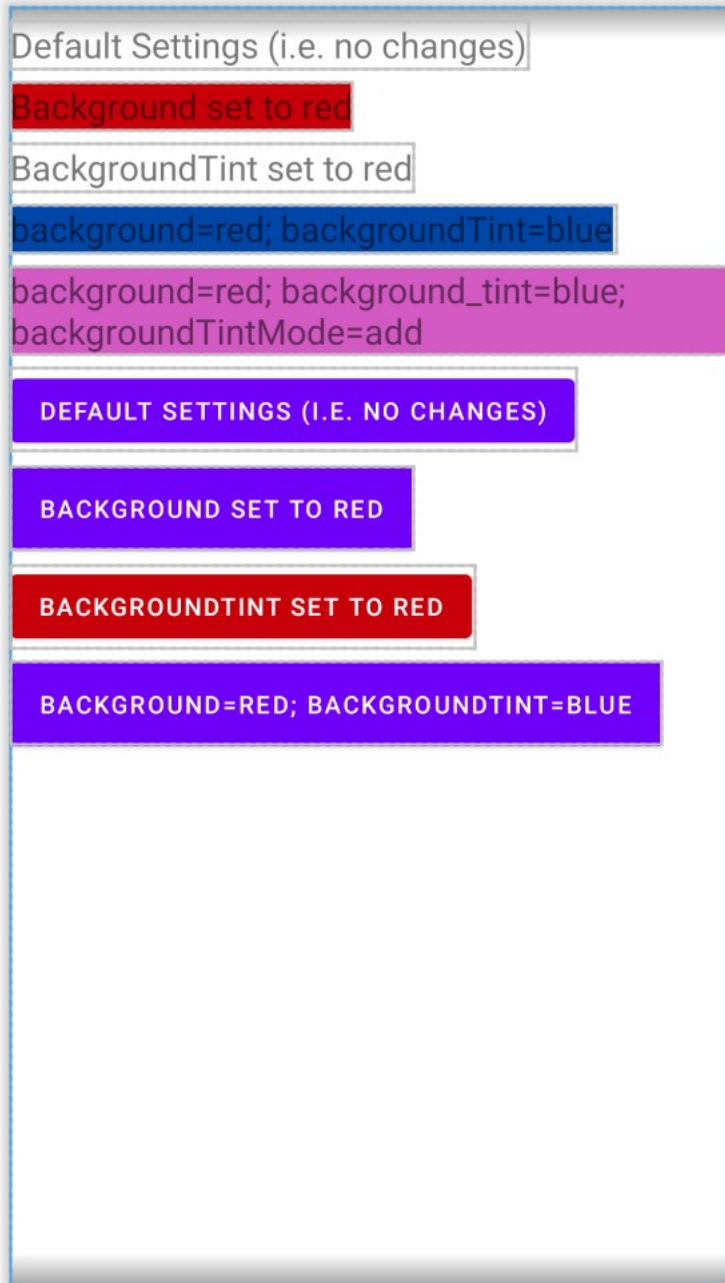
- Resource Manager (Left):** Shows the project structure. Under the `res` directory, the `drawable` folder contains `ic_launcher_background.xml`, `ic_launcher_foreground.xml (v24)`, `paolul_icon.jpg (v24)`, and `super_bayani_icon.jpg (v24)`. The `values` folder contains `colors.xml`, `strings.xml`, and `themes (2)`. The `res (generated)` folder is also visible.
- Code Editor (Center):** Displays the XML layout file. The `<LinearLayout>` tag is defined with attributes for namespace, app, tools, width, height, and orientation. Two `<ImageView>` tags are present:
  - Line 10: `<ImageView android:id="@+id/imageView3" android:layout_width="wrap_content" android:layout_height="wrap_content" tools:srcCompat="@drawable/paolul_icon" />`
  - Line 15: `<ImageView android:id="@+id/imageView4" android:layout_width="wrap_content" android:layout_height="wrap_content" app:srcCompat="@drawable/super_bayani_icon" />`
- Right Panel:** Contains a toolbar with icons for text, buttons, widgets, layouts, containers, helpers, Google, and legacy. Below the toolbar is the **Component Tree**, which shows the hierarchy: `LinearLayout (vertical...)` containing `imageView3` and `imageView4`. Both image views have a yellow warning icon next to them.
- Visual Preview (Right):** Shows a visual representation of the layout. It features a large circular logo with the word "Lulu" and a smaller image of a person wearing sunglasses.

Arrows indicate the flow of data:

- A green arrow points from the `paolul_icon.jpg (v24)` file in the Resource Manager to the `tools:srcCompat="@drawable/paolul_icon"` attribute in the `imageView3` tag.
- A red arrow points from the `super_bayani_icon.jpg (v24)` file in the Resource Manager to the `app:srcCompat="@drawable/super_bayani_icon"` attribute in the `imageView4` tag.
- A green arrow points from the `tools:srcCompat="@drawable/paolul_icon"` attribute to the visual preview of the "Lulu" logo.
- A red arrow points from the `app:srcCompat="@drawable/super_bayani_icon"` attribute to the visual preview of the person wearing sunglasses.

# View Properties – Background + Tint

- When it comes to coloring backgrounds, things might not seem so straight forward
- The ~~two~~three attributes you'd want to play around with are:
  - **background** -> refers to a drawable or color
  - **backgroundTint** -> tint applied to the background
    - **backgroundTintMode** -> blending applied to the tint
      - Add, src\_in, src\_over, screen, src\_atop, multiply



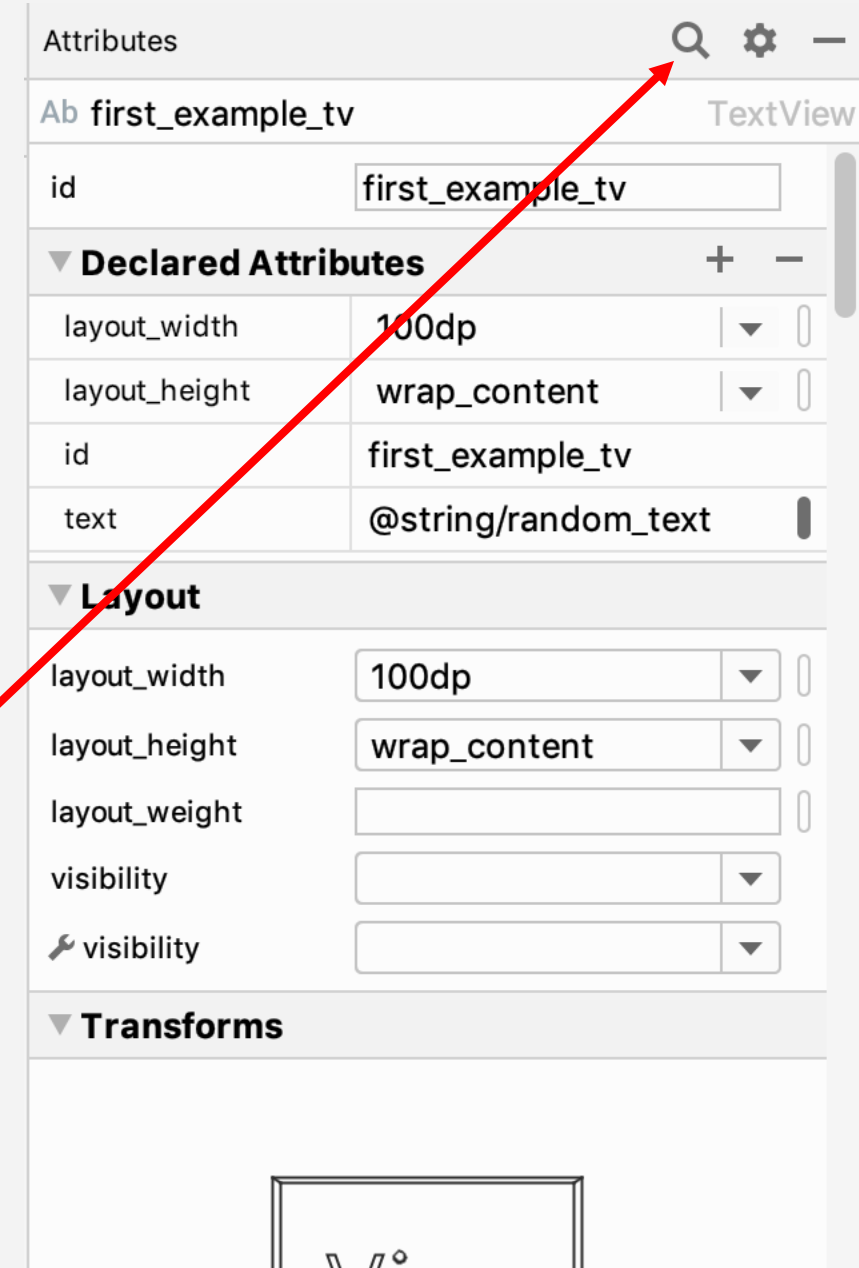
Observe the layout to the left...

- For TextViews (at most other Views):
  - Background sets a color
  - BackgroundTint only has an effect when there's a background
    - By default, there is no background; hence, why the 3<sup>rd</sup> TextView has no color, but the 4<sup>th</sup> is changed to blue
  - BackgroundTintMode blends background and tint
- For Buttons:
  - By default, uses the app's colorPrimary
  - Background, by default, is set to @empty and changing it removes the feathered style
  - BackgroundTint changes the color

# View Properties

- There are many more properties you'll interact with, like...
  - Margins, Padding
  - Text, Text size, Text style
  - Visibility, Enabled
- Take some time to read the **documentation** or use the **search bar** to look for attributes you think might exist

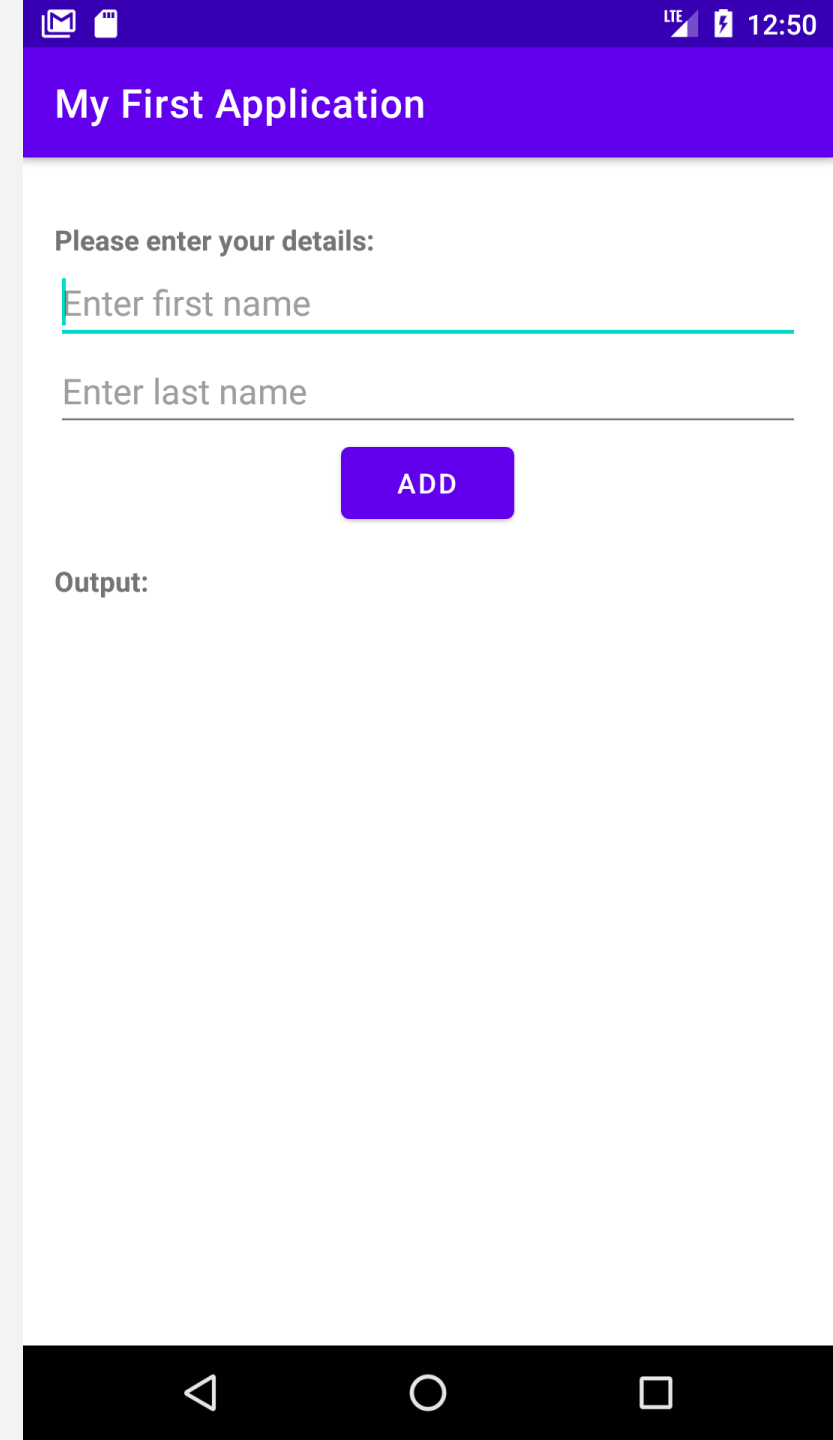
Experimentation is key!



Any questions so far?

# ViewGroups (or Layouts)

- A **ViewGroup** is a View that can contain other “child” Views
  - It can also contain other ViewGroups since its also a View
- Common ViewGroups include:
  - LinearLayout [horizontal / vertical]
  - ConstraintLayout

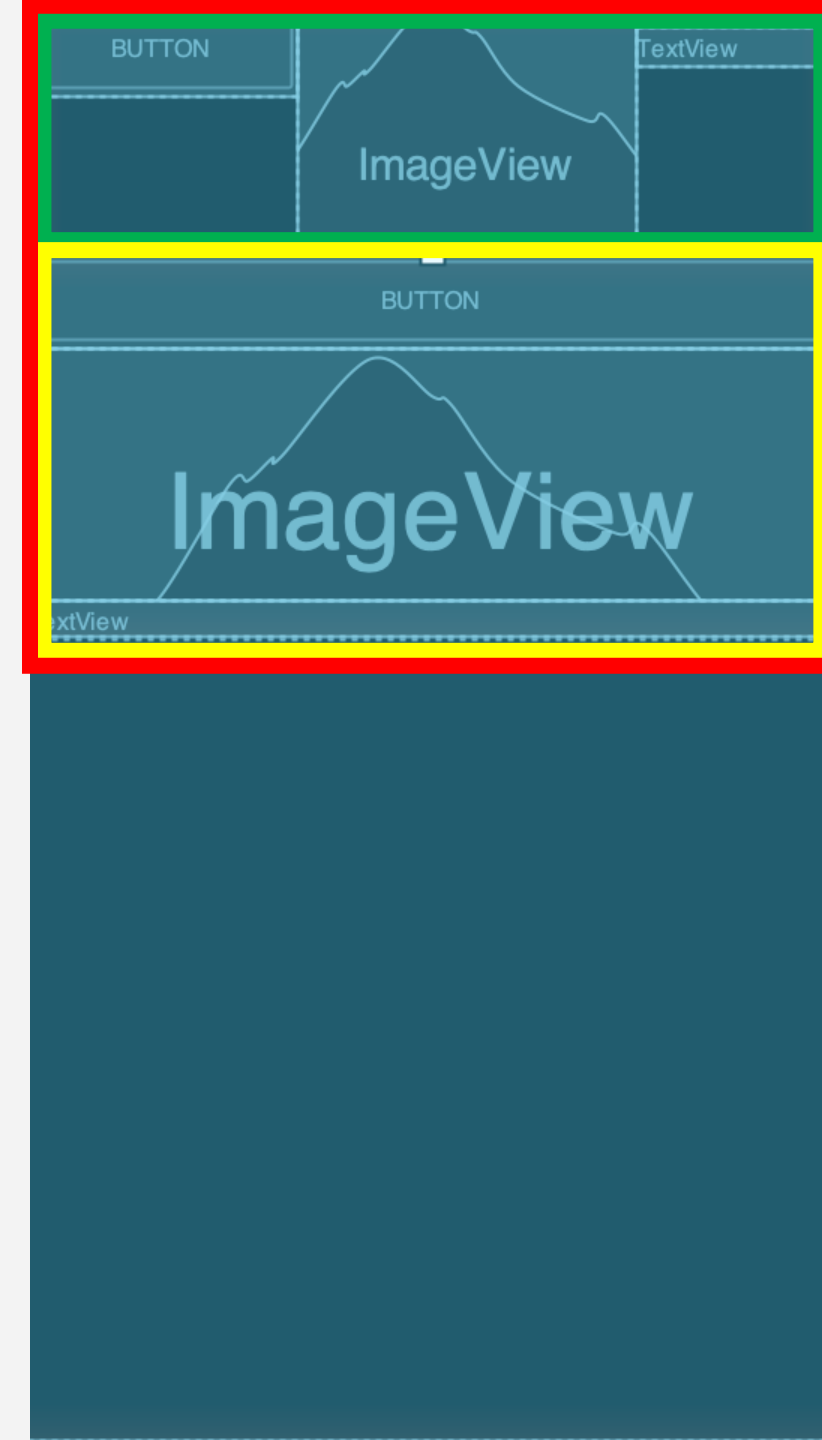
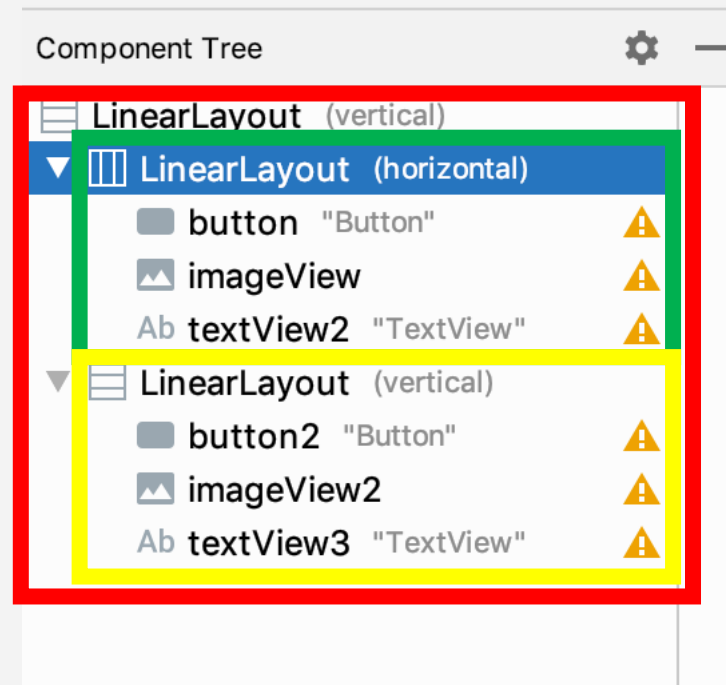




# ViewGroups (or Layouts)

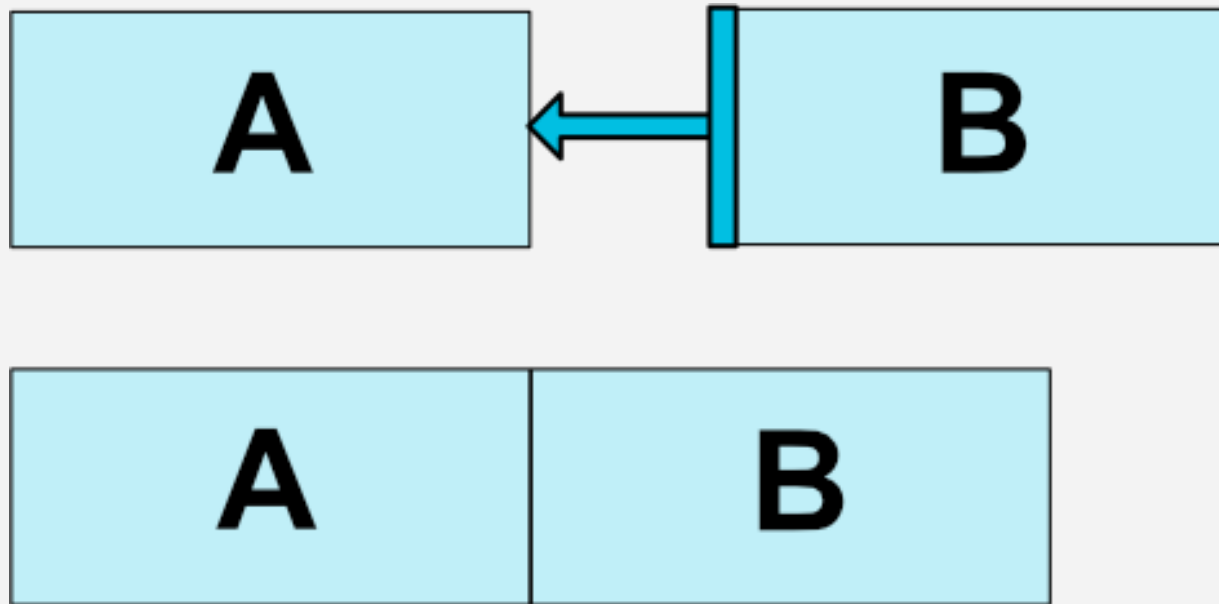
- **LinearLayouts** order Views in a line / single direction
  - **orientation** specifies the direction
    - horizontal
    - vertical

To have proper spacing, you'd want to play around with layout\_weight, margin, padding of either the parent or child views

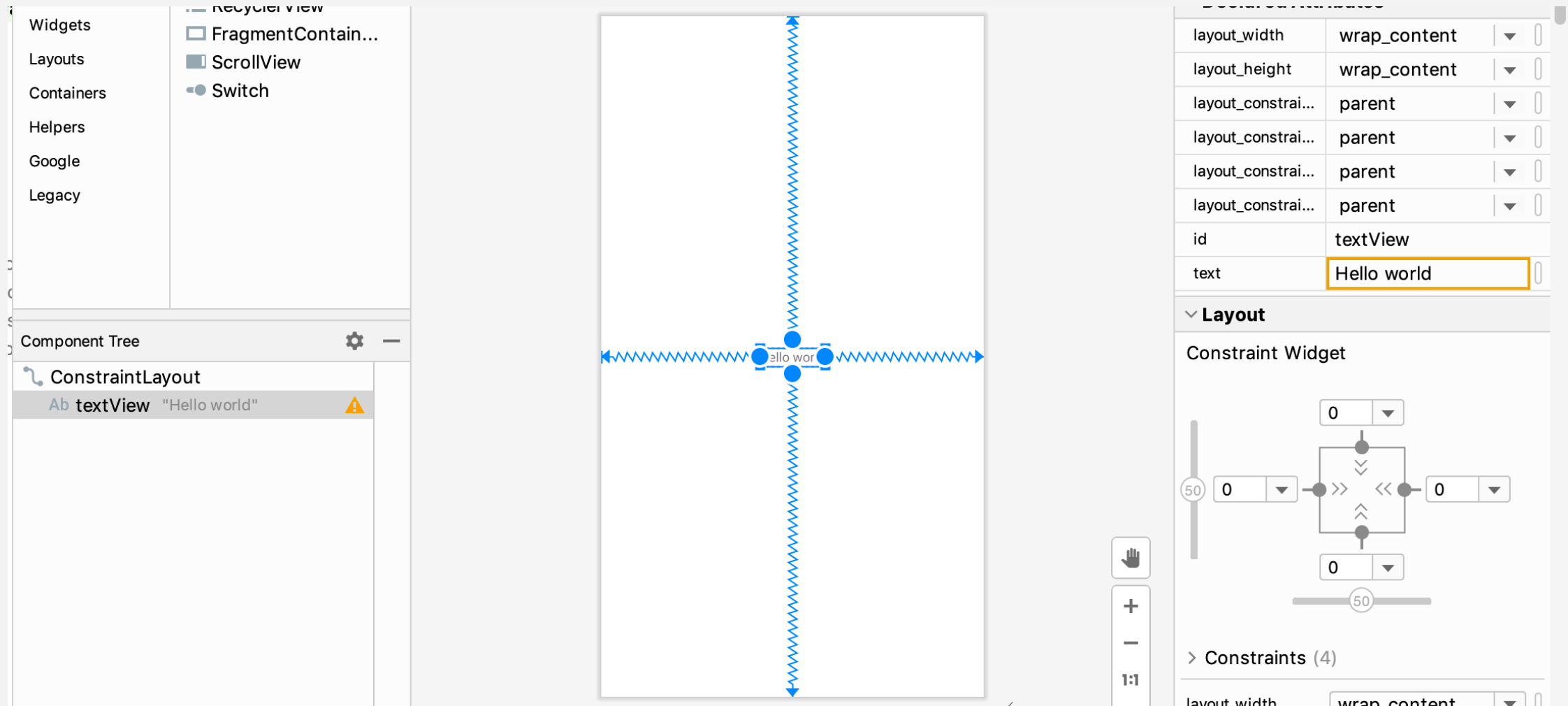


# ViewGroups (or Layouts)

- **ConstraintLayout** sets relationships between Views using constraints or relative positioning



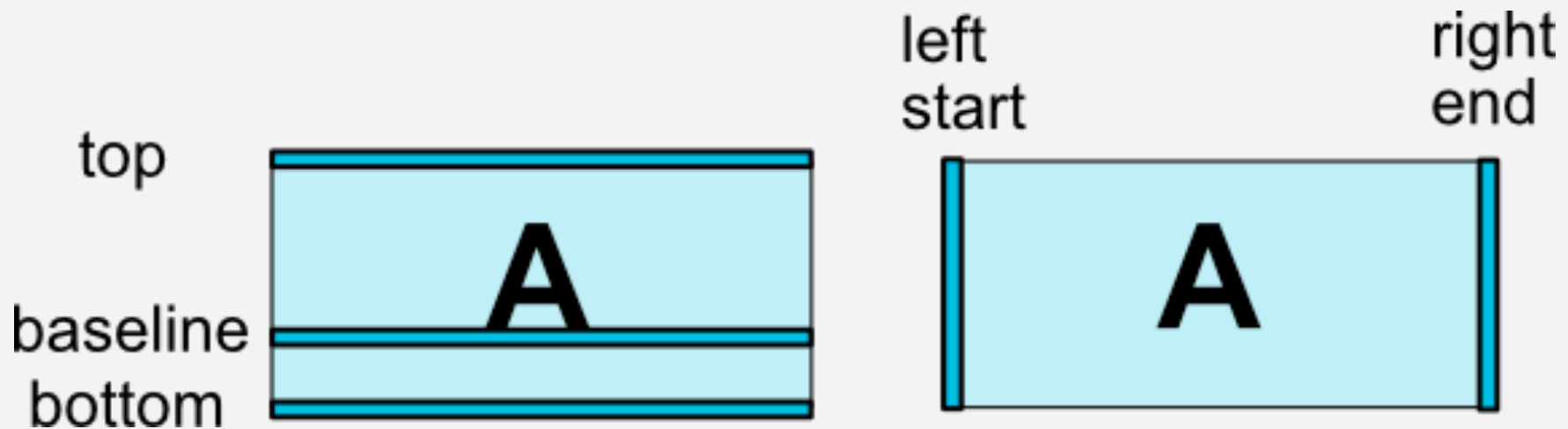
In this example, we see a TextView constrained to its parent from all sides



By "pulling" it with equal force from both sides, the TextView is centered both vertically and horizontally

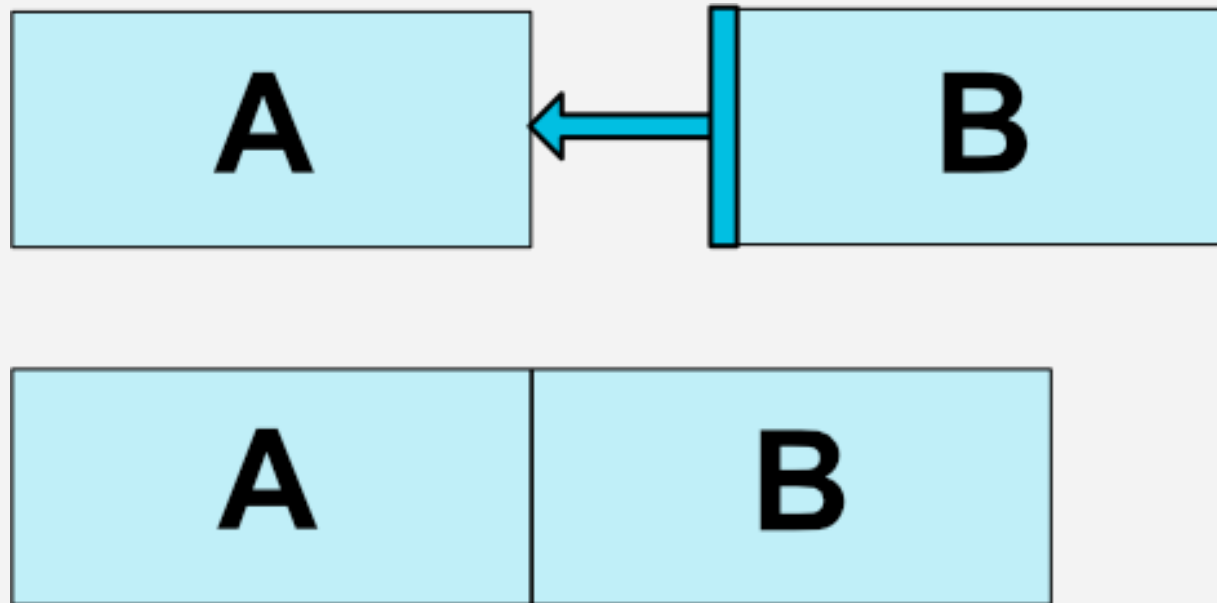
# When we talk about “**sides**”...

- We actually refer to different **positions** of a view



start and end are preferred over left and right, as this supports RTL (right to left) layouts

Q: So in our previous example, how is B constrained to A?

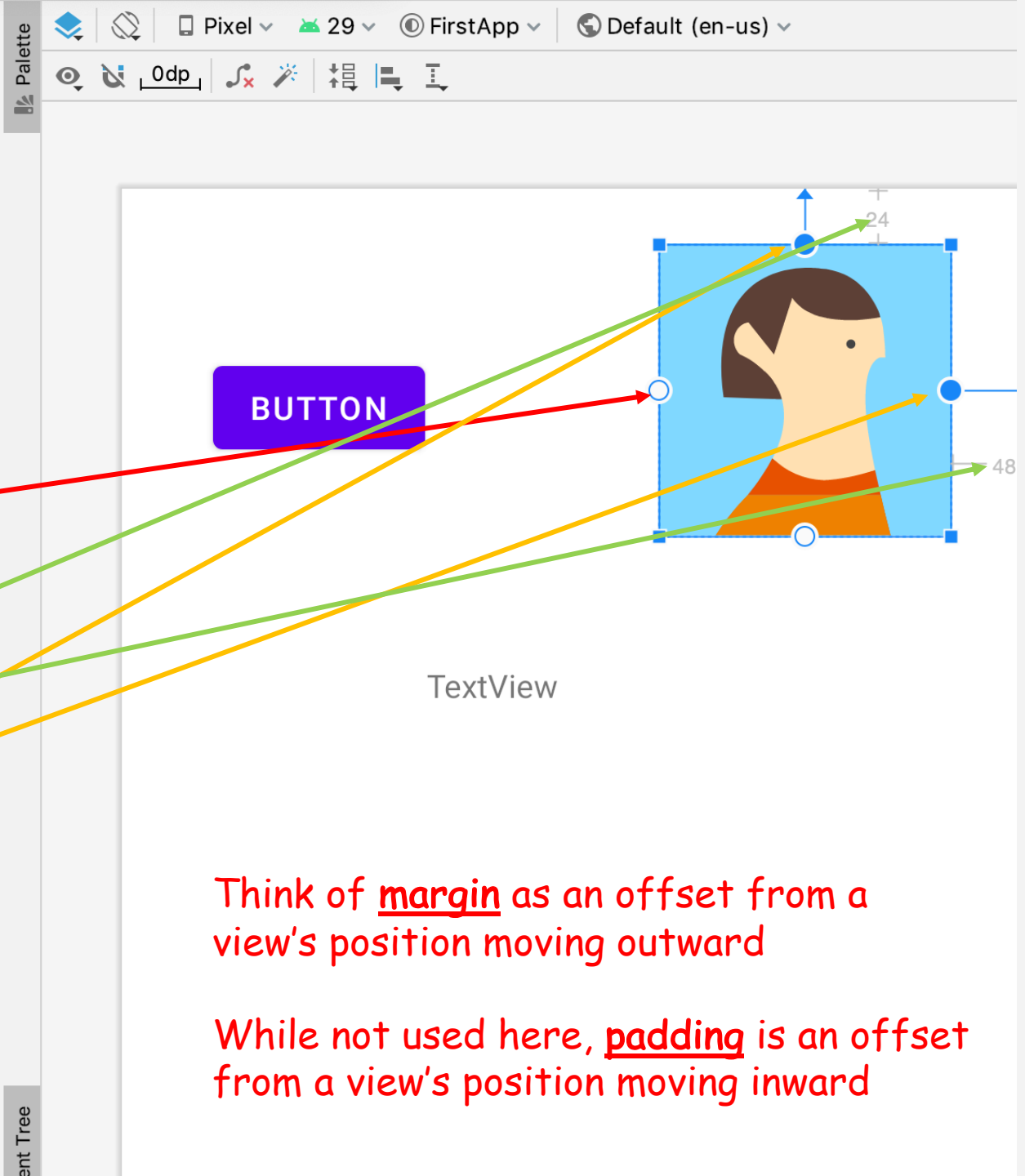


**A: B's start is constrained to A's end**

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent">
8
9     <Button
10         android:id="@+id/button4"
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:layout_marginStart="40dp"
14         android:layout_marginTop="72dp"
15         android:text="Button"
16         app:layout_constraintStart_toStartOf="parent"
17         app:layout_constraintTop_toTopOf="parent" />
18
19     <ImageView
20         android:id="@+id/imageView4"
21         android:layout_width="wrap_content"
22         android:layout_height="wrap_content"
23         android:layout_marginTop="24dp"
24         android:layout_marginEnd="48dp"
25         app:layout_constraintEnd_toEndOf="parent"
26         app:layout_constraintTop_toTopOf="parent"
27         tools:srcCompat="@tools:sample/avatars" />
28
29     <TextView
30         android:id="@+id/textView4"
31         android:layout_width="wrap_content"
32         android:layout_height="wrap_content"
33         android:layout_marginTop="88dp"
34         android:layout_marginEnd="44dp"
35         android:text="TextView"
36         app:layout_constraintEnd_toStartOf="@+id/imageView4"
37         app:layout_constraintTop_toBottomOf="@+id/button4" />
38

```



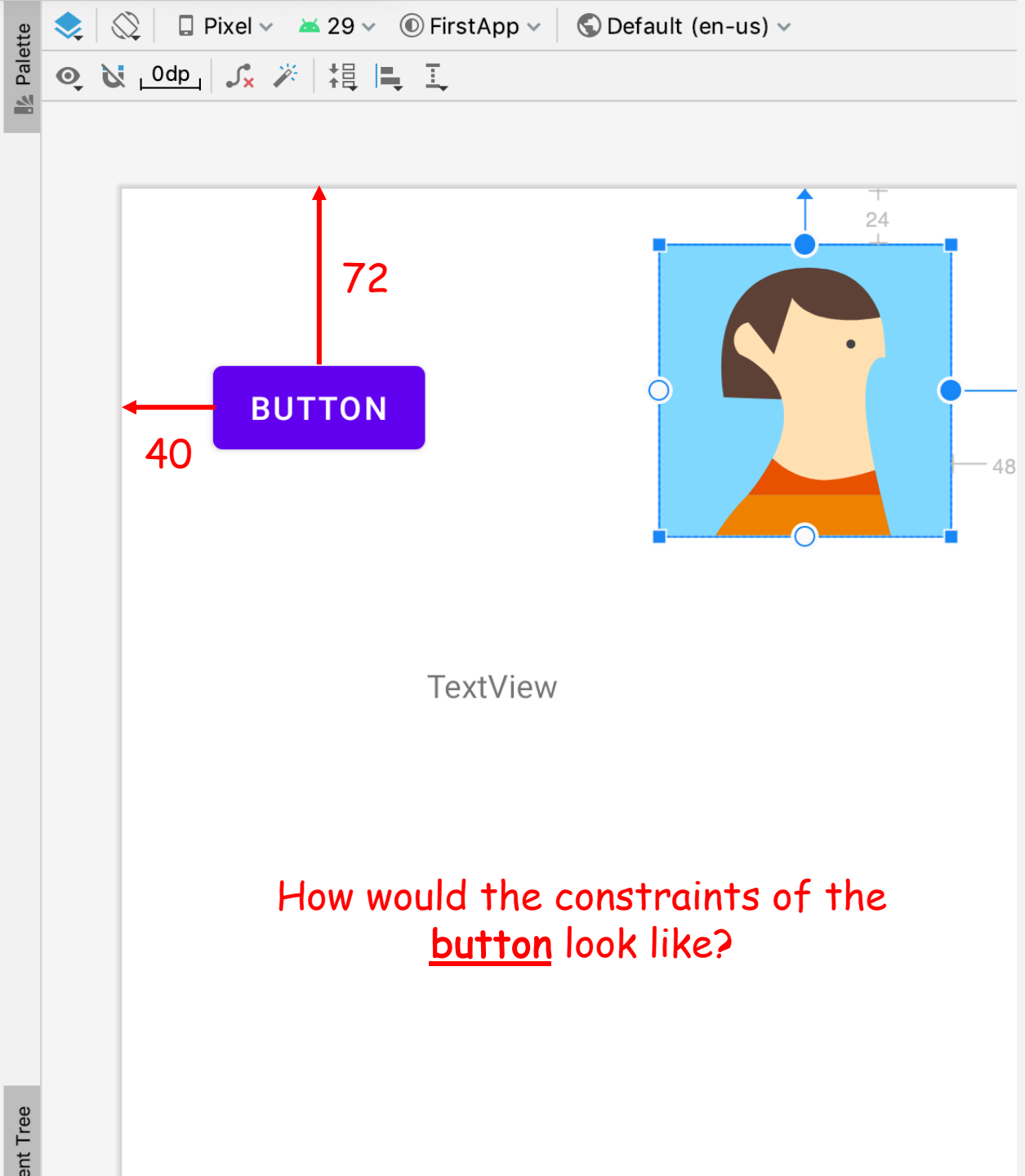
Think of margin as an offset from a view's position moving outward

While not used here, padding is an offset from a view's position moving inward

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent">
8
9     <Button
10         android:id="@+id/button4"
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:layout_marginStart="40dp"
14         android:layout_marginTop="72dp"
15         android:text="Button"
16         app:layout_constraintStart_toStartOf="parent"
17         app:layout_constraintTop_toTopOf="parent" />
18
19     <ImageView
20         android:id="@+id/imageView4"
21         android:layout_width="wrap_content"
22         android:layout_height="wrap_content"
23         android:layout_marginTop="24dp"
24         android:layout_marginEnd="48dp"
25         app:layout_constraintEnd_toEndOf="parent"
26         app:layout_constraintTop_toTopOf="parent"
27         tools:srcCompat="@tools:sample/avatars" />
28
29     <TextView
30         android:id="@+id/textView4"
31         android:layout_width="wrap_content"
32         android:layout_height="wrap_content"
33         android:layout_marginTop="88dp"
34         android:layout_marginEnd="44dp"
35         android:text="TextView"
36         app:layout_constraintEnd_toStartOf="@+id/imageView4"
37         app:layout_constraintTop_toBottomOf="@+id/button4" />
38

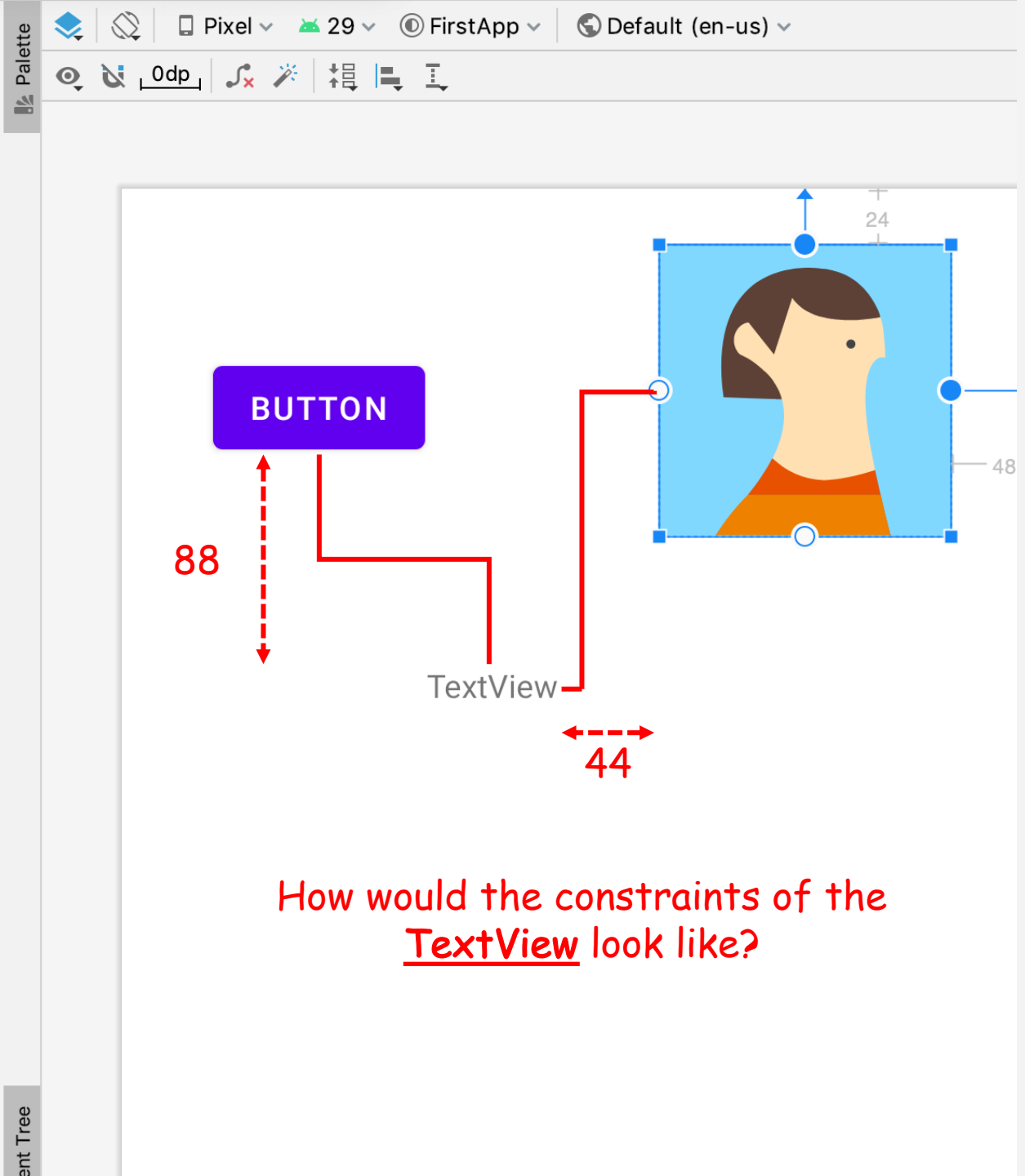
```



```

1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent">
8
9     <Button
10         android:id="@+id/button4"
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:layout_marginStart="40dp"
14         android:layout_marginTop="72dp"
15         android:text="Button"
16         app:layout_constraintStart_toStartOf="parent"
17         app:layout_constraintTop_toTopOf="parent" />
18
19     <ImageView
20         android:id="@+id/imageView4"
21         android:layout_width="wrap_content"
22         android:layout_height="wrap_content"
23         android:layout_marginTop="24dp"
24         android:layout_marginEnd="48dp"
25         app:layout_constraintEnd_toEndOf="parent"
26         app:layout_constraintTop_toTopOf="parent"
27         tools:srcCompat="@tools:sample/avatars" />
28
29     <TextView
30         android:id="@+id/textView4"
31         android:layout_width="wrap_content"
32         android:layout_height="wrap_content"
33         android:layout_marginTop="88dp"
34         android:layout_marginEnd="44dp"
35         android:text="TextView"
36         app:layout_constraintEnd_toStartOf="@+id/imageView4"
37         app:layout_constraintTop_toBottomOf="@+id/button4" />
38

```





# [Modified] View Properties – Height / Width

- Both `layout_height` and `layout_width` are used to specify the size of a View with respect to a ViewGroup / layout
- Typically:
  - A specific value (e.g. 12dp)
  - `wrap_content` – wrap dimension based on content of View
  - `match_parent` – dimension matches that of the parent View
    - Discouraged when using a ConstraintLayout
  - 0dp (`match_constraints`) – takes all available space based on constraints

There's actually a lot more to discover when it comes to styling with ConstraintLayouts

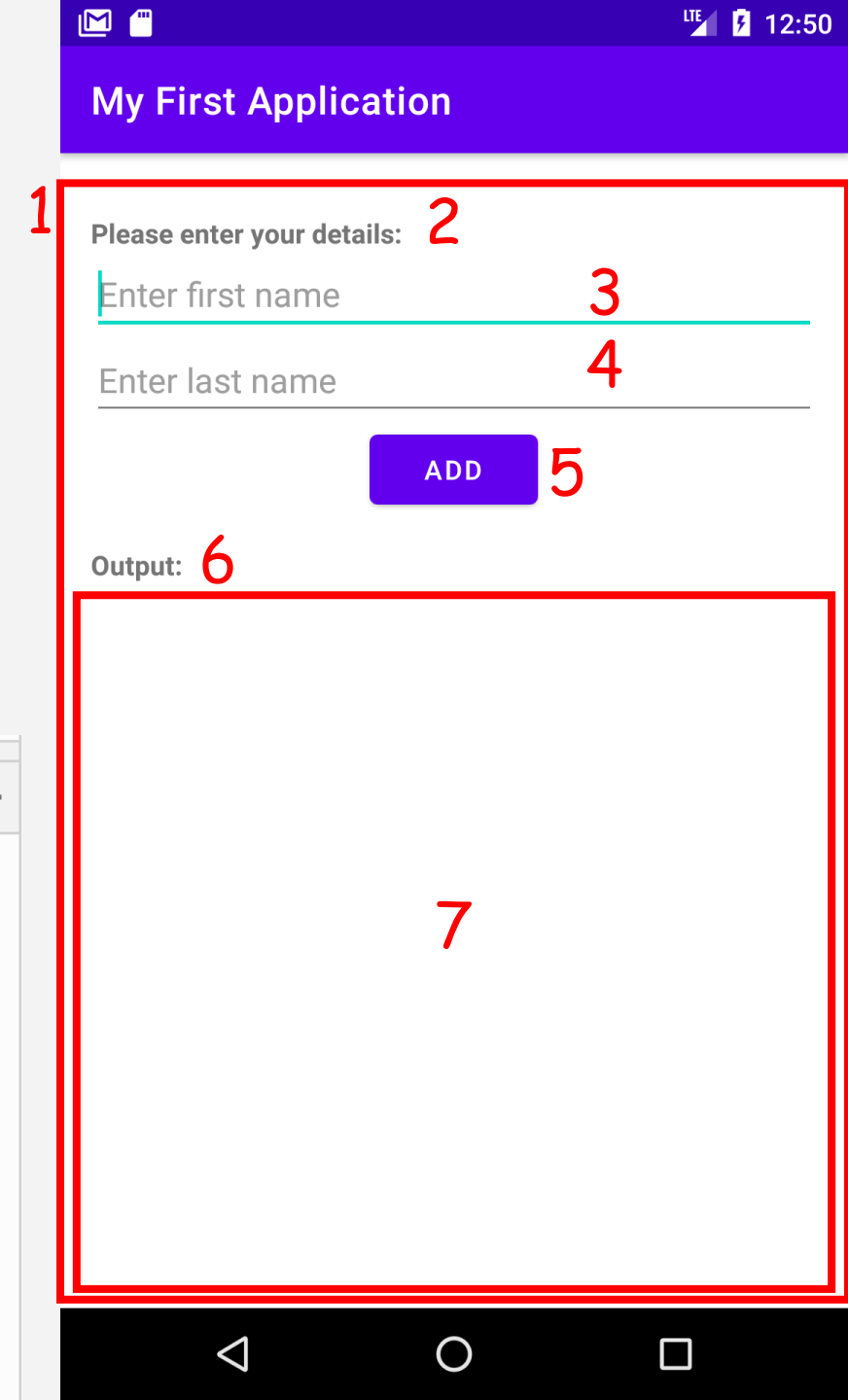
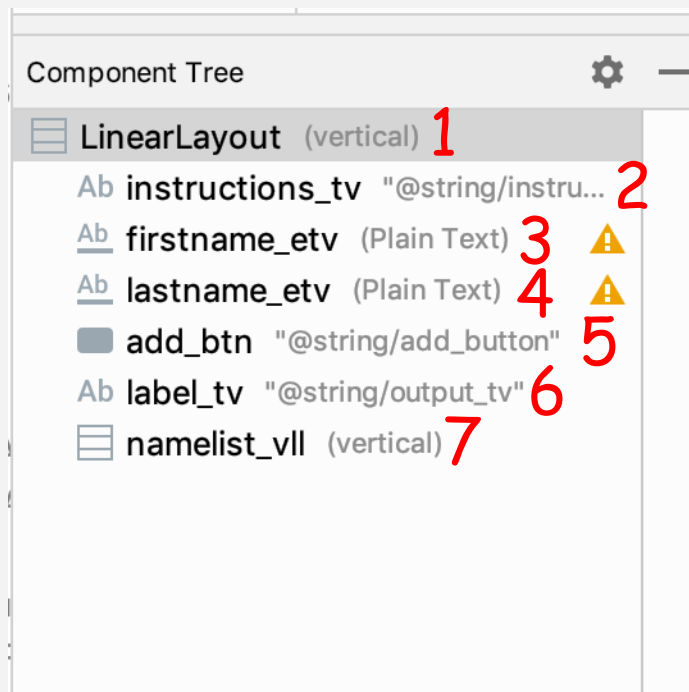
For more info, check out

<https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout>

# ViewGroups (or Layouts)

- Q: Can you guess the number of layouts use in the screenshot to the side?
  - There are actually two LinearLayouts!
- The arrangement corresponds to a tree structure known as the **Component tree**

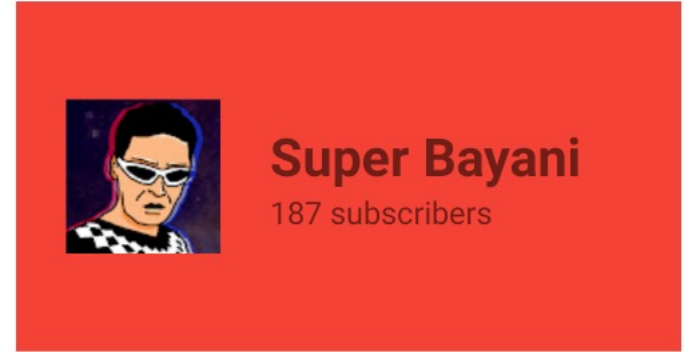
In a ConstraintLayout, order doesn't matter



Any questions? 😊

# Non-graded Exercise

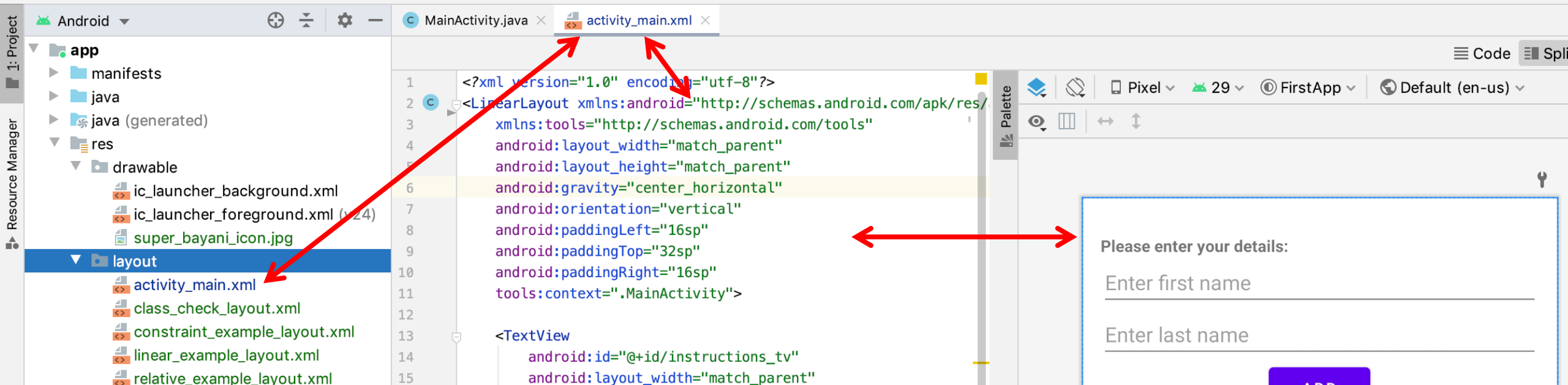
- Try to recreate the layout file shown on the right side
- There are different ways to solve this, so experiment with different ideas based on what you've learned!
- There is also a Canvas assignment for this



<https://www.youtube.com/channel/UCGn0ep0f0tjnegstKIlVgqg>

# Clarification: Layouts

- A **Layout file** is an XML file composed of Views and ViewGroups (Layout)
  - Has a root ViewGroup
  - Usually associated to an Activity but not required to be



# Clarification: Creating UI / Layouts

```
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   xmlns:tools="http://schemas.android.com/tools"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent"
7   android:gravity="center_horizontal"
8   android:orientation="vertical"
9   android:paddingLeft="16sp"
10  android:paddingTop="32sp"
11  android:paddingRight="16sp"
12  tools:context=".MainActivity">
```

Its common to just use the Design View to generate XML, but be careful of what you select

```
14 <TextView
15   android:id="@+id/instructions_tv"
16   android:layout_width="match_parent"
17   android:layout_height="wrap_content"
18   android:text="Please enter your details:"
19   android:textStyle="bold" />
```

```
21 <EditText
22   android:id="@+id/firstname_etv"
23   android:layout_width="match_parent"
24   android:layout_height="wrap_content"
25   android:ems="10"
26   android:hint="Enter first name"
27   android:inputType="textPersonName" />
```

```
29 <EditText
30   android:id="@+id/lastname_etv"
31   android:layout_width="match_parent"
32   android:layout_height="wrap_content"
33   android:ems="10"
```

The attribute *ems* actually doesn't need to be here because we're using *match\_parent* as the width

Please enter your details:

ADD

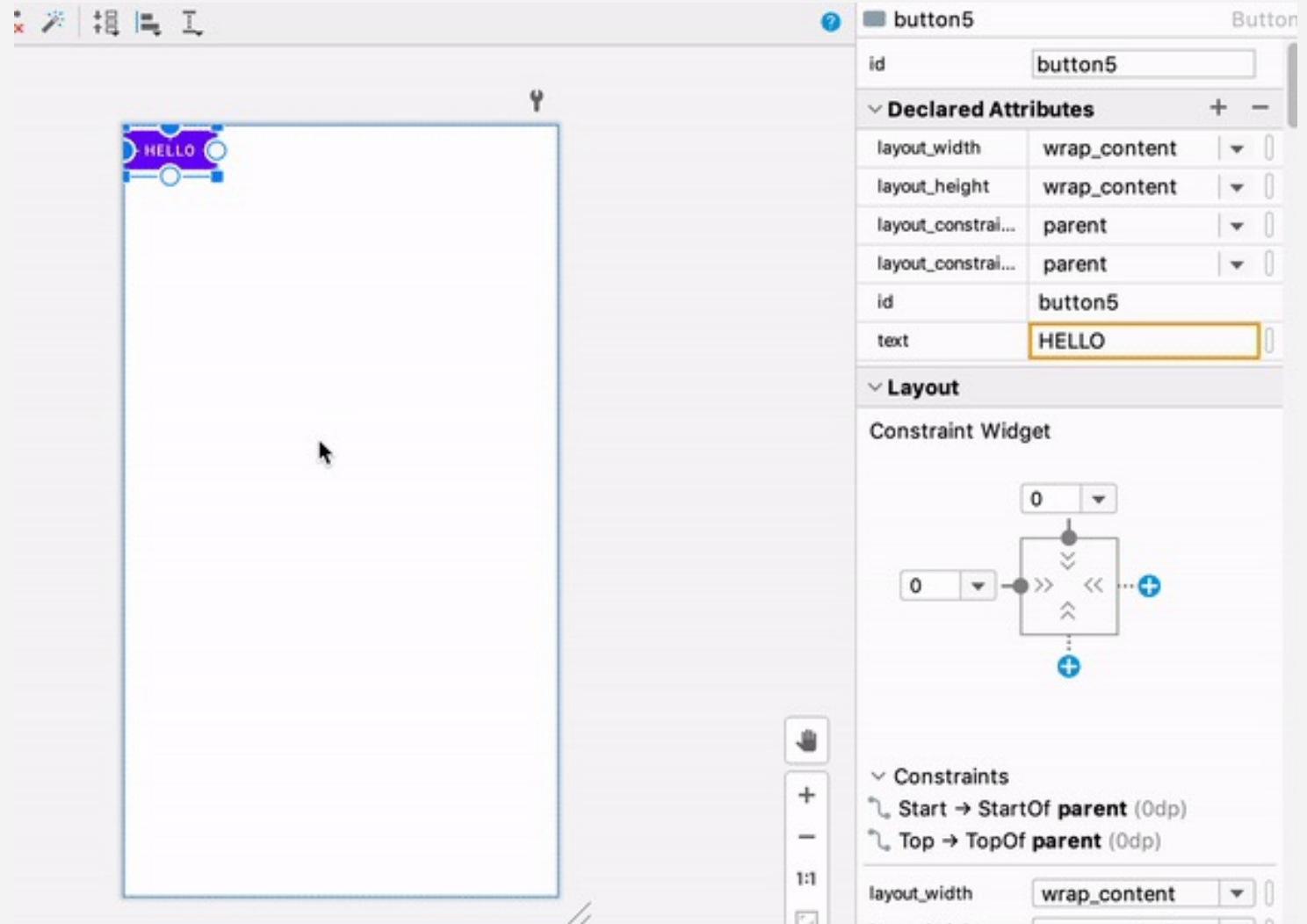
Output:

Writing out the layout will give you a better understanding of how to structure your Views

But for our class, you're allowed to use either method to construct your UI

# Clarification: Moving Views in a ConstraintLayout

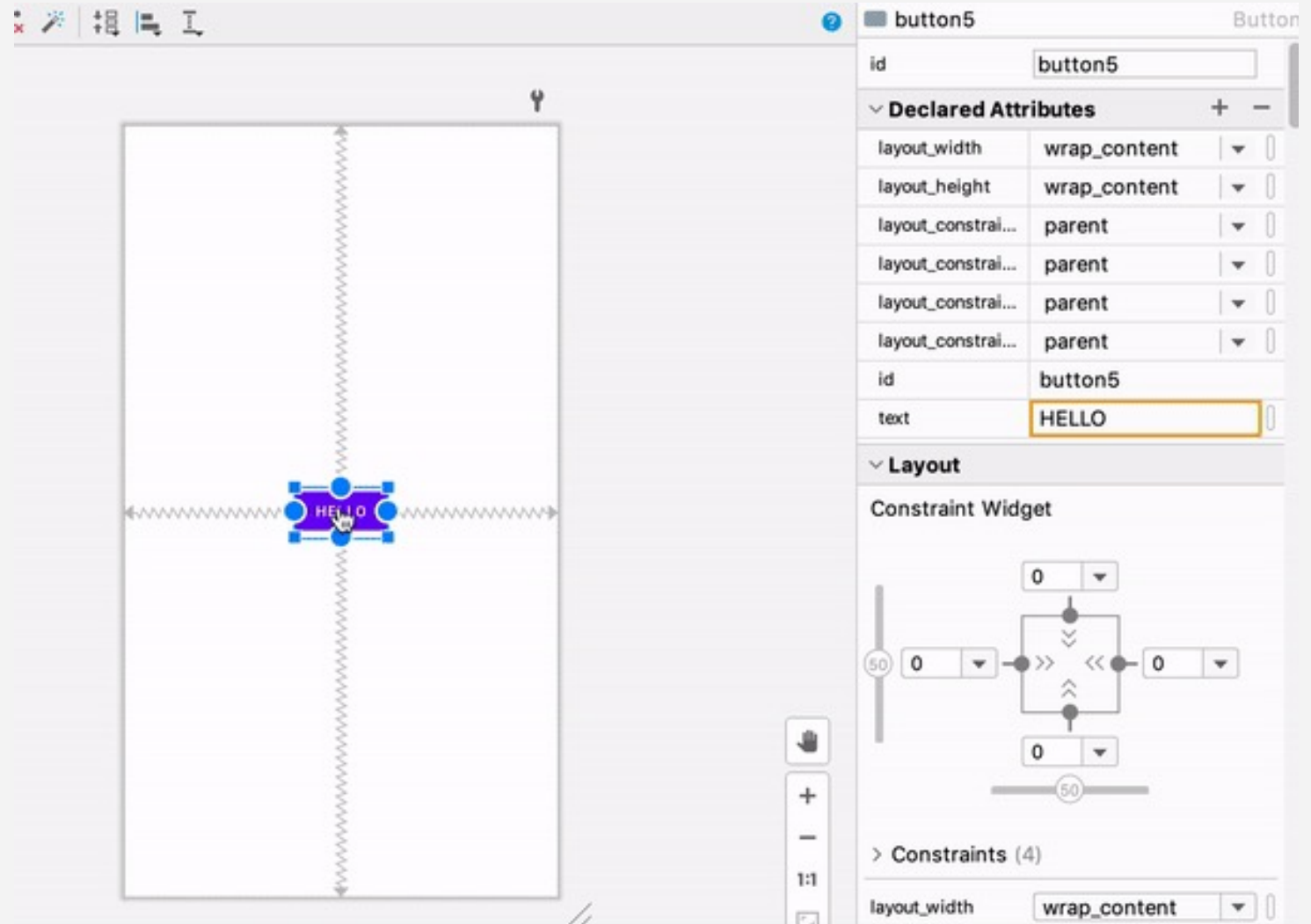
- In the example to the side, we see moving the button in design mode makes changes to the margin





# Clarification: Moving Views in a ConstraintLayout

- In this next example to the side, we see moving the fully constrained button in design mode makes changes to the bias



Any questions? 😊

# Notes

- There is an Android Project in Module 2a where you can access all the layouts shown in this slide
- We'll release exercise #1 after next session
  - Look to clarify concerns next meeting
- For the next lesson, we'll tackle the 2<sup>nd</sup> part of UI Basics: Dynamic creation of views

Thanks everyone!

See you next  
meeting! 😊

