

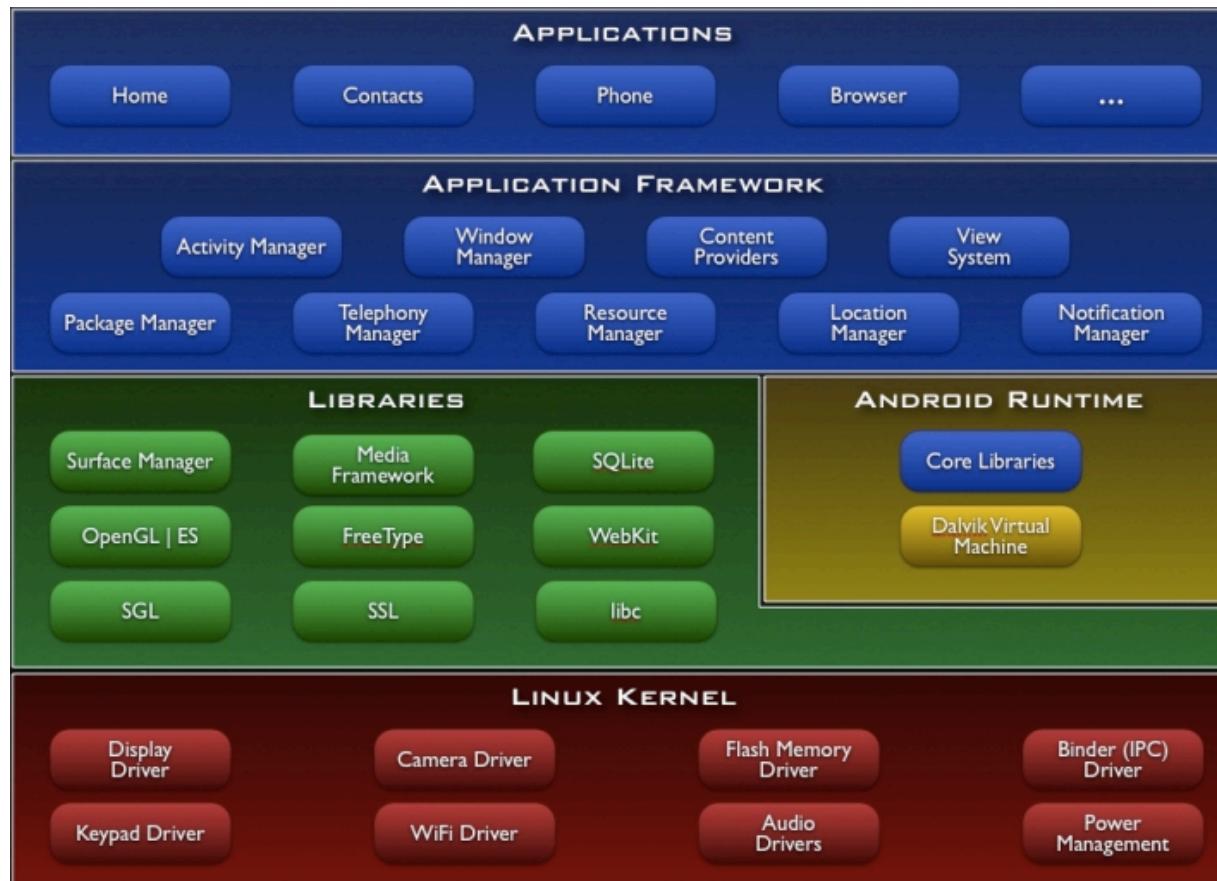
# Android Programming Tutorial

Rong Zheng

# Outline

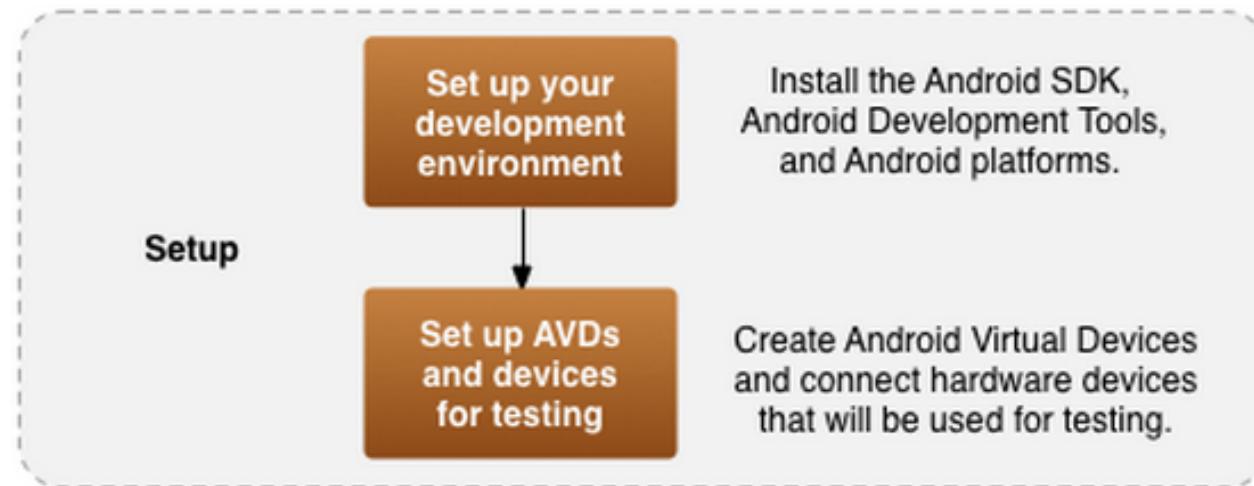
- ❑ What is Android OS
- ❑ Setup your development environment
- ❑ GUI
  - ❑ Layouts
  - ❑ Activity life cycle
  - ❑ Event handling
- ❑ Tasks
- ❑ Intent and broadcast receiver
- ❑ Resource

# What is Android



- ❑ Linux-based OS
- ❑ Dalvik virtual machine running dex-code translated from Java

# Setup



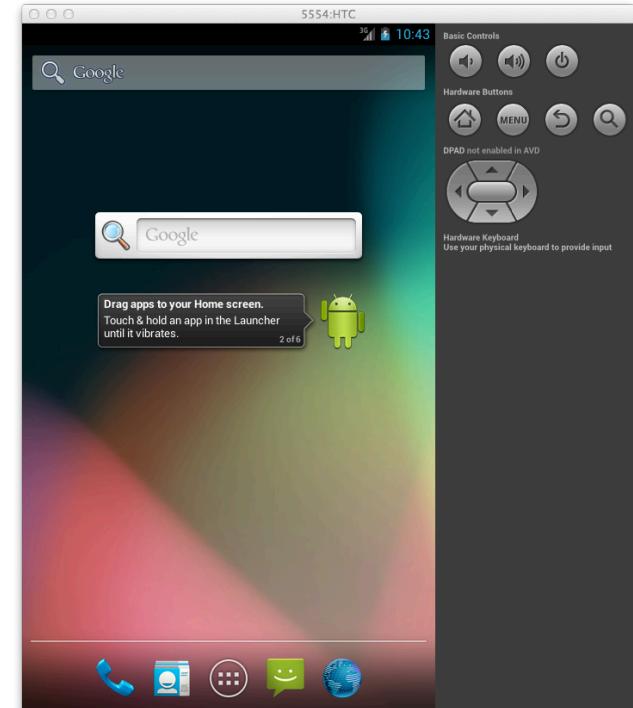
# What you need

- ❑ JDK 6
- ❑ IDE
  - ❑ ADT (Android Development Tool) bundle ✓  
or
  - ❑ Eclipse + ADT plug-in + Android SDK  
or
  - ❑ Android studio
- ❑ Download earlier SDK versions using SDK manager if needed

# Android Virtual Device (AVD)

- ❑ Android emulator allows development w/o physical device
  - ❑ Tend to be slow when launched the first time
  - ❑ Can emulate network, camera and storage
  - ❑ No support for WiFi, Bluetooth, most hardware sensors
- ❑ Possible alternatives:
  - ❑ Genymotion (previously AndroVM), Manymo, ...

Demo: create a new AVD



# Android App Essentials

## ❑ Layout

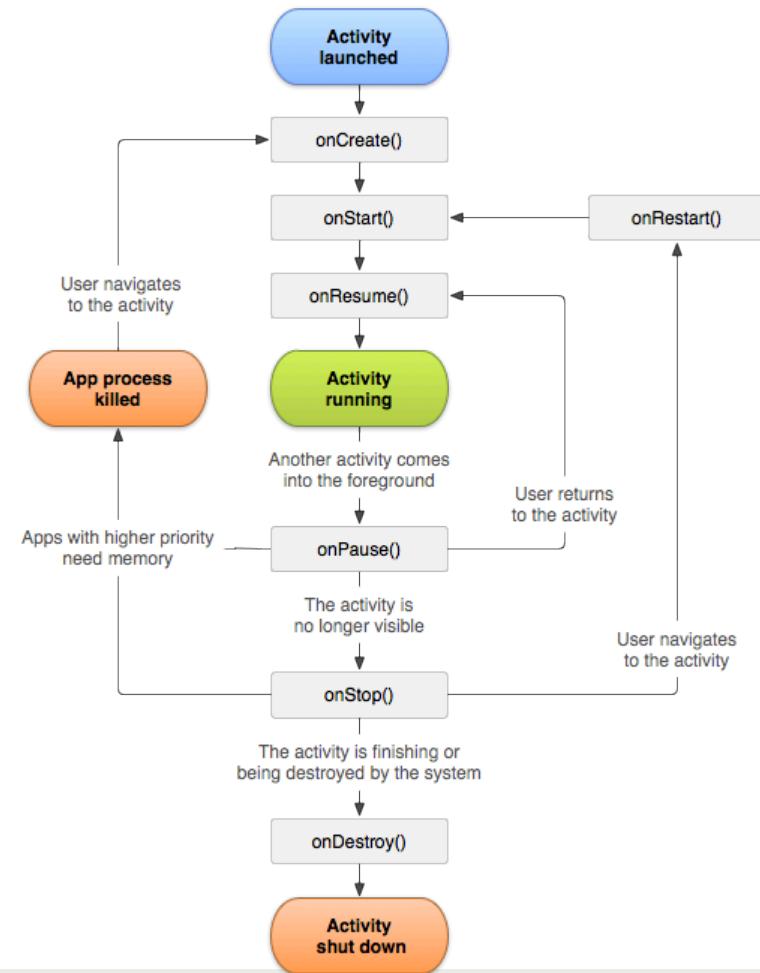
- ❑ View objects: UI widgets such as buttons, text box etc.
- ❑ Positioning view objects on the screen
- ❑ View group: invisible view containers that defines how the child views are laid out

## ❑ Activity

- ❑ Implements the real logic behind the application
- ❑ There could be multiple activities

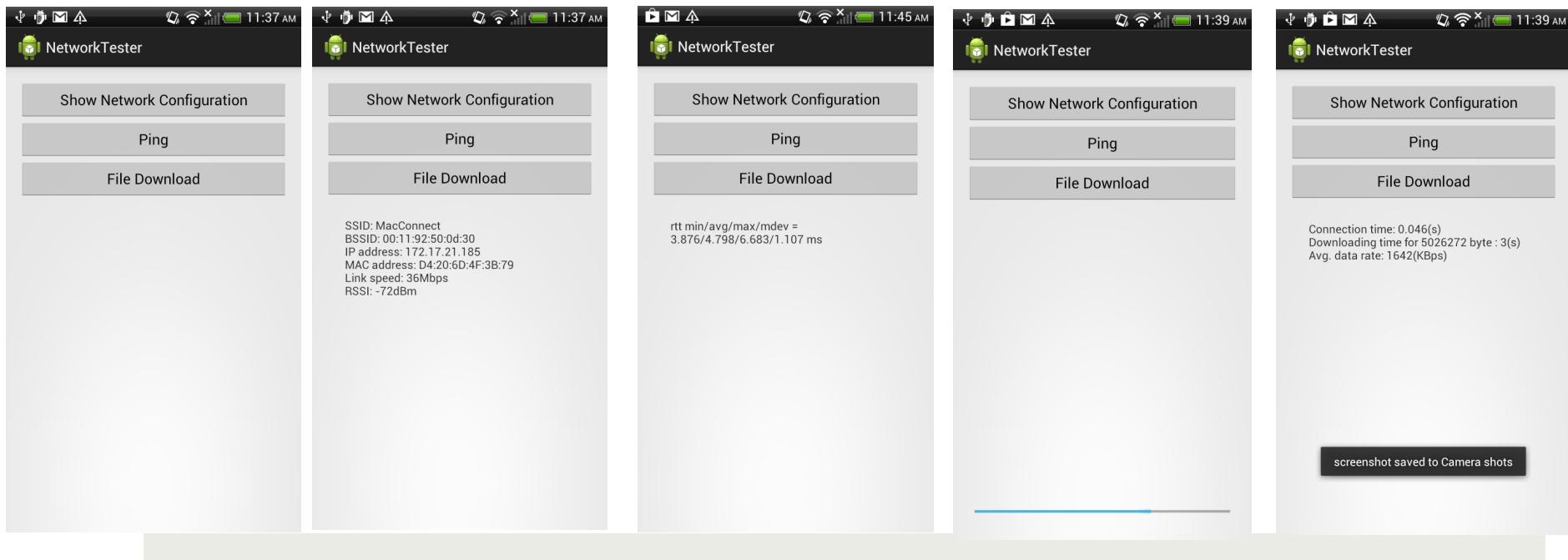
# Activity life cycle

- ❑ Screen rotation
  - ❑ Application paused, stopped and restarted
  - ❑ `onCreate()` called
- ❑ Press home button
  - ❑ Activity paused, `onPause()`, `onStop()` called;
  - ❑ `onRestart()` called when restarted
- ❑ Press back button
  - ❑ Activity destroyed, `onPause()`, `onStop`, `onDestroy()`



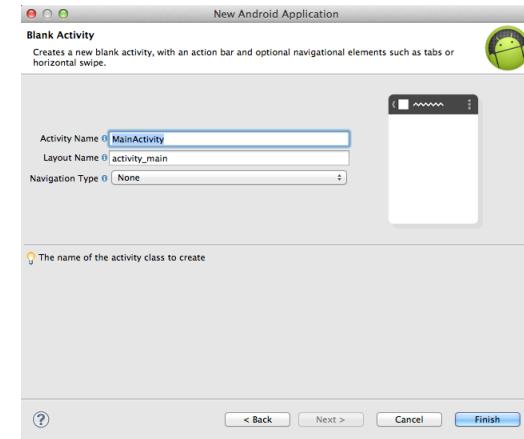
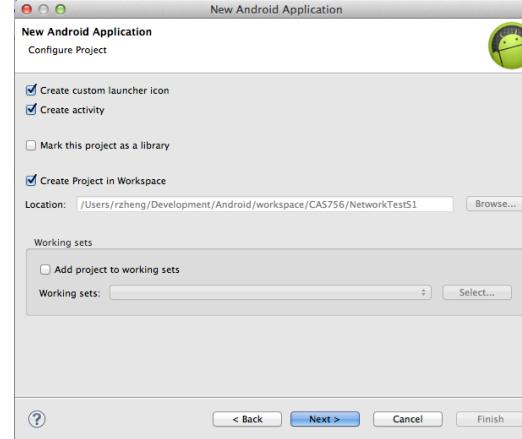
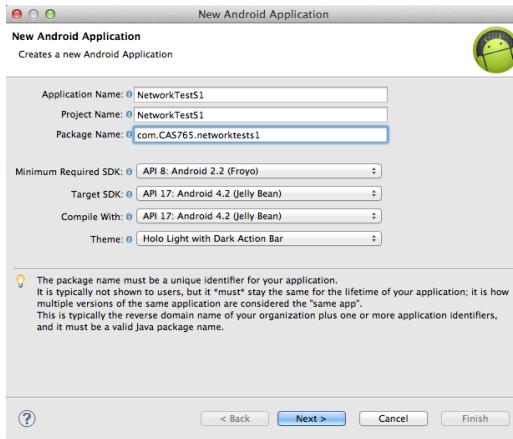
# Example: Network tester

- ❑ Display the network configuration
- ❑ Ping google.com
- ❑ Download a file from a fixed URL and determine the downloading speed



# Step 1: Create an Android Project

## ☐ New → New Android Application



# Step 2: Lay out UI

```
<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">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >

        <Button
            android:id="@+id/netinfo_button"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Show Network Configuration" />

        <Button
            android:id="@+id/ping_button"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Ping" />

        <Button
            android:id="@+id/fd_button"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="File Download" />

        <TextView
            android:id="@+id/text_display"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:maxLines="100"
            android:padding="24dp"
            android:scrollbars="vertical" />
    </LinearLayout>

    <ProgressBar
        android:id="@+id/test_progressBar"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_margin="10dp"
        android:visibility="invisible"/>

```



## Step 3: From Layout to View Objects

- ❑ `setContentView(R.layout.activity_main)` inflates a layout and put on screen
  - ❑ `R.layout.activity_main` is a resource id
  - ❑ `@+id/resource_id` automatically creates the resource id
- ❑ Wire up widgets
  - ❑ Get the references
    - ❑ `public View findViewById( int id)`
  - ❑ Set listeners

# AsyncTask

- ❑ Network operations typically take time to complete
- ❑ AsyncTask allows operations running in the background and publishing the results on the UI thread
  - ❑ Do not block the UI thread!
  - ❑ Do not access the Android UI toolkits from outside the UI thread
- ❑ Implement a class that extends AsyncTask<Param, Progress, Result>
  - ❑ Params, the type of the parameters sent to the task upon execution.
  - ❑ Progress, the type of the progress units published during the background computation.
  - ❑ Result, the type of the result of the background computation

# AsyncTask (cont)

- ❑ `doInBackground(Void... params)`
  - ❑ What to do in the background
- ❑ `protected void onProgressUpdate(Void... progress)`
  - ❑ Run in the UI thread
  - ❑ Invoked by `publishProgress` in the `doInBackground`
- ❑ `protected void onPostExecute(Void result)`
  - ❑ Run in the UI thread
- ❑ `protected void onPreExecute()`
  - ❑ Set up the task

Example code

# Little tricks

- ❑ Can execute shell commands from Android app
  - ❑ `Runtime.getRuntime().exec(Cmd)`
- ❑ Disable screen rotation

```
<activity android:name="com.CAS765.android.networktester.MainActivity"  
        android:label="@string/app_name"  
        android:screenOrientation="portrait">
```

# Little tricks (cont'd)

- ❑ User handler to cancel an AsyncTask after some time
  - ❑ public final boolean postDelayed (Runnable r, long delayMillis)

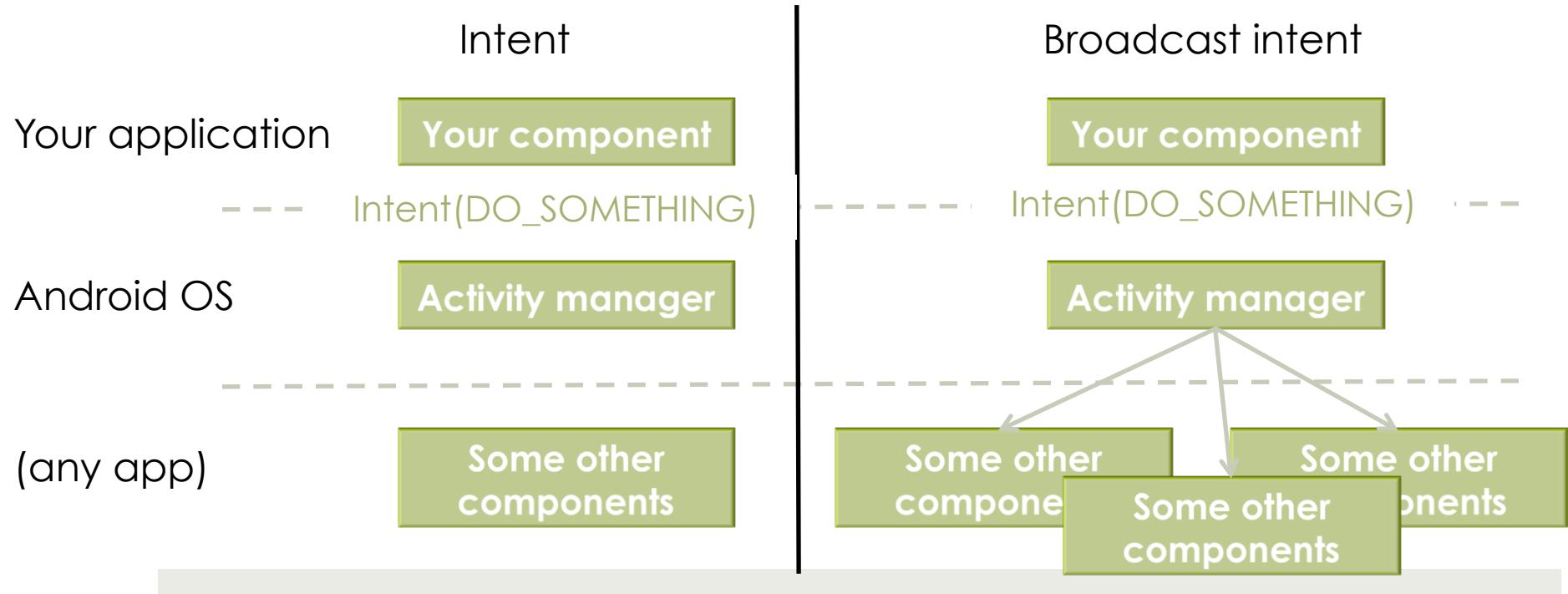
```
Handler handler = new Handler();
handler.postDelayed(new Runnable() {
    @Override public void run() {
        if (downloader.getStatus() == AsyncTask.Status.RUNNING)
            downloader.cancel(true);}}, 60000);
```

- ❑ Use toast for notification

```
Toast.makeText(MainActivity.this, "Opps...", Toast.LENGTH_SHORT)
    .show();
```

# Intent and broadcast receiver

- Intent is a mechanism in Android to allow late binding of components within and across applications
  - Message passing



# Broadcast receiver

- ❑ Allows to register for system wide or application events
  - ❑ ACTION\_BATTERY\_LOW, ACTION\_HEADSET\_PLUG, ACTION\_SCREEN\_ON, ACTION\_TIMEZONE\_CHANGED, WifiManager.SCAN\_RESULTS\_AVAILABLE\_ACTION, etc
- 1. Step1: extend a broadcast receiver class and override onReceive(Context c, Intent intent) method
- 2. Step 2: register the receiver(BroadcastReceiver receiver, IntentFilter filter)
- 3. Step 3: change AndroidManifest.xml to add the permission

# Example

```
if (mReceiverWifi == null)
    mReceiverWifi = new WifiReceiver();

registerReceiver(mReceiverWifi, new IntentFilter(
    WifiManager.NETWORK_STATE_CHANGED_ACTION));
```



description of Intent values  
to be matched

```
private class WifiReceiver extends BroadcastReceiver {
    public void onReceive(Context c, Intent intent) {
        NetworkInfo info = (NetworkInfo)intent.getExtras().getParcelable("networkInfo");
        mText.setText(info.getDetailedState().toString());
    }
}
```

# Resource

- ADT bundle  
<http://developer.android.com/sdk/index.html>
- ADT Eclipse Plug-in  
<http://developer.android.com/tools/sdk/eclipse-adt.html>
- Android API  
<http://developer.android.com/reference/packages.html>