

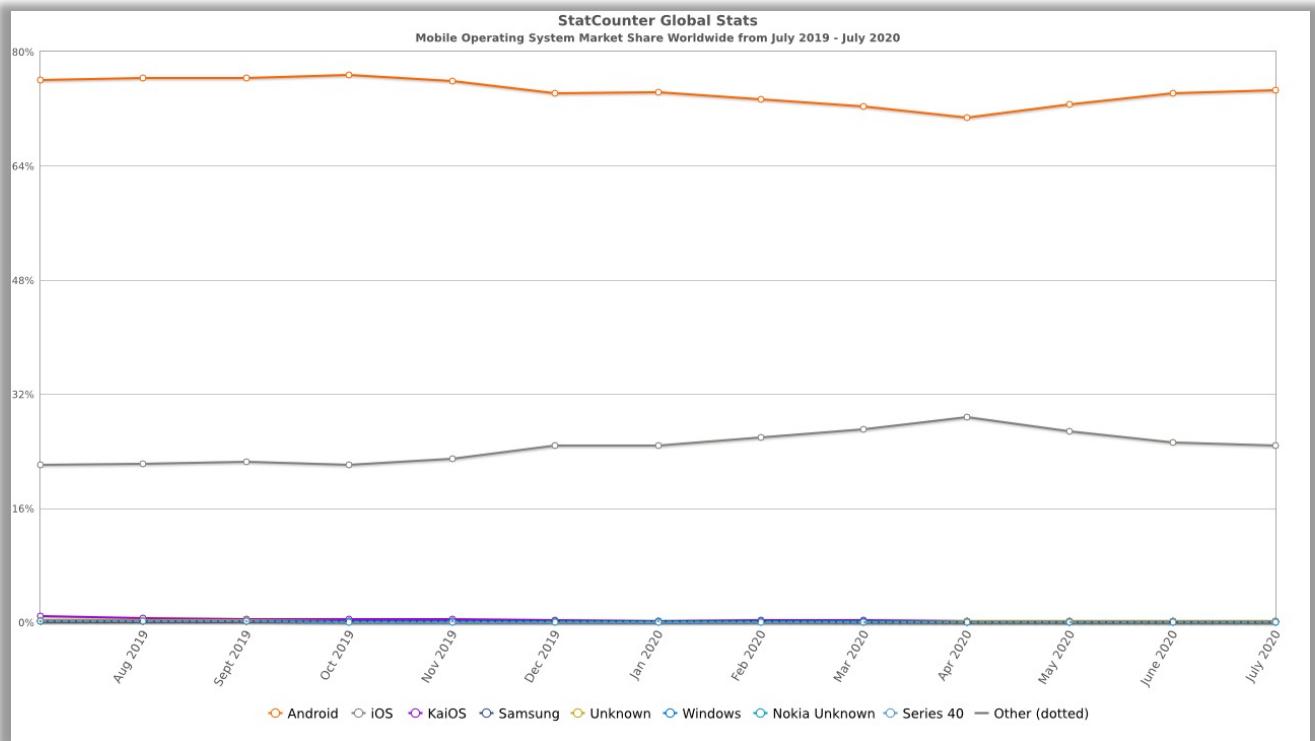
Lecture 4: Introduction to Android Development

Setting up, **creating projects**, **testing apps**

Jianjun Chen (Jianjun.Chen@xjtlu.edu.cn)

Smart Phone OS

- Android
- iOS
- Windows
- Blackberry
- Symbian
- WebOS



<https://gs.statcounter.com/os-market-share/mobile/worldwide>

Android Platform

- Android is a Linux-based operating system designed primarily for touchscreen mobile devices such as smartphones and tablets.
- Android is open source. It can be freely modified and distributed by device manufacturers, wireless carriers and enthusiast developers.
- Android was unveiled in 2007, the first Android phone was sold in October 2008

VPN

- It is recommended to get a VPN for this module.
- Some resources are easier to download with a VPN.

Android Studio and SDK

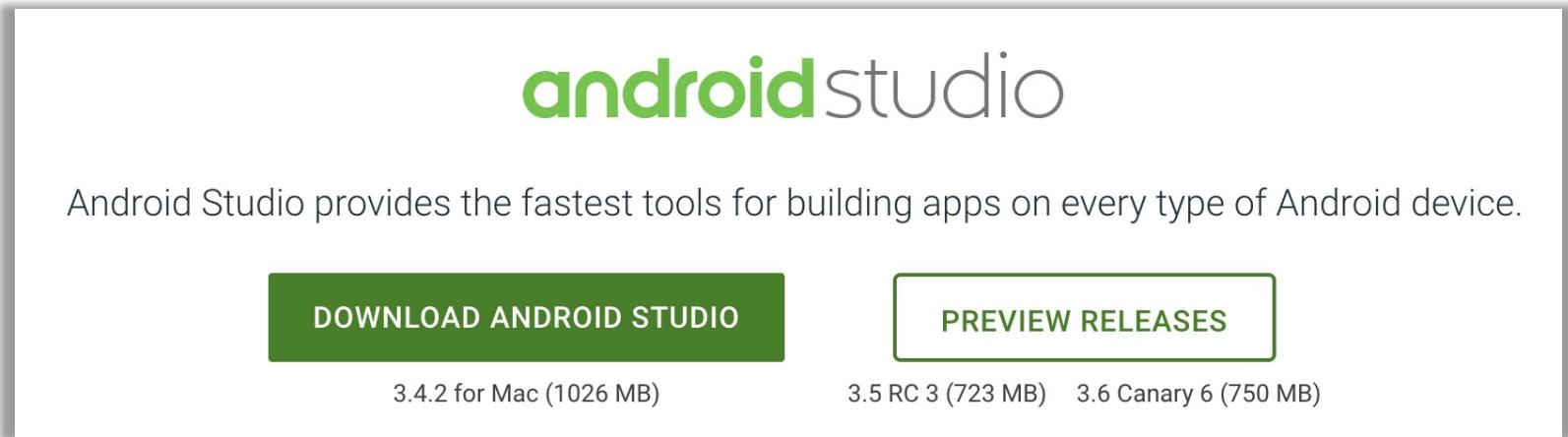
Download source, sdk versions.

Setting up Things

- We will stick to using Android Studio (AS) throughout this module.
 - But Eclipse + ADT is an alternative option.
 - You must install JDK first.
- What will be covered in this lecture?
 - Android Studio set up.
 - The Android SDK.
 - The Gradle build system for AS.
 - A real or virtual Android device.

Getting Android Studio

- Official website:
 - <https://developer.android.com/studio/>
- Alternative website:
 - <https://developer.android.google.cn/studio/>



The image shows a screenshot of the official Android Studio download page. At the top center, the "android studio" logo is displayed. Below it, a subtext states: "Android Studio provides the fastest tools for building apps on every type of Android device." Two main buttons are present: a green rectangular button on the left labeled "DOWNLOAD ANDROID STUDIO" and a white rectangular button on the right labeled "PREVIEW RELEASES". Below these buttons, three download links are listed: "3.4.2 for Mac (1026 MB)", "3.5 RC 3 (723 MB)", and "3.6 Canary 6 (750 MB)".

android studio

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

DOWNLOAD ANDROID STUDIO

PREVIEW RELEASES

3.4.2 for Mac (1026 MB) 3.5 RC 3 (723 MB) 3.6 Canary 6 (750 MB)

Android SDK and API level

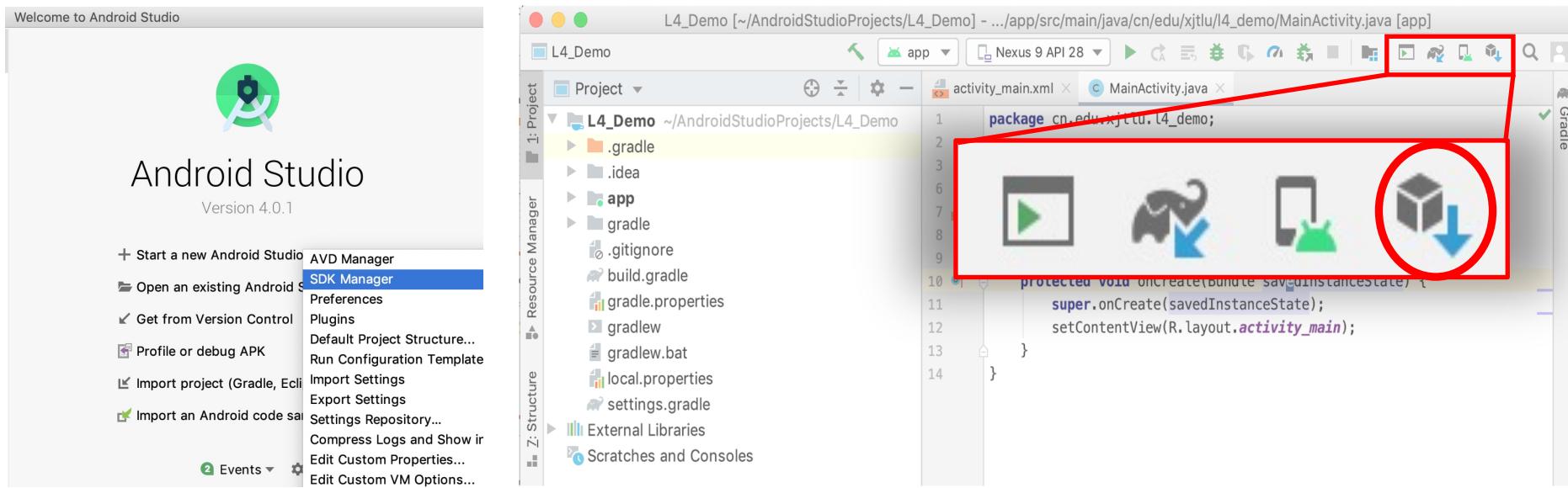
- The API decides how you call functions to access the service from the Android system. API is usually downward compatible.
 - E.g. Android 9 can run Android 4.X apps.
 - An app that requires higher version of Android API will run on **less** devices.
- SDK: A set of tools that can provide different API libraries and developer tools.

Android Versions and API Level

Codename	Version	API level/NDK release
Android10	10	API level 29
Pie	9	API level 28
Oreo	8.1.0	API level 27
Oreo	8.0.0	API level 26
Nougat	7.1	API level 25
Nougat	7.0	API level 24
Marshmallow	6.0	API level 23
Lollipop	5.1	API level 22
Lollipop	5.0	API level 21
KitKat	4.4 - 4.4.4	API level 19
Jelly Bean	4.3.x	API level 18
Jelly Bean	4.2.x	API level 17

Android SDK and API level

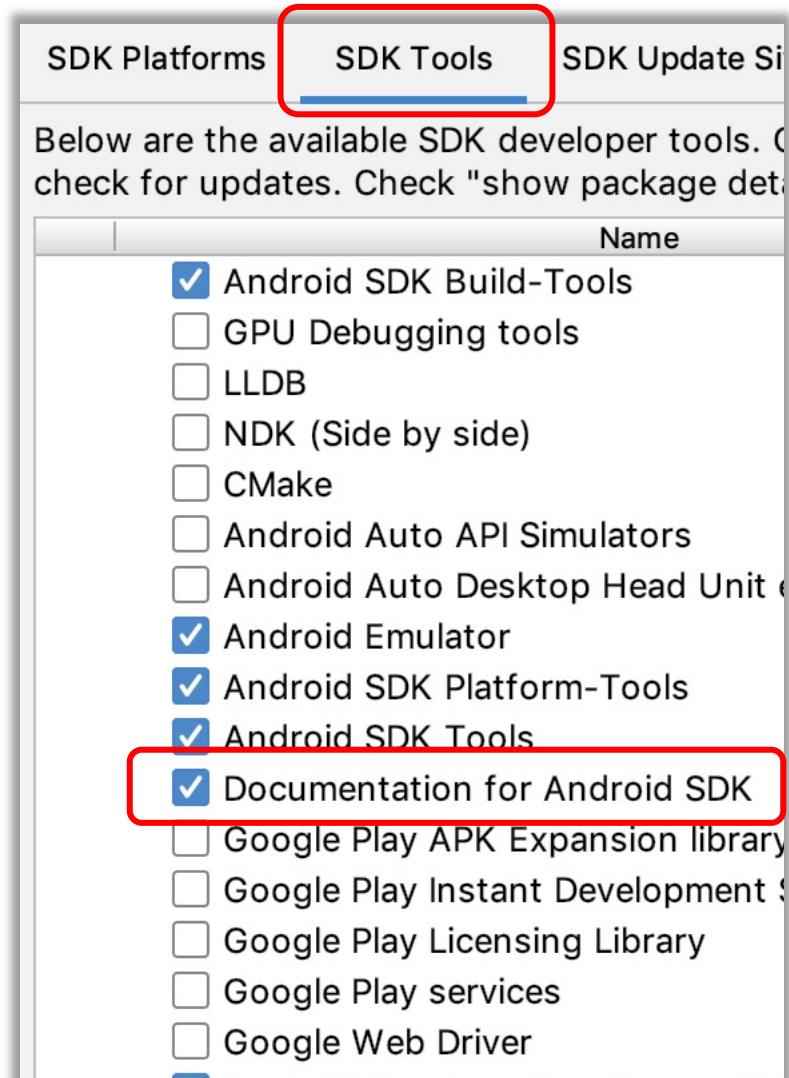
- AS does not come with SDKs pre-installed.
 - Access the SDK manager at the welcome screen
 - Menu->Tools->SDK manager
 - Also available as a button on the toolbar:



Documentation for SDK

- While the SDK manager is still open, do remember to download the documentation for Android SDK
- This enables you to easily find definitions of functions and variables.
- Once done, you can check the documentation by selecting a function or variable identifier, then:
 - Windows: Ctrl + q
 - MacOS: F1
 - Linux: ??

You might need to use VPN here if download fails



Creating a Project

- Start your Android Studio
 - Start a new Android Studio project
 - Assign an application name and a project location.
 - Select an API version for your app
 - Select “Empty Activity”
 - Confirm the activity name.

Configure your project

Name

TestApp

Package name

cn.edu.xjtlu.testapp

Save location

C:\Users\jianjun.chen\AndroidStudioProjects\TestApp

Language

Java

Minimum API level

API 18: Android 4.3 (Jelly Bean)

Empty Activity

Looking for SDKs available for download...

Your app will run on approximately **97.1%** of devices.[Help me choose](#) This project will support instant apps Use androidx.* artifacts

Creates a new empty activity

[Previous](#)[Next](#)[Cancel](#)[Finish](#)

TestApp [C:\Users\Jianjun\AndroidStudioProjects\TestApp] - ...\\app\\src\\main\\java\\cn\\edu\\xjtlu\\testapp\\... - X

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

TestApp app app MainActivity.java

1 package cn.edu.xjtlu.testapp;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7 @Override
8 protected void onCreate(Bundle savedInstanceState) {
9 super.onCreate(savedInstanceState);
10 setContentView(R.layout.activity_main);
11 }
12}
13
14 }
15

Gradle

1: Project

2: Favorites

Resource Manager

3: Structure

4: Favorites

5: Structure

6: Favorites

7: Favorites

8: Favorites

9: Favorites

10: Favorites

11: Favorites

12: Favorites

13: Favorites

14: Favorites

15: Favorites

This sync process starts automatically, you must wait until all tasks are finished

Build: Build Output Sync

TestApp: synced successfully at 2019/9/8 3:01

Run build C:\Users\Jianjun\AndroidStudioProjects\TestApp

Load build

Configure build

Calculate task graph

Run tasks

Device File Explorer

1 s 519 ms
928 ms
79 ms
693 ms
15 ms
136 ms

TODO Terminal Build Logcat Profiler Run Event Log

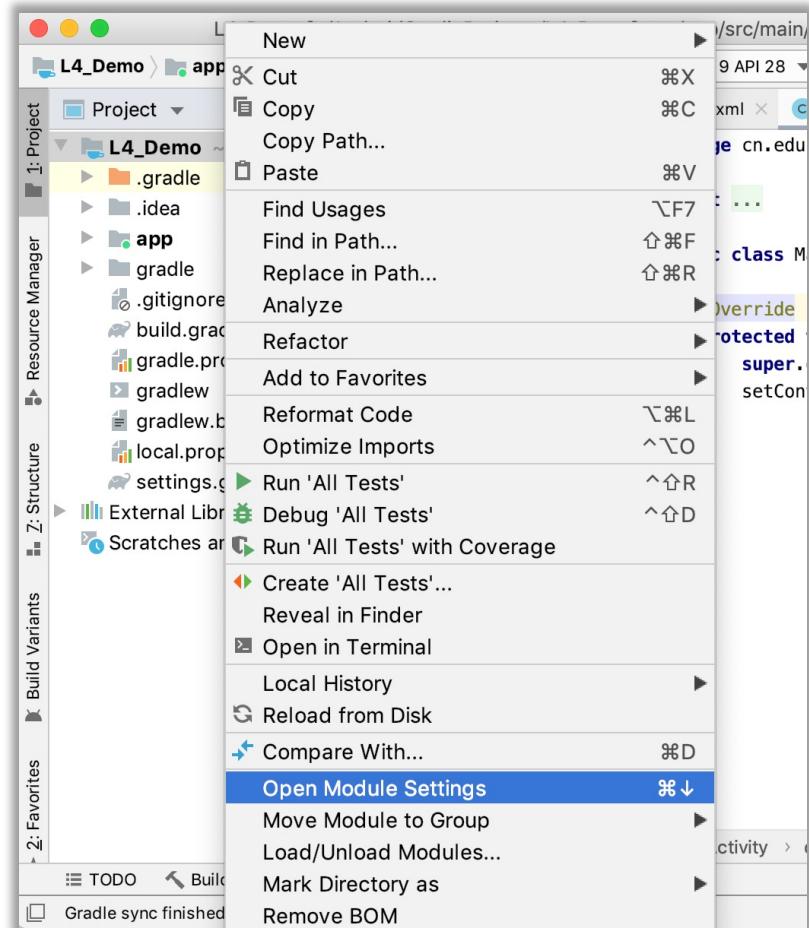
Syncing only active variant // You can disable this experimental fe... (a minute ago) 15:1 CRLF UTF-8 4 spaces

Gradle Setup

Sync issues and solution without a VPN

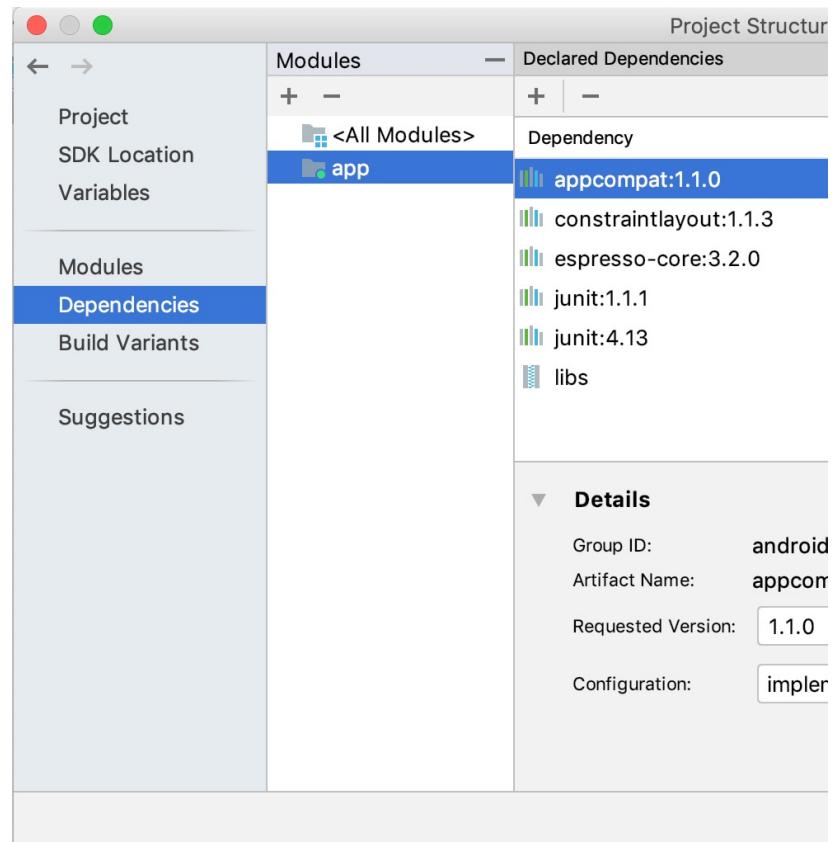
Gradle Sync Issues

- If you right click on the android project, and select “open module settings”, you will see all dependent modules of your project



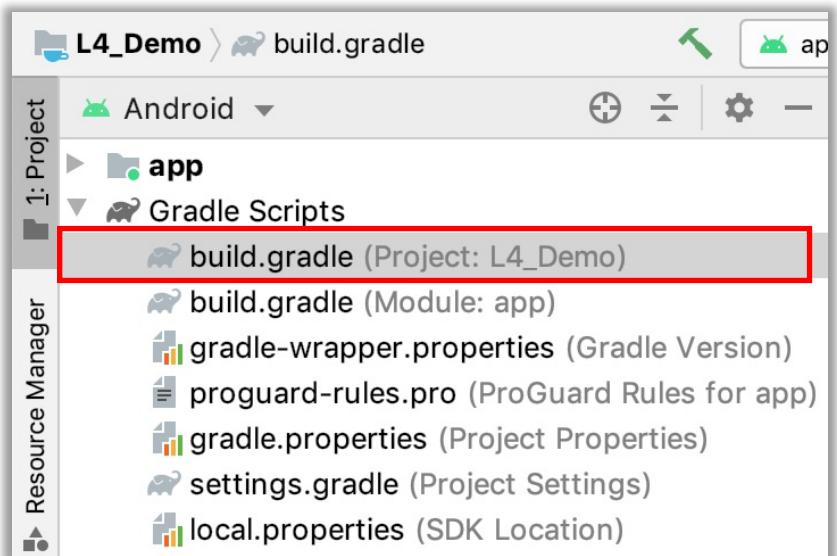
Gradle Sync Issues

- As shown on the right, some modules comes from online repository.
- This is managed by our build system: Gradle.



Gradle Sync Issues

- The online repository is defined in the main Gradle script
 - By default, it connects to Google's server.
- If your sync fails because connection to google fails, you should use the method in the next few slides.



Solving Gradle Sync Issues

1. In “buildscript” section, “repositories” subsection, replace the original content with:

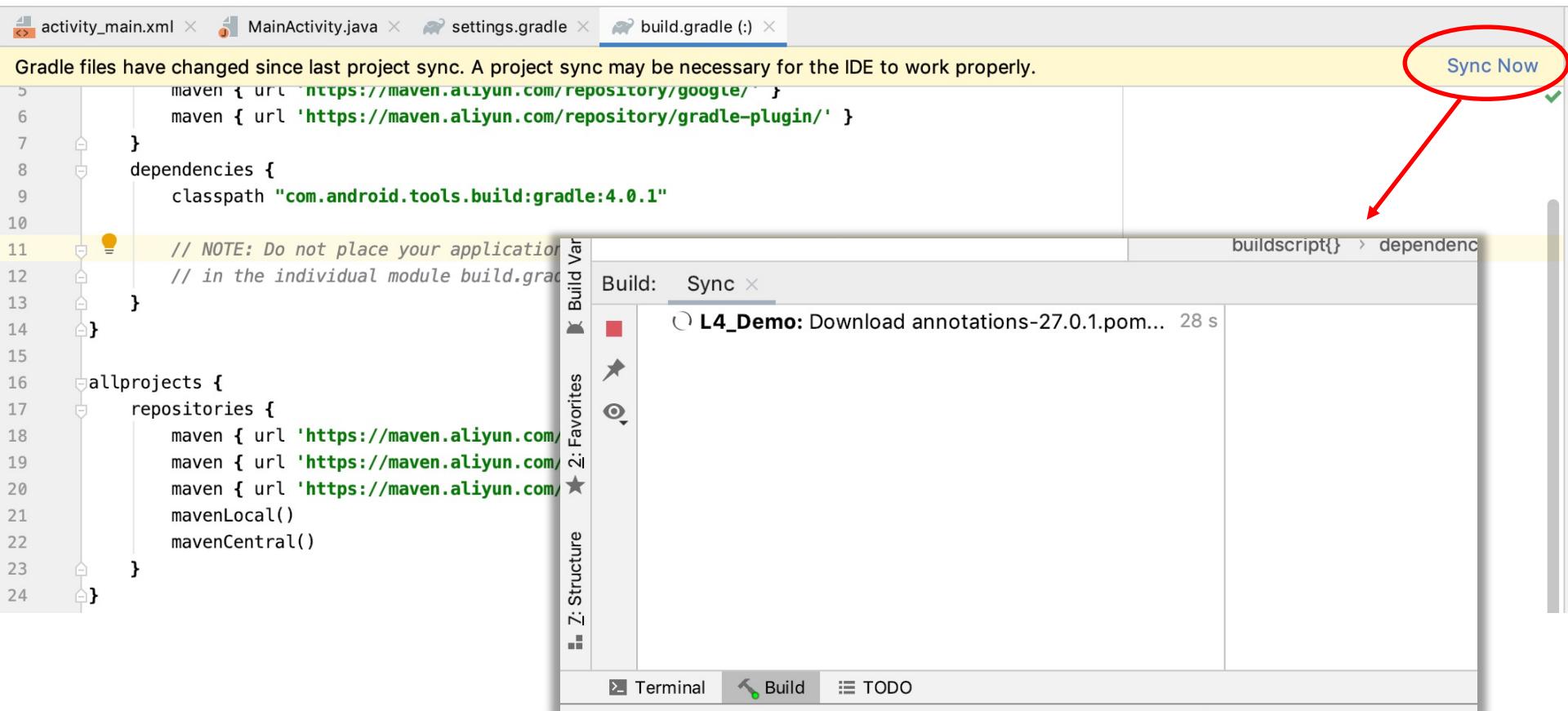
```
maven { url 'https://maven.aliyun.com/repository/public/' }
maven { url 'https://maven.aliyun.com/repository/google/' }
maven { url 'https://maven.aliyun.com/repository/gradle-plugin/' }
```

2. In “allprojects” section, “repositories” subsection, replace the original content with:

```
maven { url 'https://maven.aliyun.com/repository/public/' }
maven { url 'https://maven.aliyun.com/repository/google/' }
maven { url 'https://maven.aliyun.com/repository/gradle-plugin/' }
mavenLocal()
mavenCentral()
```

Gradle Sync Issues

- Once done, save and then resync.

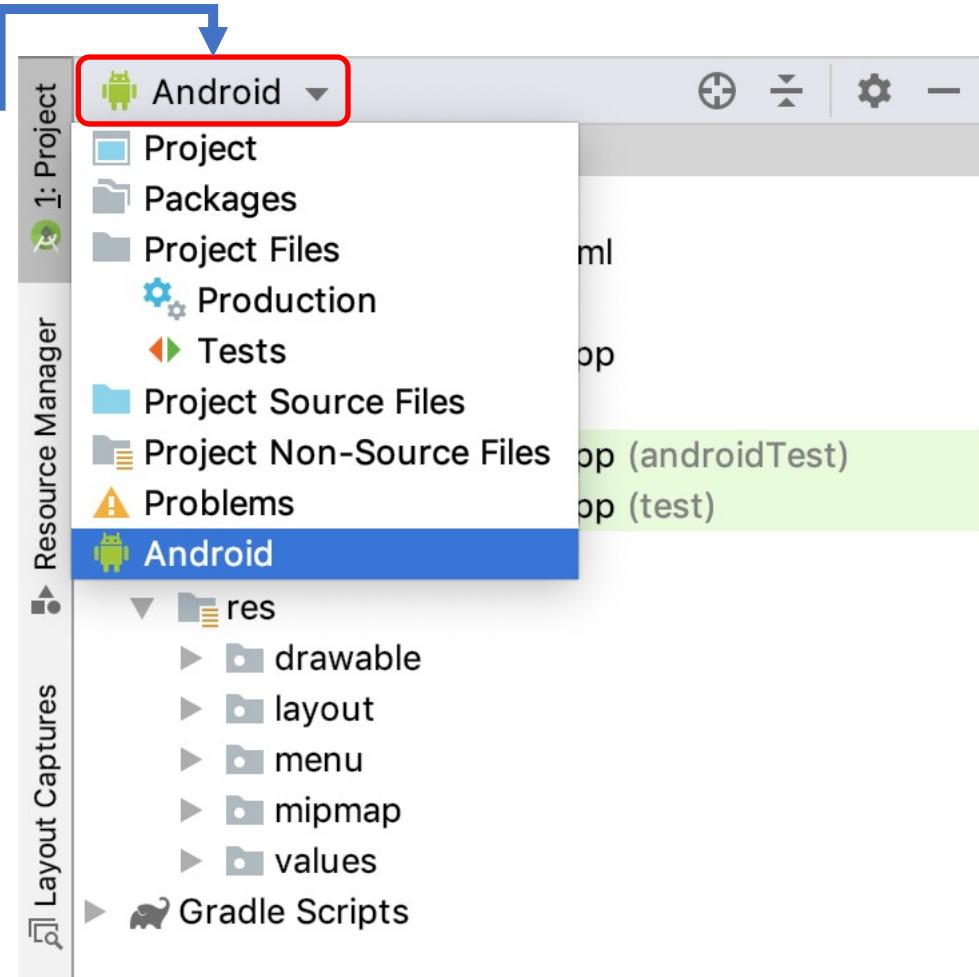


IDE Interfaces and Project File Structure

Project folders, layout files

Project Window

- There are **several view options:**
 - “Android” view is easier to understand.
 - “Project” view matches the folder structure in the file system (as seen in your file explorer)



Project Folders

- **Manifests/AndroidManifest.xml:** Basic information of the app, such as app name, its activities, permissions required etc..
- **Java:** Your source code and test code.
- **res:** Various resources used by your app
 - **mipmap:** App icons.
 - **drawable:** Images (bitmaps, jpegs, pngs) used in your app.
 - **values:** store the values for the resources, such as color, styles, dimensions etc..
 - **layout:** UI layouts.
- **Gradle scripts:** Settings and scripts for the build system, such as the location for SDK.

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project structure under `cn.edu.xjtu.testapp`. It includes:
 - `generatedJava`
 - `assets`
 - res**:
 - `drawable`
 - `layout`
 - mipmap**:
 - ic_launcher (6)**: Contains files for various screen densities: `ic_launcher.png (hdpi)`, `ic_launcher.png (mdpi)`, `ic_launcher.png (xhdpi)`, `ic_launcher.png (xxhdpi)`, `ic_launcher.png (xxxhdpi)`, and `ic_launcher.xml (anydpi-v26)`.
 - `ic_launcher_round (6)`
 - `values`
 - Gradle Scripts**: `build.gradle` (Project: TestApp)

A red arrow points from the `ic_launcher` folder in the project structure to the `android:icon` attribute in the manifest file.

AndroidManifest.xml Content:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.edu.xjtu.testapp">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher" // This line is highlighted with a red box
        android:label="TestApp"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

The screenshot shows the `MainActivity.java` file with the following content:

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

A red arrow points from the `activity_main` string in the `setContentView` call to the `activity_main` layout file in the `AndroidManifest.xml` manifest file.

The Manifest File

Layout Files

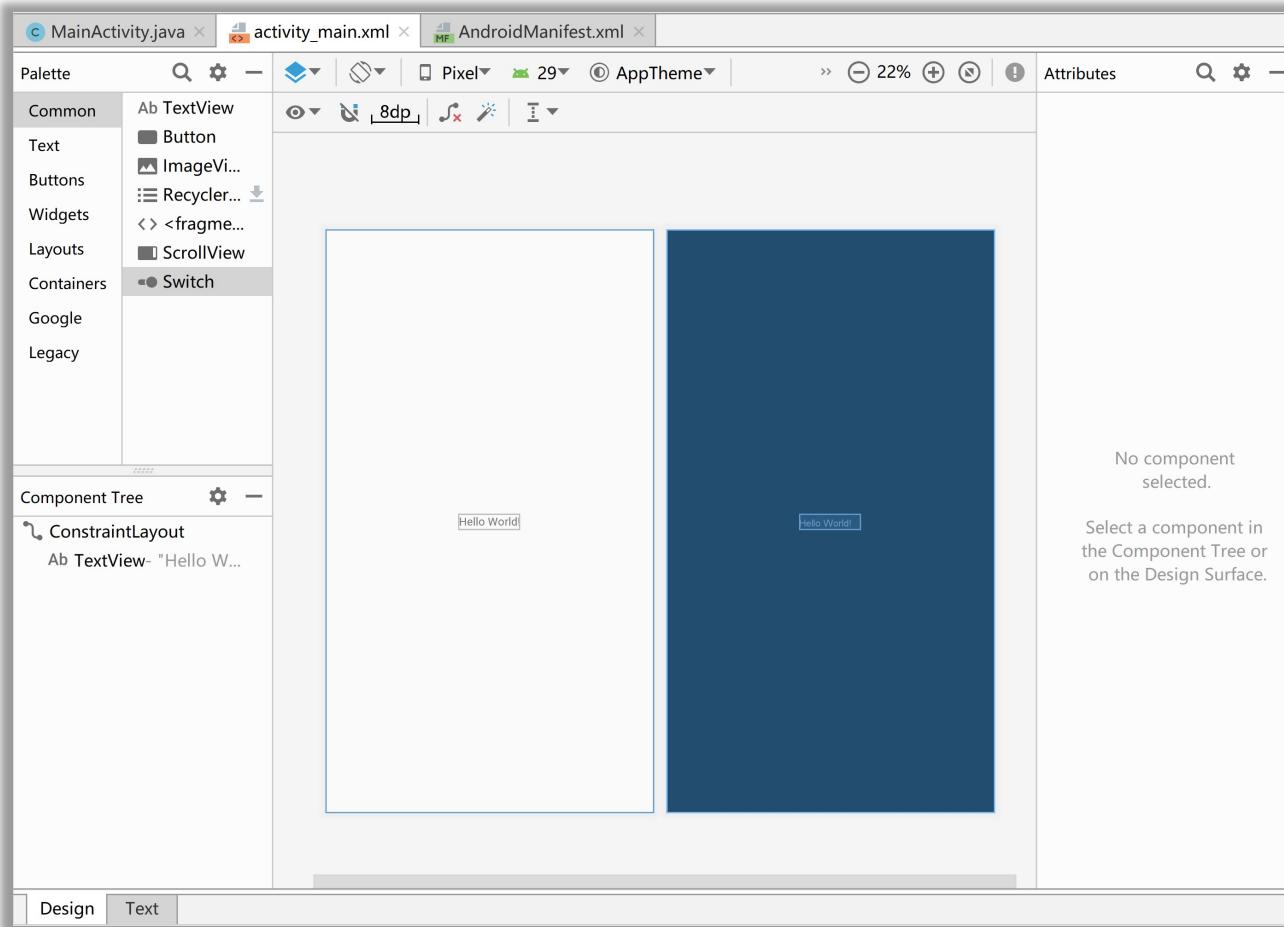
- Layout files are xml files that describe the widget settings.
 - They can be loaded from your java code.
 - Through a special java file generated automatically by android studio: R.java (will be discussed later)
- There are two different views for layout files
 - Design view
 - Text view



A screenshot of the Android Studio code editor showing the MainActivity.java file. The code defines a public class MainActivity that extends AppCompatActivity. The onCreate method is overridden, and it calls super.onCreate(savedInstanceState) and setContentView(R.layout.activity_main). The line setContentView(R.layout.activity_main) is underlined with a red error squiggle, indicating a syntax error or missing dependency.

```
1 package cn.edu.xjtu.testapp;
2
3 import ...
4
5
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14}
```

Layout Files: Design View



Layout Files: Text View



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="ht
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context=".MainActivity">
8
9     <TextView
10        android:layout_width="wrap_content"
11        android:layout_height="wrap_content"
12        android:text="Hello World!"<-- This line is highlighted with a red box
13        app:layout_constraintBottom_toBottomOf="parent"
14        app:layout_constraintLeft_toLeftOf="parent"
15        app:layout_constraintRight_toRightOf="parent"
16        app:layout_constraintTop_toTopOf="parent" />
17
18 </androidx.constraintlayout.widget.ConstraintLayout>
```

Running & Testing

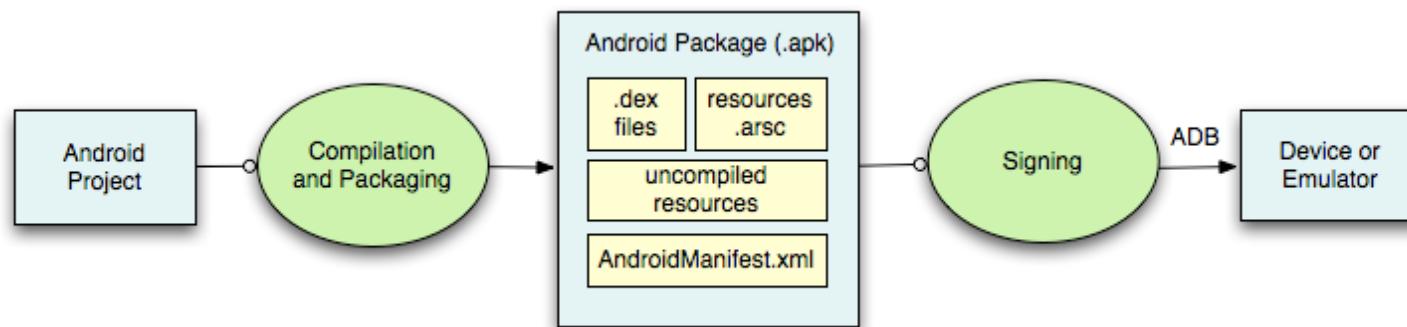
AVD setup

Connecting real devices

Tests

APK Files

- Android projects are compiled and packaged into a .apk file, the container for your application.
- To run an application on an emulator or device, the application must be signed using debug (or release) mode.
 - Signed: A certificate to identify the author of an application and establish trust relationships between applications



Running Android Apps

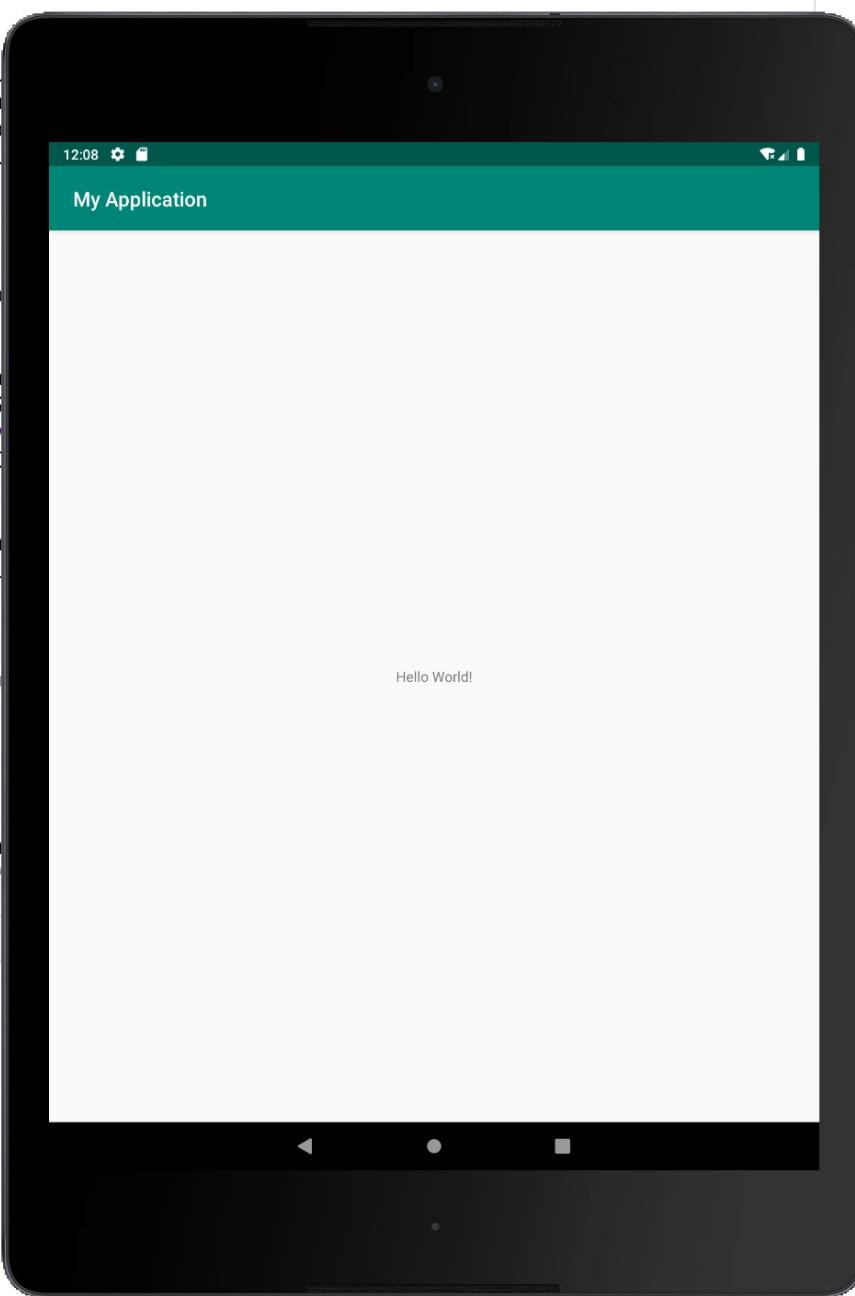
- By default, each application
 - is assigned a unique Linux user ID
 - executes in its own Linux process
- Android Apps are not Linux apps
- Android manages process creation & shutdown
 - Starts process when any of the application's code needs to be executed
 - Shuts down when process is no longer needed, and system resources are required by other applications

Android Virtual Device (AVD)

- To test the app you are developing, you'll need an android device, either virtual or real.
 - We have Xiaomi tablets this year. But the number is not enough for everyone. They will be used for the group project.
- To add an AVD:
 - Menu->Tools->AVD Manager
 - Toolbar Button



```
1 package com.example.starttestapp;
2
3 import android.os.Bundle;
4 import com.google.android.material.fl
5 import com.google.android.material.sn
6 import androidx.appcompat.app.AppCompatActivity
7 import androidx.appcompat.widget.Tool
8 import android.view.View;
9 import android.view.Menu;
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_main);
18         Toolbar toolbar = findViewById(R.id.toolbar);
19         setSupportActionBar(toolbar);
20
21         FloatingActionButton fab = findViewById(R.id.fab);
22         fab.setOnClickListener(new View.OnClickListener() {
23             @Override
24             public void onClick(View view) {
25                 Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
26                     .setAction("Action", null);
27             }
28         });
29     }
30
31     @Override
32     public boolean onCreateOptionsMenu(Menu menu) {
33         // Inflate the menu; this adds items to the action bar if it is present.
34         getMenuInflater().inflate(R.menu.main, menu);
35     }
36
37     @Override
38     public boolean onOptionsItemSelected(MenuItem item) {
39         // Handle action bar item clicks here. The action bar will
40         // automatically handle clicks on the Home/Up button, so long
41         // as you specify a parent activity in AndroidManifest.xml.
42         int id = item.getItemId();
43
44         //noinspection SimplifiableIfStatement
45         if (id == R.id.action_settings) {
46             return true;
47         }
48
49         return super.onOptionsItemSelected(item);
50     }
51 }
```



Connecting Real Devices

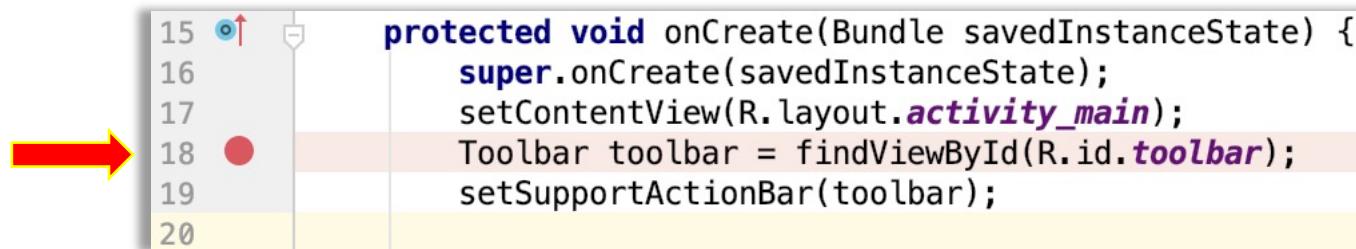
- Connect your device to your development machine
- Enable USB debugging in your device
 - Open the Settings app.
 - Scroll to the bottom and select About phone.
 - Scroll to the bottom and tap Build number 7 times.
 - Return to the previous screen to find Developer options near the bottom.
 - Open Developer options, and then scroll down to find and enable USB debugging.

Connecting Real Devices

- Run the app on your device as follows:
 - In Android Studio, click Run in the toolbar.
 - In the Select Deployment Target window, select your device, and click OK.
 - Android Studio will install the app on your device and starts it (may take a while when starting for the first time!). You should see the app on your device.

Debugging

- Adding breakpoints:
 - Single click on the left area of the code editor (the empty area at the right side of code line numbers)
 - Or press Ctrl + F8 at the line you want to add/remove breakpoint.



A screenshot of an Android Studio code editor showing the Java file `MainActivity.java`. The code is as follows:

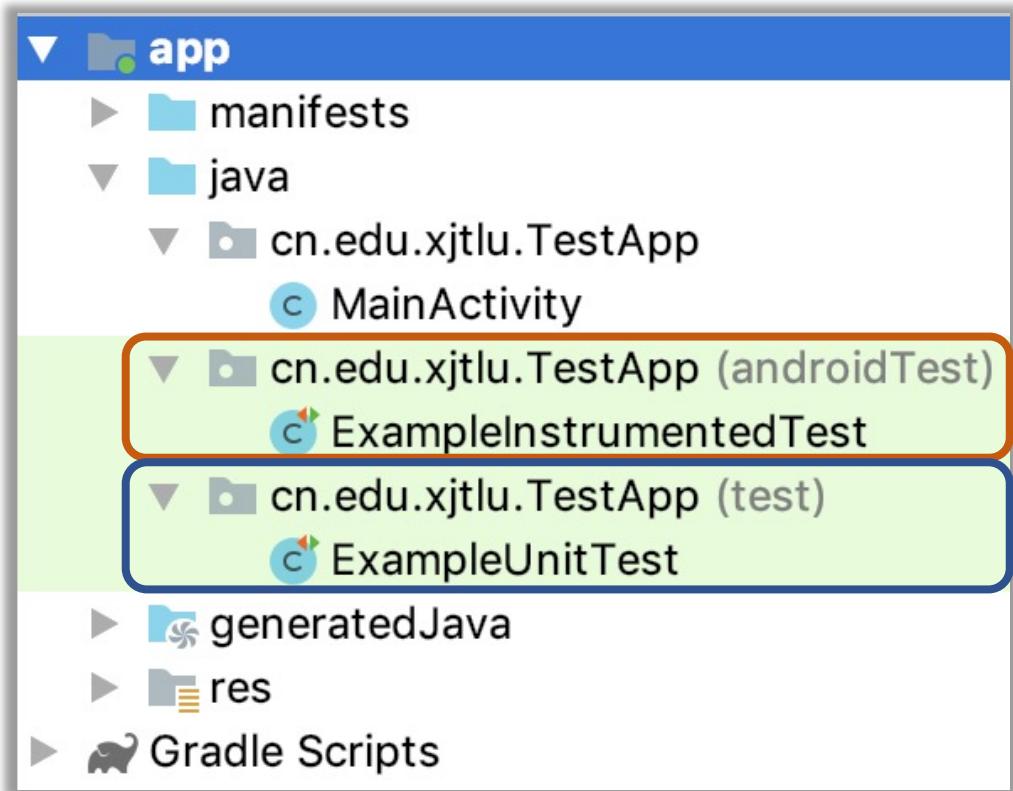
```
15  protected void onCreate(Bundle savedInstanceState) {  
16      super.onCreate(savedInstanceState);  
17      setContentView(R.layout.activity_main);  
18  ● Toolbar toolbar = findViewById(R.id.toolbar);  
19      setSupportActionBar(toolbar);  
20  }
```

A red arrow points to the line number 18, which contains the assignment statement `Toolbar toolbar = findViewById(R.id.toolbar);`. A red circle marks the breakpoint on this line. The code editor has a light gray background with colored syntax highlighting: blue for keywords like `protected`, `void`, and `super`; purple for class names like `MainActivity` and `Toolbar`; and pink for method names like `onCreate` and `findViewById`.

- Debug mode:
 - Menu -> Run -> Debug 'App'

Writing Tests

- Test cases are placed along with your main App source in the project view.
- Unit tests run locally in the IDE and can help check the correctness of single components.
- Instrumented unit tests are tests that run on physical devices and emulators



<https://developer.android.com/training/testing/unit-testing/local-unit-tests>

<https://developer.android.com/training/testing/unit-testing/instrumented-unit-tests>

Lab Session Practice

- Get yourselves familiar with Android Studio
- Practice the following tasks:
 - Create an application named “MyApp”.
 - Change the icon of your app with a picture you preferred
 - Change the name of your app
 - Change the content your app shows when launched
- Set up Android Studio on desktop if you don’t plan to use your laptop in future labs.
 - Download SDK and AVD.
 - You will have to do this again if you change seat.

Don't Forget

The official manuals and guides can be found at:

<https://developer.android.com/guide>

or

<https://developer.android.google.cn/guide>