Android “Master” Class 2013

Lecture 1, October 10, 2013

Admistration

- No homework or tests
- No attendence taking
- Purely for your benefit
- Pay attention and follow along to succeed
- Have fun!

Admistration

- Lectures will be recorded
- Check Ctools for annoucements and resources
- Check Github for code samples

Agenda

- Intro to Android
- Useful Tools
- Application Components
- Activities
- Views and ViewGroups
- Component Communication
- “Hello, world!”

Agenda

- **Intro to Android**
- Useful Tools
- Application Components
- Activities
- Views and ViewGroups
- Component Communication
- “Hello, world!”

Intro to Android

- Android, Inc. created by Andy Rubin in 2003
- Acquired by Google in 2005

Intro to Android

- Android, Inc. created by Andy Rubin in 2003
- Acquired by Google in 2005
- Lawsuit trouble with Sun
  - Dalvik JVM vs Sun JVM

Intro to Android

- Android 1.0
- 1.5 Cupcake
- 1.6 Donut
- 2.0-2.1 Éclair
- 2.2 Froyo
- 2.3 Gingerbread
- 3.0-3.2 Honeycomb
- 4.0 Ice Cream Sandwich
- 4.1-4.3 Jelly Bean
- 4.4 KitKat

Intro to Android

- Android 1.0
- 1.5 Cupcake
- 1.6 Donut
- 2.0-2.1 Éclair
- 2.2 Froyo
- 2.3 Gingerbread
- 3.0-3.2 Honeycomb
- 4.0 Ice Cream Sandwich
- 4.1-4.3 Jelly Bean
- 4.4 KitKat
  - First proprietary naming

Intro to Android

- Why Android?
  - Over half a billion Android devices activiated[1]
  - Over half of smartphones in US and worldwide run Android
  - Not just smartphones run Android; watches, cameras, even cars!
  - Mobile is the future

1 According to CNET News
2 According to comScore study

Intro to Android

- Android development is fun!
  - The system is open source
  - Very few developer restrictions and limitations

Intro to Android

- Android development is fun!
  - The system is open source
  - Very few developer restrictions and limitations
- Android development is challenging
  - Multi-threaded environment
  - Very few developer restrictions and limitations

Intro to Android

- Android development is fun!
  - The system is open source
  - Very few developer restrictions and limitations
- Android development is challenging
  - Multi-threaded environment
  - Very few developer restrictions and limitations
    - Easy to write poor-performing, data-sucking, battery-draining apps

Intro to Android

- Java
  - The all the libraries and frameworks are all written in Java (almost)

Intro to Android

- Java
  - The all the libraries and frameworks are all written in Java
  - Notes about Java

Intro to Android

- Java
  - The all the libraries and frameworks are all written in Java
  - Notes about Java
    - Everything is a pointer, but called reference

Intro to Android

- Java
  - The all the libraries and frameworks are all written in Java
  - Notes about Java
    - Everything is a pointer, but called reference
    - Garbage collection

Intro to Android

- Java
  - The all the libraries and frameworks are all written in Java
  - Notes about Java
    - Everything is a pointer, but called reference
    - Garbage collection
    - Memory leaks DO occur

Agenda

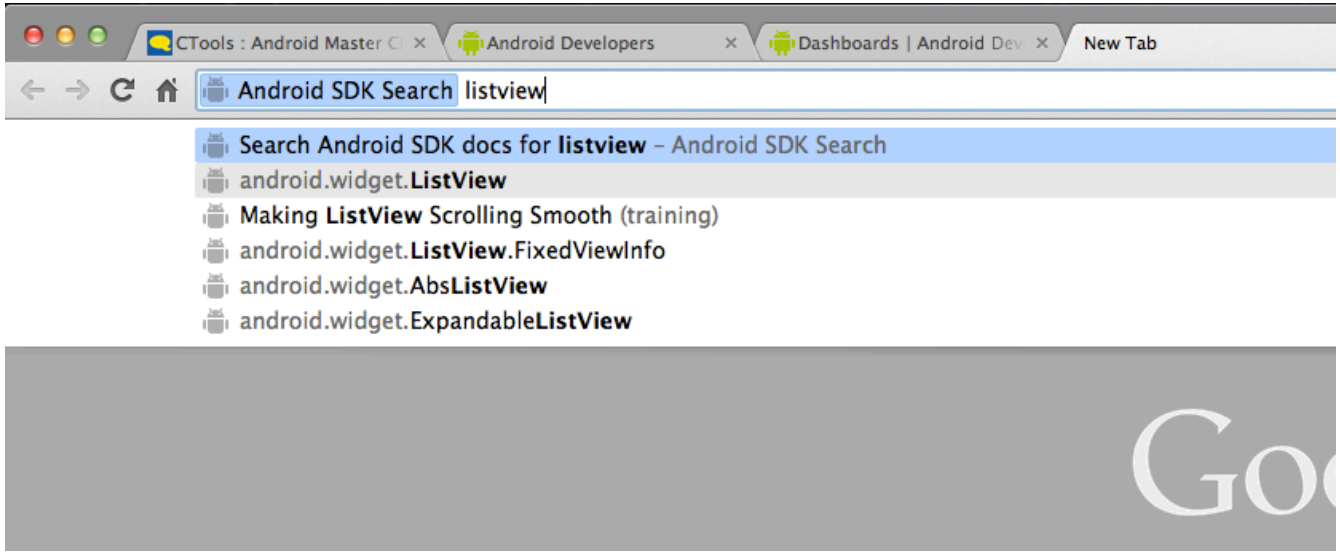
- Intro to Android
- **Useful Tools**
- Application Components
- Activities
- Views and ViewGroups
- Component Communication
- “Hello, world!”

Useful Tools

- developer.android.com
  - Most useful
  - Not just Javadoc
  - Many great tutorials

Useful Tools

- Android SDK Reference Search
  - Awesome Chrome extension
  - Searches the developer site directly



Android Open Source Project

- Known as AOSP
- Whole OS and framework is publicly available
- Code for built-in apps (phone, email, SMS, etc.)
  - Not Google apps (Gmail, Play Store, etc.)
- Acess through developer site or github.com/android

Useful Tools

- Emulator is PAINFULLY slow
- Genymotion is a great alternative
- Uses VirtualBox to run an emulator on a very fast machine

Useful Tools

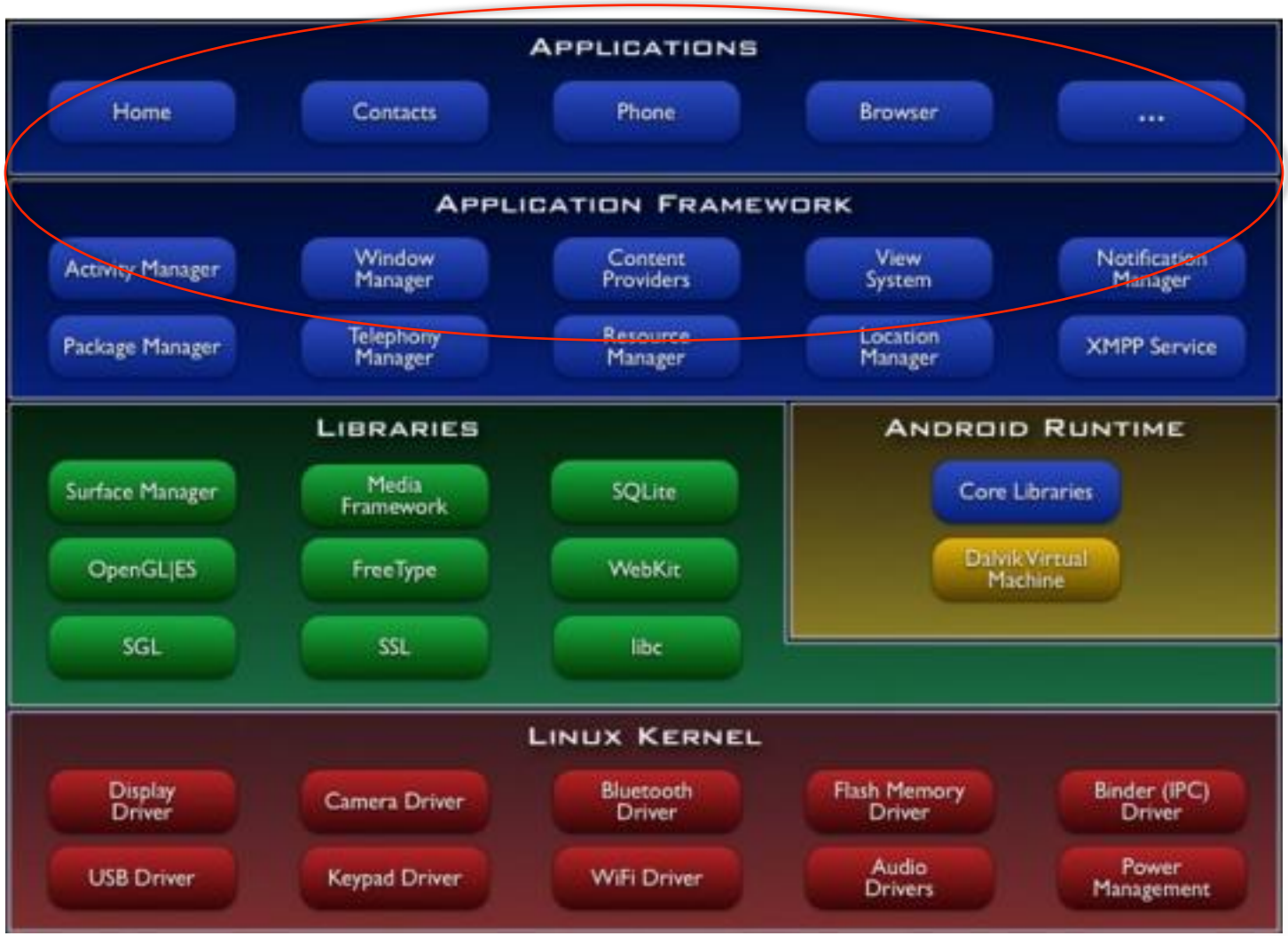
- Others
  - Stack Overflow
  - YouTuve
  - Google I/O talks
  - Android Developer Blog

Agenda

- Intro to Android
- Useful Tools
- **Application Components**
- Activities
- Views and ViewGroups
- Component Communication
- “Hello, world!”

Application Components

Application Components



Application Components

- Application
- Activity
- Service
- ContentProvider
- SyncAdapter
- BroadcastReceiver

Application Components

- How does the system know which components belong to which applications?

Application Components

- How does the system know which components belong to which applications?
  - AndroidManifest.xml

Application Components

- AndroidManifest.xml
  - Application's registry for its components
  - How to interperet Intents (more on this later)
  - Configure default events on launch, on broadcast, etc.

Application Components

- Activity now, others later

Agenda

- Intro to Android
- Useful Tools
- Application Components
- **Activities**
- Views and ViewGroups
- Component Communication
- “Hello, world!”

Activity

- android.app.Activity
- Developer page

Activity

- Part of app user can do and/or see
- One Activity on screen
- Holds View in a Window object

Activity

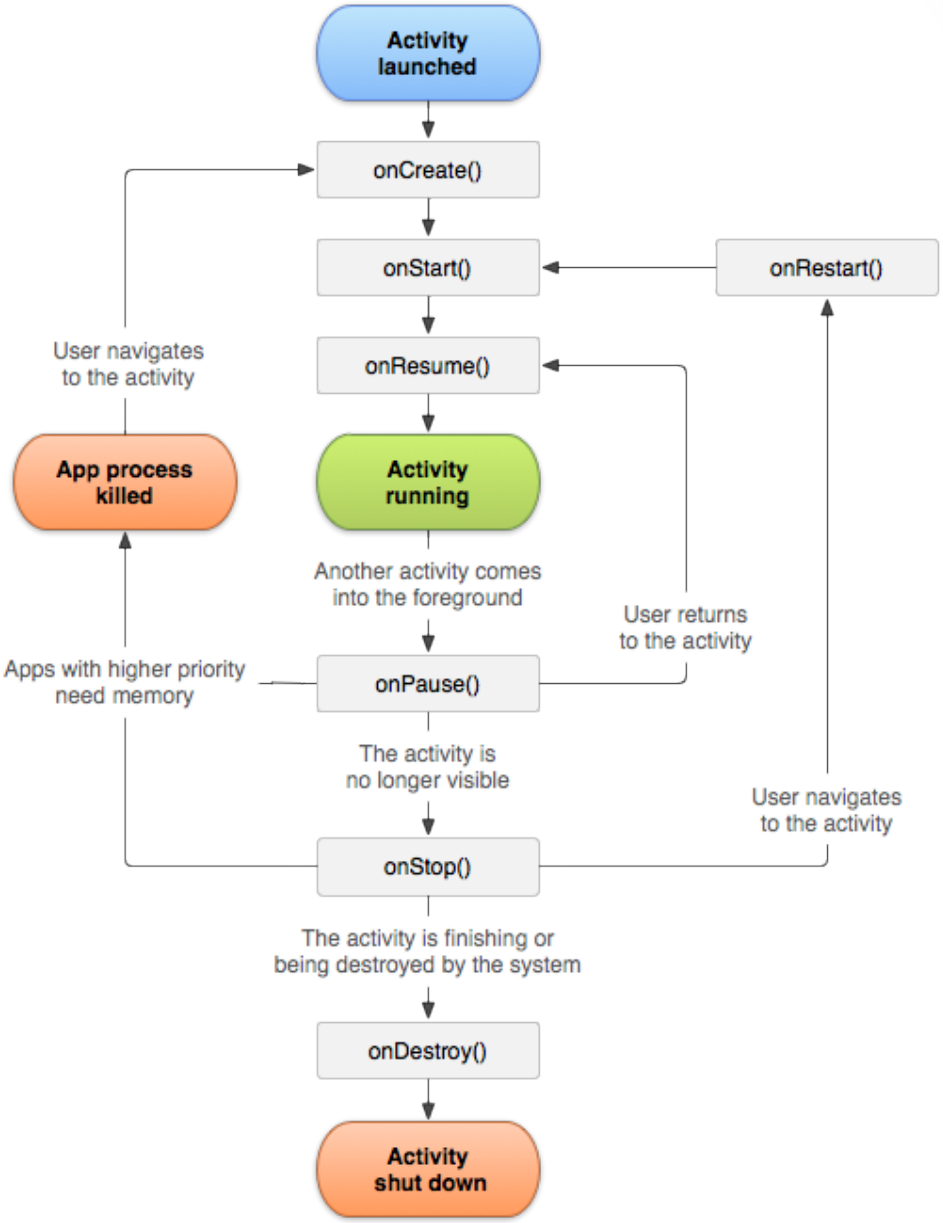
- Must have Activity to launch
- Must be registered in AndroidManifest.xml
- Can respond to Intents via IntentFilters (nore on this later)

Activity

- Quick note about Context
  - Fromt the developer page:
    - “Interface to global information about an application environment.”
  - Needed by many actions to coordinate between components at a low level
    - i.e. Making sure things run in the Context of the main thread (more on this later)
  - Activity extends Context (kinda)

Activity

- Life cycle:



Activity

- Subtle, but important, quirks about life cycle
  - Recreated on configuration change
  - Leave screen -> only state is saved, not entire instance
  - This can lead to memory leaks (Java-style)
- Why is this a big deal?

Activity

- Subtle, but important, quirks about life cycle
  - Recreated on configuration change
  - Leave screen -> only state is saved, not entire instance
  - This can lead to memory leaks (Java-style)
- Why is this a big deal?
  - Entire View hierarchy is often recreated!

Agenda

- Intro to Android
- Useful Tools
- Application Components
- Activities
- **Views and ViewGroups**
- Component Communication
- “Hello, world!”

View and ViewGroup

- android.view.View
- Developer page
- android.view.ViewGroup
- Developer page

View and ViewGroup

- A View is a rectangular space to draw on screen

View and ViewGroup

- A ViewGroup is a special View that can hold other Views

View and ViewGroup

- A ViewGroup is a special View that can hold other Views
  - Can see relationship in xml

View and ViewGroup

- Views must live inside a Context, i.e. an Activity
  - Only in Activity

View and ViewGroup

- Views must live inside a Context, i.e. an Activity
  - Only in Activity
- Cannot be modified from any thread besides main thread
  - More on this when we talk about multi threading

View and ViewGroup

- Can extend View and ViewGroup yourself

View and ViewGroup

- Can extend View and ViewGroup yourself
  - Often unnecessary

View and ViewGroup

- Can extend View and ViewGroup yourself
  - Often unnecessary
- Framework includes many useful classes
  - LinearLayout
  - RelativeLayout
  - FrameLayout
  - GridLayout
  - TextView
  - ImageView
  - ListView
  - GridView
  - Many more…

View and ViewGroup

- How do we create Views and ViewGroups?

View and ViewGroup

- How do we create Views and ViewGroups?
  - Normally, like any other class

View and ViewGroup

- How do we create Views and ViewGroups?
  - Normally, like any other class
  - Inflate from XML

View and ViewGroup

- How do we create Views and ViewGroups?
  - Normally, like any other class
  - **Inflate from XML**
    - Best for beginners

View and ViewGroup

- Basic layout structure

```
<ViewGroup>
    <View />

    <!-- Notice the nested ViewGroup -->
    <ViewGroup>
        <View />

        <!-- Can have multiple Views in a ViewGroup -->
        <View />
    </ViewGroup>
</ViewGroup>
```

View and ViewGroup

- Basic layout structure
  - We'll see a specific sample in the demo

```
<ViewGroup>
    <View />

    <!-- Notice the nested ViewGroup -->
    <ViewGroup>
        <View />

        <!-- Can have multiple Views in a ViewGroup -->
        <View />
    </ViewGroup>
</ViewGroup>
```

View and ViewGroup

- What if we have this?

```
<LinearyLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <Button
                android:id="@+id/my_btn"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Press" />
</LinearLayout>
```

View and ViewGroup

- What if we have this?
- How can the Activity tell if the Button is clicked?

```
<LinearyLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/my_btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Press" />
</LinearLayout>
```

View and ViewGroup

- What if we have this?
- How can the Activity tell if the Button is clicked?
  - **Listener design pattern**

```
<LinearyLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="match_parent"
      android:layout_height="match_parent">

      <Button
            android:id="@+id/my_btn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Press" />
</LinearLayout>
```

Agenda

- Intro to Android
- Useful Tools
- Application Components
- Activities
- Views and ViewGroups
- **Component Communication**
- “Hello, world!”

Component Communication

- Intercomponent communication
- Intracomponent communication

Component Communication

- Intercomponent communication
- **Intracomponent communication**

Component Communication

- Activty displays View with a Button

Component Communication

- Activty displays View with a Button
- Want to respond to Button clicks

Component Communication

- Activty displays View with a Button
- Want to respond to Button clicks
- Activity is said to be “listening” for a click

Component Communication

- Listener design pattern

Component Communication

- Listener design pattern (a.k.a. callbacks)

Component Communication

- Listener design pattern
  - Define Listener interface

Component Communication

- Listener design pattern
  - Define Listener interface
  - Object that is listening implements the interface

Component Communication

- Listener design pattern
  - Define Listener interface
  - Object that is listening implements the interface
  - Target objects notifies listeners when an event occrs

Component Communication

- Listener design pattern
  - Define Listener interface
  - Object that is listening implements the interface
  - Target objects notifies listeners when an event occrs
  - Listener responsds to event

Component Communication

- Not limited to Buttons or the like
  - Any class can define a listener interface
  - Must notify on certain events

Component Communication

- Example
  - OnClickListener

```
class MyActivity extends Activity implements OnClickListener {
        // … other code

        @Override
        public void onClick(View v) {
                startOtherActivity();
        }
}
```

Component Communication

- **Intercomponent communication**
- Intracomponent communication

Component Communication

- How do different compponents talk to each other?

Component Communication

- How do different compponents talk to each other?
  - They don’t

Component Communication

- How do different compponents talk to each other?
  - They don't
  - One Activity on the screen at a time

Component Communication

- How do different compponents talk to each other?
  - They don't
  - One Activity on the screen at a time
- How to start another Activity?
  - **Intents**

Component Communication

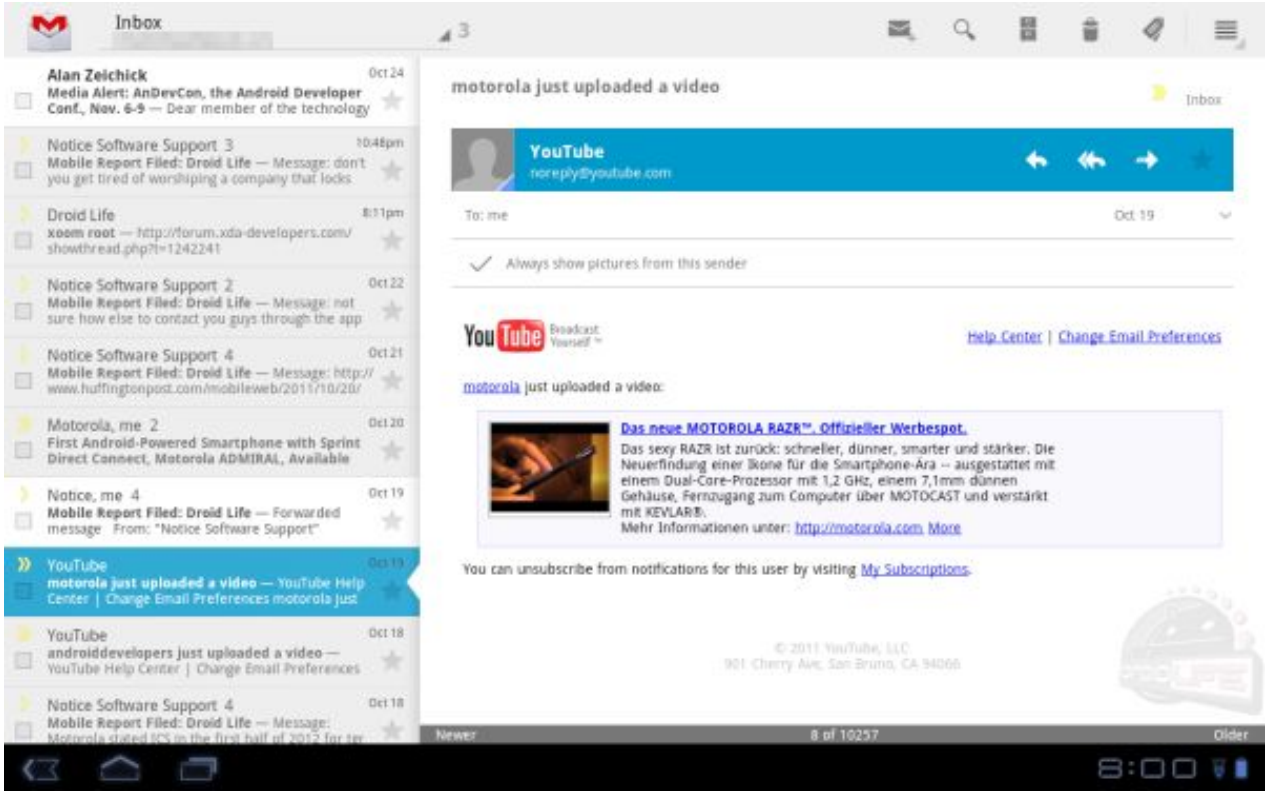
- Intents
  - App tells the system what you want to do
  - System can maintain important resources, i.e. Window
    - App tells system to swap Activities in and out of the Window

Component Communication

- Intents are MUCH more powerful than Activity switching
- Intents will pop up all the time in the class
  - This is all we need to know for now

Question?

- If only one Activity is allowed on the screen at one time, how can we put more than one user action on the screen?
- For example:



Naive Solution

- Just have a list and frame side-by-side in a layout

Naive Solution

- Just have a list and frame side-by-side in a layout
  - What if the screen is small?

Proper Solution

- Fragments!

Proper Solution

- Fragments!
  - Modular pieces of UI with life cycle like an Activity
  - Doesn’t manage a Window or other system resources

Proper Solution

- Fragments!
  - Modular pieces of UI with life cycle like an Activity
  - Doesn’t manage a Window or other system resources
  - Next time

Agenda

- Intro to Android
- Useful Tools
- Application Components
- Activities
- Views and ViewGroups
- Component Communication
- **“Hello, world!”**

“Hello, world!”

- We have all the pieces we need

“Hello, world!”

- We have all the pieces we need
- Code will be in the git repo
  - Please familiar yourself with:
  - Github (at least)
  - Command line git

Thank You

Q&A until 9pm