Lab 1 – Introduction to Android & Hello World Example

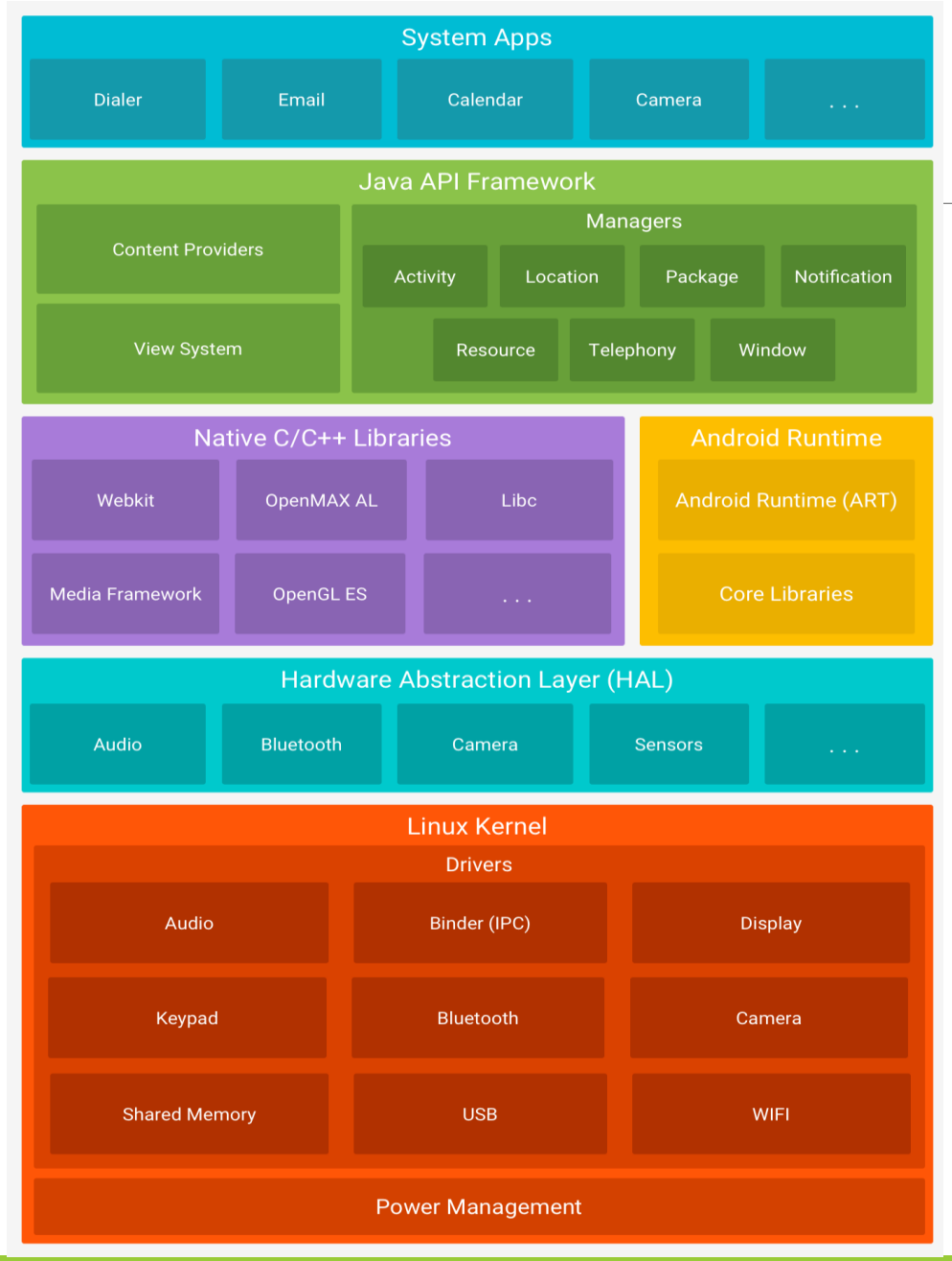
KUAN-TING LAI

2022/9/12

Android versions history

Name	Internal codename	Version number(s)	API level	Initial stable release date
Android 1.0	—	1.0	1	September 23, 2008
Android 1.1	Petit Four	1.1	2	February 9, 2009
Android Cupcake	Cupcake	1.5	3	April 27, 2009
Android Donut	Donut	1.6	4	September 15, 2009
Android Eclair	Eclair	2.0 – 2.1	5 – 7	October 27, 2009
Android Froyo	Froyo	2.2 – 2.2.3	8	May 20, 2010
Android Gingerbread	Gingerbread	2.3 – 2.3.7	9 – 10	December 6, 2010
Android Honeycomb	Honeycomb	3.0 – 3.2.6	11 – 13	February 22, 2011
Android Ice Cream Sandwich	Ice Cream Sandwich	4.0 – 4.0.4	14 – 15	October 18, 2011
Android Jelly Bean	Jelly Bean	4.1 – 4.1.2	16	July 9, 2012
Android KitKat	Key Lime Pie	4.4 – 4.4W.2	19 – 20	October 31, 2013
Android Lollipop	Lemon Meringue Pie	5.0 – 5.1.1	21 – 22	November 4, 2014
Android Marshmallow	Macadamia Nut Cookie	6.0 – 6.0.1	23	October 2, 2015
Android Nougat	New York Cheesecake	7.0 – 7.1.2	24 – 25	August 22, 2016
Android Oreo	Oatmeal Cookie	8.0 – 8.1	26 – 27	August 21, 2017
Android Pie	Pistachio Ice Cream	9	28	August 6, 2018
Android 10	Quince Tart	10	29	September 3, 2019
Android 11	Red Velvet Cake	11	30	September 8, 2020
Android 12 – Android 12L	Snow Cone – Snow Cone v2	12 – 12.1	31 – 32	October 4, 2021
Android 13	Tiramisu	**13**	33	August 15, 2022

References: https://en.wikipedia.org/wiki/Android_version_history



Platform Architecture

- Linux Kernel
- Hardware Abstraction Layer (HAL)
- Android Runtime (ART)
- Native C/C++ Libraries
- Java API Framework
  - View System
  - Resource Manager
  - Notification Manager
  - Activity Manager
  - Content Providers

References: https://developer.android.com/guide/platform

New Android Language: Kotlin

Application Fundamentals

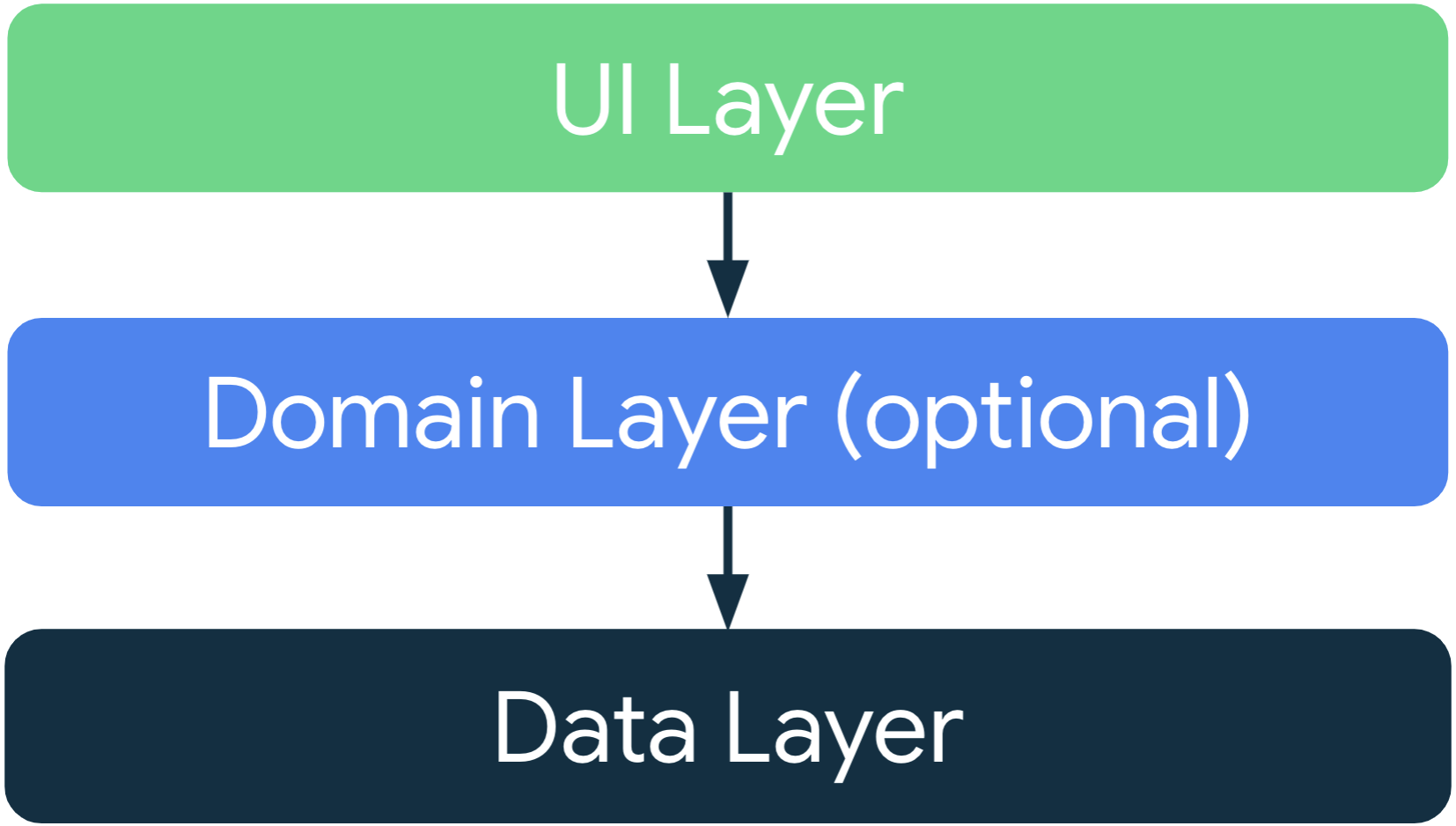
- The Android operating system is a multi-user Linux system
- By default, the system assigns each app a unique Linux user ID
- Each process has its own virtual machine (VM)
- Every app runs in its own Linux process

References: https://developer.android.com/guide/components/fundamentals

APP Components

Activities	Handle UI and interact with user Ex: A photo app calls an email app to share photos
Services	Run background process Ex: Music playback
Broadcast Receivers	Receive system events Ex: Alarm, battery low, …
Content Providers	Manage APP data Ex: SQLite database

Recommended app architecture



References: https://developer.android.com/topic/architecture

Activities

- Activity enables one app to invoke another app
- One screen, one activity
- Use **Intent** to communicate

References: https://developer.android.com/guide/components/activities/intro-activities#java

Services

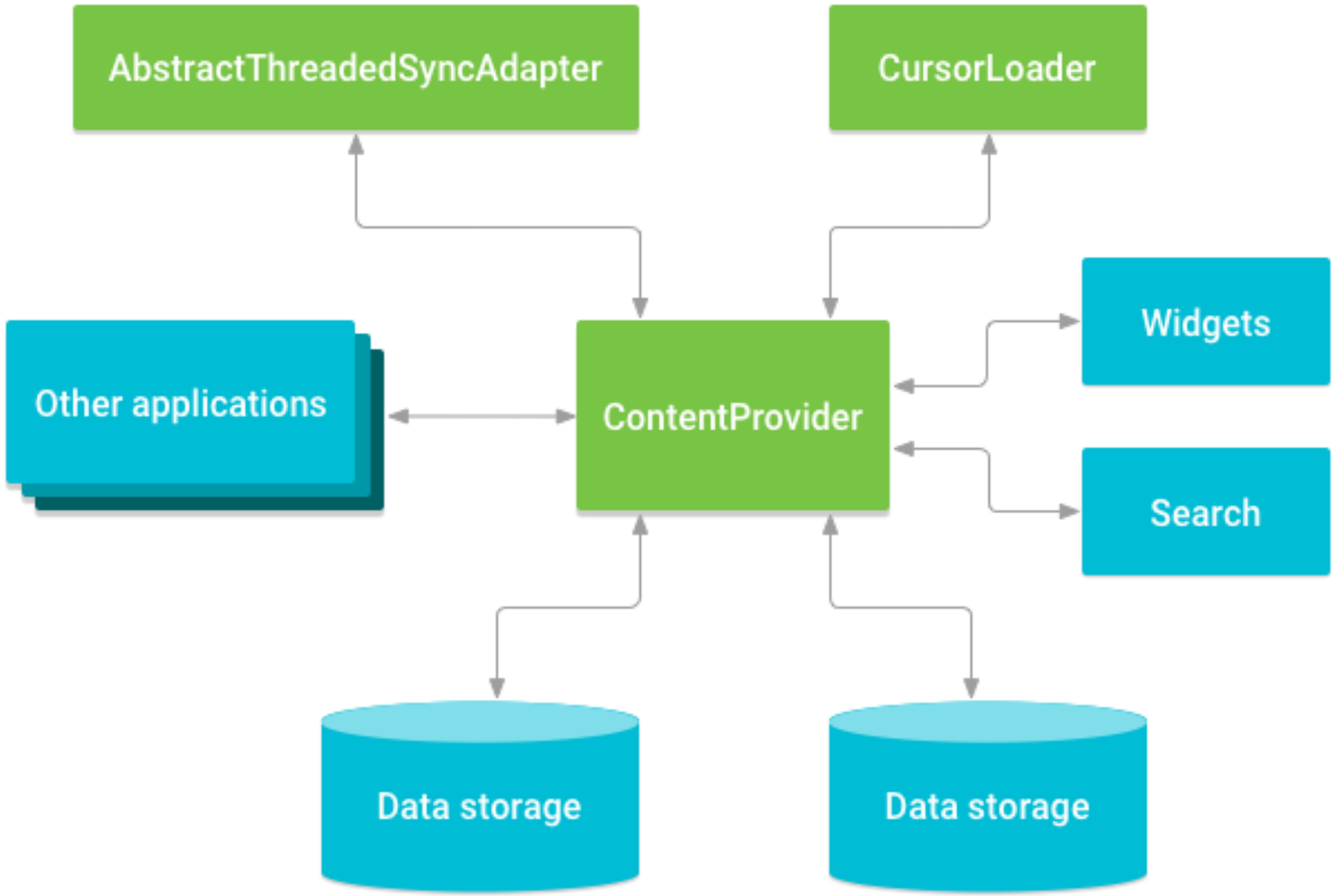
- Running in background
- Create a background service
- Send work requests to a service
- Report work status
- Bound services

References: https://developer.android.com/guide/components/services

Broadcast Receivers

- Send or receive broadcast messages from the Android system and other Android apps
- Publish-subscribe design pattern
- Messages are wrapped in **Intent**

References: https://developer.android.com/guide/components/broadcasts



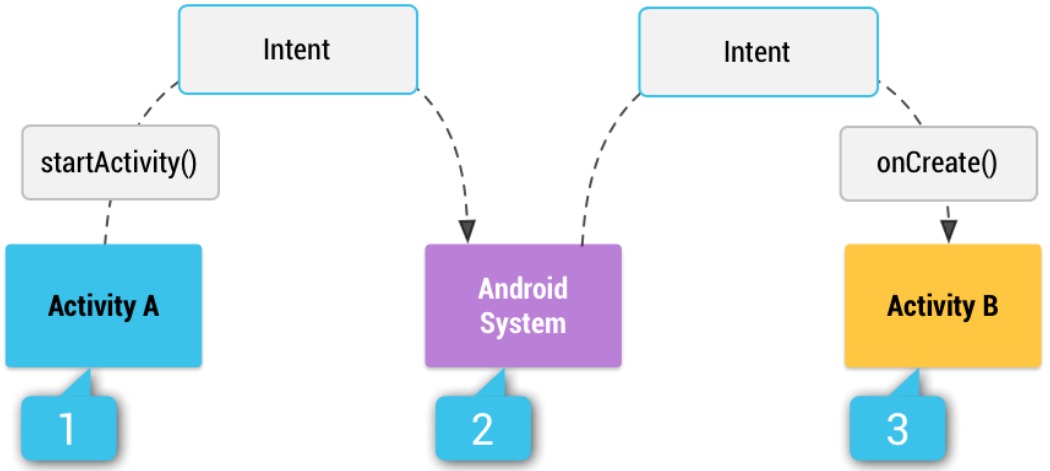
Content Providers

- Sharing data with other apps
- Sending data to a widget
- Returning custom search suggestions through the search framework using SearchRecentSuggestionsProvider
- Synchronizing application data with your server using an implementation of AbstractThreadedSyncAdapter
- Loading data in your UI using a CursorLoader

References: https://developer.android.com/guide/topics/providers/content-provider-basics

Intent and Intent Filters

- A message object used to invoke other components
- Starting an activity
- Starting a service
- Delivering a broadcast
- Explicit intents and implicit intents



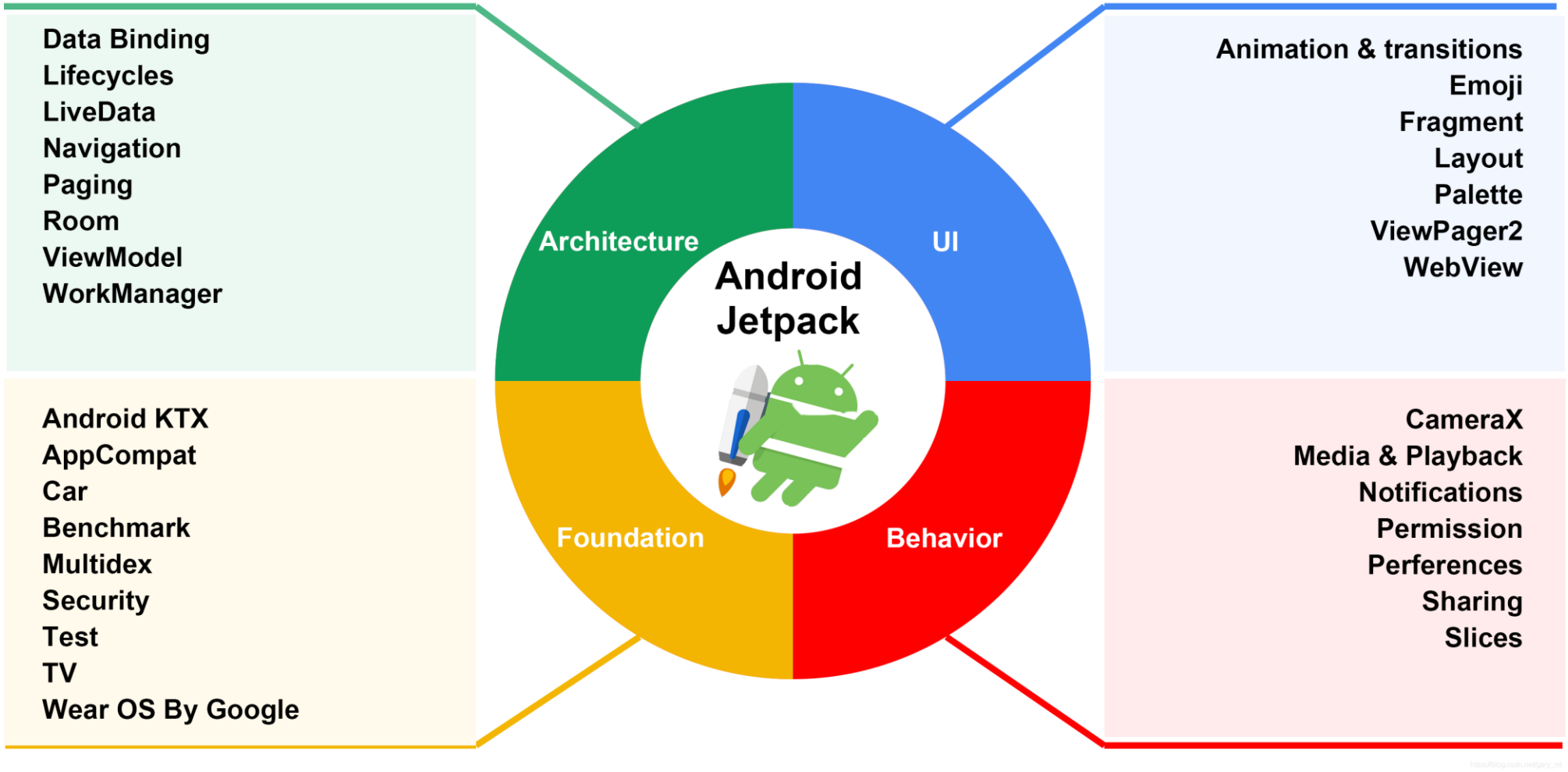
Implicit Intent Delivery

References: https://developer.android.com/guide/components/intents-filters#java

Other Components

Fragments	Represent a behavior or a portion of user interface in an Activity
Views	UI elements that are drawn onscreen including buttons, lists forms etc.
Layouts	View hierarchies that control screen format and appearance of the views
Intents	Messages wiring components together
Resources	External elements, such as strings, constants and drawable pictures
Manifest	Configuration file for the application

Android Jetpack



Start Jetpack: https://developer.android.google.cn/jetpack/

Install Android Studio

Android Studio provides the fastest tools for building apps on every type of Android device.

Click → Download Android Studio

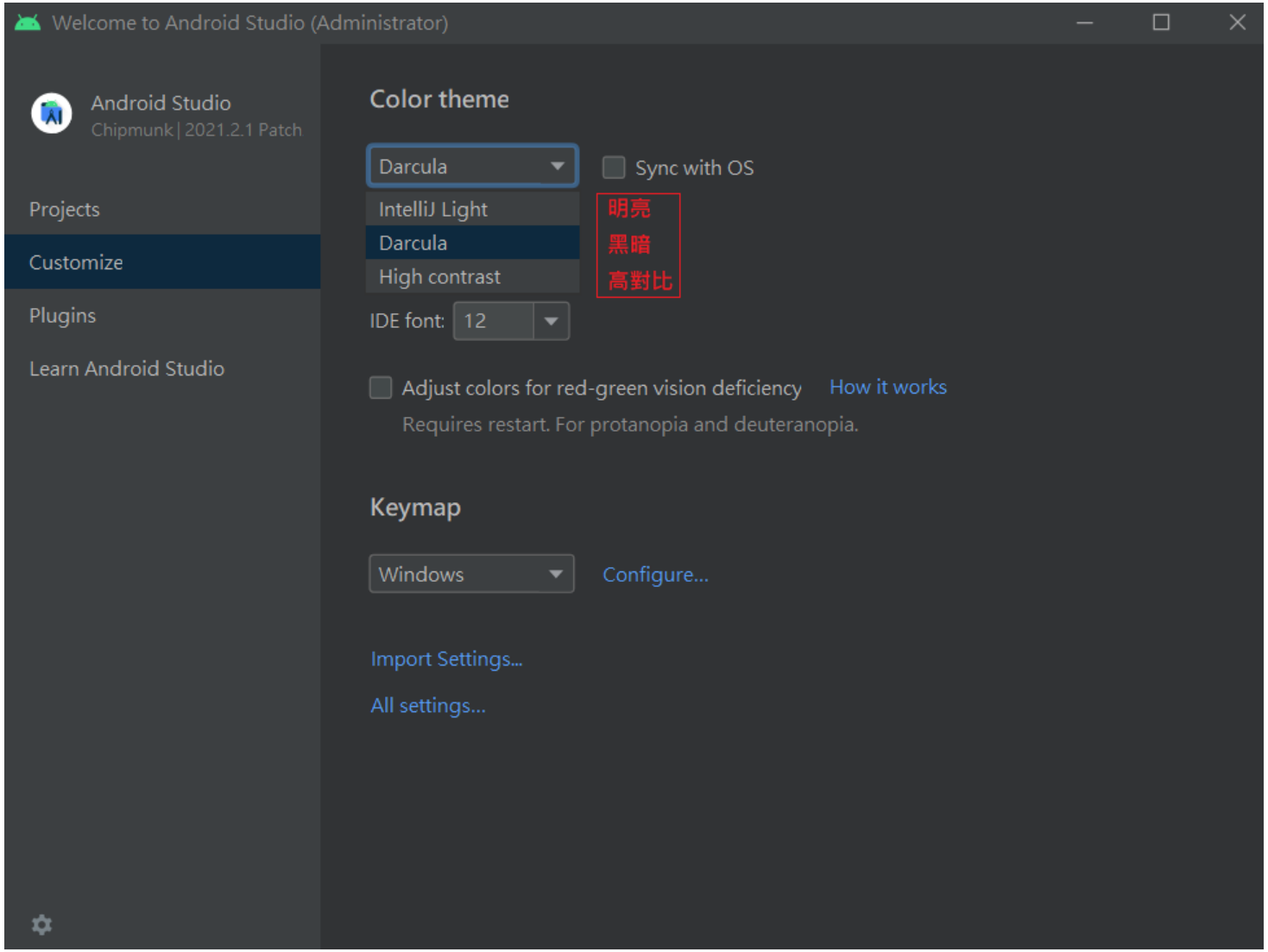
Android Studio Chipmunk | 2021.2.1 Patch 2 for Windows 64-bit (929 MiB)

Download options

Release notes

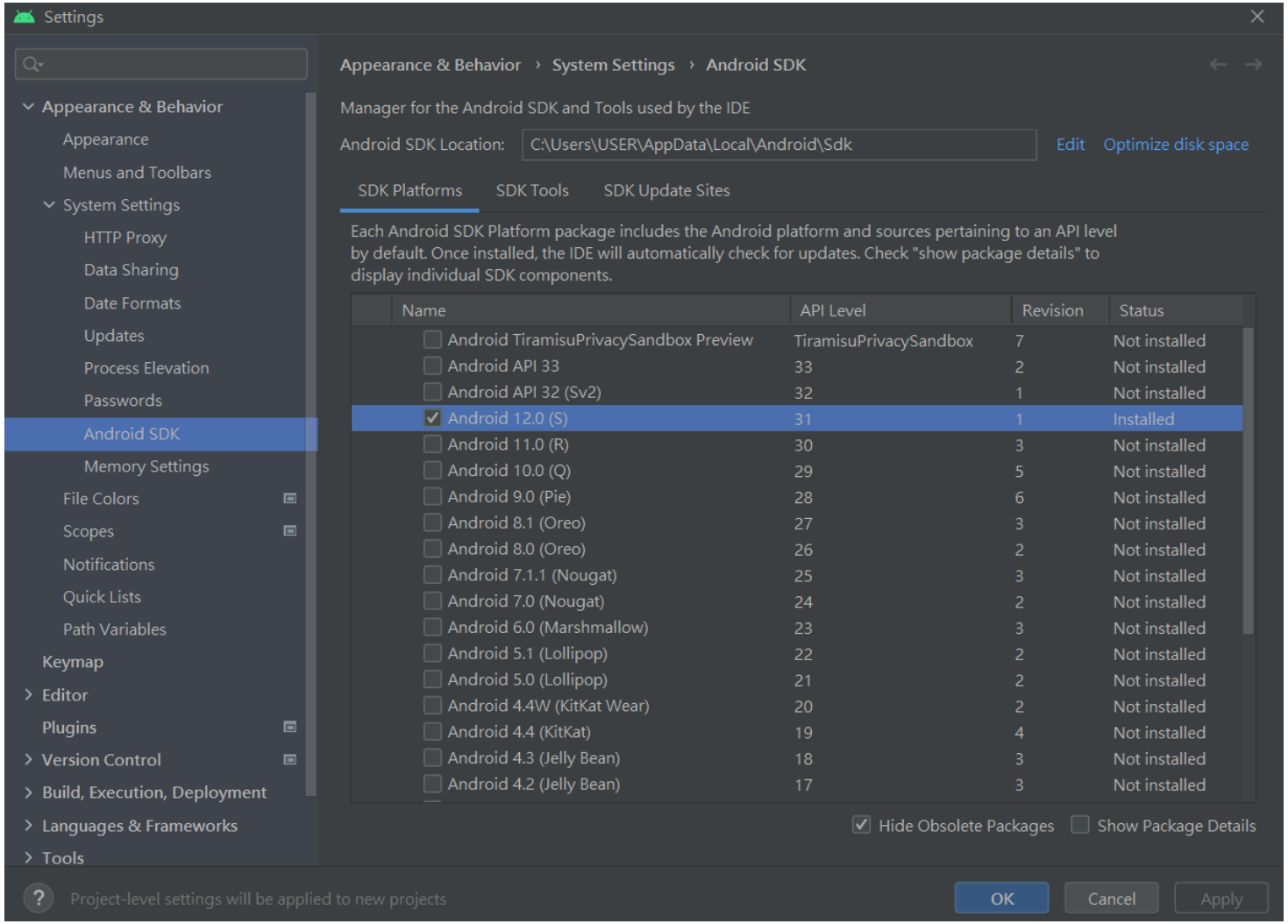
Download Android Studio: https://developer.android.com/studio

Android Studio



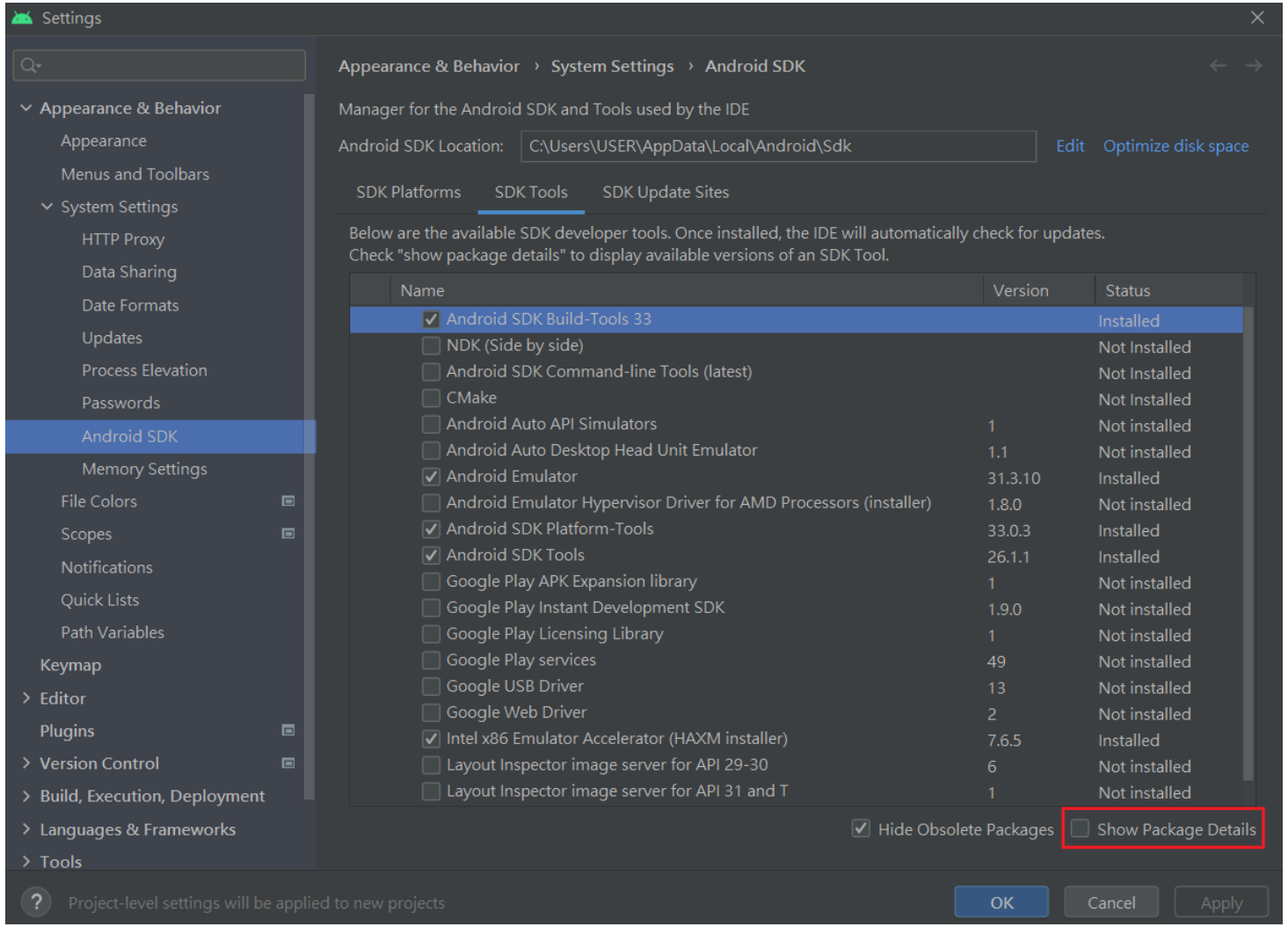
- Choose the color theme you want
- Click “All settings” to check plugin version

Check Android SDK version



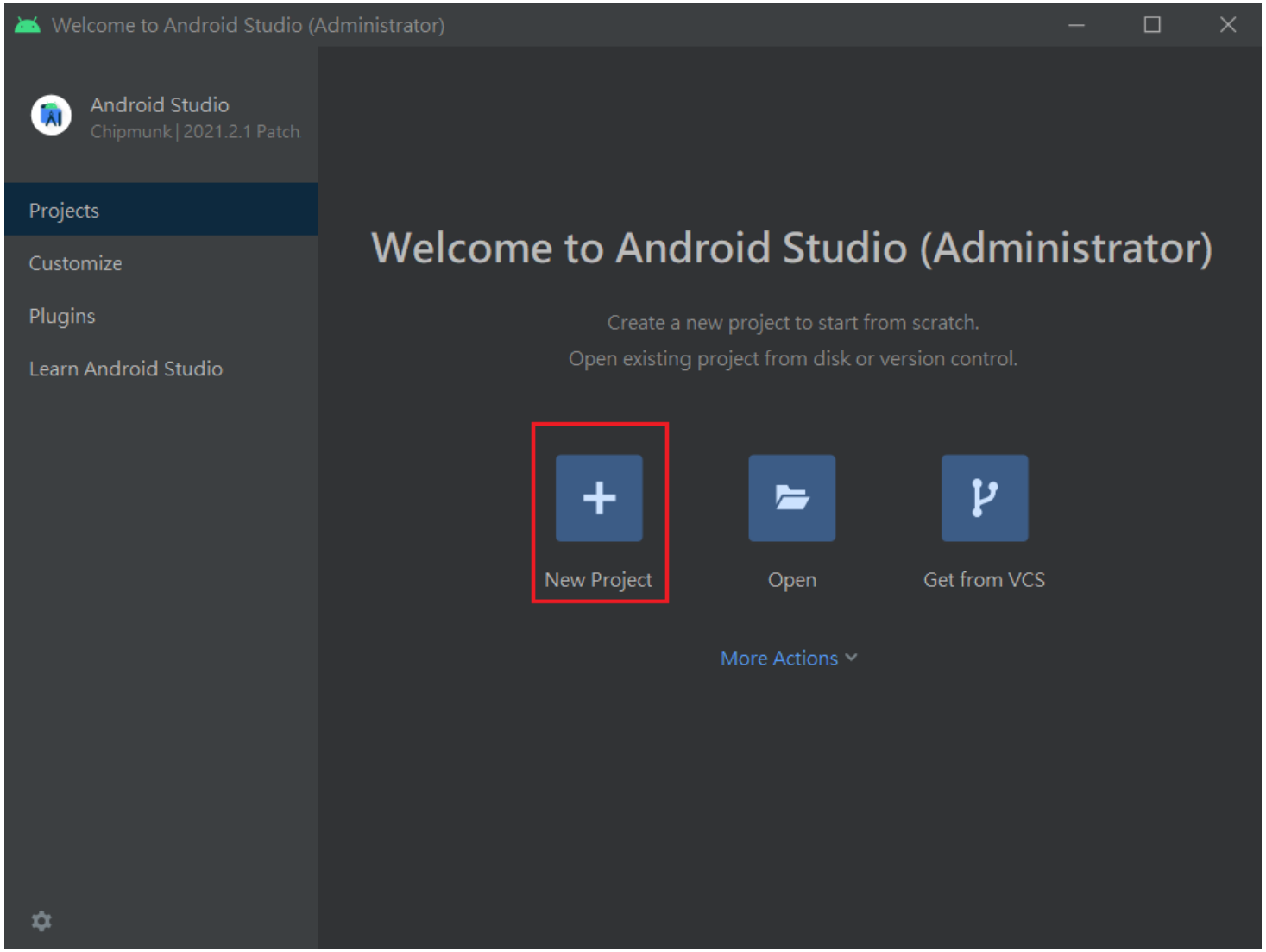
- Go to Appearance & Behavior → System Settings → Android SDK
- Android SDK version should be 12 (API level 31)

Check Android SDK Build-Tools version



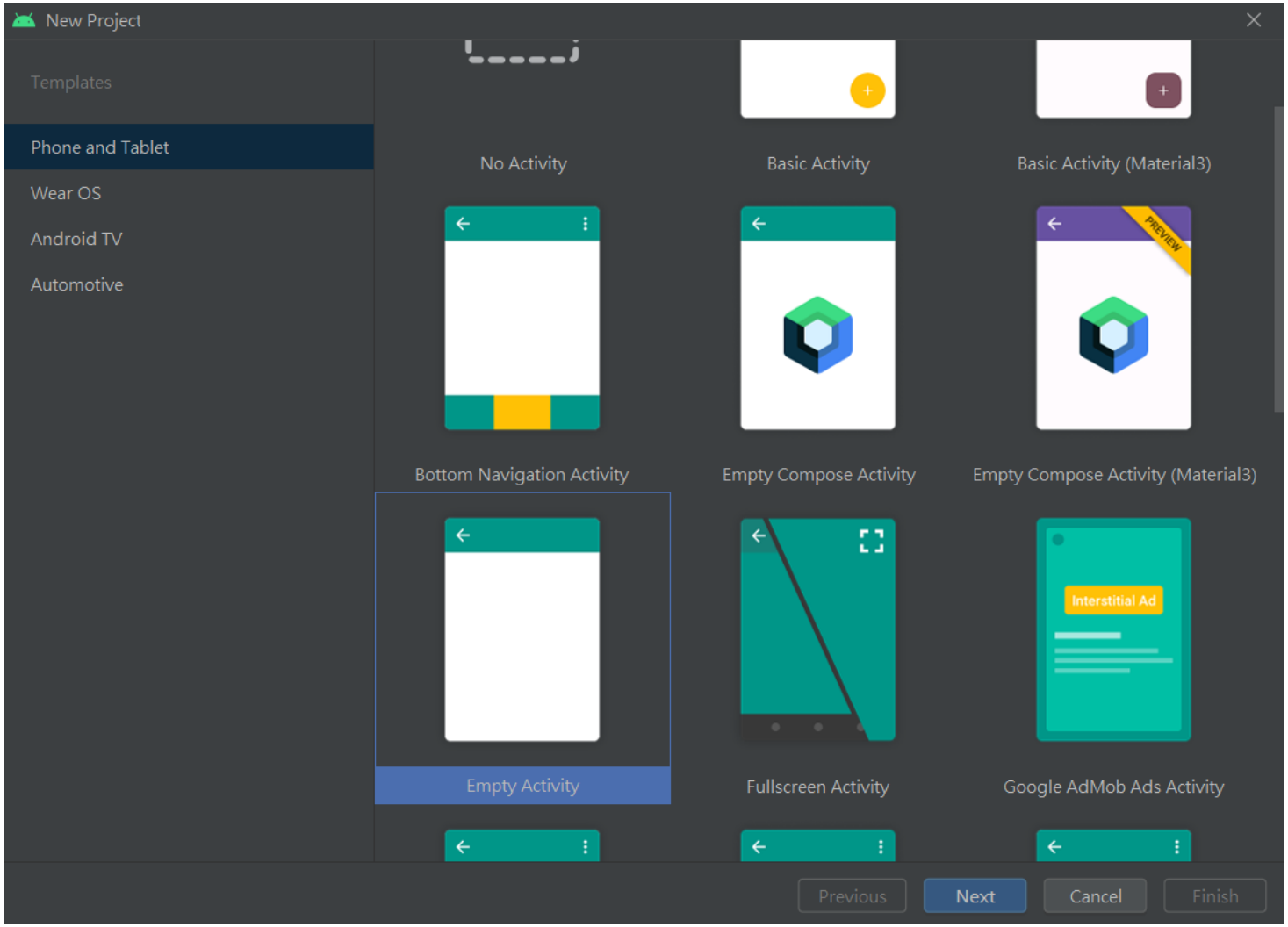
- You can choose to click “Show Package Details to select a specific version
- Please check the box to install these SDK Tools

Create an Android Project



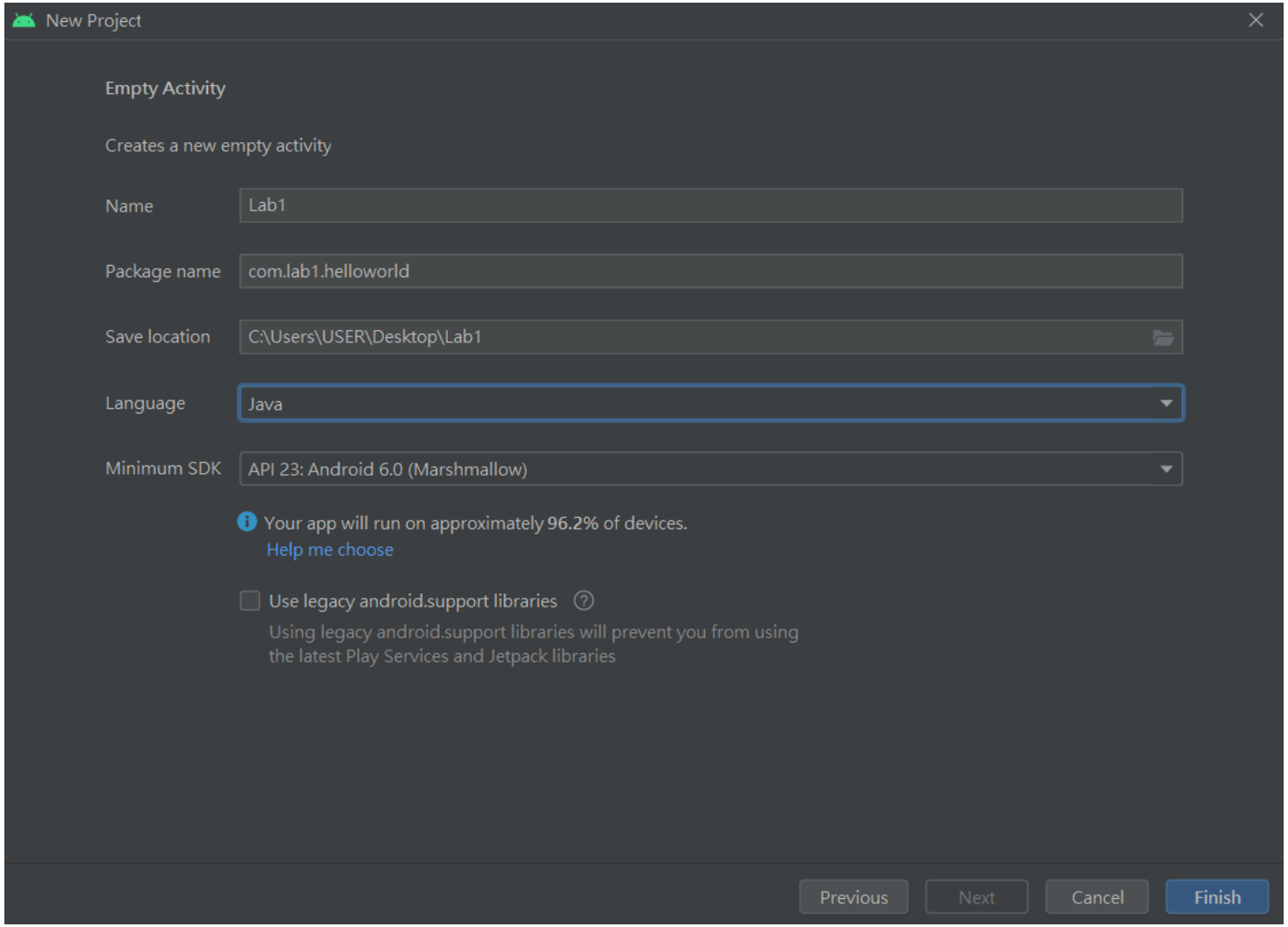
- Click New Project

Create an Android Project (Cont.)



- Choose “Empty Activity”
- Click “Next”

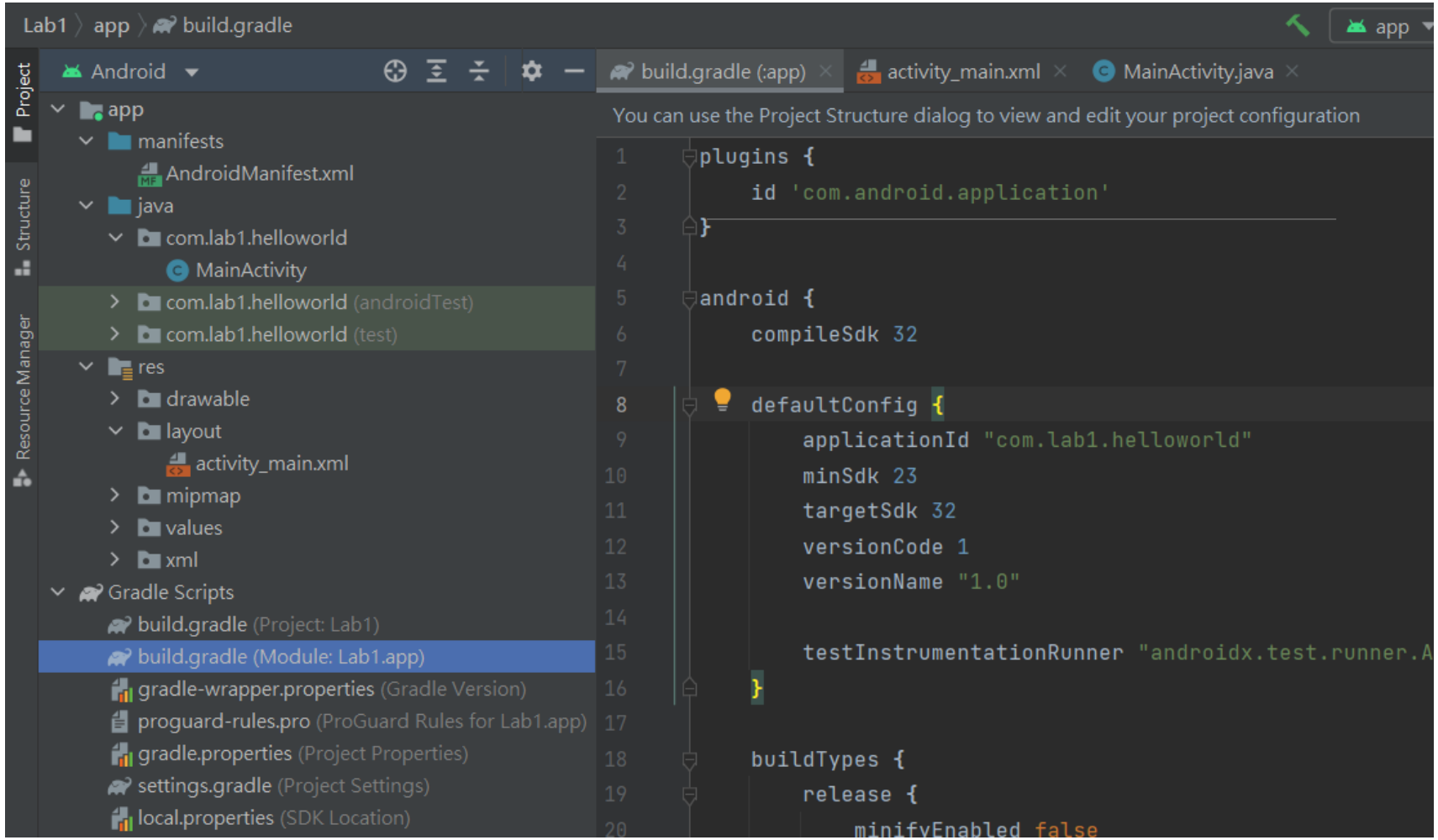
Create an Android Project (Cont.)



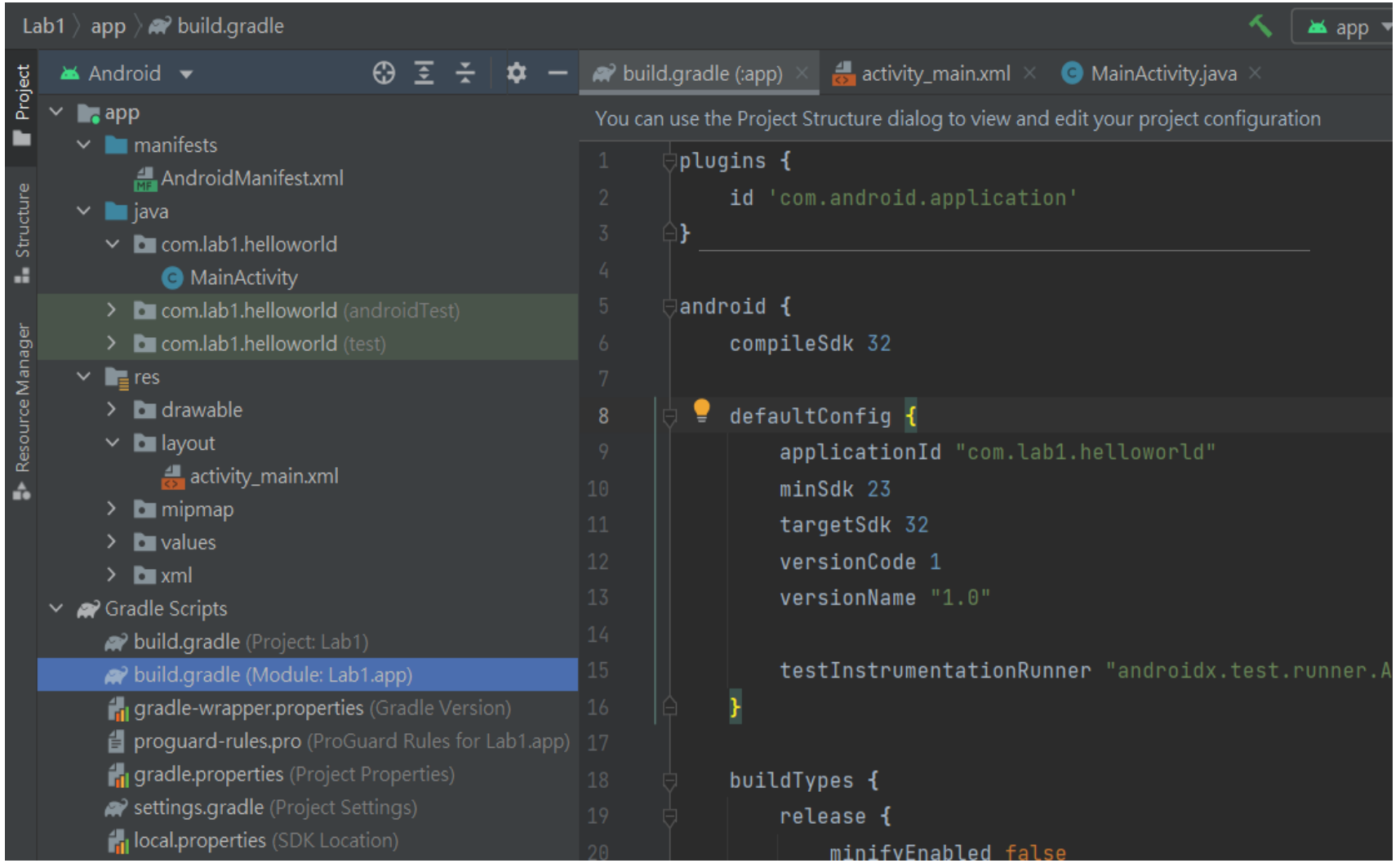
- Remember the “Save location” you set
- Language choose “Java”
- Click “Finish”

Project Files

- app > java > com.lab1.helloworld > MainActivity
- app > res > layout > activity_main.xml
- app > manifests > AndroidManifest.xml
- Gradle Scripts > build.gradle



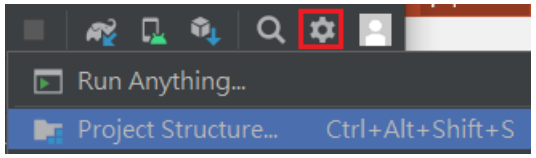
build.gradle

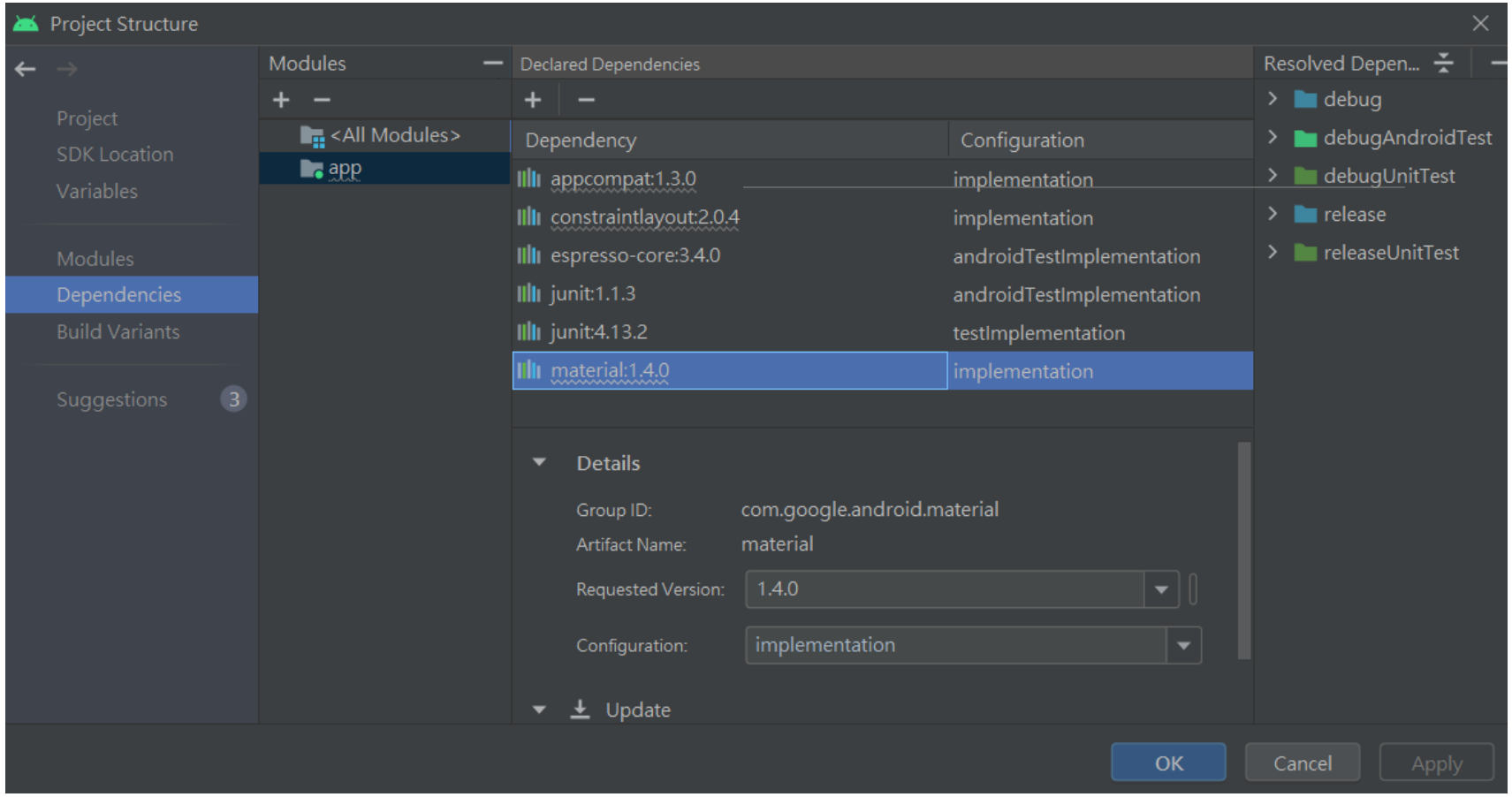


- minSdk (lowest possible) <= targetSdk == compileSdk (latest SDK) <= build-Tools Version (latest Build-Tools)

Dependencies

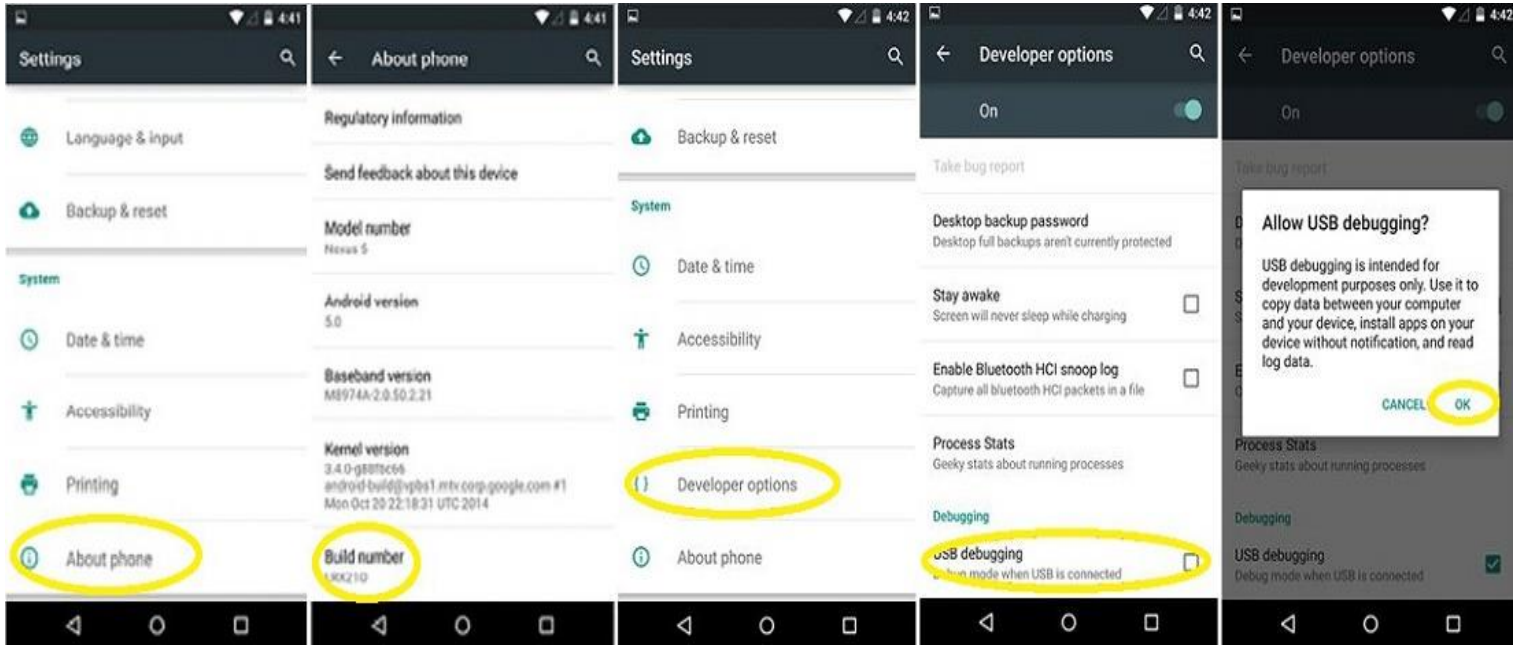
- If a dependency has a tilde, a newer version of the dependency is available

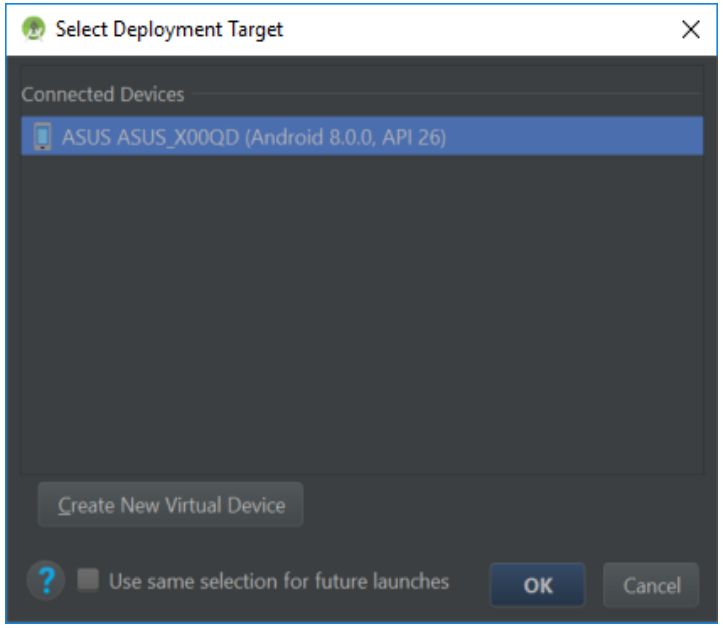




Run Your App

- On real device (Your phone)
  - Enable USB debugging
  - Settings → System → About Phone → Build number (CLICK 7 TIMES)
  - Developer options -> USB Debugging (Allow)
  - Click **Run**

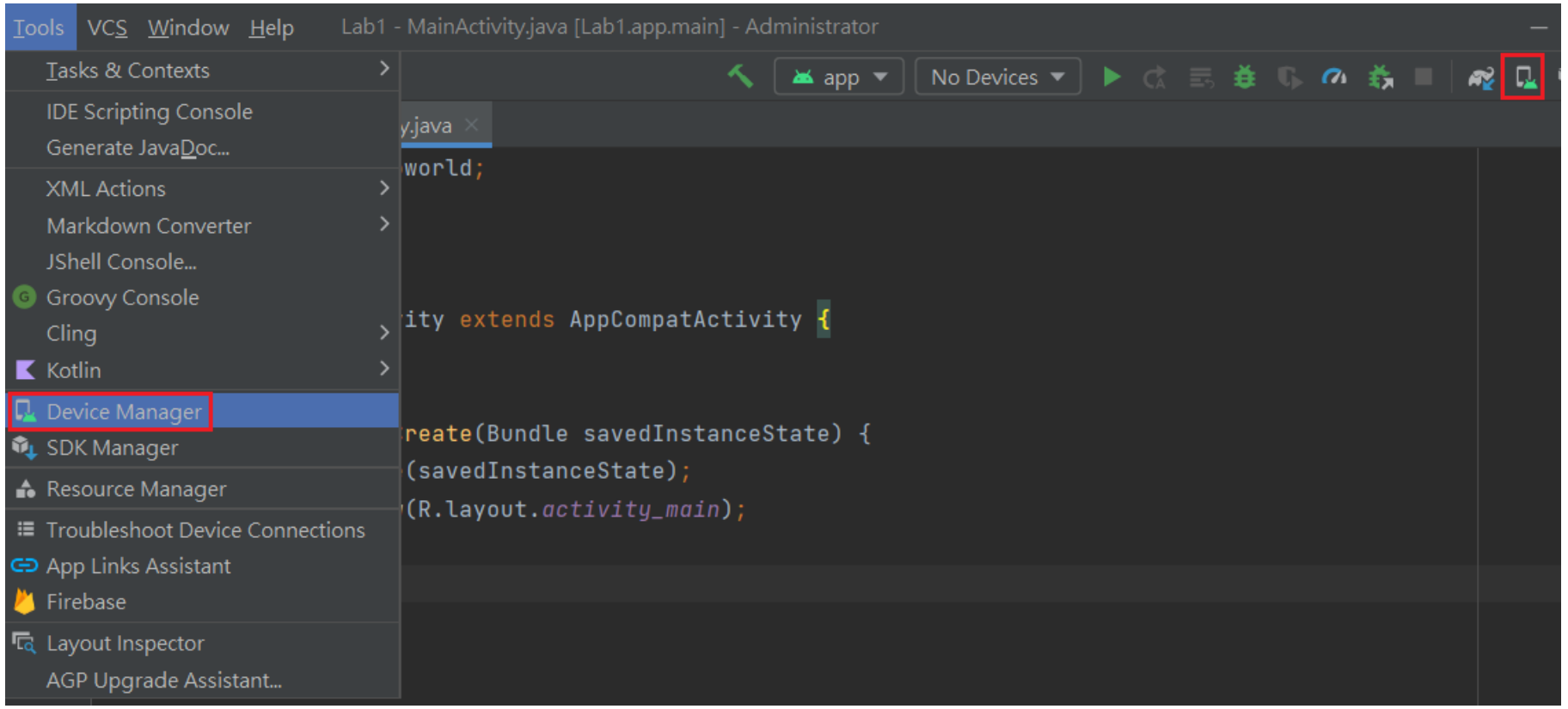




References: https://developer.android.com/studio/run/device

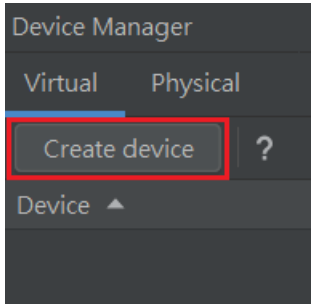
Create Android Virtual Device

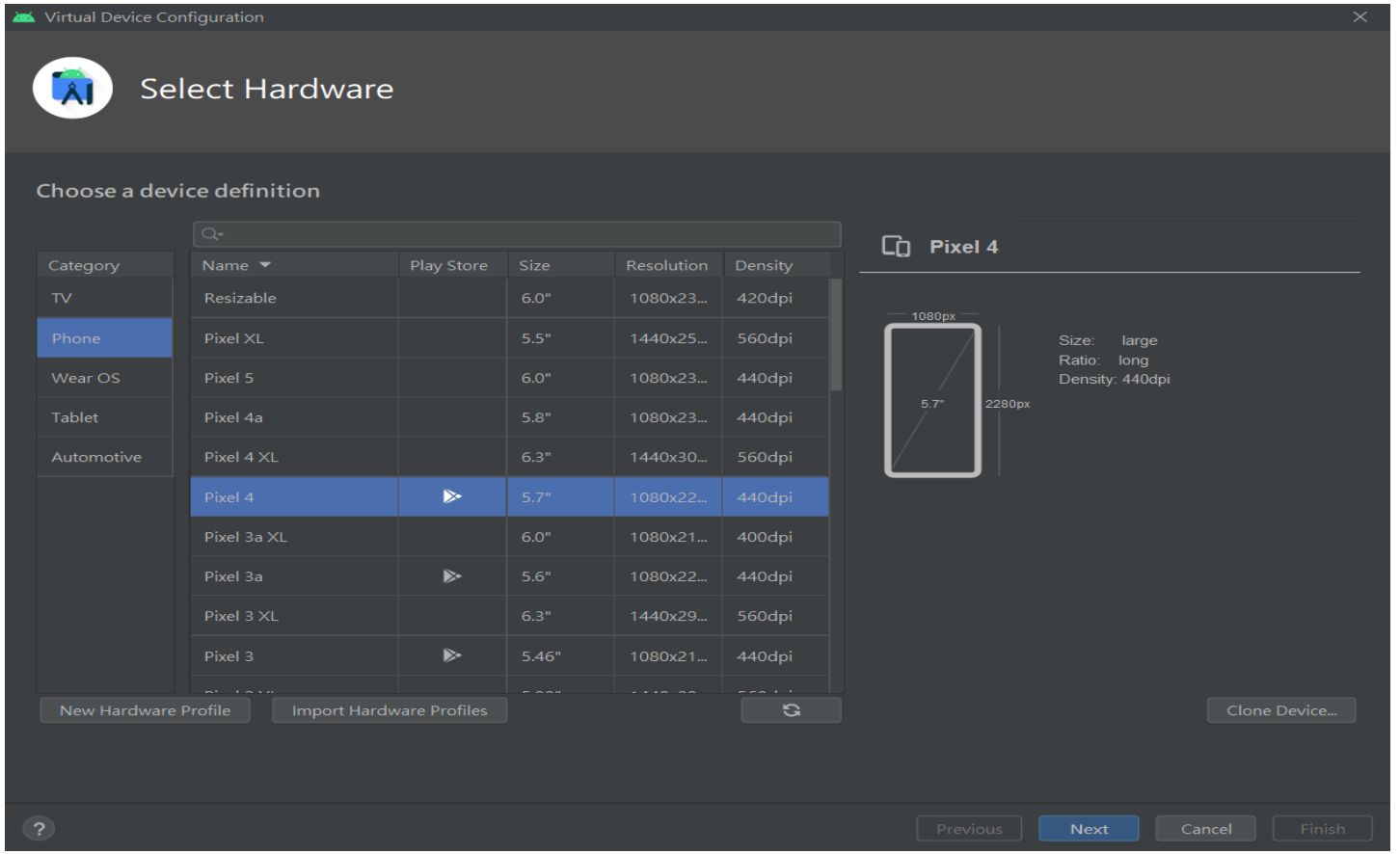
- Click **Device Manager**



Create Android Virtual Device (Cont.)

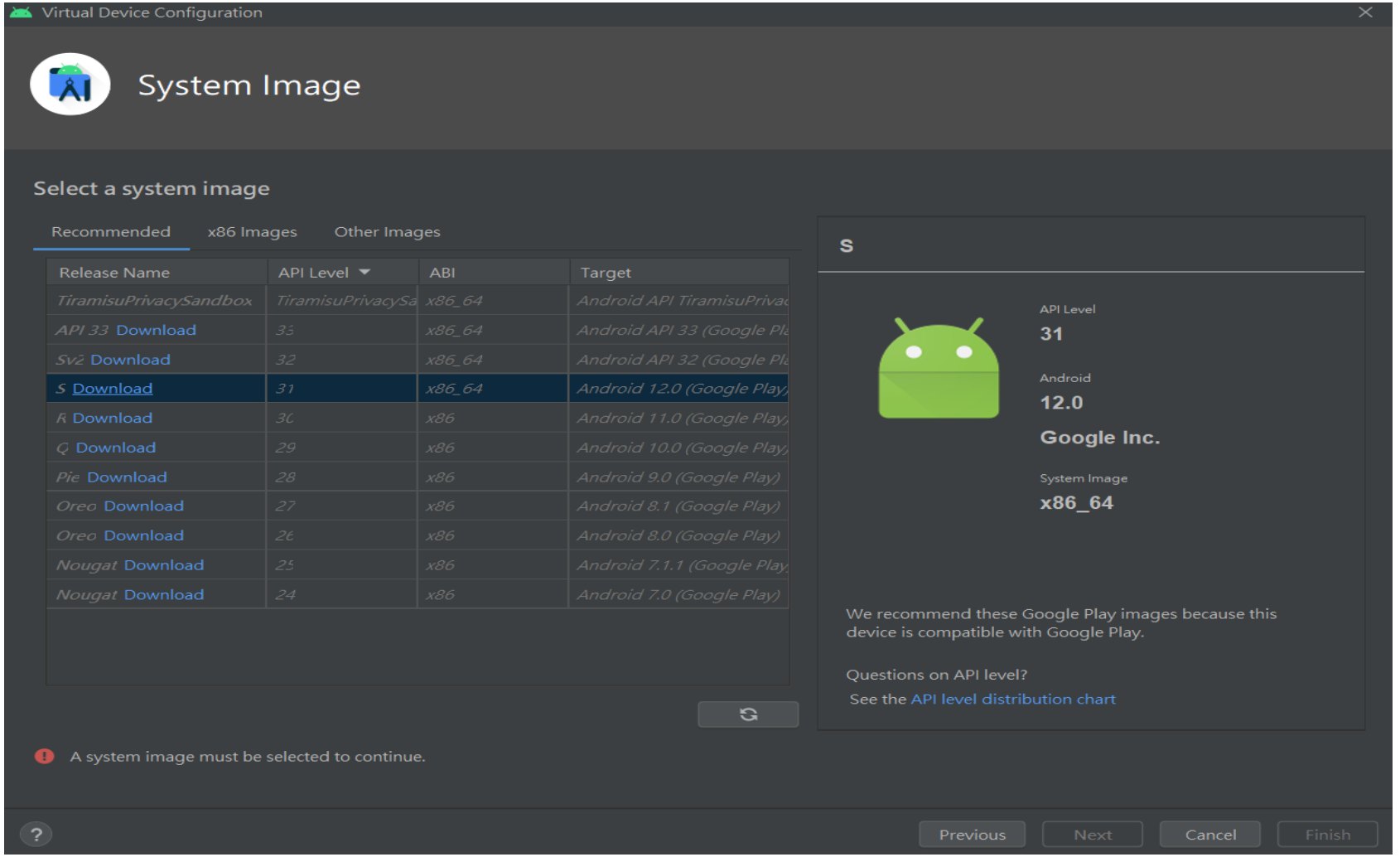
- Click **“Create Device”**
- Select a device → Next





Download System Image

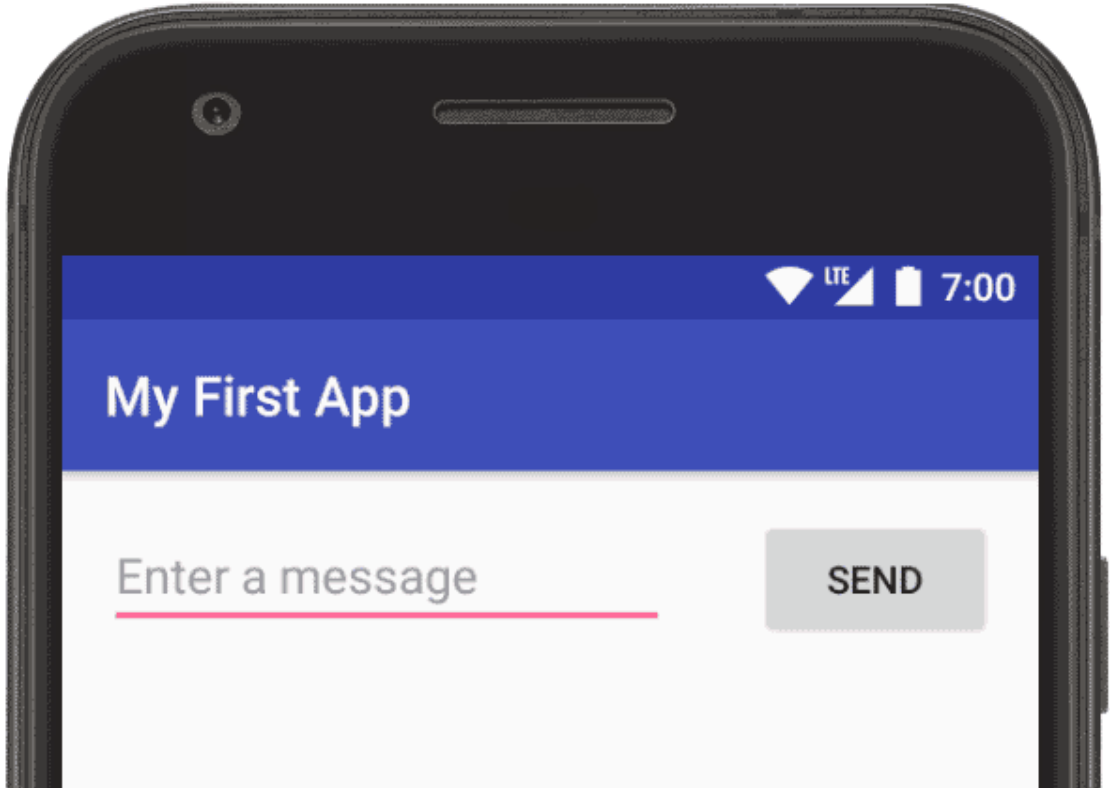
Click Download



Today’s Assignment

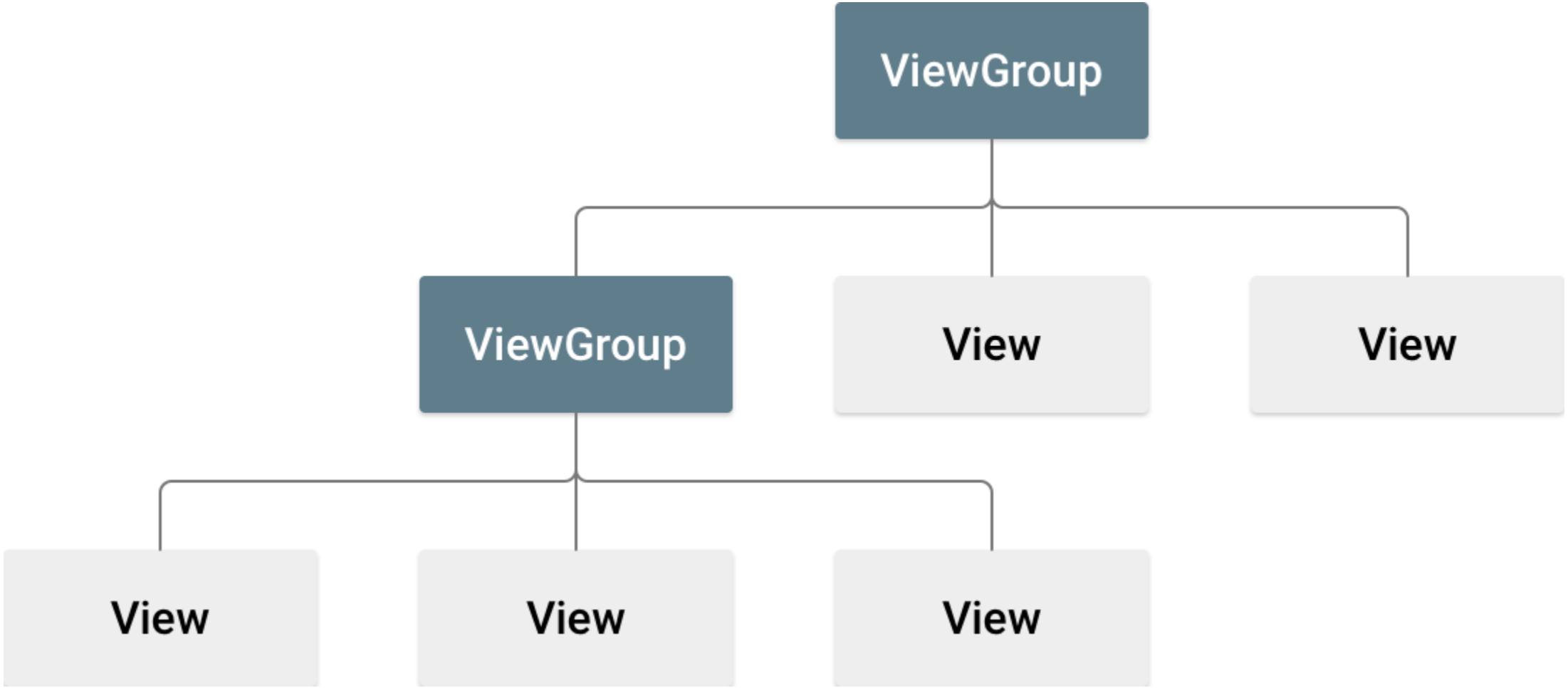
Simple Text Sending APP

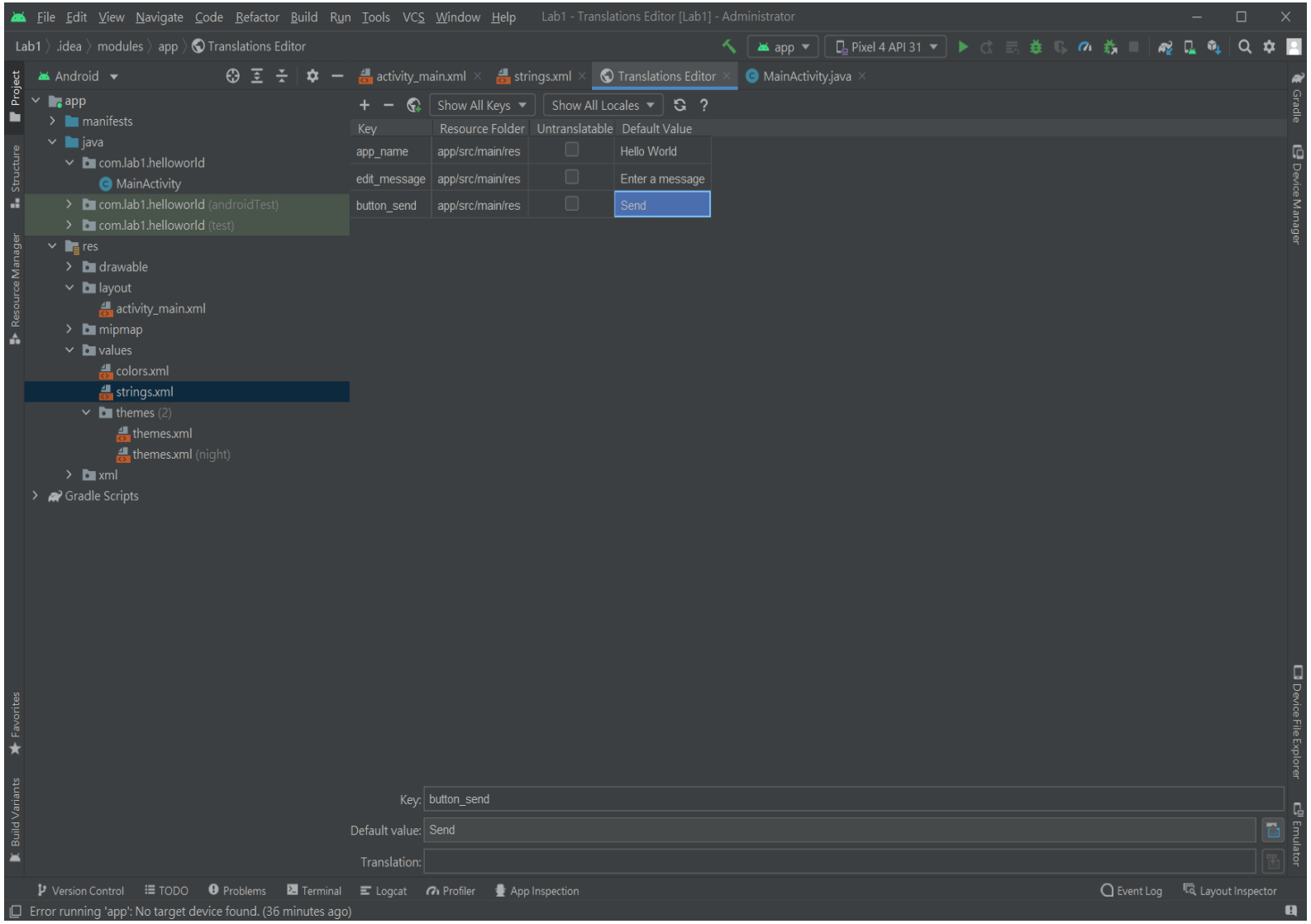
- Create a layout that includes a text box and a button
- Sending the content of the text box to another activity



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

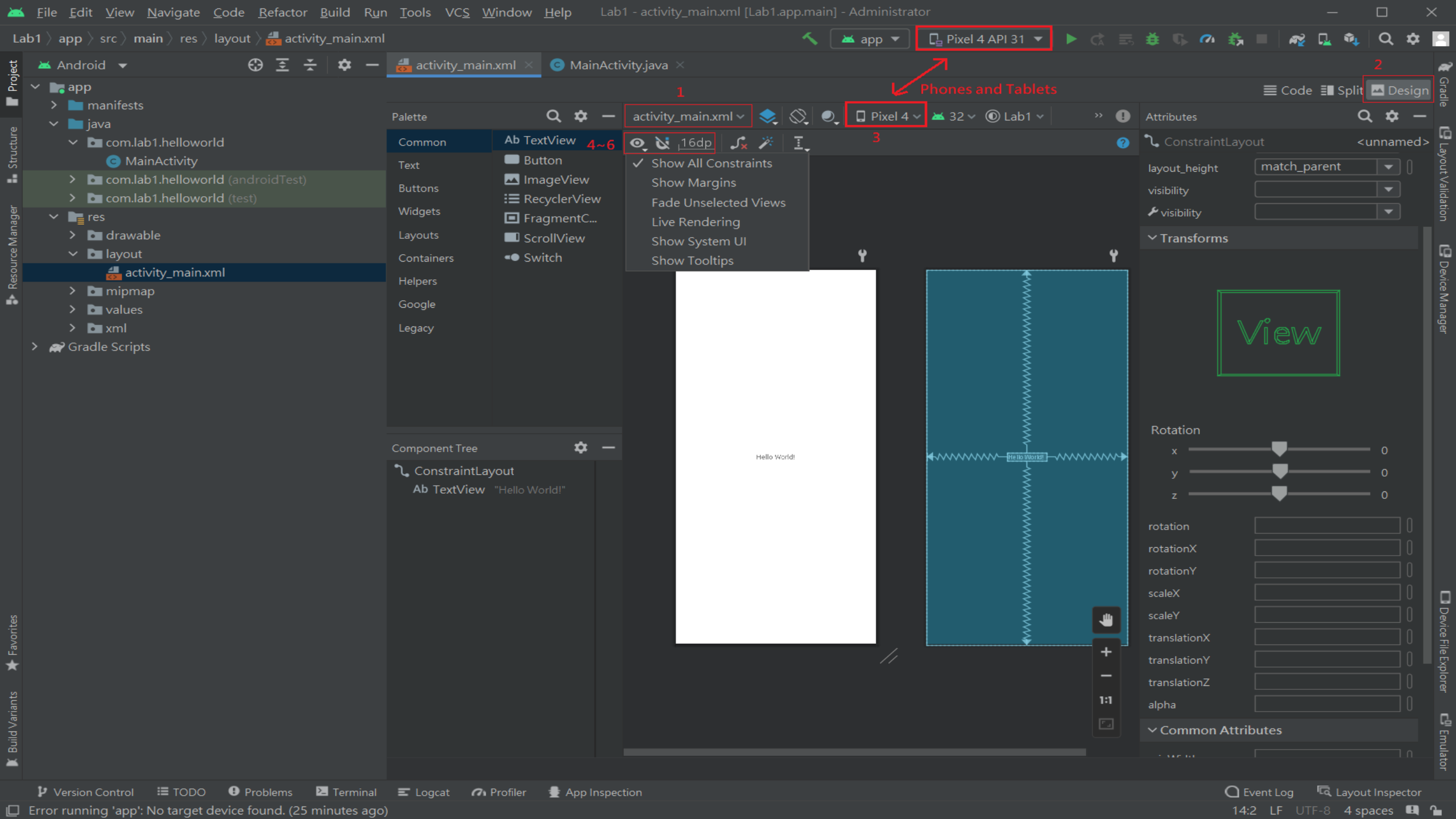
Hierarchy of Layouts



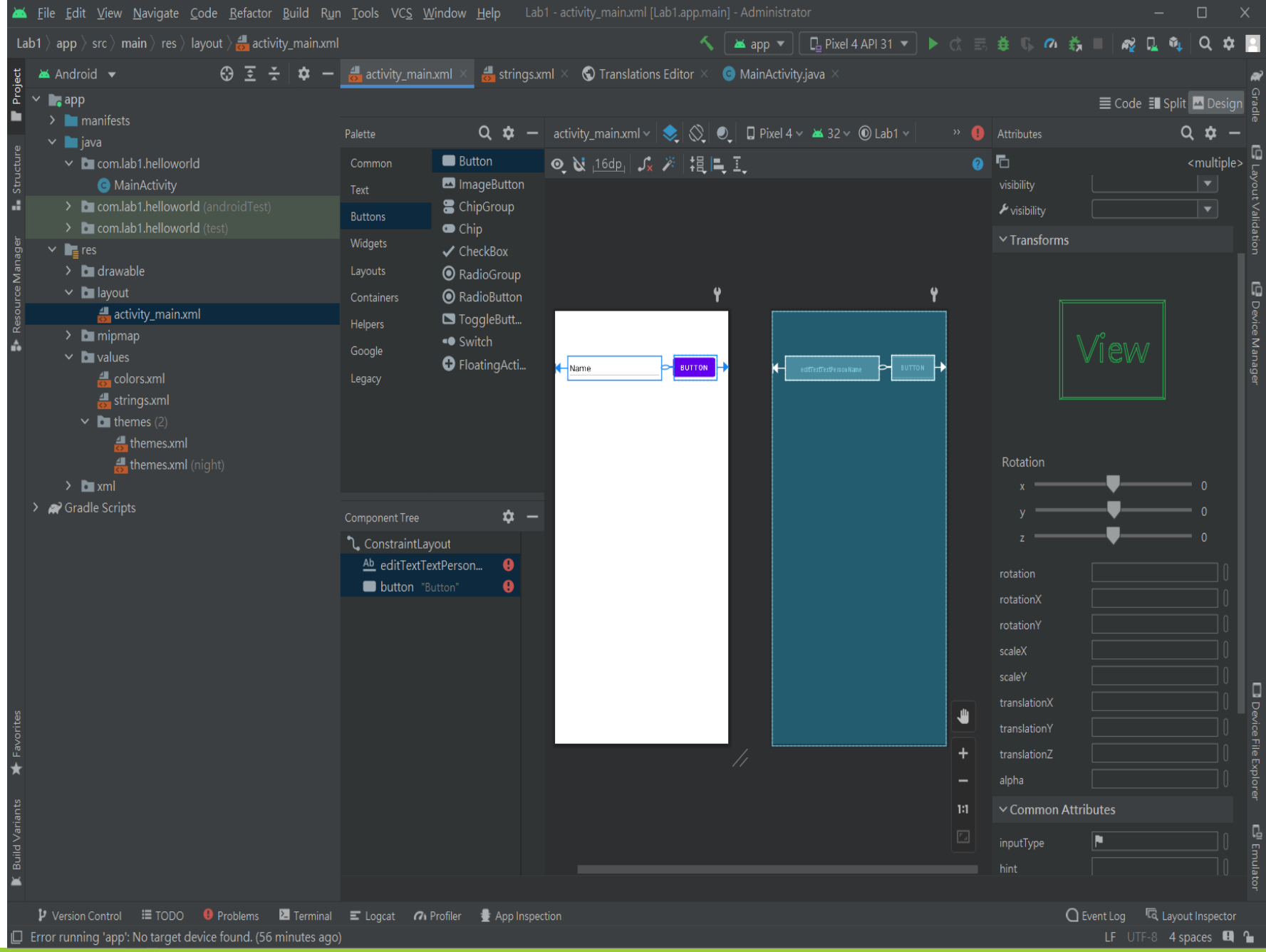


Change UI Strings

- **app > res > values > strings.xml**
- strings.xml -> Open Editor
- **Add Key** +
- Add two string pairs (key => value):
  - edit_message => “Enter a message”
  - button_send => “Send”

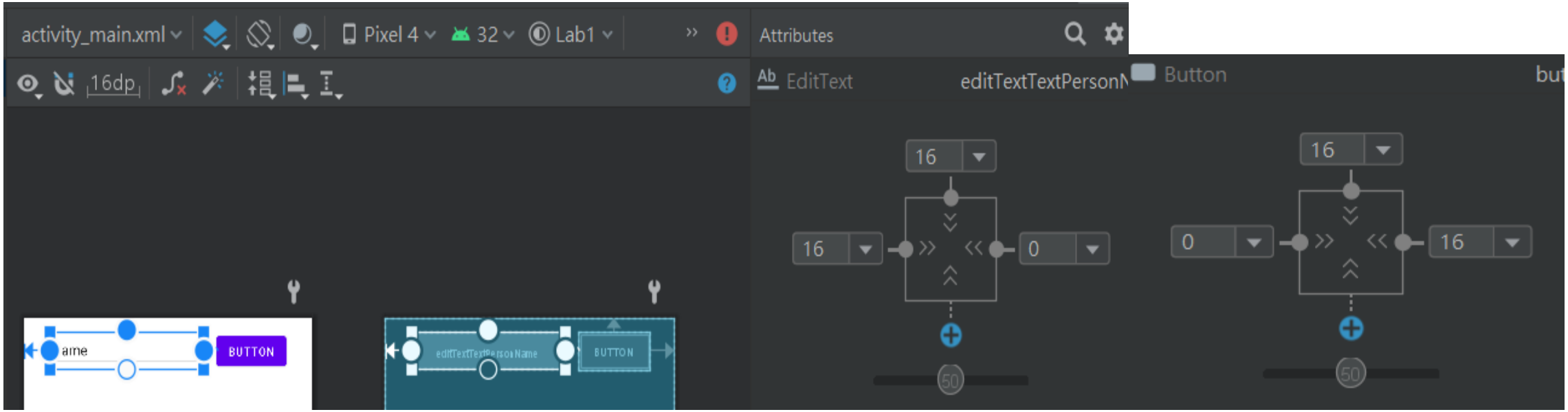

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Lab1 - activity_main.xml [Lab1.app.main] - Administrator
Lab1 app src main res layout activity_main.xml
app
Pixel 4 API 31
Android
activity_main.xml
MainActivity.java
1
Phones and Tablets
2
Code Split Design
Palette
activity_main.xml
Pixel 4
32
Lab1
Common
Text
Buttons
Widgets
Layouts
Containers
Helpers
Google
Legacy
Ab TextView
Button
ImageView
RecyclerView
FragmentC...
ScrollView
Switch
4~6
16dp
3
Show All Constraints
Show Margins
Fade Unselected Views
Live Rendering
Show System UI
Show Tooltips
Component Tree
ConstraintLayout
Ab TextView "Hello World!"
Hello World!
Attributes
ConstraintLayout
<unnamed>
layout_height
match_parent
visibility
visibility
Transforms
View
Rotation
x
y
z
rotation
rotationX
rotationY
scaleX
scaleY
translationX
translationY
translationZ
alpha
Common Attributes
Project
Structure
Resource Manager
Favorites
Build Variants
Gradle
Layout Validation
Device Manager
Device File Explorer
Emulator
Version Control
TODO
Problems
Terminal
Logcat
Profiler
App Inspection
Event Log
Layout Inspector
Error running 'app': No target device found. (25 minutes ago)
14:2 LF UTF-8 4 spaces

- Drag & drop “PlainText” & “Button”
- Select both PlainText & Button
- Right click -> Chain > Create Horizontal Chain



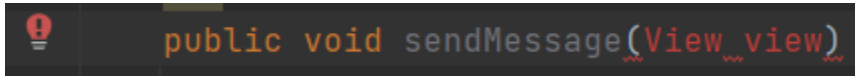
Set Button & PlainText Margin

- Select the button and open the **Attributes** window
- Set right and Top margin to 16
- Select PlainText and set left and Top margin to 16

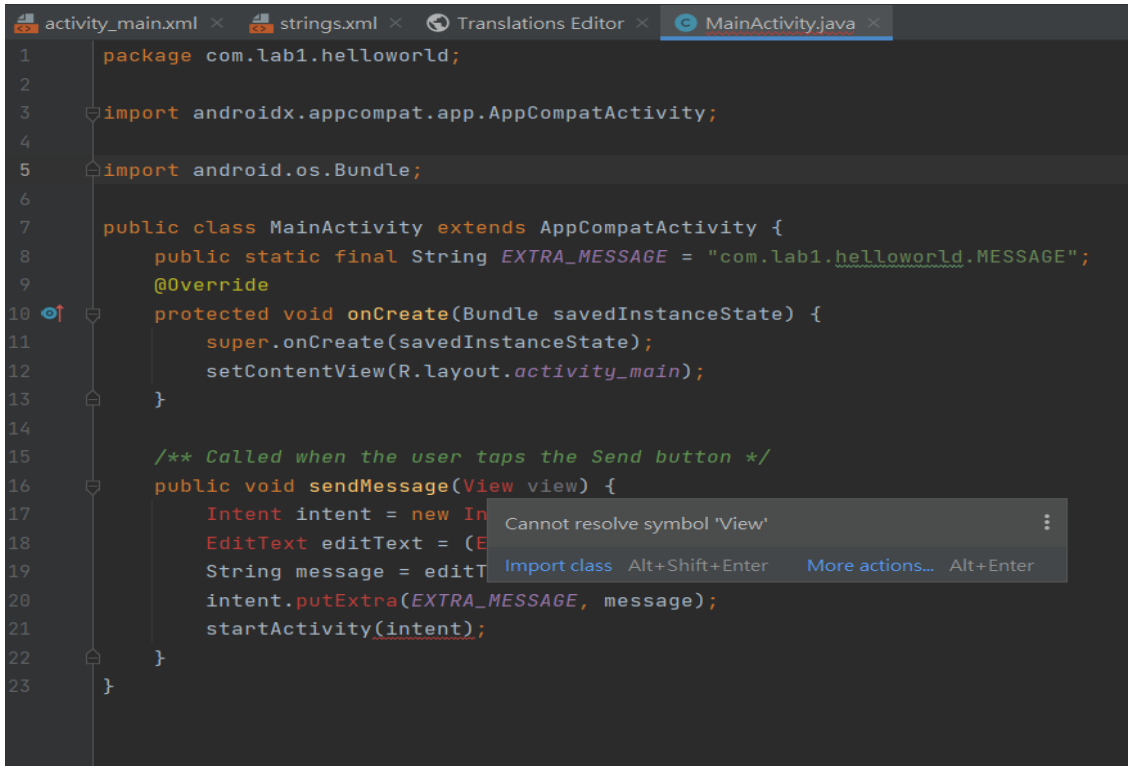


Start Another Activity

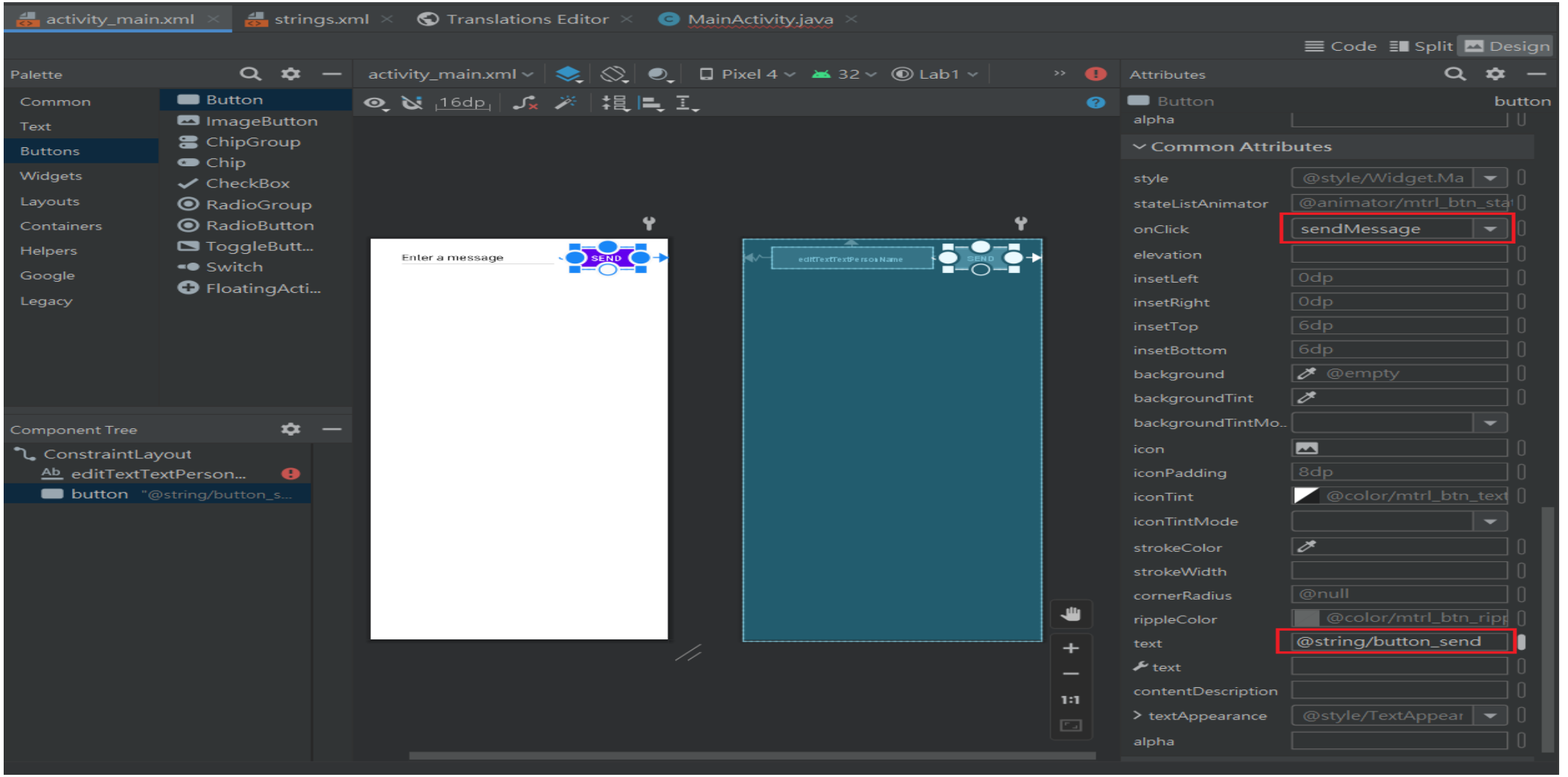
- Open MainActivity.java and add "**sendMessage()**"
- Auto fix error (Alt + Enter)
- Ex: "Import classs"




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

    /** Called when the user taps the Send button */
    public void sendMessage(View view) {
        // Do something in response to button
    }
}
```



Assign **sendMessage()** to Button *onClick*



Build an Intent

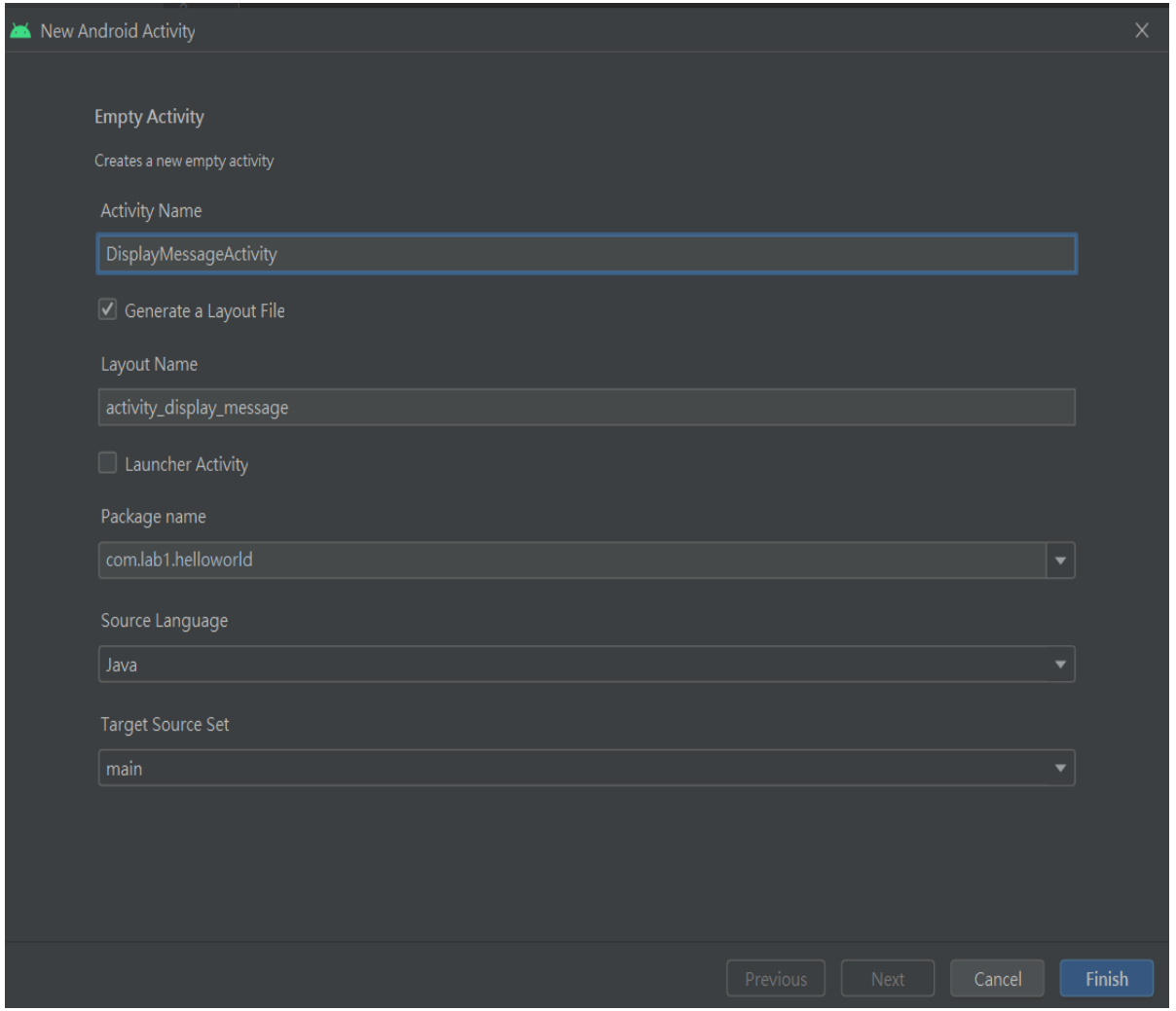
```
public class MainActivity extends AppCompatActivity {
    // EXTRA_MESSAGE is just a Key, and you can pick any string (ex: "ABC")
    public static final String EXTRA_MESSAGE = "com.lab1.helloworld.MESSAGE";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    /** Called when the user taps the Send button */
    public void sendMessage(View view) {
        Intent intent = new Intent(this, DisplayMessageActivity.class);
        // R.id.editTextTextPersonName is set in activity_main.xml > PlainText > attributes > id
        EditText editText = findViewById(R.id.editTextTextPersonName);
        String message = editText.getText().toString();
        intent.putExtra(EXTRA_MESSAGE, message);
        startActivity(intent);
    }
}
```

Explain code in SendMessage()

- The Intent constructor takes two parameters:
  - A Context as its first parameter (this is used because the Activity class is a subclass of Context)
  - The Class of the app component to which the system should deliver the Intent
- The putExtra() method adds the EditText's value to the intent. An Intent can carry data types as key-value pairs called extras.
- Your key is a public constant EXTRA_MESSAGE because the next activity uses the key to retrieve the text value
- Define keys for intent extras using your app's package name as a prefix to make unique keys
- The startActivity() method starts an instance of the DisplayMessageActivity specified by the Intent

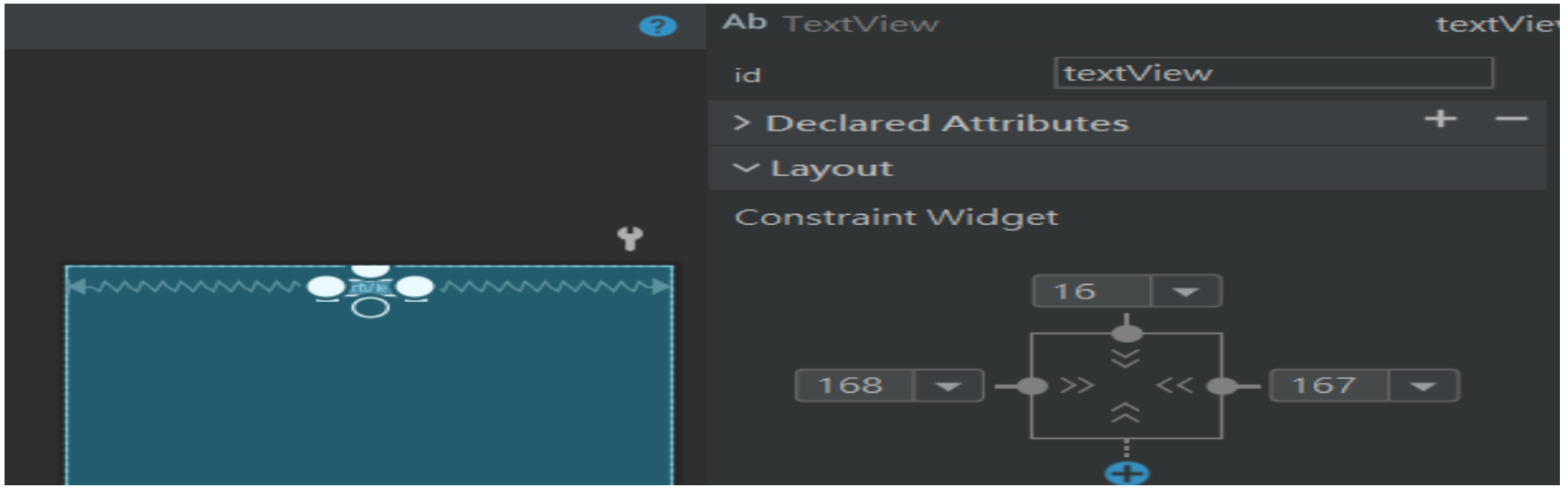
Create DisplayMessageActivity

- In the **Project** window, right-click the **app** folder and select **New > Activity > Empty Activity**.
- In the **Configure Activity** window, enter "DisplayMessageActivity" for **Activity Name** and click **Finish** (leave all other properties set to the defaults).



Add a TextView to DisplayMessageActivity

- Open the file **app > res > layout > activity_display_message.xml**
- In the **Palette** window, click **Text** and then drag a **TextView** into the layout
- Create one more constraint from the top of the text view to the top of the layout, so it appears as shown in figure below



Display the Message

- Add the following code in “DisplayMessageActivity.java”

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_display_message);
    // Get the Intent that started this activity and extract the string
    Intent intent = getIntent();
    String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
    // Capture the layout's TextView and set the string as its text
    TextView textView = findViewById(R.id.textView);
    textView.setText(message);
}
```

Add Navigation

- Open the file at **app > manifests > AndroidManifest.xml**
- Locate the <activity> tag for DisplayMessageActivity, and replace it with the following

```
<activity android:name=".DisplayMessageActivity"
    android:parentActivityName=".MainActivity">
</activity>
```

Final Result

- Click **Run**
- Send “Hello World!” message in your APP!
- Show your result to TA