ITW202: Mobile Application

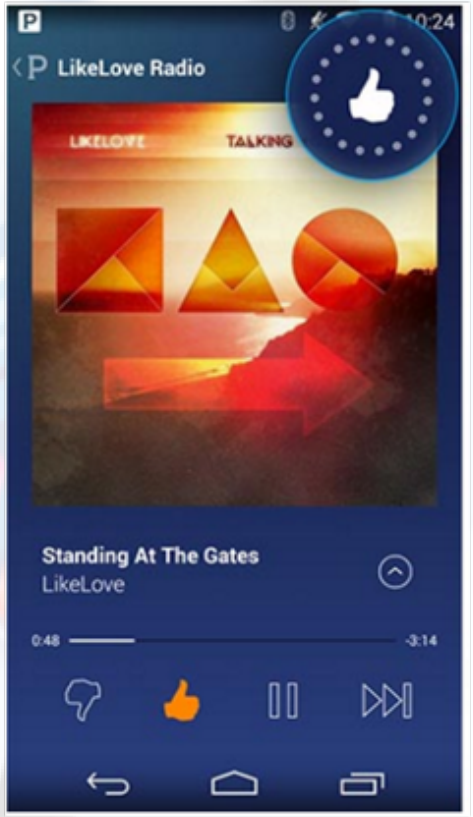
Unit IV: Developing for Android

Ms. Sonam Wangmo

Gyalpozhing College of Information Technology
Royal University of Bhutan

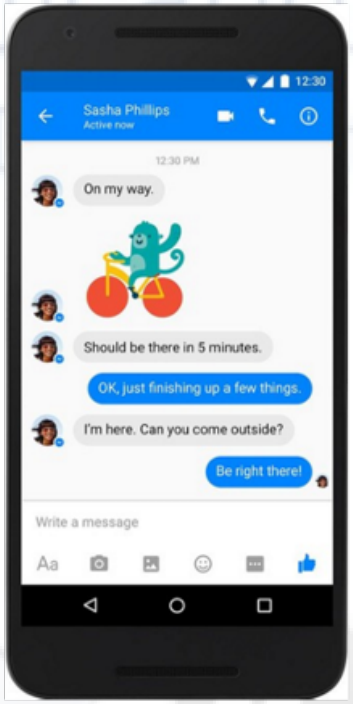
March 1, 2021

Android App Example



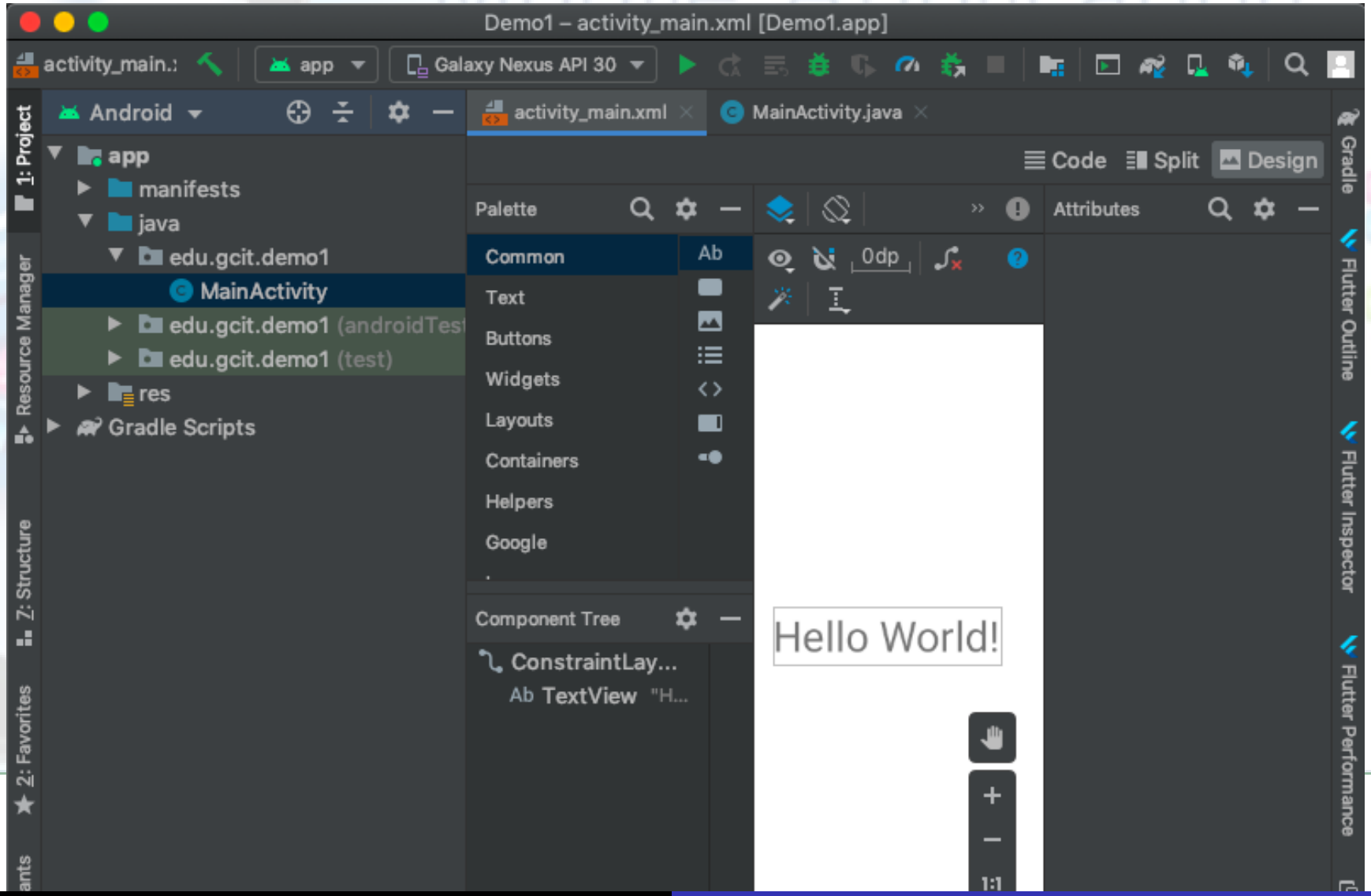
Pandora

Pokemon GO



Facebook Messenger

Android Studio



Android Studio

- Official Android IDE
- Develop, run, debug, test, and package apps
- Monitors and performance tools
- Virtual devices
- Project views
- Visual layout editor

What is an Android App?

- One or more interactive screens
- Written using Java Programming Language and XML
- Uses the Android Software Development Kit (SDK)
- Uses Android libraries and Android Application Framework
- Executed by Android Runtime Virtual machine (ART)

Challenges of Android development

Multiple screen sizes and resolutions

Performance: make your apps responsive and smooth

Security: keep source code and user data safe

Compatibility: run well on older platform versions

Marketing: understand the market and your users

Hint: It doesn't have to be expensive, but it can be.

App Building Blocks

Resources: layouts, images, strings, colors as XML and media files

Components: activities, services, . . . , and helper classes as Java code

Manifest: information about app for the runtime

Build configuration: APK versions in Gradle config files

Component Types

Activity is a single screen with a user interface

Service performs long-running tasks in background

Content provider manages shared set of data

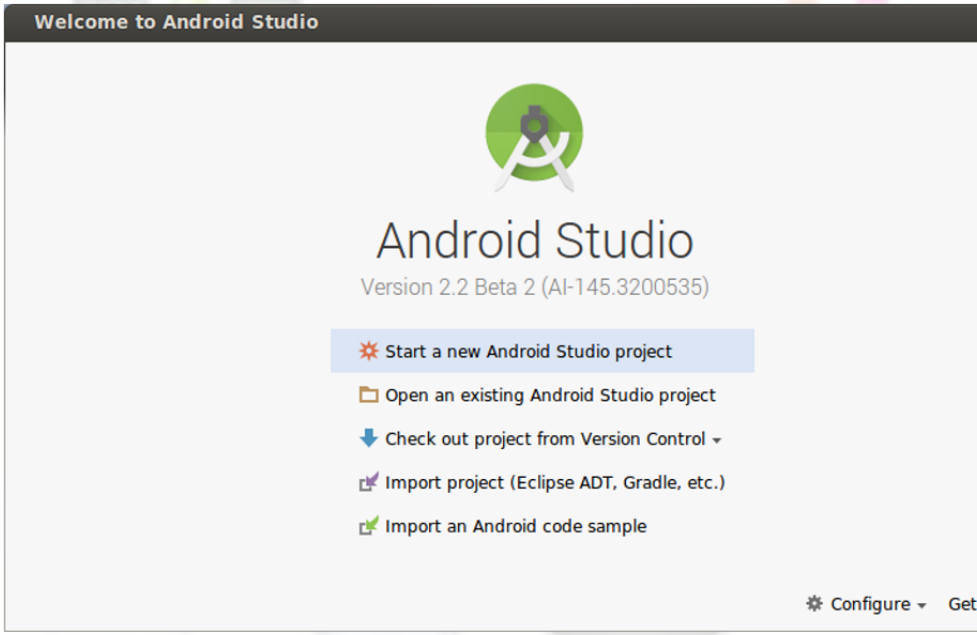
Broadcast receiver responds to system-wide announcements

Think of Android as a hotel

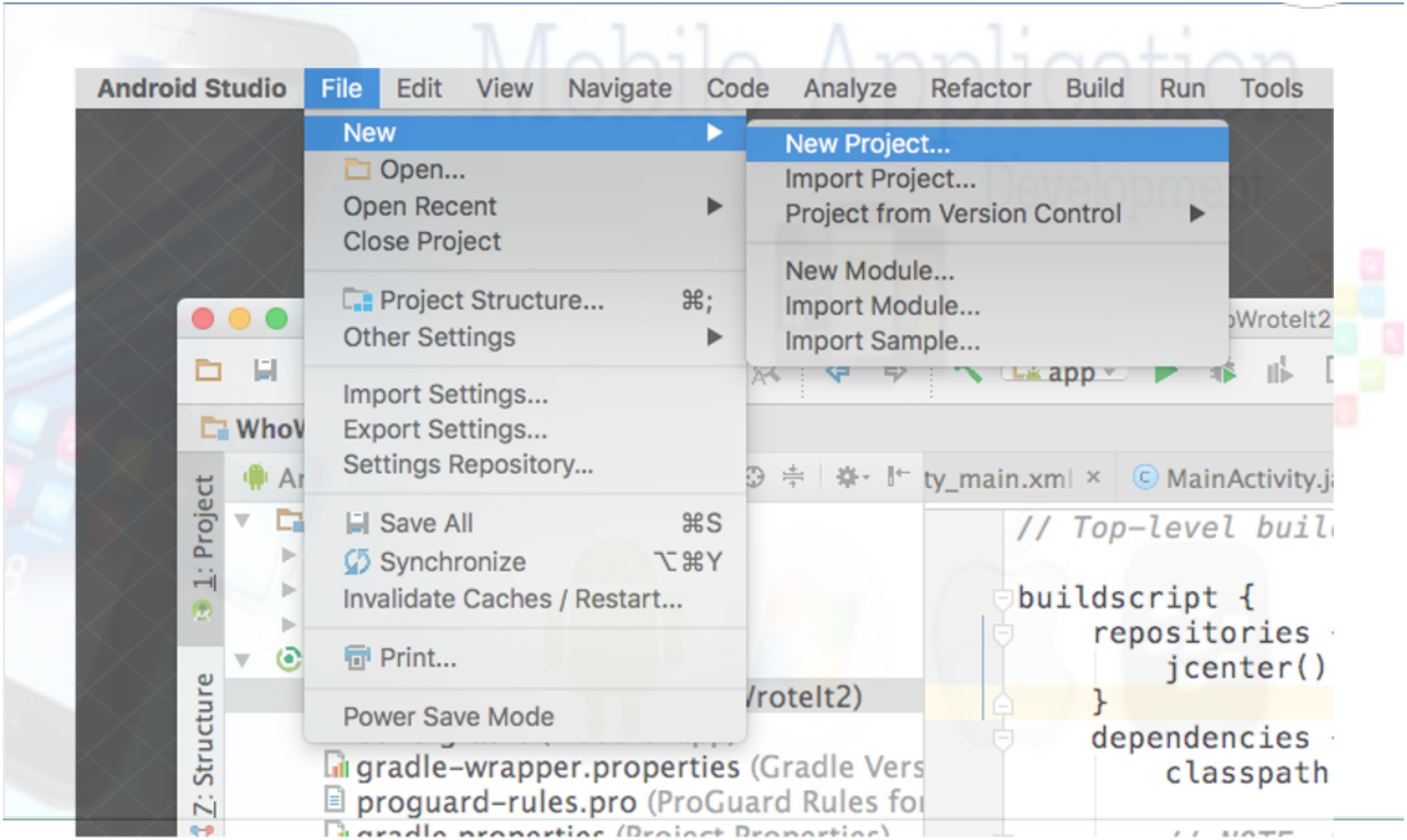
- Your app is the guest
- The Android System is the hotel manager
- Services are available when you request them (intents)
  - In the foreground (activities) such as registration
  - In the background (services) such as laundry
- Calls you when a package has arrived (broadcast receiver)
- Access the city's tour companies (content provider)

Creating your First Android App: Start Android Studio

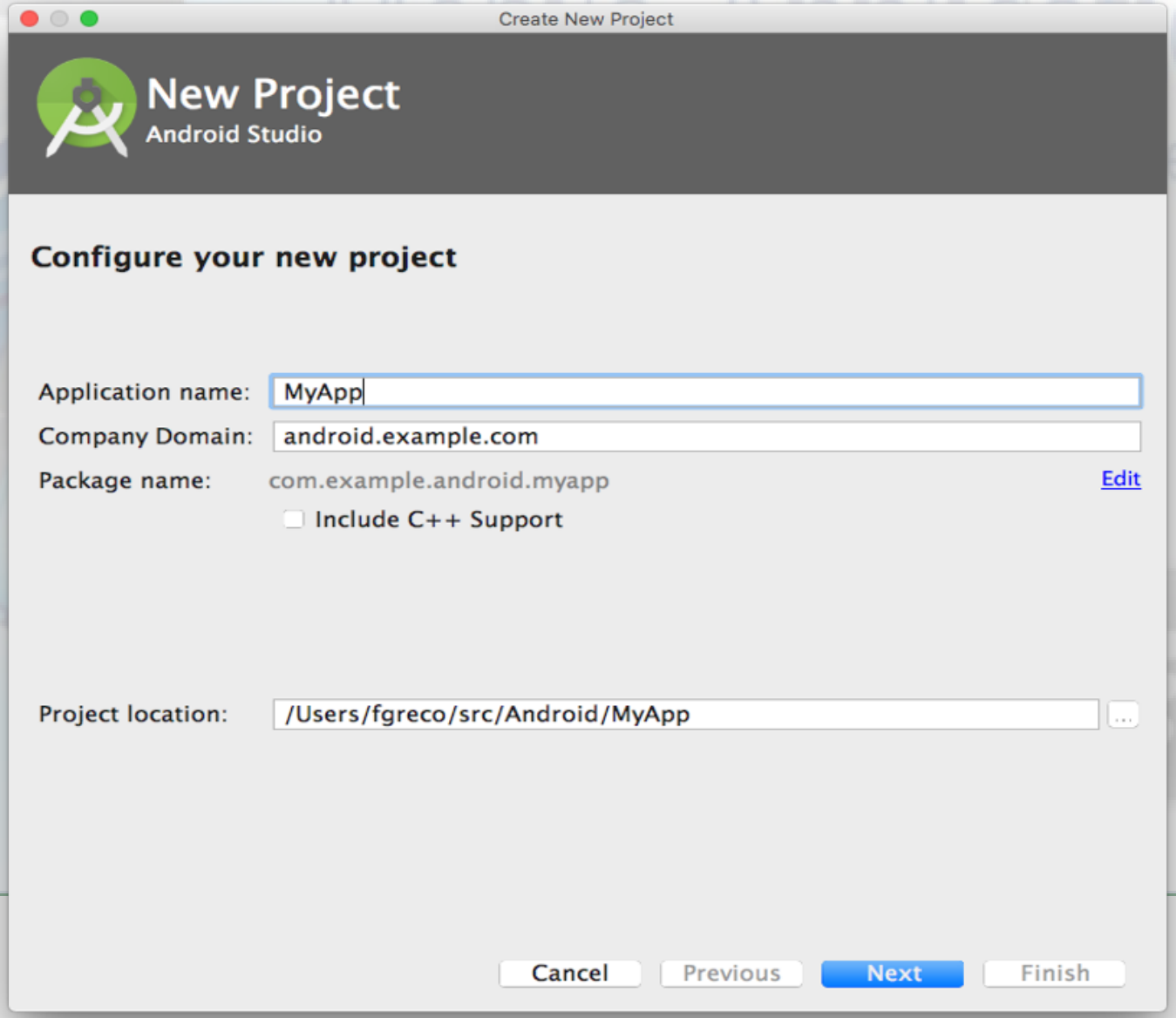
⇓



Create a project inside Android Studio

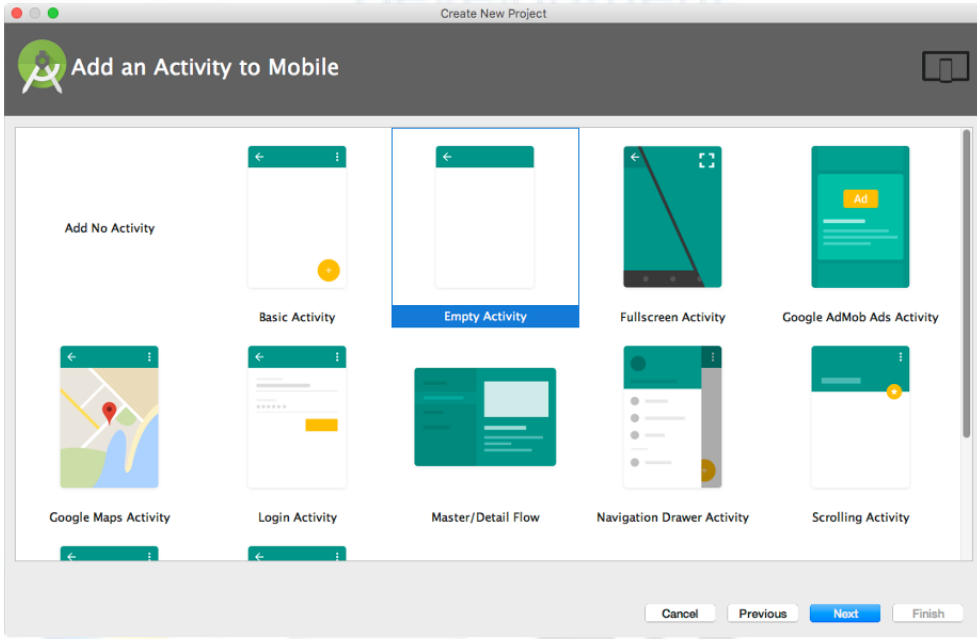


Name your app



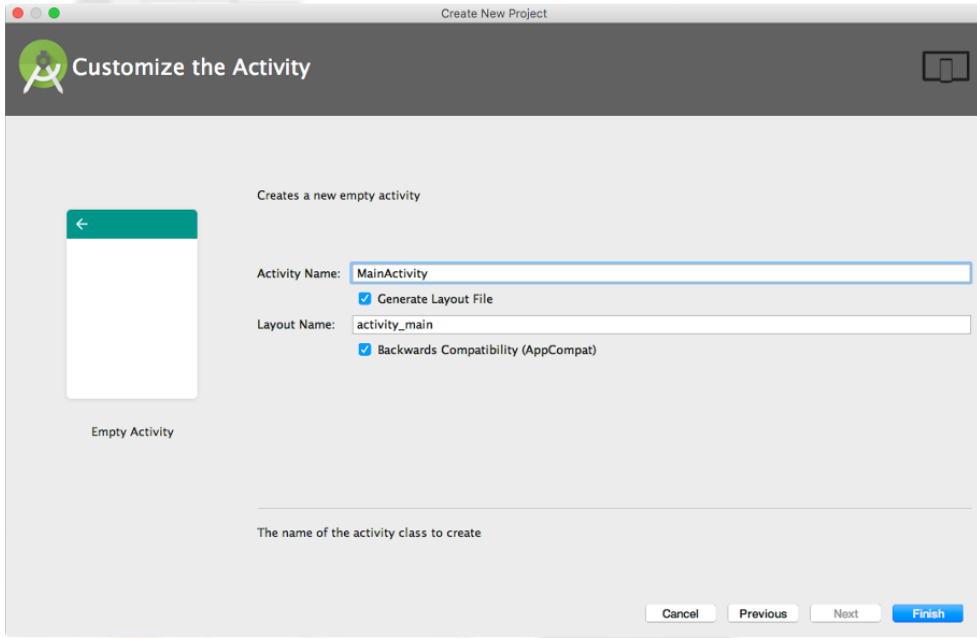
Pick activity template

- Choose templates for common activities, such as maps or navigation drawers.
- Pick Empty Activity or Basic Activity for simple and custom activities.

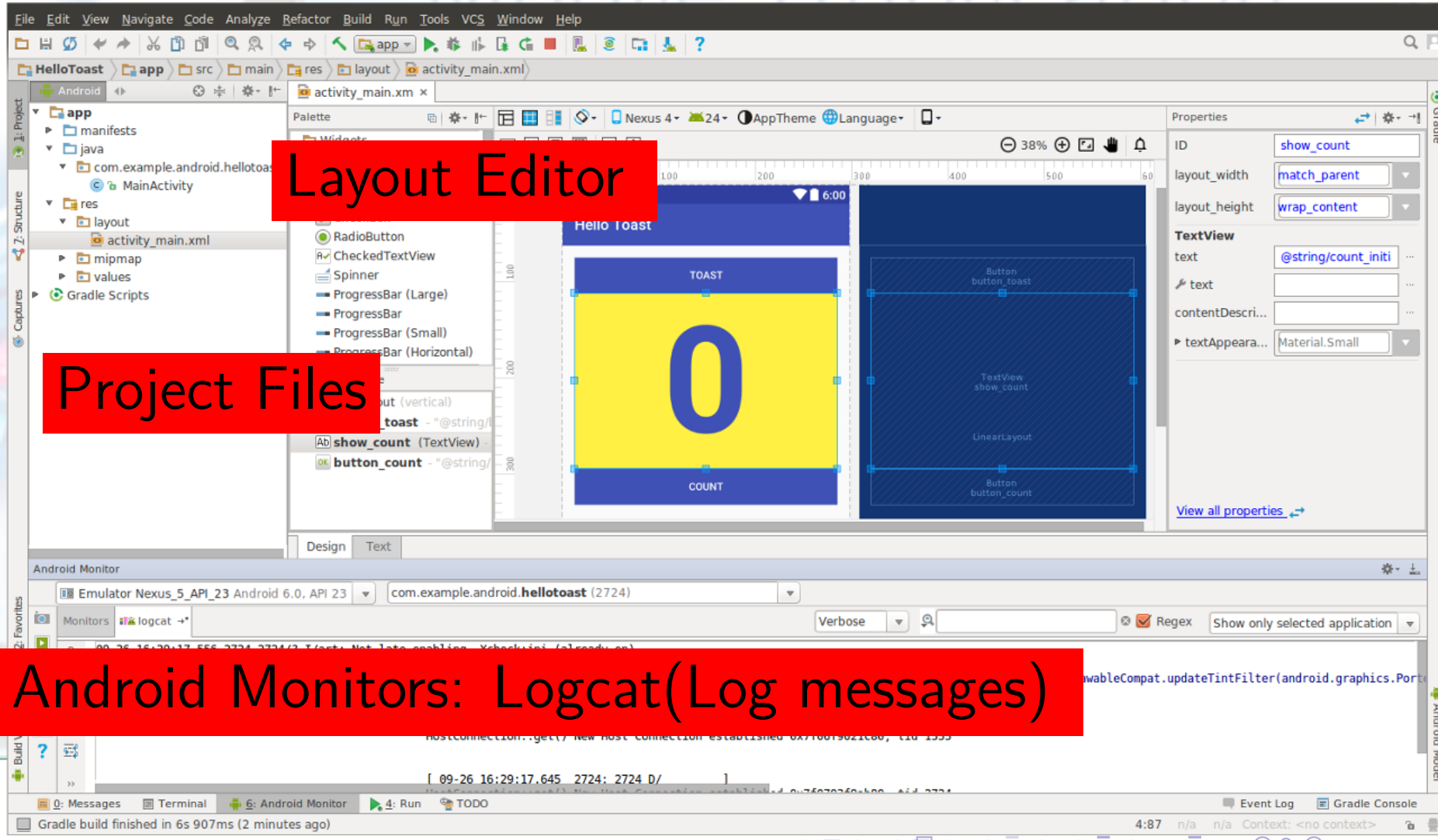


Name your activity

- Good practice to name main activity MainActivity and activity_main layout
- Use AppCompat
- Generating layout file is convenient

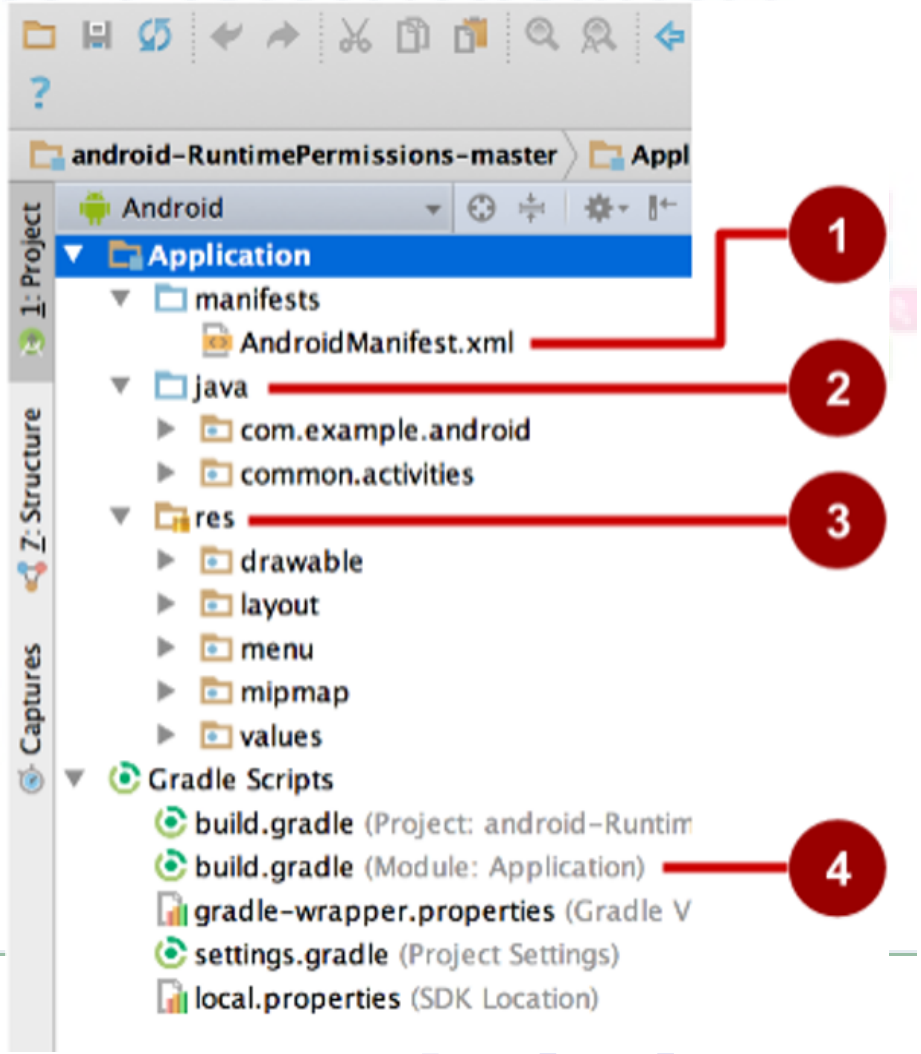


Android Studio Panes



Project Folder

1. Manifests—Android Manifest file - description of app read by the Android runtime
2. Java—Java source code packages
3. res—Resources (XML) - layout, strings, images, dimensions, colors...
4. build.gradle—build files



Gradle build system

Gradle

Is a build automation tool in development process for compilation and packaging to testing, deployment and publishing.

Gradle build system

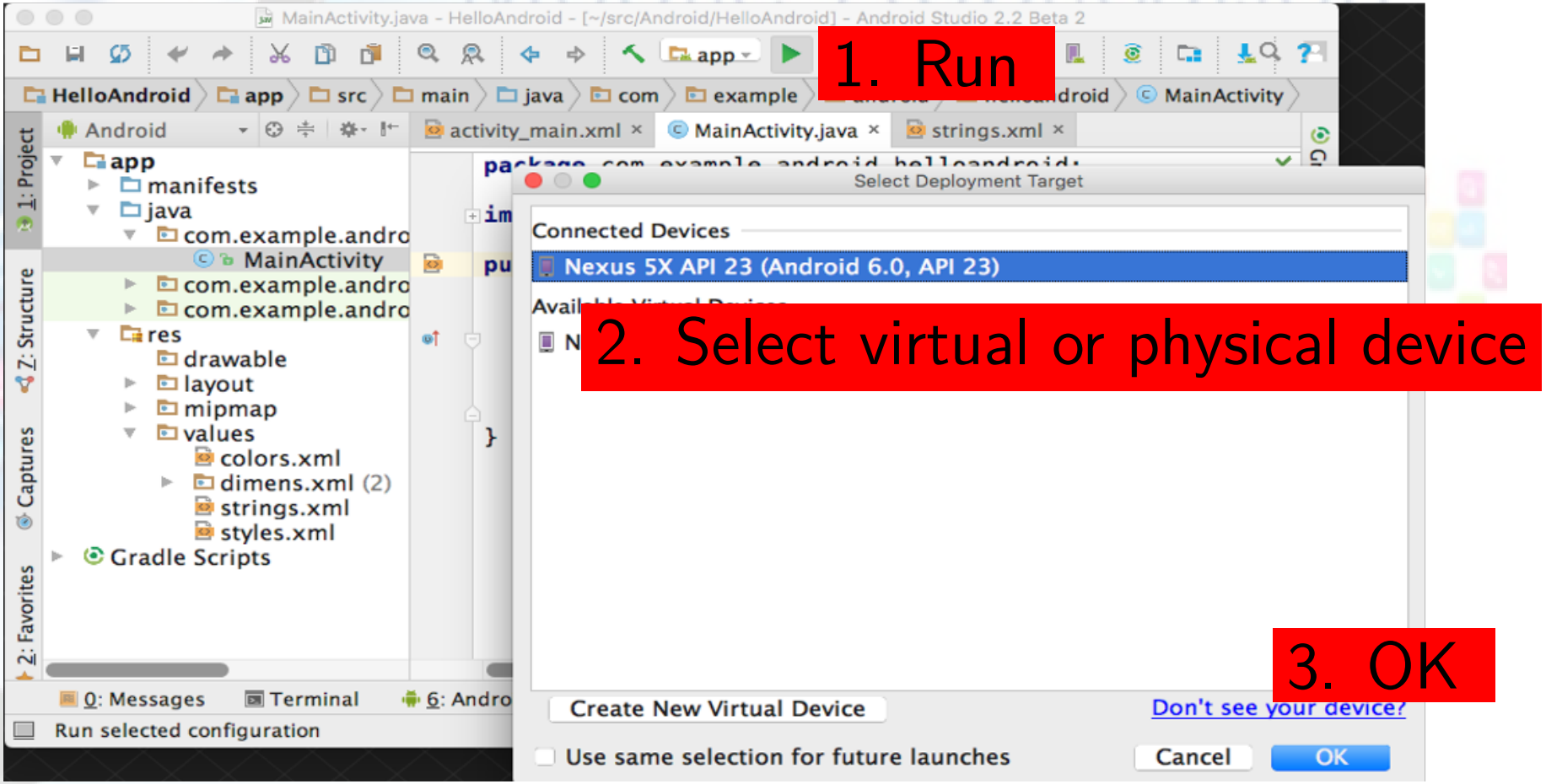
Example: Lets say you want to use youtube player in your application. Here, you just have to add the dependency in build.gradle file. Then gradle will download the code for youtube video player and you can use it in your project.
Thus, gradle is a build tools that will handle the configuration of every file and other codes and generate the apk file.

Gradle build system

apk file

apk Android Package is the package file format used by the Android operating system and a number of other Android-based operating systems for distribution and installation of mobile apps.

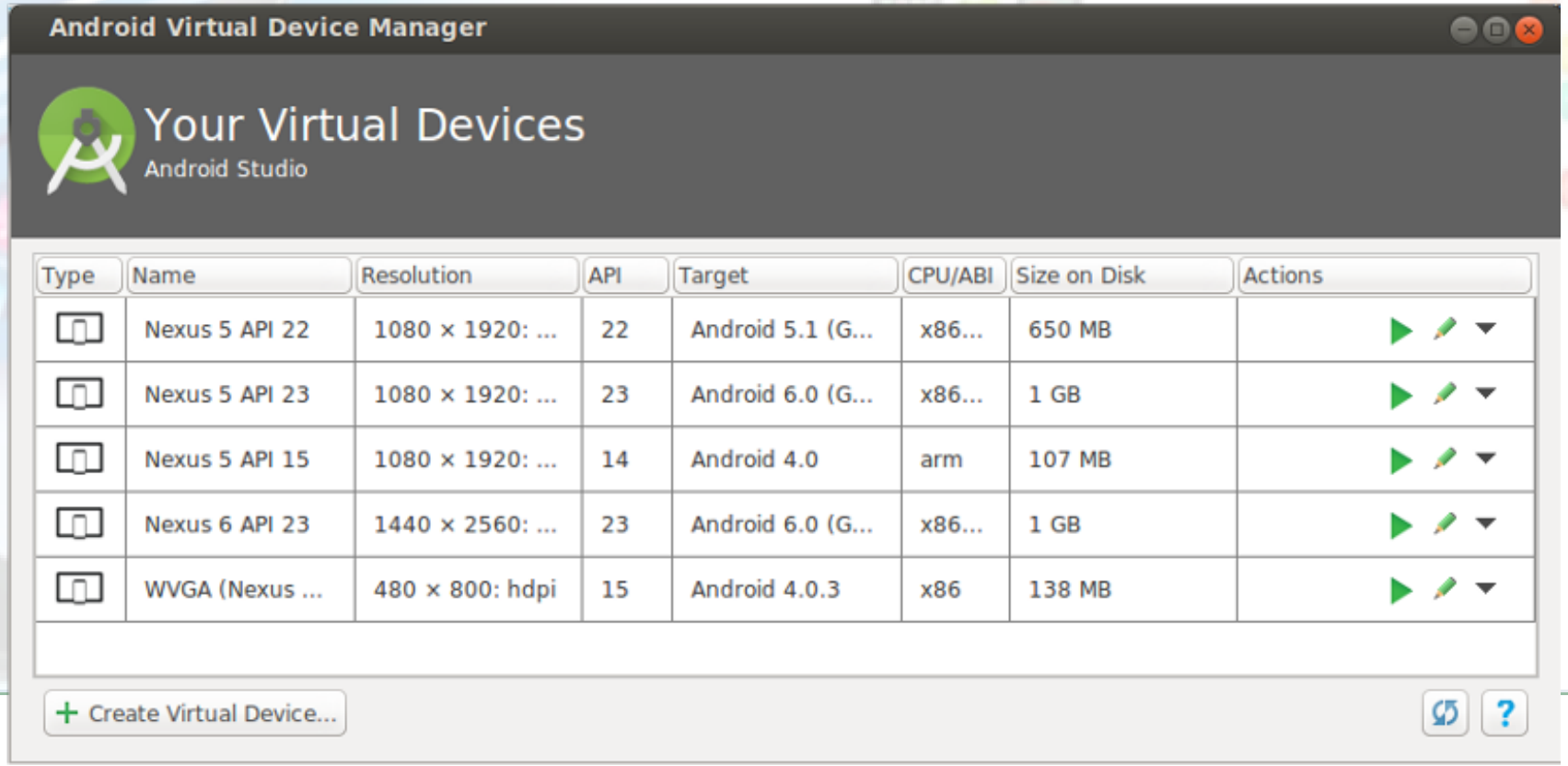
Run your app


MainActivity.java - HelloAndroid - [~/src/Android/HelloAndroid] - Android Studio 2.2 Beta 2
1. Run
HelloAndroid
app
src
main
java
com
example
MainActivity
Android
activity_main.xml
MainActivity.java
strings.xml
manifests
java
com.example.andro
MainActivity
com.example.andro
com.example.andro
res
drawable
layout
mipmap
values
colors.xml
dimens.xml (2)
strings.xml
styles.xml
Gradle Scripts
Select Deployment Target
Connected Devices
Nexus 5X API 23 (Android 6.0, API 23)
2. Select virtual or physical device
3. OK
Create New Virtual Device
Don't see your device?
Use same selection for future launches
Cancel
OK
0: Messages
Terminal
Run selected configuration

Create a virtual device

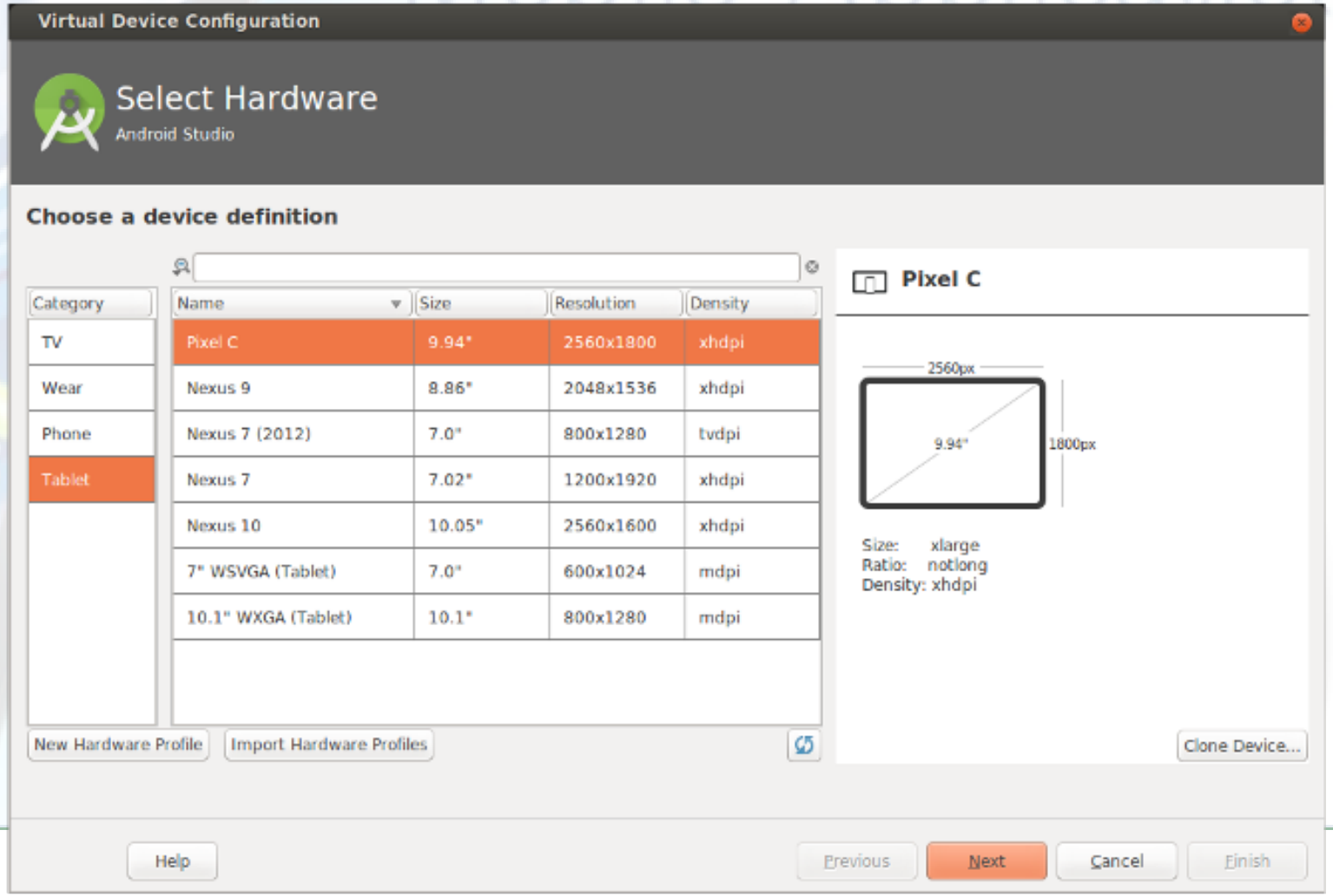
Use emulators to test app on different versions of Android and form factors.

Tools -> Android -> AVD Manager



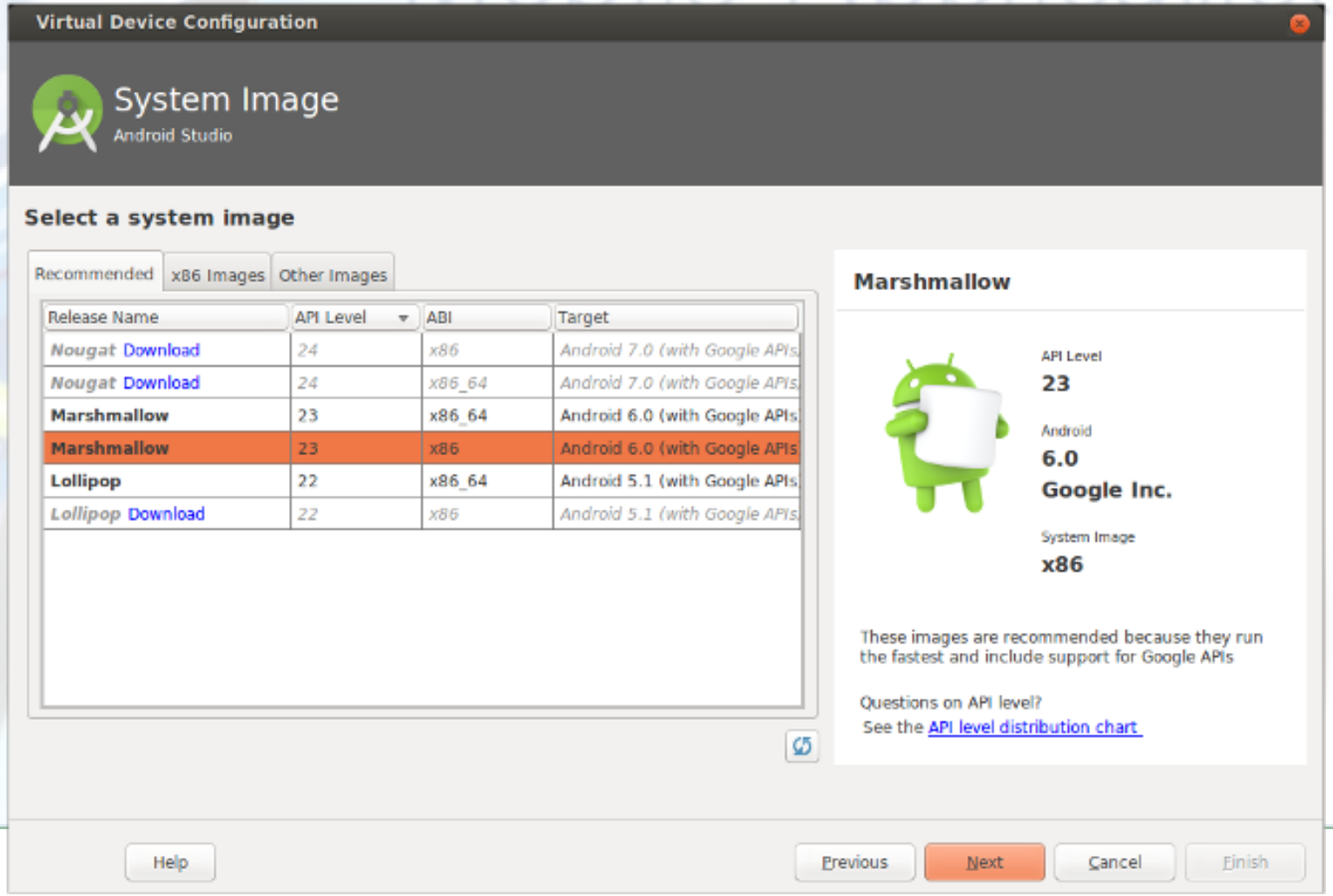
Configure virtual device

1. Choose hardware



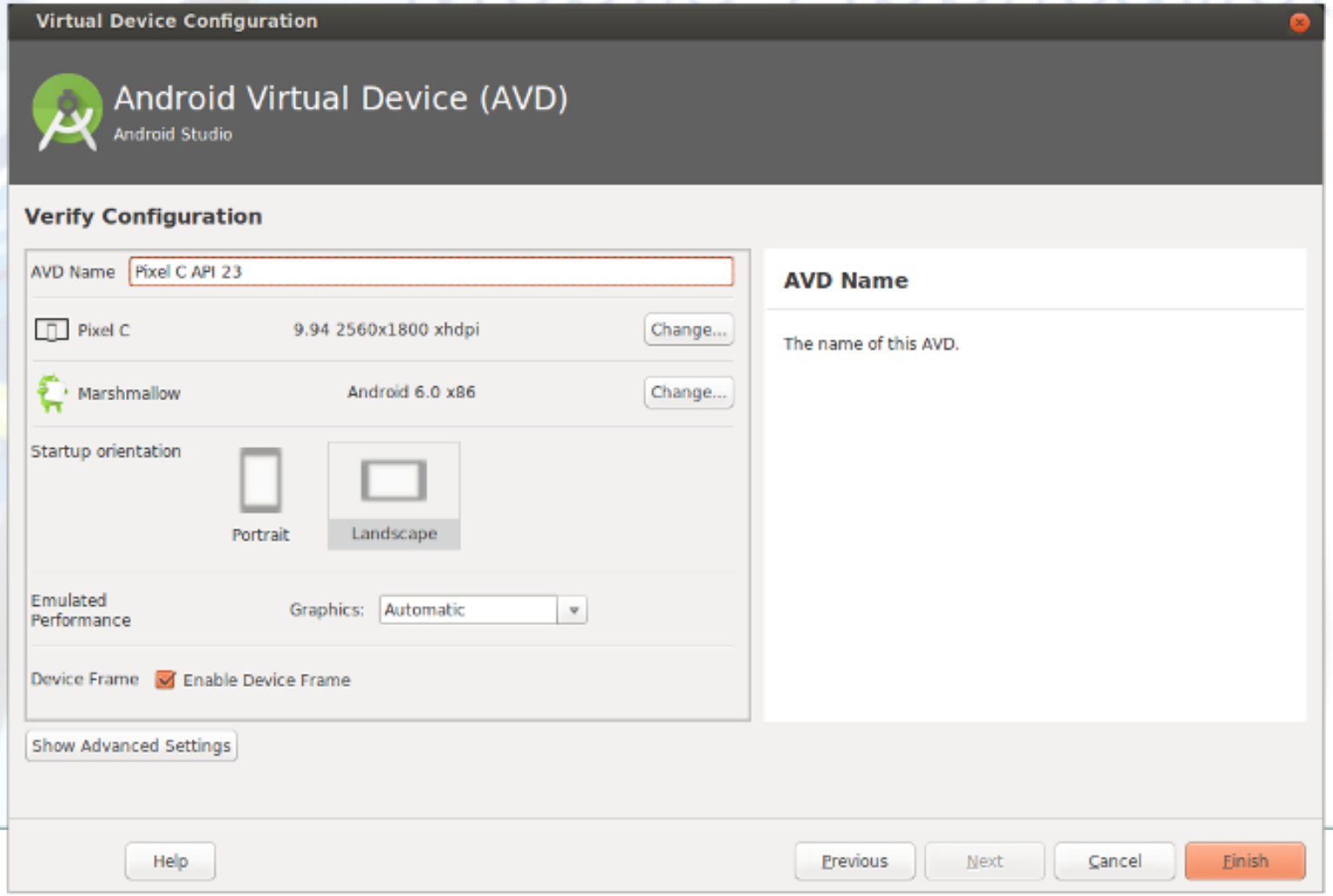
Configure virtual device

2. Select Android Version

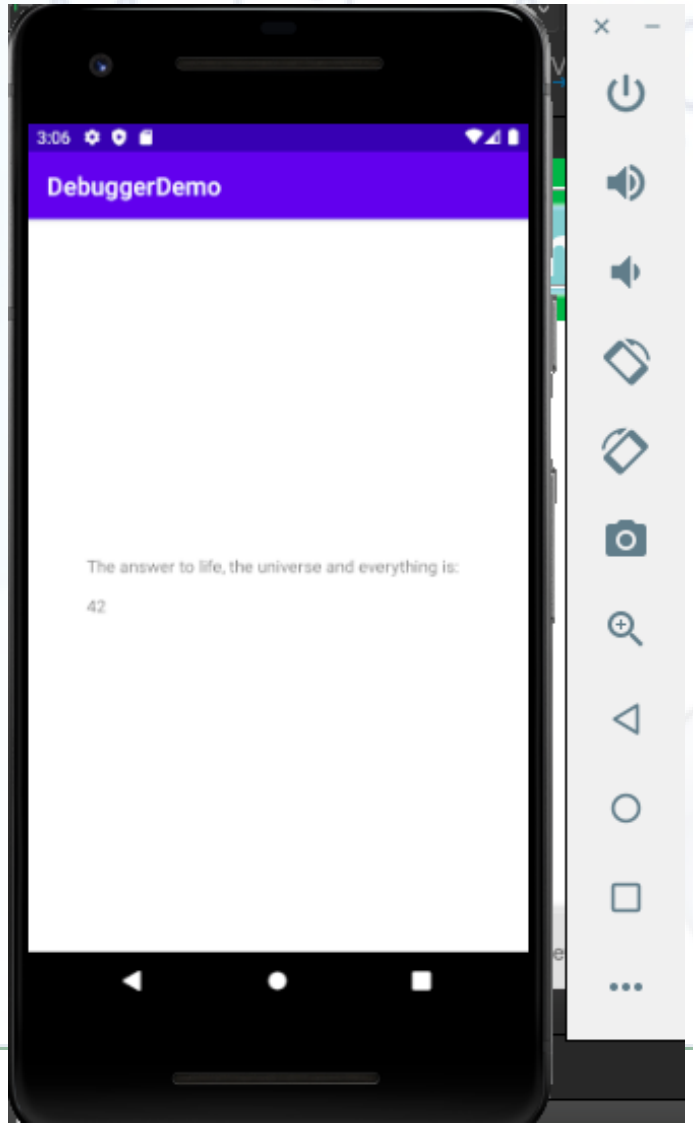


Configure virtual device

3. Finalize



Run on a virtual device

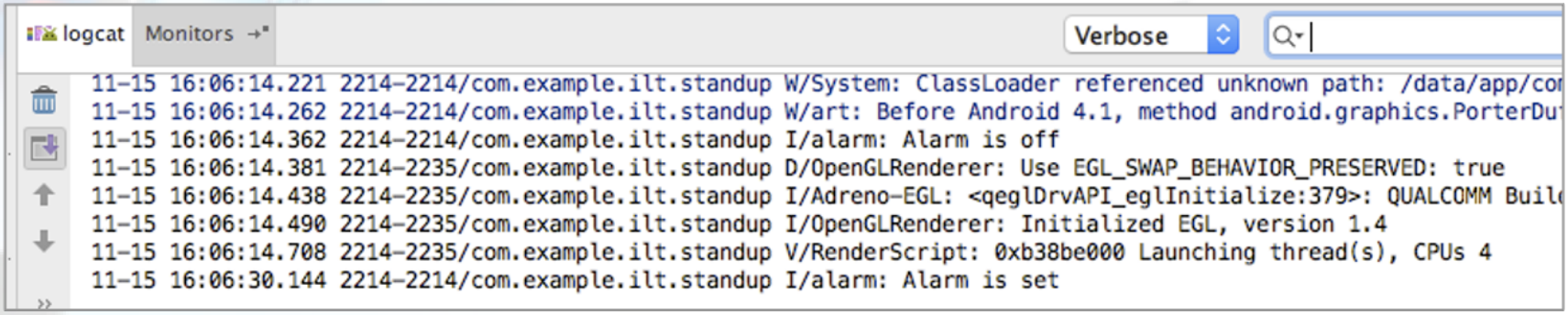


Run on a physical device

1. Turn on Developer Options:
   - Settings -> About phone
   - Tap Build number seven times
2. Turn on USB Debugging
   - Settings -> Developer Options -> USB Debugging
3. Connect phone to computer with cable

Get feedback as your app runs

- As the app runs, Android Monitor logcat shows information
- You can add logging statements to your app that will show up in logcat.



Logging

```
import android.util.Log;

// Use class name as tag
private static final String TAG =
    MainActivity.class.getSimpleName();

// Show message in Android Monitor, logcat pane
// Log.<log-level>(TAG, "Message");
Log.d(TAG, “Creating the URI…”);
```

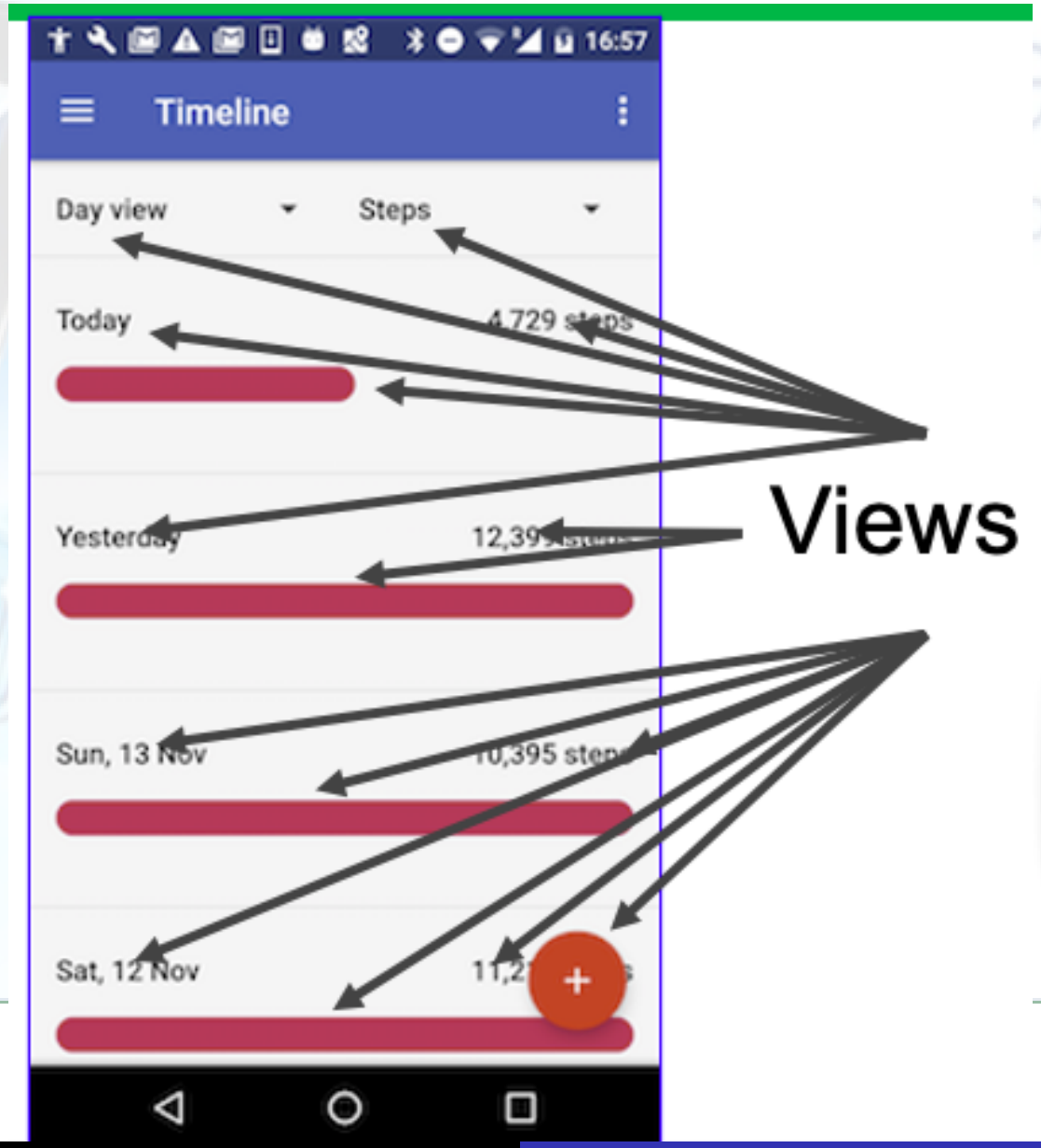
Views, Layouts, and Resources

Views

Everything you see is a view

If you look at your mobile device, every user interface element that you see is a View.

Everything you see is a view



What is a View

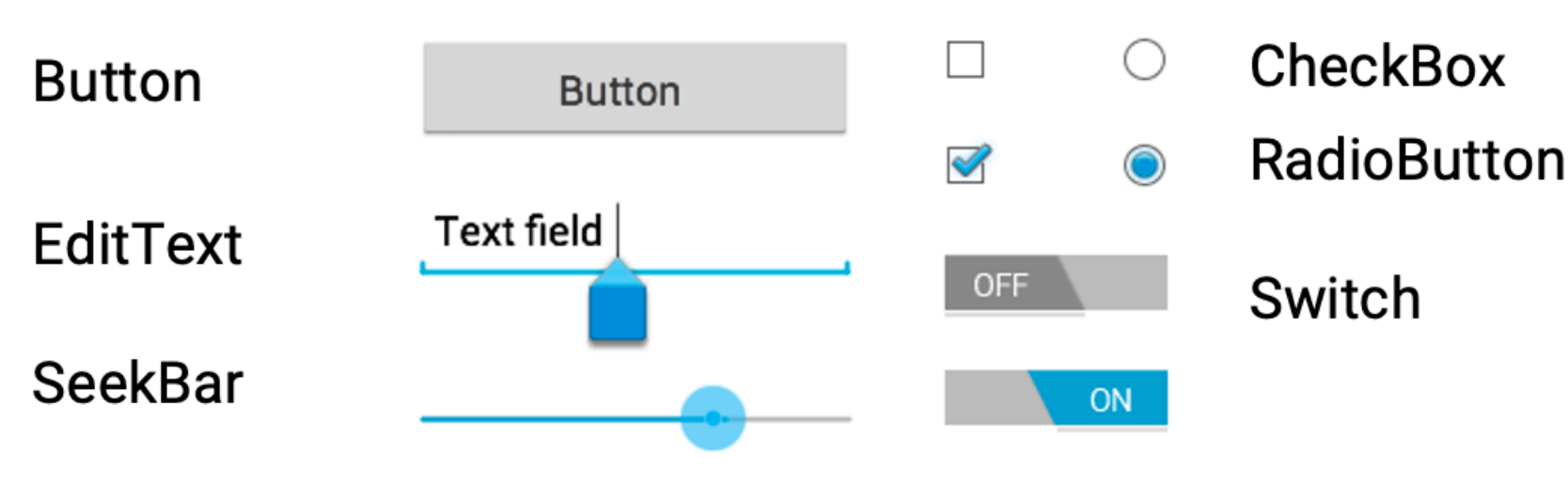
Views are Android's basic user interface building blocks.

- display text (TextView class), edit text (EditText class)
- buttons (Button class), menus, other controls
- scrollable (ScrollView, RecyclerView)
- show images (ImageView)
- subclass of View class

Views have properties

- Have properties (e.g., color, dimensions, positioning)
- May have focus (e.g., selected to receive user input)
- May be interactive (respond to user clicks)
- May be visible or not
- Have relationships to other views

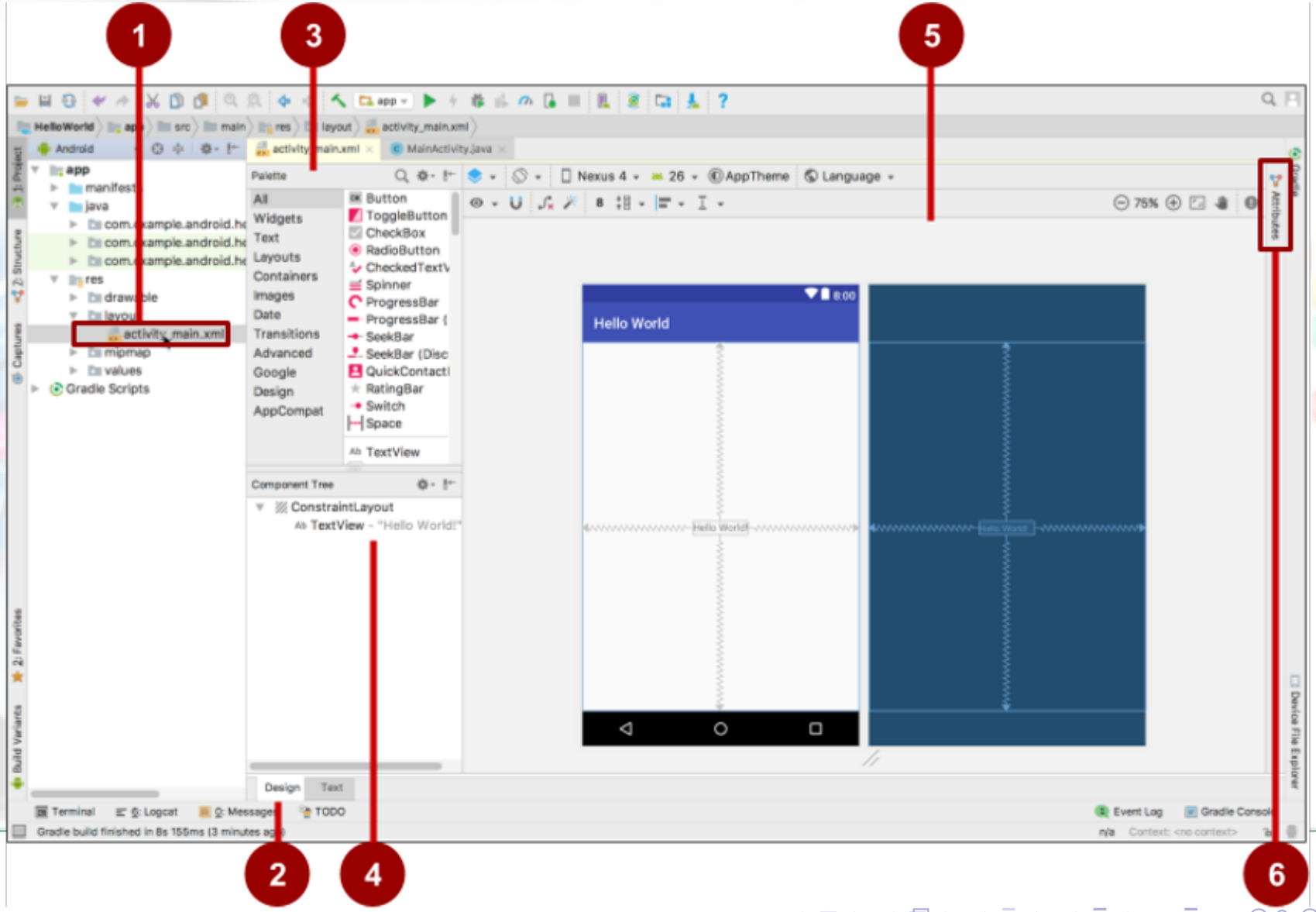
Examples of views



Create views and layouts

- Android Studio layout editor: visual representation of XML
- XML editor
- Java code

Android Studio layout editor



Android Studio layout editor

1. XML layout file
2. Design and Text tabs
3. Palette pane
4. Component Tree
5. Design and blueprint panes
6. Attributes tab

View defined in XML

```
<TextView
android:id="@+id/show_count"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="@color/myBackgroundColor"
android:text="@string/count_initial_value"
android:textColor="@color/colorPrimary"
android:textSize="@dimen/count_text_size"
android:textStyle="bold"
/>
```

View attributes in XML

android:<property_name>="<property_value>"
Example: android:layout_width="match_parent"
android:<property_name>=
"@<resource_type>/resource_id"
Example:
android:text="@string/button_label_next"
android:<property_name>="@+id/view_id"
Example: android:id="@+id/show_count"

Create View in Java code

In an Activity:

context

```
TextView myText = new TextView(this);
myText.setText("Display this text!");
```

What is the context?

Context is an interface to global information about an application environment

Views

Over 100 (!) different types of views available from the Android system, all children of the View class.

ViewGroup and View hierarchy

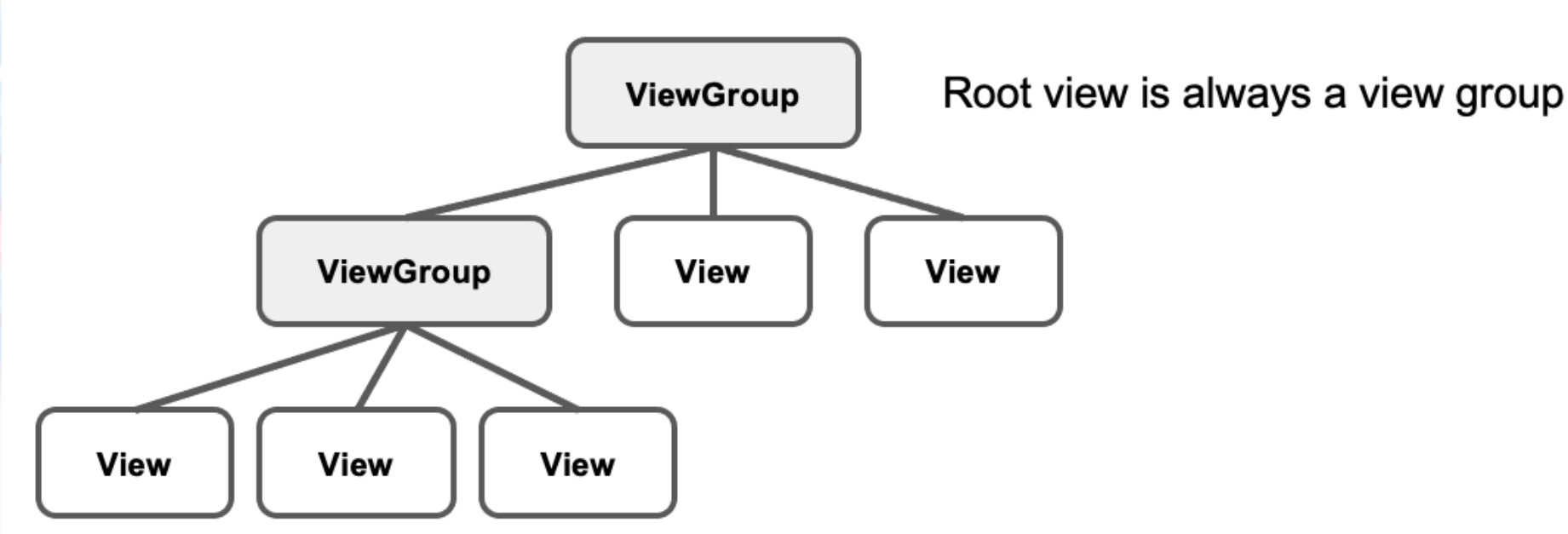
ViewGroup views

A ViewGroup (parent) is a type of view that can contain other views (children)
ViewGroup is the base class for layouts and view containers

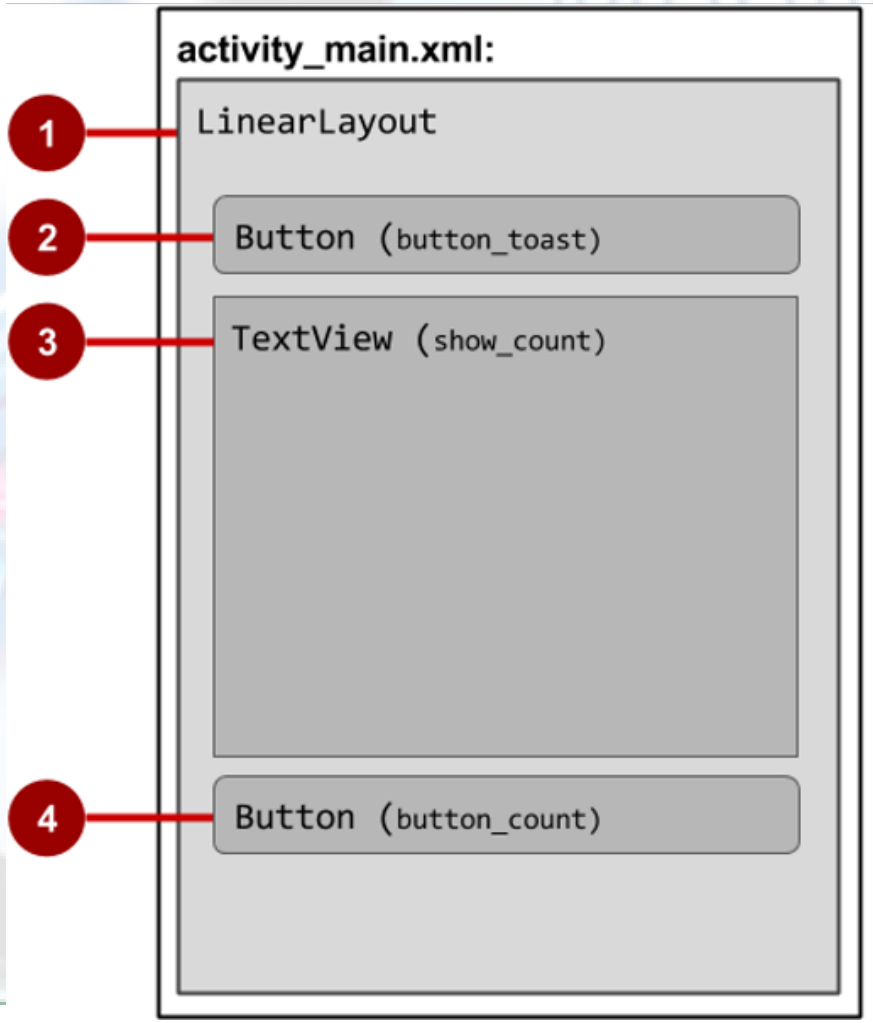
ViewGroup views

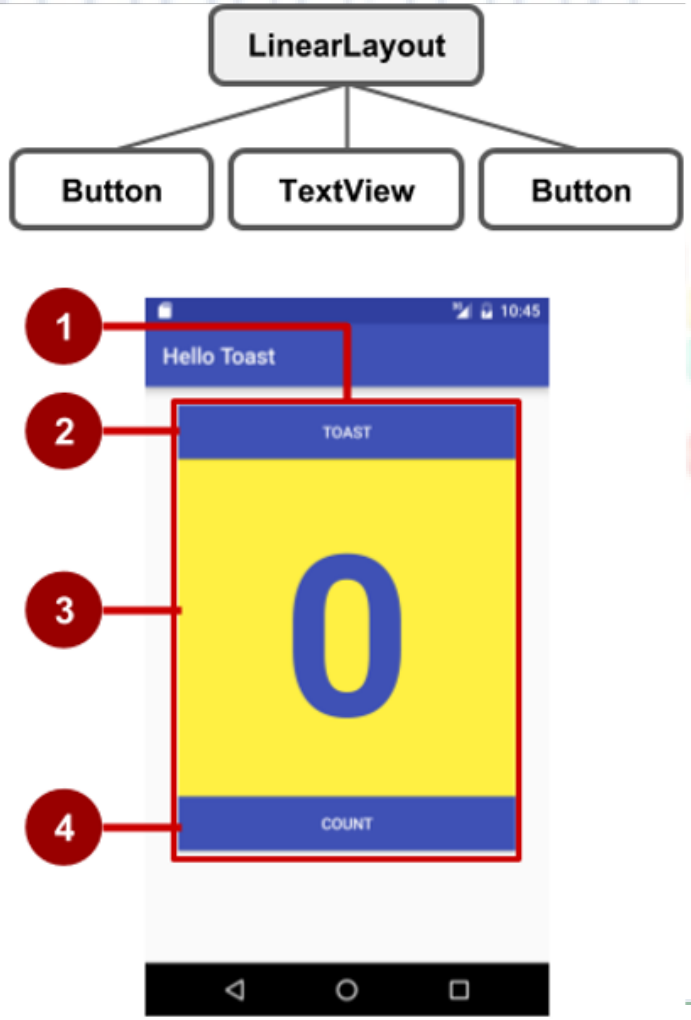
- ScrollView—scrollable view that contains one child view
- LinearLayout—arrange views in horizontal/vertical row
- RecyclerView—scrollable "list" of views or view groups

Hierarchy of view groups and views

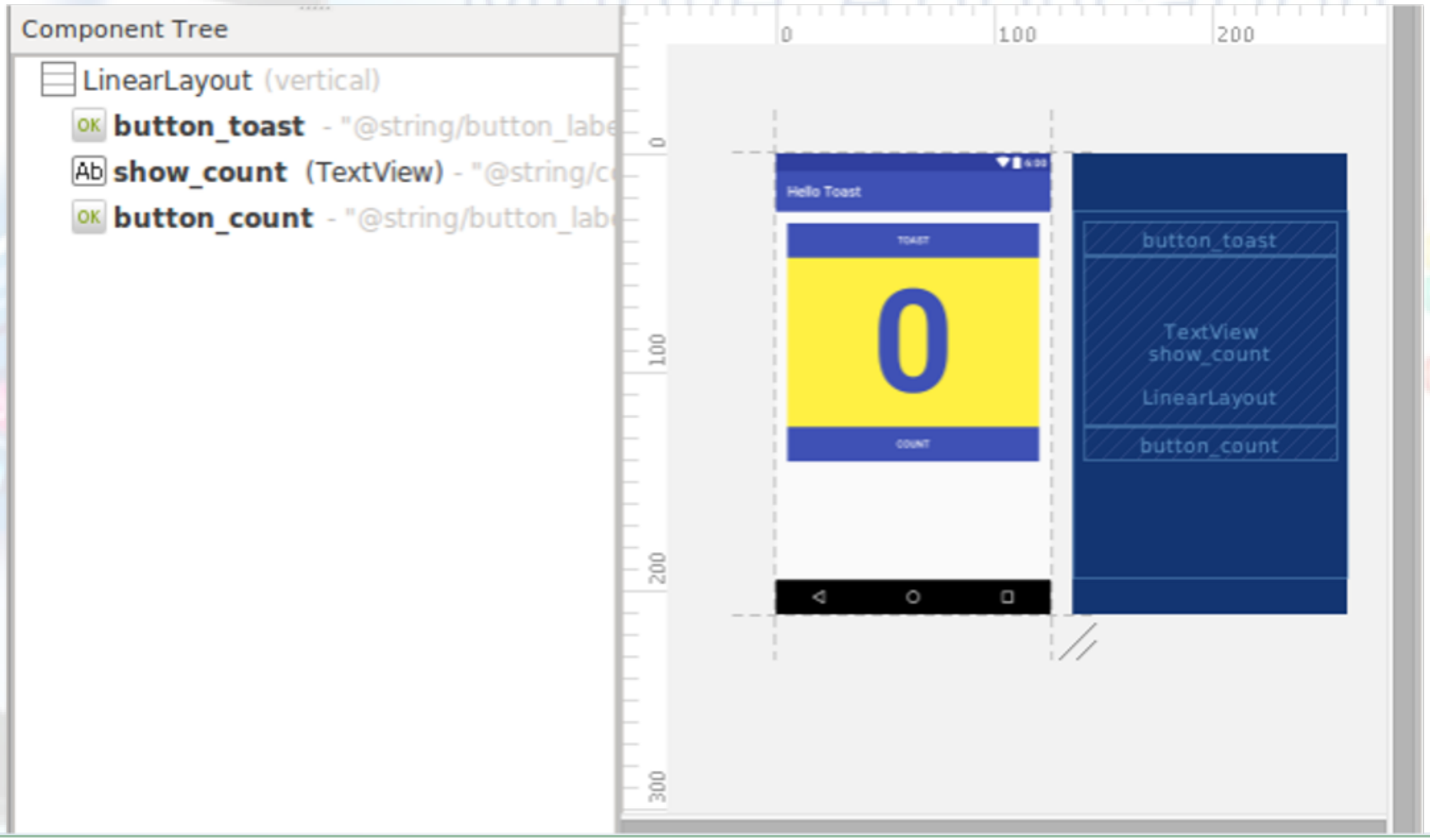


View hierarchy and screen layout





View hierarchy in the component tree



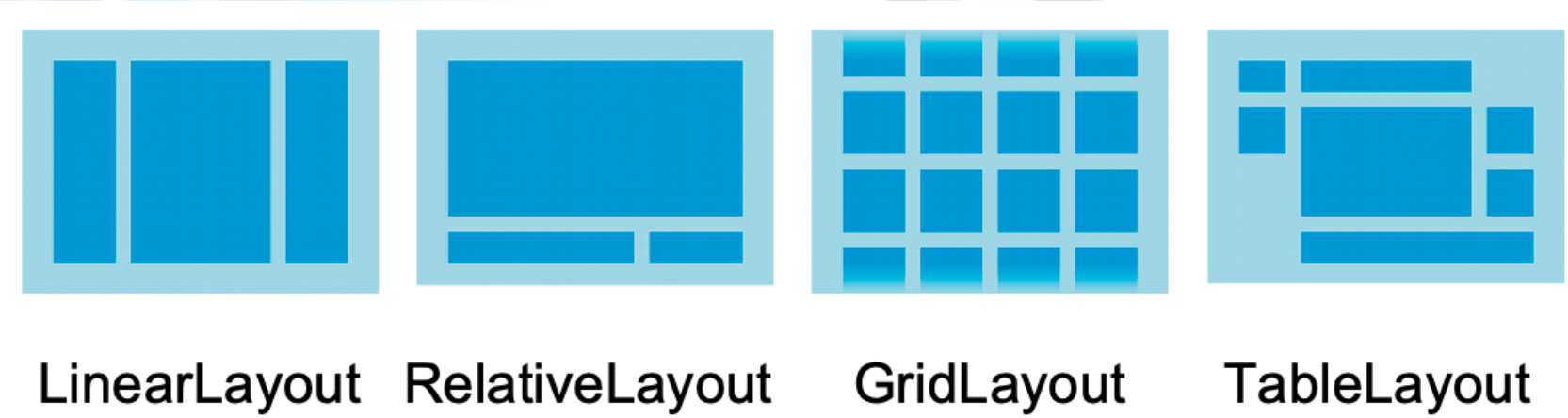
Layouts

Layout Views

Layouts

- are specific types of view groups
- are subclasses of ViewGroup
- contain child views
- can be in a row, column, grid, table, absolute

Common Layout Classes



Common Layout Classes

- ConstraintLayout - connect views with constraints
- LinearLayout - horizontal or vertical row
- RelativeLayout - child views relative to each other
- TableLayout - rows and columns
- FrameLayout - shows one child of a stack of children
- GridView - 2D scrollable grid

Class Hierarchy vs. Layout Hierarchy

- View class-hierarchy is standard object-oriented class inheritance
  - For example, Button is-a TextView is-a View is-a Object
  - Superclass-subclass relationship
- Layout hierarchy is how Views are visually arranged
  - For example, LinearLayout can contain Buttons arranged in a row
  - Parent-child relationship

Layout created in XML

```
<LinearLayout
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
    <EditText
        ... />
    <Button
        ... />
</LinearLayout
```

Layout created in Java Activity code

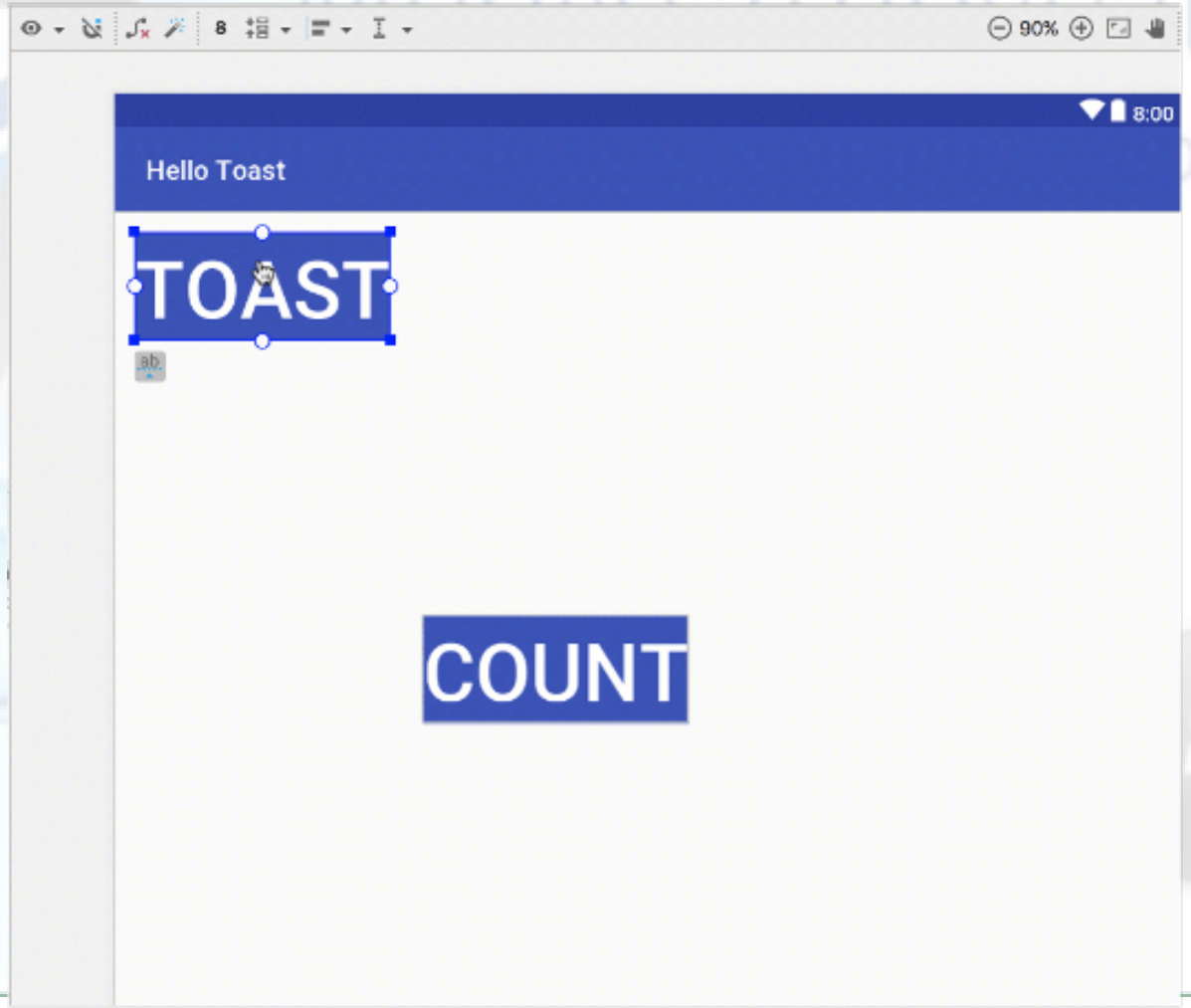
```
LinearLayout linearL = new LinearLayout(this);
linearL.setOrientation(LinearLayout.VERTICAL);

TextView myText = new TextView(this);
myText.setText("Display this text!");

linearL.addView(myText);
setContentView(linearL);
```

The layout editor and ConstraintLayout

The layout editor with ConstraintLayout

- Connect UI elements to parent layout
- Resize and position elements
- Align elements to others
- Adjust margins and dimensions
- Change attributes

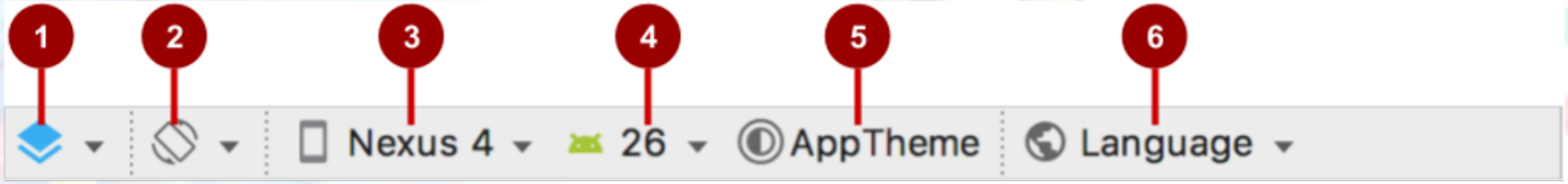
The layout editor with ConstraintLayout



What is ConstraintLayout?

- Default layout for new Android Studio project
- ViewGroup that offers flexibility for layout design
- Provides constraints to determine positions and alignment of UI elements
- Constraint is a connection to another view, parent layout, or invisible guideline

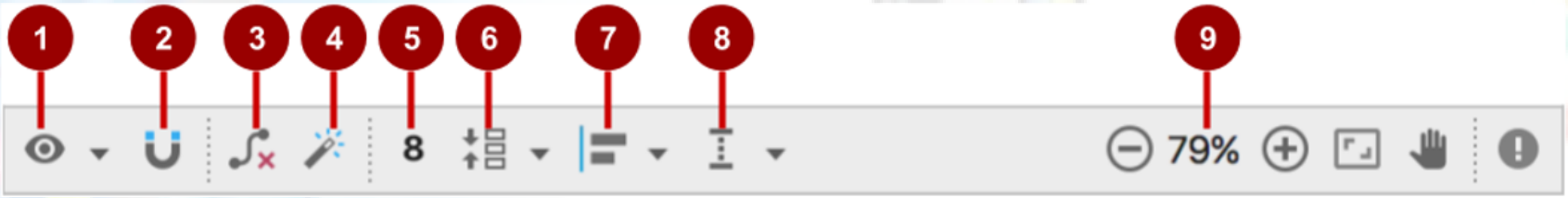
Layout editor main toolbar



Layout editor main toolbar

1. Select Design Surface: Design and Blueprint panes
2. Orientation in Editor: Portrait and Landscape
3. Device in Editor: Choose device for preview
4. API Version in Editor: Choose API for preview
5. Theme in Editor: Choose theme for preview
6. Locale in Editor: Choose language/locale for preview

ConstraintLayout toolbar in layout editor



ConstraintLayout toolbar in layout editor

1. Show: Show Constraints and Show Margins
2. Autoconnect: Enable or disable
3. Clear All Constraints: Clear all constraints in layout
4. Infer Constraints: Create constraints by inference
5. Default Margins: Set default margins
6. Pack: Pack or expand selected elements
7. Align: Align selected elements
8. Guidelines: Add vertical or horizontal guidelines
9. Zoom controls: Zoom in or out

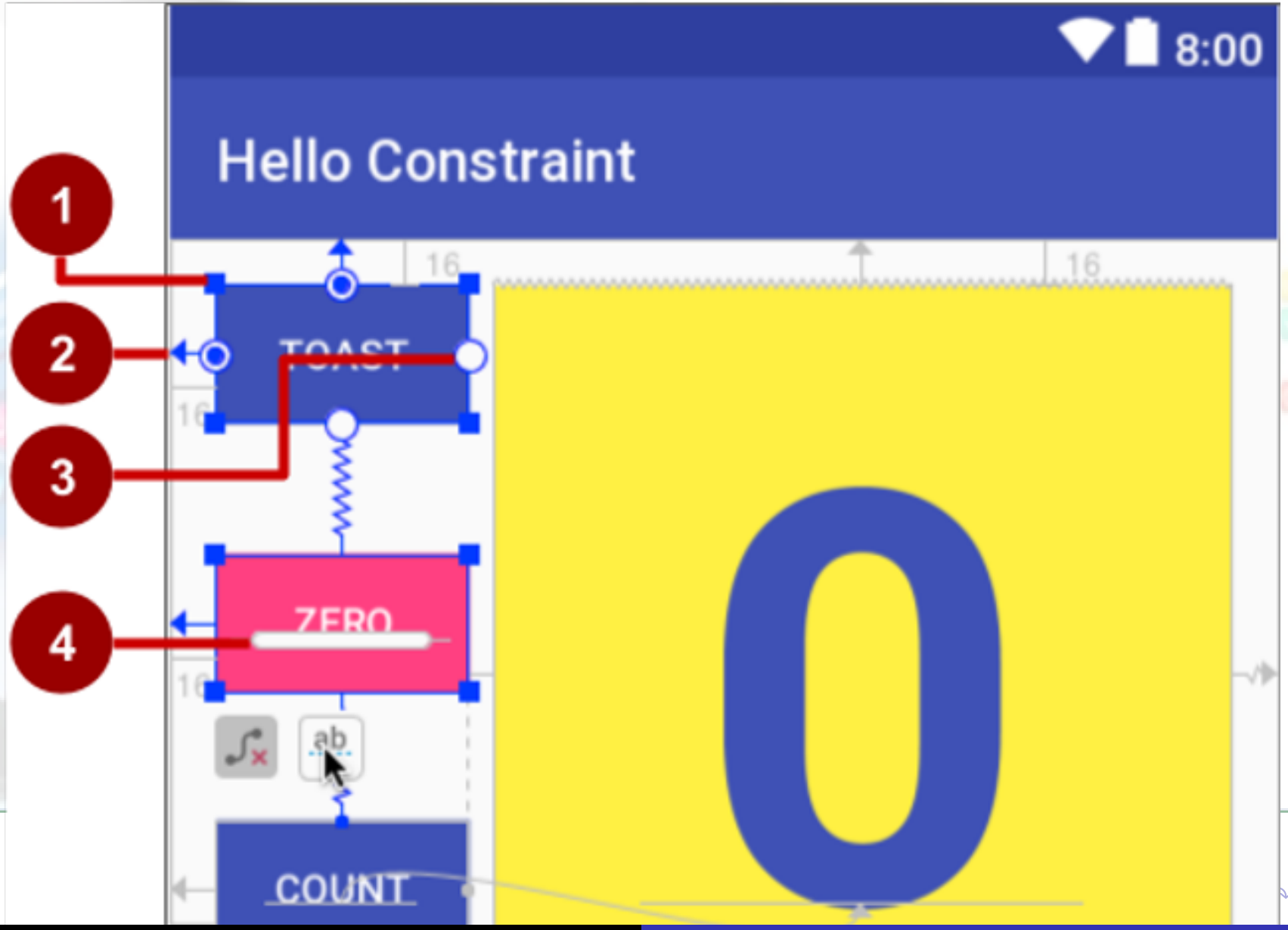
Autoconnect

- Enable Autoconnect in toolbar if disabled
- Drag element to any part of a layout
- Autoconnect generates constraints against parent layout

ConstraintLayout handles

1. Resizing handle
2. Constraint line and handle
3. Constraint handle
4. Baseline handle

ConstraintLayout handles



Event Handling

Events

Something that happens

- In UI: Click, tap, drag
- Events are "noticed" by the Android system

Event Handlers

Methods that do something in response to a click

- A method, called an event handler, is triggered by a specific event and does something in response to the event.

Handling clicks in XML & Java

Attach handler to view in layout:

```
android:onClick="showToast"
```

Implement handler in activity:

```
public void showToast(View view) {
  String msg = "Hello Toast!";
  Toast toast = Toast.makeText(
        this, msg, duration);
  toast.show();
  }
}
```

Setting click handlers in Java

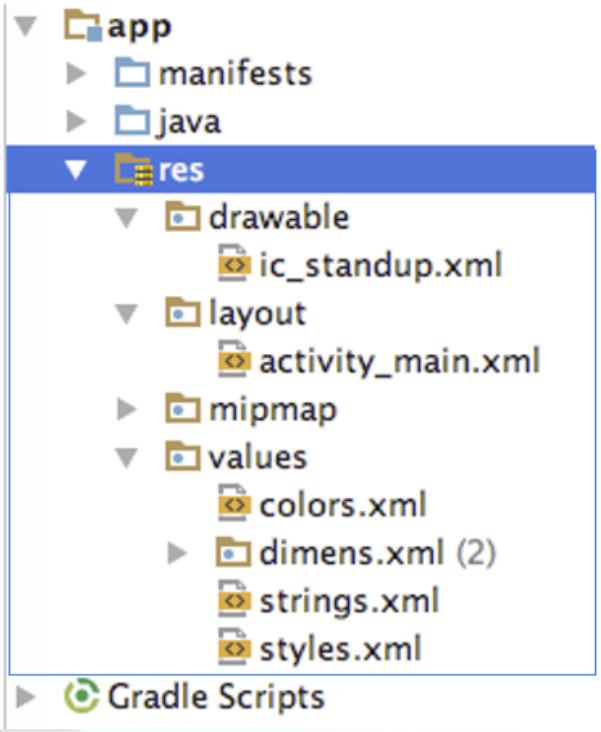
```
final Button button = (Button) findViewById(R.id.button_id);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        String msg = "Hello Toast!";
        Toast toast = Toast.makeText(this, msg, duration);
        toast.show();
        }
     });
```

Resources

Resources

- Separate static data from code in your layouts.
- Strings, dimensions, images, menu text, colors, styles

Where are the resources in your project?



resources and resource files stored in **res** folder

Refer to resources in code

- Layout:

  ```
  R.layout.activity_main
  setContentView(R.layout.activity_main);
  ```

- View:

  ```
  R.id.recyclerview
  rv = (RecyclerView) findViewById(R.id.recyclerview);
  ```

- String:

  In Java: `R.string.title`

  In XML: `android:text="@string/title"`

Measurements

- Device Independent Pixels (dp) - for Views
- Scale Independent Pixels (sp) - for text

Don't use device-dependent units:

- ~~Actual Pixels (px)~~
- ~~Actual Measurement (in, mm)~~
- ~~Points - typography 1/72 inch (pt)~~

THANK YOU