AD340 Mobile Application Development

Week 02

Outline

- Jeff Eng
- Android Studio Tour
- Create your first Android Studio Project
- Device Manager & Virtual Device
- Layouts
- App Resources
- Activity

Jeff Eng

Outreach and student success manager

has been an advisor within the advising office for a number of years

help recruit and advise AD/CS students

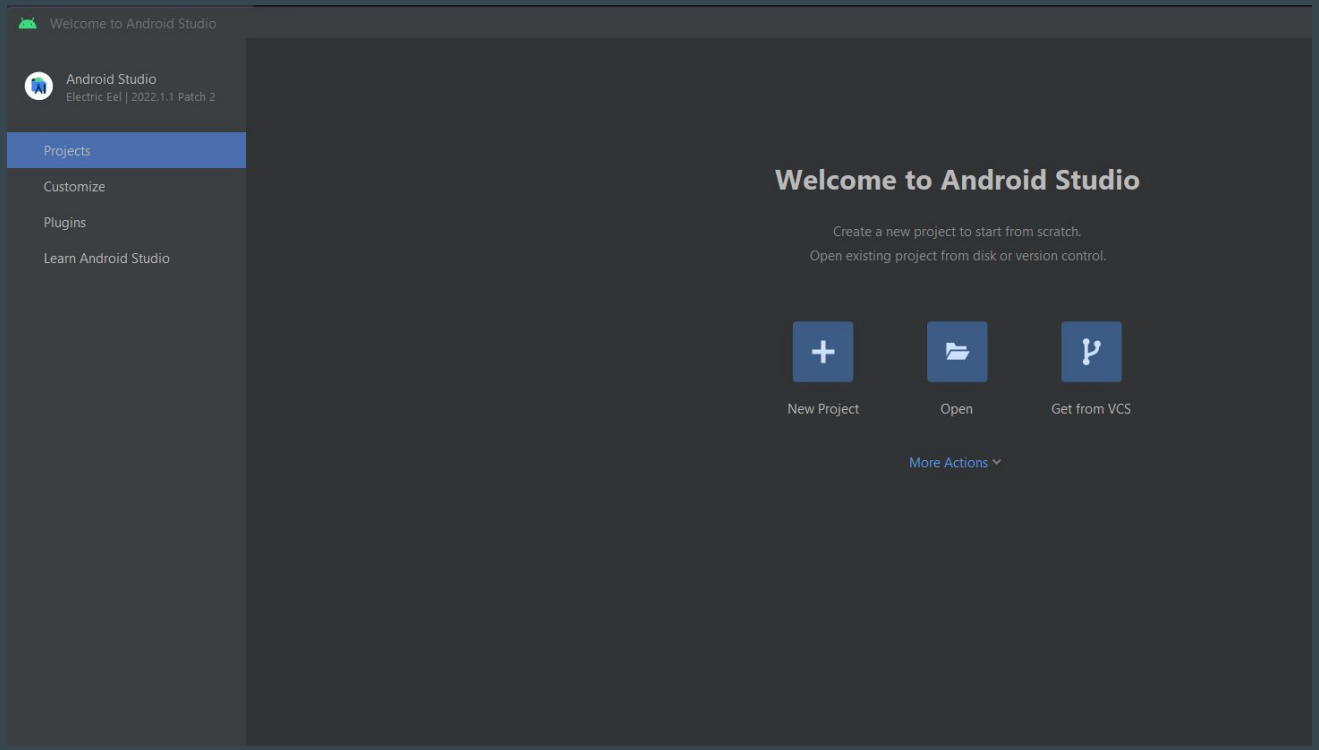
Questions

- What do you do when you are not working?
- What is the best piece of advice you've ever been given?

Android Studio

- Official IDE for building Android apps
- It's built on IntelliJ IDEA
- Has lots of features to make it faster and easier to develop Android apps.
- https://developer.android.com/studio

Open Android Studio

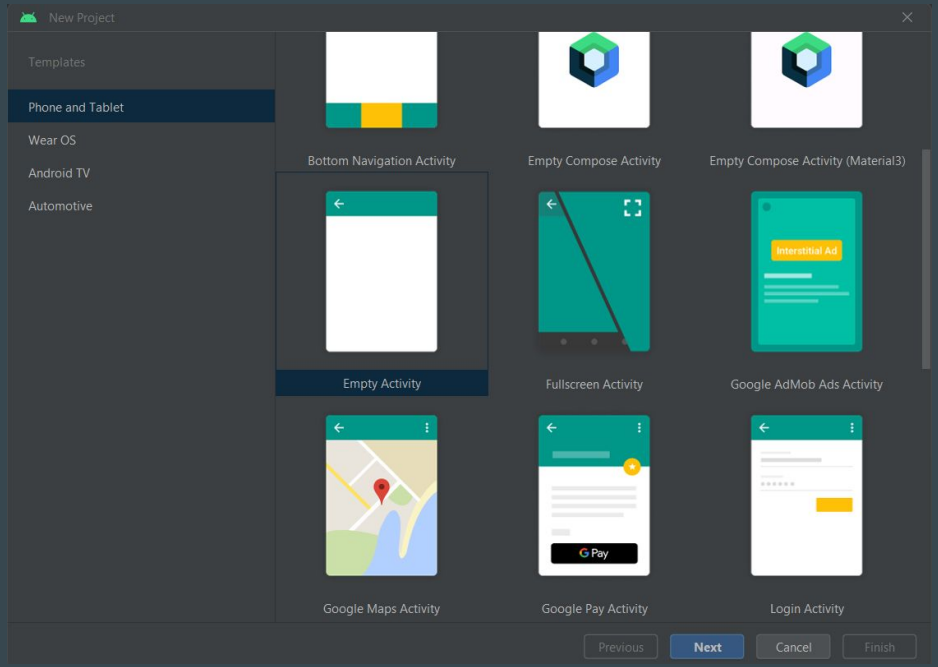


Create new project

lots of templates here that contain starter projects

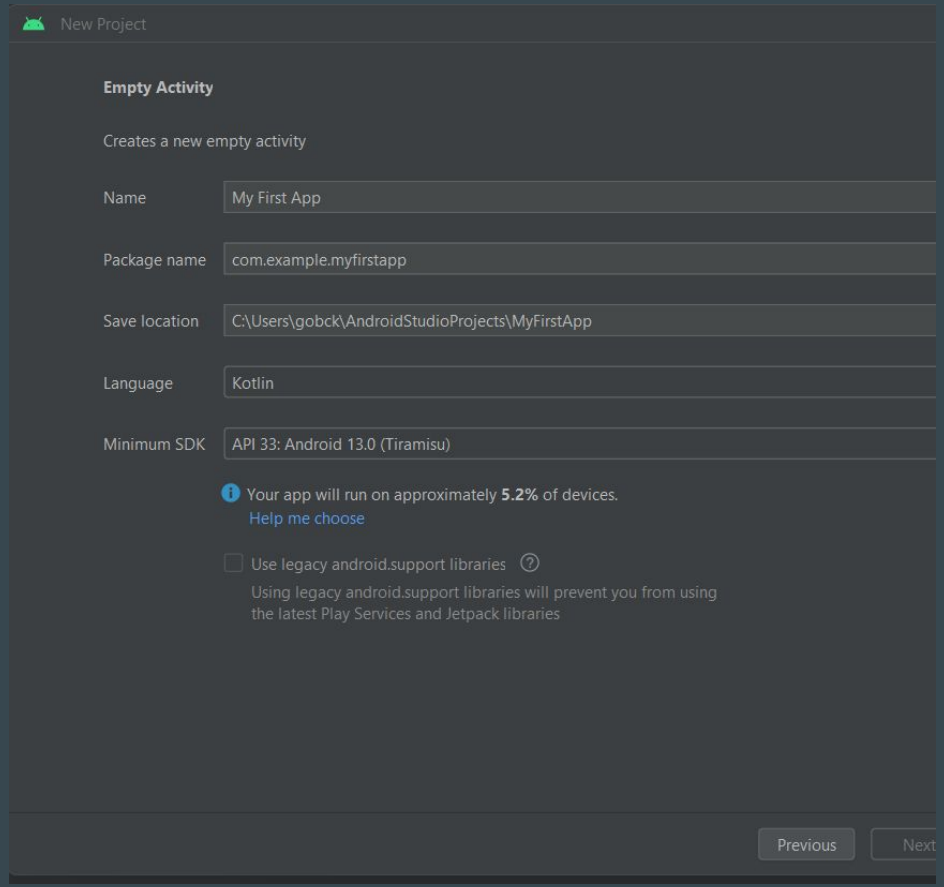
- Phone and Tablet
- Wear OS
- Android TV
- Automotive

Select the Empty Activity template



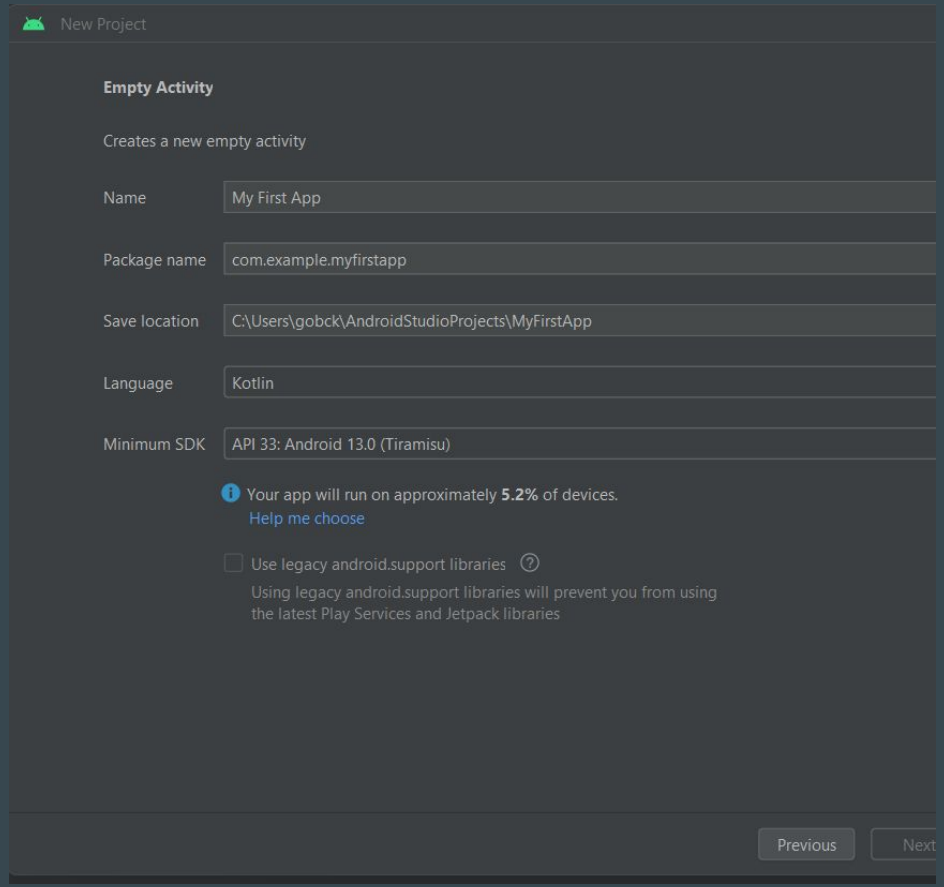
Project Details - Name

Enter a name for your application.



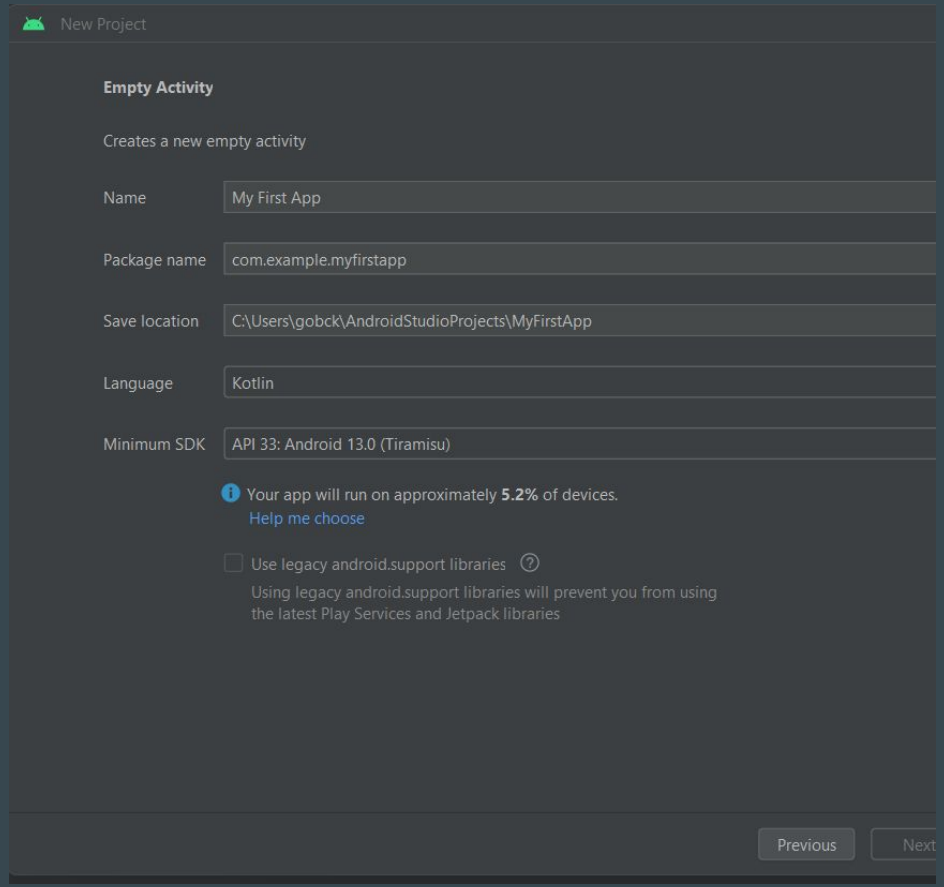
Project Details - Package Name

- A globally unique name that represents your app
- Similar to a web address
- We are not going to publish the first app, You can use the default



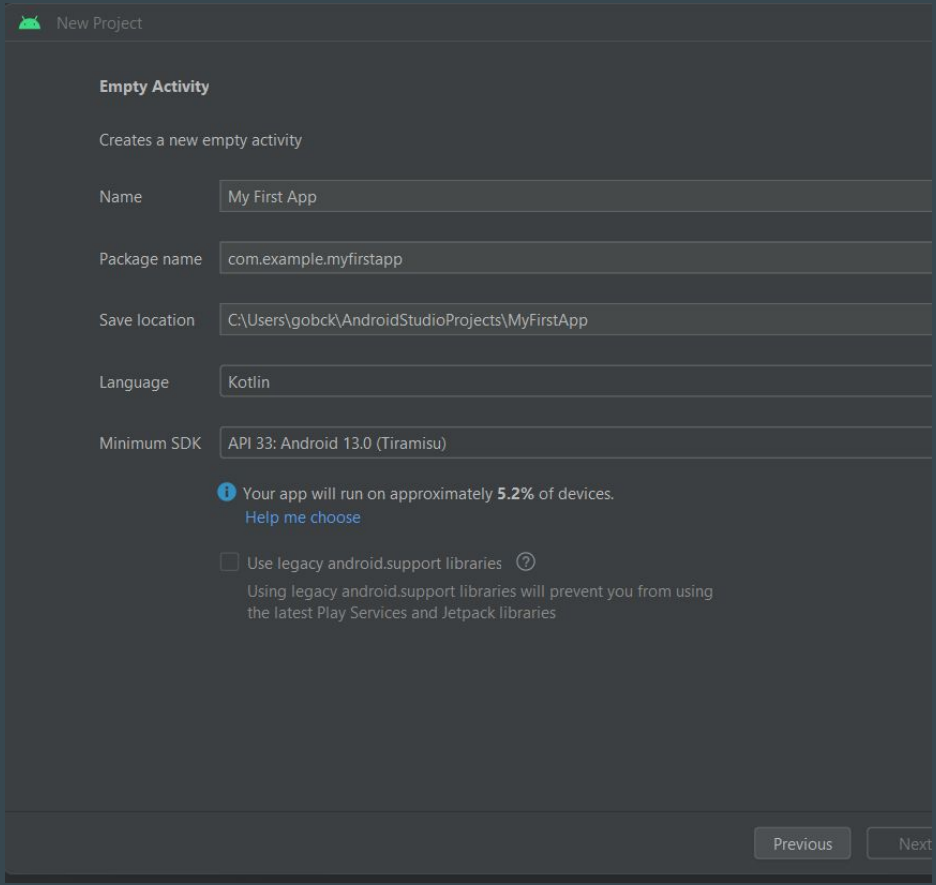
Project Details - Save Location

- where your app is stored on your computer



Project Details - Language

- Kotlin and the Java programming language are both supported.



Project Details - Minimum SDK

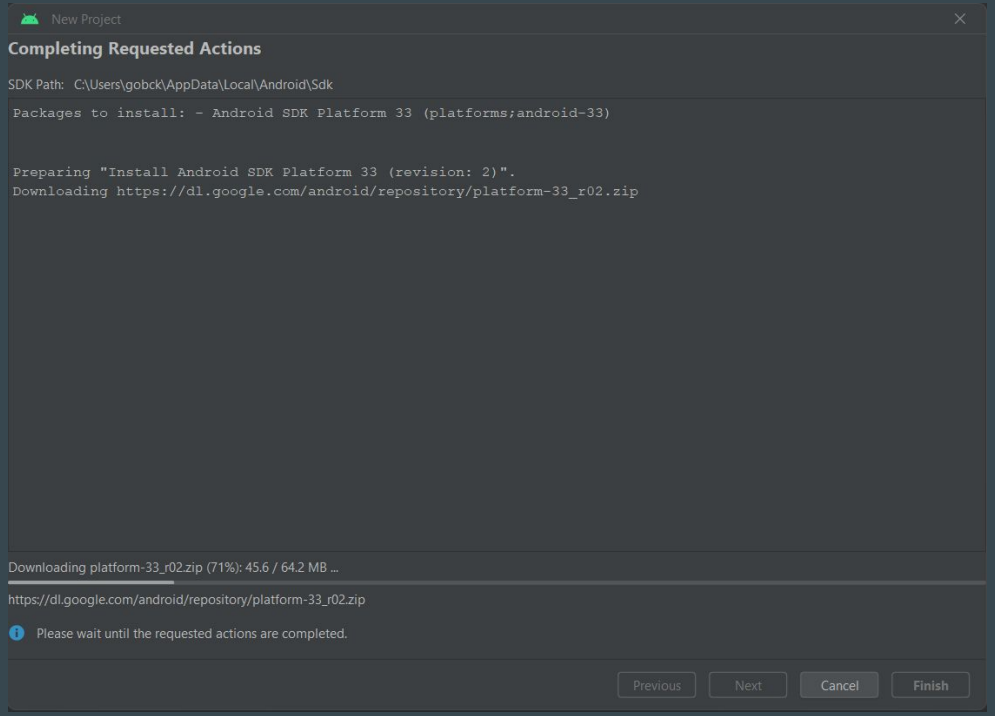
- Click Help me choose, to see information about API levels
- Android devices may be running different versions of Android. Each Android release has a
  - Platform Version
  - API level
  - Version Code (ex. Oreo, Q, R, S)

Android Platform/API Version Distribution

ANDROID PLATFORM VERSION		API LEVEL	CUMULATIVE DISTRIBUTION
4.4	KitKat	19	
5.0	Lollipop	21	99.3%
5.1	Lollipop	22	99.0%
6.0	Marshmallow	23	97.2%
7.0	Nougat	24	94.4%
7.1	Nougat	25	92.5%
8.0	Oreo	26	90.7%
8.1	Oreo	27	88.1%
9.0	Pie	28	81.2%
10.	Q	29	68.0%
11.	R	30	48.5%
12.	S	31	24.1%
13.	T	33	5.2%

Last updated: January 6th, 2023

Completing Requested Actions



Three API levels

Minimum SDK: Device needs at least this API level to install

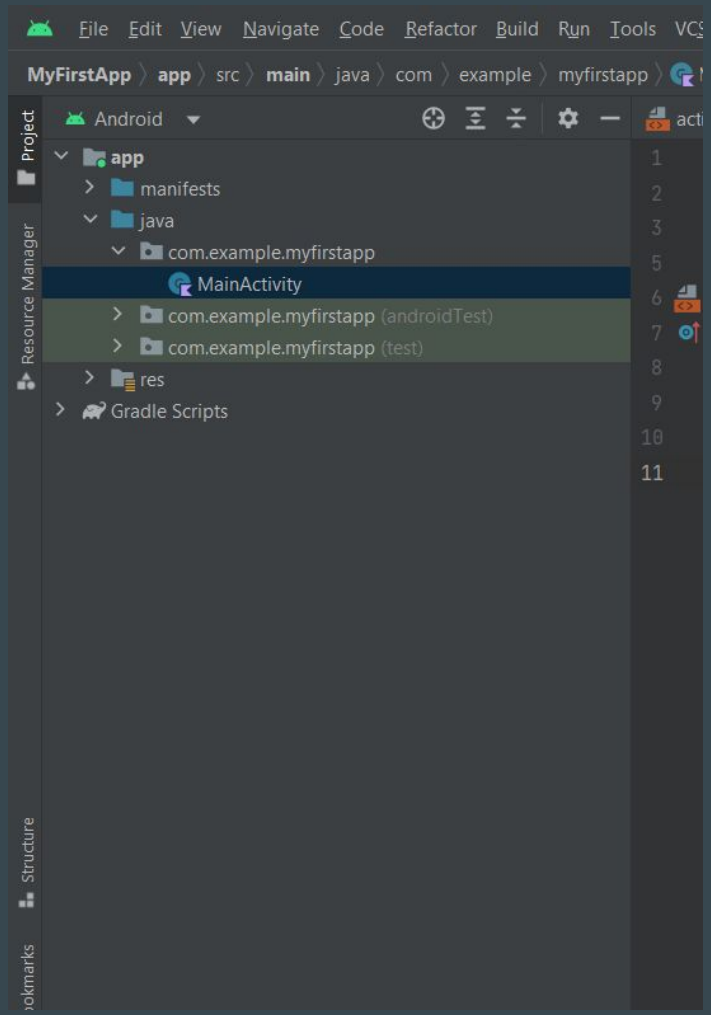
Target SDK: API version and highest Android version tested

Compile SDK: Android OS library version compiled with

minSdkVersion <= targetSdkVersion <= compileSdkVersion

Android Studio Tour - Project

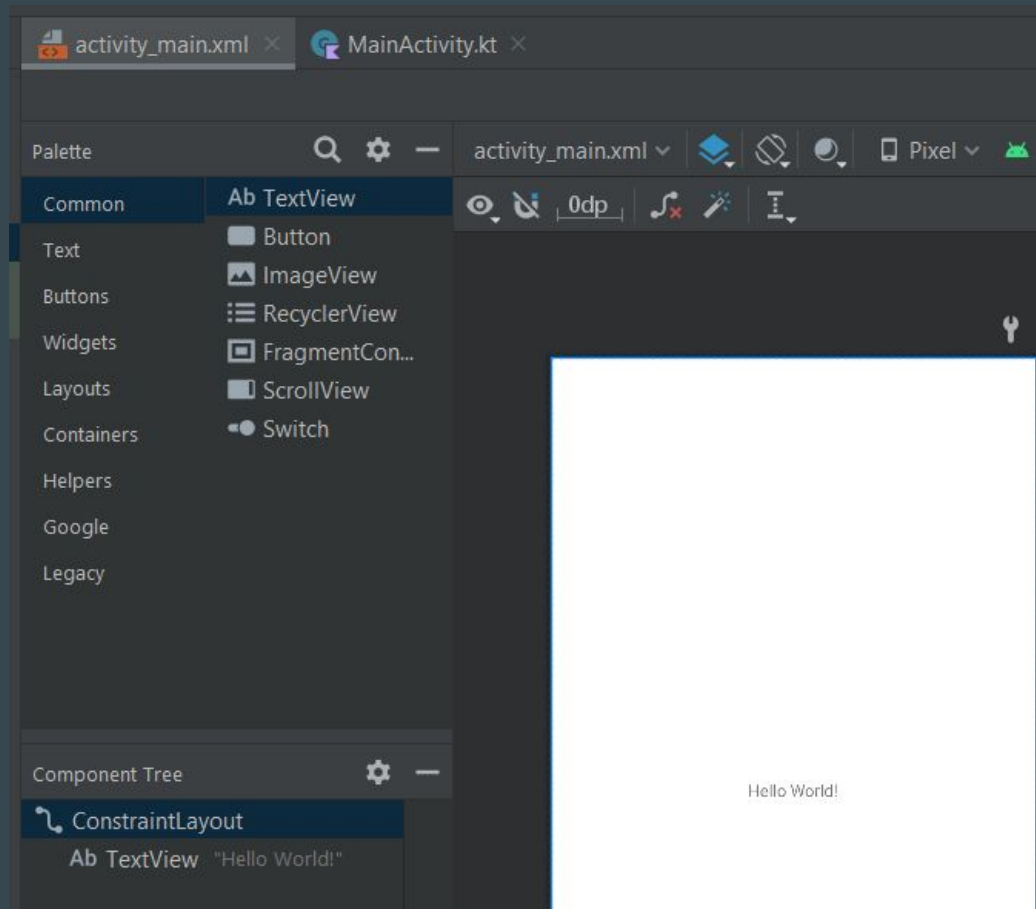
Project window shows the files and folders for your project.



Android Studio Tour - Palette

Palette shows the components and layouts

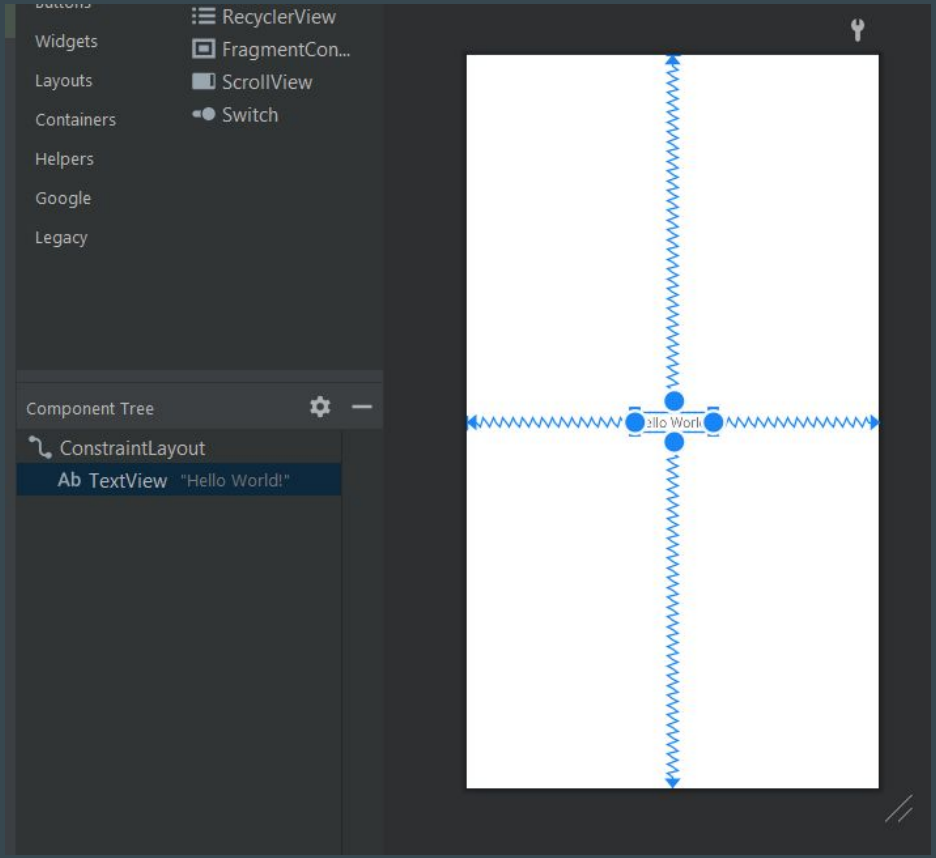
you can drag into your project, such as TextViews, ImageViews, and Buttons



Android Studio Tour - Component Tree

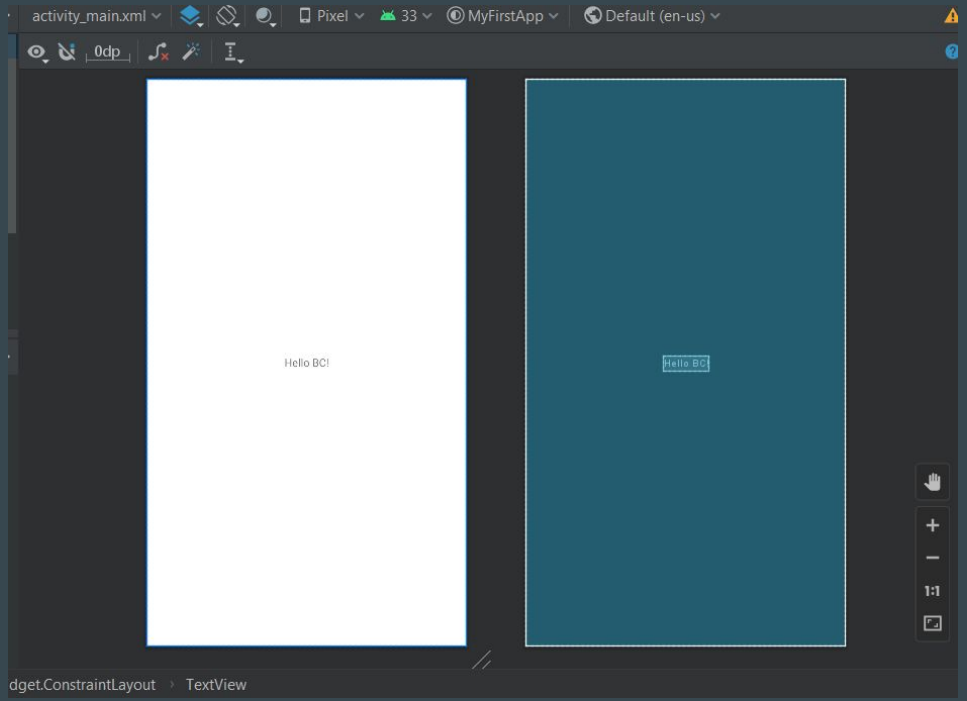
Component Tree shows the view hierarchy for your layout.

Click a component or layout to show it in the Design Editor.



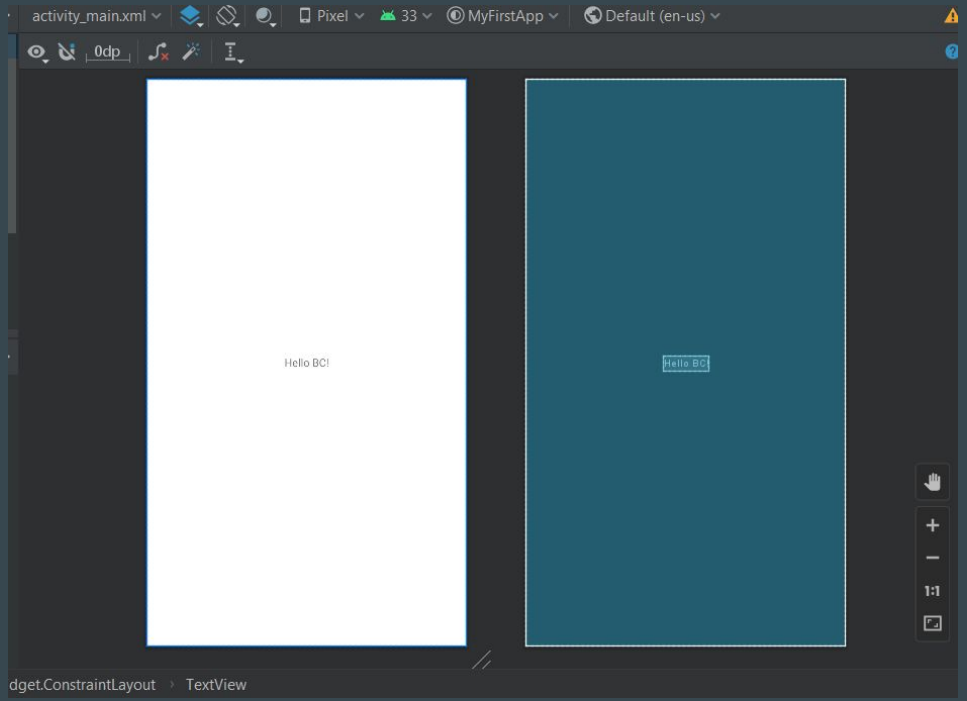
Android Studio Tour - Design Editor

Design Editor displays a Design view and a Blueprint view to give you a visual representation of your layout.



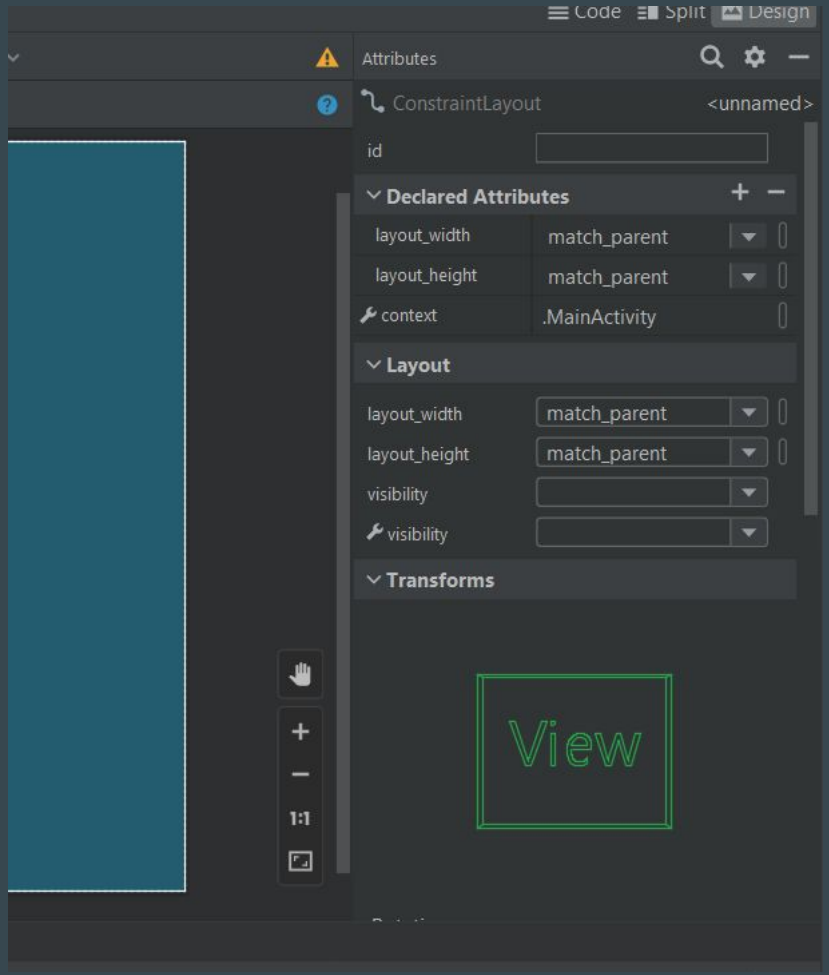
Android Studio Tour - Design Editor

Design Editor displays a Design view and a Blueprint view to give you a visual representation of your layout.



Android Studio Tour - Attributes

Attributes window contains a list of properties you can set for your component.

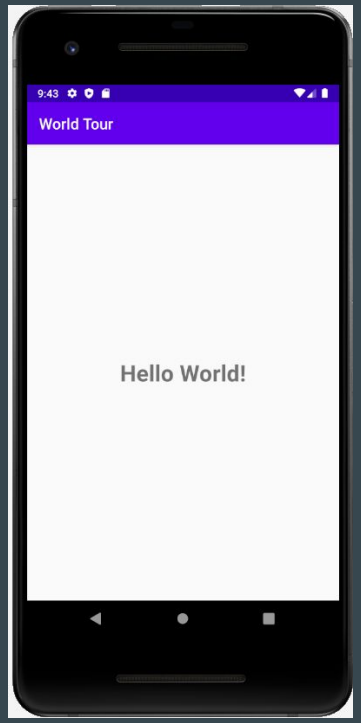


How to run your app

Two ways

Android device (phone, tablet)

Emulator on your computer

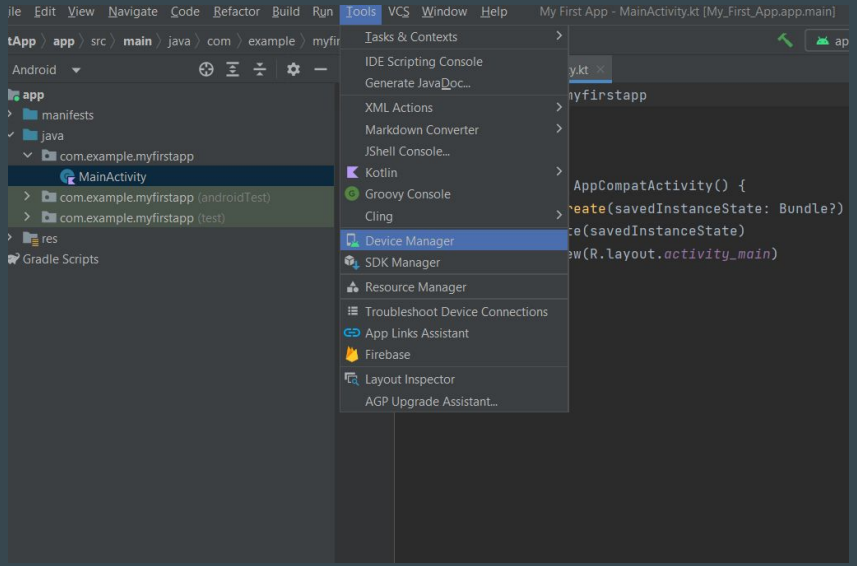


Android Virtual Device Manager

can use the emulator to emulate many different Android form factors

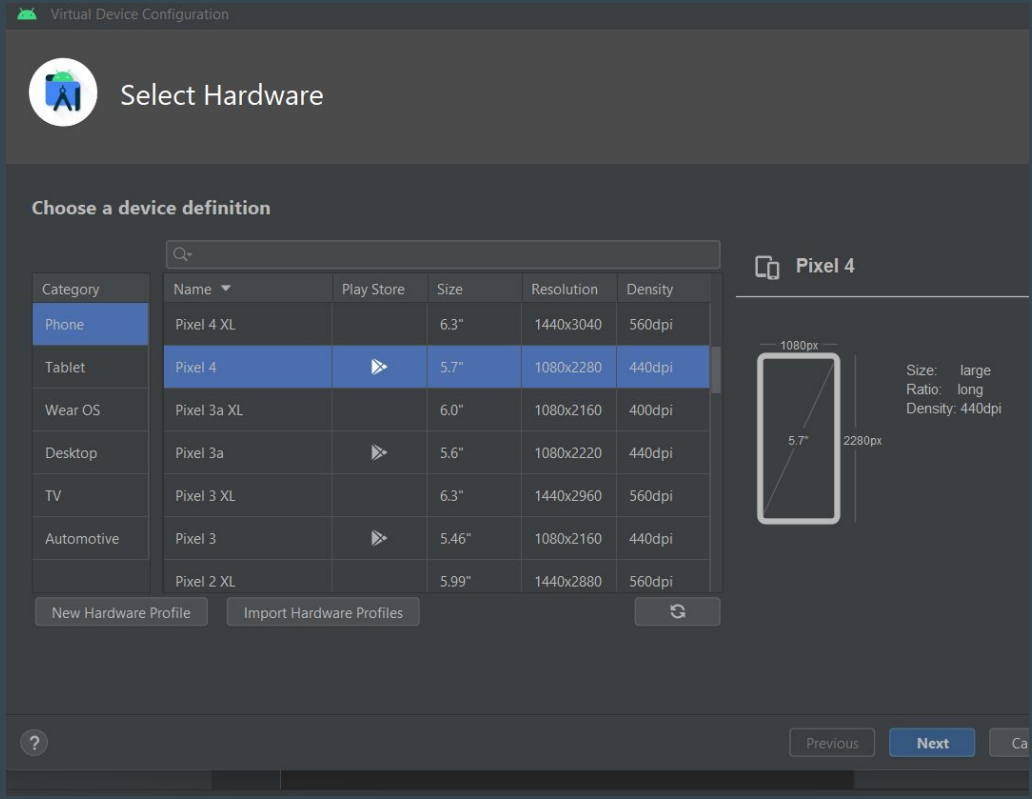
Tools -> Device Manager

Device Manager -> Create device



Select Hardware

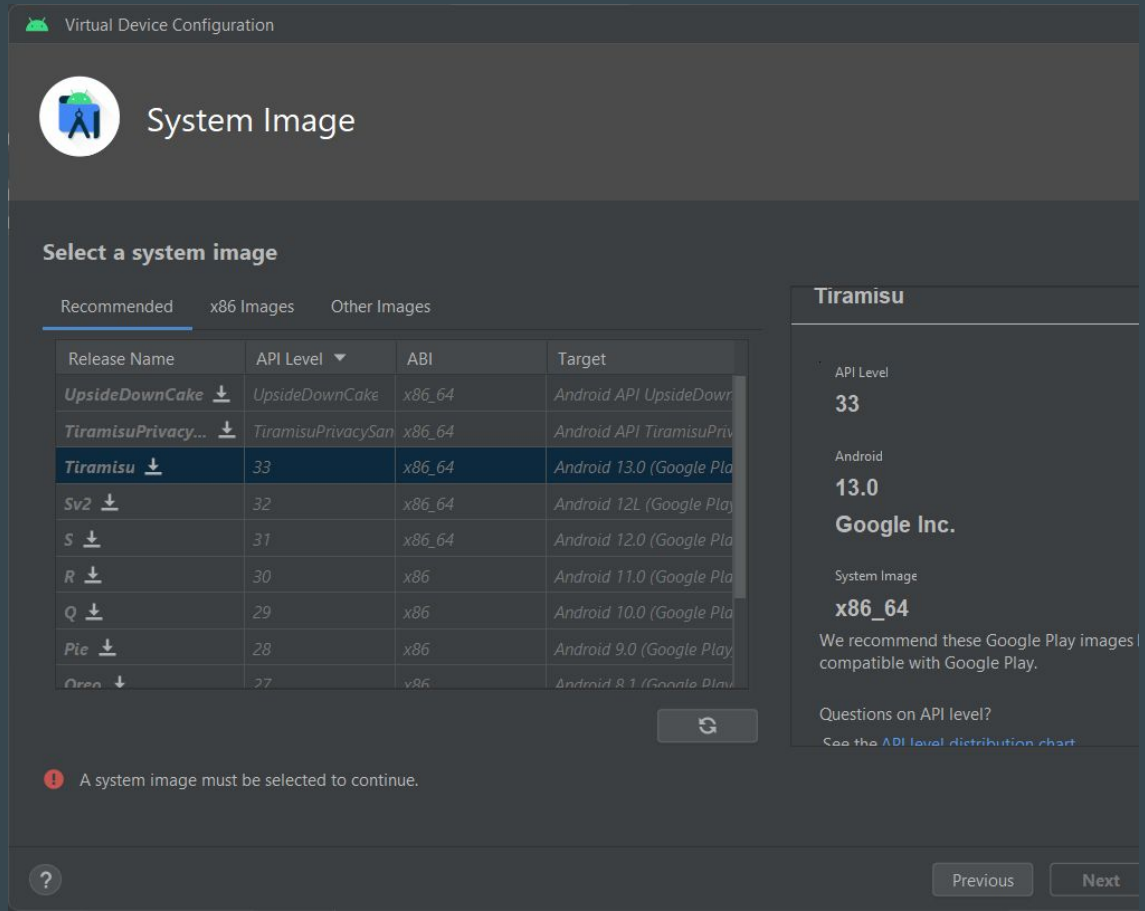
Pixel 4



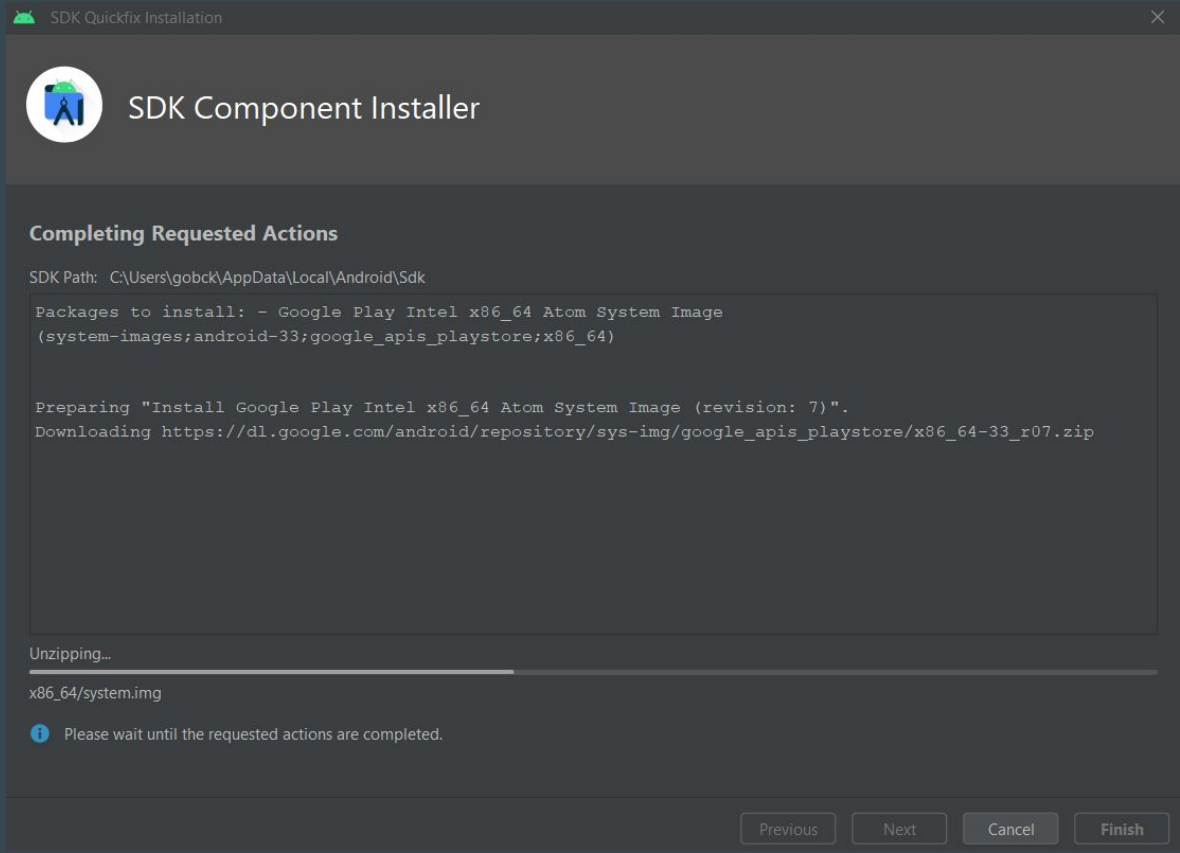
System Image

Release Name: Tiramisu

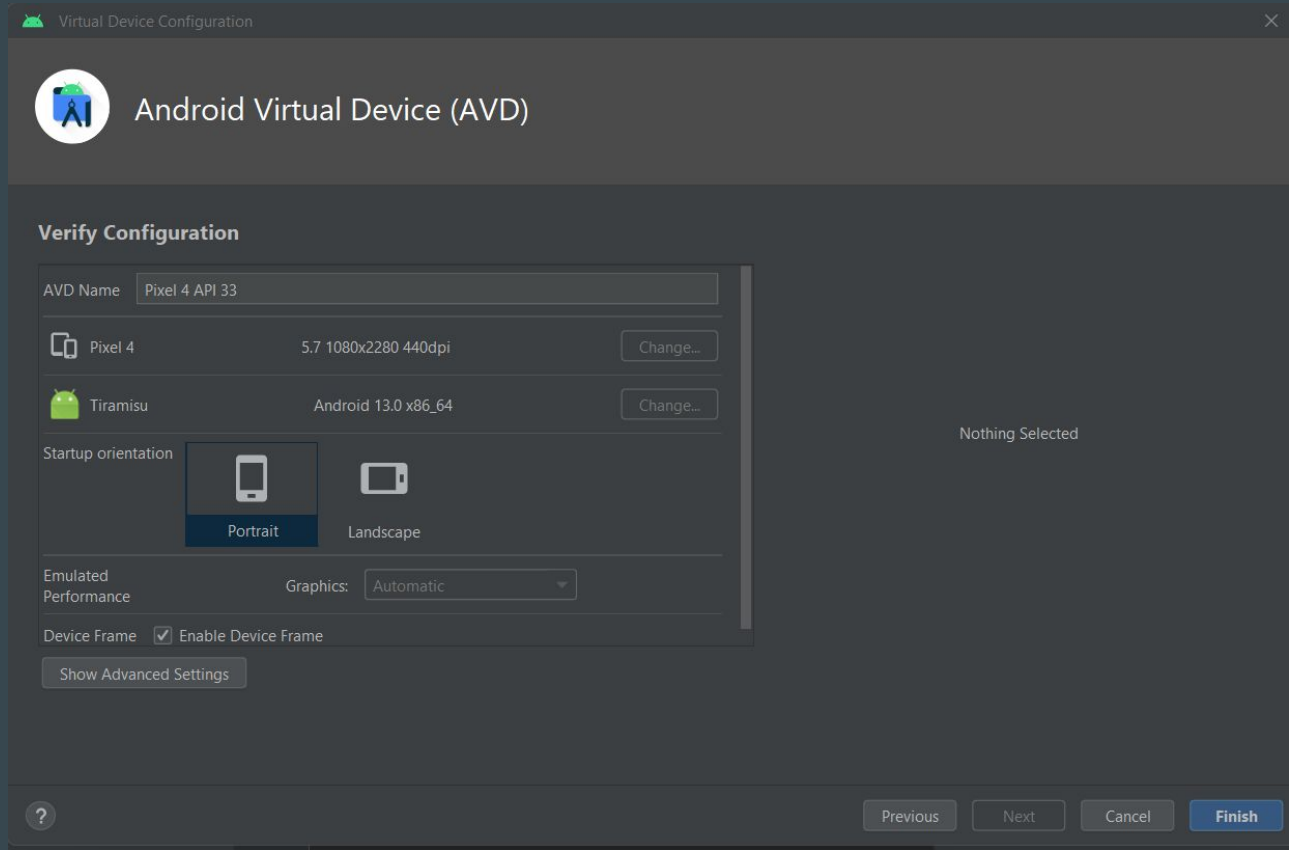
API Level: 33



SDK Component Installer



Android Virtual Device (AVD)

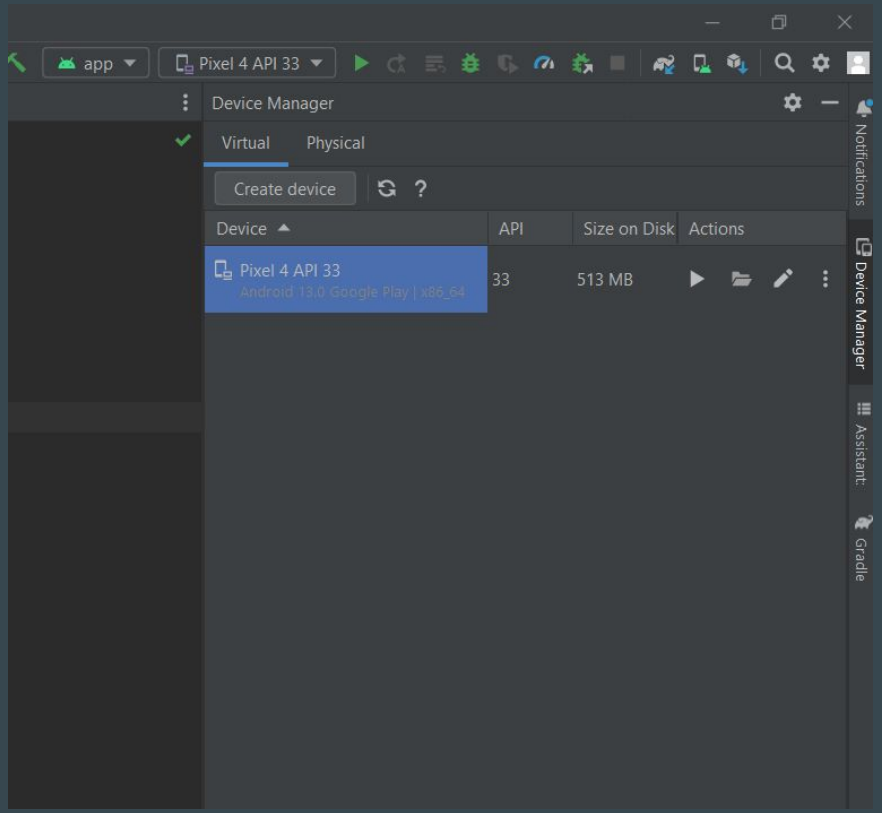


How to run

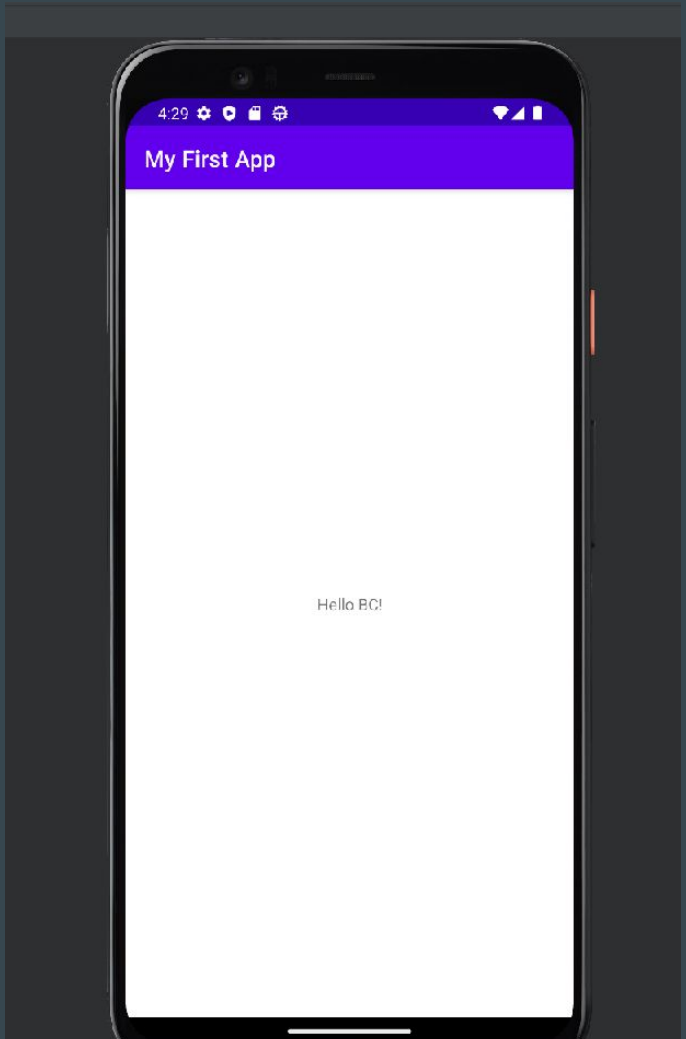
Check your virtual device is selected

- Ex. Pixel 4 API 33

Click Run App or Shift + F10



Virtual Device Result



Overview of basic app project

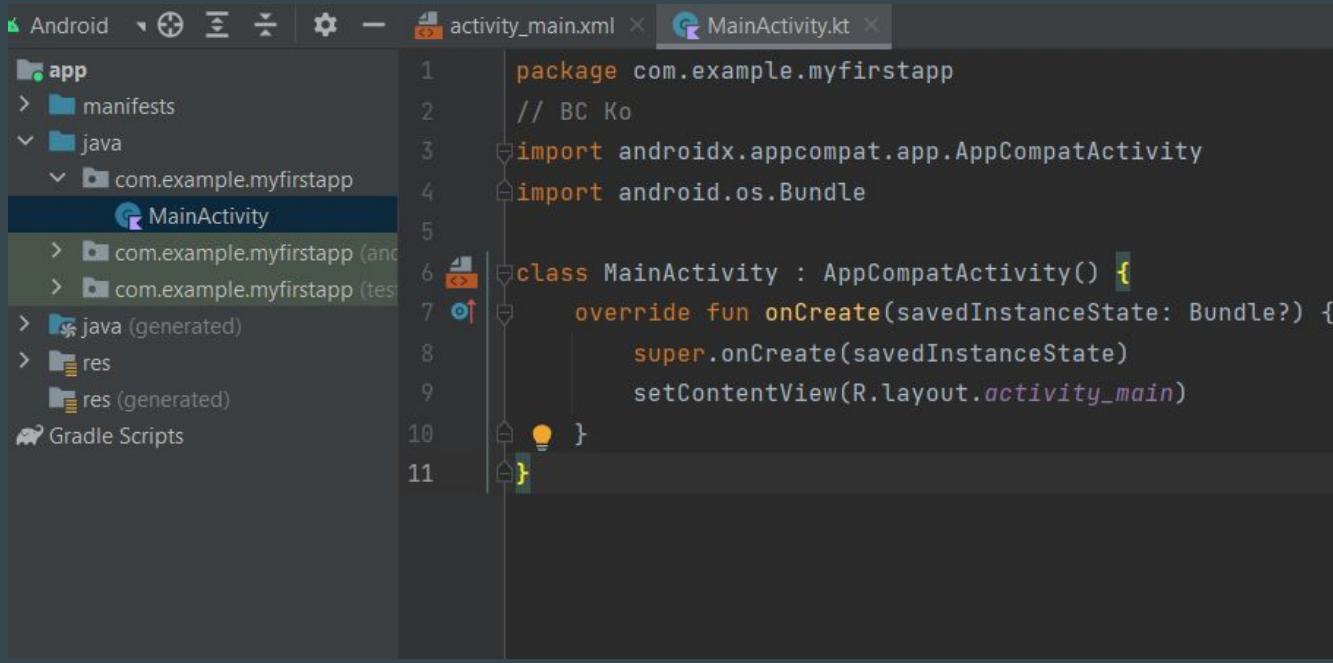
Activity

Resources

Gradle files

Overview of basic app project - Activity

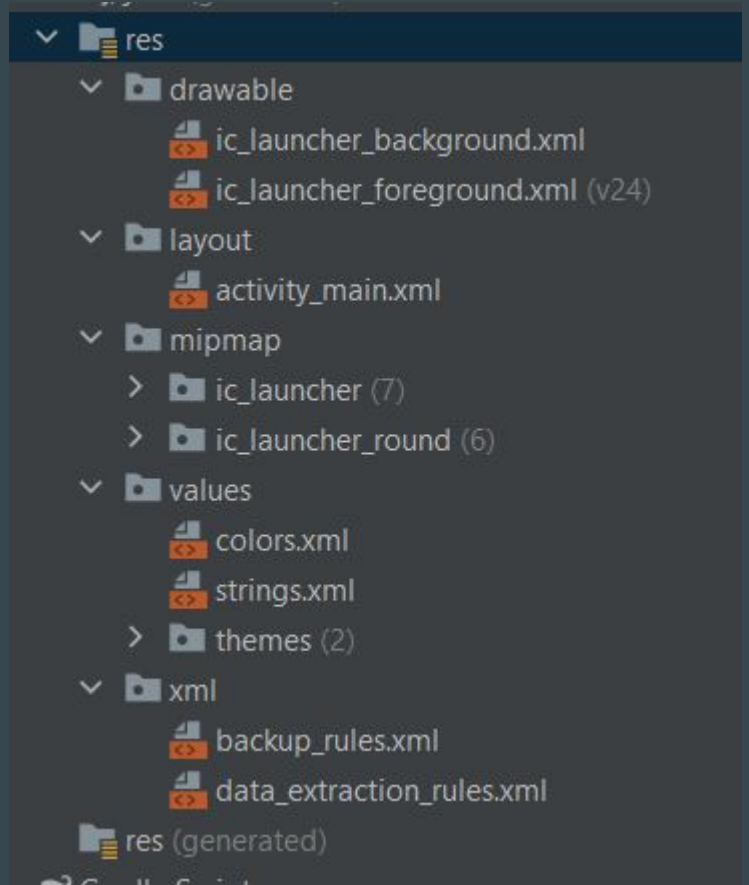
An Activity handles user input and creates a window on the screen to display your user interface.



Overview of basic app project - Resources

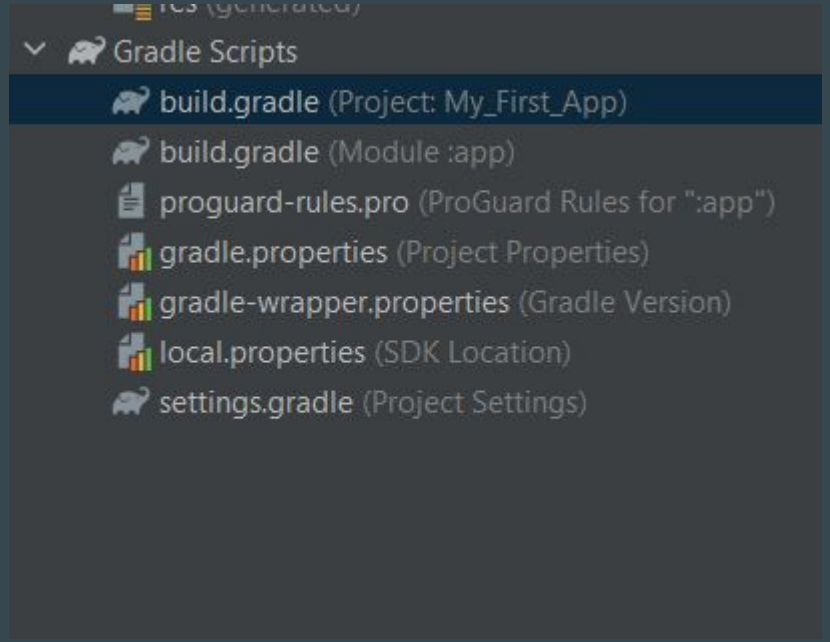
Resources are additional files that your code uses

- layout files
- Images
- audio files
- Themes
- Colors
- And more.



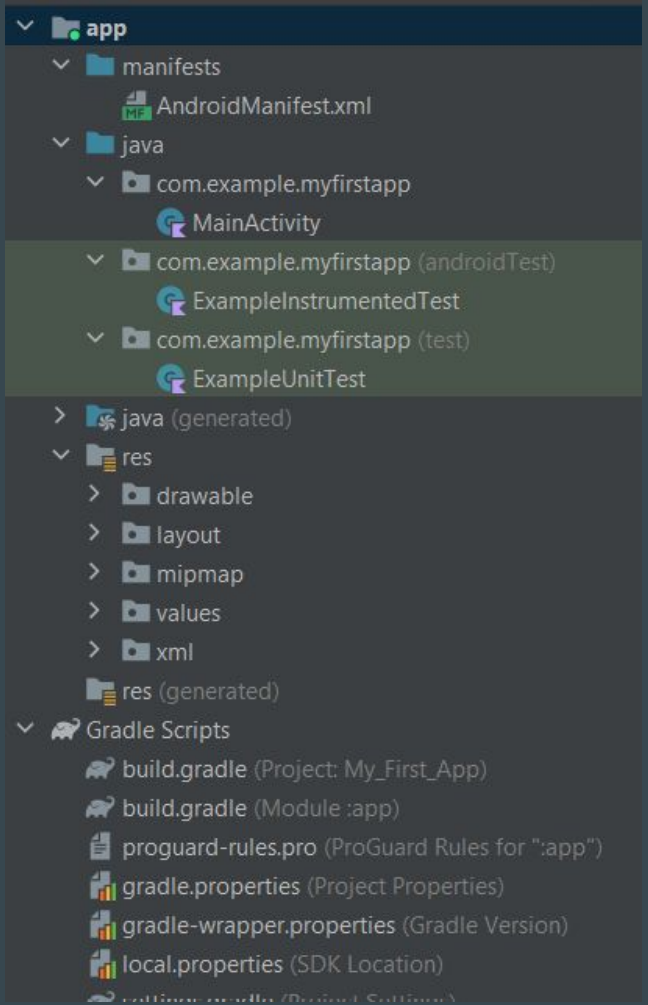
Overview of basic app project - Gradle files

- Gradle files are scripts that control how your app is built, so that it can be installed on a device.



App project structure

- **app** - stores source code, tests, and resources for your app
- **libs** - stores local libraries your app depends on
- **androidTest** - test code that's specific to Android
- **main** - Java and Kotlin app files
- **test** - local unit tests that will execute on your computer
- **AndroidManifest.xml** - declares essential information for your app
- **build.gradle** - controls how your application builds, tests, and deploys itself



Layouts and resources

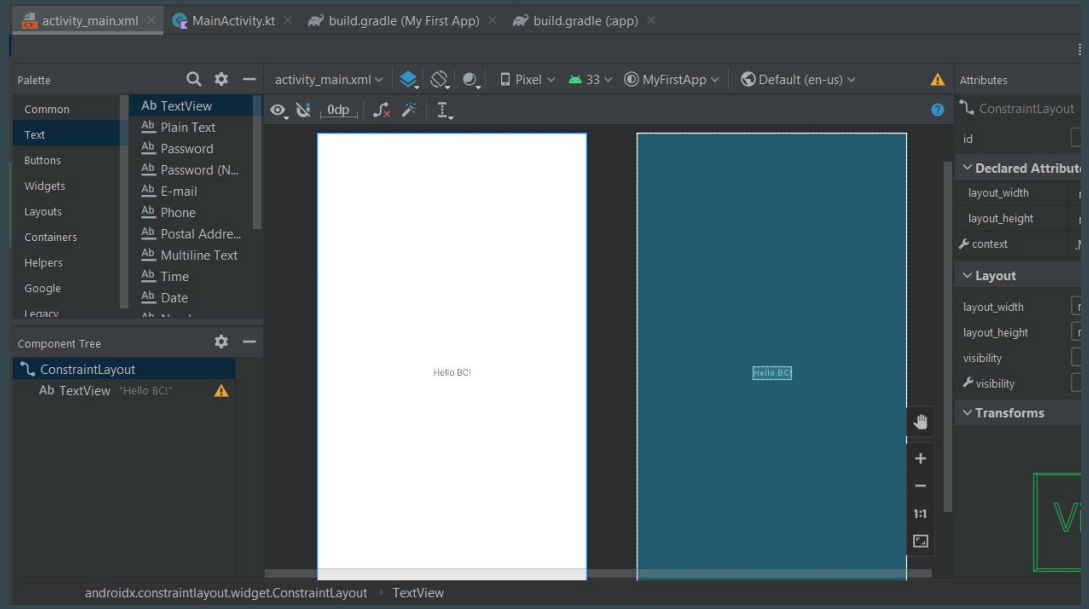
Views

- Views are the user interface building blocks in Android
  - Bounded by a rectangular area on the screen
  - Responsible for drawing and event handling
  - Examples: TextView, ImageView, Button
  - Each View has different attributes
- Can be grouped to form more complex user interfaces

Layout Editor

You can build your layout with the layout editor

- Drag and drop components from the Palette into the Design view
- See a preview of your layout in the Design view
- Modify the attributes of the views on the right hand side in the Attributes window .



XML Layouts

You can also edit your layout in XML.

- Android uses XML to specify the layout of user interfaces (including View attributes)
- Each View in XML corresponds to a class in Kotlin that controls how that View functions

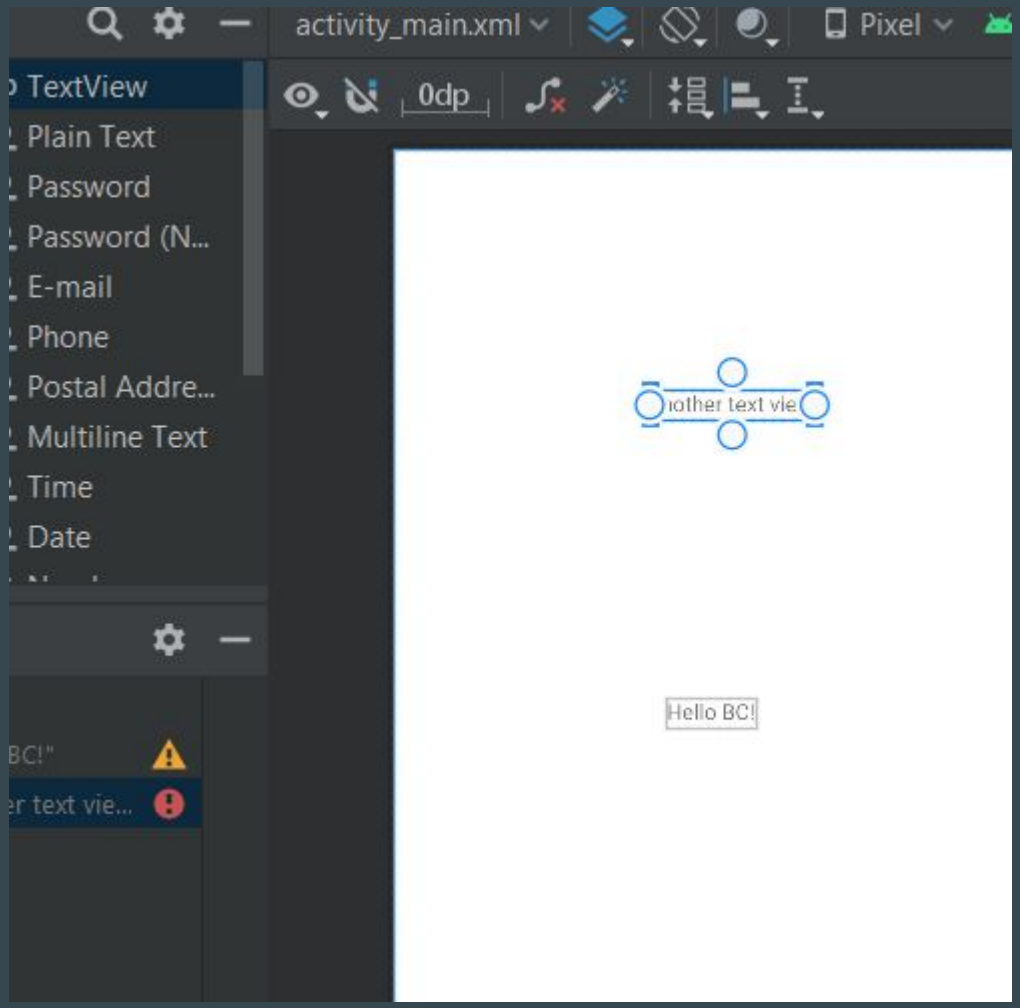
activity_main.xml | MainActivity.kt | build.gradle (My First App) | build.gradle

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:andr
    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 BC!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

androidx.constraintlayout.widget.ConstraintLayout › TextView

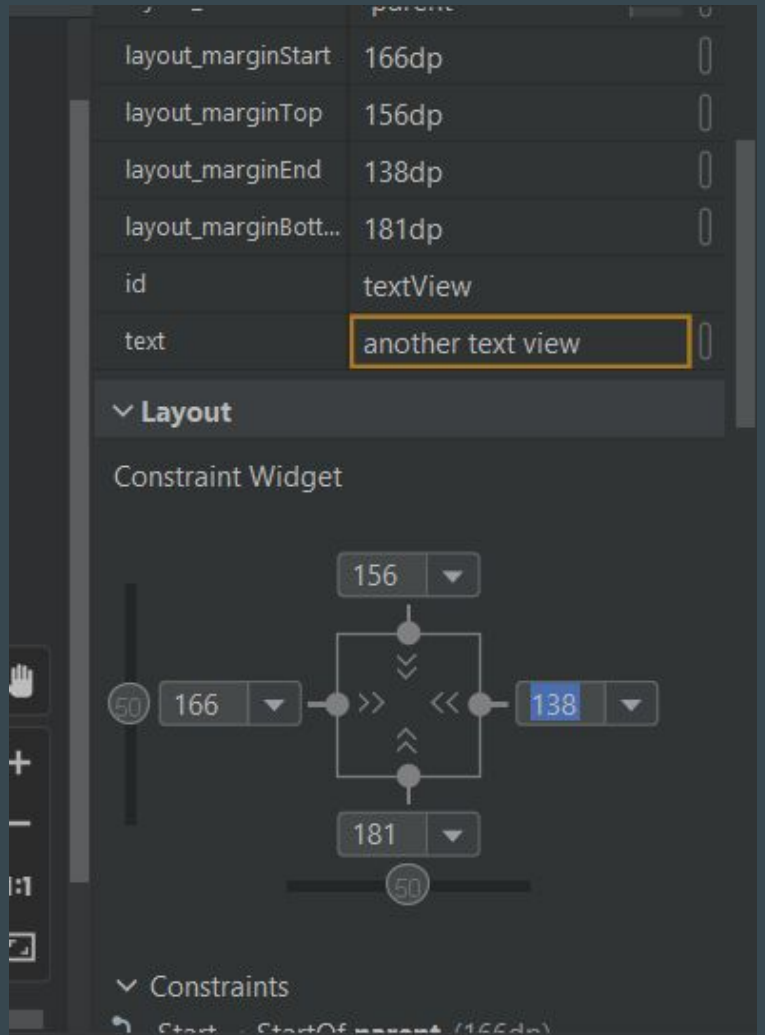
TextView - Layout Editor

Drag and drop TextView components from the Palette into the Design view



TextView - Layout Editor

Modify the attributes of the views on the right hand side in the Attributes window .



TextView - XML

Here's the XML added using the Layout editor.

You can manually add in the XML editor as well

We see attributes on the TextView

- Width
- Height
- Text
- Margin
- id

```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="166dp"
    android:layout_marginTop="156dp"
    android:layout_marginEnd="138dp"
    android:layout_marginBottom="181dp"
    android:text="another text view"
    app:layout_constraintBottom_toTopOf="@+id/textView2"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

View Size Attributes

wrap_content

- use only as much space as needed to display the content within the View.
- android:layout_width="wrap_content"
- Example: If you want the View to be as wide as the text within the TextView, use wrap_content.

match_parent

- use the dimension of the parent
- android:layout_width="match_parent"
- Example: if you want the ImageView to take up the full size of the parent, set match_parent for its width and height.

Fixed value (use dp units)

- set a specific dp (density-independent pixels) value
- android:layout_width="48dp"

ViewGroups

To show more than one view on screen

a container for views, and controls how views are organized and laid out on screen

Three different ViewGroups

- FrameLayout
- LinearLayout
- ConstraintLayout

ViewGroups - FrameLayout

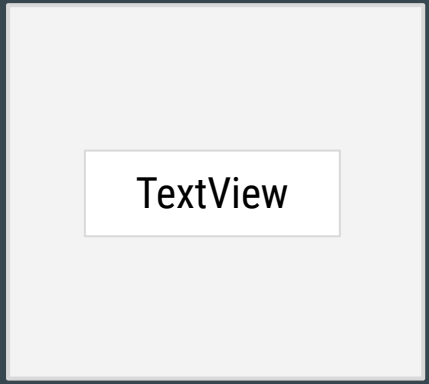
holds a single TextView

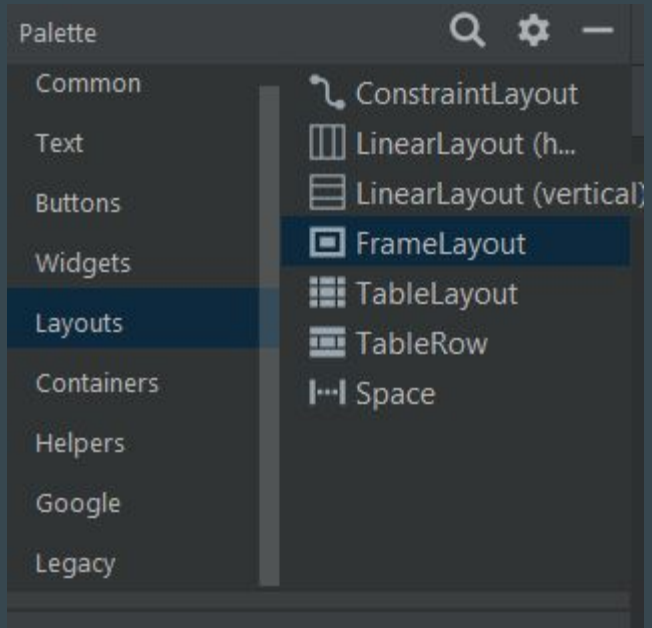
think of FrameLayout as a picture frame

Meant to show one thing

What if you add more than one child?

- The views would overlap



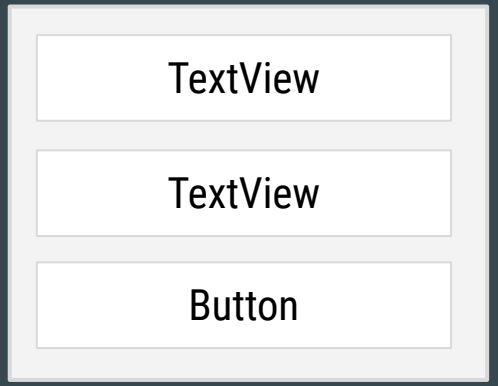


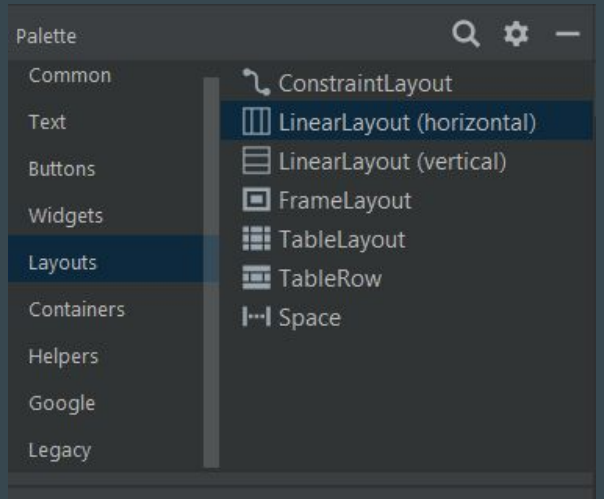
ViewGroups - LinearLayout

lays out child views in a row or column

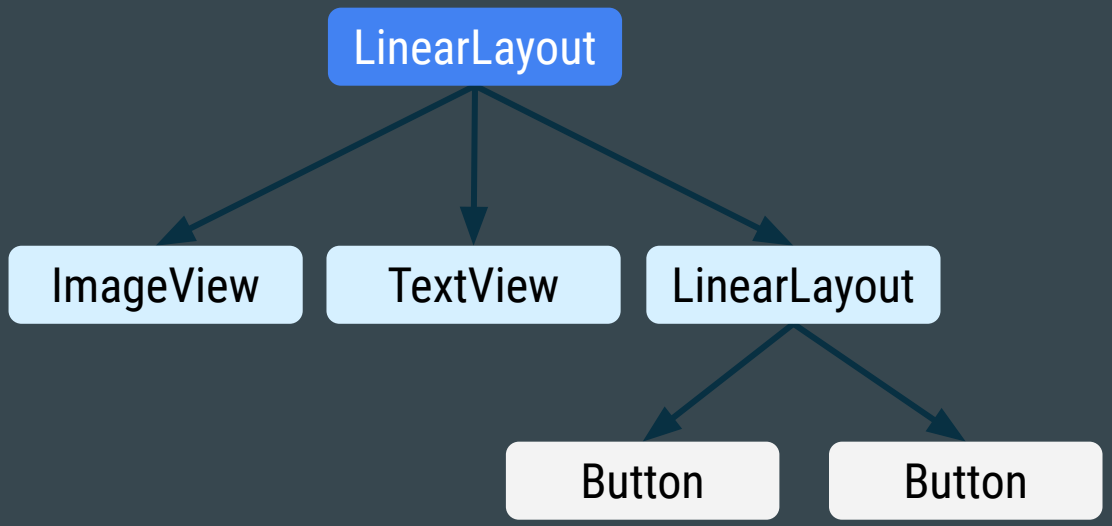
layout direction is determined by the orientation attribute on the LinearLayout

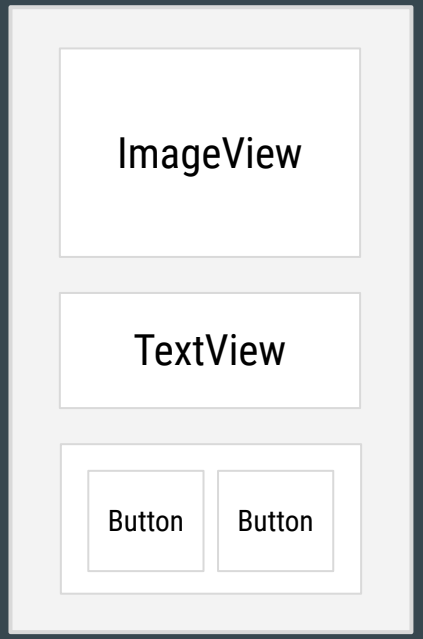
Vertical





View hierarchy





App resources

Resource Directories

drawable stores all files related to drawing images and related assets.

layout contains all the layout XML files for your application. There may be more than one if your app needs to handle different orientations or densities.

mipmap contains files for your app icon (known as your launcher icon) at different screen densities.

values contains files related to simple collections of strings, colors, integers, and styles.

- Example: localizing your app, more than one values directory

```
main
├── java
└── res
    ├── drawable
    ├── layout
    ├── mipmap
    └── values
```

Resource IDs

Each resource has a resource ID to access it.

When naming resources, the convention is to use all lowercase with underscores (for example, activity_main.xml).

Android autogenerates a class file named R.java with references to all resources in the app.

Individual items are referenced with: R.<resource_type>.<resource_name>

```
R.drawable.ic_launcher (res/drawable/ic_launcher.xml)
R.layout.activity_main (res/layout/activity_main.xml)
```

Resource IDs for views

Individual views can also have resource IDs.

You can access them using R.id.<resource_name>.

Be sure to use unique resource names so it is clear which element you want to refer to

Example

- R.id.textView2

```
<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello BC!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Activity

What is an Activity?

An Activity is a means for the user to accomplish one main goal.

An Android app is composed of one or more activities.

Examples of activities:

- Displaying a list of emails
- Displaying details of one specific item
- Taking a photo using the camera

MainActivity.kt

automatically generated because we selected the Empty Activity

extends from the AppCompatActivity class

- inherit behavior from the Android framework about how an Activity works
- ensures that newer features are available to legacy versions of Android

one function that is overriding the onCreate function that was defined in the superclass

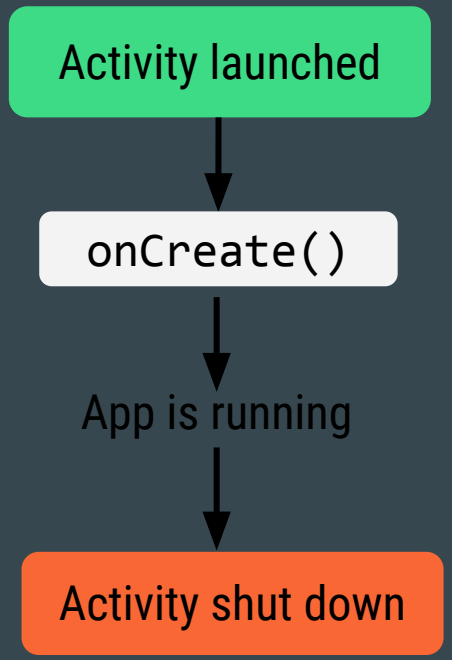
```
package com.example.myfirstapp
// BC Ko
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

How an Activity runs

starts your app through an Activity instance

Flow

- user taps your app icon on the device
- opens your app by launching the main Activity of your app
- onCreate() method is called when your Activity is created
    - one of the Activity callback methods that is invoked from the system at certain stages of the Activity lifecycle (will discuss more)
- Activity continues to run until the user or system takes action to shut it down



onCreate() callback

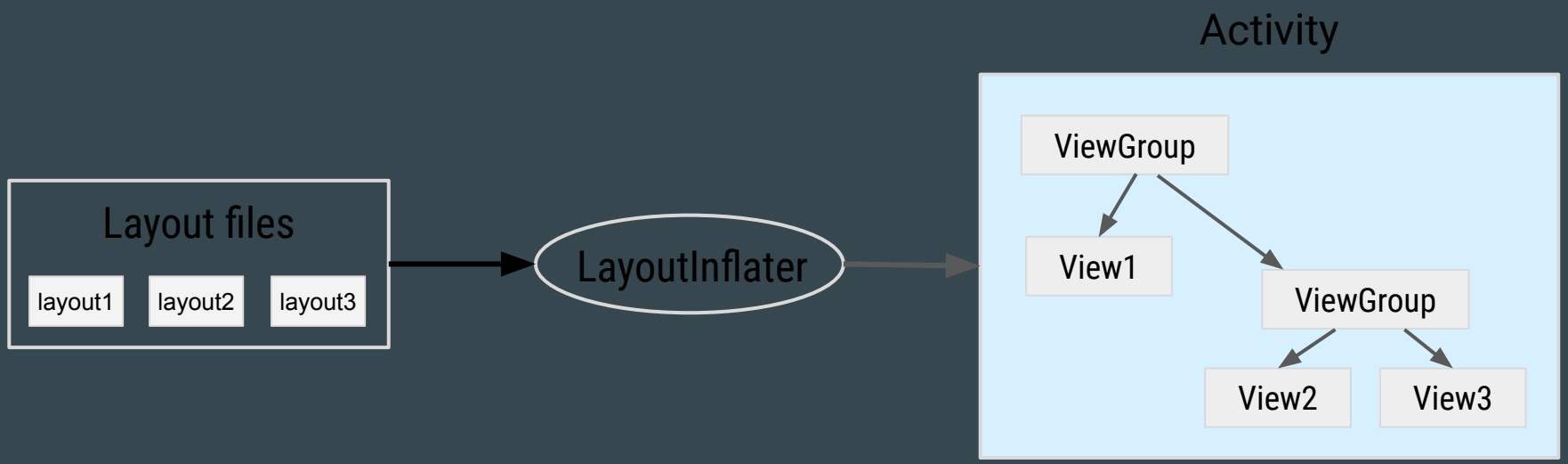
Called when the system creates your Activity

superclass onCreate() method to set up the Activity

call setContentView() so that the layout defined in the activity_main.xml file is displayed

```
package com.example.myfirstapp
// BC Ko
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

Layout inflation



- XML file parsed by LayoutInflator
- Hierarchy of View objects is created to be displayed in the Activity