Tutorial: Setup for Android Development

Adam C. Champion, Ph.D.

CSE 5236: Mobile Application Development

Autumn 2019

Based on material from C. Horstmann [1], J. Bloch [2], C. Collins et al. [4], M.L. Sichitiu (NCSU), V. Janjic (Imperial College London), CSE 2221 (OSU), and other sources

Outline

- **Getting Started**
- Android Programming

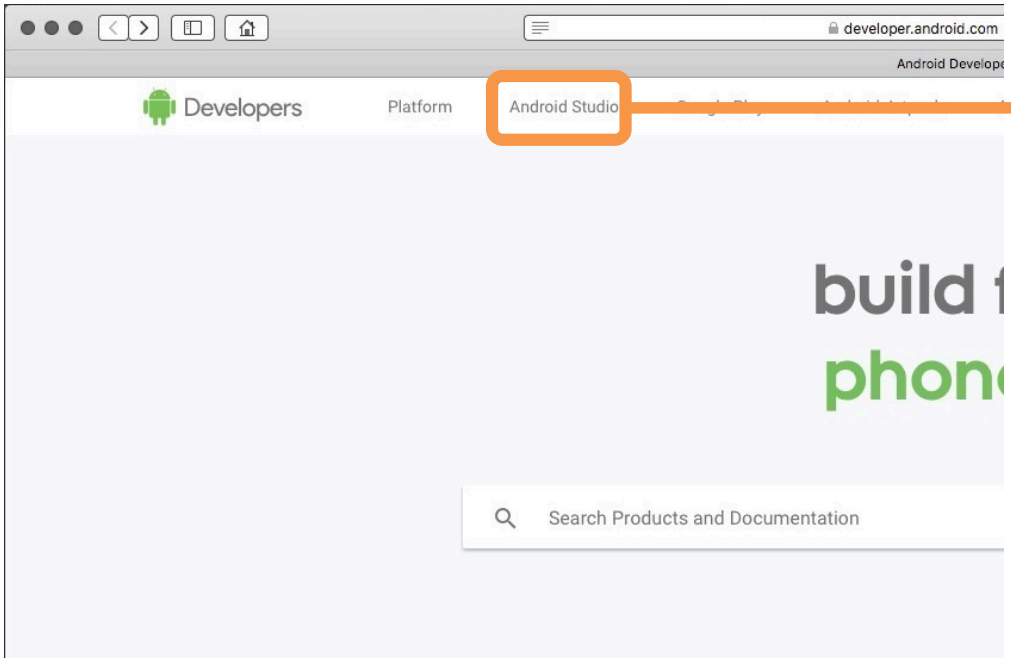
Getting Started (1)

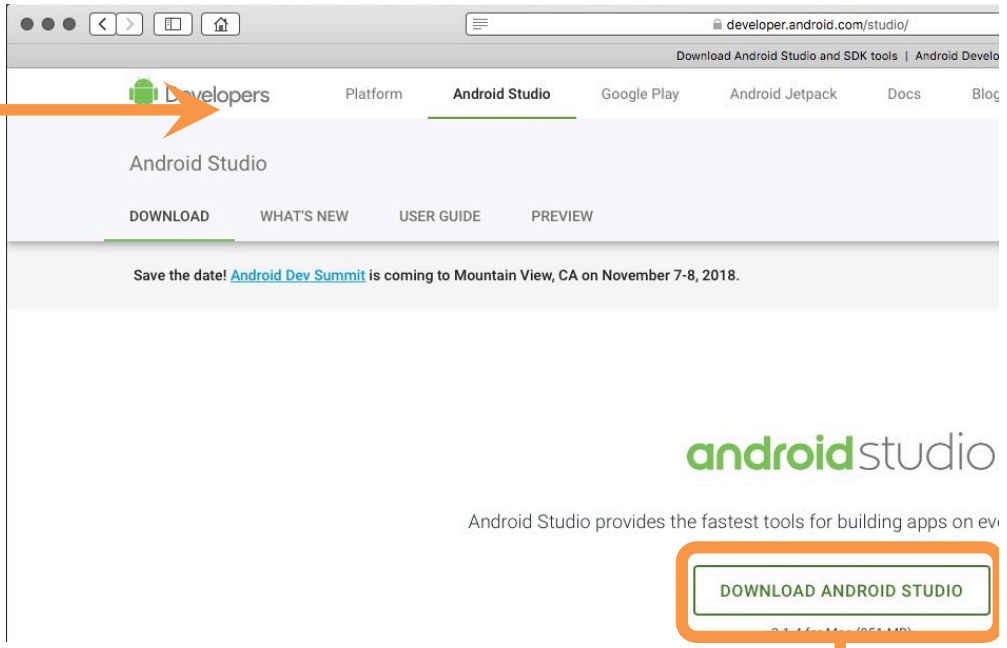
- Need to install Java Development Kit (JDK) (**not** Java Runtime Environment (JRE)) to write Android programs
- Download JDK for your OS: https://adoptopenjdk.net/ *
- Alternatively, for OS X, Linux:
  - OS X: Install Homebrew (http://brew.sh) via Terminal,
  - Linux:
    - Debian/Ubuntu: `sudo apt install openjdk-8-jdk`
    - Fedora/CentOS: `yum install java-1.8.0-openjdk-devel`

* Why OpenJDK 8? Oracle changed Java licensing (commercial use costs $$$); Android SDK tools require version 8.

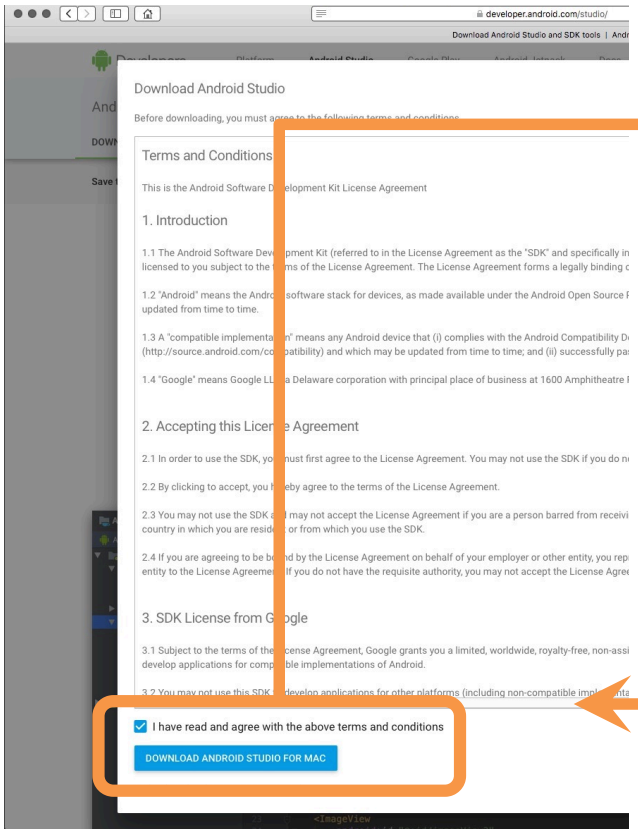
Getting Started (2)

- After installing JDK, download Android SDK from http://developer.android.com
- Simplest: download and install Android Studio bundle (including Android SDK) for your OS
- Alternative: `brew cask install android-studio` (Mac/Homebrew)
- We'll use Android Studio with SDK included (easiest)


developer.android.com
Developers
Platform
Android Studio
Search Products and Documentation

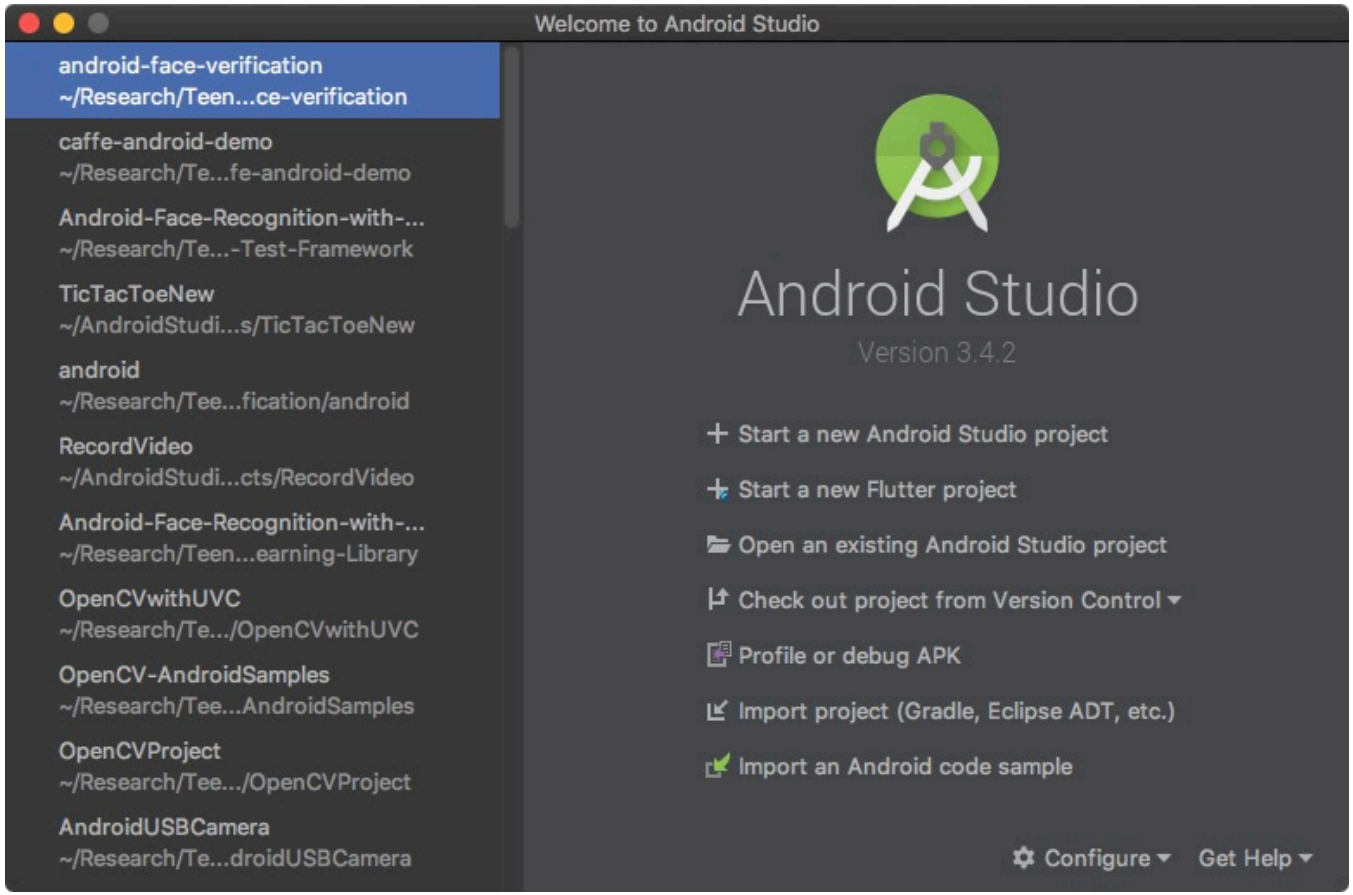

developer.android.com/studio/
Developers
Platform
Android Studio
Google Play
Android Jetpack
Docs
Android Studio
DOWNLOAD
WHAT'S NEW
USER GUIDE
PREVIEW
Save the date! Android Dev Summit is coming to Mountain View, CA on November 7-8, 2018.
android studio
Android Studio provides the fastest tools for building apps on ev
DOWNLOAD ANDROID STUDIO

Install!


Download Android Studio
Terms and Conditions
1. Introduction
2. Accepting this License Agreement
3. SDK License from Google
I have read and agree with the above terms and conditions
DOWNLOAD ANDROID STUDIO FOR MAC

Getting Started (3)

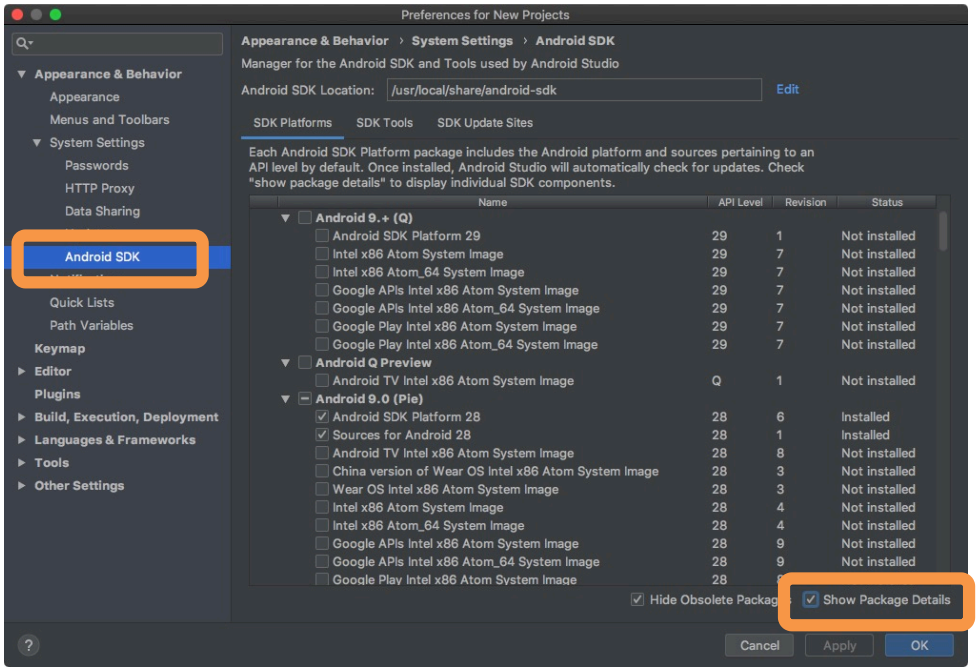
- Install Android Studio directly (Windows, Mac); unzip to directory `android-studio`, then run `./android-studio/bin/studio64.sh` (Linux)



Getting Started (4)

- Strongly recommend testing with real Android device
  - Android emulator: *slow*
  - Faster emulator: Genymotion [14], [15]
  - Install USB drivers for your Android device!
- Bring up Android SDK Manager
  - Install Android 5.x–8.x APIs, Google support repository, Google Play services
  - Don't worry about non-x86 system images

Android Studio menu → *Preferences…* or *File → Settings…*



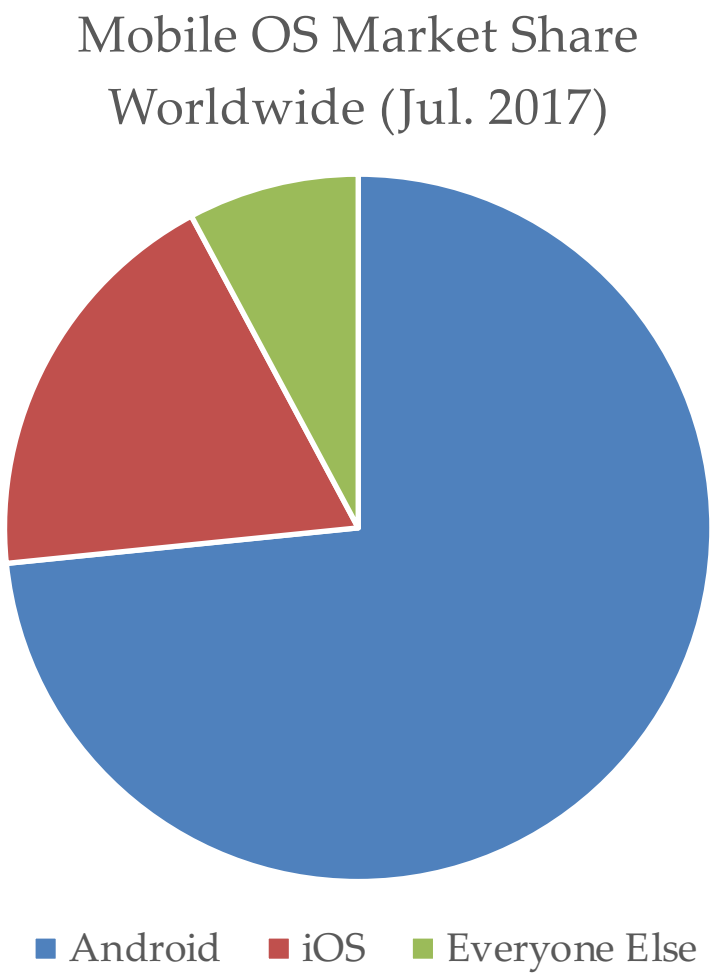
Now you're ready for Android development!

Outline

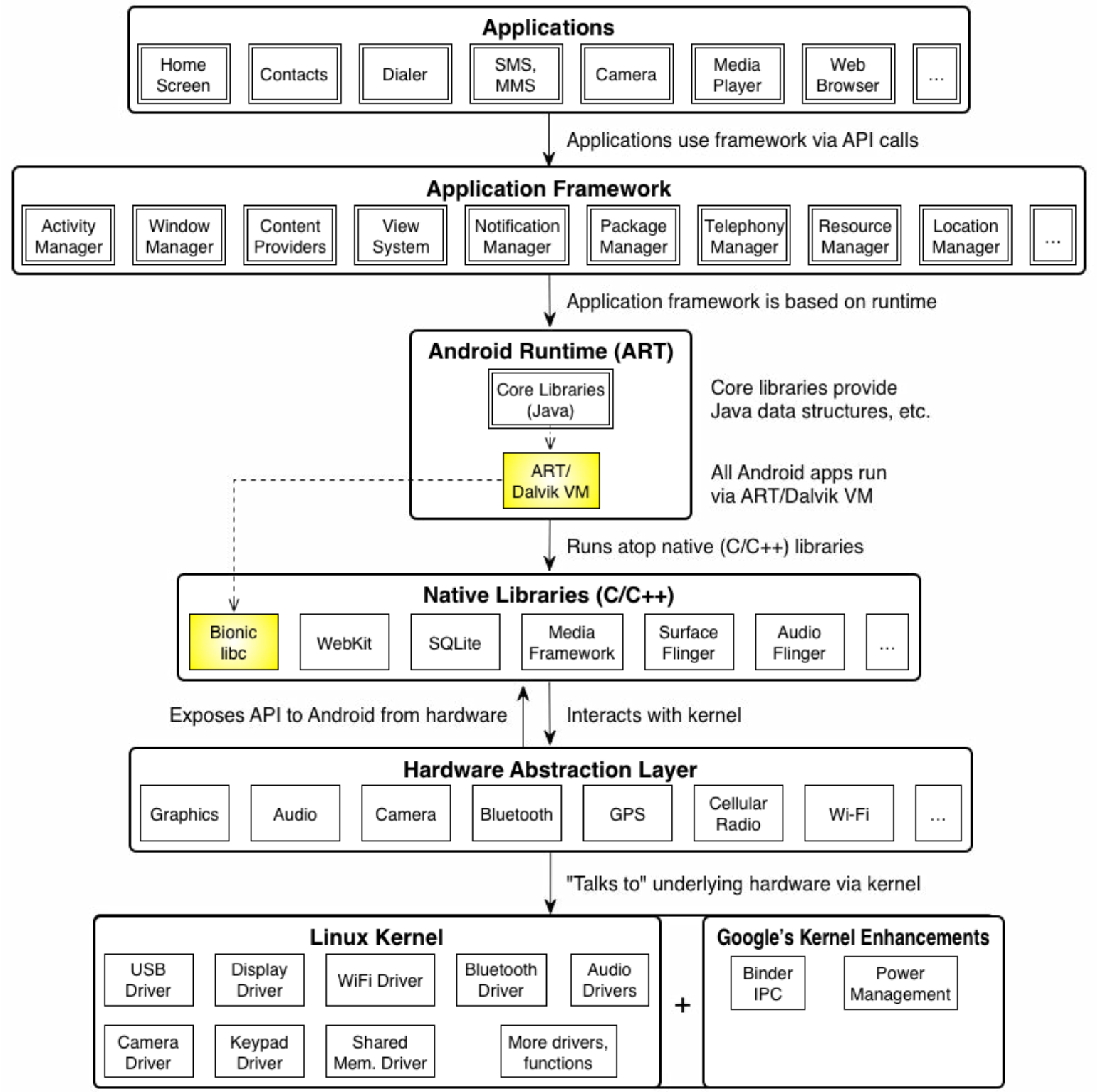
- Getting Started
- **Android Programming**

Introduction to Android

- Popular mobile device OS: 73% of worldwide smartphone market [8]
- Developed by Open Handset Alliance, led by Google
- Google claims 2 billion Android devices in use worldwide [9]

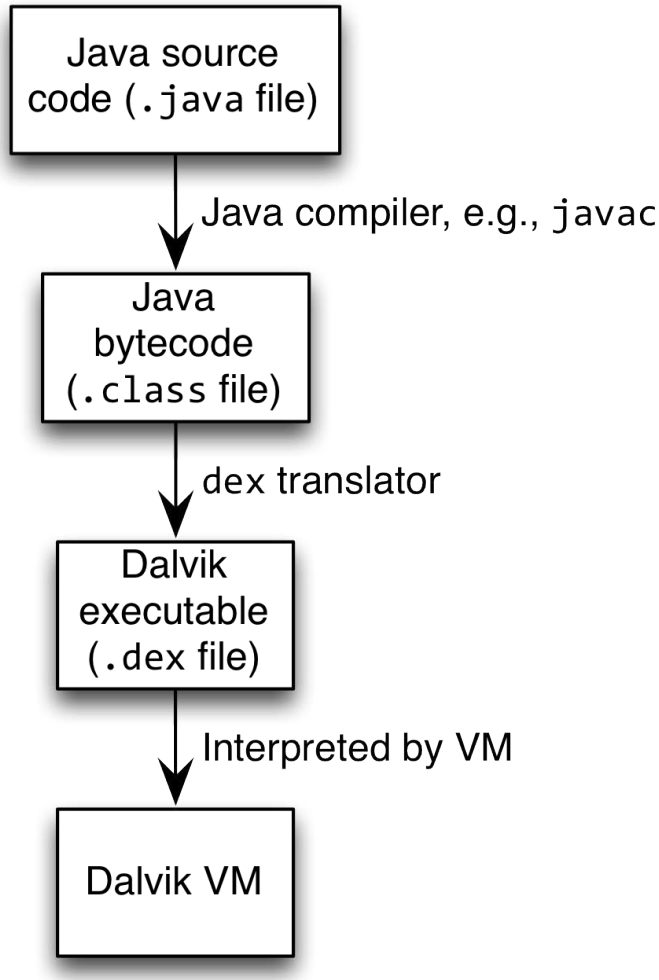


Source: [8]


Applications
Home Screen
Contacts
Dialer
SMS, MMS
Camera
Media Player
Web Browser
…
Applications use framework via API calls
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Notification Manager
Package Manager
Telephony Manager
Resource Manager
Location Manager
…
Application framework is based on runtime
Android Runtime (ART)
Core Libraries (Java)
ART/ Dalvik VM
Core libraries provide Java data structures, etc.
All Android apps run via ART/Dalvik VM
Runs atop native (C/C++) libraries
Native Libraries (C/C++)
Bionic libc
WebKit
SQLite
Media Framework
Surface Flinger
Audio Flinger
…
Exposes API to Android from hardware
Interacts with kernel
Hardware Abstraction Layer
Graphics
Audio
Camera
Bluetooth
GPS
Cellular Radio
Wi-Fi
…
"Talks to" underlying hardware via kernel
Linux Kernel
USB Driver
Display Driver
WiFi Driver
Bluetooth Driver
Audio Drivers
Camera Driver
Keypad Driver
Shared Mem. Driver
More drivers, functions
+
Google's Kernel Enhancements
Binder IPC
Power Management

Android Highlights (1)

- Android apps execute on Android Runtime (ART) (version of JVM)
  - Optimized for efficiency
  - Register-based VM, unlike Oracle's stack-based JVM
  - Java `.class` bytecode translated to Dalvik EXecutable (DEX) bytecode that ART interprets
  - Android 5+: ART with ahead-of-time, just-in-time compilation at *install* time



Android Highlights (2)

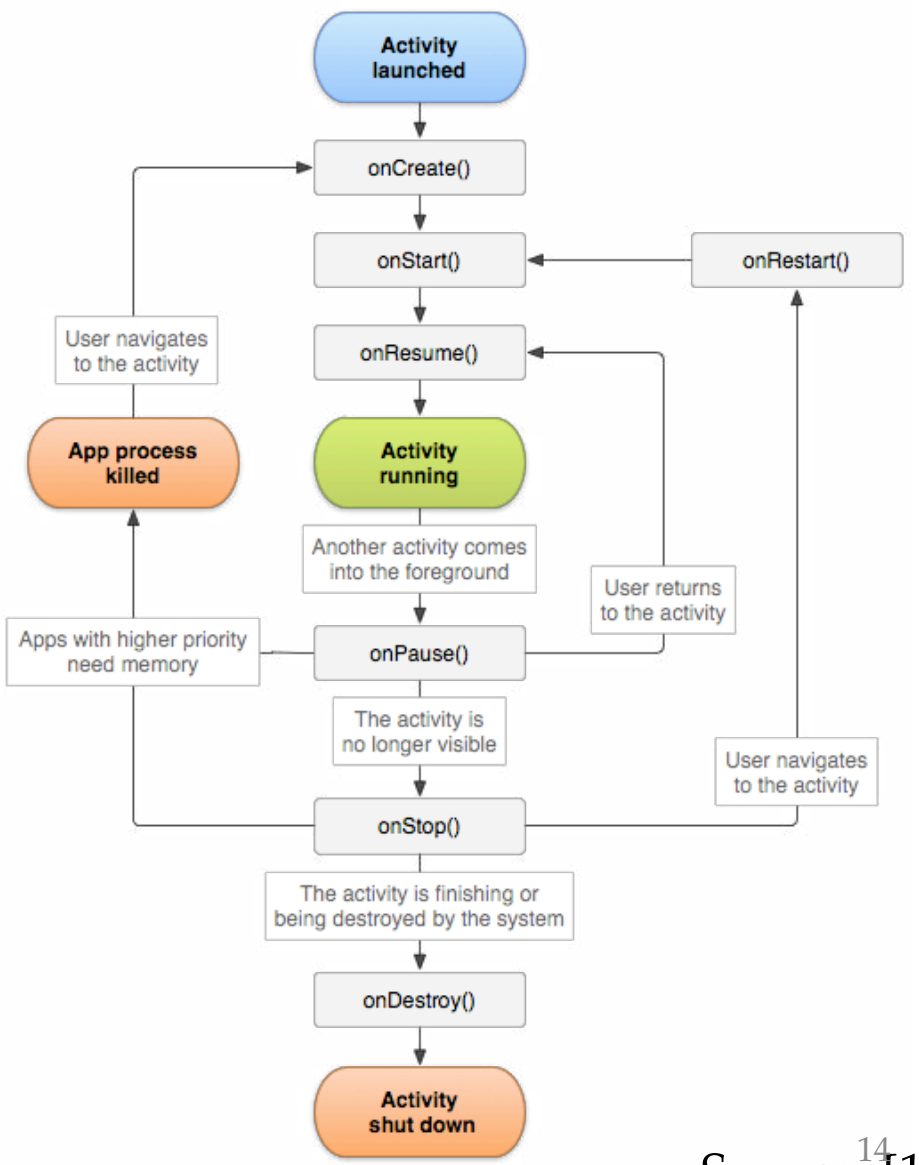
- Android apps written in Java 7+, Kotlin
- Apps use four main components:
  - `Activity`: A "single screen" that's visible to user
  - `Service`: Long-running background "part" of app (*not* separate process or thread)
  - `ContentProvider`: Manages app data (usually stored in database) and data access for queries
  - `BroadcastReceiver`: Component that listens for particular Android system events (*e.g.*, "found wireless device"), responds accordingly

App Manifest

- Each Android app must include an `AndroidManifest.xml` file describing functionality
- The manifest specifies:
  - App's Activities, Services, etc.
  - Permissions requested by app
  - Minimum API required
  - Hardware features required (*e.g.*, camera with autofocus)

Activity Lifecycle

- `Activity`: key building block of Android apps
- Extend `Activity` class, override `onCreate()`, `onPause()`, `onResume()` methods
- ART can stop any `Activity` without warning, so saving state is important!
- Activities must be responsive, otherwise user gets warning:
  - Place lengthy operations in `Runnables`, `AsyncTasks`



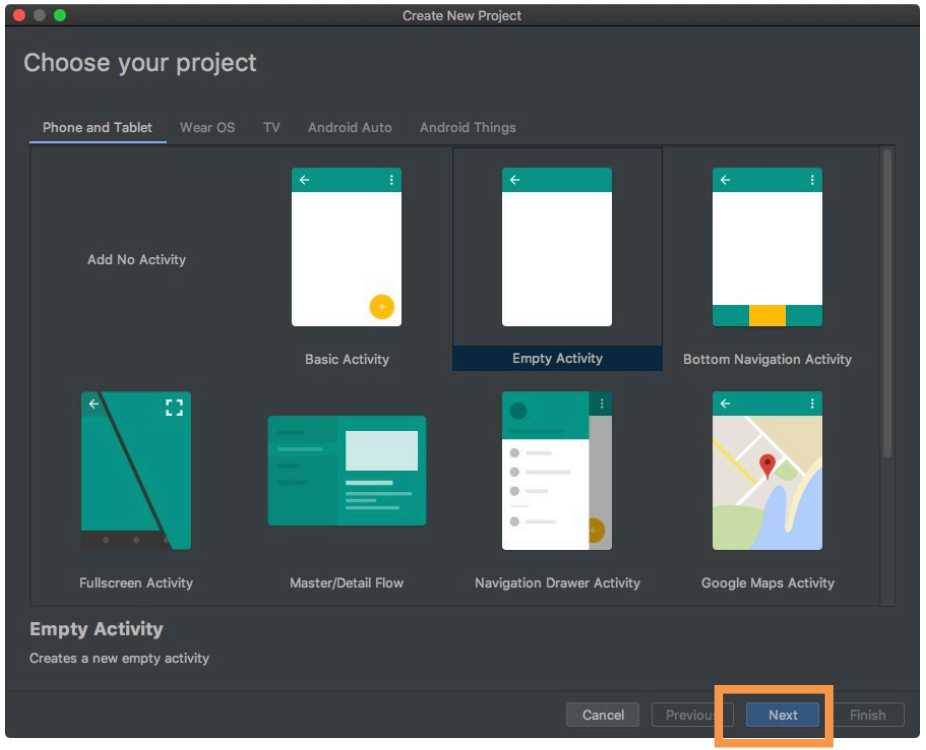
Source: [12]

App Creation Checklist

- If you own an Android device:
  - Ensure drivers are installed
  - Enable developer options on device: Go to *Settings*→ *About phone*, press *Build number* 7 times, check *USB debugging*
- For Android Studio:
  - Go to *File*→*Settings,* check "Show tool window bars" (Appearance)
  - Log state via `android.util.Log`'s `Log.d(APP_TAG_STR, "debug")`, where `APP_TAG_STR` is a `final String`
  - Other commands: `Log.e()` (error); `Log.i()` (info); `Log.w()` (warning); `Log.v()` (verbose) – same parameters

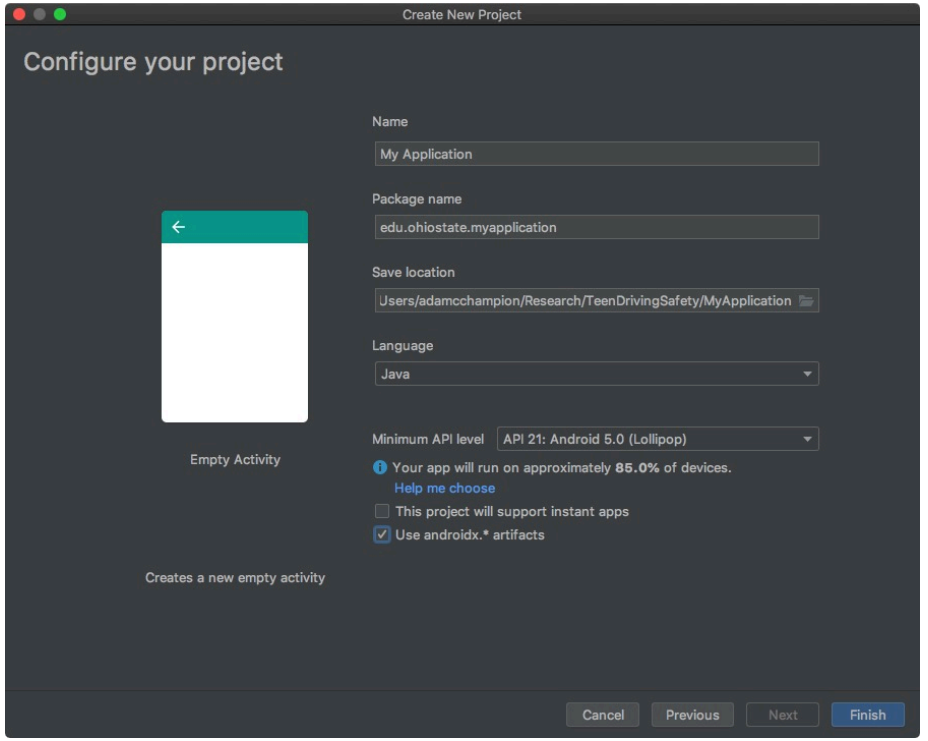
Creating Android App (1)

- Creating new project in Android Studio:
  - Go to *File→New Project*
  - Determine what kind of Activity to create (chose Empty Activity for simplicity)
  - Click *Next*



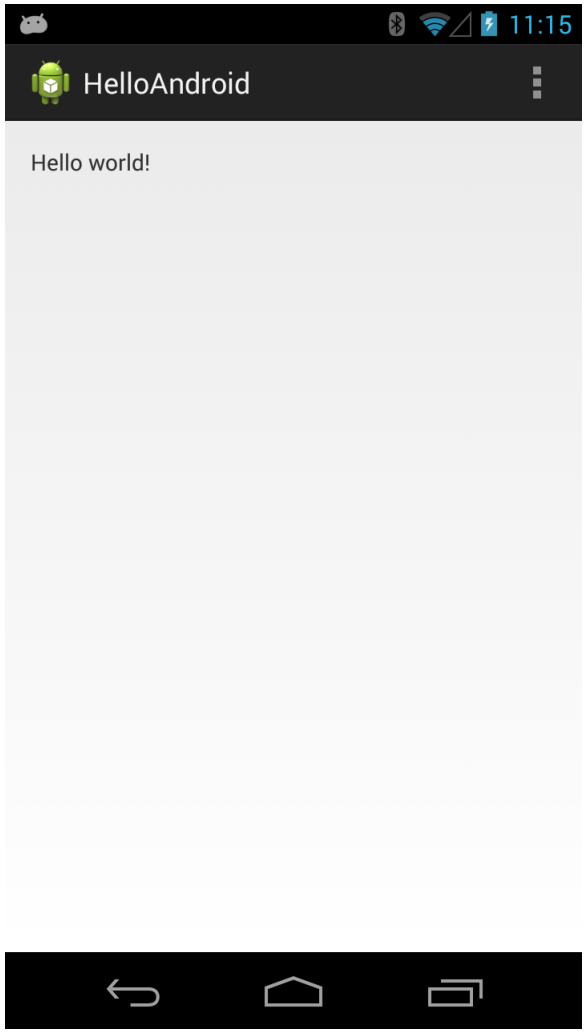
Creating Android App (2)

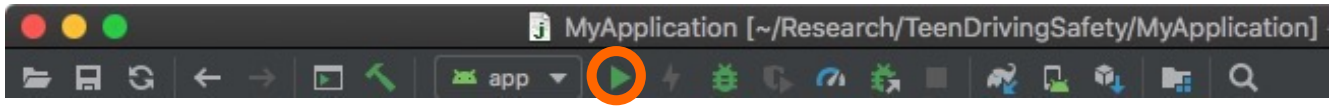
- Enter your app name (reverse DNS syntax)
- Can select:
  - Kotlin support
  - Minimum Android version (5+)
  - Using `androidx` support packages (recommend)
- This creates a "Hello World" app



Deploying the App

- Two choices for deployment:
  - Real Android device
  - Android virtual device
- Plug in your real device; otherwise, create an Android virtual device
- Emulator is slow. Try Intel accelerated one, or Genymotion
- Run the app: press “Run” button in toolbar





Underlying Source Code

src/…/MainActivity.java

```
package edu.osu.myapplication;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

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

Underlying GUI Code

res/layout/activity_main.xml

```
<RelativeLayout
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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</RelativeLayout>
```

– `RelativeLayouts` are complicated; see [13] for details

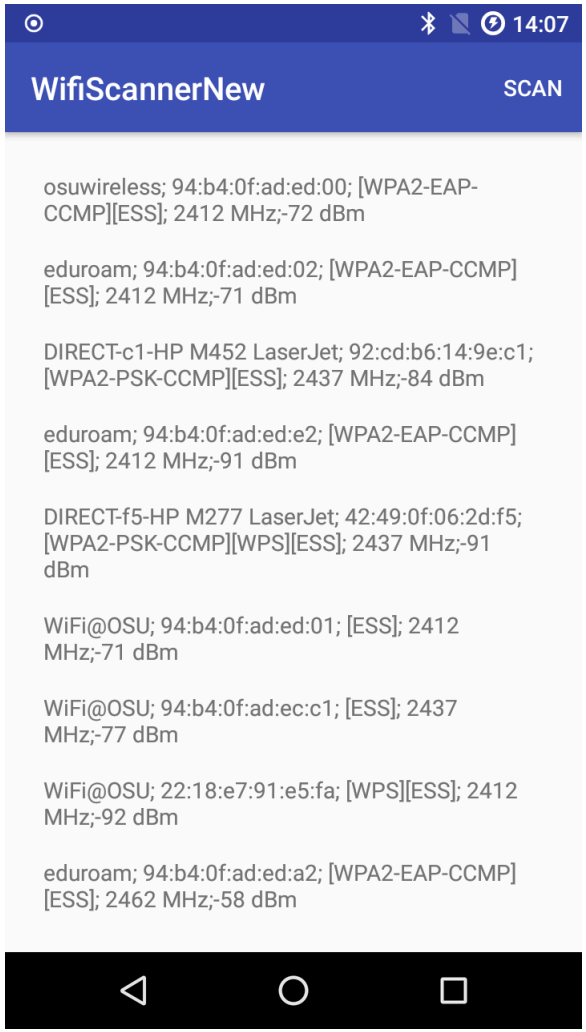
The App Manifest

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest package="edu.ohiostate.myapplication"
  xmlns:android="http://schemas.android.com/apk/res/android">
  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    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>
```

A More Interesting App

- We'll now examine an app with more features: WiFi Scanner (code on class website)
- Press a button, scan for Wi-Fi access points (APs), display them
- Architecture: Activity creates single Fragment with app



Underlying Source Code (1)

```
// WifiScanActivity.java
public class WifiScanActivity extends SingleFragmentActivity {
    @Override
    protected Fragment createFragment() {return new WifiScanFragment(); }
}
// WifiScanFragment.java. Uses RecyclerView to display dynamic list of Wi-Fi ScanResults.
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    View v = inflater.inflate(R.layout.fragment_wifi_scan, container, false);
    mScanResultRecyclerView = (RecyclerView) v.findViewById(R.id.scan_result_recyclerview);
    mScanResultAdapter = new ScanResultAdapter(mScanResultList);
    mScanResultRecyclerView.setAdapter(mScanResultAdapter);
    mScanResultRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));

    setupWifi();
    mIntentFilter = new IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION);

    setHasOptionsMenu(true); setRetainInstance(true);

    return v;
}

private void setupWifi() {
    try {
        Context context = getActivity().getApplicationContext();
        if (context != null) {
            mWifiManager = (WifiManager) context.getSystemService(Context.WIFI_SERVICE);
        }
    } catch (NullPointerException npe) { /* Error handling */ }
}
```

Underlying Source Code (2)

- Get system `WifiManager`
- Register BroadcastReceiver to listen for `WifiManager`'s "finished scan" system event (`Intent WifiManager.SCAN_RESULTS_AVAILABLE_ACTION`)
- Unregister Broadcast Receiver when leaving `Fragment`

```
@Override
public void onResume() { // . . .
  super.onResume(); // . . .
  SharedPreferences sharedPreferences =
    PreferenceManager.getDefaultSharedPreferences(getActivity().getApplicationContext());
    boolean hideDialog =
      sharedPreferences.getBoolean(getResources().getString(R.string.suppress_dialog_key), false);
    if (!hideDialog) { // Show user dialog asking them to accept permission request
        FragmentManager fm = getActivity().getSupportFragmentManager();
        DialogFragment fragment = new NoticeDialogFragment();
        fragment.show(fm, "info_dialog"); }
    getActivity().registerReceiver(mReceiver, mIntentFilter);
}

@Override
public void onPause() {
    super.onPause();
    getActivity().unregisterReceiver(mReceiver);
}
```

Underlying Source Code (3)

- Register menu-item listener to perform Wi-Fi scan
- Get user permission first for "coarse" location (required in Android 6+)

```
// WifiScanFragment.java
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) { /* Call super.onCreate...() */
    inflater.inflate(R.menu.menu, menu); }

public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {
    case R.id.menu_scan:
      if (!hasLocationPermission()) { requestLocationPermission(); }
      else { doWifiScan(); }
    return true; }
  return false; }

private void requestLocationPermission() {
  if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    if (!hasLocationPermission()) {
      requestPermissions(new String[]{Manifest.permission.ACCESS_COARSE_LOCATION}, PERM_REQUEST_LOCATION);
}}}

public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
    if (requestCode == PERMISSION_REQUEST_LOCATION) {
        if (grantResults[0] == PackageManager.PERMISSION_GRANTED) { doWifiScan(); }
        else { // Error } }}}
```

The Broadcast Receiver

```
// WifiScanFragment.java
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
    // Override onReceive() method to implement our custom logic.
    @Override
    public void onReceive(Context context, Intent intent) {
        // Get the Intent action.
        String action = intent.getAction();

        // If the WiFi scan results are ready, iterate through them and
        // record the WiFi APs' SSIDs, BSSIDs, WiFi capabilities, radio
        // frequency, and signal strength (in dBm).
        if (WifiManager.SCAN_RESULTS_AVAILABLE_ACTION.equals(action)) {
            // Ensure WifiManager is not null first.
            if (mWifiManager == null) { setupWifi(); }

            List<ScanResult> scanResults = mWifiManager.getScanResults();
            mScanResultList.addAll(scanResults);
            mScanResultAdapter.notifyDataSetChanged();
        }
    }
};
```

User Interface

Updating UI in code

- Two inner classes handle `RecyclerView` items:
  - `ScanResultAdapter` (extends `RecyclerView.Adapter<ScanResultHolder>`)
  - `ScanResultHolder` (extends `RecyclerView.ViewHolder`)
- See code, Big Nerd Ranch (Chapter 8) for details

UI Layout (XML)

```
<!-- fragment_wifi_scan.xml
     (for the RecyclerView fragment) -->
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
  <android.support.v7.widget.RecyclerView
    android:id="@+id/scan_result_recyclerview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
</LinearLayout>

<!-- item_wifi_scan.xml
     (for each RecyclerView item) -->
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <TextView
        android:id="@+id/scan_result_textview"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="TextView"/>
</LinearLayout>
```

Android Programming Notes

- Android apps have multiple entry points; no `main()` method
  - Cannot "sleep" in Android
  - During each entrance, certain `Objects` may be `null`
  - Null-check programming is very useful to avoid crashes
- Java concurrency techniques are required
  - Don't block the main thread in Activities
  - Implement long-running tasks (e.g, network) as `AsyncTasks`
  - Recommendation: read [4]; chapter 20 [10]; [11]
- Logging state via `Log.d()` in app is essential when debugging
- Request only the permissions that you need; otherwise, app crash!
- Event handling in Android GUIs entails listener `Objects`

Concurrency: Threads (1)

- Thread: program unit (within process) running independently
- Basic idea: create class that implements Runnable interface
  - Runnable has one method, run(), that has code to execute
  - Example:

```
public class OurRunnable implements Runnable {
    public void run() {
        // run code
    }
}
```

- Create a Thread object from Runnable and start() Thread, e.g.,

```
Runnable r = new OurRunnable();
Thread t = new Thread(r);
t.start();
```

- Problems: cumbersome, does not reuse Thread code

Concurrency: Threads (2)

- Easier approach: anonymous inner classes, e.g.,

```
Thread t = new Thread(new Runnable( {
        public void run() {
            // code to run
        }
    });
t.start();
```

- Idiom essential for *one-time* network connections in Activities
- However, Threads are hard to synchronize, especially with UI thread; AsyncTasks beter

Concurrency: AsyncTasks

- AsyncTask handles background task, interacts with UI thread:

```
public class AsyncTask<ParamsType, ProgressType, ResultType> {
    protected Result doInBackground(ParamType param) {
        // code to run in background
        publishProgress(ProgressType progress); // UI
        …
        return Result;
    }

    protected void onProgressUpdate(ProgressType progress) {
        // invoke method in Activity to update UI
    }
}
```

- Extend AsyncTask as you like (see http://developer.android.com)

Thank You

Any questions?

References (1)

1. C. Horstmann, *Big Java Late Objects*, Wiley, 2012. Online: http://proquest.safaribooksonline.com.proxy.lib.ohio–state.edu/book/–/9781118087886
2. J. Bloch, *Effective Java*, 2nd ed., Addison–Wesley, 2008. Online: http://proquest.safaribooksonline.com.proxy.lib.ohio–state.edu/book/programming/java/9780137150021
3. S.B. Zakhour, S. Kannan, and R. Gallardo, *The Java® Tutorial: A Short Course on the Basics*, 5th ed., Addison–Wesley, 2013. Online: http://proquest.safaribooksonline.com.proxy.lib.ohio–state.edu/book/programming/java/9780132761987
4. C. Collins, M. Galpin, and M. Kaeppler, *Android in Practice*, Manning, 2011. Online: http://proquest.safaribooksonline.com.proxy.lib.ohio–state.edu/book/programming/android/9781935182924
5. M.L. Sichitiu, 2011, http://www.ece.ncsu.edu/wireless/MadeInWALAN/AndroidTutorial/PPTs/javaReview.ppt
6. Oracle, http://docs.oracle.com/javase/1.5.0/docs/api/index.html
7. Wikipedia, https://en.wikipedia.org/wiki/Vehicle_Identification_Number
8. StatCounter Global Stats, "Mobile operating system market share worldwide," Jul. 2017, http://gs.statcounter.com/os-market-share/mobile/worldwide

References (2)

10. http://bcs.wiley.com/he-bcs/Books?action=index&itemId=1118087887&bcsId=7006
11. B. Goetz, T. Peierls, J. Bloch, J. Bowbeer, D. Holmes, and D. Lea, Java Concurrency in Practice, Addison-Wesley, 2006, online at http://proquest.safaribooksonline.com/book/programming/java/0321349601
12. https://developer.android.com/guide/components/activities.html
13. https://developer.android.com/guide/topics/ui/declaring-layout.html#CommonLayouts
14. https://cloud.genymotion.com/page/doc/#collapse4
15. http://blog.zeezonline.com/2013/11/install-google-play-on-genymotion-2-0/