Tutorial: Setup for Android Development

Adam C. Champion

CSE 5236: Mobile Application Development

Autumn 2015

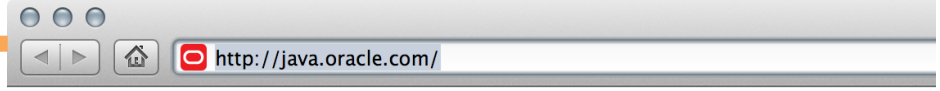
Based on material from C. Horstmann [1], J. Bloch [2], C. Collins et al. [4], M.L. Sichitiu (NCSU), V. Janjic (Imperial College London), CSE 2221 (OSU), and other sources

Outline

- **Getting Started**
- Android Programming

Getting Started (1)

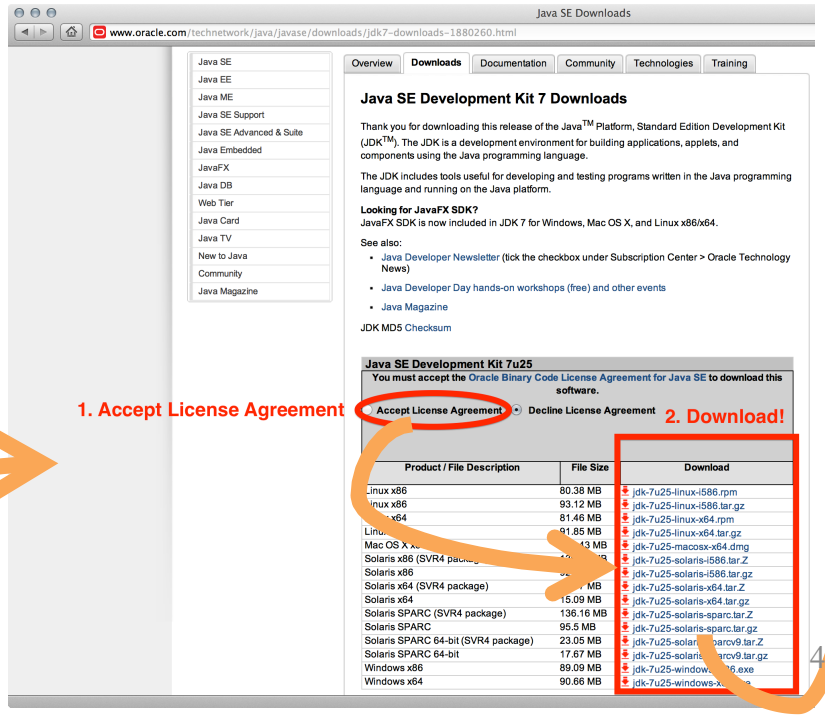
- Need to install Java Development Kit (JDK) to write Java (and Android) programs
  - **Do not** install Java Runtime Environment (JRE); JDK and JRE are different!
- Can download the JDK for your OS at http://java.oracle.com
- Alternatively, for OS X, Linux:
  - OS X:
    - Open `/Applications/Utilities/Terminal.app`
    - 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/


Oracle Technology Network for Java Developers
www.oracle.com/technetwork/java/index.html
ORACLE
Sign In/Register for Account
Help
Select Country/Region
Communities
I am a...
I want to...
PRODUCTS AND SERVICES
SOLUTIONS
DOWNLOADS
STORE
SUPPORT
TRAINING
PARTNERS
ABOUT
Oracle Technology Network > Java
Java
1 2 3
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

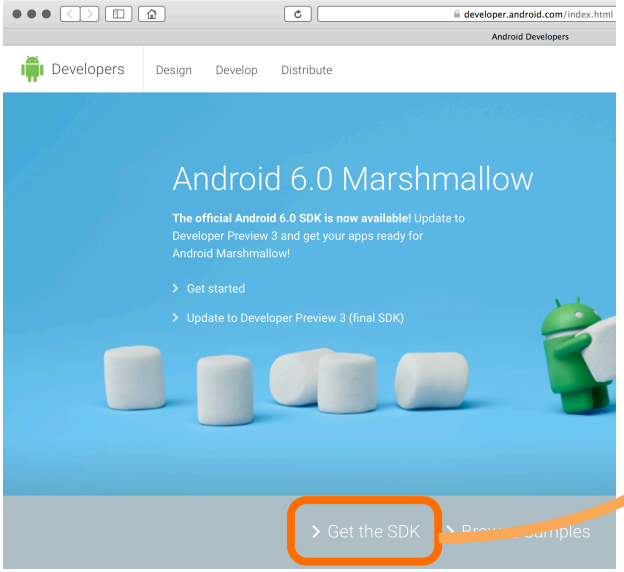
Install!

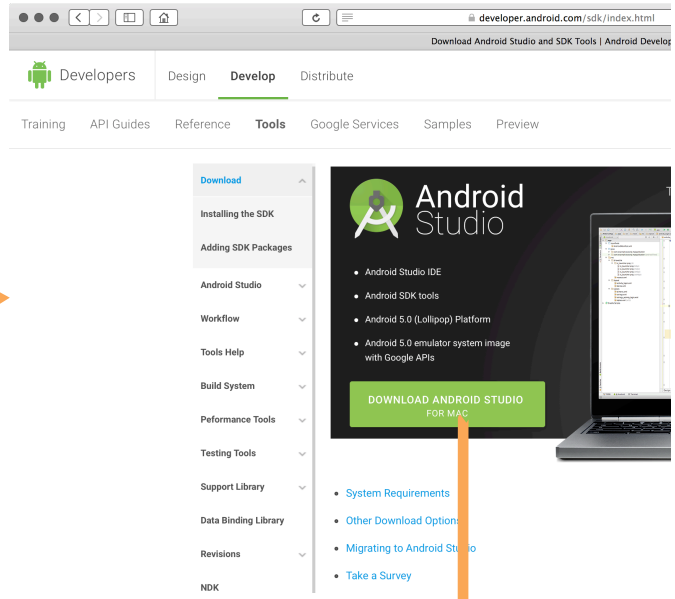

Java SE Dow
www.oracle.com/technetwork/java/javase/downloads/index.html
ORACLE
Sign In/Register for Account
Help
PRODUCTS AND SERVICES
SOLUTIONS
DOWNLOADS
STORE
Oracle Technology Network > Java > Java SE > Downloads
Java SE
Java EE
Java ME
Java SE Support
Java SE Advanced & Suite
Java Embedded
JavaFX
Java DB
Web Tier
Java Card
Java TV
New to Java
Community
Java Magazine
Overview
Downloads
Documentation
Java SE Downloads
Next Releases (Early Access)
DOWNLOAD
Java Platform (JDK) 7u25

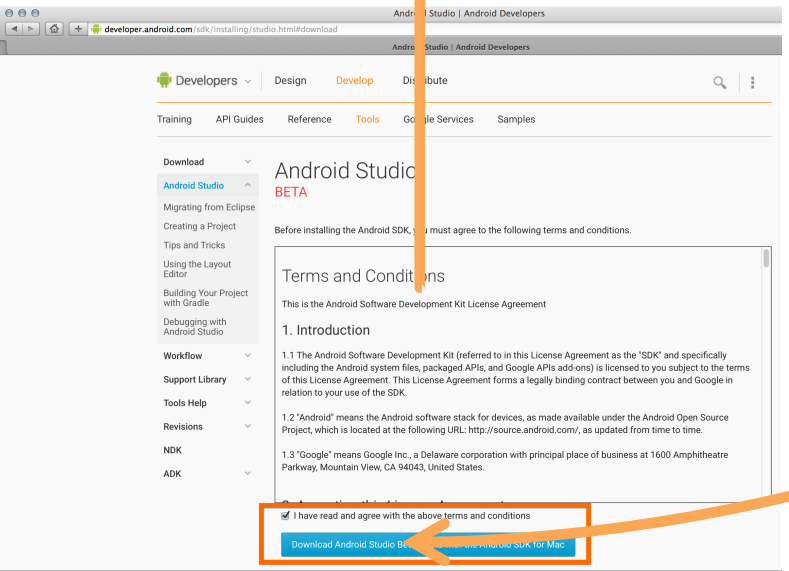

Java SE Downloads
www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html
Java SE
Java EE
Java ME
Java SE Support
Java SE Advanced & Suite
Java Embedded
JavaFX
Java DB
Web Tier
Java Card
Java TV
New to Java
Community
Java Magazine
Overview
Downloads
Documentation
Community
Technologies
Training
Java SE Development Kit 7 Downloads
Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.
The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.
Looking for JavaFX SDK?
JavaFX SDK is now included in JDK 7 for Windows, Mac OS X, and Linux x86/x64.
See also:
Java Developer Newsletter (tick the checkbox under Subscription Center > Oracle Technology News)
Java Developer Day hands-on workshops (free) and other events
Java Magazine
JDK MD5 Checksum
Java SE Development Kit 7u25
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.
1. Accept License Agreement
Accept License Agreement
Decline License Agreement
2. Download!
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 73 MB jdk-7u25-macosx-x64.dmg
Solaris x86 (SVR4 package) jdk-7u25-solaris-i586.tar.Z
Solaris x86 jdk-7u25-solaris-i586.tar.gz
Solaris x64 (SVR4 package) 15.09 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-sparcv9.tar.Z
Solaris SPARC 64-bit 17.67 MB jdk-7u25-solaris-sparcv9.tar.gz
Windows x86 89.09 MB jdk-7u25-windows-i586.exe
Windows x64 90.66 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)


developer.android.com/index.html
Android Developers
Developers
Design
Develop
Distribute
Android 6.0 Marshmallow
The official Android 6.0 SDK is now available! Update to Developer Preview 3 and get your apps ready for Android Marshmallow!
Get started
Update to Developer Preview 3 (final SDK)
Get the SDK

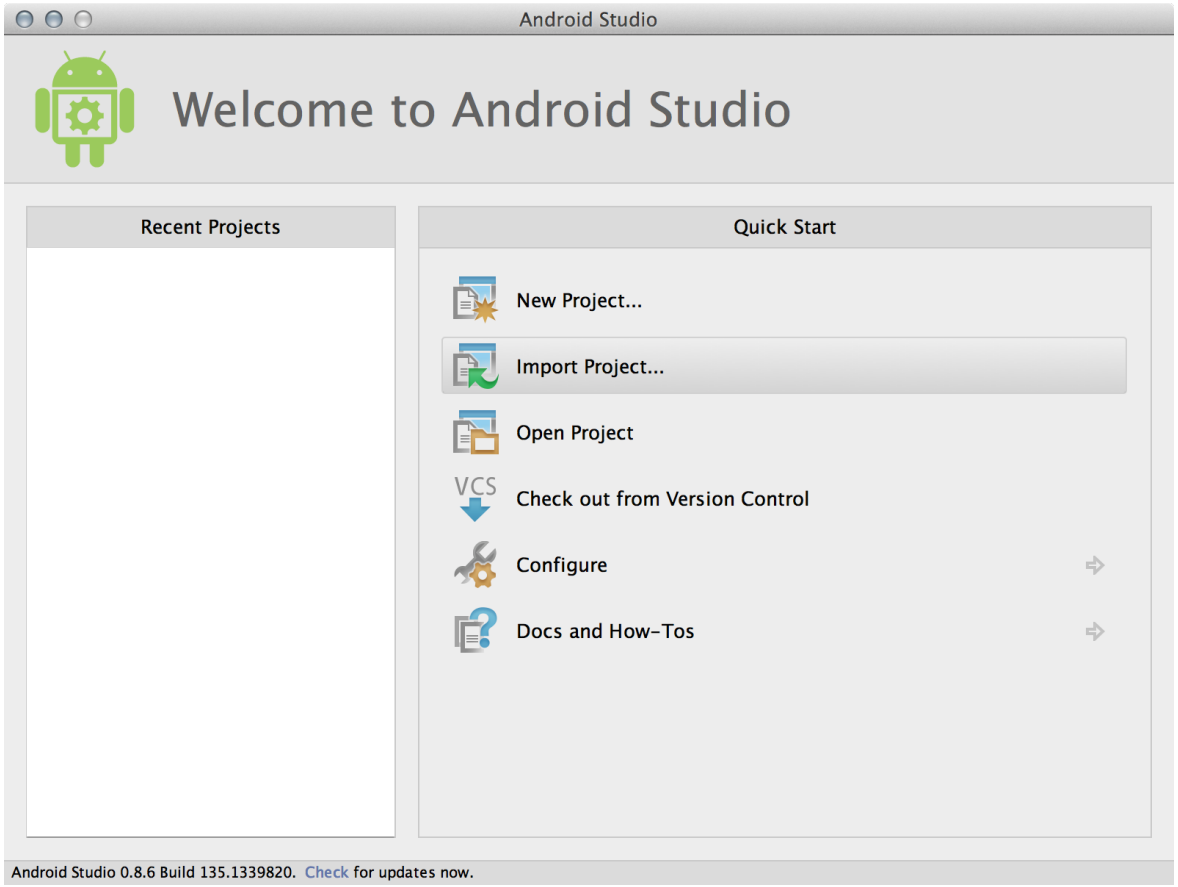

developer.android.com/sdk/index.html
Developers
Design
Develop
Distribute
Training
API Guides
Reference
Tools
Google Services
Samples
Preview
Download
Installing the SDK
Adding SDK Packages
Android Studio
Workflow
Tools Help
Build System
Peformance Tools
Testing Tools
Support Library
Data Binding Library
Revisions
NDK
Android Studio
Android Studio IDE
Android SDK tools
Android 5.0 (Lollipop) Platform
Android 5.0 emulator system image with Google APIs
DOWNLOAD ANDROID STUDIO
FOR MAC
System Requirements
Other Download Options
Migrating to Android Studio
Take a Survey


developer.android.com/sdk/installing/studio.html#download
Developers
Design
Develop
Training
API Guides
Reference
Tools
Samples
Download
Android Studio
Migrating from Eclipse
Creating a Project
Tips and Tricks
Using the Layout Editor
Building Your Project with Gradle
Debugging with Android Studio
Workflow
Support Library
Tools Help
Revisions
NDK
ADK
Android Studio
BETA
Terms and Conditions
This is the Android Software Development Kit License Agreement
1. Introduction
1.1 The Android Software Development Kit (referred to in this License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of this License Agreement. This License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.
1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: http://source.android.com/, as updated from time to time.
1.3 "Google" means Google Inc., a Delaware corporation with principal place of business at 1600 Amphitheatre Parkway, Mountain View, CA 94043, United States.
I have read and agree with the above terms and conditions

Install!

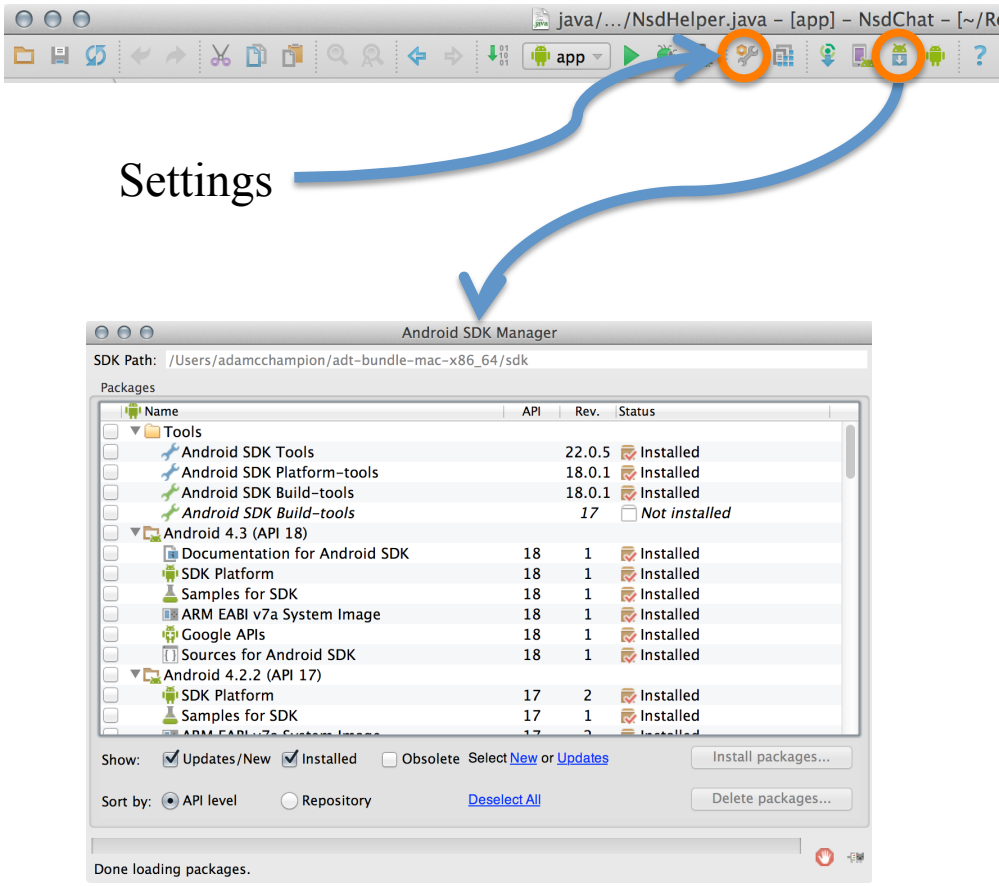
Getting Started (3)

- 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
  - Recommended: Install Android 6.0, 5.x, 4.x, 2.3.3 APIs, Google support repository, Google Play services
  - Don't worry about Intel x86, MIPS, Auto, TV system images



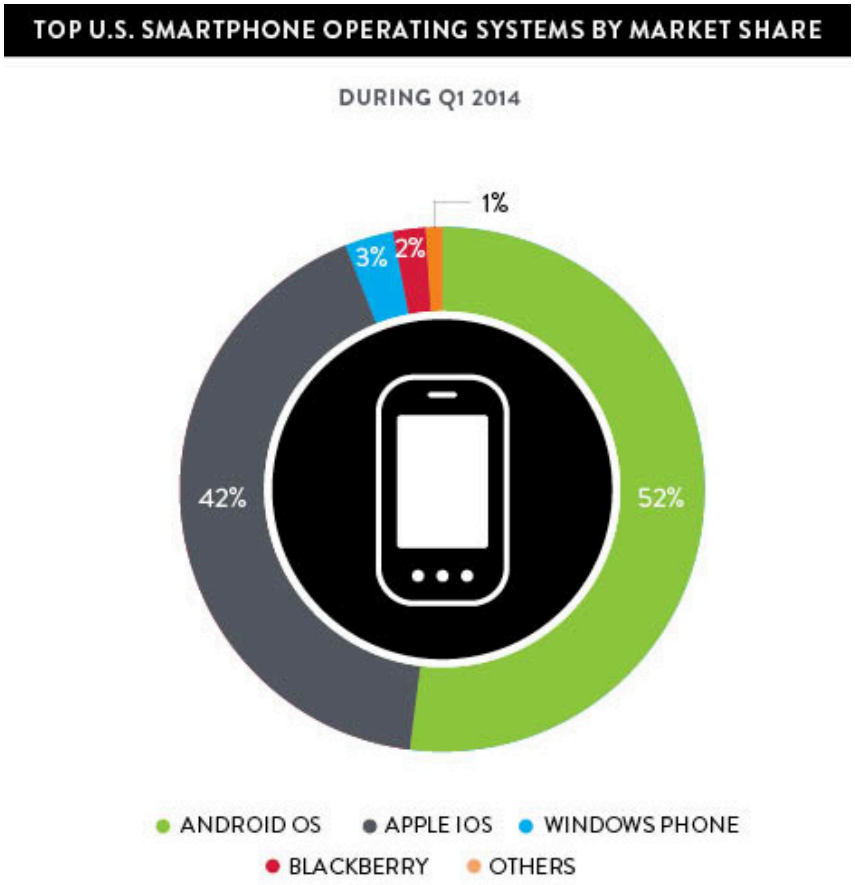
Now you're ready for Android development!

Outline

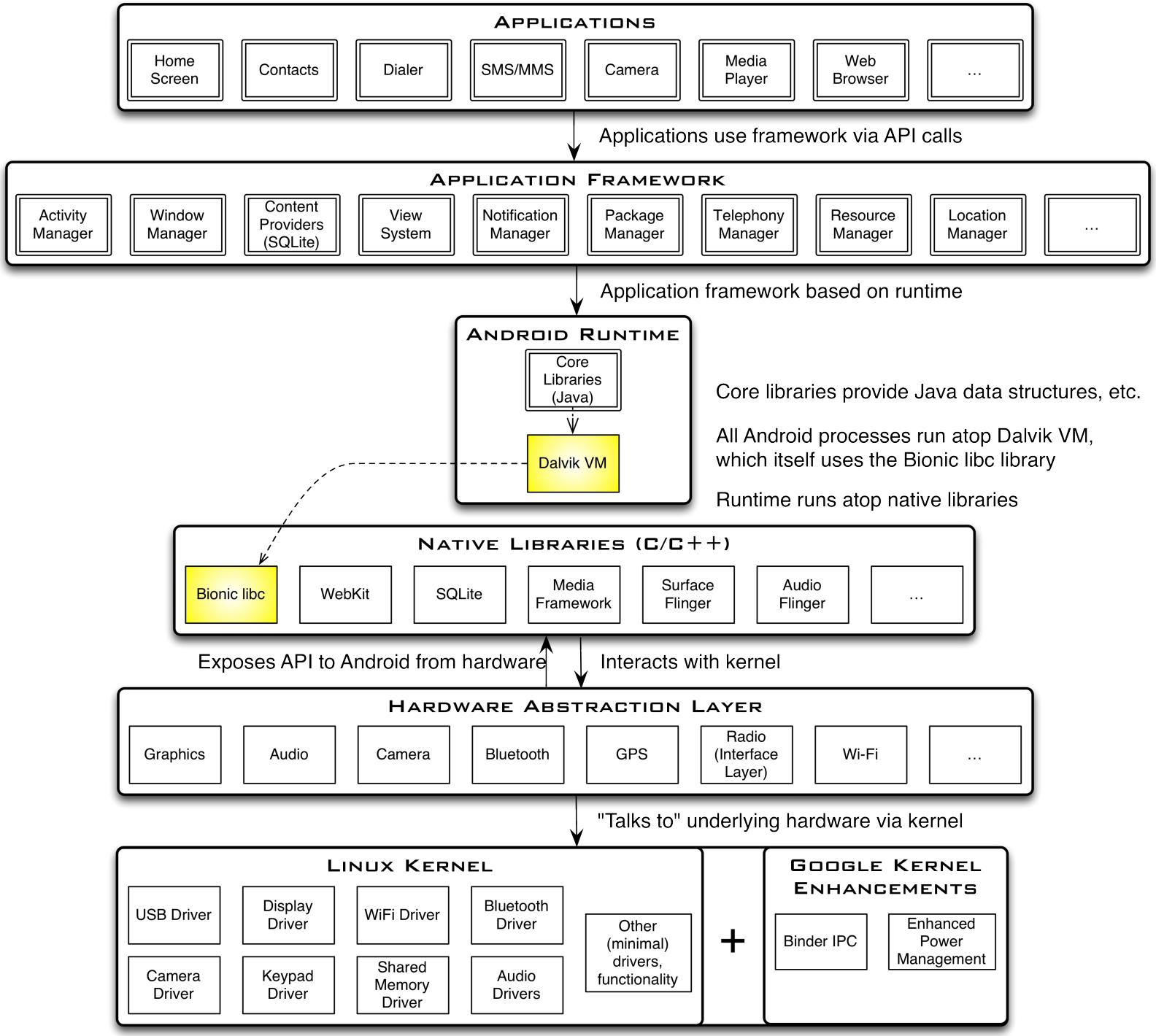
- Getting Started
- **Android Programming**

Introduction to Android

- Popular mobile device OS: 52% of U.S. smartphone market [8]
- Developed by Open Handset Alliance, led by Google
- Google claims 900,000 Android device activations [9]

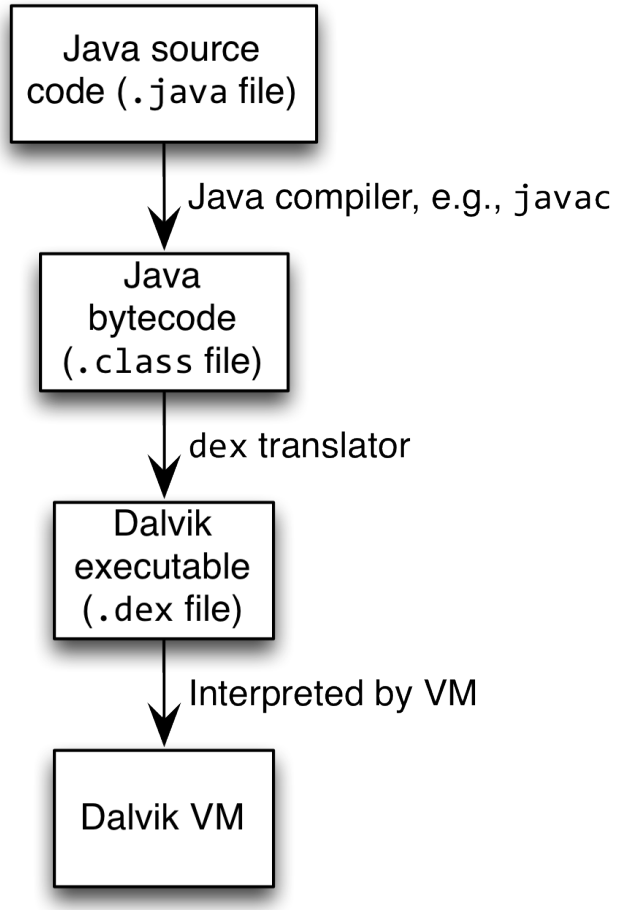


Source: [8]


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 (SQLite)
View System
Notification Manager
Package Manager
Telephony Manager
Resource Manager
Location Manager
…
Application framework based on runtime
Android Runtime
Core Libraries (Java)
Dalvik VM
Core libraries provide Java data structures, etc.
All Android processes run atop Dalvik VM, which itself uses the Bionic libc library
Runtime runs atop native 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
Radio (Interface Layer)
Wi-Fi
…
"Talks to" underlying hardware via kernel
Linux Kernel
USB Driver
Display Driver
WiFi Driver
Bluetooth Driver
Camera Driver
Keypad Driver
Shared Memory Driver
Audio Drivers
Other (minimal) drivers, functionality
+
Google Kernel Enhancements
Binder IPC
Enhanced Power Management

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)

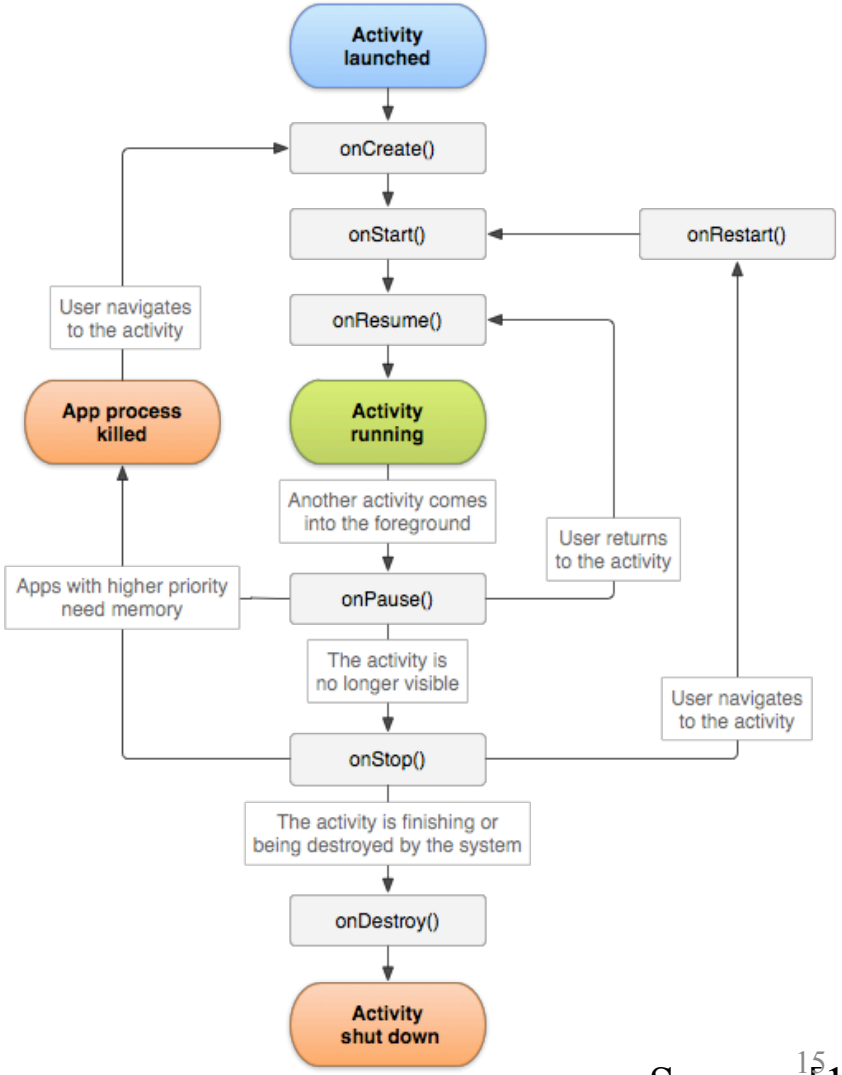
- Android apps written in Java 5
  - Actually, a Java dialect (Apache Harmony)
  - Everything we've learned still holds
- Apps use four main components:
  - `Activity`: A "single screen" that's visible to user
  - `Service`: Long-running background "part" of app (*not* separate process or thread)
  - `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`



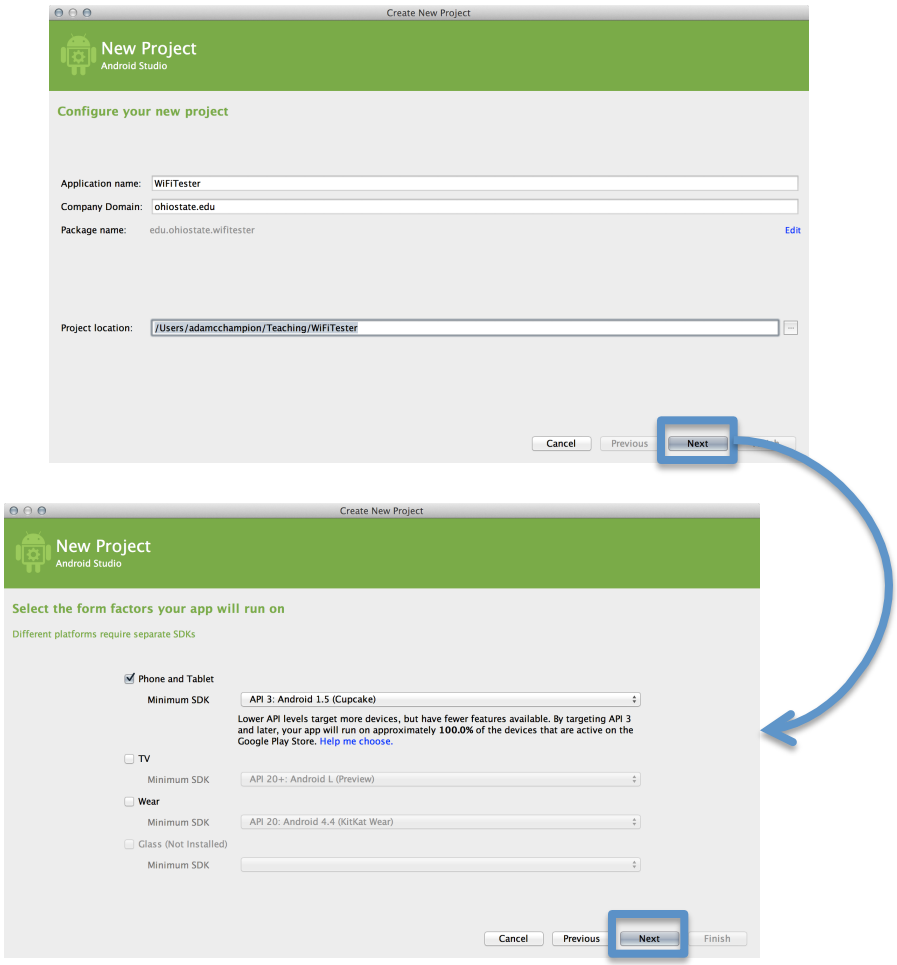
Source: [12]

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

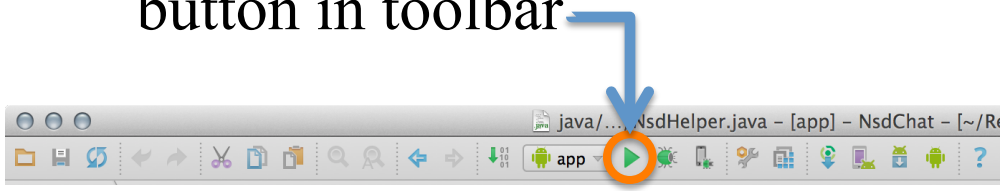


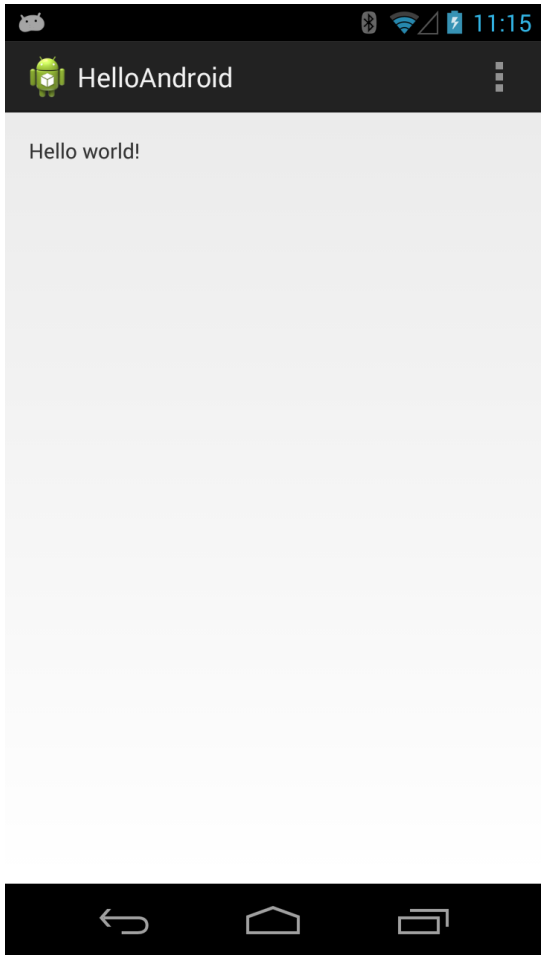
Creating Android App (2)

- 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>
```

– `RelativeLayouts` are quite complicated. See [13] for details

The App Manifest

AndroidManifest.xml

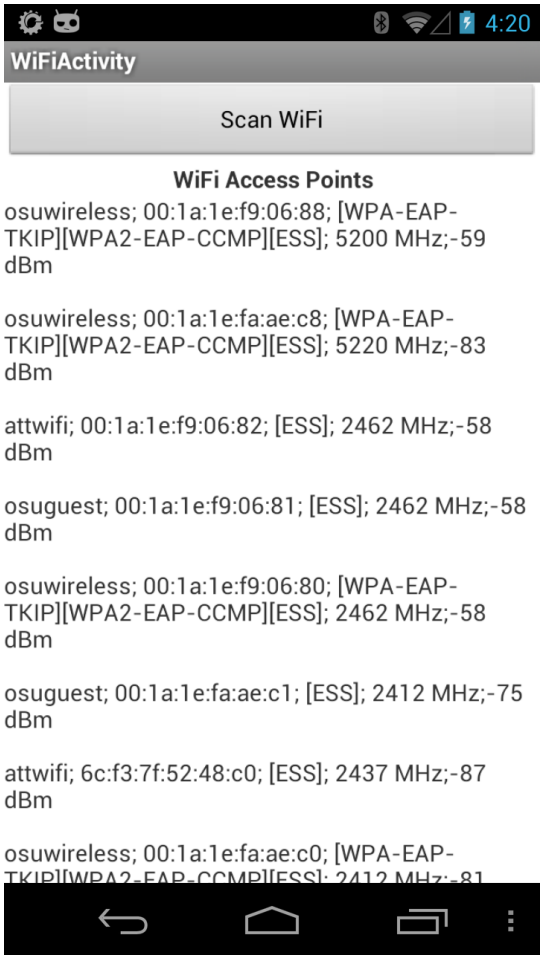
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.osu.helloandroid"
    android:versionCode="1"
    android:versionName="1.0" >

    <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 Tester (code on class website)
- Press a button, scan for WiFi access points (APs), display them



Underlying Source Code (1)

```
@Override
public void onCreate(Bundle savedInstanceState)
{
	super.onCreate(savedInstanceState);
	setContentView(R.layout.activity_wi_fi);

	// Set up WifiManager.
	mWifiManager = (WifiManager) getSystemService(Context.WIFI_SERVICE);

	// Create listener object for Button. When Button is pressed, scan for
	// APs nearby.
	Button button = (Button) findViewById(R.id.button);
	button.setOnClickListener(new View.OnClickListener()
	{
		public void onClick(View v)
		{
			boolean scanStarted = mWifiManager.startScan();

			// If the scan failed, log it.
			if (!scanStarted) Log.e(TAG, "WiFi scan failed...");
		}
	});

	// Set up IntentFilter for "WiFi scan results available" Intent.
	mIntentFilter = new IntentFilter();
	mIntentFilter.addAction(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION);
}
```

Underlying Source Code (2)

- Code much more complex
- First get system `WifiManager`
- Create listener `Object` for button that performs scans
- We register Broadcast Receiver, `mReceiver`, to listen for `WifiManager`'s "finished scan" system event (expressed as `Intent WifiManager.SCAN_RESULTS_AVAILABLE_ACTION`)
- Unregister Broadcast Receiver when leaving `Activity`

```
@Override
protected void onResume()
{
    super.onResume();
    registerReceiver(mReceiver,
mIntentFilter);
}

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

The Broadcast Receiver

```
private final BroadcastReceiver mReceiver = new BroadcastReceiver()
{
      @Override
      public void onReceive(Context context, Intent intent)
      {
            String action = intent.getAction();
            if (WifiManager.SCAN_RESULTS_AVAILABLE_ACTION.equals(action))
            {
                  Log.e(TAG, "Scan results available");
                  List<ScanResult> scanResults = mWifiManager.getScanResults();
                  mApStr = "";
                  for (ScanResult result : scanResults)
                  {
                        mApStr = mApStr + result.SSID + "; ";
                        mApStr = mApStr + result.BSSID + "; ";
                        mApStr = mApStr + result.capabilities + "; ";
                        mApStr = mApStr + result.frequency + " MHz;";
                        mApStr = mApStr + result.level + " dBm\n\n";
                  }
                  // Update UI to show all this information.
                  setTextView(mApStr);
            }
      }
};
```

User Interface

Updating UI in code

```
private void setTextView(String str)
{
    TextView tv = (TextView)
findViewById(R.id.textview);
    tv.setMovementMethod(new
ScrollingMovementMethod());
    tv.setText(str);
}
```

- This code simply has the UI display all collected WiFi APs, makes the text information scrollable

UI Layout (XML)

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/
res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/button"
        android:text="@string/button_text"/>

    <TextView
        android:id="@+id/header"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/ap_list"
        tools:context=".WiFiActivity"
        android:textStyle="bold"
        android:gravity="center">
    </TextView>

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        tools:context=".WiFiActivity"
        android:id="@+id/textview"
         android:scrollbars="vertical">
    </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 contains code to be executed
  - 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();
```

- Problem: this is cumbersome unless Thread code is reused

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. AsyncTasks are better suited for this

Concurrency: AsyncTasks

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

```
public class AsyncTask<Params, Progress, Result>
{
    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. 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. Nielsen Co., "Smartphone Milestone: Half of Mobile Subscribers Ages 55+ Own Smartphones", 22 Apr. 2014, http://www.nielsen.com/us/en/insights/news/2014/smartphone-milestone-half-of-americans-ages-55-own-smartphones.html
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/