COMP 90018 Mobile Computing Systems Programming

Tutorial on Android Development

Chu Luo, Eman Bin Khunayn

{chu.luo, eman.bin}@unimelb.edu.au

Welcome!

Outcomes of this tutorial:

1. **Know the platform - Android**
2. **Know how Android Apps work**
3. **Create your first Android app**

1. Android

Android Overview

1. **Open source platform by Google**
2. **Linux kernel**
3. **Currently version 7.1 (updated very fast...)**

Devices of Android Platform

- **A large family of devices run on the Android platform**

HTC One M8

Samsung Smart TV

Pebble SmartWatch

Samsung Galaxy Note 4

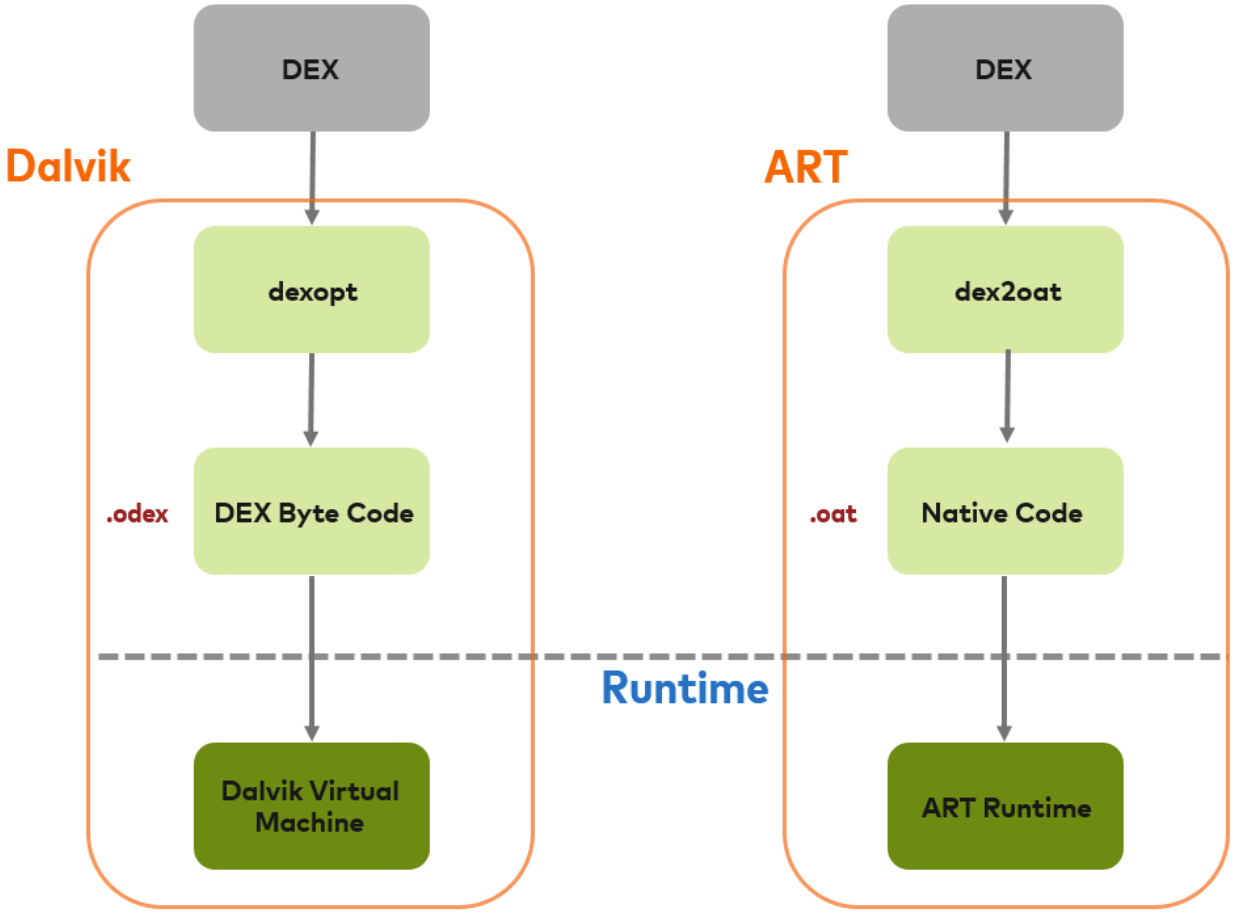
Google Nexus 7 Tablet

Android

2. Android Apps

Android Apps

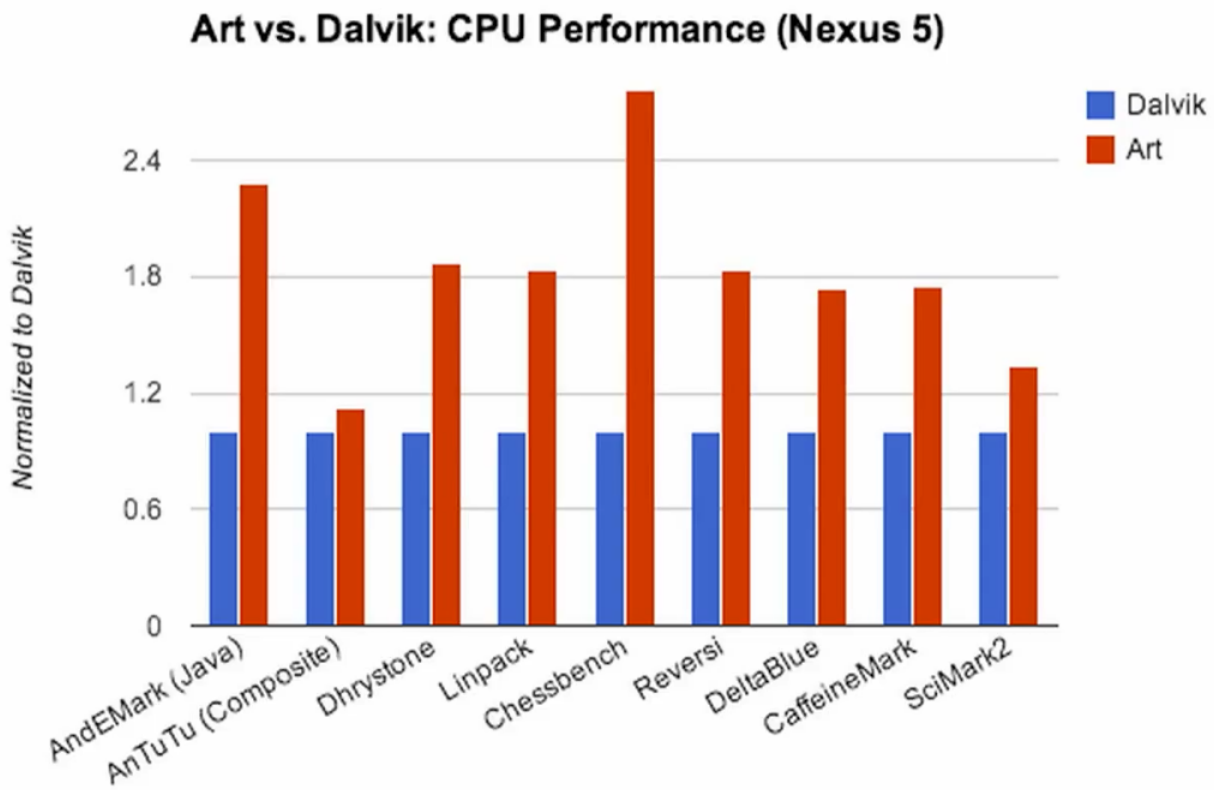
1. **Written in Java**
2. **Runtime Environment : Android Runtime (ART), on Android 5.0+**
3. **Not in Dalvik virtual machine (used before 5.0) any more**



Dalvik vs ART

https://android.jlelse.eu/closer-look-at-android-runtime-dvm-vs-art-1dc5240c3924

Performance Boosting Thing, realized



http://www.anandtech.com/show/8231/a-closer-look-at-android-runtime-art-in-android-l

Four Components in Android Apps

1. **Activity: foreground program**
2. **Service: background task**
3. **Broadcast Receiver: respond to events**
4. **Content Provider: read/write data on phone storage**

3. Create your first Android app

Get the tool creating Android Apps

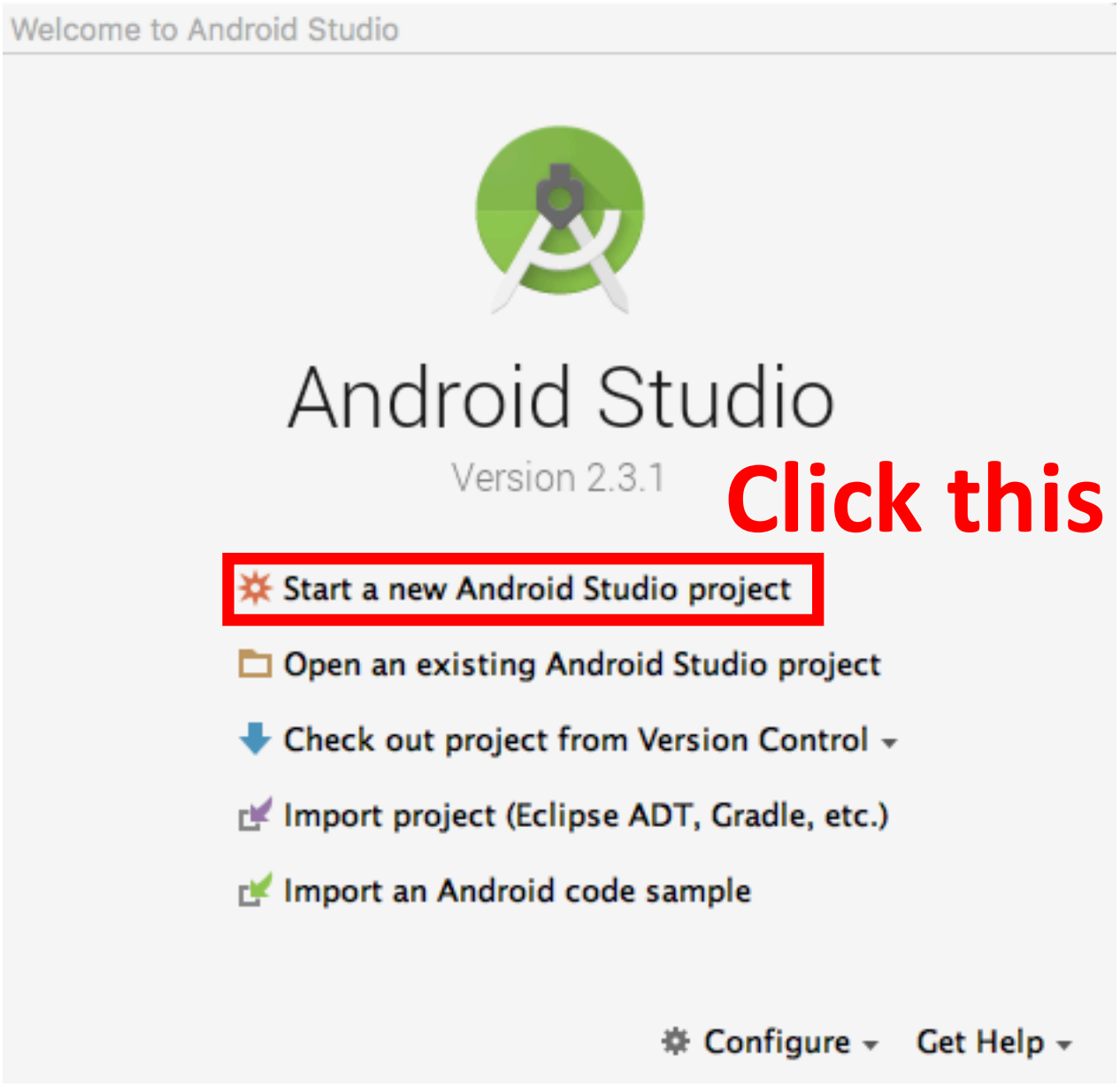
Android Studio: for Win, Mac, Linux

Current version 2.3.3

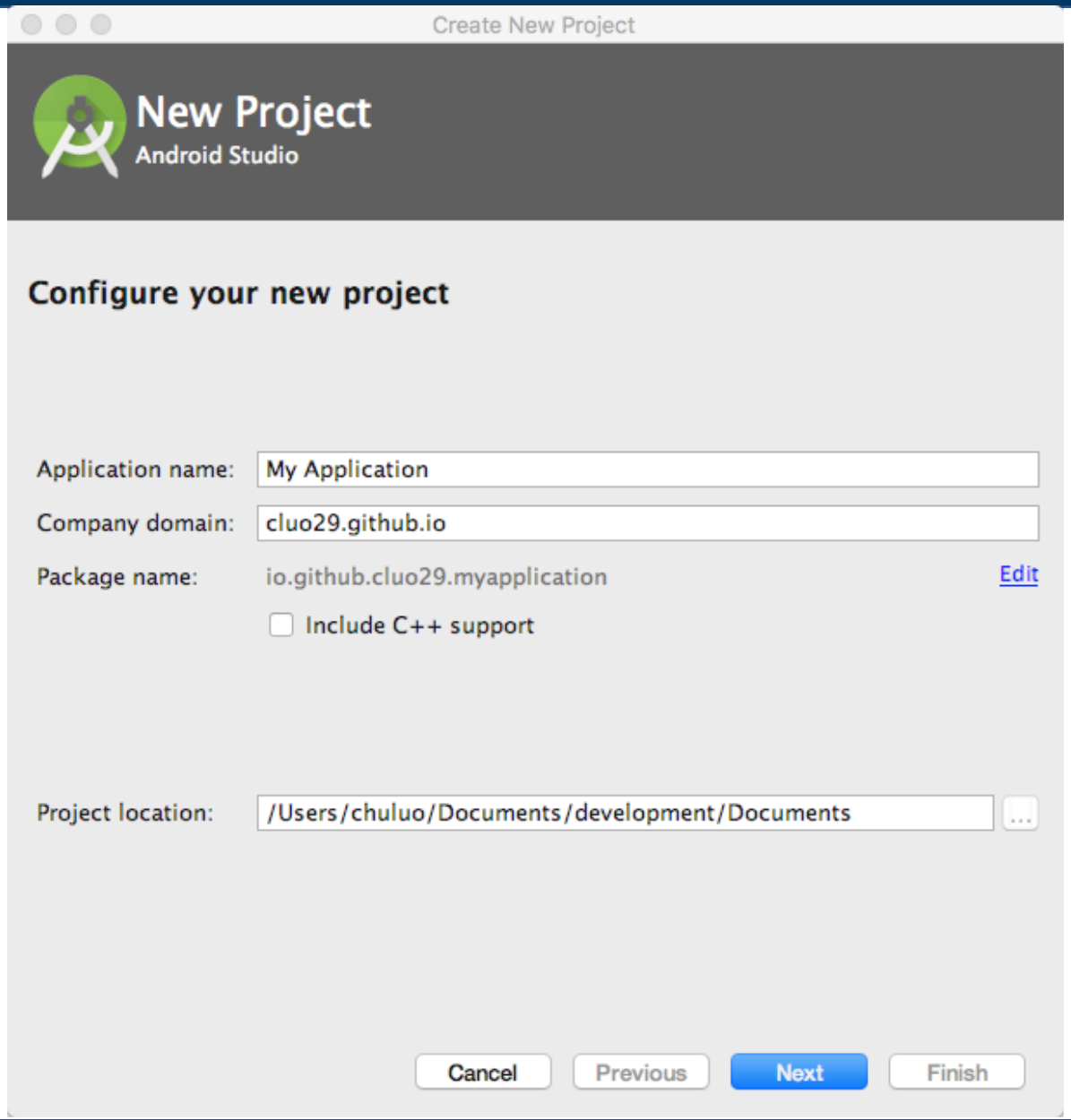
Also install Android SDK

Android Studio will help you install it

Create an app to show Hello World



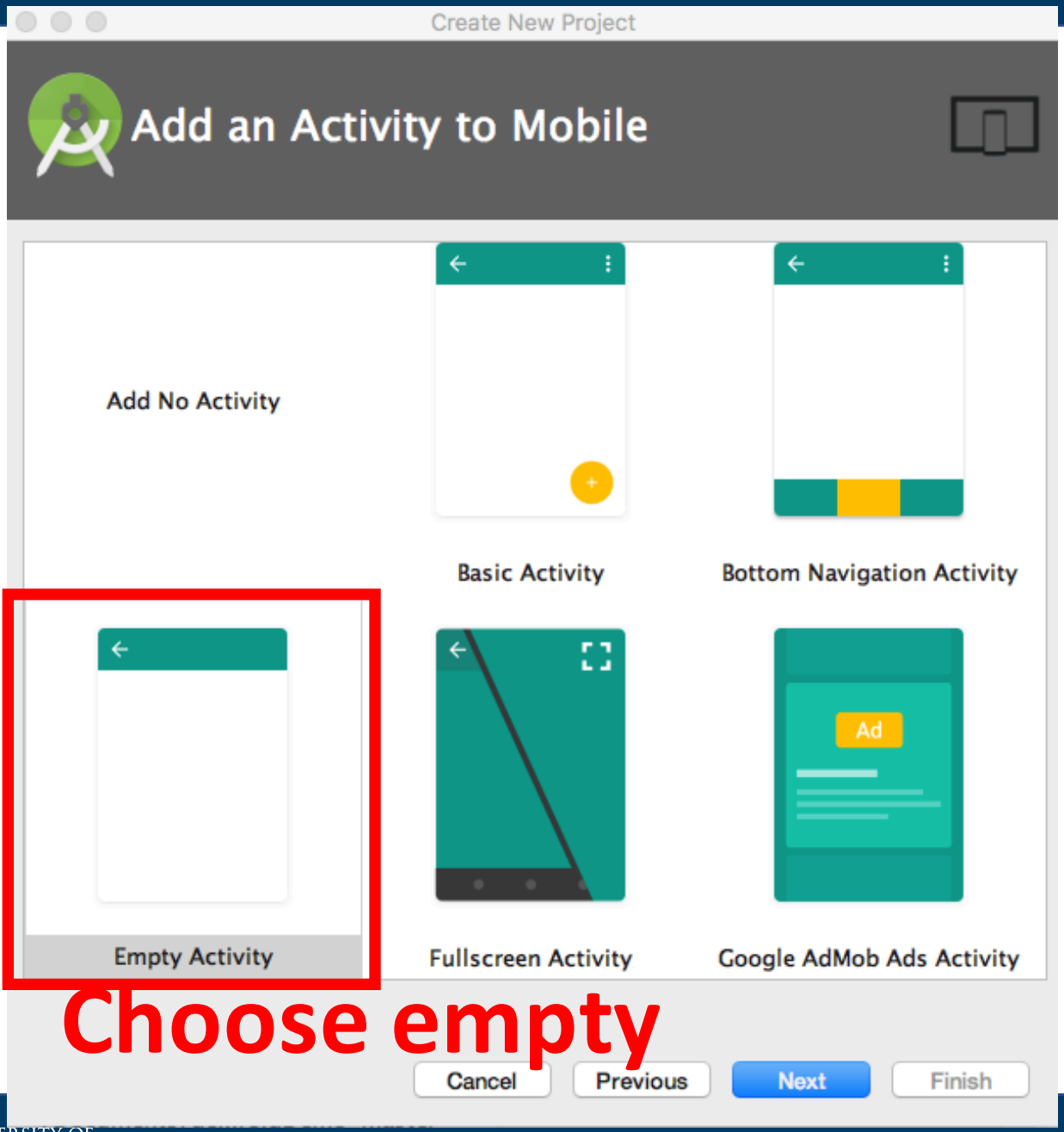
Input app info

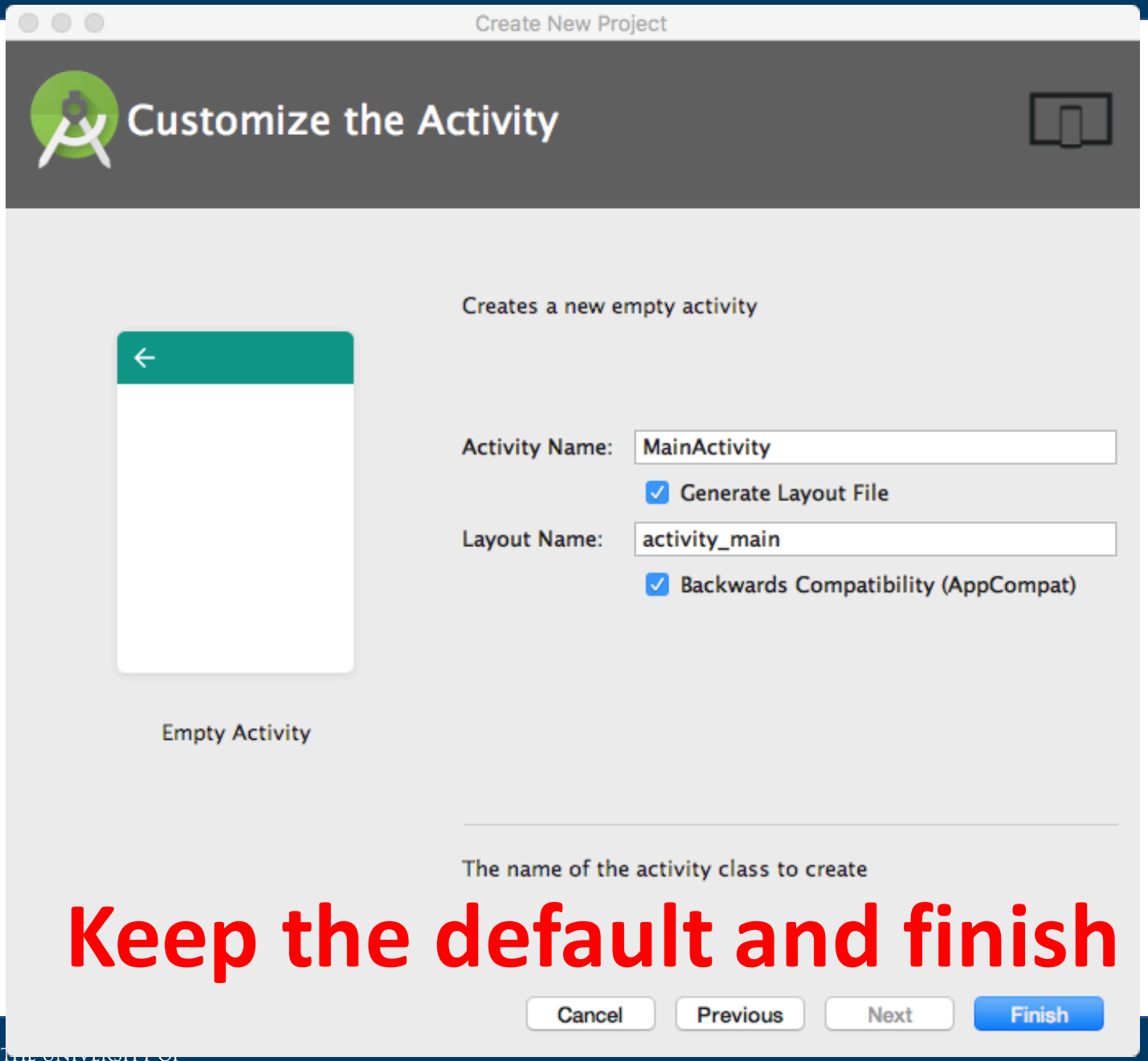


Min API

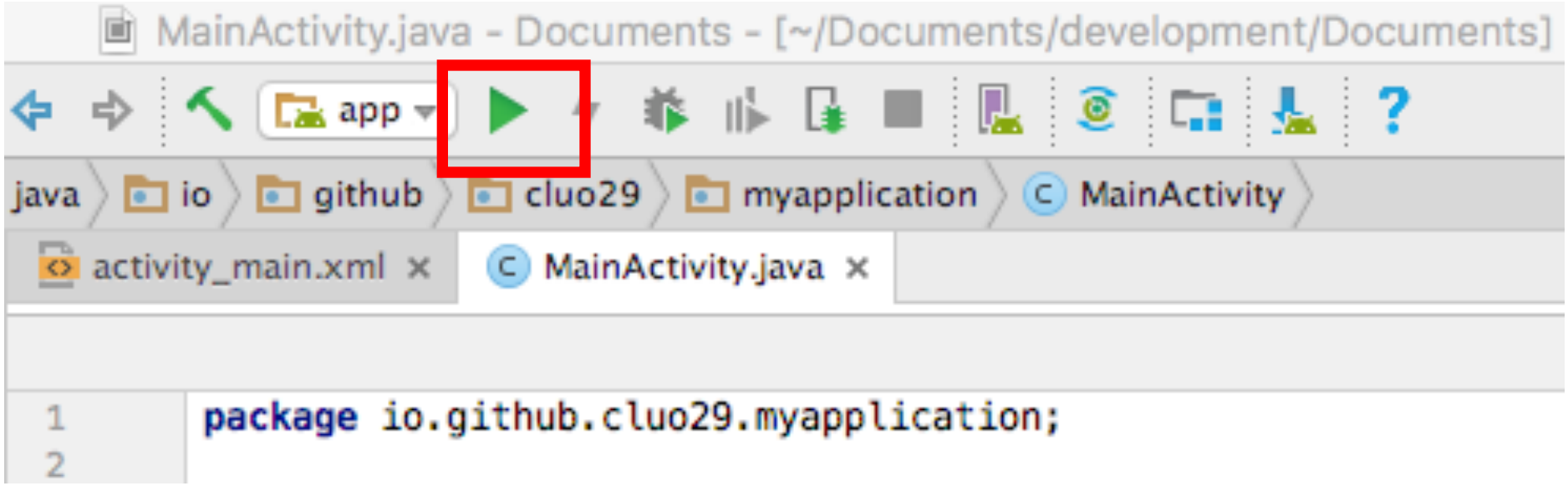

Create New Project
Target Android Devices
Select the form factors your app will run on
Different platforms may require separate SDKs
Must be lower than devices!
Phone and Tablet
Minimum SDK
API 19: Android 4.4 (KitKat)
Lower API levels target more devices, but have fewer features available.
By targeting API 19 and later, your app will run on approximately
90.1% of the devices
that are active on the Google Play Store.
Help me choose
Wear
Minimum SDK
API 21: Android 5.0 (Lollipop)
TV
Minimum SDK
API 21: Android 5.0 (Lollipop)
Android Auto
Cancel
Previous
Next
Finish

Activity


Create New Project
Add an Activity to Mobile
Add No Activity
Basic Activity
Bottom Navigation Activity
Empty Activity
Fullscreen Activity
Ad
Google AdMob Ads Activity
Choose empty
Cancel
Previous
Next
Finish


Create New Project
Customize the Activity
Creates a new empty activity
Activity Name:
MainActivity
Generate Layout File
Layout Name:
activity_main
Backwards Compatibility (AppCompat)
Empty Activity
The name of the activity class to create
Keep the default and finish
Cancel
Previous
Next
Finish

Click this to run the app



E.g., Create an Emulator and run

Homework: Run the app on a real device (phone/tablet)

Hint: <How to Enable USB Debugging Mode on Android>

https://www.kingoapp.com/root-tutorials/how-to-enable-usb-debugging-mode-on-android.htm

Implement your app

Add some Java code in onCreate()

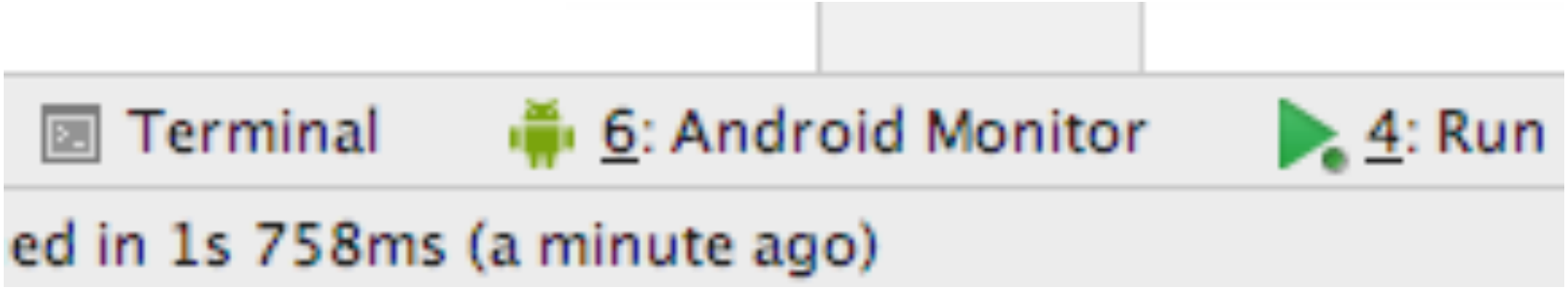
E.g.:

```
int intA = 0;
int intB = 1;
Log.d("MyApp","value of A + B =" +
(intA+intB));

Log.d("MyApp","line number =" + 18);
```

Log.*d*(””,””) for Debugging

In Android Monitor (bottom bar)

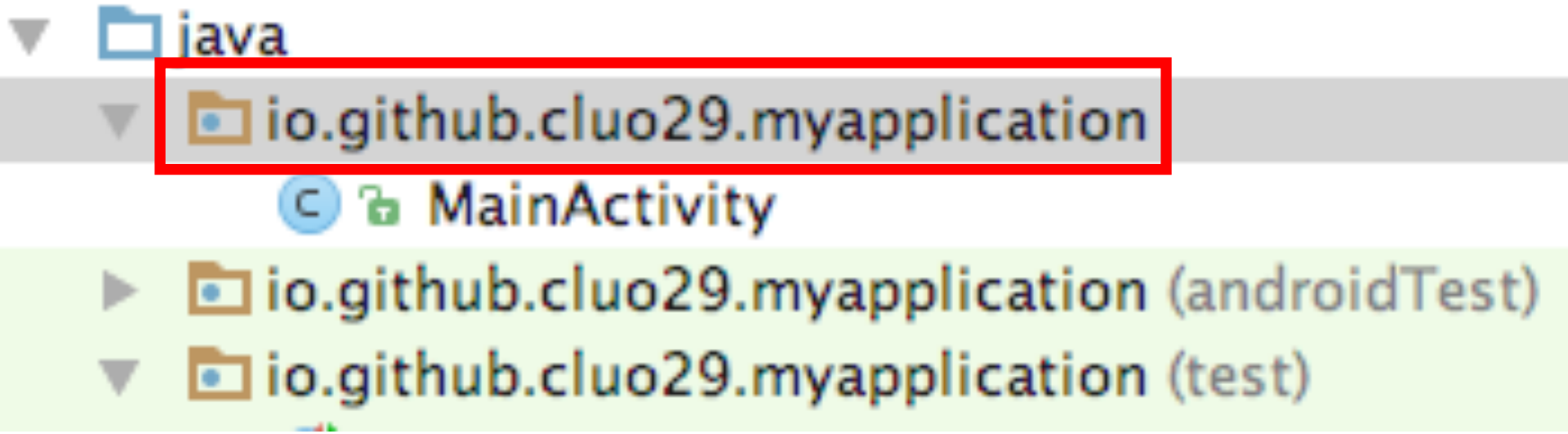


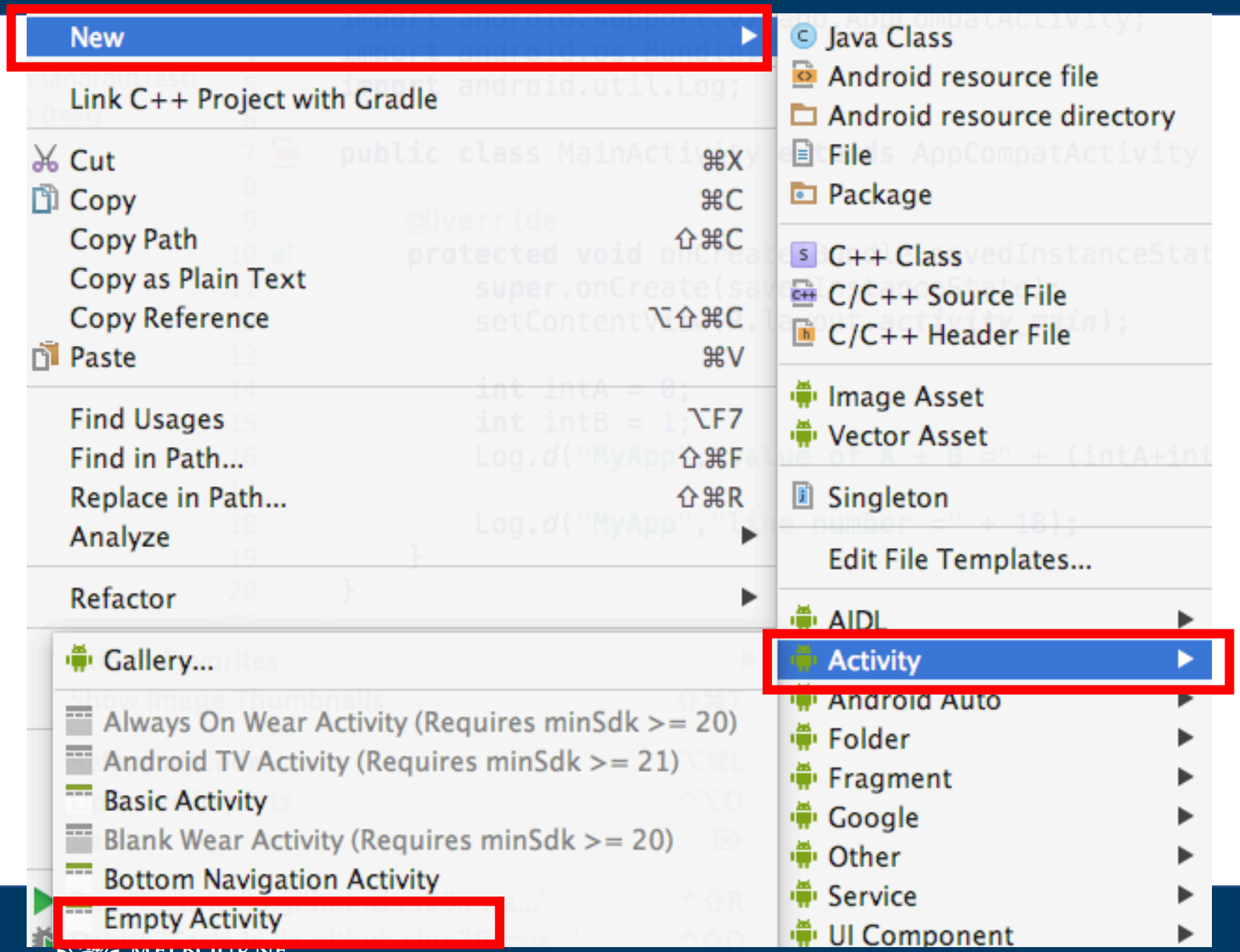
It shows the output:

```
D/MyApp: value of A + B =1
D/MyApp: line number =18
```

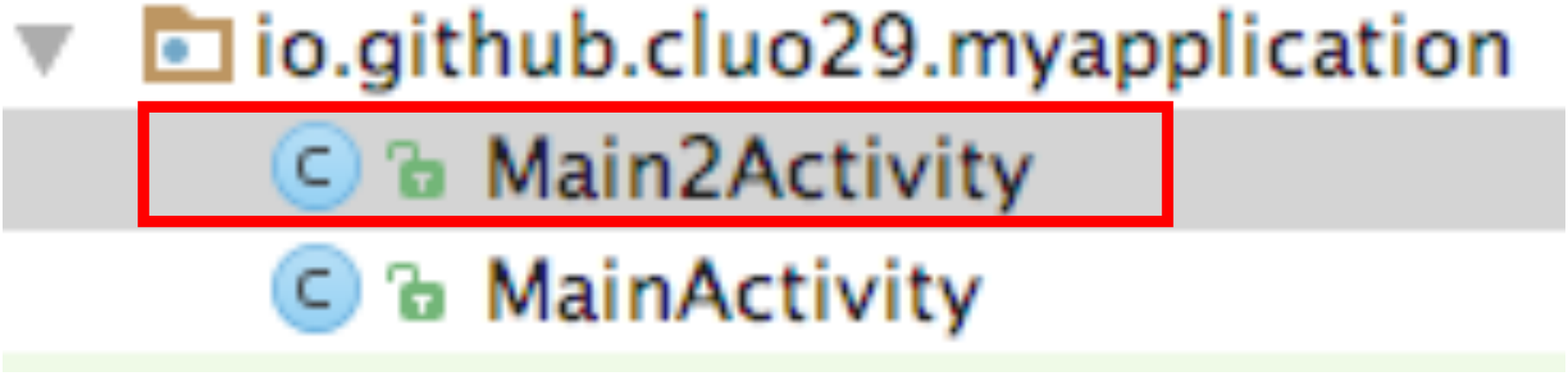
Add more activities?

[Right Click] your project package, then everything is easy




New
Link C++ Project with Gradle
Cut ⌘X
Copy ⌘C
Copy Path ⇧⌘C
Copy as Plain Text
Copy Reference ⌥⇧⌘C
Paste ⌘V
Find Usages ⌥F7
Find in Path... ⇧⌘F
Replace in Path... ⇧⌘R
Analyze
Refactor
Gallery...
Always On Wear Activity (Requires minSdk >= 20)
Android TV Activity (Requires minSdk >= 21)
Basic Activity
Blank Wear Activity (Requires minSdk >= 20)
Bottom Navigation Activity
Empty Activity
Java Class
Android resource file
Android resource directory
File
Package
C++ Class
C/C++ Source File
C/C++ Header File
Image Asset
Vector Asset
Singleton
Edit File Templates...
AIDL
Activity
Android Auto
Folder
Fragment
Google
Other
Service
UI Component

Now you have 2 activities



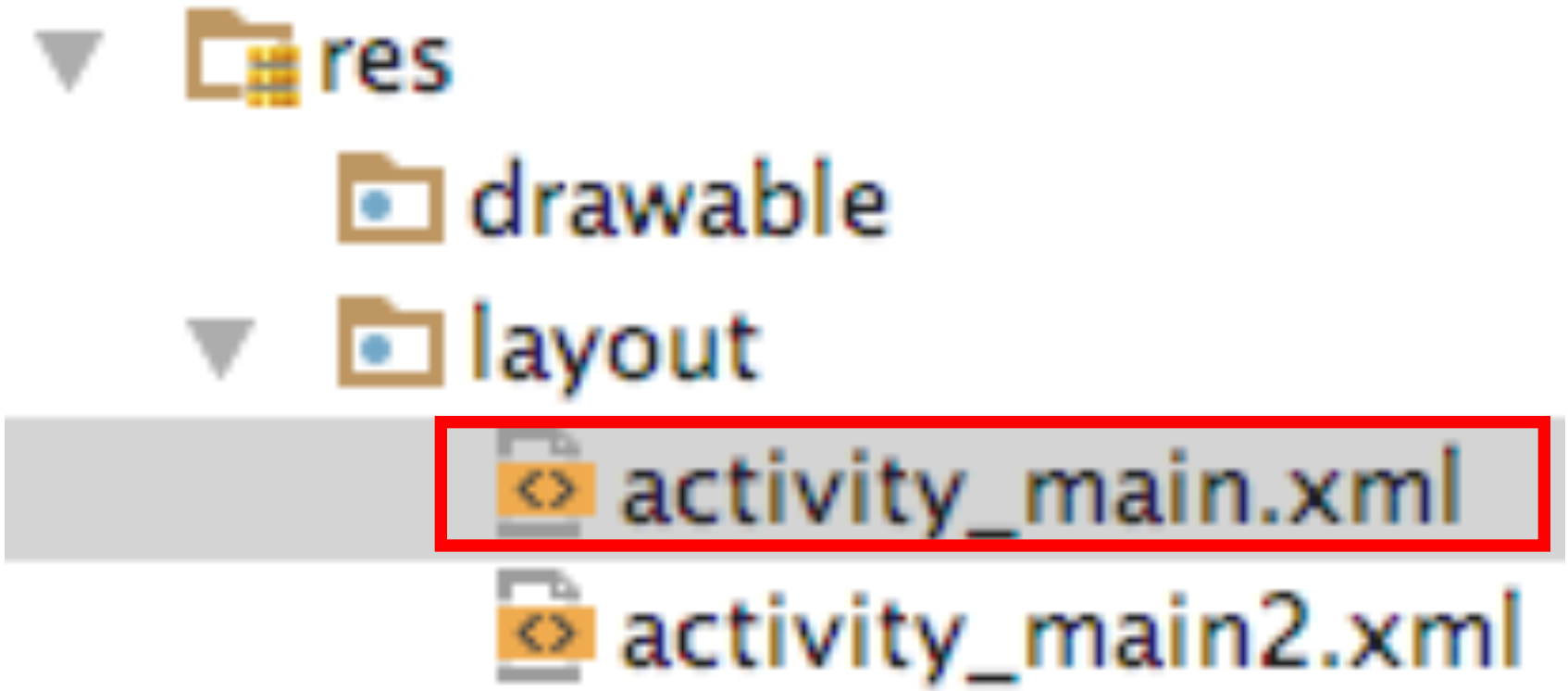
Link them together!

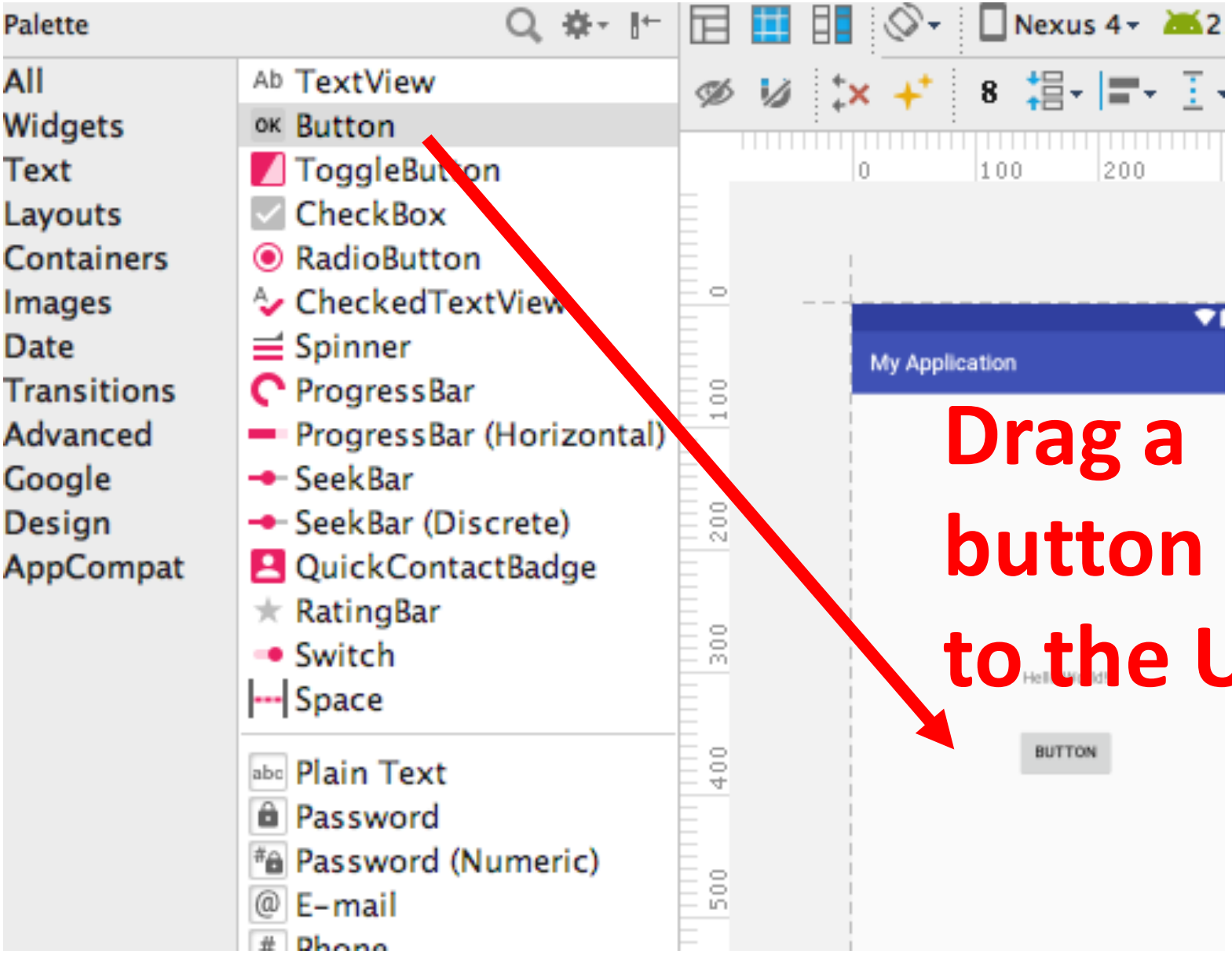
So let Activity 1 have a button.

Press that button to jump to Activity 2

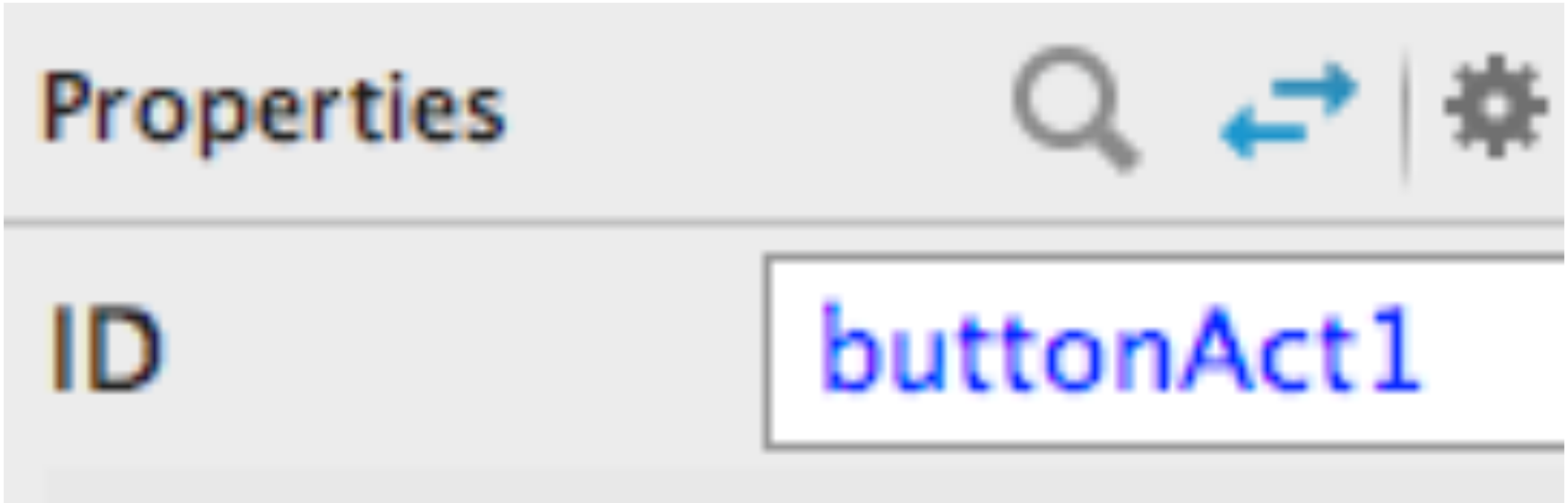
Add a button to the UI of Activity 1

In [project folder], go to res->layout

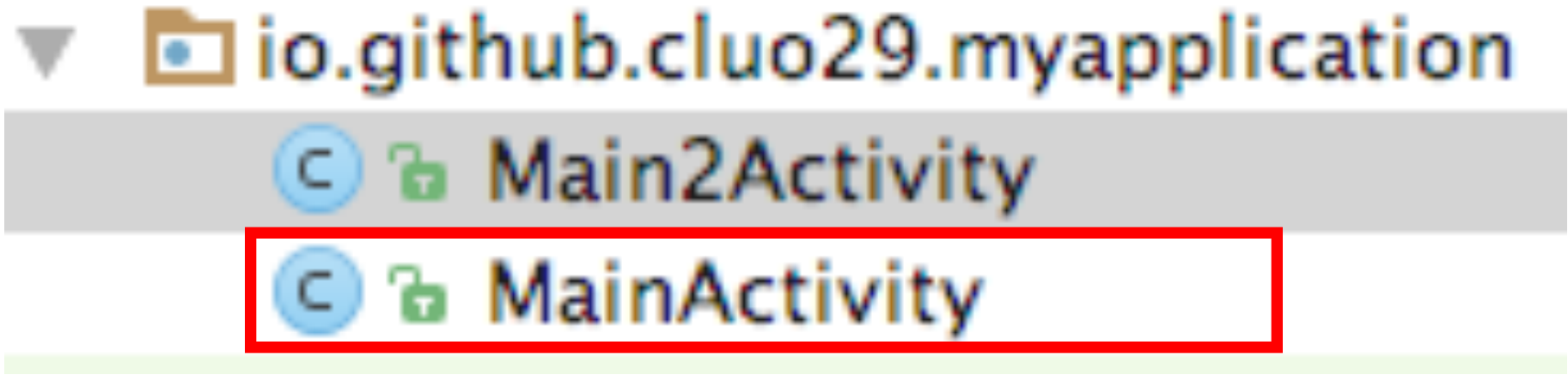



Palette
All
Widgets
Text
Layouts
Containers
Images
Date
Transitions
Advanced
Google
Design
AppCompat
TextView
Button
ToggleButton
CheckBox
RadioButton
CheckedTextView
Spinner
ProgressBar
ProgressBar (Horizontal)
SeekBar
SeekBar (Discrete)
QuickContactBadge
RatingBar
Switch
Space
Plain Text
Password
Password (Numeric)
E-mail
Nexus 4
My Application
BUTTON
Drag a button to the UI

Change the button ID if you want. Make sure every UI item has its [UNIQUE] ID! (also understandable to avoid bugs!)



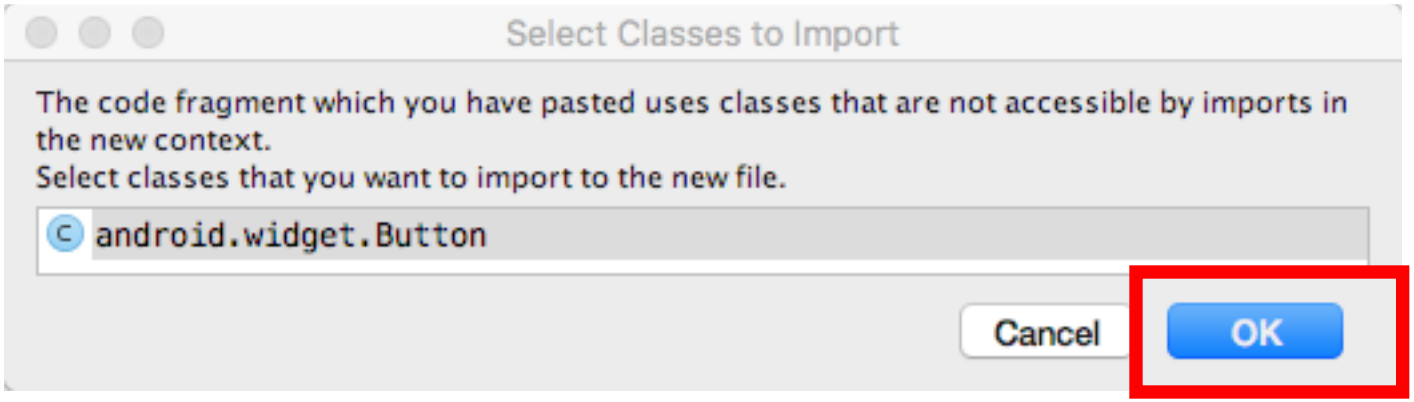
Go back to MainActivity.java



Add the code for the button

```
public class MainActivity exter

    Button button;
```



Android Studio auto import the class

Link the button from UI to logic layer

Do it in onCreate()

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    button = (Button)findViewById(R.id.buttonAct1);
    //intent.putExtra("keyOne","Hello world");
```

(Type) findViewByID(R.id.#itemID#)

Add the code to jump from 1 to 2

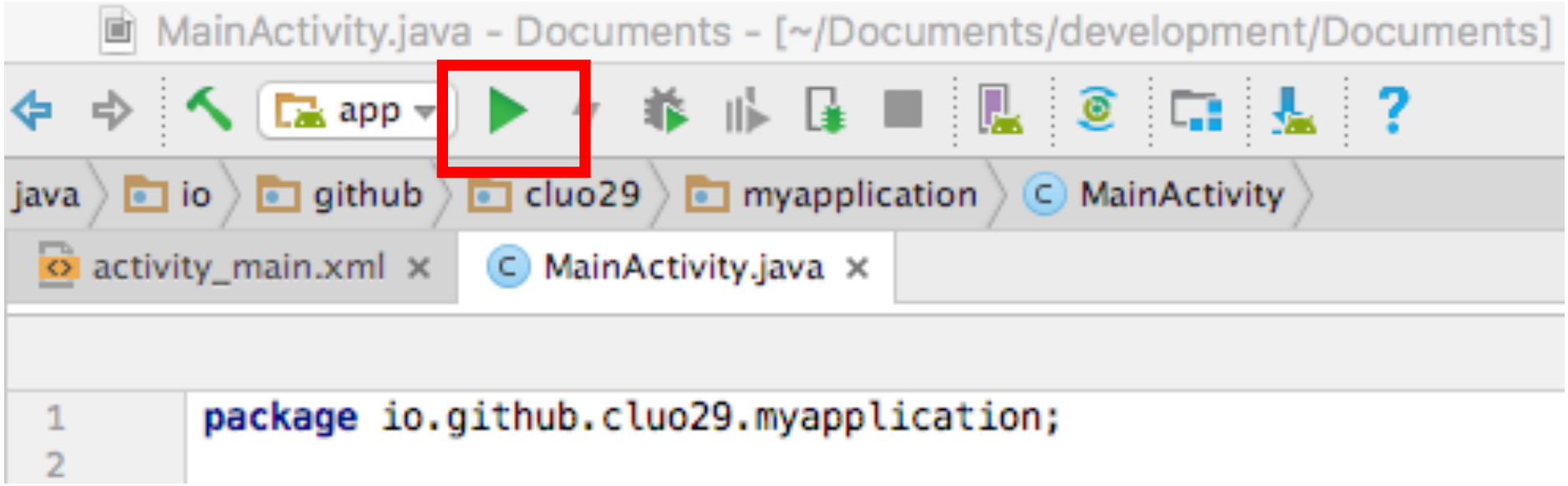
```
button.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // move from ActivityOne to ActivityTwo
        Intent intent = new Intent(MainActivity.this,
                Main2Activity.class);
        startActivity(intent);
    }});
```

To debug, add some code in Activity 2

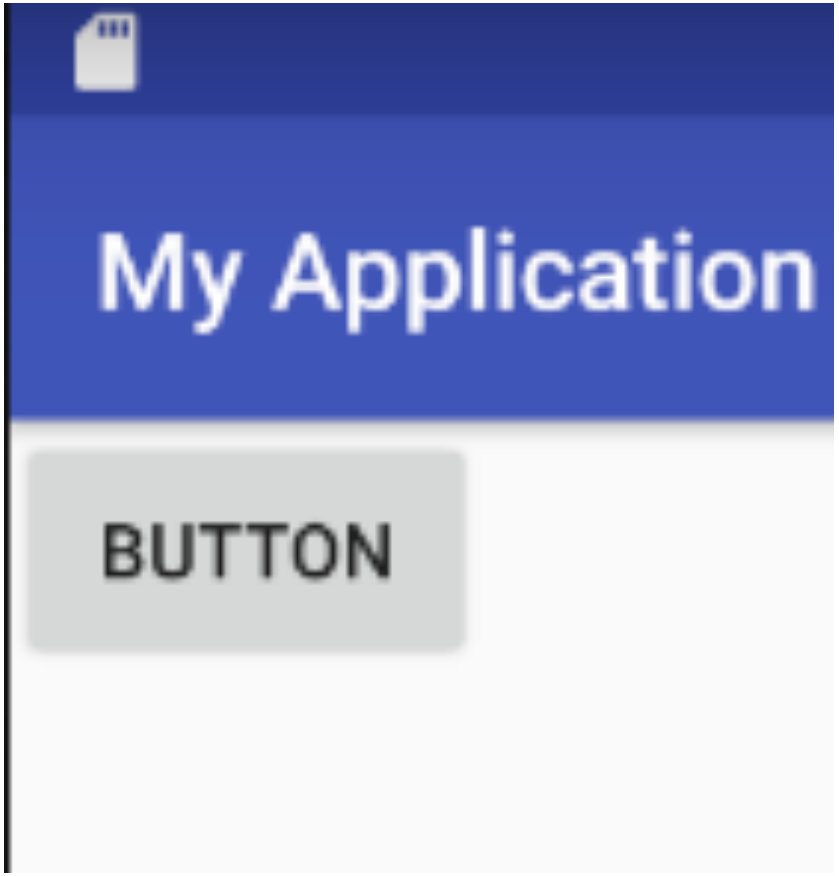
In onCreate() (Activity 2's java, not 1's!)

Add: Log.*d*("**MyApp**","**I am Activity 2**");

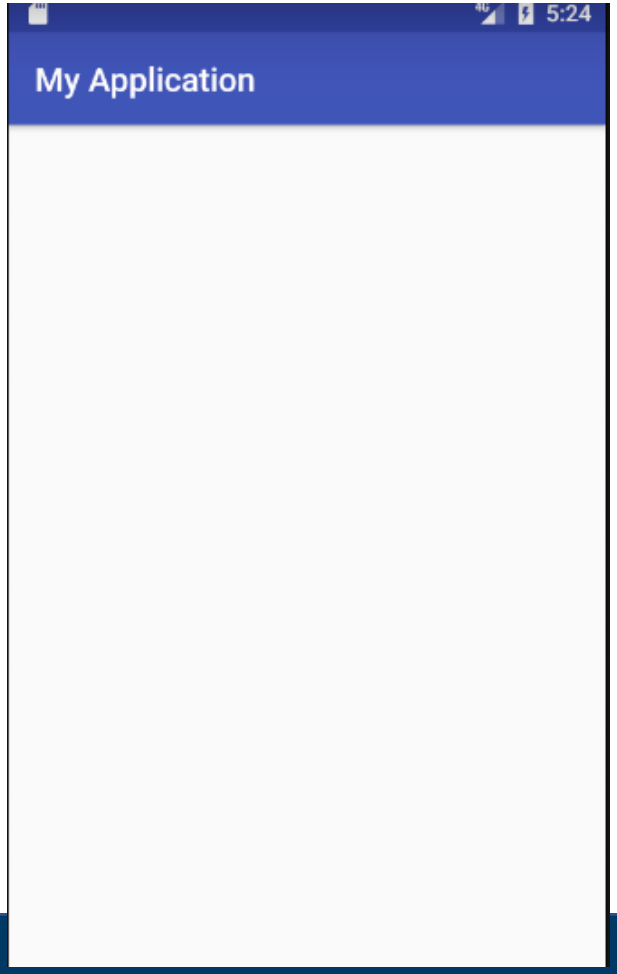
Click this to run the app



Press Button and See the effects



Effect 1: You see the empty UI from Activity 2



Effect 2: You see debugging info

```
D/MyApp: I am Activity 2
```

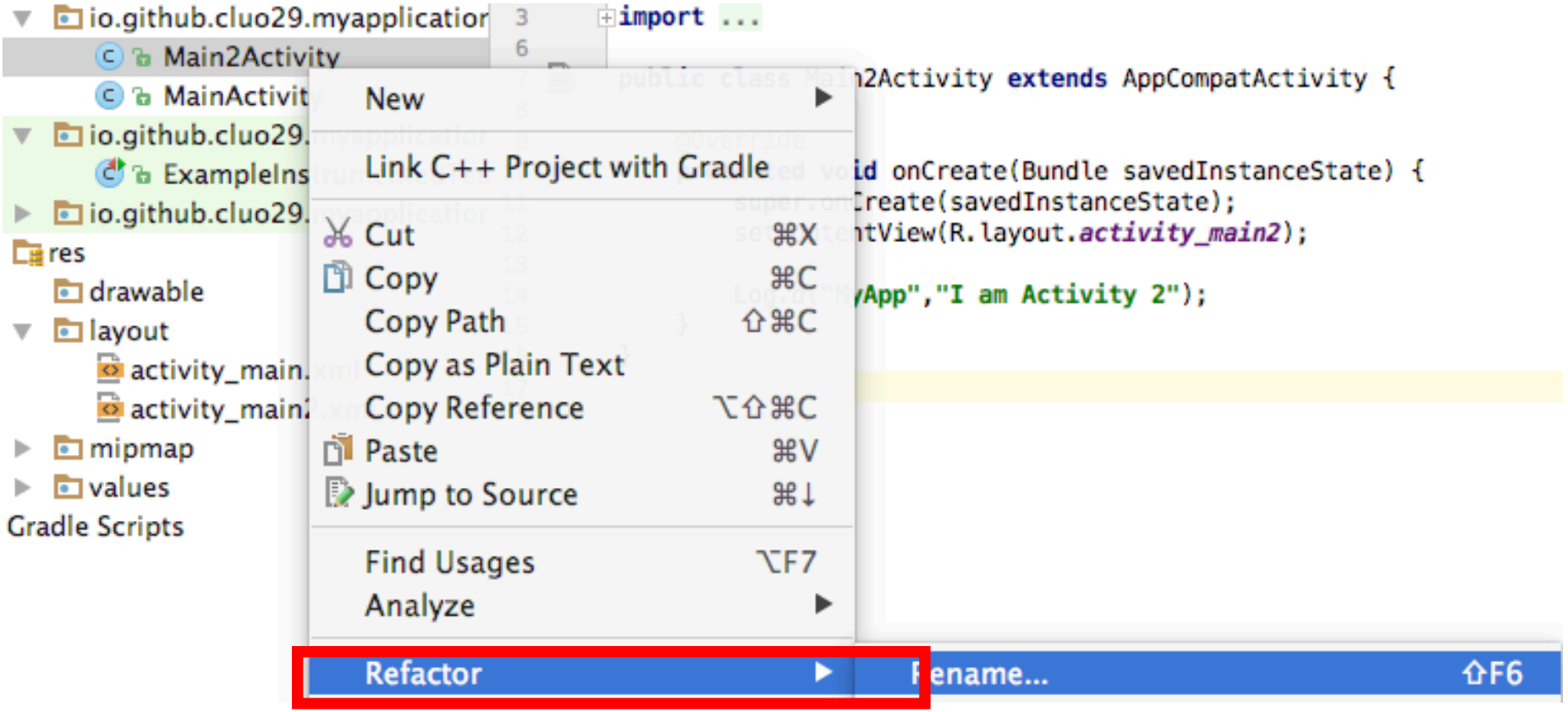
More skills: Refactoring

The code is currently ugly!

But we can improve it!

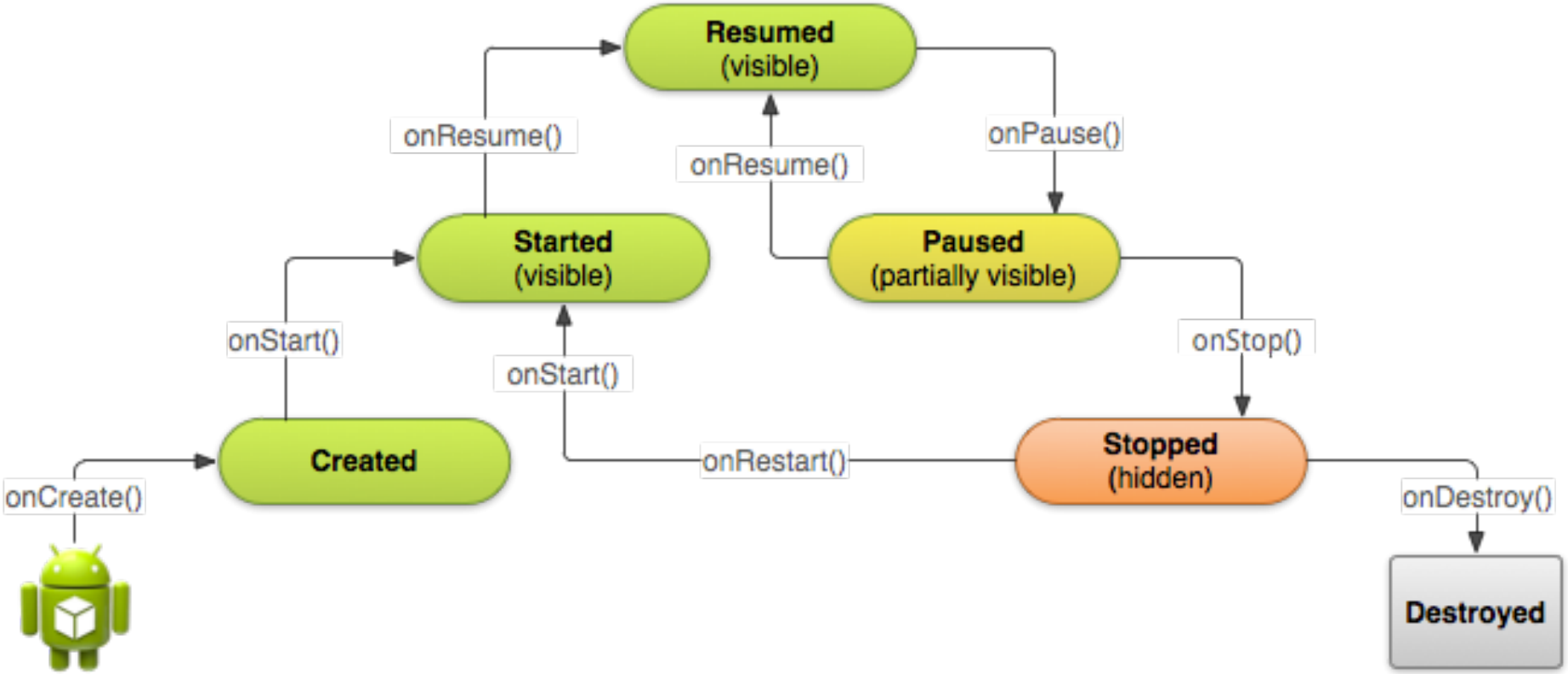
By refactoring!

E.g., Give the activity a better name, Just [Right Click] it and Refactor!



Build better activities

1. Managing the Lifecycle of an Activity.
2. Passing data across Activities.



Visit this link for more details!

/stackoverflow.com/questions/8515936/android-activity-life-cycle-what-are-all-these-methods-for

Passing data across Activities

1. A send data to B when A creates B
2. B send data to A when B is about to die (so that A will be visible)

A send data to B when A creates B

1. Use putExtra() when A creates B
2. Use getStringExtra() (if the data type is String) when B receives the data

B send data to A when B is about to die

1. Use startActivityForResult() when A creates B
2. Use onActivityResult() Receive data from B

Tip: A should use a requestCode to identify which activity sends results

Well done!

Outcomes of this tutorial:

1. **Know the platform - Android**
2. **Know how Android Apps work**
3. **Create your first Android app**

To Build better Android apps

1. Use Log.d() to debug
2. Make code readable (refactor)
3. Manage the Activity Lifecycle
4. Pass data across Activities

When Seeing Errors/Bugs:

1. Read error message, then fix
2. Can't fix? Google it
3. Or, go to Stackoverflow.com
4. If still not good, use LMS discussion board

What About Next Week?

1. **Learn to use – Github (via Android Studio)**
2. **Android UI Design**
3. **Change UI content via Logic Layer**

See you next week

COMP 90018

Tutorial on Android Development

Chu Luo, Eman Bin Khunayn

{chu.luo, eman.bin}@unimelb.edu.au