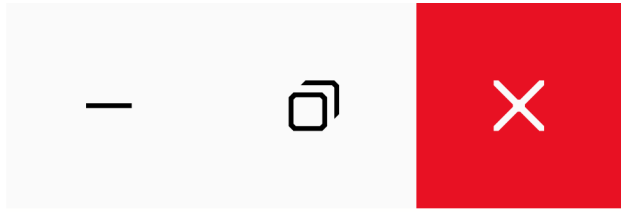
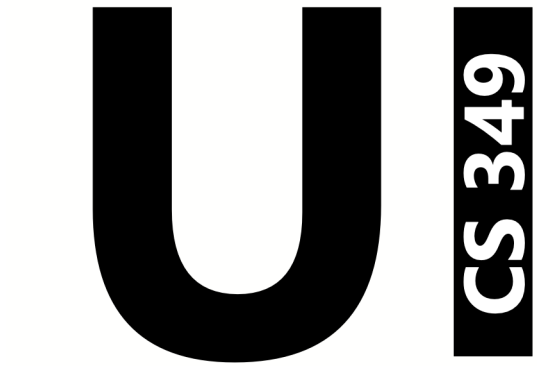


Android

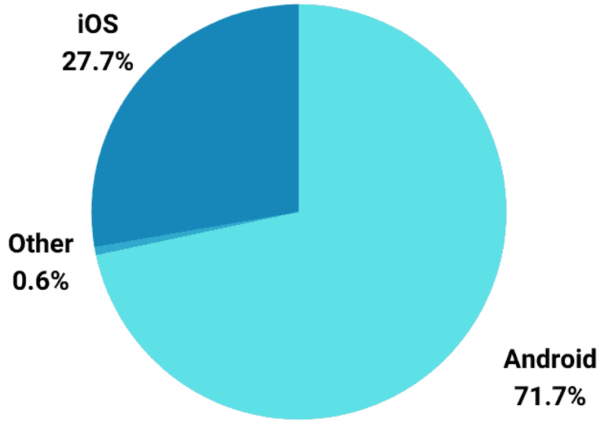
Development Fundamentals
Layouts
Widgets



July 10

Development Fundamentals

Why Android?



June 2023 Global Market Share

Pervasive Mobile Platform

- World's most popular/installed OS
- Runs hundreds of millions of devices
- Mobile-first

Developer Friendly

- State-of the art (Kotlin) and legacy (Java) languages
- Multi-platform development support (win, mac, linux)
- Open Source (by minimum-definition)

Exposure to different UI development paradigms

- **Android Views**: *imperative* using code to build interface, or *declarative* using XML description of interface
- **Jetpack Compose**: *declarative* using composable functions
- **MVVM** architecture (variation of MVC)

Android Architecture

Android is based on Linux.

Every app has distinct user profile

- App permissions restrict access (resources, data)

Every app runs in its own process

- Apps must request access to shared resources (e.g., file system, camera)
- Strict separation of data/memory/resources

System manages application “lifecycle”

- the OS launches and manages apps
- It can also terminate apps that have not been used in a while

Development Process

Apps written in Kotlin (or Java or C++) using an Android SDK

- Applications include source code + resources (media files, layouts)
- SDK libraries extend device capabilities to your source code.
- XML *manifest* file describes the application and contains information that the OS needs to install and run your application.

Compiler generates an Android Package file (APK)

- Includes manifest, code, resources, etc.

APK can be installed on a physical Android phone (with developer mode enabled), or through an online App Store.

APK can also run on an Android Virtual Device (AVD) for testing/debugging.

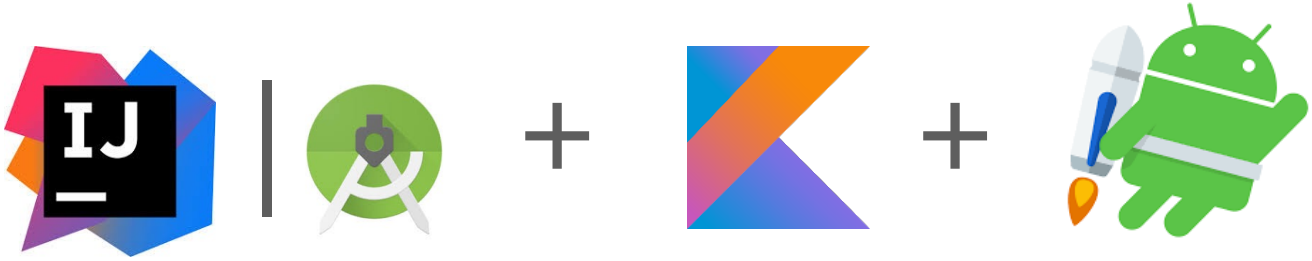
Development Environment

Toolchain

- IntelliJ (latest version), Kotlin
- Java JDK (11.0.17) and Gradle (7.5, or whatever is auto-installed)
- Android SDK: Android Tiramisu – API Level 33

Android Device Emulator (AVD)

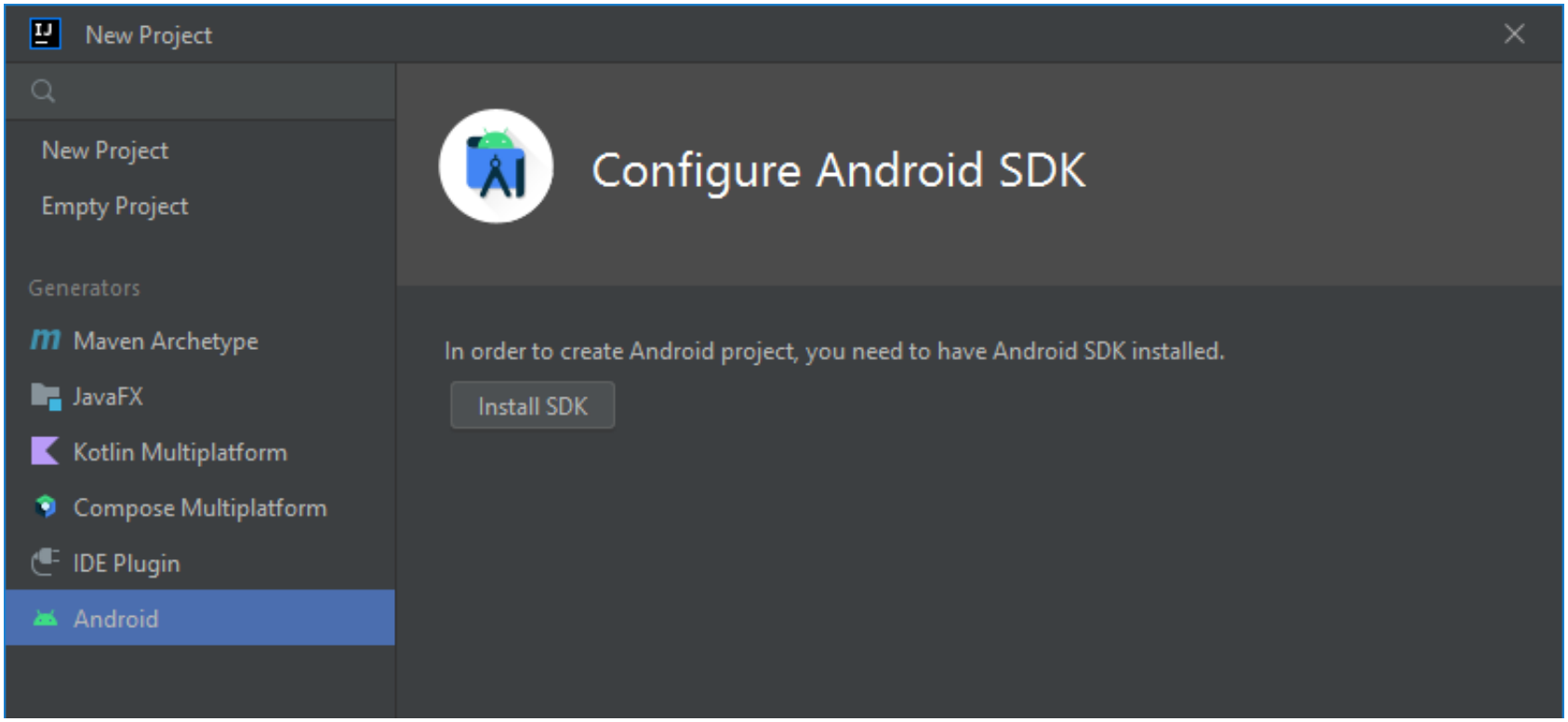
- Pixel 4a API 33



Environment Setup

IntelliJ already lists “Android” as new project type

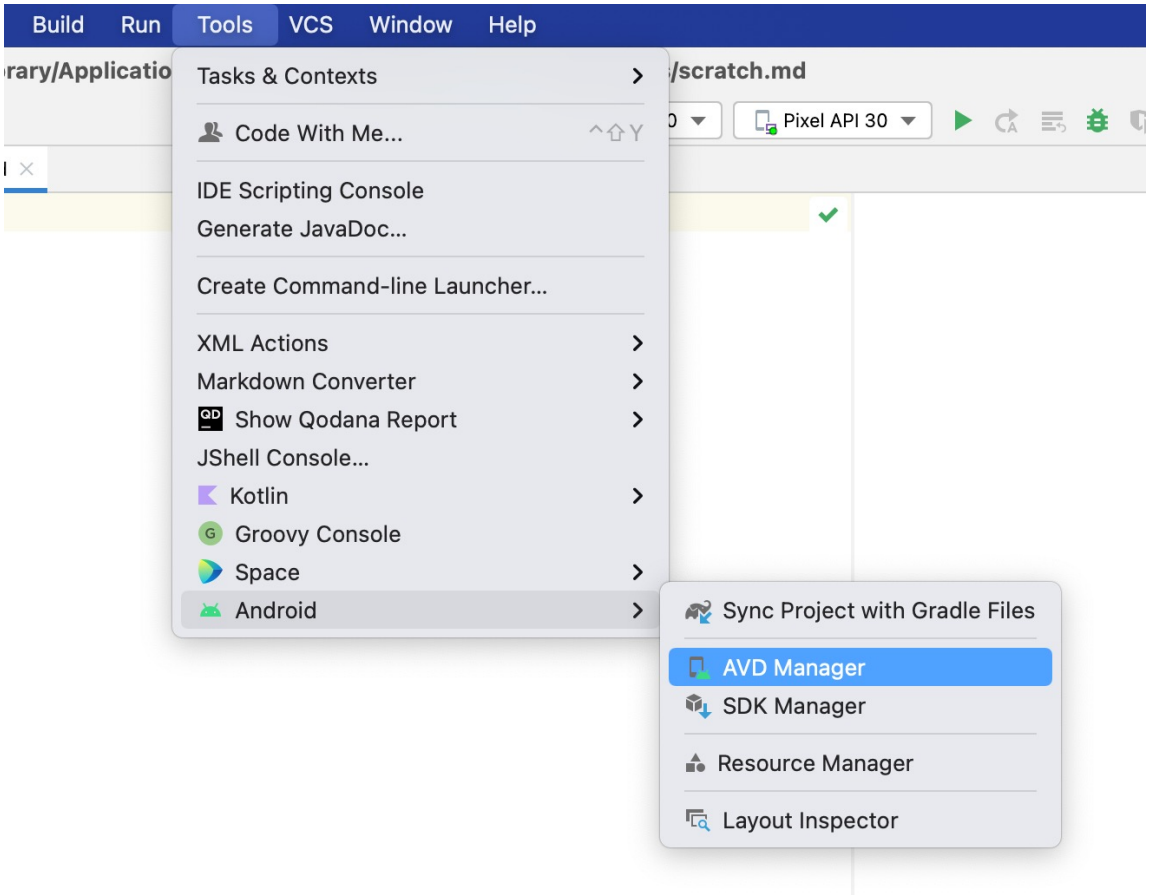
First time an Android project is created, it walks through installing the Android SDK and an Android Device Emulator (AVD). Follow these steps using SDK and AVD versions on previous slide: https://www.jetbrains.com/help/idea/create-your-first-android-application.html



AVD and SDK Manager

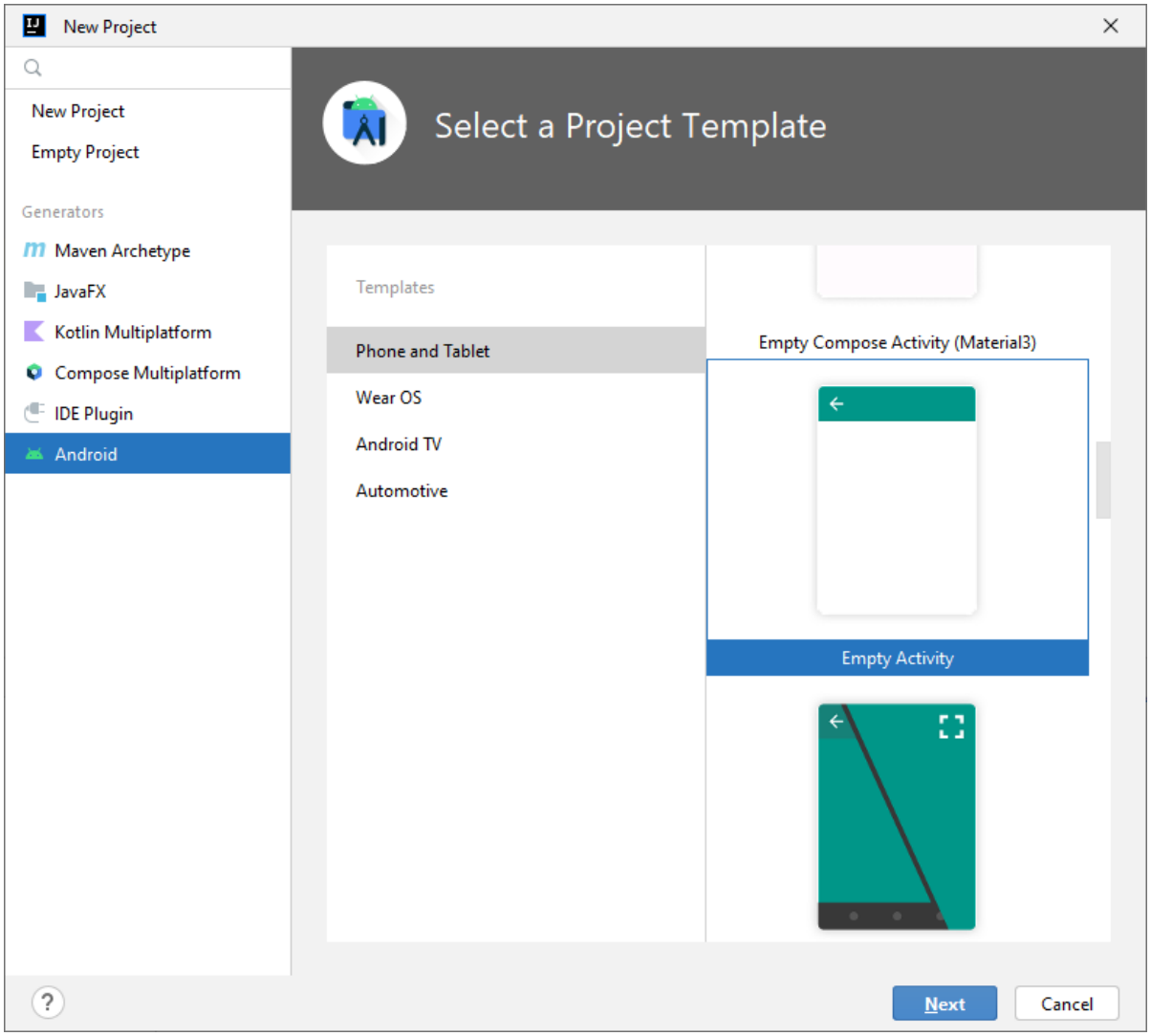
In IntelliJ “Tools/Android” menu:

- AVD Manager: add and manage Android Virtual Devices (AVDs)
- SDK Manager: add and maintain Android SDK and tools



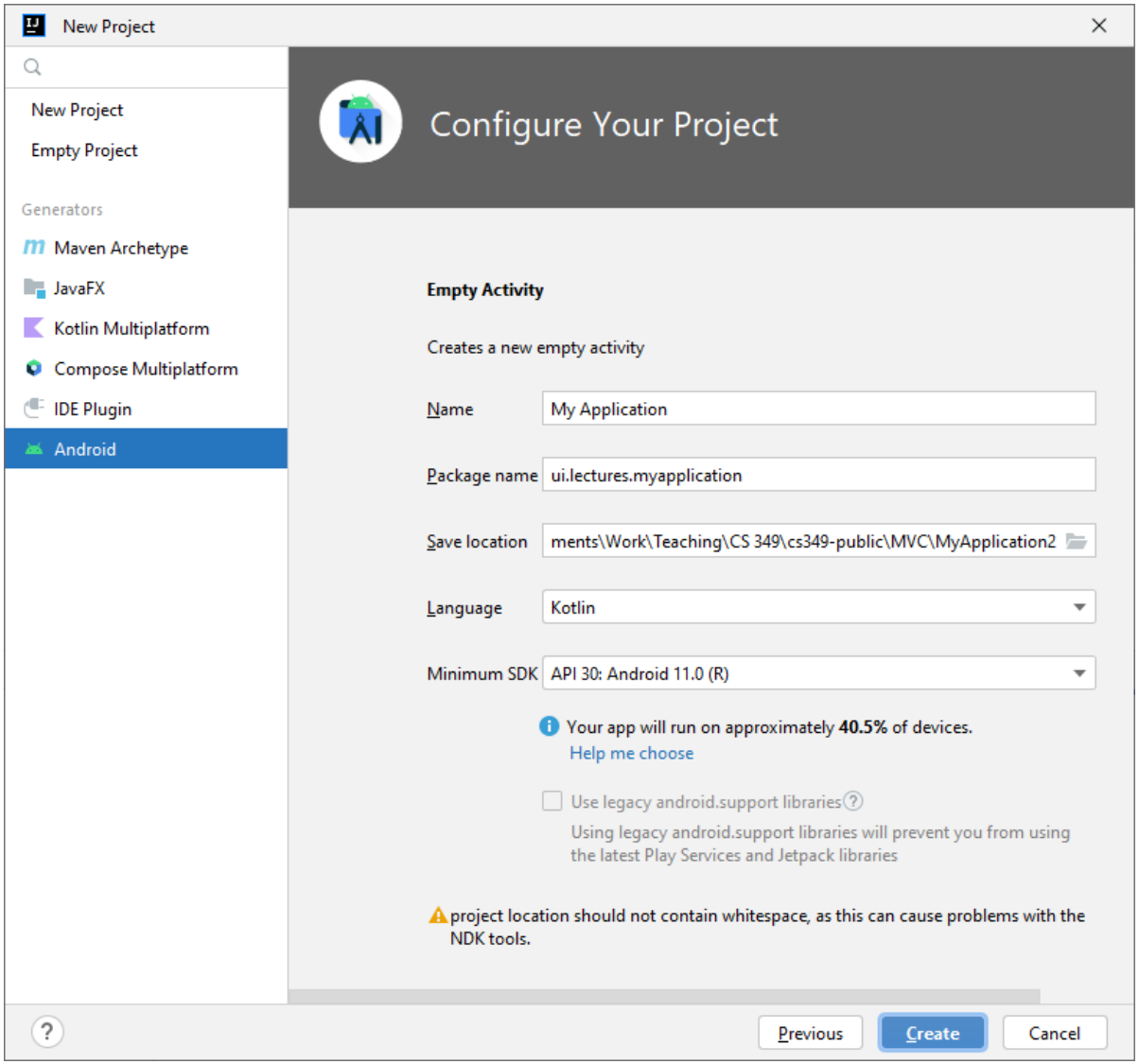
Basic Project Walkthrough: New Android Project

Pick “Empty Activity” project template



Basic Project Walkthrough: Configure

Use Kotlin and Minimum SDK API 30: Android 11.0 (R)



Run The App

Verify the right Android SDK and AVD Version is selected.

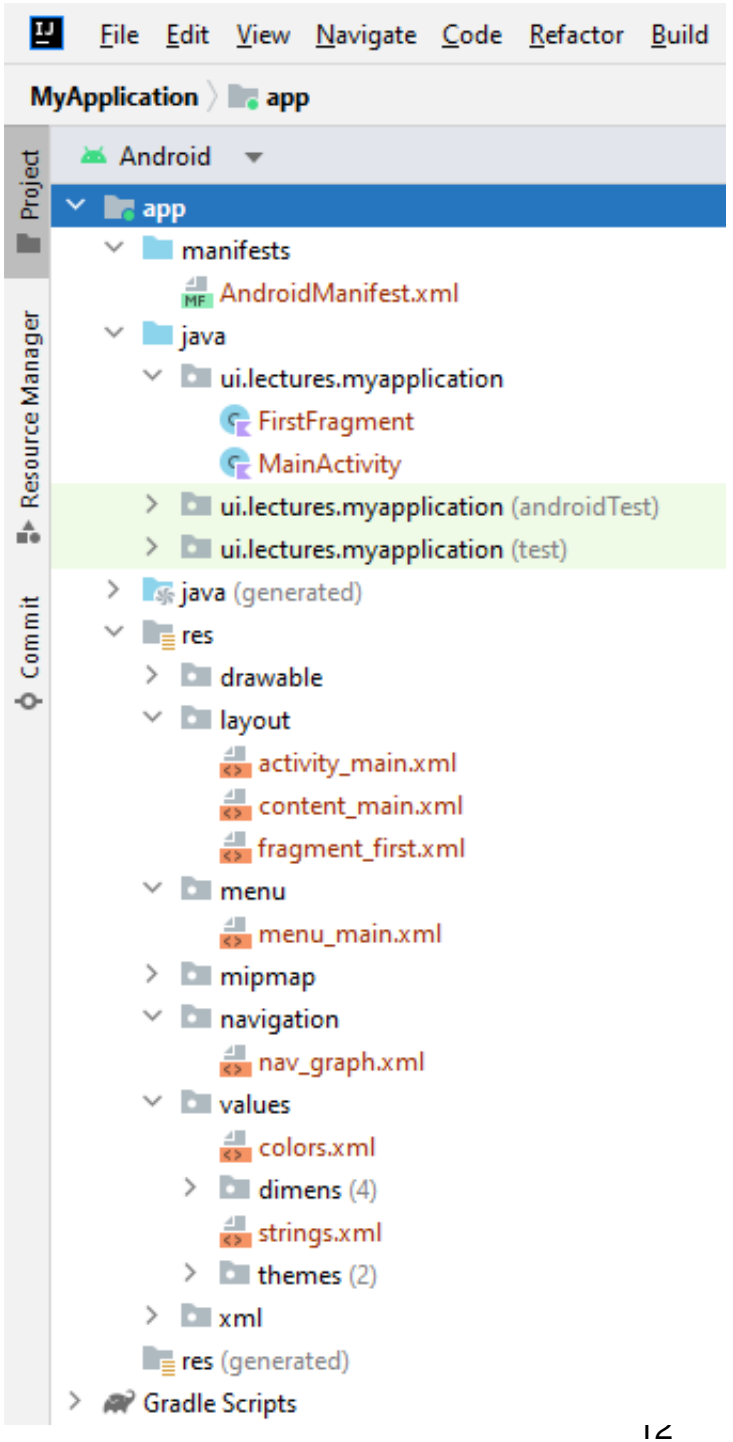
To test it, compile and run the default app:

After compiling, the AVD will boot and the app will load:

Android Project Structure

Select “Android” in Project Tab dropdown to get more optimized project view

- **Manifest** (app/manifests/): application settings (.xml)
- **Source Code** (app/java/): app behaviour (.kt)
- **Resources** (app/res/):
  - layout/: UI layout and View definitions (.xml)
  - navigation/: Navigation between activities and fragments (.xml)
  - values/: constants, e.g., strings, colours, colour schemes, … (.xml)
  - mipmap/: raster images (.webp, .jpg, etc.)
  - drawable/: vector images (.xml, .svg, etc.)



Manifest (AndroidManifest.xml)

Metadata about the application:

- Settings, such as, icon, name, and design theme
- Application components or “activities”
- “intent” filters: e.g., which activity to launch as main activity

```
<?xml version="1.0" encoding="utf-8"?>
<manifest …>
    <application
            android:icon="@mipmap/ic_launcher"
            android:label="@string/app_name"
            android:theme="@style/Theme.MyApplication"
            tools:targetApi="33">
        <activity
                android:name=".MainActivity"
                android:label="@string/app_name"
                android:theme="@style/Theme.MyApplication.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Application Components

Applications can have multiple “application components”

Each component is an entry point for a user (or the system) into some part of your application

Components	Description
Activity	Entry point for interacting with the user; a single screen with a user interface.
Service	General-purpose entry point for keeping an app running in the background
Content provider	Manages a shared set of app data that you can store in the file system
Broadcast receiver	Component that enables the system to deliver events to the app

Activity

An **Activity** is a crucial component of an Android app and a fundamental part of the Android application model.

The Activity class creates a window to host a user interface. This window is almost always full-screen.

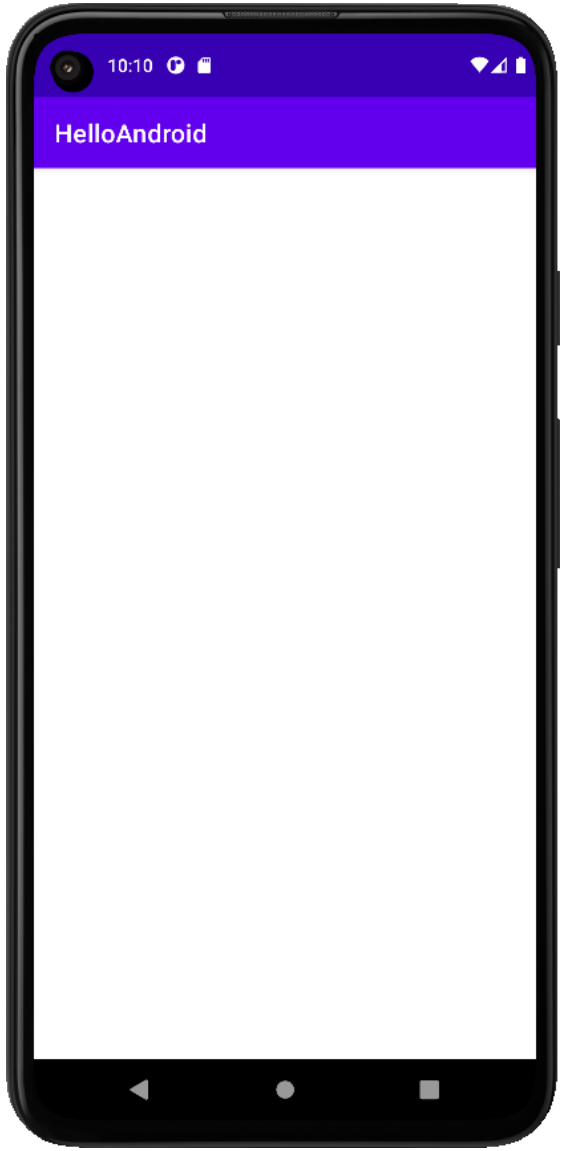
When one app invokes another, the calling app invokes an activity in the other app, rather than the app as an atomic whole. You implement an activity as a subclass of the Activity class.

Activity

An **Activity** is a crucial component of an Android app and a fundamental part of the Android application model.

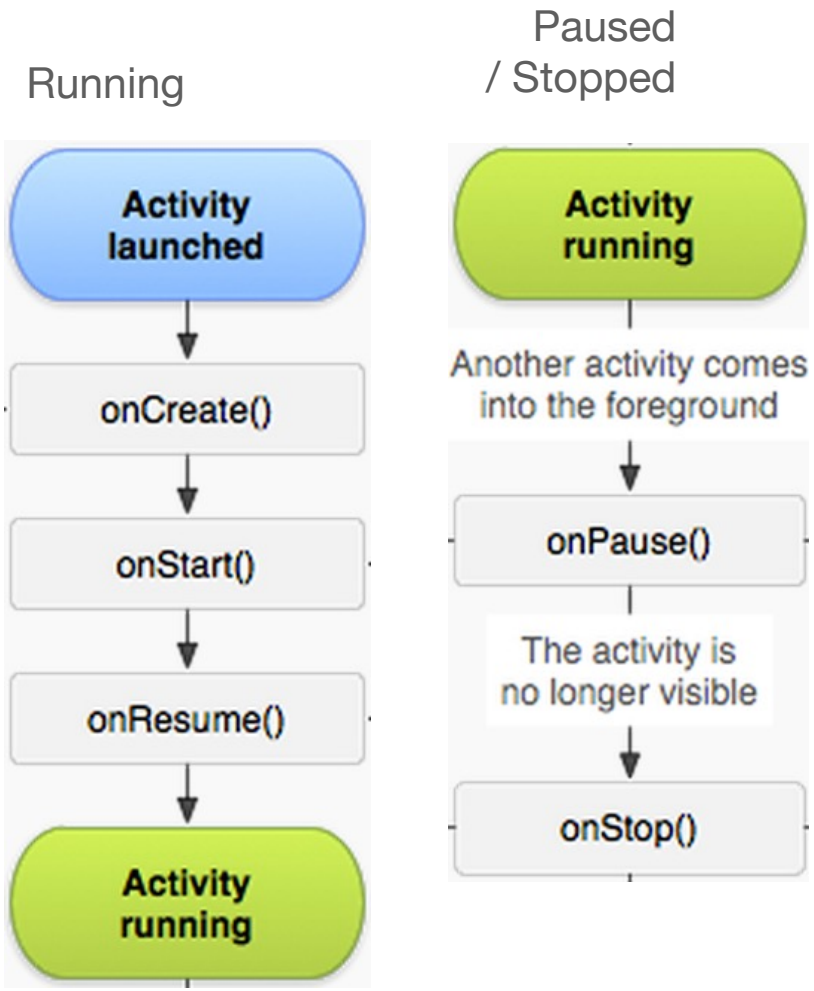
The Activity class creates a window to host a user interface. This window is almost always full-screen.

When one app invokes another, the calling app invokes an activity in the other app, rather than the app as an atomic whole. You implement an activity as a subclass of the Activity class.



Activity Lifecycle

- Activities have an explicit lifecycle, and have a state reflecting what they are doing
  - e.g. “started” or “stopped”
- Changing state fires a *callback method* that corresponds to that state
  - e.g. going from “stopped” to ”started” causes the *onStart()* method to fire.

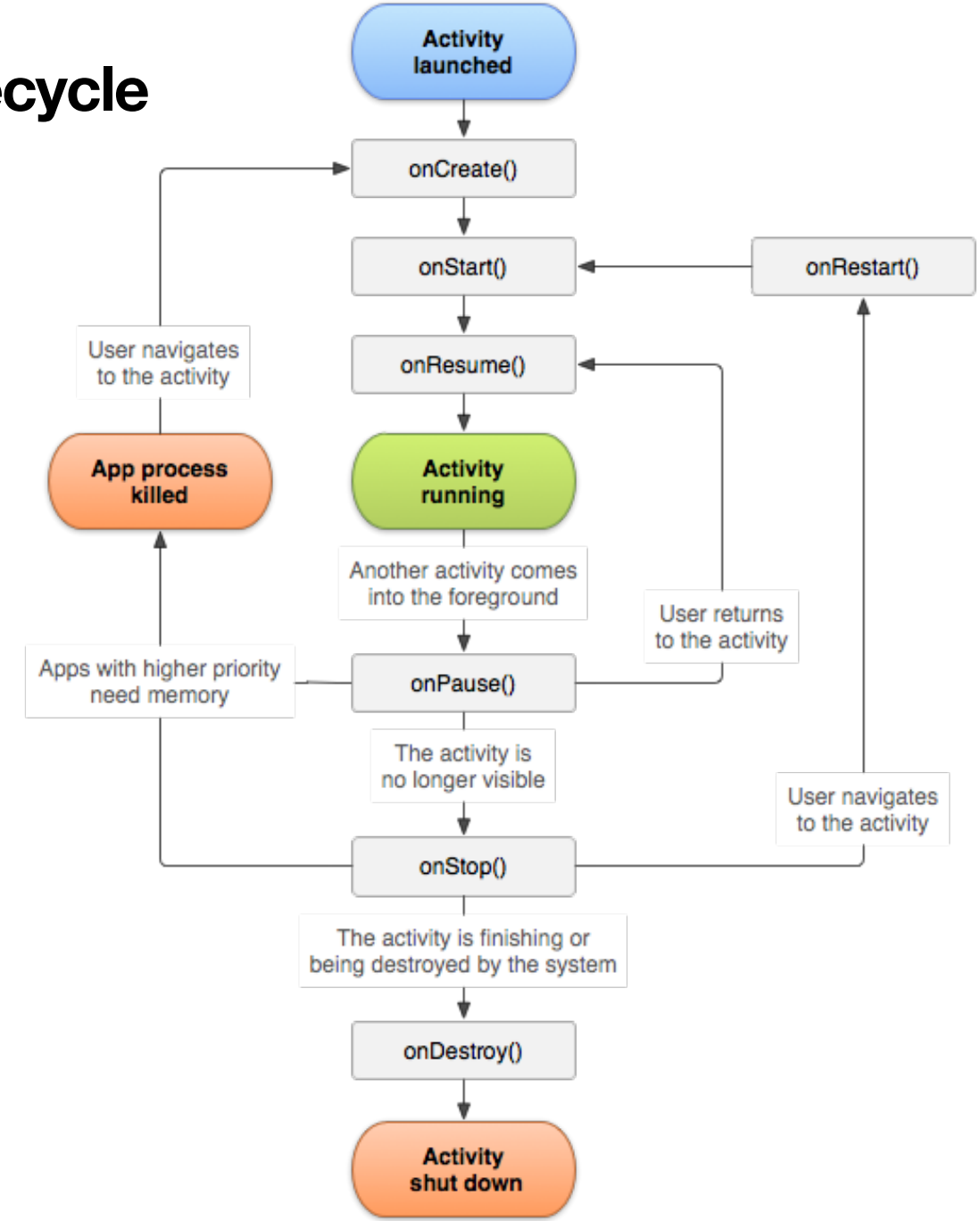


https://developer.android.com/guide/components/activities/activity-lifecycle.html

Managing the Activity Lifecycle

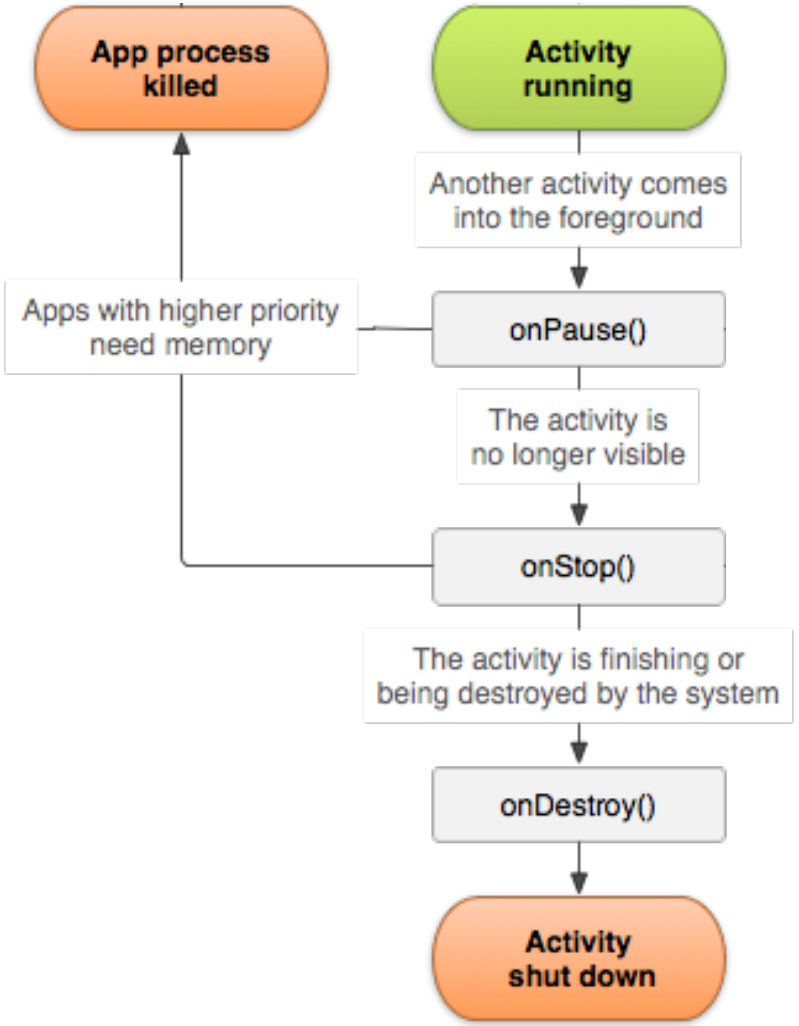
Core callback functions:

- onCreate() -- *required*
  - being created or launching
- onStart()
  - becomes visible to user
- onResume()
  - prior to user interaction
- onPause()
  - loses focus or background
- onStop()
  - no longer visible to user
- onDestroy()
  - being recycled and freed



Interrupted Workflow

- Applications can stop at any time (i.e. user quits, OS kills it, user presses the Back button, orientation changes.
  - the activity transitions through the onPause(), onStop(), and onDestroy() callbacks.
  - the activity is also removed from the stack.
- onRestoreInstanceState() is automatically called by the system after onStart().
- onSaveInstanceState() is automatically called by the system after onStop().
- To preserve simple transient-state data, override these methods
  - Save data in onSaveInstanceState()
  - Restore data in onRestoreInstanceState()



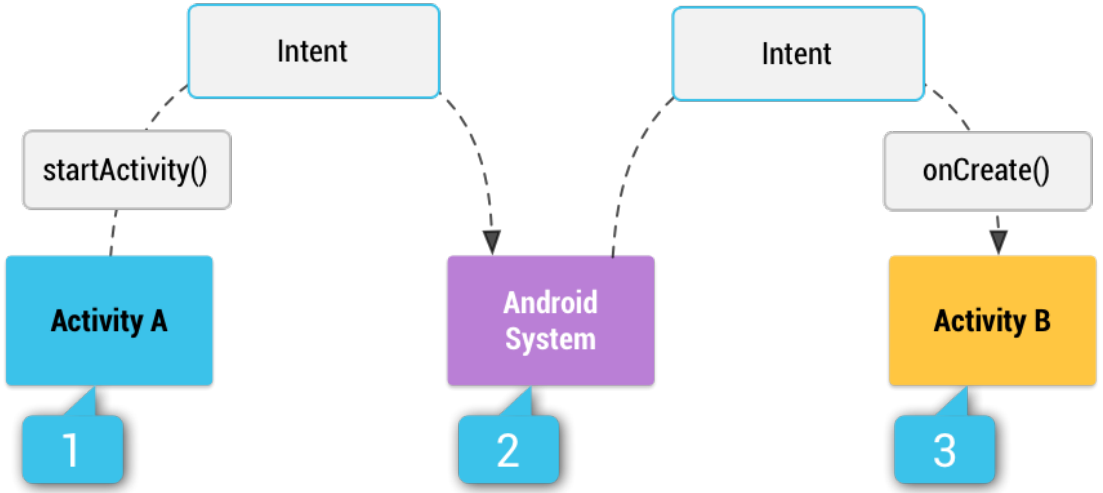
Intents

- An **Intent** is a messaging object you can use to request an action from another application component
  - Starting an activity
  - Starting a service
  - Delivering a broadcast
- This allows an application to use other application services! e.g. Instagram app can request access to the Camera activity to take a picture.
  - Eliminates the need to have functionality embedded in an application.
  - Allows the OS to control access/permissions to services.
- We use **intents** to pass data between activities.
  - Basically a data structure holding an abstract description of an action

https://developer.android.com/guide/components/intents-filters.html

Intents

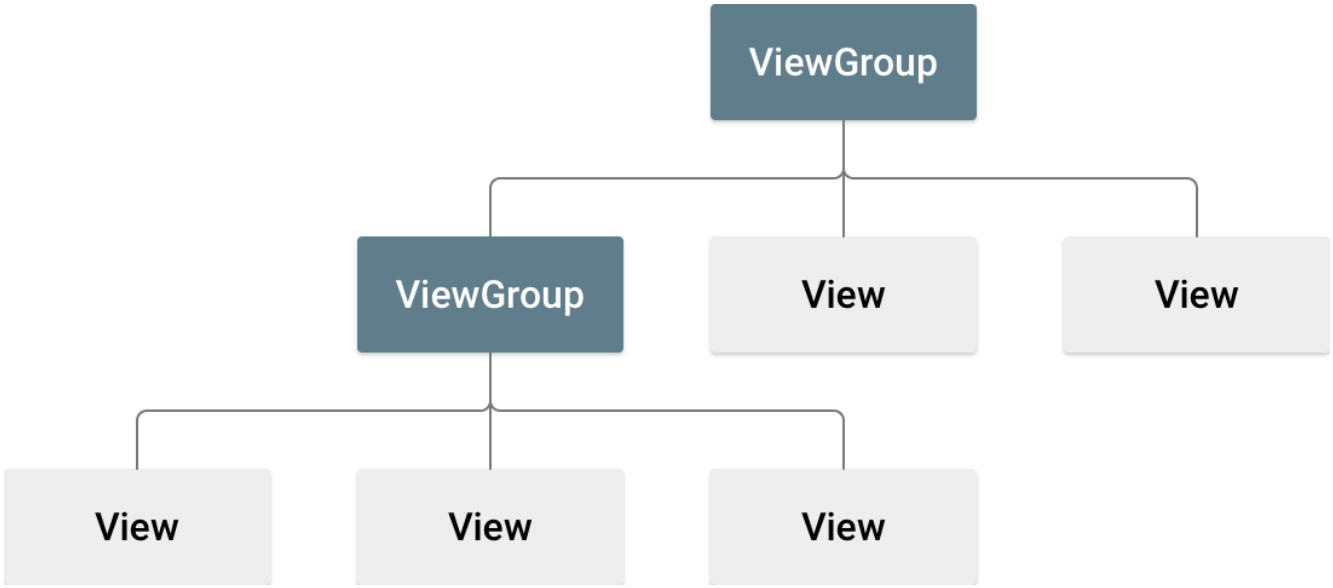
- Use *startActivity()* method to launch an activity with an intent.
  - Can call *explicit* named activity (e.g. mySettingsActivity) or an *implicit* activity based on its capabilities (e.g. some camera activity)



Android Scene Graph

Hierarchy of `ViewGroup` and `View` objects

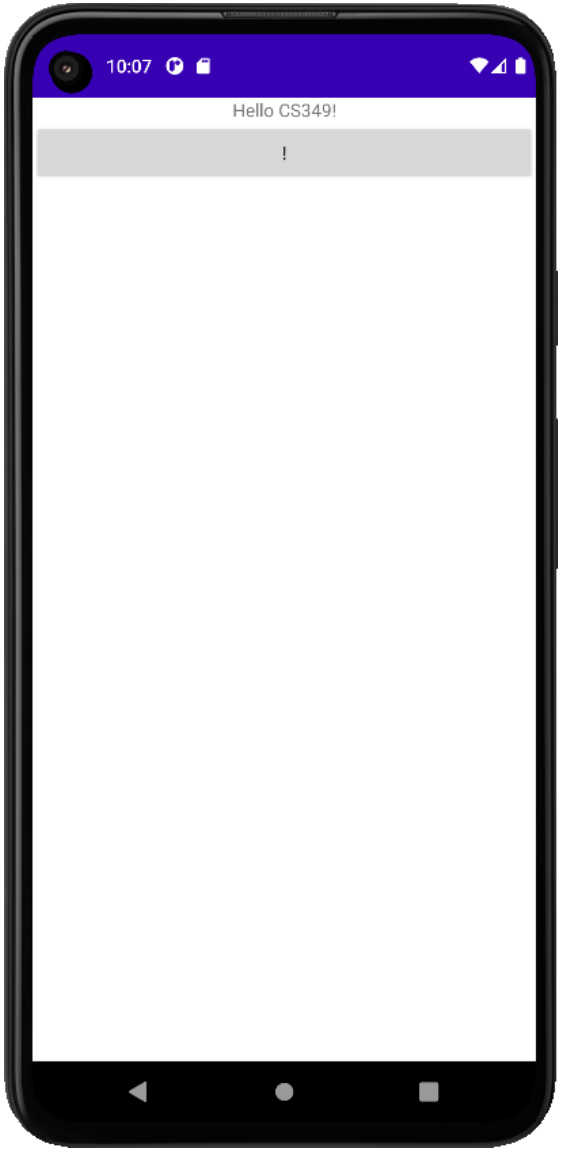
- A **`ViewGroup`** is is usually a **layout** container for child `Views` (e.g., `LinearLayout`)
- `A` **`View`** object is usually a **widget** (e.g., `Button`, `TextView`)



Imperative UI

MainActivity.kt

```
class MainActivity : FragmentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        val layout = LinearLayout(this).apply {
            orientation = LinearLayout.VERTICAL
        }

        val myText = TextView(this).apply {
            text = "Hello CS349!"
            textAlignment = View.TEXT_ALIGNMENT_CENTER
            layout.addView(this)
        }

        val button = Button(this).apply {
            text = "!"
            setOnClickListener {
                myText.text = "${myText.text}!"
            }
            layout.addView(this)
        }

        setContentView(layout)
    }
}
```



Declarative UI

activity_main.xml

```
<?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:orientation="vertical"
        tools:context=".MainActivity">

    <TextView android:id="@+id/myText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/greeting"/>

    <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/exclabutton"
            android:onClick="addExclamation"/>

</LinearLayout>
```

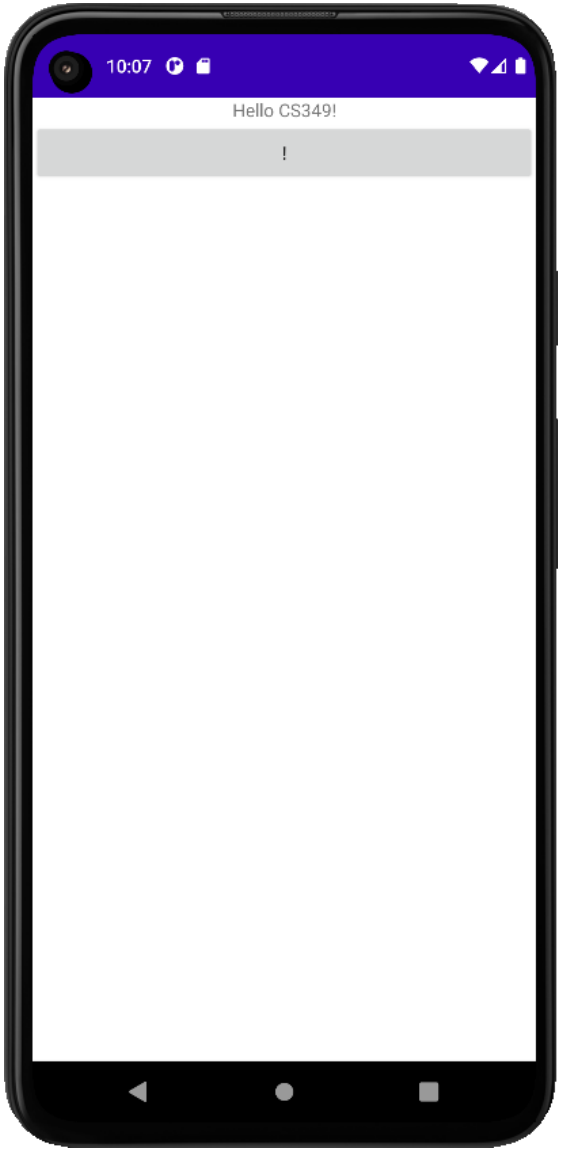
Declarative UI

MainActivity.kt

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

    fun addExclamation(view: View) {
        val text = findViewById<TextView>(R.id.myText)
        text.text = "${text.text}!"

    }
}
```

strings.xml

```
<resources>
    <string name="app_name">HelloAndroid</string>
    <string name="greeting">Hello CS349!</string>
    <string name="exclabutton">!</string>
</resources>
```



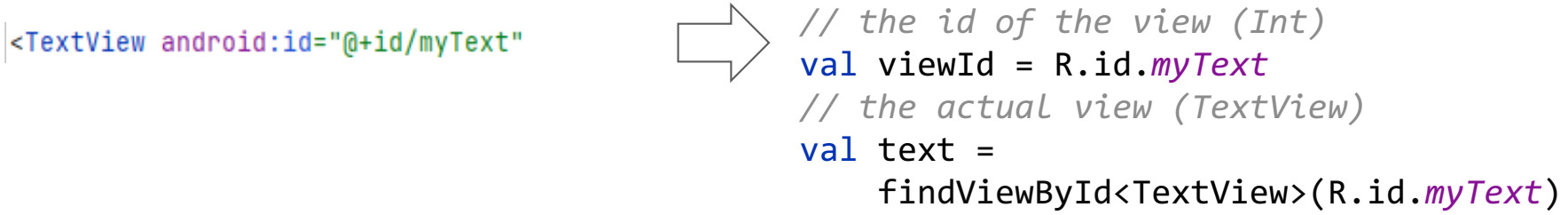
Resources and the R class

R is a static class with members generated from resources in /res

- R.layout.* are references to xml layouts in /layouts

- R.string.* are refs to string elements in /values/strings.xml

- R.id.* are refs to nodes in a layout with an id attribute



Example – R.values.color.xml

colors.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="math1">#FFFFBEEF</color>
    <color name="math2">#FFC60078</color>
</resources>
```

layout.xml-files:

```
<TextView android:id="@+id/myText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAlignment="center"
        android:textColor="@color/math1"
        android:text="@string/greeting"/>
```

.kt-files:

```
findViewById<TextView>(R.id.myText).apply {
    setTextColor(getColor(R.color.math2))
}
```

Example – R.values.strings.xml

```
<resources>
    <string name="app_name">HelloCS349</string>
    <string name="greeting">Hello, %1$s %2$d!</string>
</resources>
```

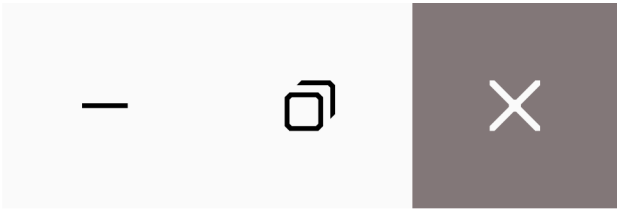
layout .xml-files:

```
<TextView android:id="@+id/myText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAlignment="center"
        android:textColor="@color/math1"
        android:text="@string/greeting"/>
```

.kt-files:

```
findViewById<TextView>(R.id.myText).apply {
    text.text = String.format(getString(R.string.greeting), "CS", 349)
}
```

Layouts



CS 349

Android Layout Units

Device pixels-per-inch is the pixel density and measured in **dpi**

- e.g., Google Pixel 6 Phone is 441 dpi = 441 pixels-per-inch

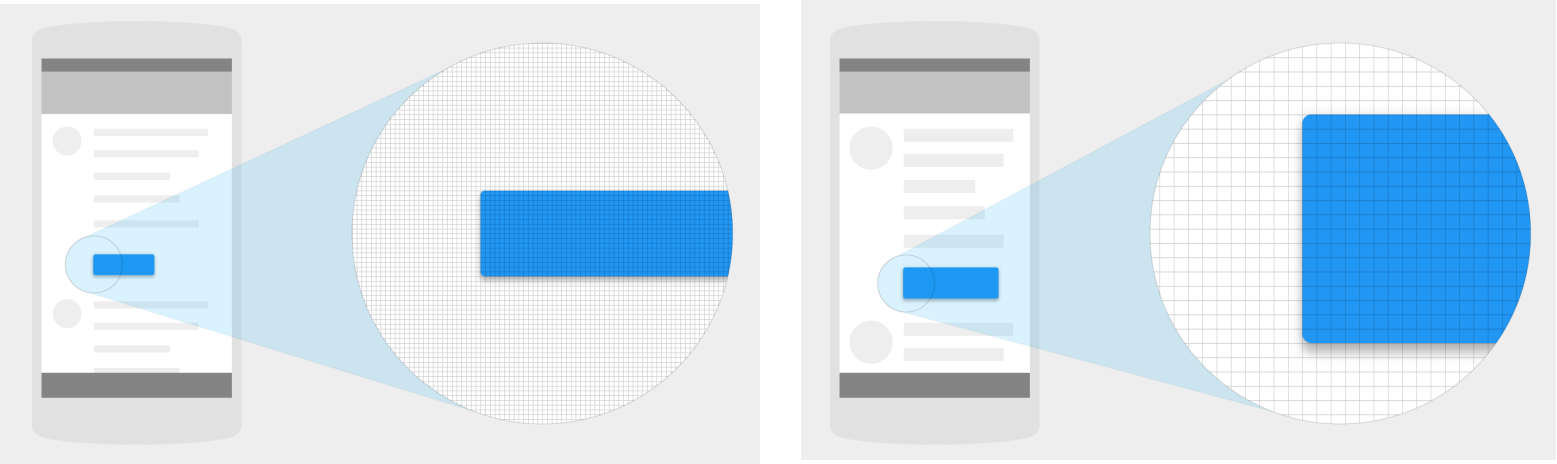
Views typically defined in **dp** (density-independent pixels: “dips”)

- 1 dp “virtual pixel” is equal to 1 pixel on a 160dpi phone

Text sizes should be defined in **sp** (scalable pixels)

- sp is the same as dp by default, but changes according to user’s preferred text size system setting

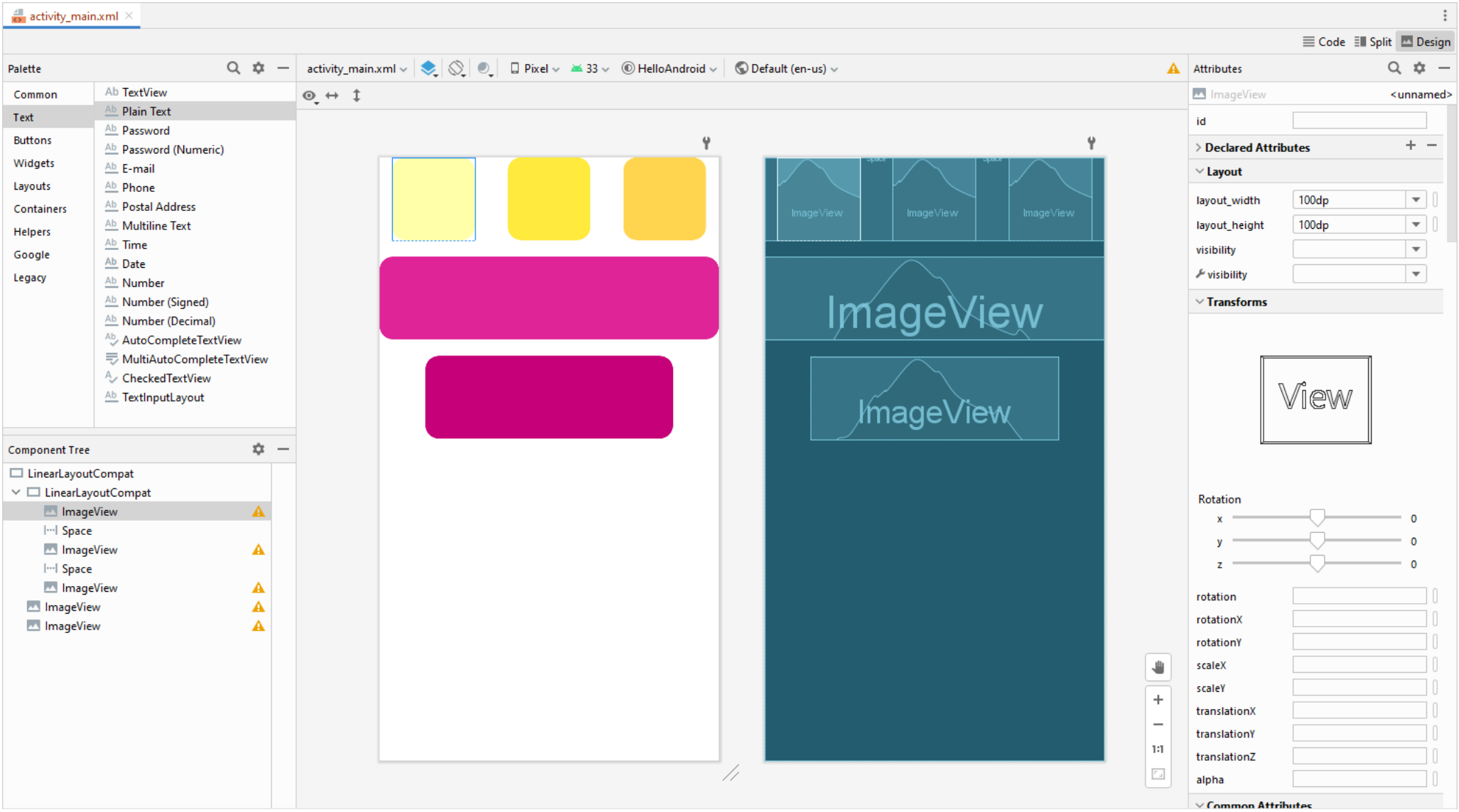
- Other units supported, e.g., **px**, **mm**, ...



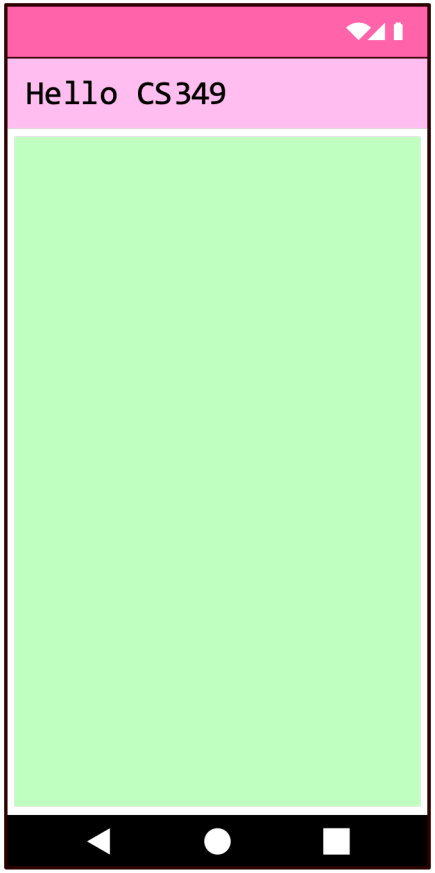
https://developer.android.com/training/multiscreen/screendensities

GUI Designer

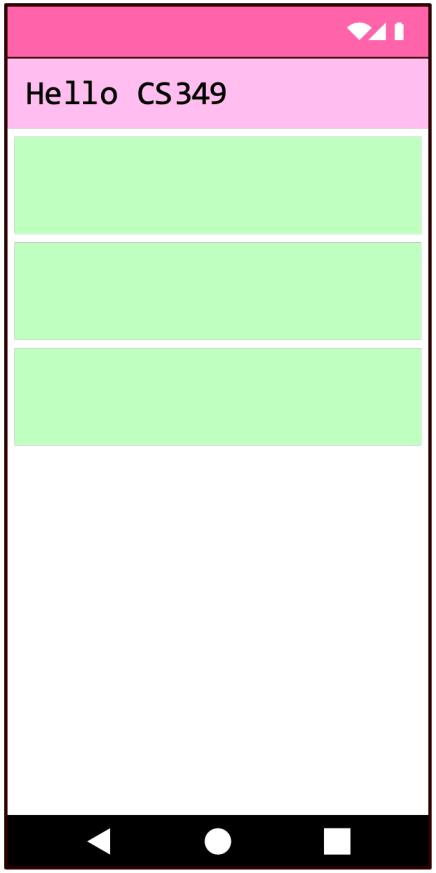
The GUI Designer allows to create a UI using a GUI instead of XML code.



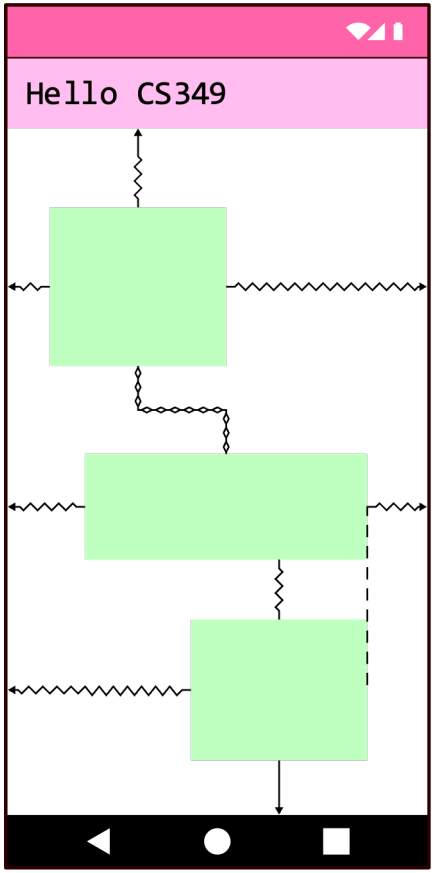
Common Layouts



`FrameLayout`
Displays a single `View` in its area.



LinearLayout
Organizes its children into a single (horizontal) row or (vertical) column.



ConstraintLayout
Relative positioning of children based on relationships and constraints.

LinearLayout

res.drawable.rectangle.xml

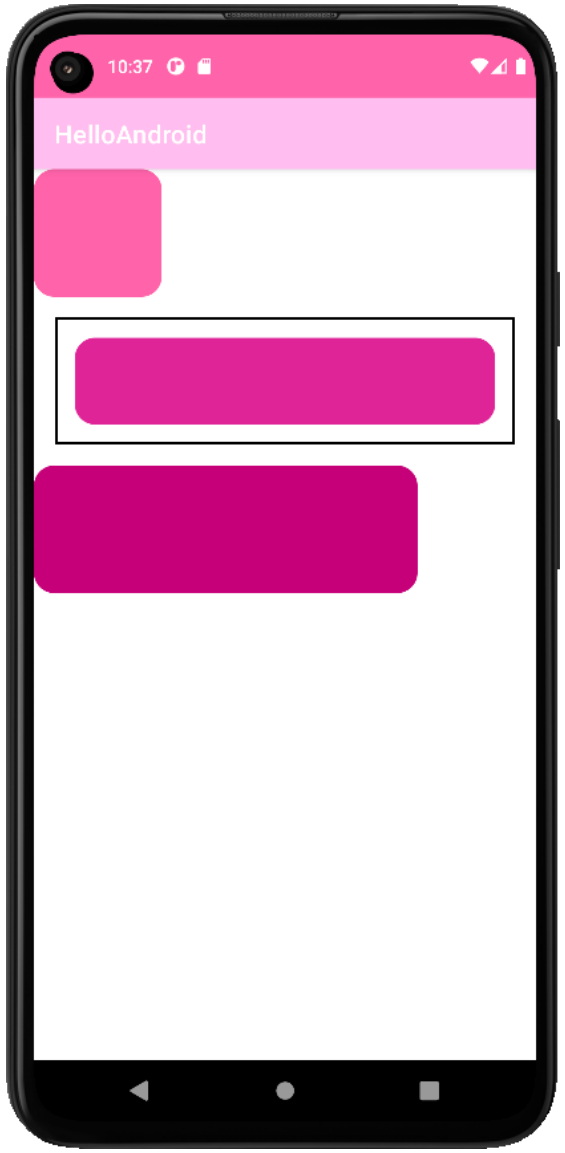
```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="..." android:shape="rectangle">
    <corners android:radius="16dp" />
</shape>
```

res.layout.activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="..." xmlns:tools="..."
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        tools:context=".MainActivity">

    <ImageView android:layout_height="100dp"
               android:layout_width="100dp"
               android:src="@drawable/rectangle"
               app:tint="@color/math1" />

    <ImageView android:layout_height="100dp"
               android:layout_width="200dp"
               android:src="@drawable/rectangle"
               app:tint="@color/math2" />

    <ImageView android:layout_height="100dp"
               android:layout_width="300dp"
               android:src="@drawable/rectangle"
               app:tint="@color/math3" />
</LinearLayout>
```

`LinearLayout` – Children's Attributes

```
<ImageView android:layout_height="100dp"
           android:layout_width="100dp"
           android:src="@drawable/rectangle"
           app:tint="@color/math1" />
<ImageView android:layout_height="100dp"
           android:layout_width="match_parent"
           android:src="@drawable/rectangle"
           app:tint="@color/math2"
           android:padding="16dp"
           android:layout_margin="16dp" />
<ImageView android:layout_height="100dp"
           android:layout_width="300dp"
           android:src="@drawable/rectangle"
           app:tint="@color/math3"
           android:onClick="rectTouched" />
```

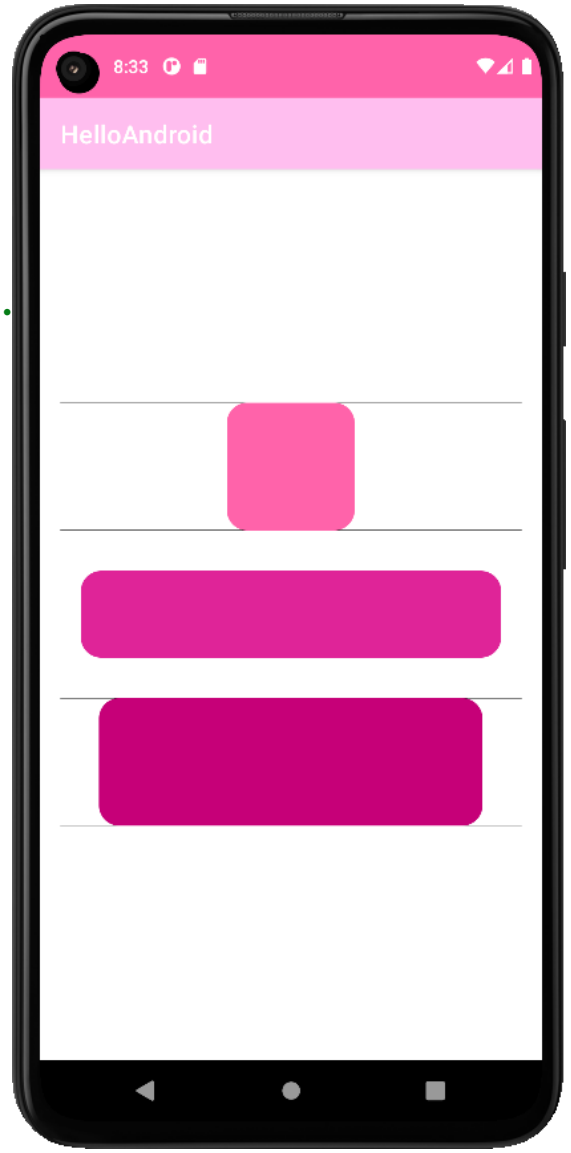


- `layout_width="match_parent"`: view dimension expands to match parent (also `layout_height`)
- `layout_width="wrap_content"`: view dimension matches size of internal content
- `padding="16dp"`: add space inside the boundary
- `layout_margin="16dp"`: add space outside the boundary
- `onClick="rectTouched"`: function called on touch

LinearLayout – Attributes

res.layout.activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
        xmlns:android="..." xmlns:app="..." xmlns:tools="..
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:gravity="center"
        app:divider="@color/black"
        app:dividerPadding="16dp"
        app:showDividers="end|middle|beginning"
        tools:context=".MainActivity">

    ...

</LinearLayout>
```



- `orientation="vertical"`: shows content as a column (also `"horizontal"`)
- `gravity="center"`: how children are positioned
- `showDividers="end|middle|beginning"`: shows dividers between children

`LinearLayout` – Spacing

`LinearLayout` has no attribute for spacing between child views.

Workarounds include:

- Adding margins to each child nodes:

```
android:layout_marginVertical="10dp"
```

- Inserting `Space` views between child nodes:

```
<Space android:layout_width="match_parent“
       android:layout_height="20dp" />
```

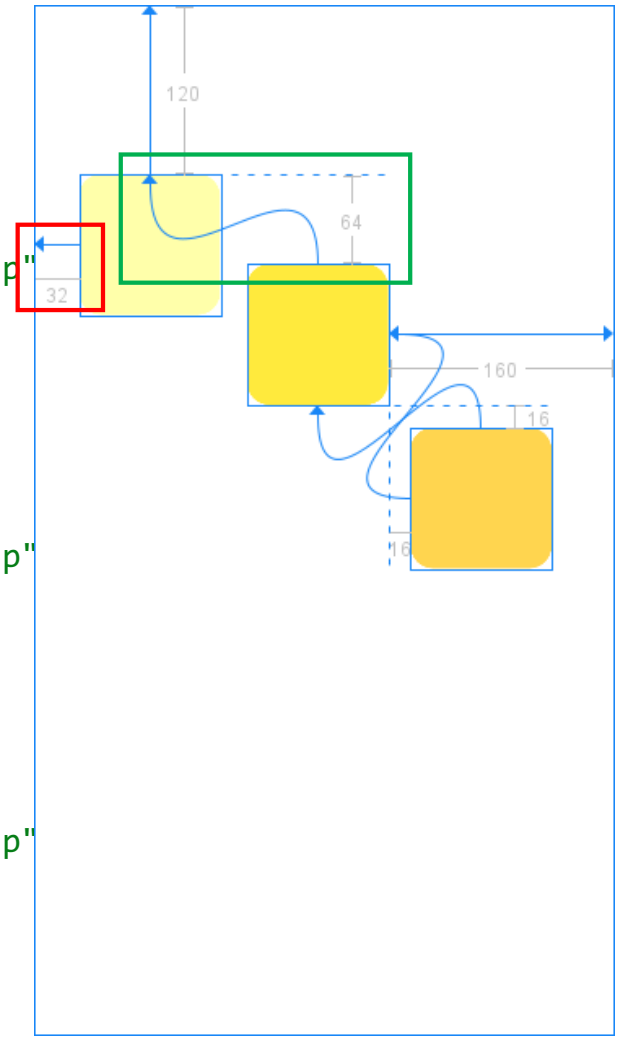
LinearLayout – Nesting Layouts

```
<LinearLayout
        xmlns:android="..." xmlns:app="..." xmlns:tools="..."
        android:layout_width="match_parent" android:layout_height="match_parent"
        android:orientation="vertical" android:gravity="center_horizontal"
        app:divider="@drawable/divider" app:showDividers="middle"
        tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent" android:layout_height="wrap_content"
        android:orientation="horizontal" android:gravity="center_horizontal">

        <ImageView android:layout_height="100dp" android:layout_width="100dp"
                   android:src="@drawable/rectangle" app:tint="@color/uw1" />
        <Space android:layout_height="0dp" android:layout_width="40dp" />
        <ImageView android:layout_height="100dp" android:layout_width="100dp"
                   android:src="@drawable/rectangle" app:tint="@color/uw2" />
        <Space android:layout_height="0dp" android:layout_width="40dp" />
        <ImageView android:layout_height="100dp" android:layout_width="100dp"
                   android:src="@drawable/rectangle" app:tint="@color/uw3" />
    </LinearLayout>

    <ImageView android:layout_height="100dp" android:layout_width="match_parent"
               android:src="@drawable/rectangle" app:tint="@color/math3" />

    <ImageView android:layout_height="100dp" android:layout_width="300dp"
               android:src="@drawable/rectangle" app:tint="@color/math4" />


</LinearLayout>
```

`ConstraintLayout` – Absolute Positioning

Absolute positioning, relative to parent or another node:

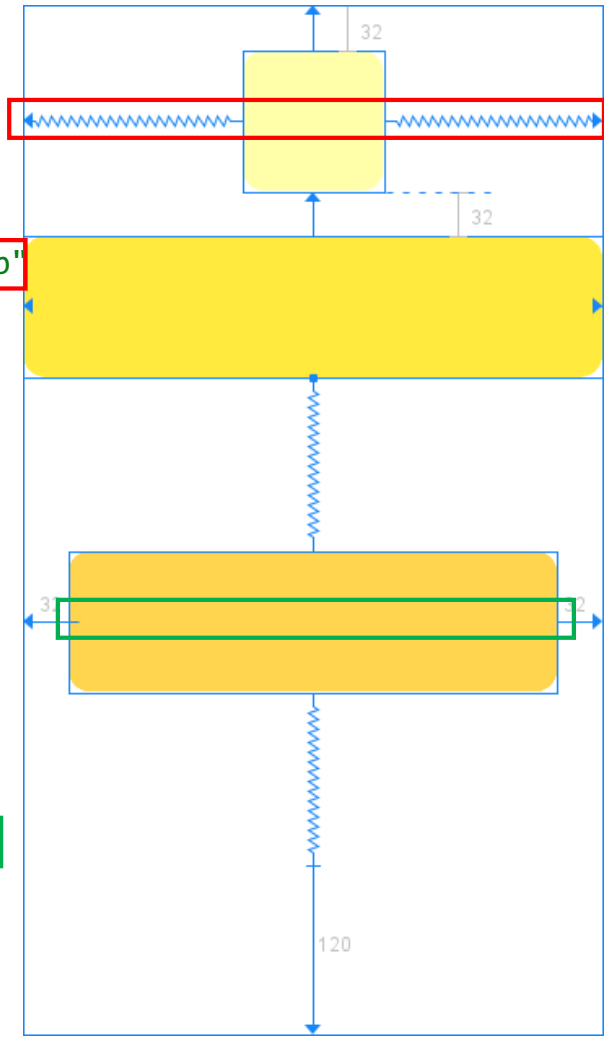
```
<ImageView android:id="@+id/imageView1"
           android:layout_height="100dp" android:layout_width="100dp"
           android:src="@drawable/rectangle" app:tint="@color/uw1"
           app:layout_constraintStart_toStartOf="parent"
           app:layout_constraintTop_toTopOf="parent"
           android:layout_marginStart="32dp"
           android:layout_marginTop="120dp" />
<ImageView android:id="@+id/imageView2"
           android:layout_height="100dp" android:layout_width="100dp"
           android:src="@drawable/rectangle" app:tint="@color/uw2"
           app:layout_constraintTop_toTopOf="@+id/imageView1"
           app:layout_constraintEnd_toEndOf="parent"
           android:layout_marginEnd="160dp"
           android:layout_marginTop="60dp" />
<ImageView
           android:layout_height="100dp" android:layout_width="100dp"
           android:src="@drawable/rectangle" app:tint="@color/uw3"
           app:layout_constraintTop_toBottomOf="@+id/imageView2"
           app:layout_constraintStart_toEndOf="@+id/imageView2"
           android:layout_marginTop="16dp"
           android:layout_marginStart="16dp" />
```



`ConstraintLayout` – Centered Positioning

Positioning between two points with fixed or calculated node dimension.

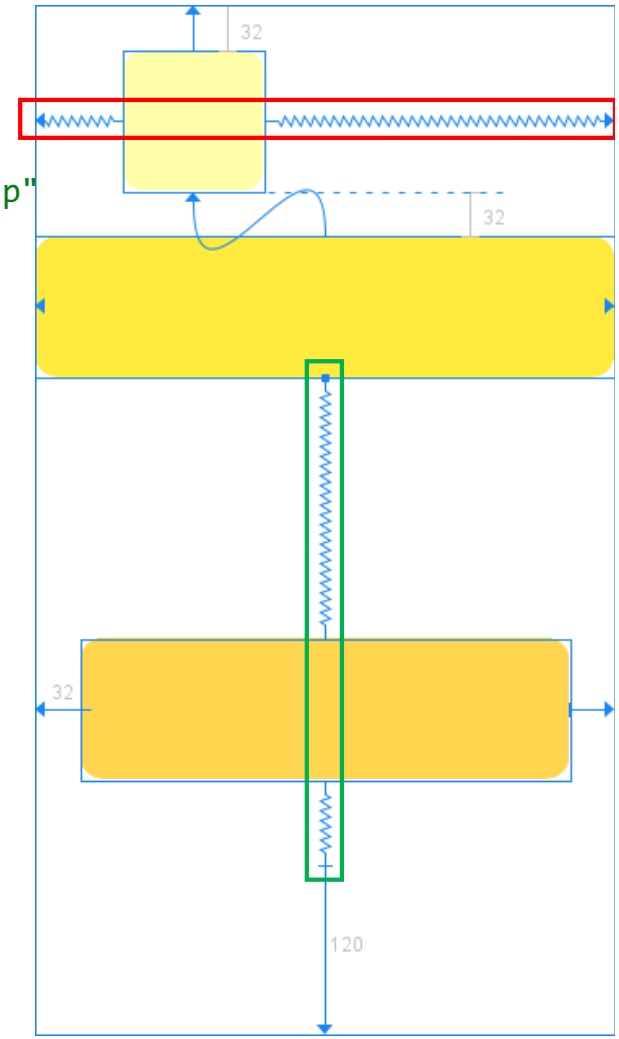
```
<ImageView android:id="@+id/imageView1"
           android:layout_height="100dp" android:layout_width="100dp"
           android:src="@drawable/rectangle" app:tint="@color/uw1"
           app:layout_constraintStart_toStartOf="parent"
           app:layout_constraintEnd_toEndOf="parent"
           app:layout_constraintTop_toTopOf="parent"
           android:layout_marginTop="32dp"/>
<ImageView android:id="@+id/imageView2"
           android:layout_height="100dp" android:layout_width="0dp"
           android:src="@drawable/rectangle" app:tint="@color/uw2"
           app:layout_constraintStart_toStartOf="parent"
           app:layout_constraintEnd_toEndOf="parent"
           app:layout_constraintTop_toBottomOf="@+id/imageView1"
           android:layout_marginTop="32dp"/>
<ImageView
           android:layout_height="100dp" android:layout_width="0dp"
           android:src="@drawable/rectangle" app:tint="@color/uw3"
           app:layout_constraintTop_toBottomOf="@+id/imageView2"
           app:layout_constraintBottom_toBottomOf="parent"
           app:layout_constraintStart_toStartOf="parent"
           app:layout_constraintEnd_toEndOf="parent"
           android:layout_marginBottom="120dp"
           android:layout_marginStart="32dp"
           android:layout_marginEnd="32dp"/>
```



ConstraintLayout – Centered Positioning

Offsetting (“biasing”) nodes

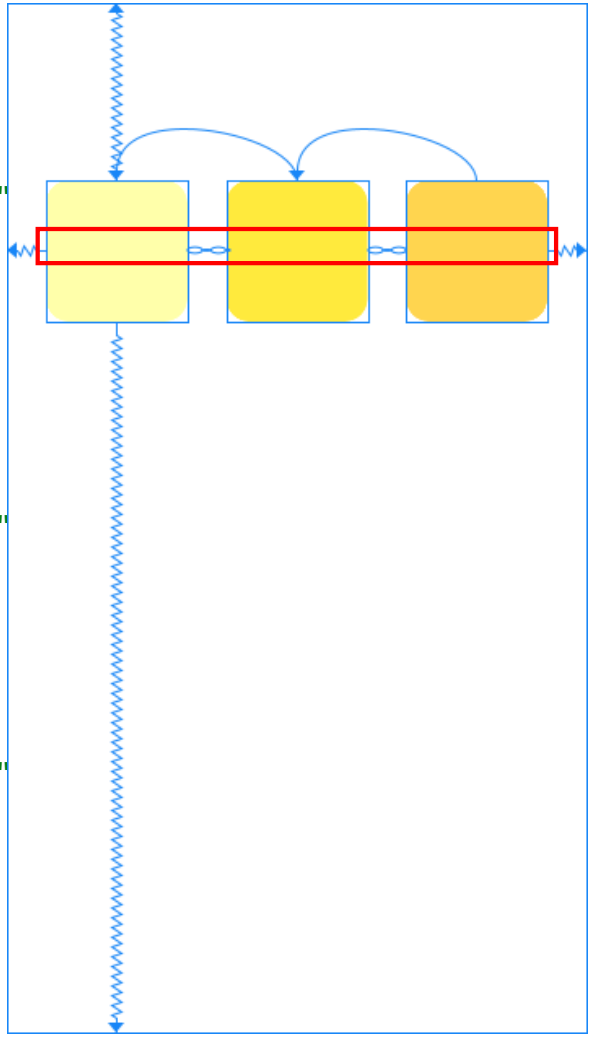
```
<ImageView android:id="@+id/imageView1"
           android:layout_height="100dp" android:layout_width="100dp"
           android:src="@drawable/rectangle" app:tint="@color/uw1"
           app:layout_constraintStart_toStartOf="parent"
           app:layout_constraintEnd_toEndOf="parent“
           app:layout_constraintHorizontal_bias="0.2"
           app:layout_constraintTop_toTopOf="parent"
           android:layout_marginTop="32dp"/>
<ImageView android:id="@+id/imageView2"
           android:layout_height="100dp" android:layout_width="0dp"
           android:src="@drawable/rectangle" app:tint="@color/uw2"
           app:layout_constraintStart_toStartOf="parent"
           app:layout_constraintEnd_toEndOf="parent"
           app:layout_constraintTop_toBottomOf="@+id/imageView1"
           android:layout_marginTop="32dp"/>
<ImageView
           android:layout_height="100dp" android:layout_width="0dp"
           android:src="@drawable/rectangle" app:tint="@color/uw3"
           app:layout_constraintTop_toBottomOf="@+id/imageView2"
           app:layout_constraintBottom_toBottomOf="parent"
           app:layout_constraintStart_toStartOf="parent"
           app:layout_constraintEnd_toEndOf="parent“
           app:layout_constraintVertical_bias="0.75"
           android:layout_marginBottom="120dp"
           android:layout_marginStart="32dp"
           android:layout_marginEnd="32dp"/>
```



`ConstraintLayout` – Chains

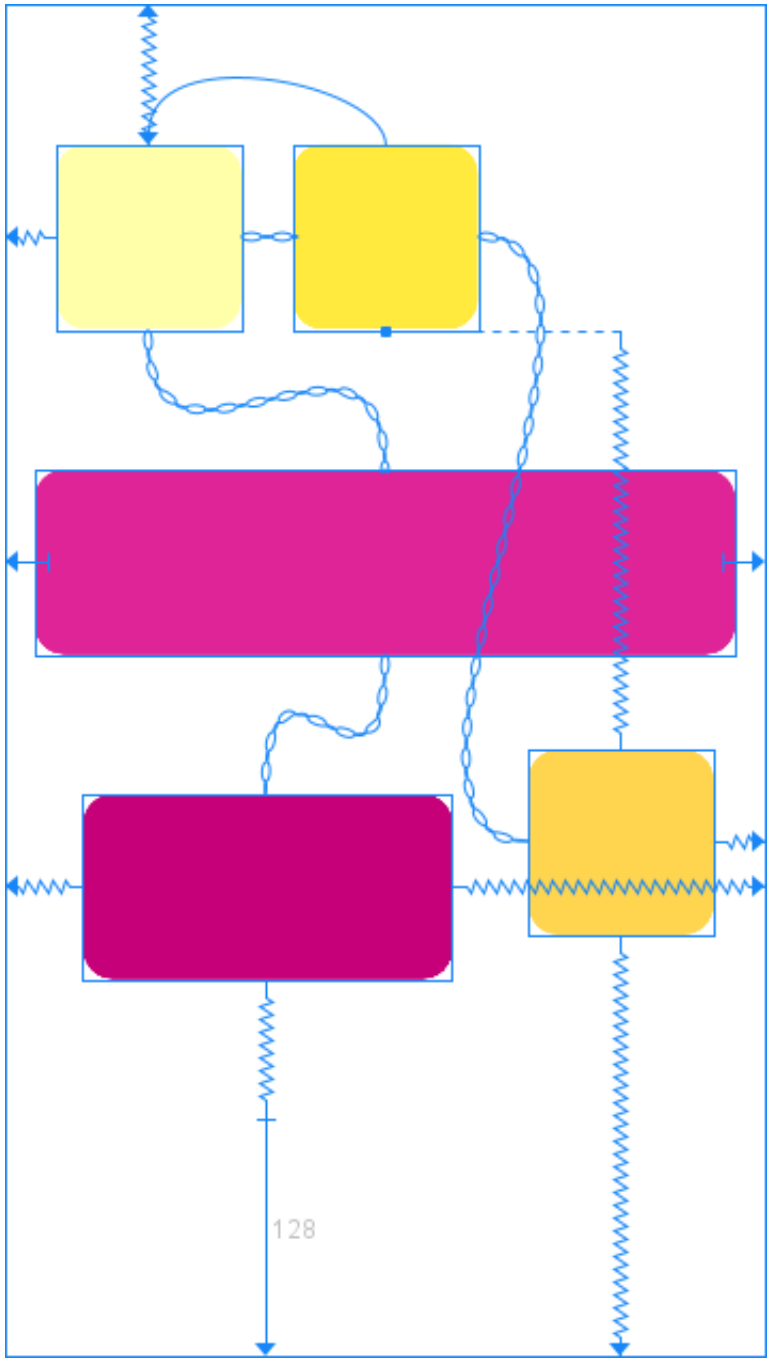
Chains group nodes together in one dimension:

```
<ImageView android:id="@+id/imageView1"
           android:layout_height="100dp" android:layout_width="100dp"
           android:src="@drawable/rectangle" app:tint="@color/uw1"
           app:layout_constraintStart_toStartOf="parent"
           app:layout_constraintEnd_toStartOf="@+id/imageView2"
           app:layout_constraintTop_toTopOf="parent"
           app:layout_constraintBottom_toBottomOf="parent"
           app:layout_constraintVertical_bias="0.2" />
<ImageView android:id="@+id/imageView2"
           android:layout_height="100dp" android:layout_width="100dp"
           android:src="@drawable/rectangle" app:tint="@color/uw2"
           app:layout_constraintStart_toEndOf="@+id/imageView1"
           app:layout_constraintEnd_toStartOf="@+id/imageView3"
           app:layout_constraintTop_toTopOf="@+id/imageView1" />
<ImageView android:id="@+id/imageView3"
           android:layout_height="100dp" android:layout_width="100dp"
           android:src="@drawable/rectangle" app:tint="@color/uw3"
           app:layout_constraintStart_toEndOf="@+id/imageView2"
           app:layout_constraintEnd_toEndOf="parent"
           app:layout_constraintTop_toTopOf="@+id/imageView2" />
```



`ConstraintLayout` – Chains

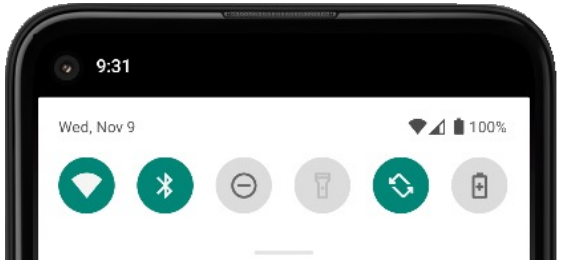
Combining multiple chains…



Layouts and Device Rotation

By default, Android handles device rotation by rotating the content of the app. Part of this process includes recreating the app, i.e., calling `onDestroy()` and `onCreate()`.

When testing, make sure AVD has device rotation ON:

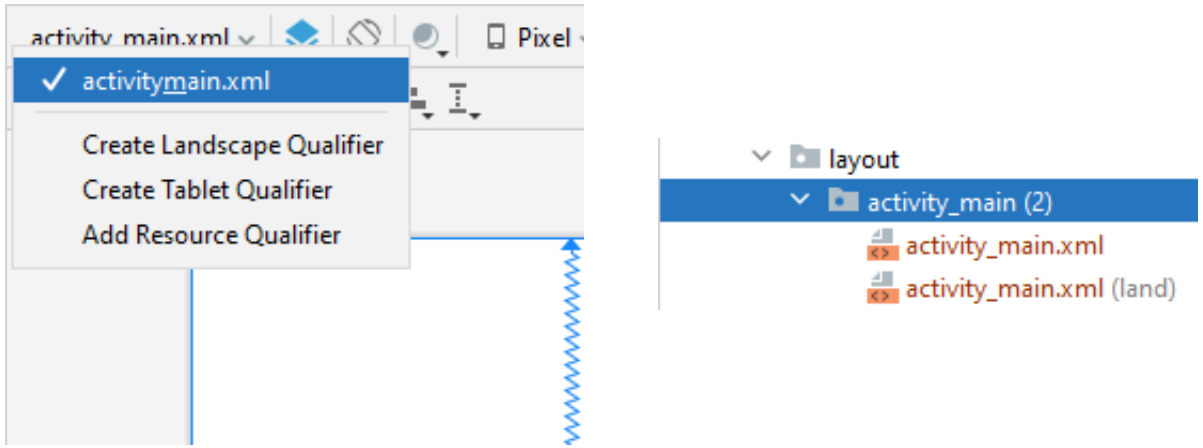


If the default behaviour is not sufficient, it is possible to handle device rotation within the app.

Layouts and Device Rotation

Rotating the device causes a *configuration change*

You may specify separate layouts for landscape and portrait orientation or use the same for both.



Layouts and Device Orientation

To react programmatically to orientation change:

- Add this to your main activity in `AndroidManifest.xml`:

```
android:configChanges="orientation|screenSize"
```

- Add this to your main activity, e.g., in `MainActivity.kt`:

```
override fun onConfigurationChanged(newConfig: Configuration) {
    super.onConfigurationChanged(newConfig)
    when (newConfig.orientation) {
        Configuration.ORIENTATION_LANDSCAPE -> {
            Log.i("OrientationChange", "Landscape")}
        Configuration.ORIENTATION_PORTRAIT -> {
            Log.i("OrientationChange", "Portrait") }
        else -> {
            Log.e("OrientationChange", "Whaaat...") }
    }
}
```

Layouts and Device Rotation

For more fine-grain information about the rotation angle:

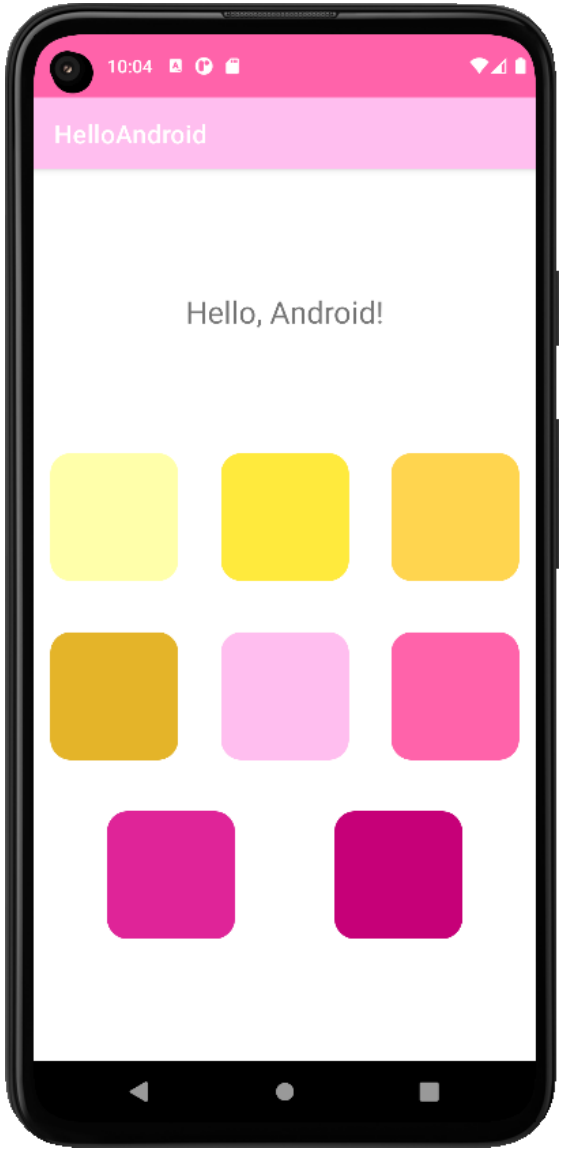
- Add this to your main activity in `AndroidManifest.xml`:

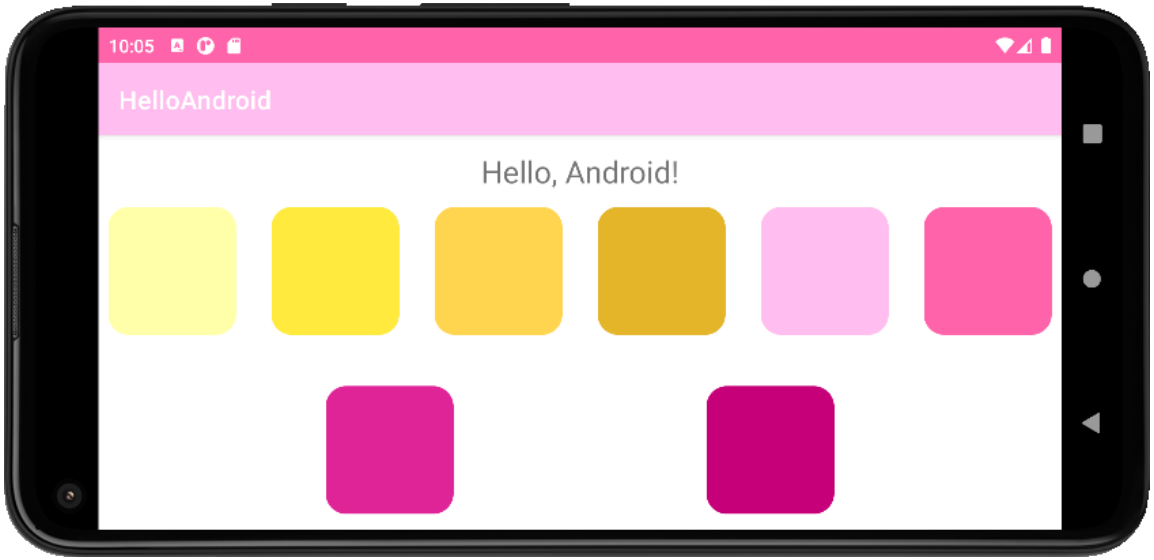
```
android:configChanges="orientation|screenSize"
```

- Add this to your `.kt` file, e.g., `MainActivity.kt`:

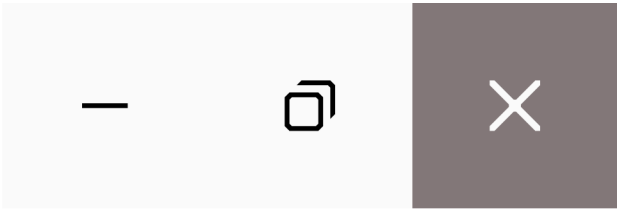
```
val oel = object : OrientationEventListener(applicationContext) {
    override fun onOrientationChanged(degrees: Int) {
        Log.d("OrientationEvent",
              "New device orientation is $degrees degrees.")
    }
}
oel.enable()
```

Layouts and Device Rotation





Android Widgets



CS 349

TextView

`TextView` display text to the user.

```
<TextView android:id="@+id/myTextView"
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:text="@string/greeting"
          android:layout_marginTop="32dp" />
```

TextView – Common Properties

.xml-file:

```
<TextView android:id="@+id/textView"
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:text="@string/greeting"
          android:textSize="24sp"
          android:typeface="monospace"
          android:textAlignment="center"
          android:textColor="@color/math4"
          android:background="@color/uw3"
          android:layout_margin="@dimen/viewMargin"
          android:padding="16dp" />
```

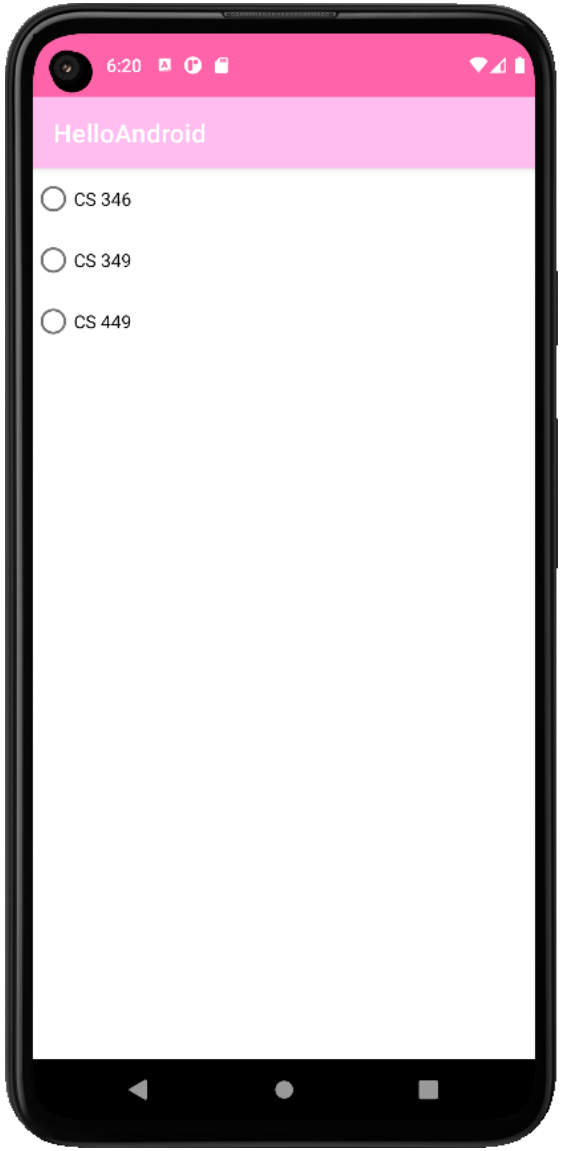
.kt-file:

```
fun textViewTap(view: View) {
    findViewById<TextView>(R.id.textView).apply {
        text = "$text!"
    }
    // alternative implementation using casting
    (view as TextView).apply {
        text = "$text!"
    }
}
```

RadioButton

Radio buttons allow the user to select one option from a set. Use radio buttons if you think that the user needs to see all available options side-by-side.

```
<RadioGroup android:layout_width="wrap_content"
            android:layout_height="wrap_content">
    <RadioButton android:id="@+id/radBtn346"
            android:text="@string/radBtnLbl346"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    <RadioButton android:id="@+id/radBtn349"
            android:text="@string/radBtnLbl346"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    <RadioButton android:id="@+id/radBtn449"
            android:text="@string/radBtnLbl449"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
</RadioGroup>
```



RadioButton – Common Properties

.xml-file:

```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" android:gravity="center"
    android:checkedButton="@id/rb349">
    <RadioButton android:id="@+id/rb349"
                 android:text="@string/rbl349"
                 android:layout_width="wrap_content"
                 android:layout_height="wrap_content"
                 android:onClick="radioGroupClicked" />
    <Space android:layout_width="40dp"
           android:layout_height="wrap_content" />
    <RadioButton android:id="@+id/rb449"
                 android:text="@string/rbl449"
                 android:layout_width="wrap_content"
                 android:layout_height="wrap_content"
                 android:onClick="radioGroupClicked" />
</RadioGroup>
```

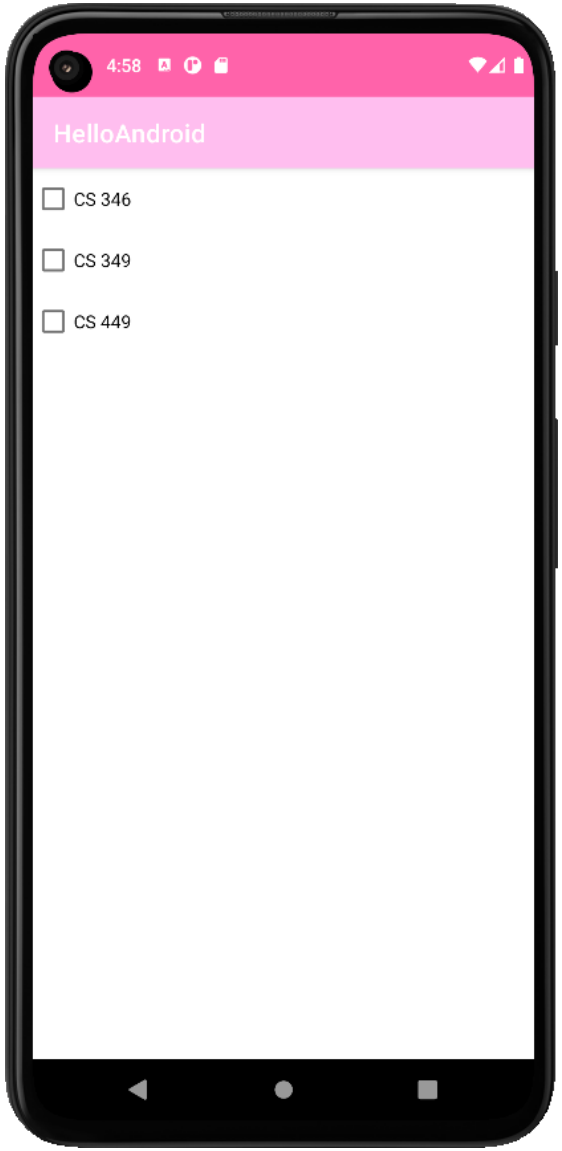
.kt-file:

```
fun radioGroupClicked(view: View) {
    Log.i("RADIO", view.id.toString())
    when (view.id) {
        R.id.rb349 -> { }
        R.id.rb449 -> { }
        else -> { }
    }
}
```

CheckBox

Checkboxes allow the user to select one or more options from a set. Typically, they should be presented in a vertical list.

```
<CheckBox android:id="@+id/cb346"
          android:text="@string/cbl346"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content" />
<CheckBox android:id="@+id/cb349"
          android:text="@string/cbl349"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content" />
<CheckBox android:id="@+id/cb449"
          android:text="@string/cbl449"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content" />
```



CheckBox – Common Properties

.xml-file

```
<CheckBox android:id="@+id/cb349"
          android:text="@string/cbl349"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:onClick="checkBoxClicked" />
<CheckBox android:id="@+id/cb449"
          android:text="@string/cbl449"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:checked="true"
          android:onClick="checkBoxClicked" />
```

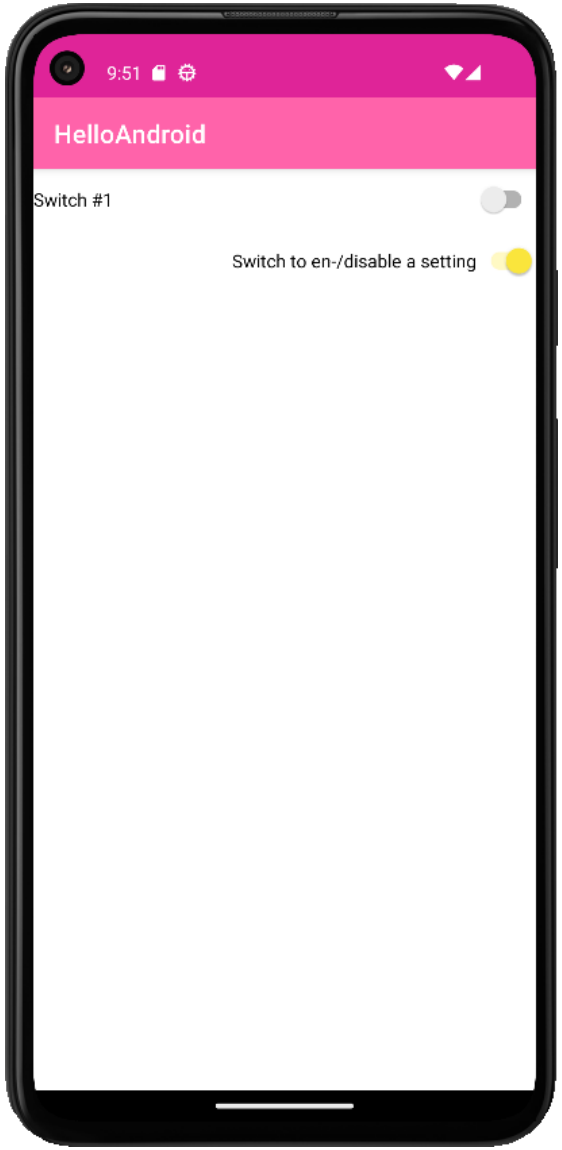
.kt-file

```
fun checkBoxClicked(view: View) {
    view as CheckBox
    Log.i("CHECK", "${view.id}:${view.isChecked}")
    when (view.id) {
        R.id.cb349 -> { }
        R.id.cb449 -> { }
        else -> { }
    }
}
```

Switch

Switches allow the user to chose between two options from a single property.

```
<Switch
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Switch #1" />
<Switch
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Switch to en-/disable a setting"/>
```



Switch – Custom Styling

.xml-file:

```
<Switch android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:checked="true" android:showText="true"
        android:track="@drawable/track" android:thumb="@drawable/thumb
        android:textOn="Right" android:textOff="Left"
        android:text="Side" android:padding="@dimen/viewPadding" />
<Switch android:id="@+id/colorSwitch"
        android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:background="#FFF0F0F0"
        android:text="Switch" android:padding="@dimen/viewPadding" />
```

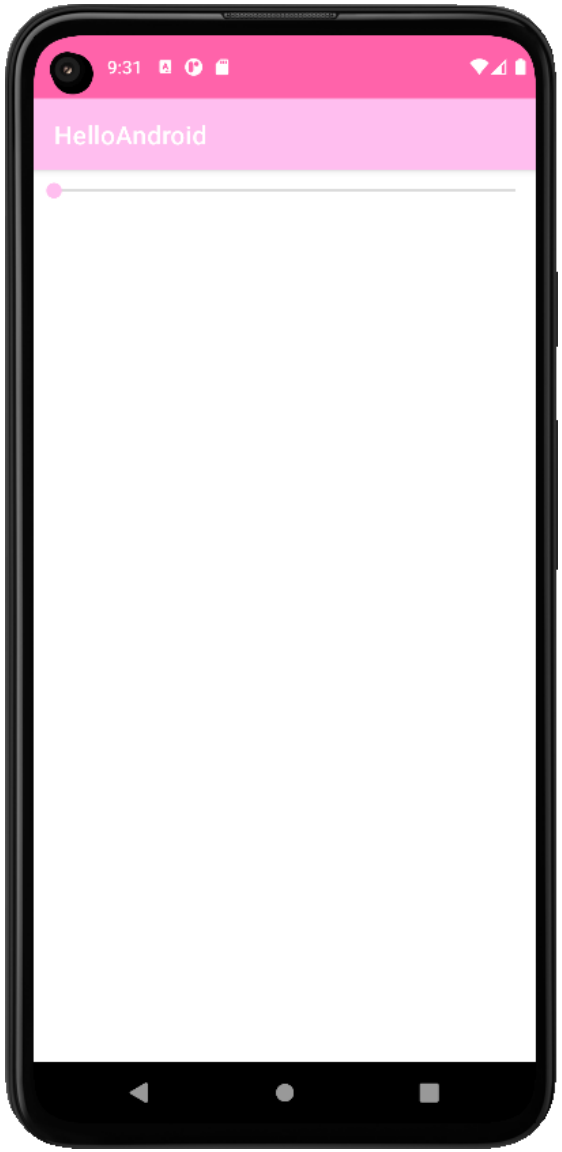
.kt-file:

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

    findViewById<Switch>(R.id.colorSwitch).setOnCheckedChangeListener
    { view, isChecked ->
        view as Switch
        view.text = String.format(getString(R.string.switchText),
                                  isChecked)
        if (isChecked)
            view.thumbDrawable.setTint(getColor(R.color.white))
        else
            view.thumbDrawable.setTint(getColor(R.color.black))
    }
}
```

SeekBar

Seek bars allow users to select a continues value for a property.

```
<SeekBar android:id="@+id/seekBar"
         android:layout_width="match_parent"
         android:layout_height="32dp" />
```



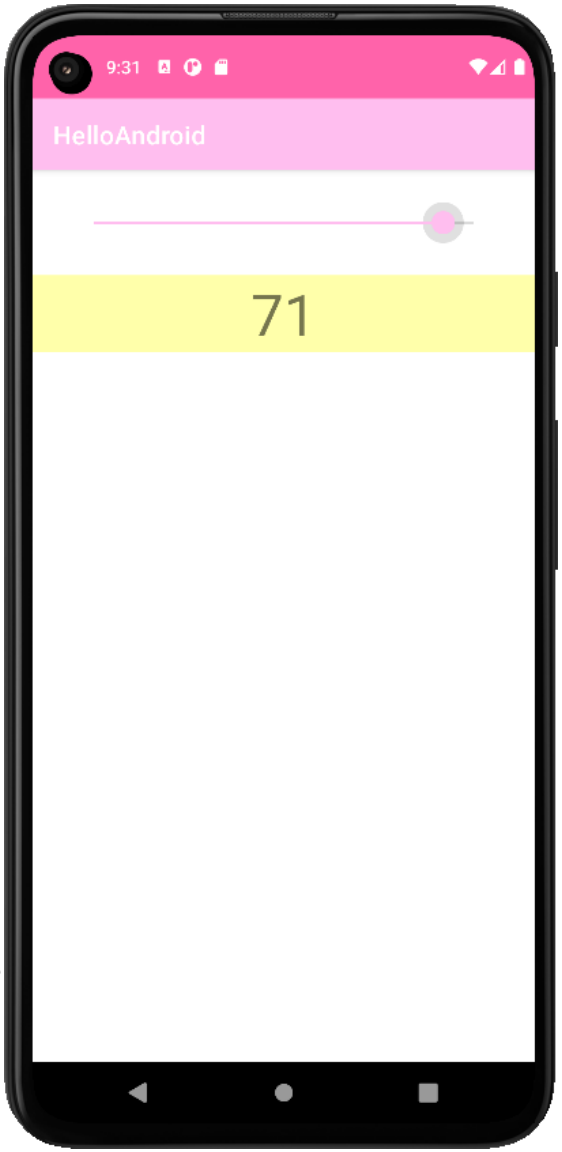
SeekBar – Custom Styling

.xml-file:

```
<SeekBar android:id="@+id/seekBar"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:min="25" android:max="75"
         android:layout_margin="@dimen/viewMargin" />
<TextView android:id="@+id/seekBarValue"
          android:layout_width="match_parent"
          android:layout_height="60dp"
          android:autoSizeTextType="uniform"
          android:textAlignment="center"
          android:background="@color/uw1" />
```

.kt-file:

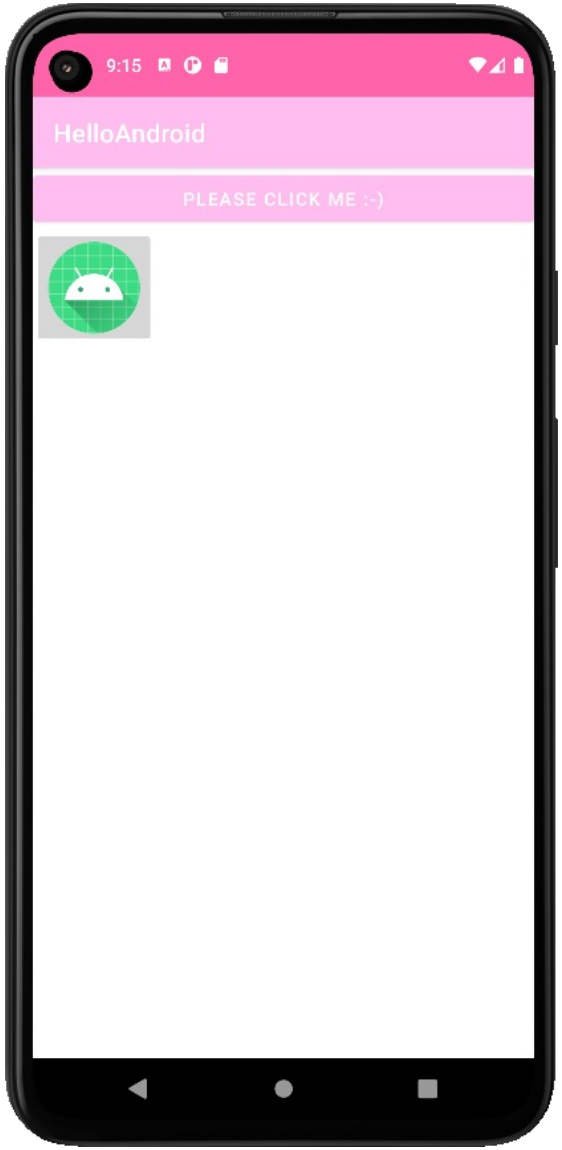
```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    val seekBar = findViewById<SeekBar>(R.id.seekBar)
    val seekText = findViewById<TextView>(R.id.seekBarValue)
    seekBar.setOnSeekBarChangeListener(object: OnSeekBarChangeListener
        override fun onProgressChanged(view: SeekBar?, value: Int,
                                       fromUser: Boolean) {
            seekText.text = value.toString()
        }
        override fun onStartTrackingTouch(p0: SeekBar?) {  }
        override fun onStopTrackingTouch(p0: SeekBar?) {  }
    })
}
```



Button, ImageButton

A button consists of text or an icon (or both text and an icon), which should communicate what action occurs when the user touches it.

```
<Button android:id="@+id/myBtn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/btnText" />

<ImageButton android:id="@+id/myImgBtn"
             android:layout_width="match_parent"
             android:layout_height="wrap_content"
             android:src="@mipmap/ic_launcher" />
```



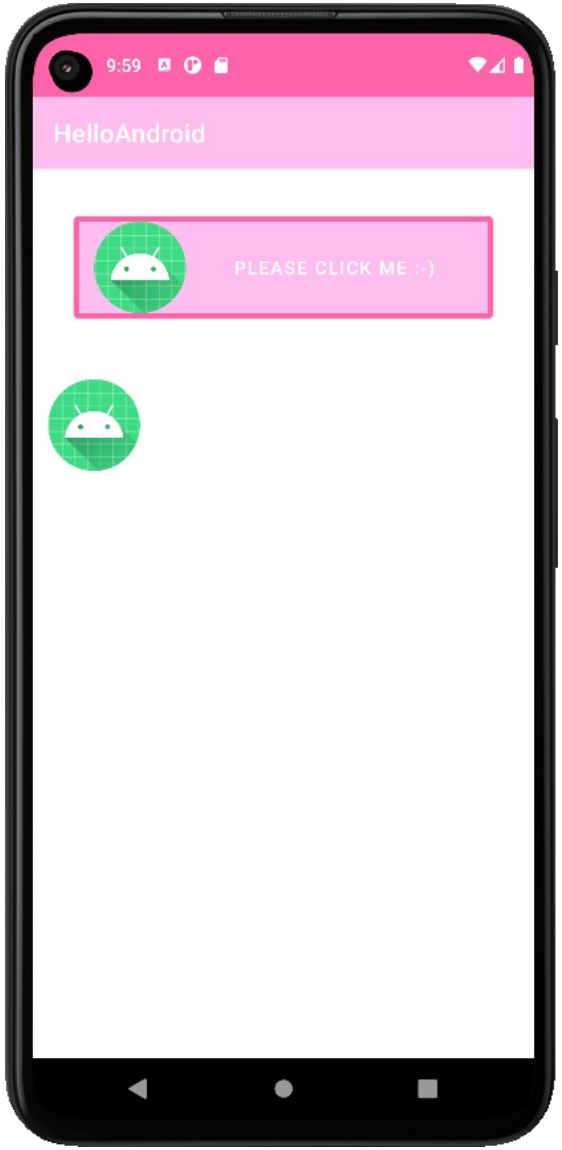
Button, ImageButton – Common Properties

.xml-file:

```
<Button android:id="@+id/myBtn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="@dimen/viewMargin"
        android:text="@string/btnText"
        android:drawableLeft="@mipmap/ic_launcher"
        android:onClick="btnClicked"
        app:strokeWidth="4dp"
        app:strokeColor="@color/math2"
        app:cornerRadius="4dp" />

<ImageButton android:id="@+id/myImgBtn"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:src="@mipmap/ic_launcher"
             android:backgroundTint="@color/transparent"
             android:onClick="btnClicked" />
```

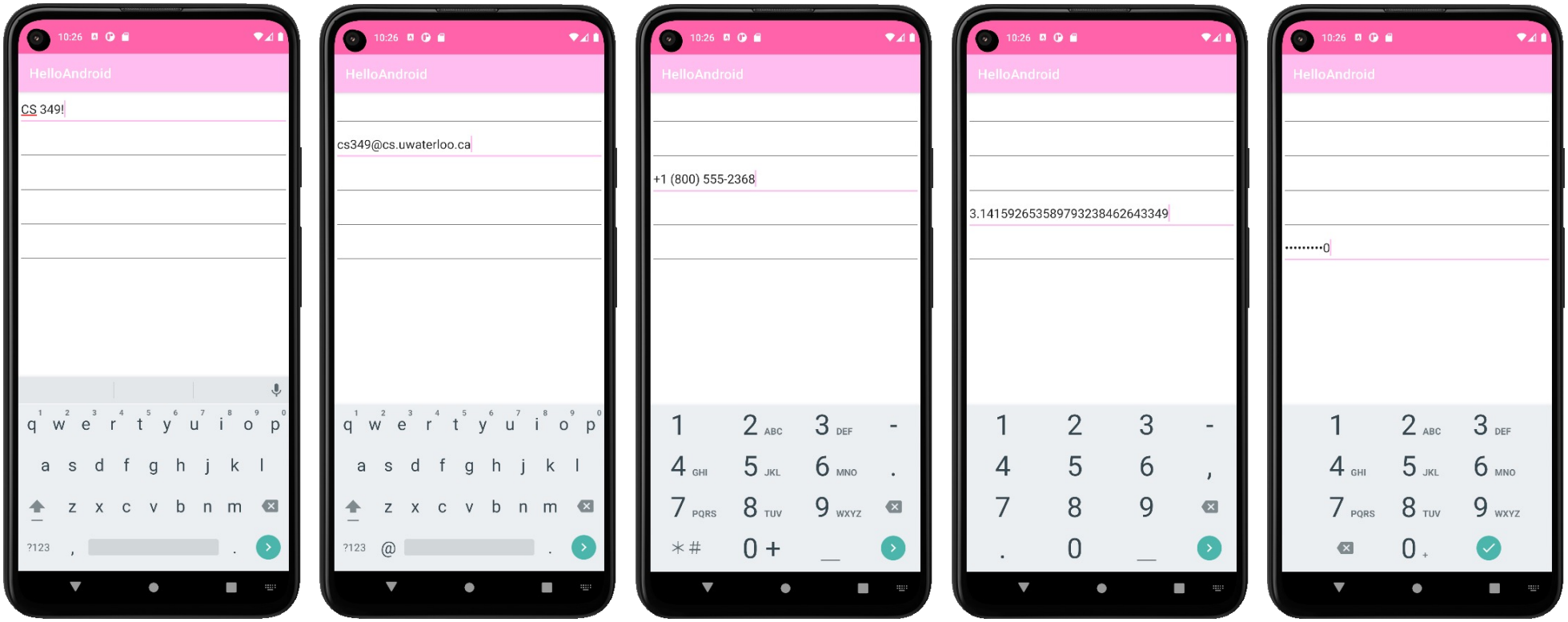
.kt-file:

```
fun btnClicked(view: View) {
    when(view.id) {
        R.id.myBtn -> { Log.i("BUTTON", "myBtn clicked") }
        R.id.myImgBtn -> { Log.i("BUTTON", "myIMgBtn clicked") }
    }
}
```



EditText

```
<EditText android:inputType="text"
          android:layout_width="match_parent" android:layout_height="50dp" />
<EditText android:inputType="textEmailAddress"
          android:layout_width="match_parent" android:layout_height="50dp" />
<EditText android:inputType="phone"
          android:layout_width="match_parent" android:layout_height="50dp" />
<EditText android:inputType="numberDecimal"
          android:layout_width="match_parent" android:layout_height="50dp" />
<EditText android:inputType="numberPassword"
          android:layout_width="match_parent" android:layout_height="50dp" />
```



EditText – Validation

.xml-file:

```
<EditText android:id="@+id/email"
          android:layout_width="match_parent"
          android:layout_height="50dp"
          android:inputType="textEmailAddress“
          android:maxLength="@integer/emailMaxLen" />
<LinearLayout
       android:layout_height="wrap_content"
       android:layout_width="match_parent"
       android:gravity="end">
    <Button android:id="@+id/btnSubmit"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Submit" android:onClick="btnSubmit"
            android:enabled="false" />
    <Button android:id="@+id/btnClear"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Clear" android:onClick="btnClear"
            android:enabled="false" />
</LinearLayout>
```

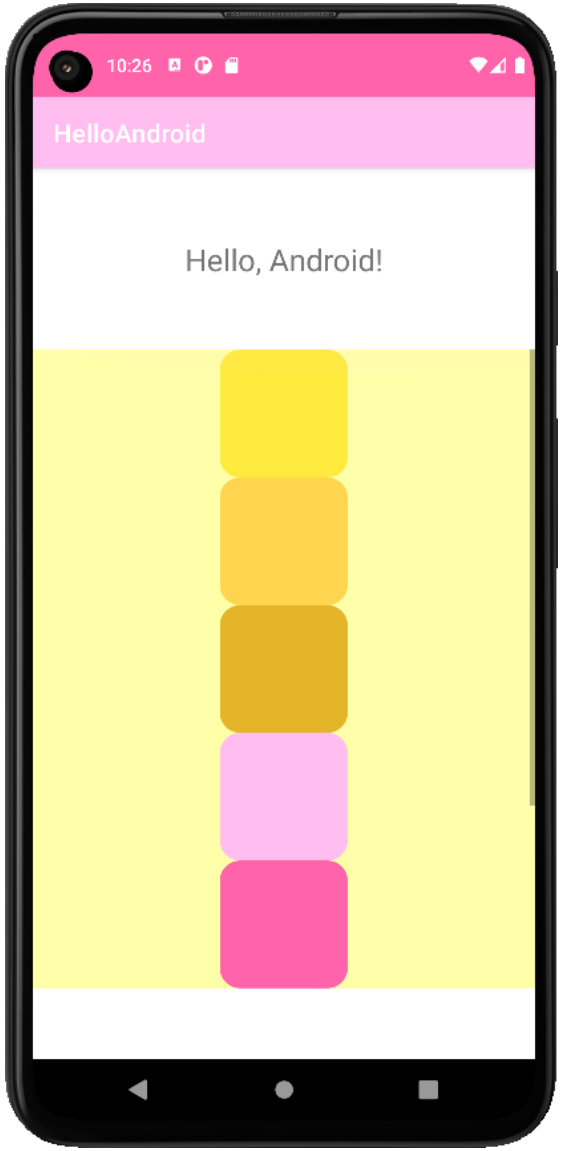
EditText – Validation

.kt-file:

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    // ...
    findViewById<EditText>(R.id.email).addTextChangedListener(
        object : TextWatcher {
        override fun beforeTextChanged(p0: CharSequence?,
                                       p1: Int, p2: Int, p3: Int) { }
        override fun onTextChanged(seq: CharSequence?,
                                   p1: Int, p2: Int, p3: Int) {
            val emailRegex =
                Regex("^[a-zA-Z0-9+]+@[a-zA-Z0-9]{2,}\\.(?:ca|com)\$")
            findViewById<Button>(R.id.btnSubmit).isEnabled =
                seq?.matches(emailRegex) ?: false
            findViewById<Button>(R.id.btnClear).isEnabled =
                seq.isNullOrEmpty().not()
        }
        override fun afterTextChanged(p0: Editable?) { }
    })
}

fun btnSubmit(view: View) {
    Log.i("BUTTON",
          "Submitting ${findViewById<EditText>(R.id.email).text}.")
}

fun btnClear(view: View) {
    findViewById<EditText>(R.id.email).text.clear()
}
```

Layouts and Scrolling

```
<TextView android:id="@+id/textView" android:text="@string/greeting"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:textSize="24sp"
          app:layout_constraintStart_toStartOf="parent“
          app:layout_constraintEnd_toEndOf="parent"
          app:layout_constraintTop_toTopOf="parent"
          app:layout_constraintBottom_toTopOf="@+id/scrollView2" />
<ScrollView android:id="@+id/scrollView"
            android:layout_width="match_parent"
            android:layout_height="500dp"
            app:layout_constraintTop_toBottomOf="@+id/textView"
            app:layout_constraintBottom_toBottomOf="parent">
        <LinearLayout android:layout_width="match_parent“
                      android:layout_height="wrap_content"
                      android:orientation="vertical“
                      android:gravity="center"
                      android:backgroundTint="@color/uw1">
                      <ImageView android:layout_height="100dp"
                                 android:layout_width="100dp“
                                 android:src="@drawable/rectangle"
                                 app:tint="@color/uw2" />
                      ...
        </LinearLayout>
</ScrollView>
```



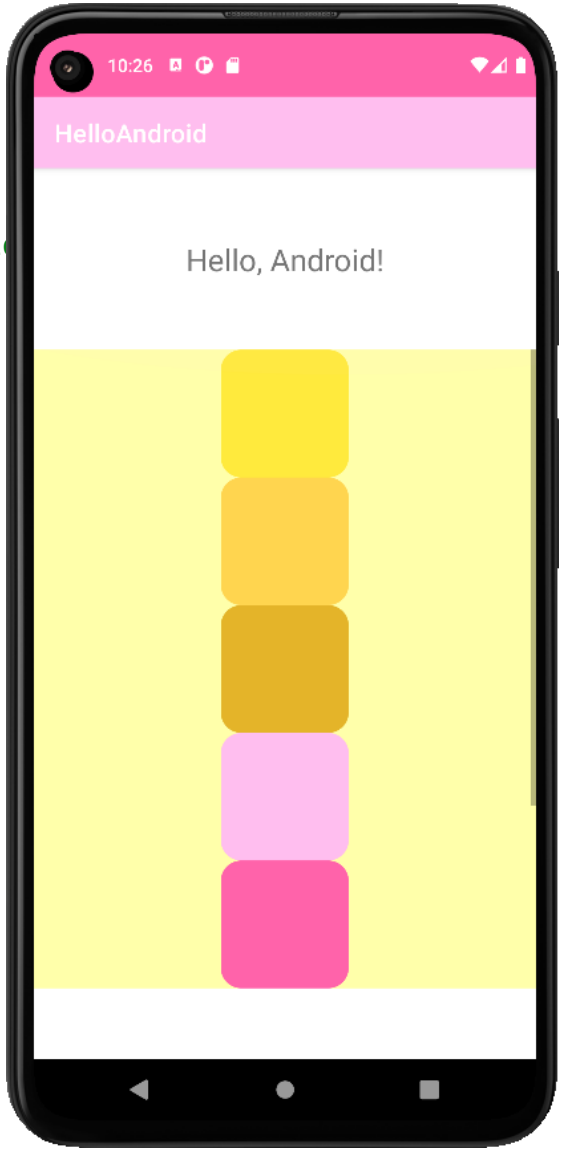
Layouts and Scrolling – Including Layouts

rectangle_scroll_view.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/androi
              xmlns:app="http://schemas.android.com/apk/res-auto"
              android:layout_width="match_parent"
              android:layout_height="wrap_content"
              android:orientation="vertical"
              android:gravity="center"
              android:backgroundTint="@color/uw1">
    <ImageView android:layout_height="100dp“
               android:layout_width="100dp“
               android:src="@drawable/rectangle"
               app:tint="@color/uw2"/>
    ...
</LinearLayout>
```

.xml-file:

```
<ScrollView android:id="@+id/scrollView2"
        android:layout_width="match_parent"
        android:layout_height="500dp"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        app:layout_constraintBottom_toBottomOf="parent"
        tools:layout_editor_absoluteX="0dp">
        <include layout="@layout/rectangle_scroll_view" />
</ScrollView>
```



End of the Chapter

Any further questions?