

Tutorial: Setup for Android Development

Adam C. Champion

CSE 5236: Mobile Application Development
Spring 2018

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) to write Java (and Android) programs
 - **Do not** install Java Runtime Environment (JRE); JDK and JRE are different!
- Can download the JDK for your OS at
<http://java.oracle.com>
- Alternatively, for OS X, Linux:
 - OS X:
 - Open /Applications/Utilities/Terminal.app
 - Type javac at command line, install Java at prompt
 - Linux:
 - Debian/Ubuntu: sudo apt-get install java-package, download the JDK <jdk>.tar.gz file from Oracle, run make-jpkg <jdk>.tar.gz, then sudo dpkg -i <resulting-deb-file>
 - Fedora/OpenSuSE: download the JDK .rpm file from Oracle, install

http://java.oracle.com/

www.oracle.com/technetwork/java/index.html

Oracle Technology Network for Java Developers

ORACLE

PRODUCTS AND SERVICES SOLUTIONS DOWNLOADS STORE SUPPORT TRAINING PARTNERS ABOUT

Oracle Technology Network > Java

ARTICLE

Java API for JSON Processing: An Introduction to JSON

The Java API for JSON Processing provides portable APIs to parse, generate, transform, and query JSON.

Posted 7/18/13 // Tags: java, JSON, javascript // Headlines Archive

Software Downloads

Top Downloads

- Java SE
- Java EE and GlassFish
- JavaFX
- Java ME
- JDeveloper 11g and ADF
- Enterprise Pack for Eclipse
- NetBeans IDE
- Pre-Built VM for Java Devs

java Downloads

Install!

Java SE Downloads

www.oracle.com/technetwork/java/javase/downloads/index.html

ORACLE

PRODUCTS AND SERVICES SOLUTIONS DOWNLOADS STORE SUPPORT

Oracle Technology Network > Java > Java SE > Downloads

Java SE Downloads

Overview Downloads Documentation Community Technologies Training

Java SE Development Kit 7 Downloads

Java SE Development Kit 7u25 Downloads

Accept License Agreement

Download

Java Platform (JDK) 7u25

Product / File Description	File Size	Download
Linux x86	80.38 MB	jdk-7u25-linux-i586.rpm
Linux x86	93.12 MB	jdk-7u25-linux-i586.tar.gz
Linux x64	81.46 MB	jdk-7u25-linux-x64.rpm
Linux x64	91.85 MB	jdk-7u25-linux-x64.tar.gz
Mac OS X x64	14.43 MB	jdk-7u25-macosx-x64.dmg
Solaris x86 (SVR4 package)	136.19 MB	jdk-7u25-solaris-i586.tar.Z
Solaris x86	92.2 MB	jdk-7u25-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	220.7 MB	jdk-7u25-solaris-x64.tar.Z
Solaris x64	15.09 MB	jdk-7u25-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	136.16 MB	jdk-7u25-solaris-sparc.tar.Z
Solaris SPARC	95.5 MB	jdk-7u25-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	23.05 MB	jdk-7u25-solaris-sparc9.tar.Z
Solaris SPARC 64-bit	17.67 MB	jdk-7u25-solaris-sparc9.tar.gz
Windows x86	89.09 MB	jdk-7u25-windows-32bit.exe
Windows x64	90.06 MB	jdk-7u25-windows-x64.exe

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
- Alternatives:
 - Download/install Android Developer Tools from this site (based on Eclipse)
 - Install Android SDK tools by themselves, then install ADT for Eclipse separately (from this site)
- We'll use Android Studio with SDK included (easy)

Android 8.0 Oreo

Smarter, faster, and more powerful than ever. The world's favorite cookie is your new favorite Android release.

> Learn more

> Get Android Studio

Android Studio

← Back to Developers

DOWNLOAD

FEATURES

USER GUIDE

PREVIEW

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

World-class code editing, debugging, and flexible build system, and an instant build allow you to focus on building unique apps.

DOWNLOAD ANDROID STUDIO

2.3.3 FOR MAC (463 MB)

> Read the docs > See the release notes

> Features > Learn more

Before downloading, you must agree to the following terms and conditions:

Terms and Conditions

This is the Android Software Development Kit License Agreement ("Agreement").

1. Introduction

1.1 The Android Software Development Kit (referred to as "the Kit") includes files, packaged APIs, and Google APIs add-ons) is licensed to you under forms of a legally binding contract between you and Google.

1.2 "Android" means the Android software stack found at the following URL: <http://source.android.com/>, as last updated.

1.3 A "compatible implementation" means any And

I have read and agree with the above terms and conditions.

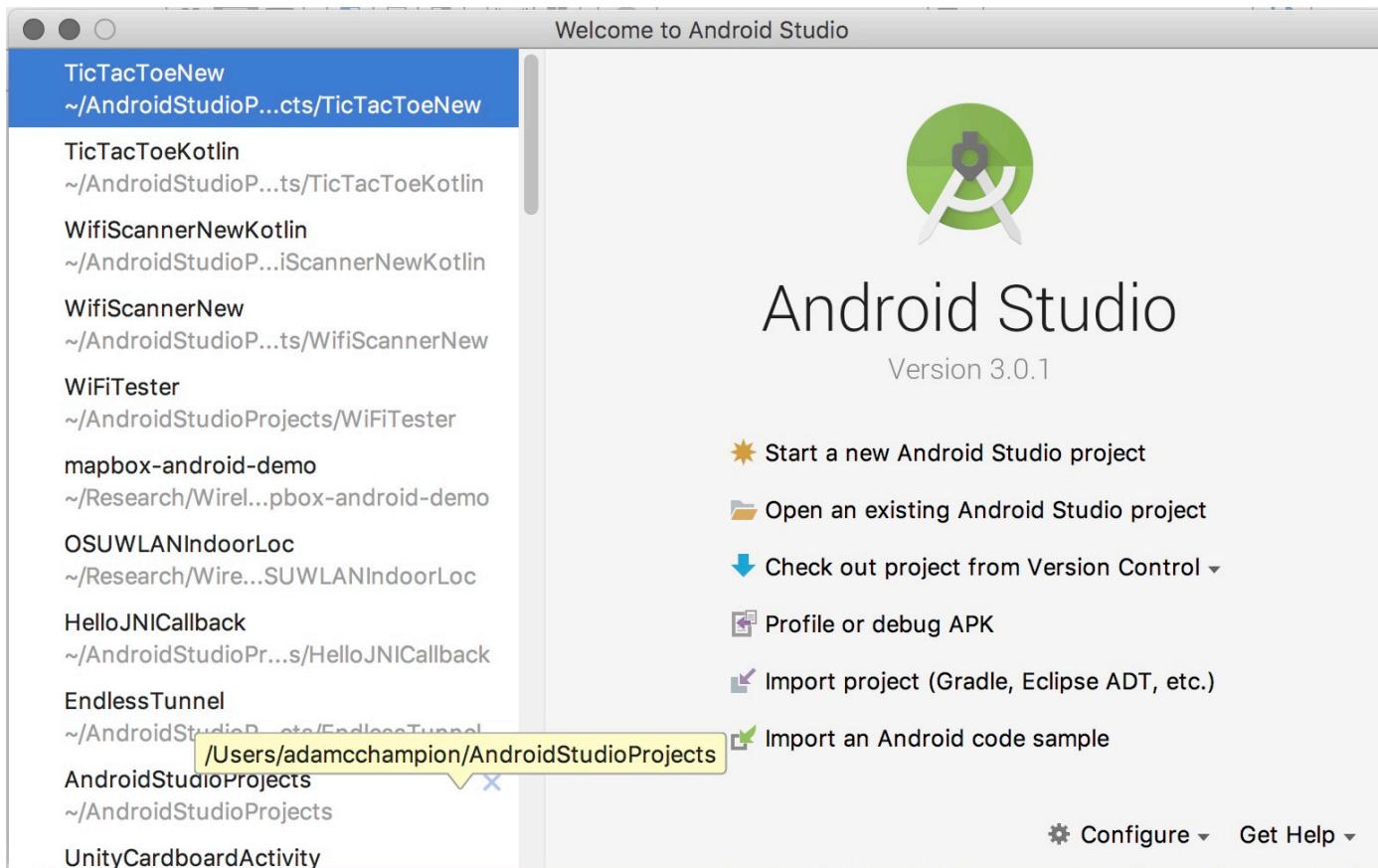
DOWNLOAD ANDROID STUDIO FOR MAC

This site uses cookies to store your preferences for site-specific

6

Getting Started (3)

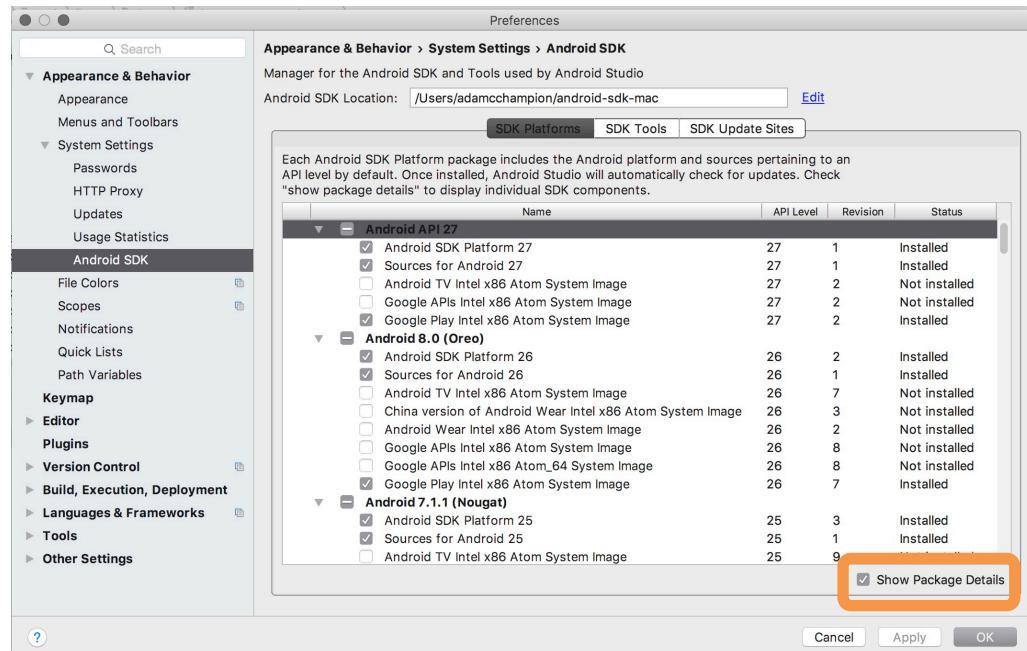
- Install Android Studio directly (Windows, Mac); unzip to directory android-studio, then run ./android-studio/bin/studio.sh (Linux)
- You should see this:



Getting Started (4)

- Strongly recommend testing with real Android device
 - Android emulator: *very* slow
 - Faster emulator: Genymotion [14], [15]
 - Install USB drivers for your Android device!
- Bring up Android SDK Manager
 - Recommend installing Android 5.x–7.x APIs, Google support repository, Google Play services
 - Don't worry about ARM, MIPS, Auto, TV system images

Android Studio menu → Preferences... or File → Settings...



Now you're ready for Android development!

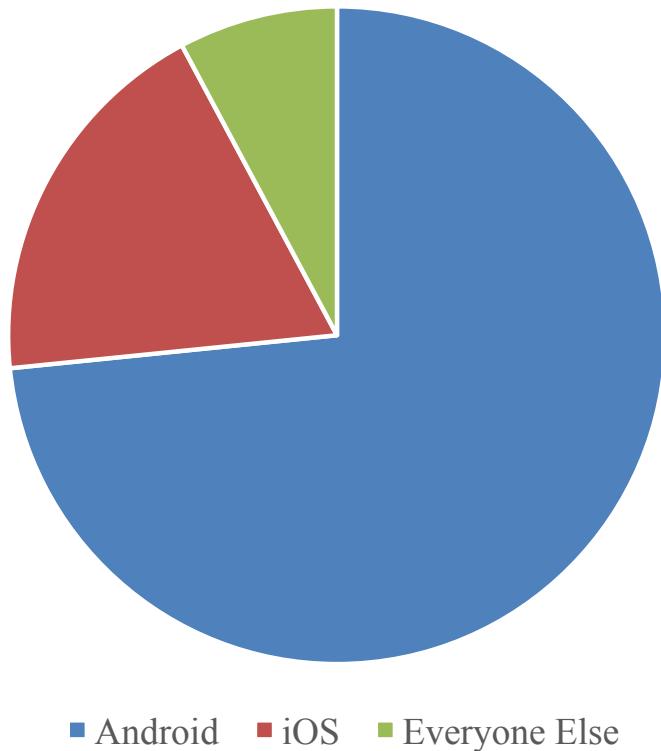
Outline

- Getting Started
- **Android Programming**

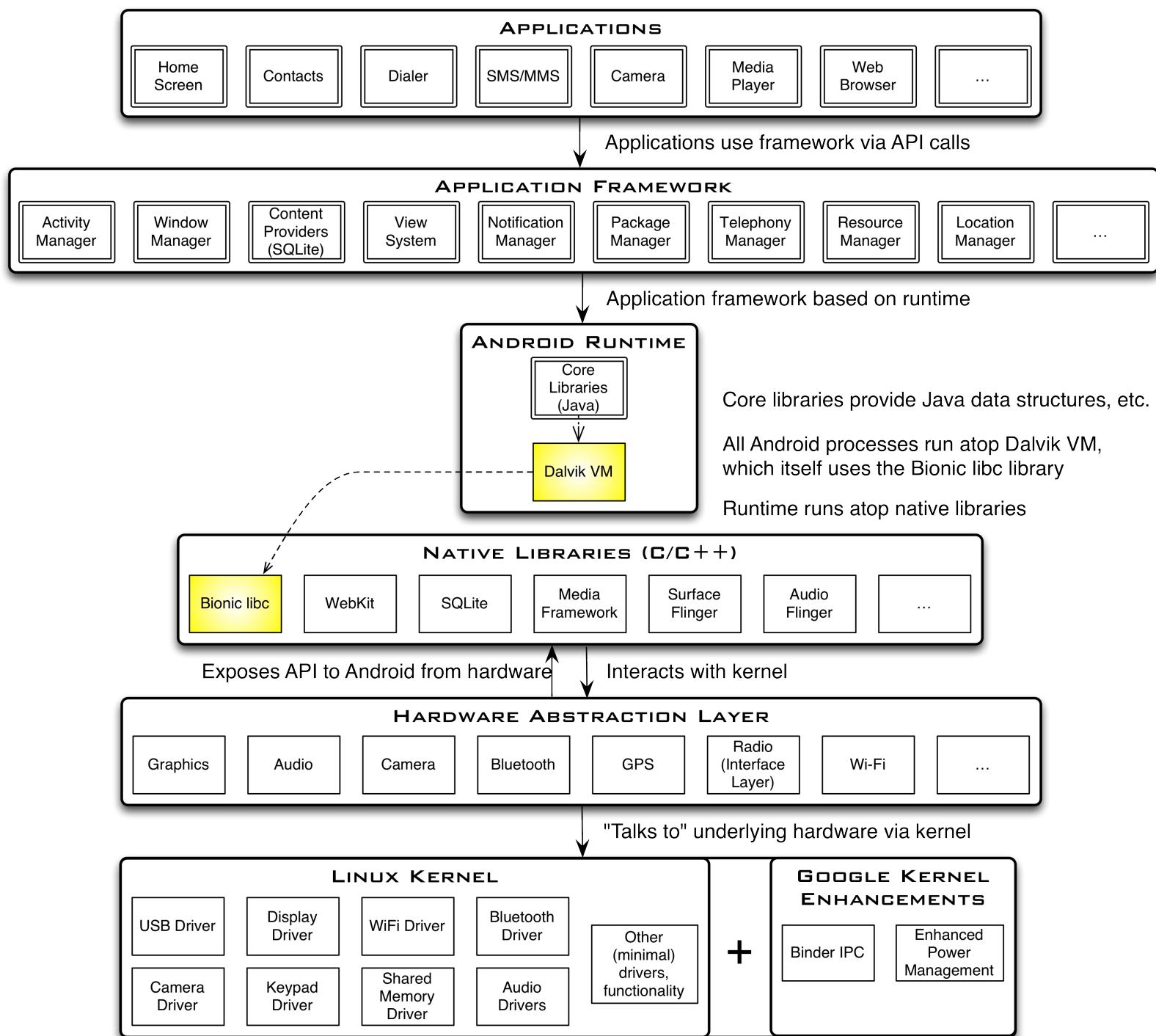
Introduction to Android

- 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]

Mobile OS Market Share Worldwide (Jul. 2017)

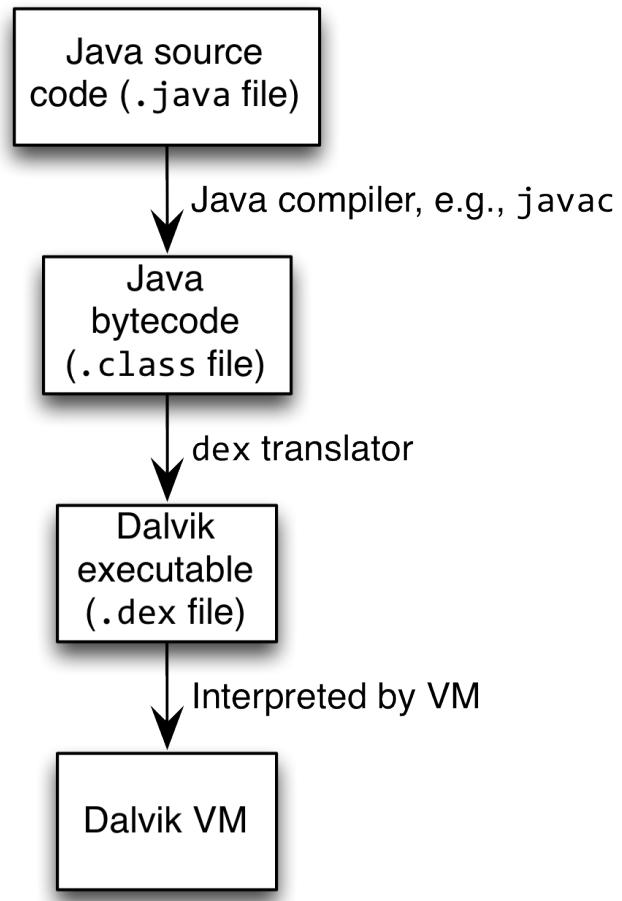


Source: [8]



Android Highlights (1)

- Android apps execute on Dalvik VM, a “clean-room” implementation of JVM
 - Dalvik optimized for efficient execution
 - Dalvik: register-based VM, unlike Oracle’s stack-based JVM
 - Java .class bytecode translated to Dalvik EXecutable (DEX) bytecode, which Dalvik interprets



Android Highlights (2)

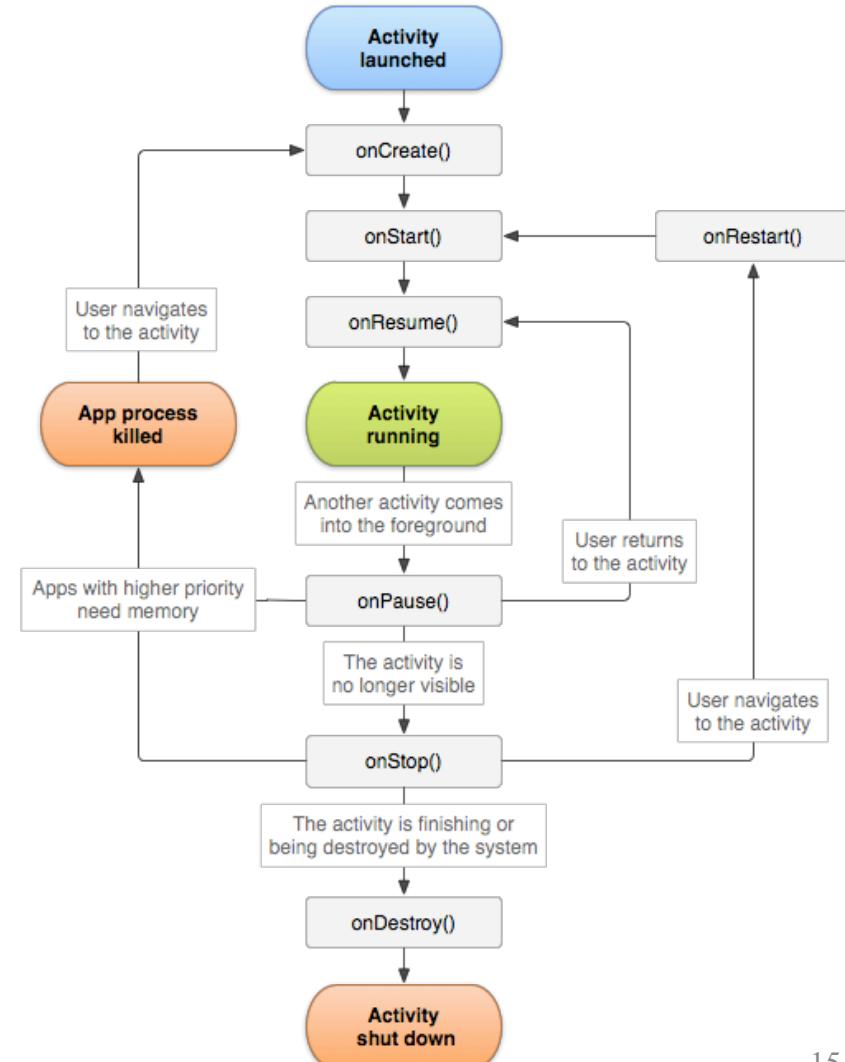
- Android apps written in Java 7
 - Actually, a Java dialect (Apache Harmony)
 - Everything we've learned still holds
- Apps use four main components:
 - Activity: A “single screen” that’s visible to user
 - Service: Long-running background “part” of app (*not* separate process or thread)
 - ContentProvider: 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”, and responds accordingly

App Manifest

- Every 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
 - External libraries to which app is linked, e.g., Google Maps library

Activity Lifecycle

- Activity: key building block of Android apps
- Extend Activity class, override onCreate(), onPause(), onResume() methods
- Dalvik VM can stop any Activity without warning, so saving state is important!
- Activities need to be “responsive”, otherwise Android shows user “App Not Responsive” warning:
 - Place lengthy operations in Runnable Threads, AsyncTasks

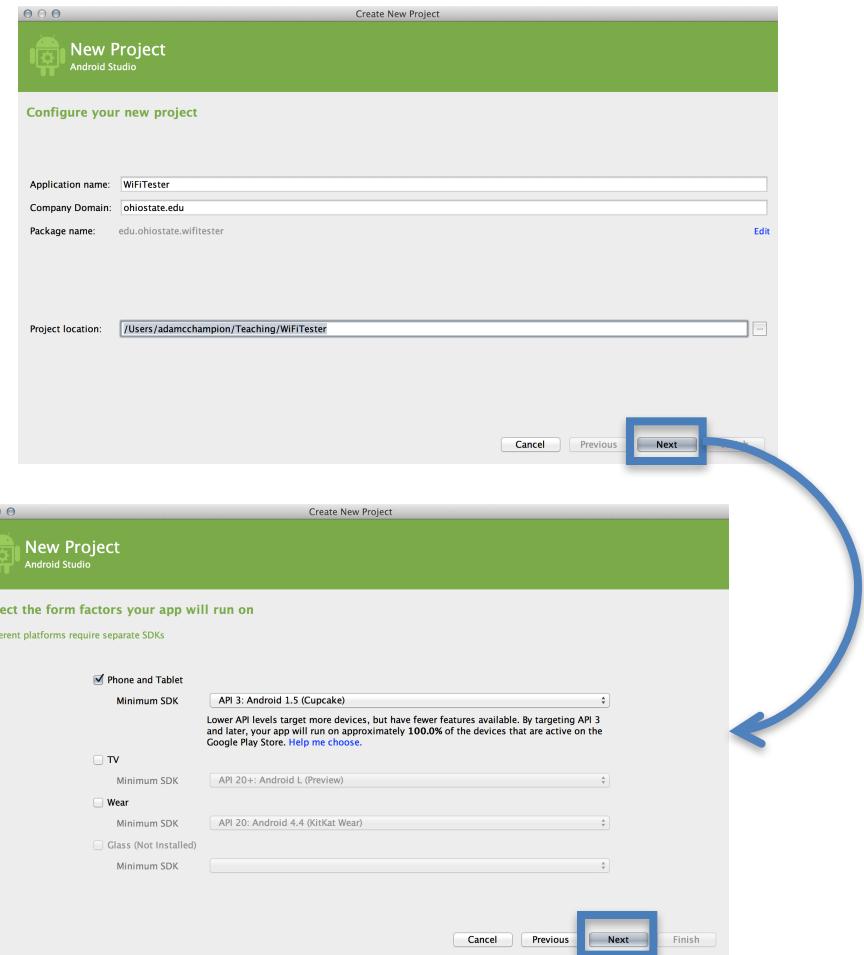


App Creation Checklist

- If you own an Android device:
 - Ensure drivers are installed
 - Enable developer options on device under *Settings*, specifically *USB Debugging*
 - Android 4.2+: Go to *Settings*→*About phone*, press *Build number* 7 times to enable developer options
- For Android Studio:
 - Under File→*Settings*→*Appearance*, enable “Show tool window bars”; the *Android* view shows LogCat, devices
 - Programs should log states via `android.util.Log`'s `Log.d(APP_TAG_STR, "debug")`, where `APP_TAG_STR` is a `final String` tag denoting your app
 - Other commands: `Log.e()` (error); `Log.i()` (info); `Log.w()` (warning); `Log.v()` (verbose) – same parameters

Creating Android App (1)

- Creating Android app project in Android Studio:
 - Go to *File*→*New Project*
 - Enter app, project name
 - Choose package name using “reverse URL” notation, e.g., `edu.osu.myapp`
 - Select APIs for app, then click Next



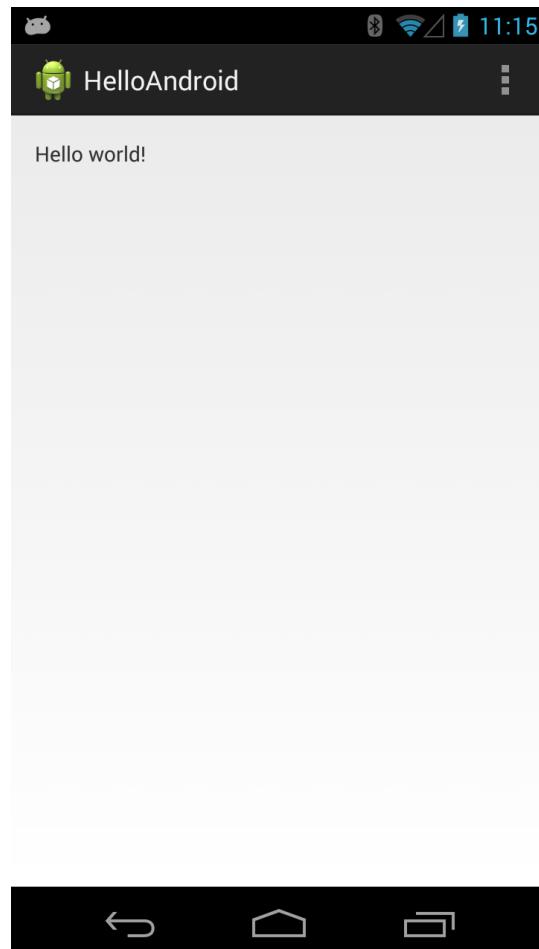
Creating Android App (2)

- Determine what kind of Activity to create; then click Next
 - We'll choose a Blank Activity for simplicity
- Enter information about your Activity, then click Finish
- 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 version, or perhaps <http://www.genymotion.com/>
- Run the app: press “Run” button in toolbar



Underlying Source Code

src/.../MainActivity.java

```
package edu.osu.helloandroid;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

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`s are quite complicated. See [13] for details

The App Manifest

AndroidManifest.xml

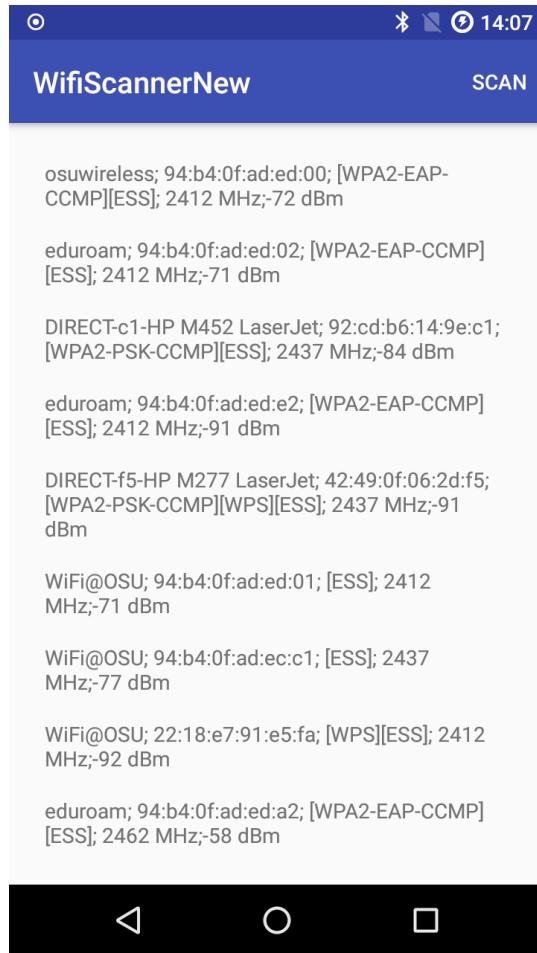
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.osu.helloandroid"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="edu.osu.helloandroid.MainActivity"
            android:label="@string/app_name" >
            <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 logic (flexibility)



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(@NonNull 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) {
        Log.e(TAG, "Error setting up Wi-Fi");
    }
}
```

Underlying Source Code (2)

- Get system `WifiManager`
- Register Broadcast Receiver to listen for `WifiManager`'s “finished scan” system event (expressed as `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) {
    super.onCreateOptionsMenu(menu, inflater);
    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}, PERMISSION_REQUEST_LOCATION); }}}

public void onRequestPermissionsResult(int requestCode, @NonNull 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 points of entry: no `main()` method
 - Cannot “sleep” in Android
 - During each entrance, certain `Objects` may be `null`
 - Defensive programming is very useful to avoid crashes, e.g.,
`if (!(myObj == null)) { // do something }`
- Java concurrency techniques are required
 - Don’t block the “main” thread in Activities
 - Implement long-running tasks such as network connections asynchronously, e.g., as `AsyncTasks`
 - Recommendation: read [4]; chapter 20 [10]; [11]
- Logging state via `android.util.Log` throughout app is essential when debugging (finding root causes)
- Better to have “too many” permissions than too few
 - Otherwise, app crashes due to security exceptions!
 - Remove “unnecessary” permissions before releasing app to public
- Event handling in Android GUIs entails many listener `Objects`

Concurrency: Threads (1)

- Thread: program unit (within process) executing 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 can be difficult to synchronize, especially with UI thread in Activity, Fragment; AsyncTasks more suitable

Concurrency: AsyncTasks

- `AsyncTask` encapsulates asynchronous task that interacts with UI thread in `Activity`:

```
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` with your own class
- Documentation at <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. Kaepler, *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>
9. Android Open Source Project, <http://www.android.com>

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/>