Android 101

HackWeek Code Class - Q1 2013

Jan Chong
Twitter for Android, Manager
@lessachu

Mohammad Almalkawi
Twitter for Android, Engineer
@moh

Pre-Requisites

- Does everyone have intelliJ or Eclipse installed?
  - intelliJ available here: http://download.jetbrains.com/idea/ideaIC-12.0.1.dmg
- Does everyone have the Android SDK installed?
  - http://dl.google.com/android/android-sdk_r21.0.1-macosx.zip
  - Install, type "android" at the command line and select the following:
    - Android SDK tools
    - Android SDK Platform-tools
    - Android 4.2 (API 17) Documentation, SDK Platform, Google APIs, Sources
    - **ARM EABI v7a System Image**
- Campfire Room: **androidclass**

Syllabus

Day 1	Day 2
INTRO TO ANDROID • The State of Android • Intro to Dev Tools and Debugging • Hello World	DYNAMIC UI • Using Fragments • Class Exercise
LUNCH	LUNCH
APPLICATION BASICS • Activities • Intents • Activity Lifecycle • AdapterViews and Adapters	DATA PERSISTENCE • SQLite Tools • Database APIs and Cursors • Content Provider
BREAK	BREAK
LAB • Create TODO App	LAB • *Class Choice:* Advanced Topic

Android History

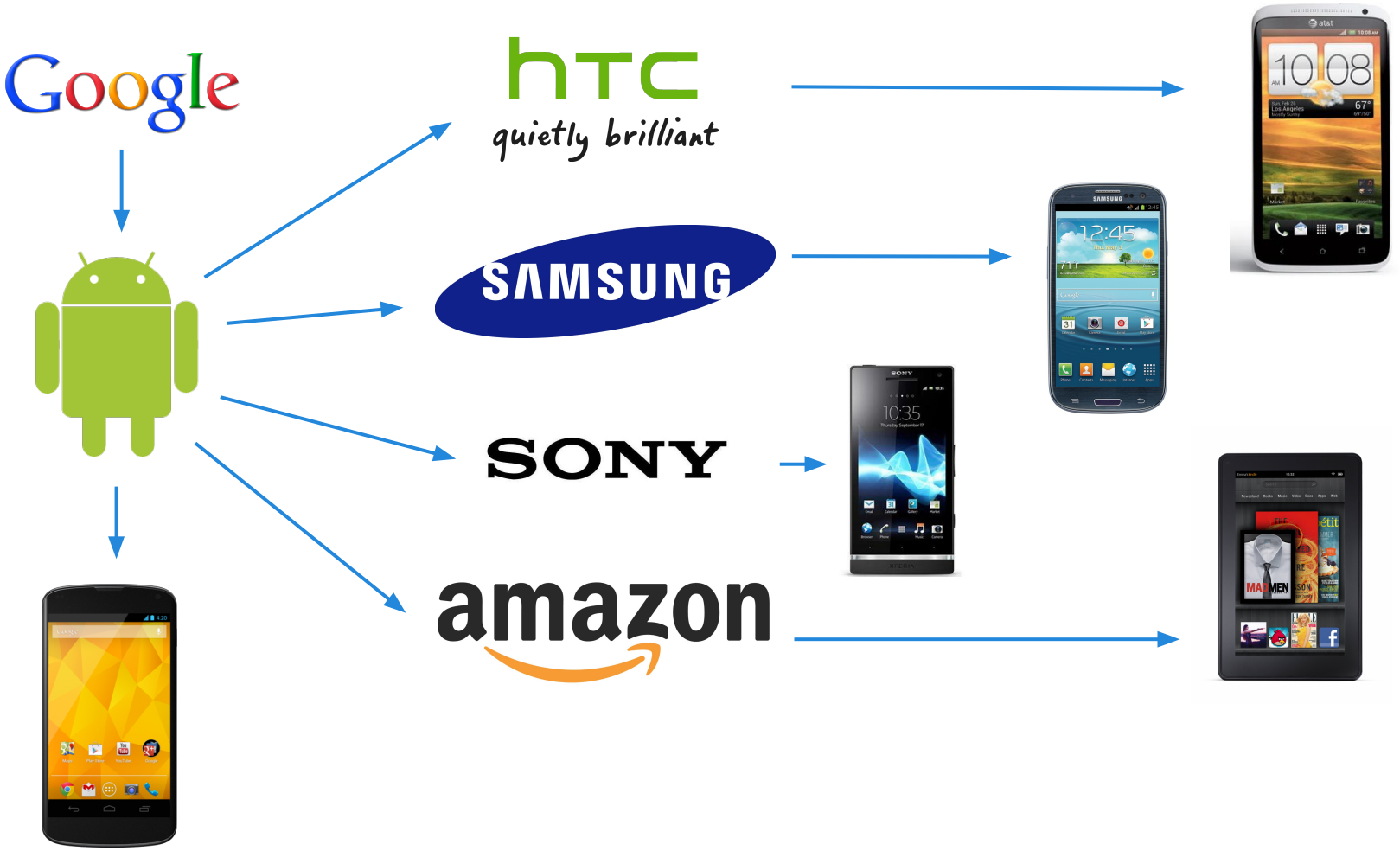
What is Android?

- Linux-based open source mobile operating system
  - Released under the Apache License
  - Source available at source.android.com

Early History

- Originally developed by a startup called Android, Inc in 2003
- Acquired by Google in 2005
- First commercially available phone released on Oct 22, 2008 (T-Mobile G1)

Android Ecosystem

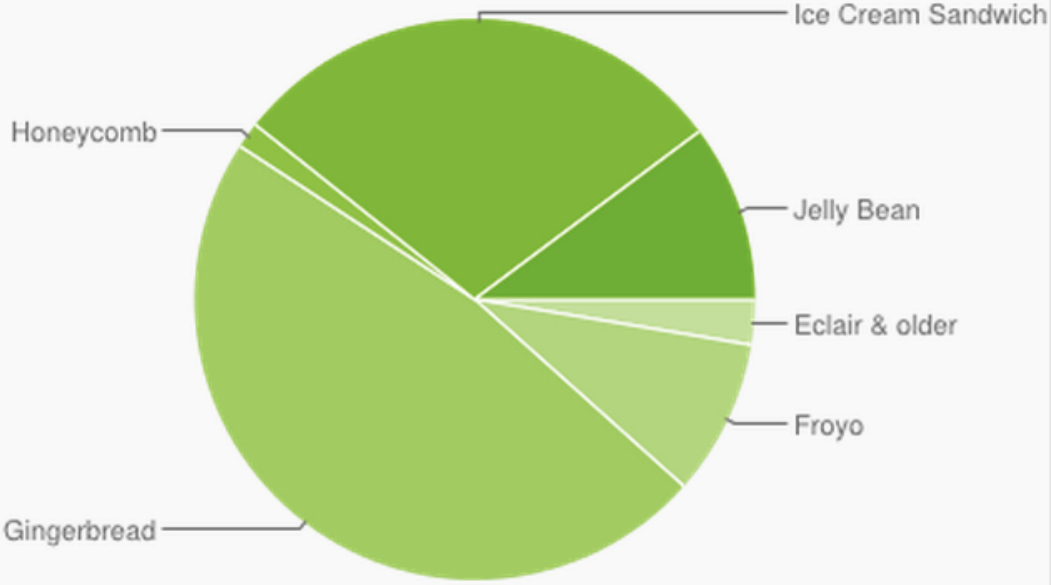


Android API History

Android 1	API 1	Oct 2008	
Android 1.1	API 2	Feb 2009	Released for the G1 only, bug fixes
Android 1.5 (Cupcake)	API 3	Apr 2009	UI tweaks, on screen keyboard, widgets
Android 1.6 (Donut)	API 4	Sept 2009	multiple screen resolutions, market place
Android 2.0-2.1 (Eclair)	API 7	Oct 2009	multiple account support, live wallpapers
Android 2.2 (FroYo)	API 8	May 2010	enterprise support, redesigned home screen
Android 2.3/2.3.3 (Gingerbread)	API 9/10	Dec 2010	reskin, lower level access to hardware for gaming, front facing camera
Android 3.0-3.2 (Honeycomb)	API 11/12/13	Feb 2011	Tablet-only
Android 4.0 (Ice Cream Sandwich)	API 14	Oct 2011	UI redesign, menu button removed
Android 4.2 (Jelly Bean)	API 17	June 2012	touch and visual performance improvements, google now

Android API Fragmentation

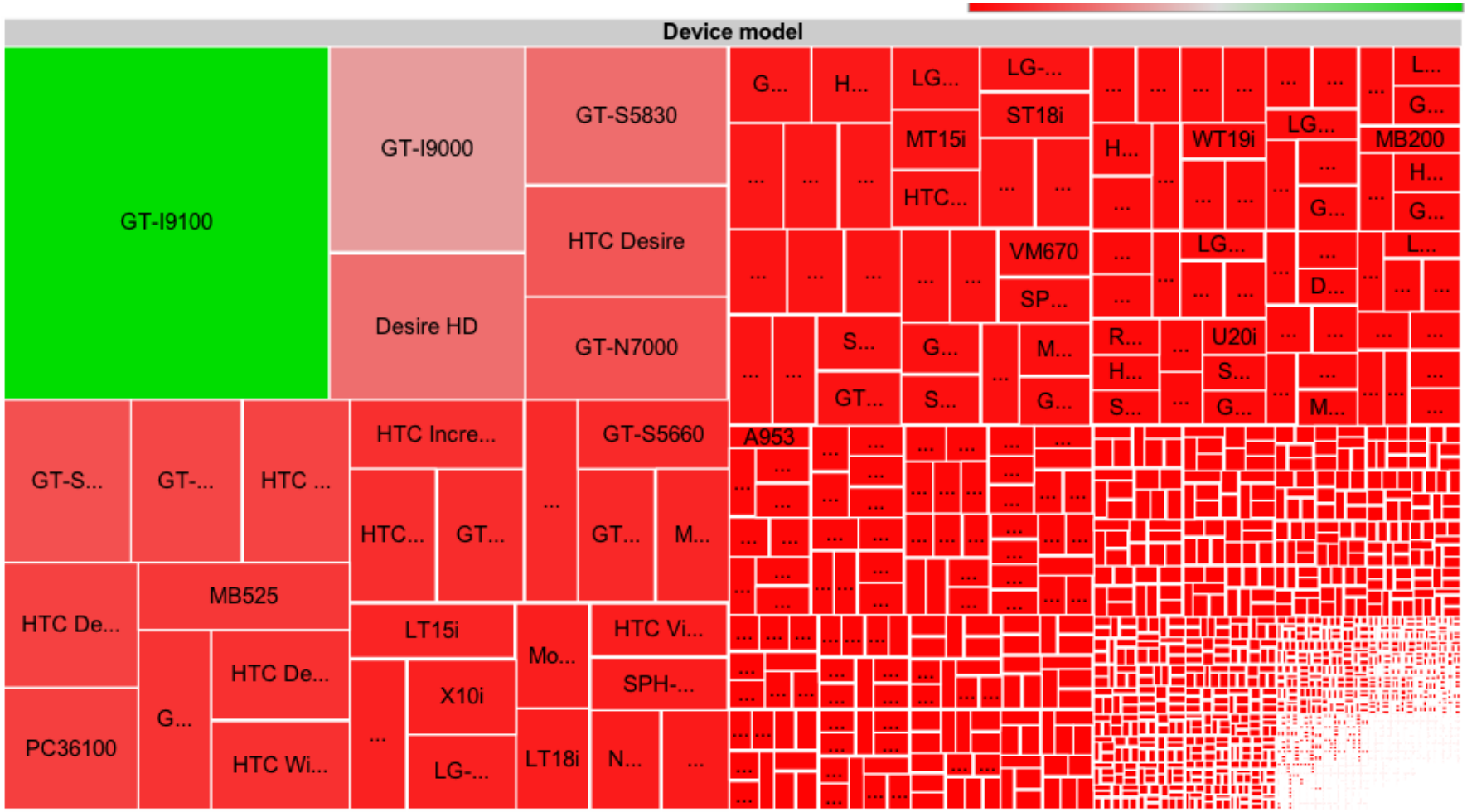
Version	Codename	API	Distribution
1.6	Donut	4	0.2%
2.1	Eclair	7	2.4%
2.2	Froyo	8	9.0%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7		10	47.4%
3.1	Honeycomb	12	0.4%
3.2		13	1.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	29.1%
4.1	Jelly Bean	16	9.0%
4.2		17	1.2%



from http://developer.android.com/about/dashboards/index.html

Data collected during a 14-day period ending on January 3, 2013

Android Device Fragmentation

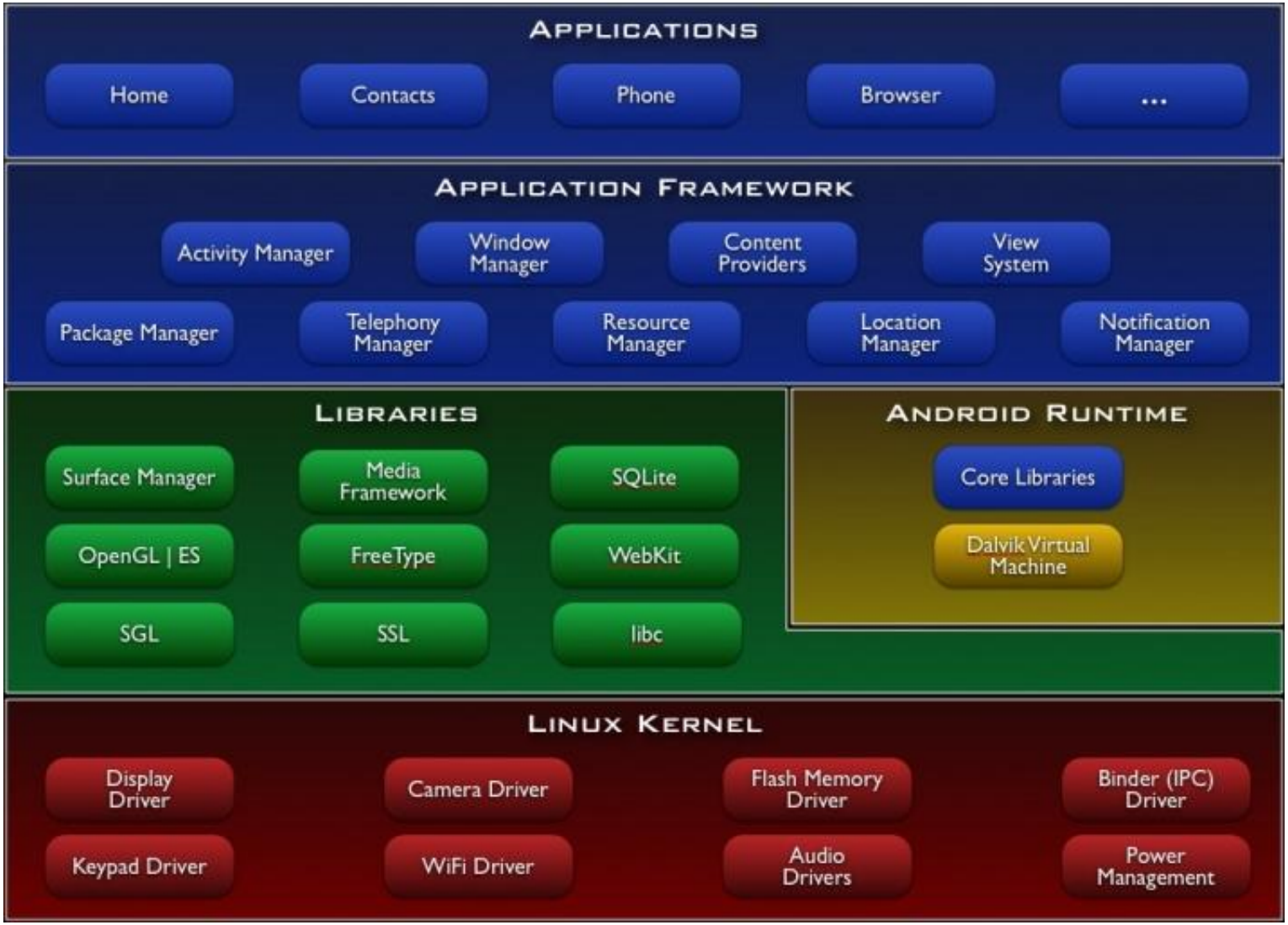


http://opensignal.com/reports/fragmentation.php

App Distribution

Google Play	Amazon AppStore
$25 fee, one time	$99 yearly fee (currently waived)
Google takes 30% of all sales	Amazon takes 30% of all sales
No certification, apps released immediately (1-4 hr distribution time)	Certification takes about a week
developer.android.com/distribute/index.html	developer.amazon.com/welcome.html

Android Architecture



Hello Android!

Building your first Android Project

Setting up your phone for dev

Enable USB debugging

Android 3.2 and below	Settings > Application > Development
Android 4.0	Settings > Developer options
Android 4.2	It may be hidden. If so, first go to: Settings > About phone and tap "Build number" seven times

You may also want:

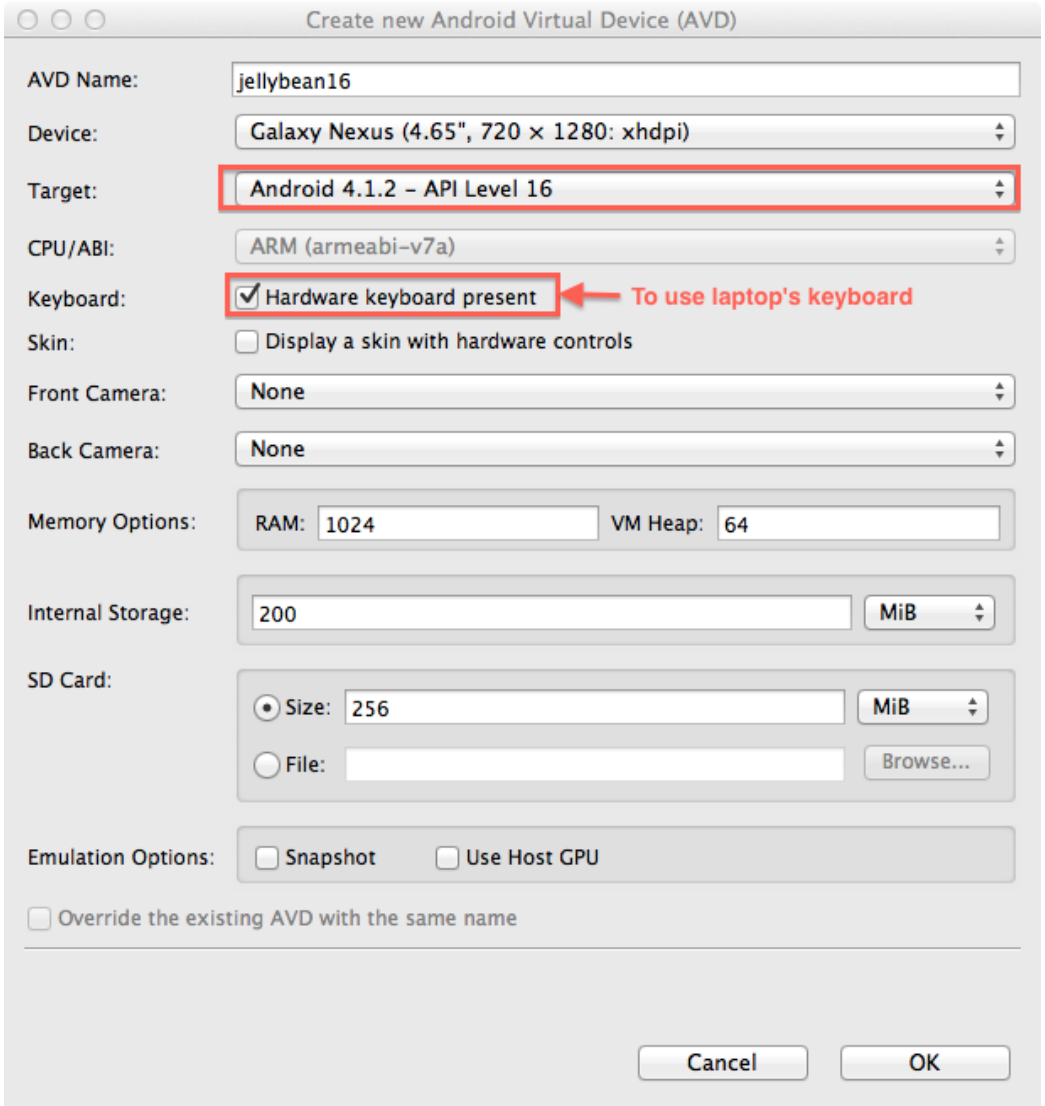
Allow installation from Unknown Sources

Setting up the emulator

```
> android avd
```

Click "New"

```
> emulator -avd
      [avd name]
```



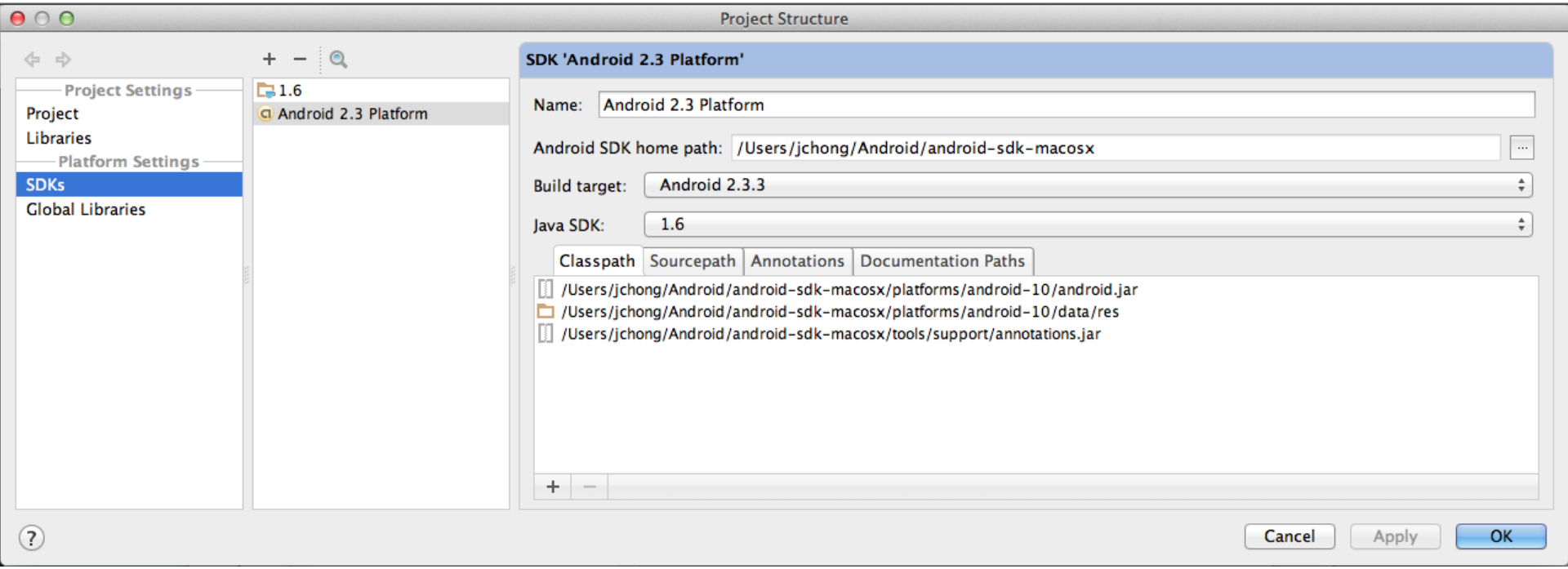
Adding the JDK and SDK paths

intellij: Configure > Project Defaults > Project Structure

JDK 1.6 path is (OS 10.7):

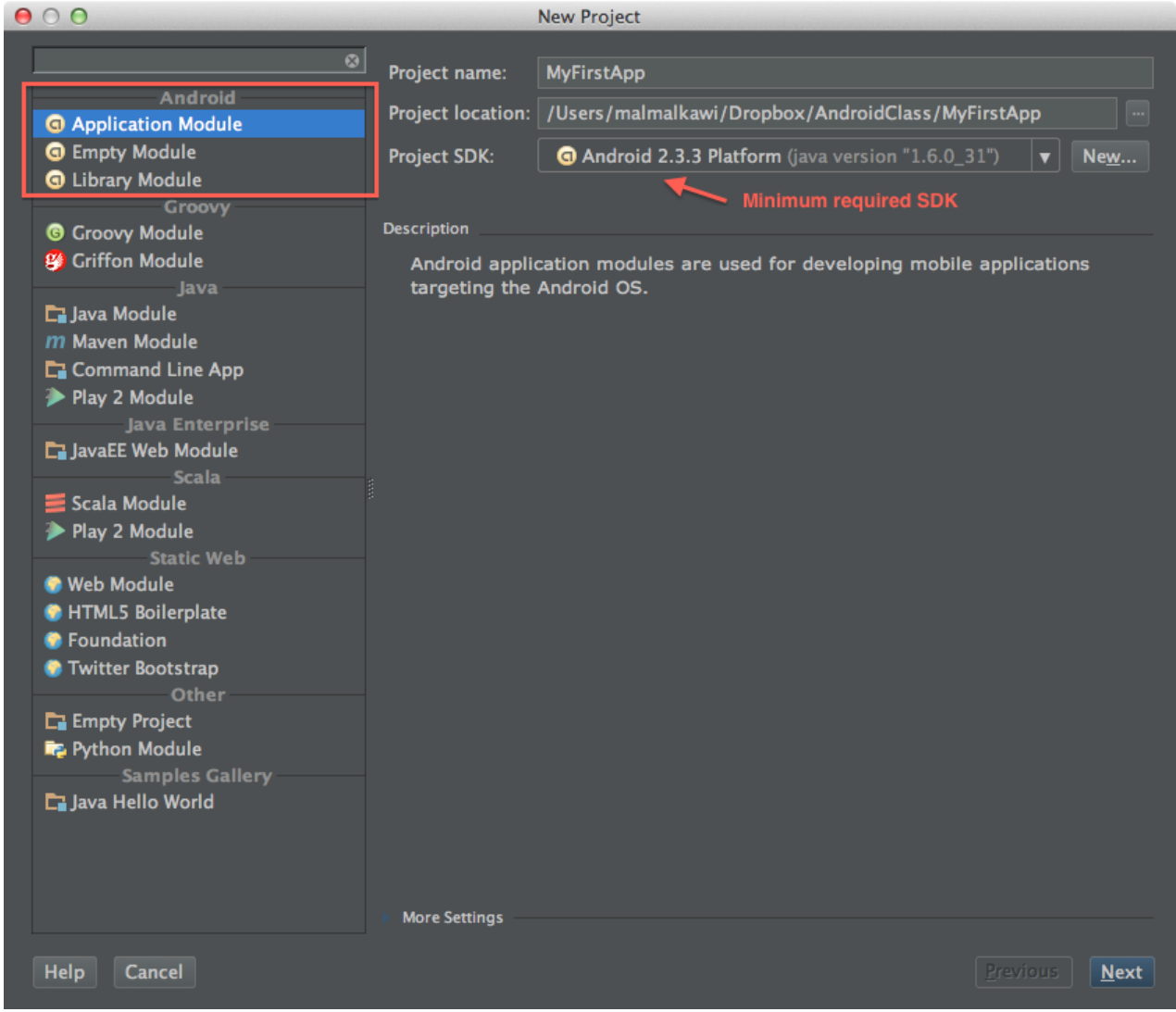
```
/System/Library/Frameworks/JavaVM.framework/Versions/1.6/Home
```

SDK: add the lowest API you want to support

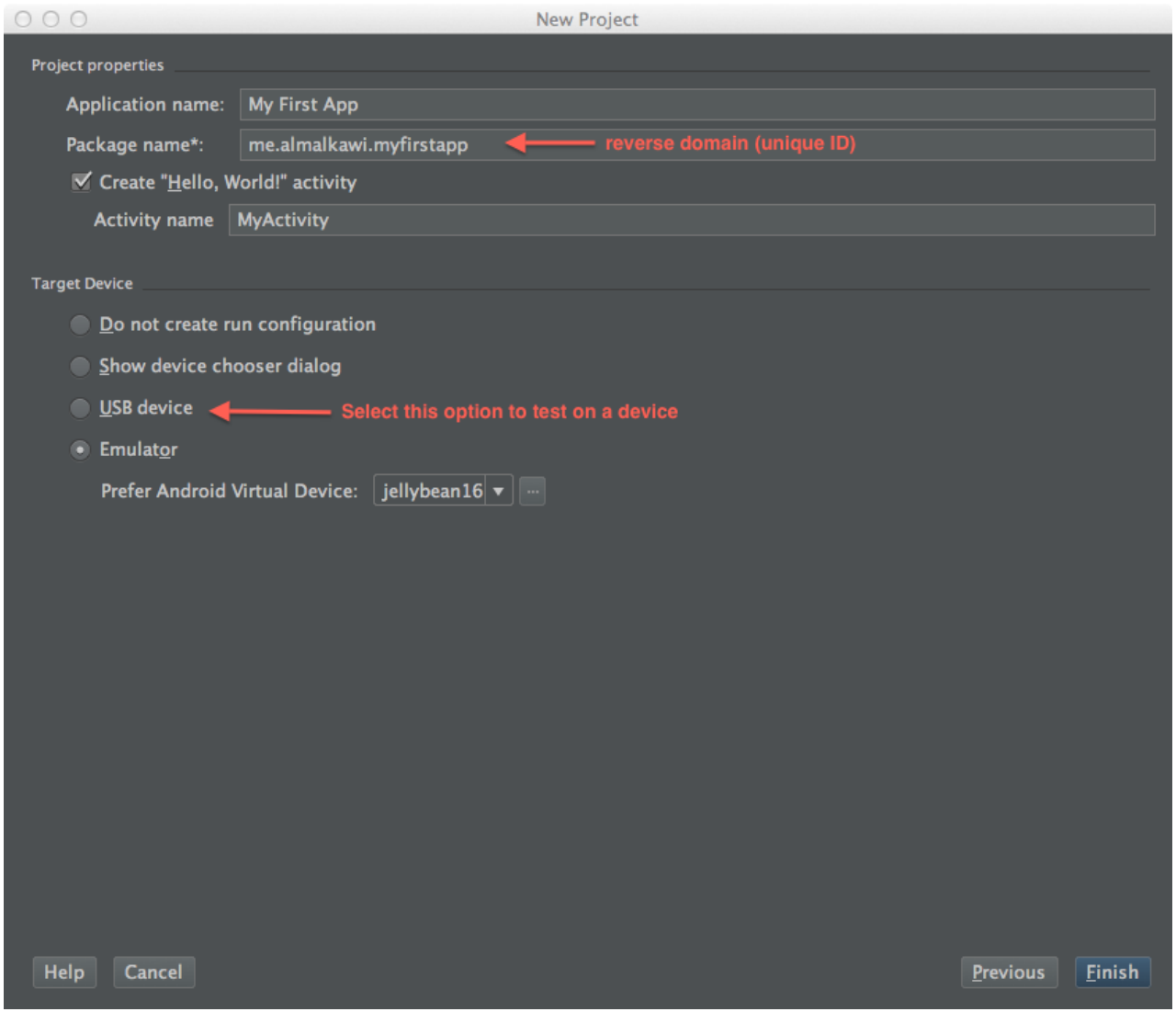


Create hello world project

File>New Project



New project properties



Default project structure

src/

- Implementation code (Java)

res/

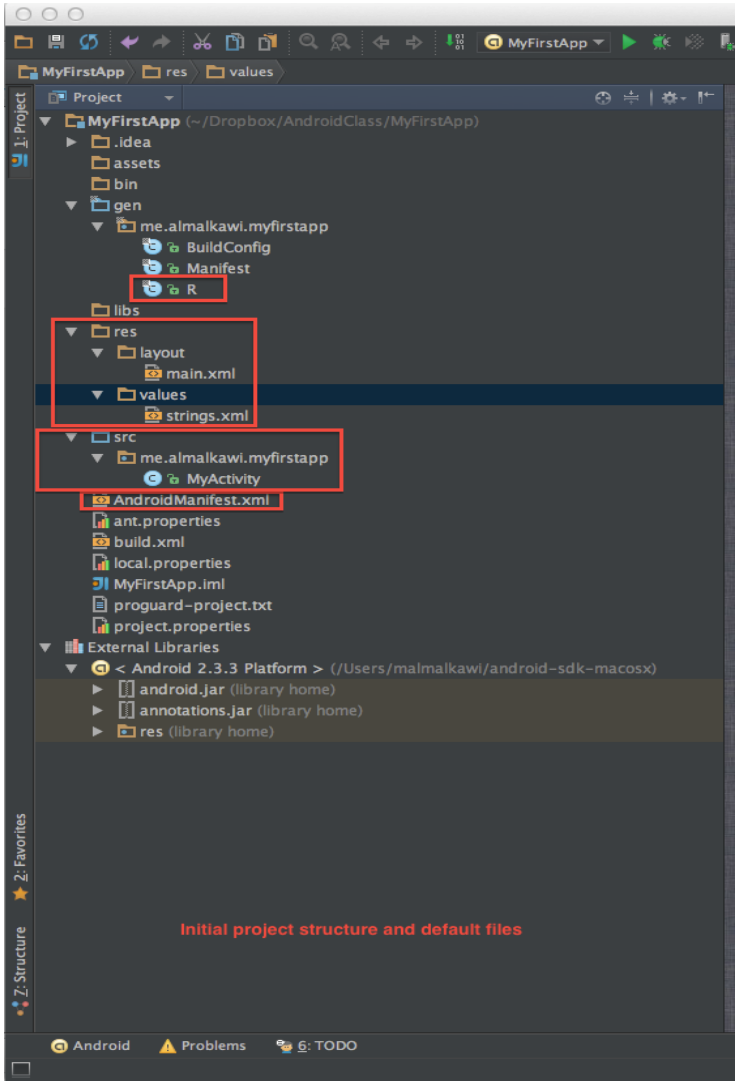
- Images
- XML layouts
- Strings
- Styles
- Constants

AndroidManifest.xml

- Information about the app

R.java file

- Autogenerated by build
- Contains resource IDs of res/ contents

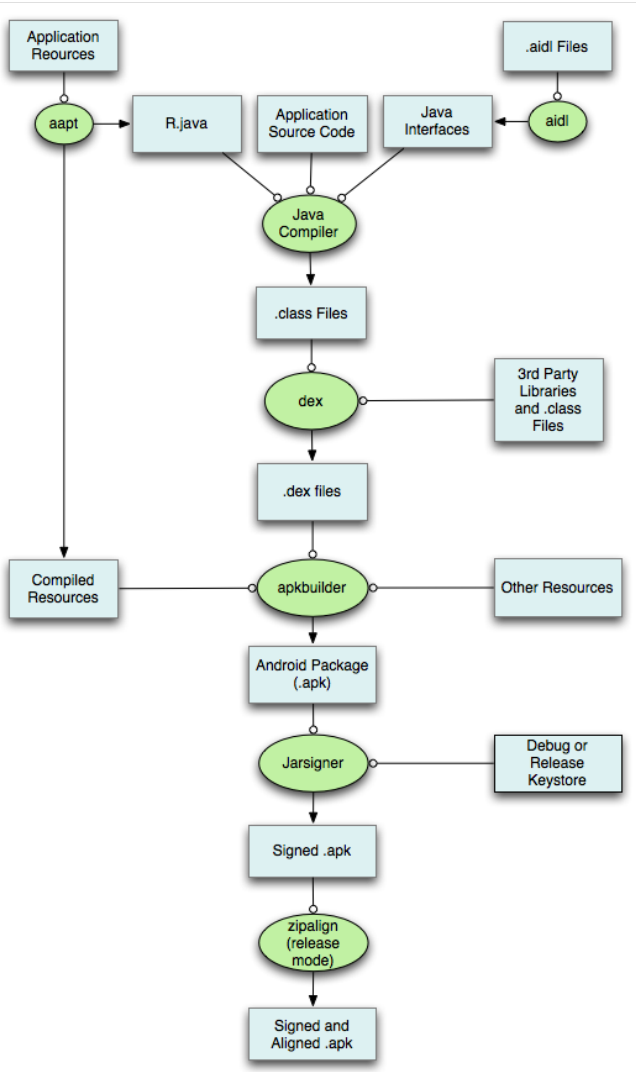


Building and Running Android app

IntelliJ: Shift + F10

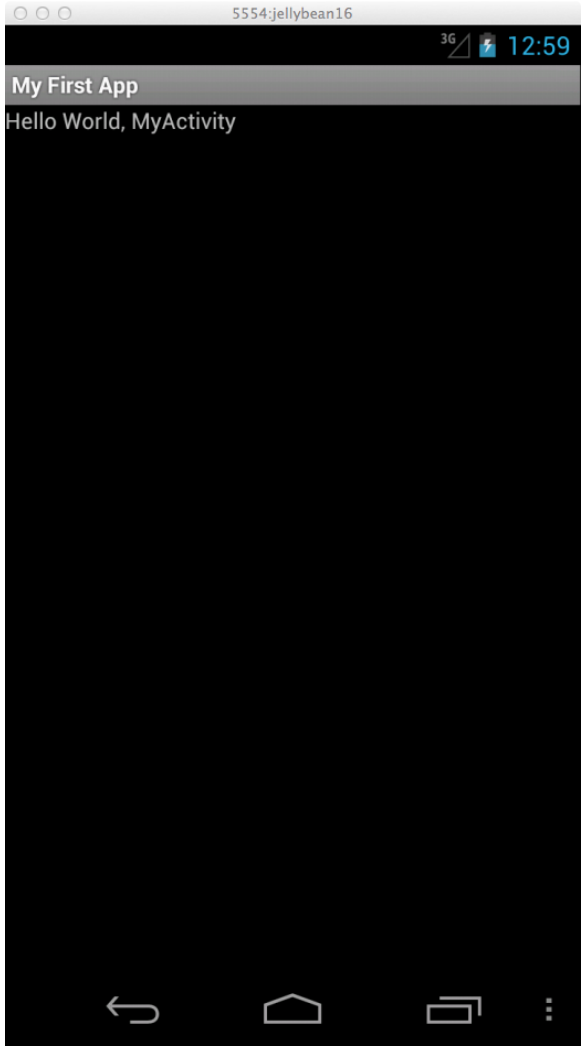
Eclipse: Shift + Command + F11

```
> ant debug install
```



Source:
http://developer.android.com/images/build.png

Hello World app running in emulator



APK (Application Package)

- Location:

IntelliJ: out/production/MyFirstApp/MyFirstApp.apk

Ant: bin/MyFirstApp-debug.apk

- APK is a zip archive of compiled app

```
> cp MyFirstApp.apk MyFirstApp.zip
> unzip -l MyFirstApp.zip
```

```
 Length     Date   Time    Name
--------    ----   ----    ----
     552  01-01-13 12:39   res/layout/main.xml
    1444  01-01-13 12:39   AndroidManifest.xml
     704  01-01-13 12:39   resources.arsc
    2936  01-01-13 12:39   classes.dex
     327  01-01-13 12:39   META-INF/MANIFEST.MF
     380  01-01-13 12:39   META-INF/CERT.SF
     776  01-01-13 12:39   META-INF/CERT.RSA
```

ADB (Android Debug Bridge)

```
> adb devices                          # List connected devices

> adb shell pm list packages           # List installed packages

> adb shell pm list features           # List supported features

> adb install <apk path>               # Install APK

> adb uninstall <package-name>         # Uninstall App

> adb logcat                           # View logs
```

Documentation

API Reference: http://developer.android.com/reference/packages.html

* Exercise:

- Add logging to Hello World app:

    Add a debug log message in onCreate that outputs something when onCreate is called

    HINT: Answer should only require you to add two lines to the code

* Steps:

- Search for "log" in API reference
- First hit is for class android.util.Log (method: Log.d).
- Add to MyActivity.java's onCreate method

    Log.d(“hackweek”, “onCreate is called”);

```
> adb logcat | grep hackweek
```

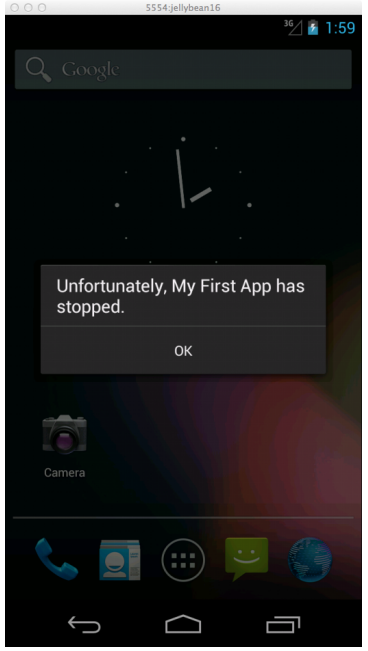
- Run app

Crashes

- Add to onCreate:

```
final ArrayList<String> list = null;
list.add("Android"); // boom! NPE.
```

- Run (Shift+F10)



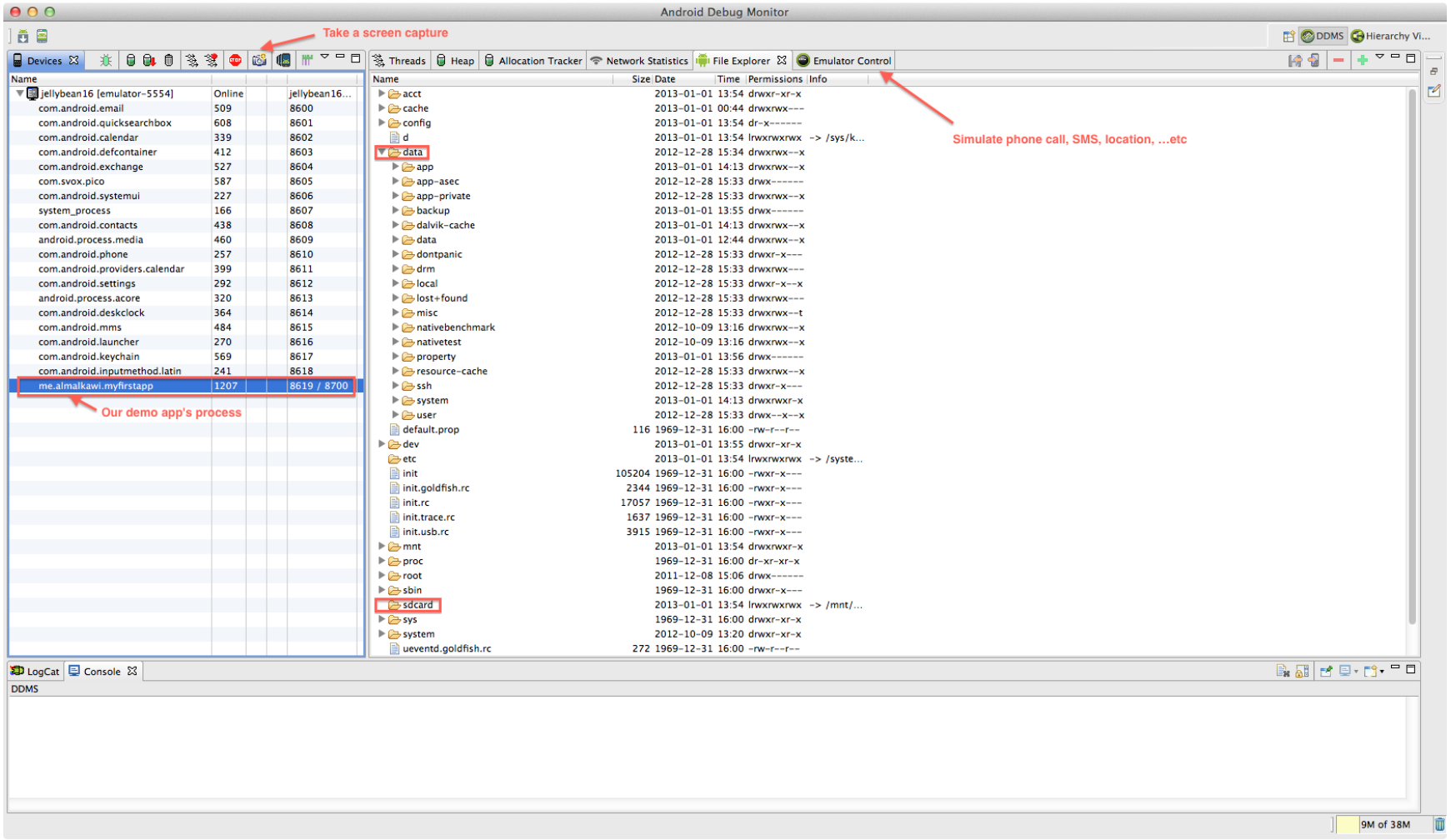
- Get crash call stack

```
> adb logcat *:E
```

```
E/AndroidRuntime(  635): FATAL EXCEPTION: main
E/AndroidRuntime(  635): java.lang.RuntimeException: Unable to start activity ComponentInfo{me.almalkawi.myfirstapp/me.almalkawi.myfirstapp.MyActivity}: java.lang.NullPointerException
E/AndroidRuntime(  635):        at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2059)
E/AndroidRuntime(  635):        at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2084)
E/AndroidRuntime(  635):        at android.app.ActivityThread.access$600(ActivityThread.java:130)
E/AndroidRuntime(  635):        at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1195)
E/AndroidRuntime(  635):        at android.os.Handler.dispatchMessage(Handler.java:99)
E/AndroidRuntime(  635):        at android.os.Looper.loop(Looper.java:137)
E/AndroidRuntime(  635):        at android.app.ActivityThread.main(ActivityThread.java:4745)
E/AndroidRuntime(  635):        at java.lang.reflect.Method.invokeNative(Native Method)
E/AndroidRuntime(  635):        at java.lang.reflect.Method.invoke(Method.java:511)
E/AndroidRuntime(  635):        at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:786)
E/AndroidRuntime(  635):        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:553)
E/AndroidRuntime(  635):        at dalvik.system.NativeStart.main(Native Method)
E/AndroidRuntime(  635): Caused by: java.lang.NullPointerException
E/AndroidRuntime(  635):        at me.almalkawi.myfirstapp.MyActivity.onCreate(MyActivity.java:18)
E/AndroidRuntime(  635):        at android.app.Activity.performCreate(Activity.java:5008)
E/AndroidRuntime(  635):        at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1079)
E/AndroidRuntime(  635):        at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2023)
E/AndroidRuntime(  635):        ... 11 more
```

Android Monitor

IntelliJ: Tools->Android->Monitor (or `> monitor`)

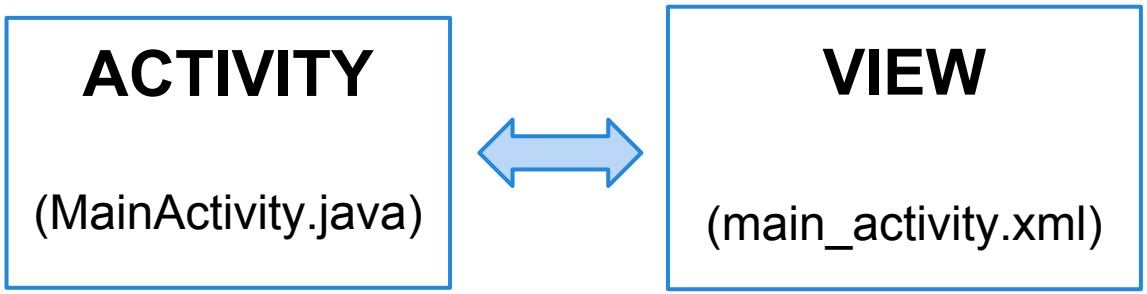


Hello Android!

Building basic UI

Activities

- Roughly analogous to Controller in MVC model
- On a basic level, you can think of an Android app as interacting with a series of Activities
- Only one activity has focus on an Android device at any one time
- Use *intents* to move between activities (more on that later)
- Usually, but not always, associated with a View (via an xml layout file)



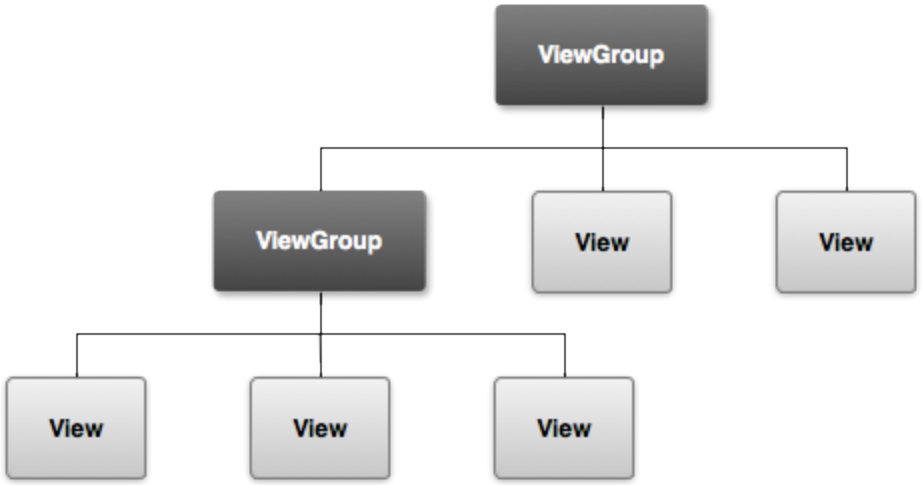
UI: Hierarchy of Views

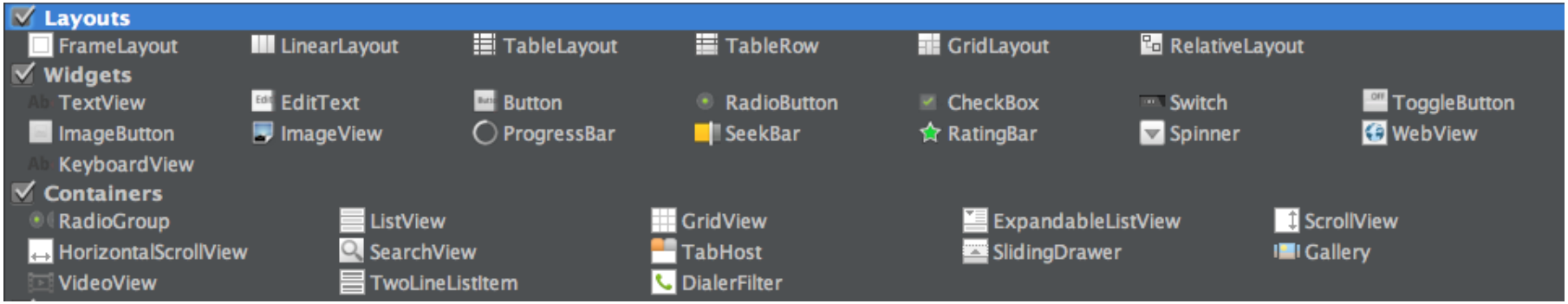
- ViewGroups:

Layouts and Containers
(e.g. LinearLayout)

- Views:

Widgets (e.g. Button)





Create New Project (SimpleUI)

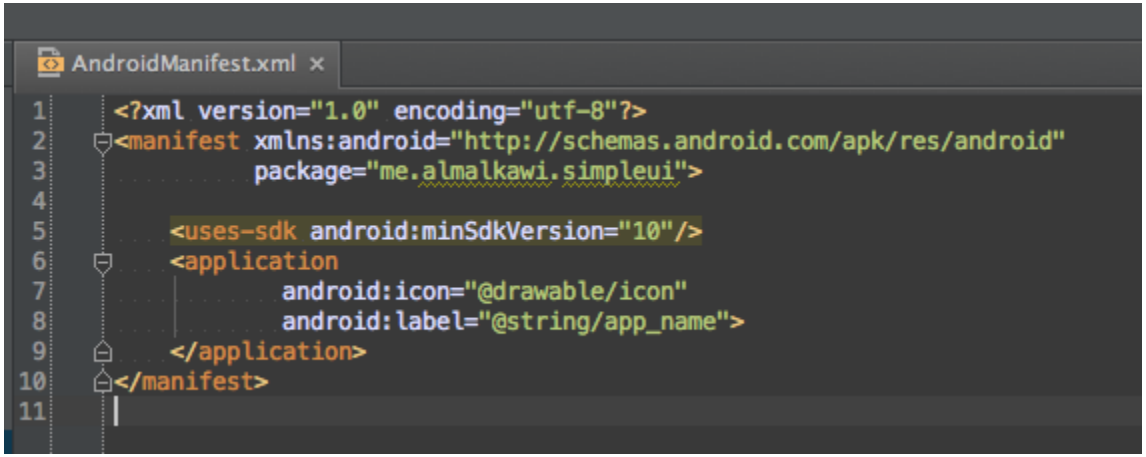
- Create new Android project:

Name: SimpleUI

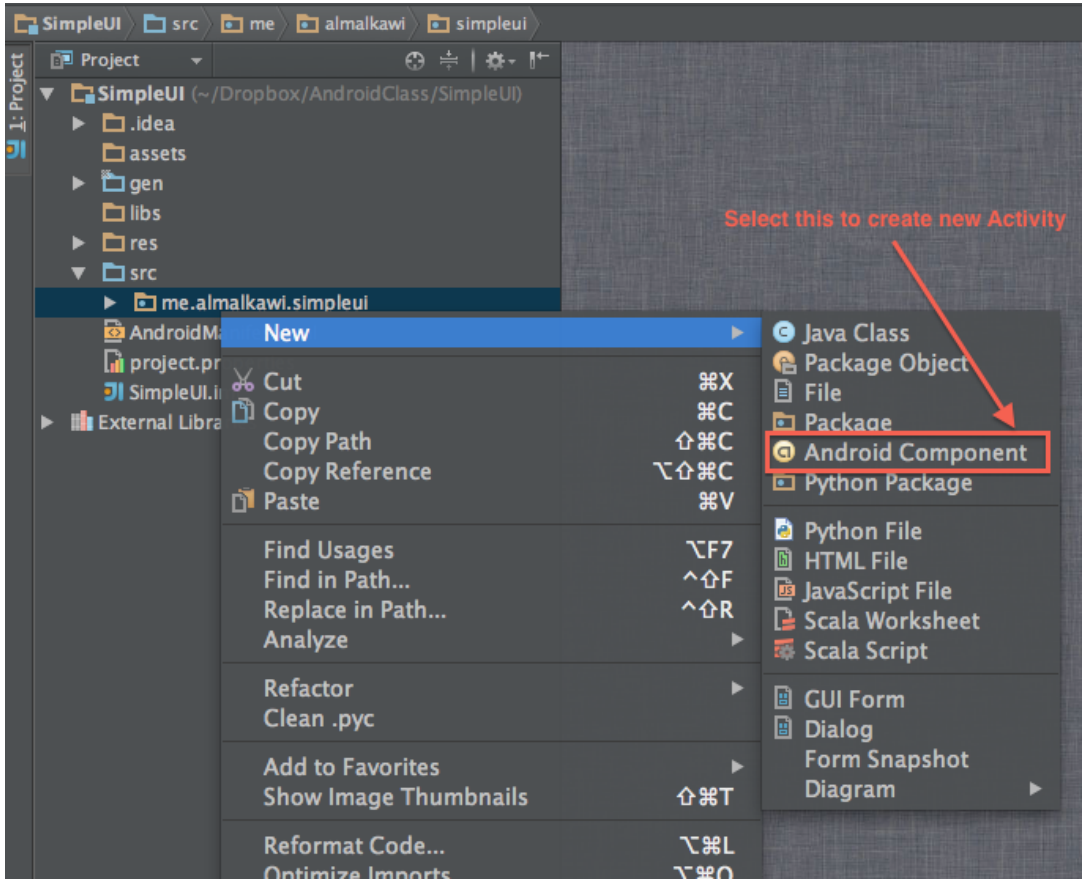
Package: com.example.simpleui

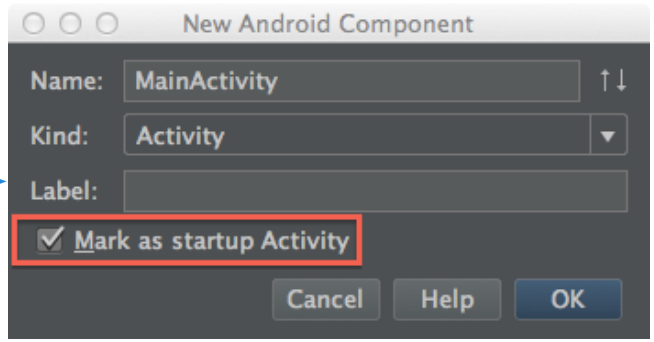
Uncheck: "Create Hello World activity"

- Open the AndroidManifest.xml file:



Create startup Activity

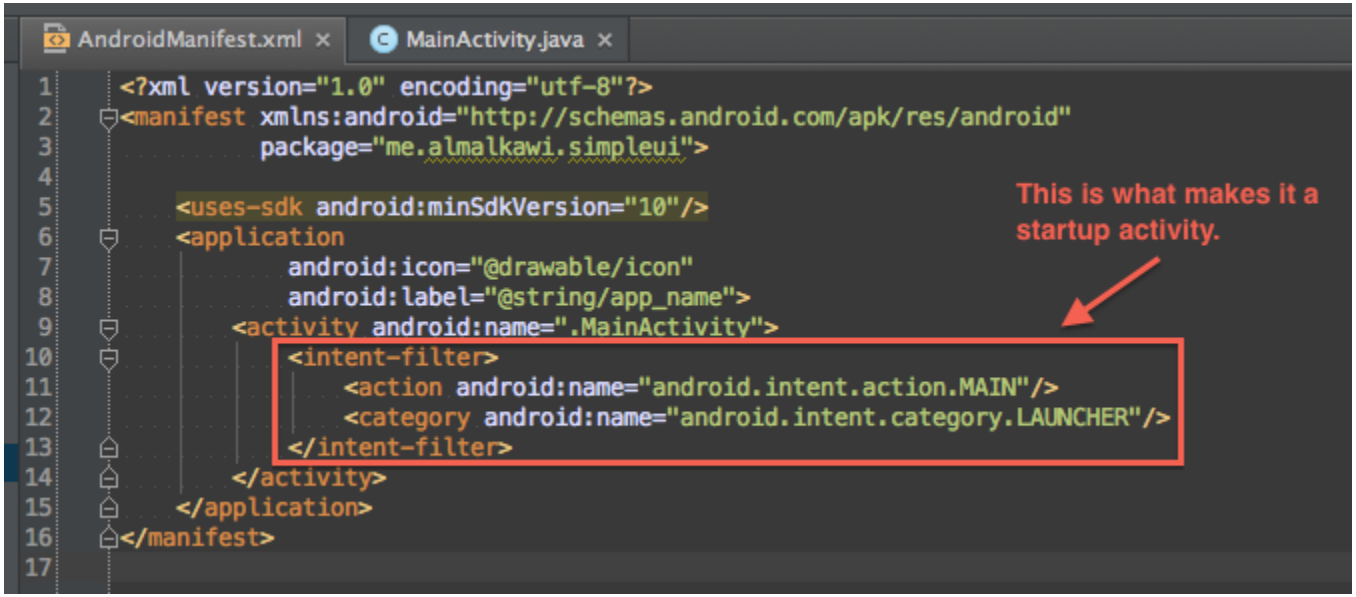




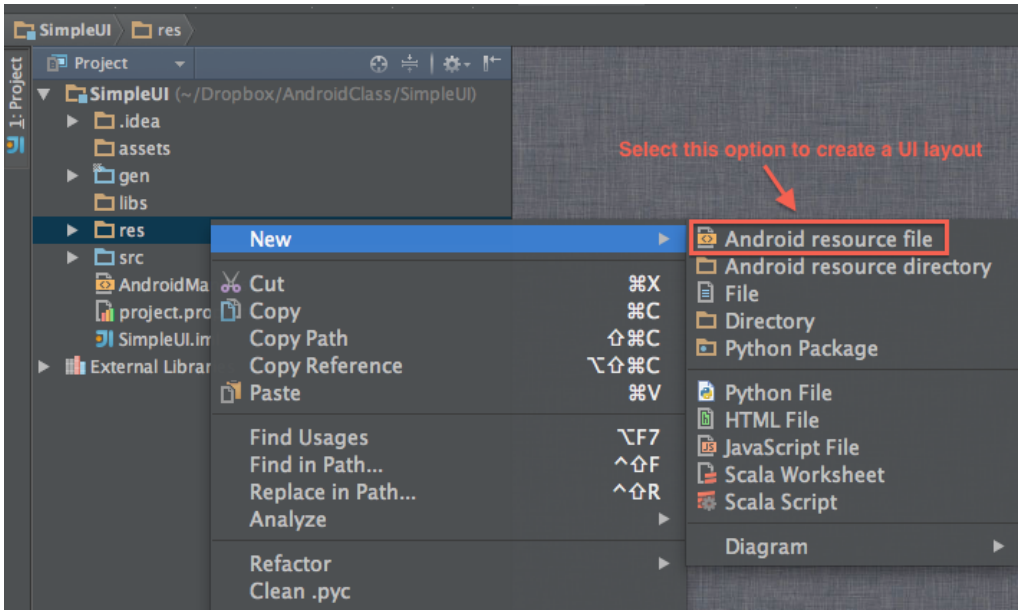
Activity in AndroidManifest.xml

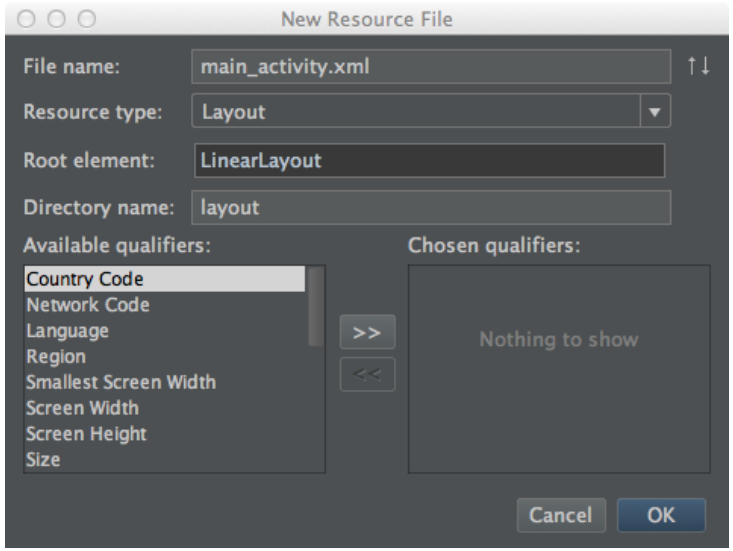
- Every activity is declared in manifest

- Declaration added automatically by IDE

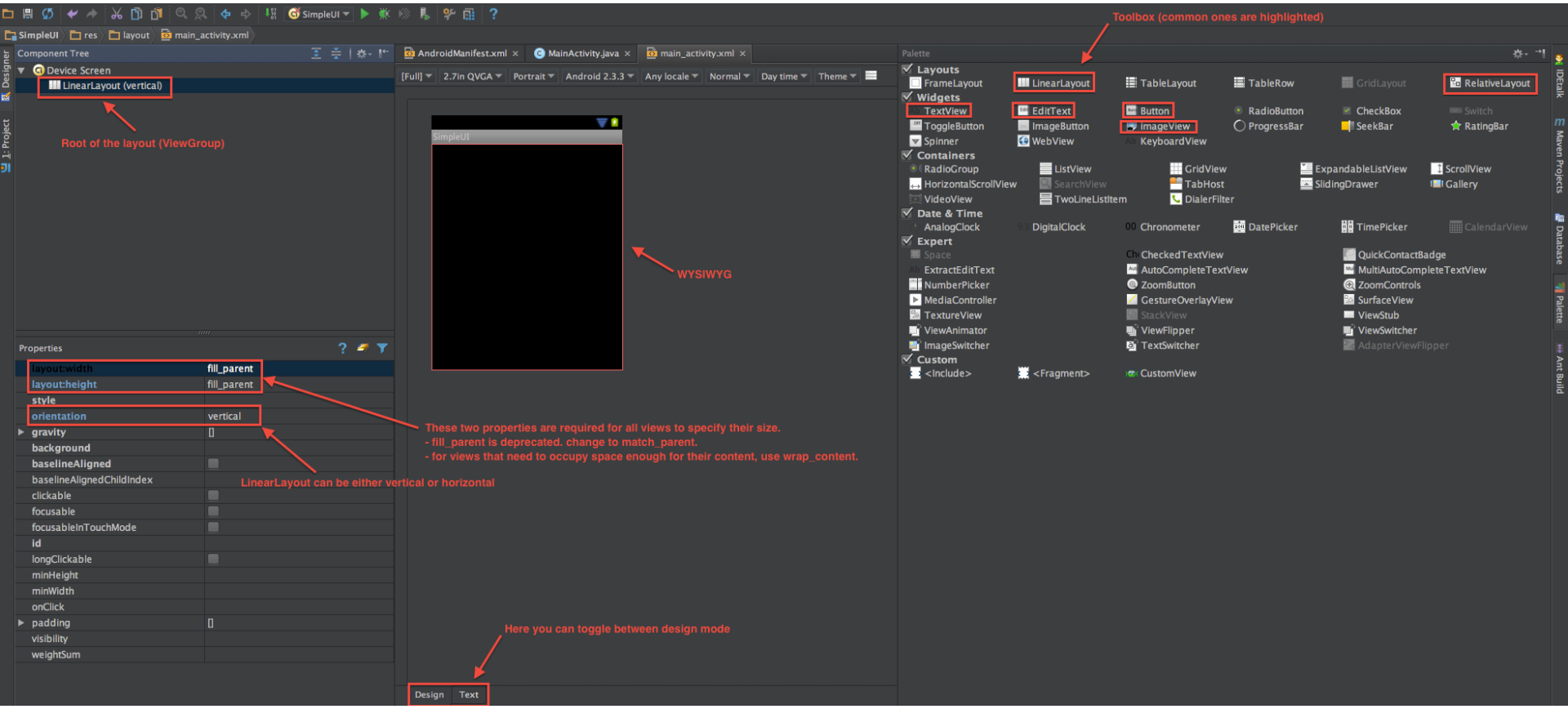


Create UI Layout





UI Designer


Toolbox (common ones are highlighted)
Component Tree
Device Screen
LinearLayout (vertical)
Root of the layout (ViewGroup)
AndroidManifest.xml
MainActivity.java
main_activity.xml
WYSIWYG
Palette
Layouts
FrameLayout
LinearLayout
TableLayout
TableRow
GridLayout
RelativeLayout
Widgets
TextView
EditText
Button
RadioButton
CheckBox
Switch
ToggleButton
ImageButton
ImageView
ProgressBar
SeekBar
RatingBar
Spinner
WebView
KeyboardView
Containers
RadioGroup
ListView
GridView
ExpandableListView
ScrollView
HorizontalScrollView
SearchView
TabHost
SlidingDrawer
Gallery
VideoView
TwoLineListItem
DialerFilter
Date & Time
AnalogClock
DigitalClock
Chronometer
DatePicker
TimePicker
CalendarView
Expert
Space
CheckedTextView
QuickContactBadge
ExtractEditText
AutoCompleteTextView
MultiAutoCompleteTextView
NumberPicker
ZoomButton
ZoomControls
MediaController
GestureOverlayView
SurfaceView
TextureView
StackView
ViewStub
ViewAnimator
ViewFlipper
ViewSwitcher
ImageSwitcher
TextSwitcher
AdapterViewFlipper
Custom
<include>
<Fragment>
CustomView
Properties
layout:width
fill_parent
layout:height
fill_parent
style
orientation
vertical
gravity
background
baselineAligned
baselineAlignedChildIndex
clickable
focusable
focusableInTouchMode
id
longClickable
minHeight
minWidth
onClick
padding
visibility
weightSum
These two properties are required for all views to specify their size.
- fill_parent is deprecated. change to match_parent.
- for views that need to occupy space enough for their content, use wrap_content.
LinearLayout can be either vertical or horizontal
Here you can toggle between design mode
Design
Text

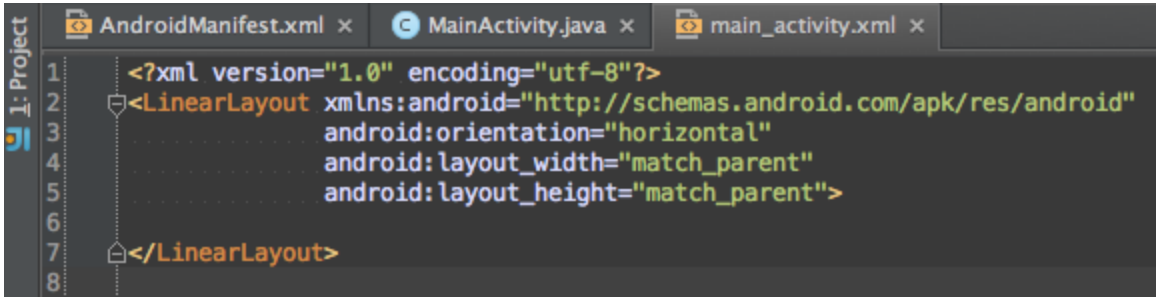
Basic View properties

The `layout_height` and `layout_width` can have the following values:

- `FILL_PARENT` - deprecated in API 8
- `MATCH_PARENT` - match the height or width of my View the height or width of the parent View
- `WRAP_CONTENT` - make my View as large as it needs to be to display the content inside
- `<numerical value>` - "`30dp`", can use px (pixels), dp (density-independent pixels), sp (scaled pixels), in (inches) or mm (millimeters)

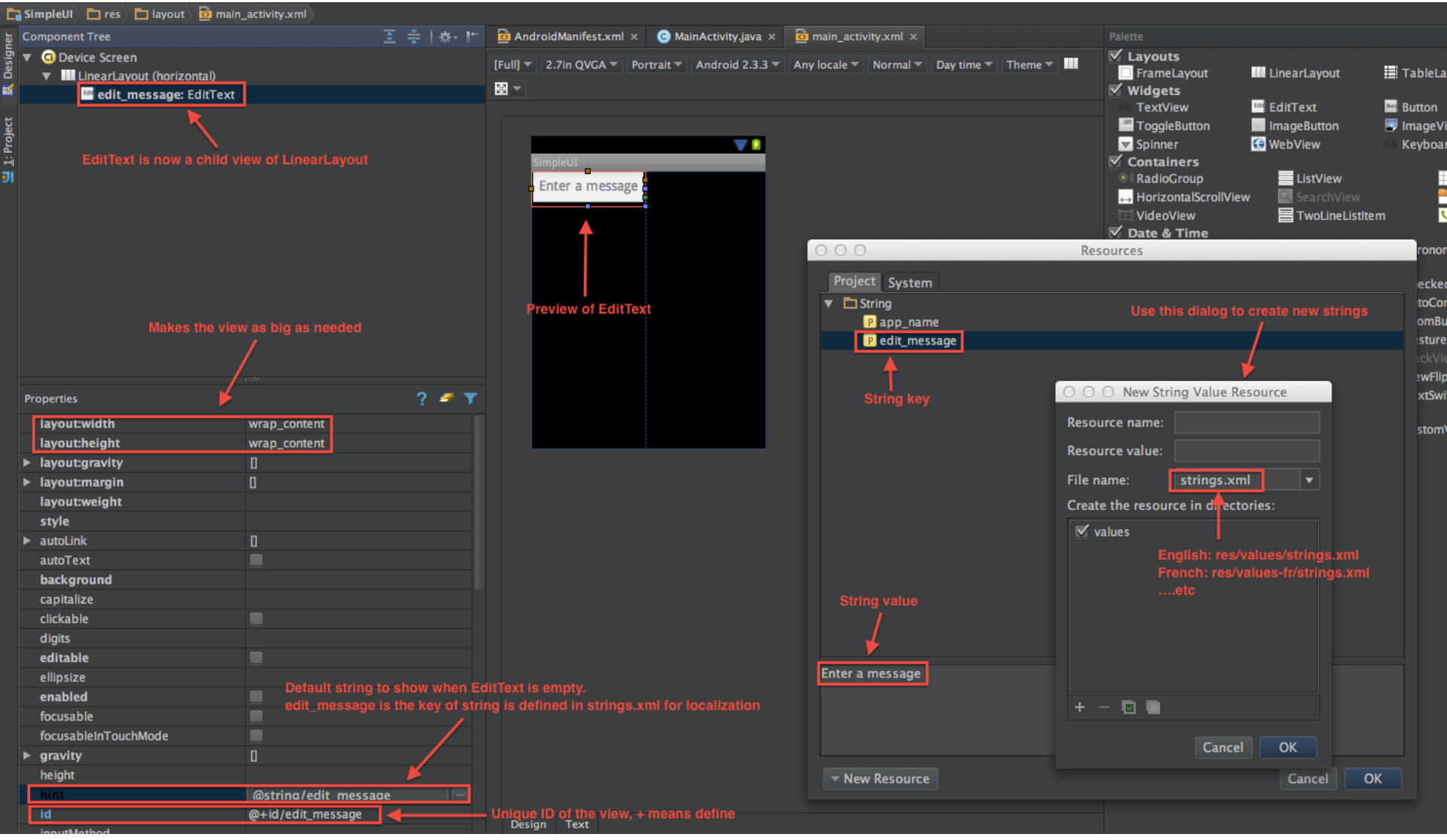
UI Design (text mode)

- XML layout file (main_activity.xml) - The Real Thing!



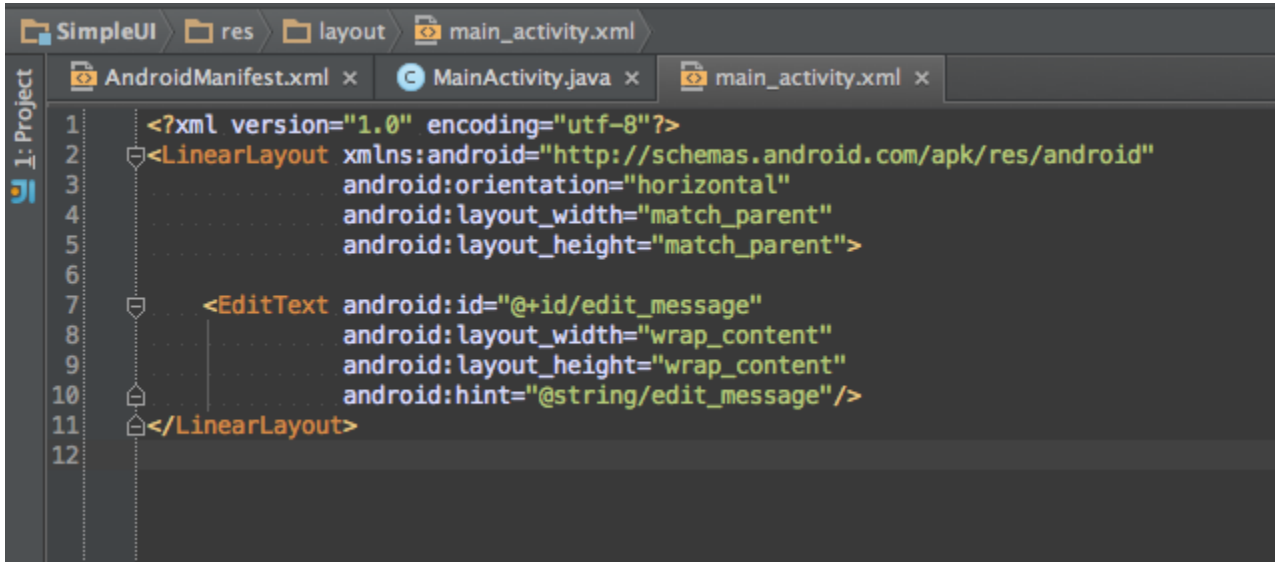
- Used to tweak things

Add EditText View

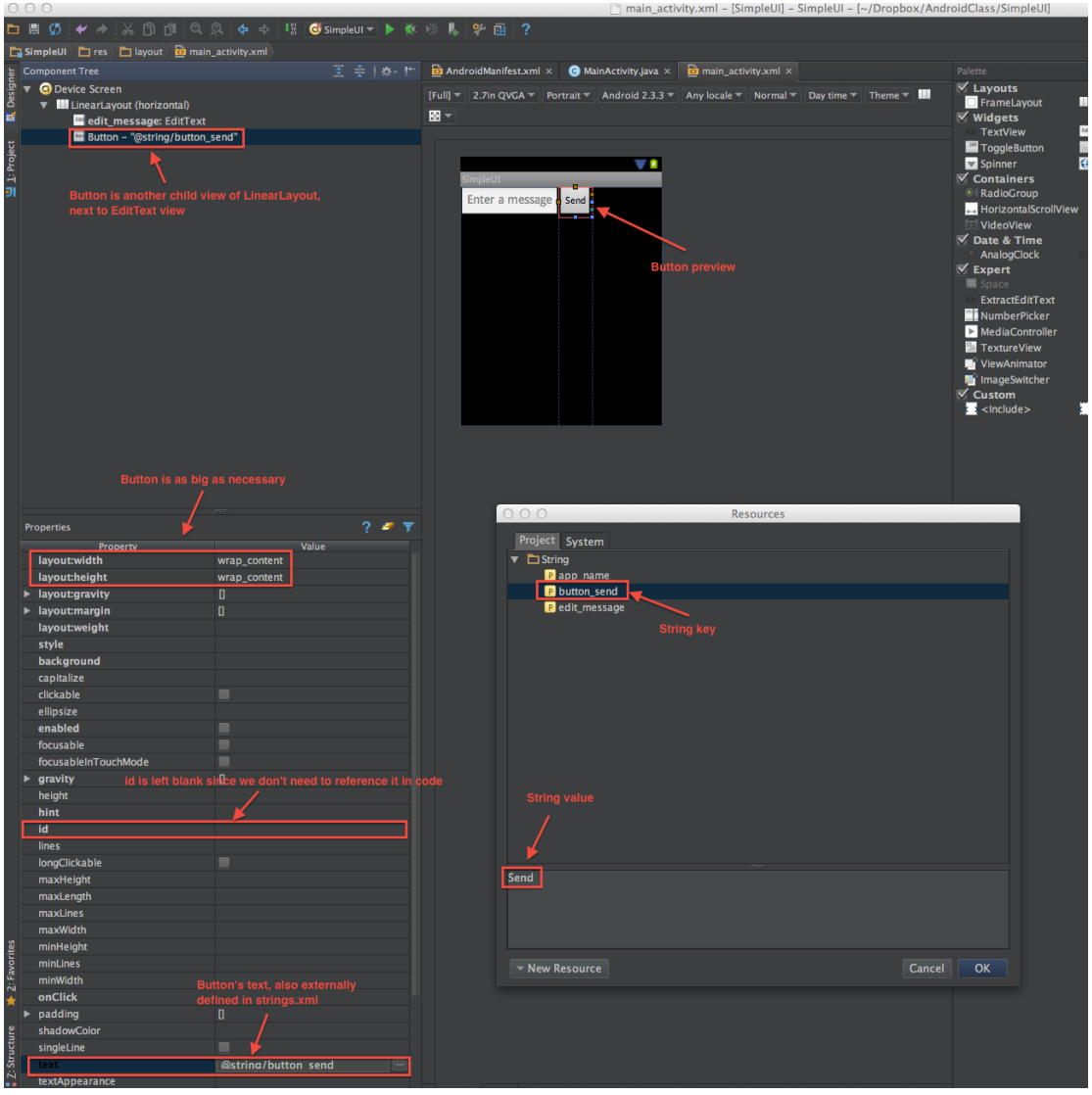


EditText in XML

The updated XML after adding EditText:

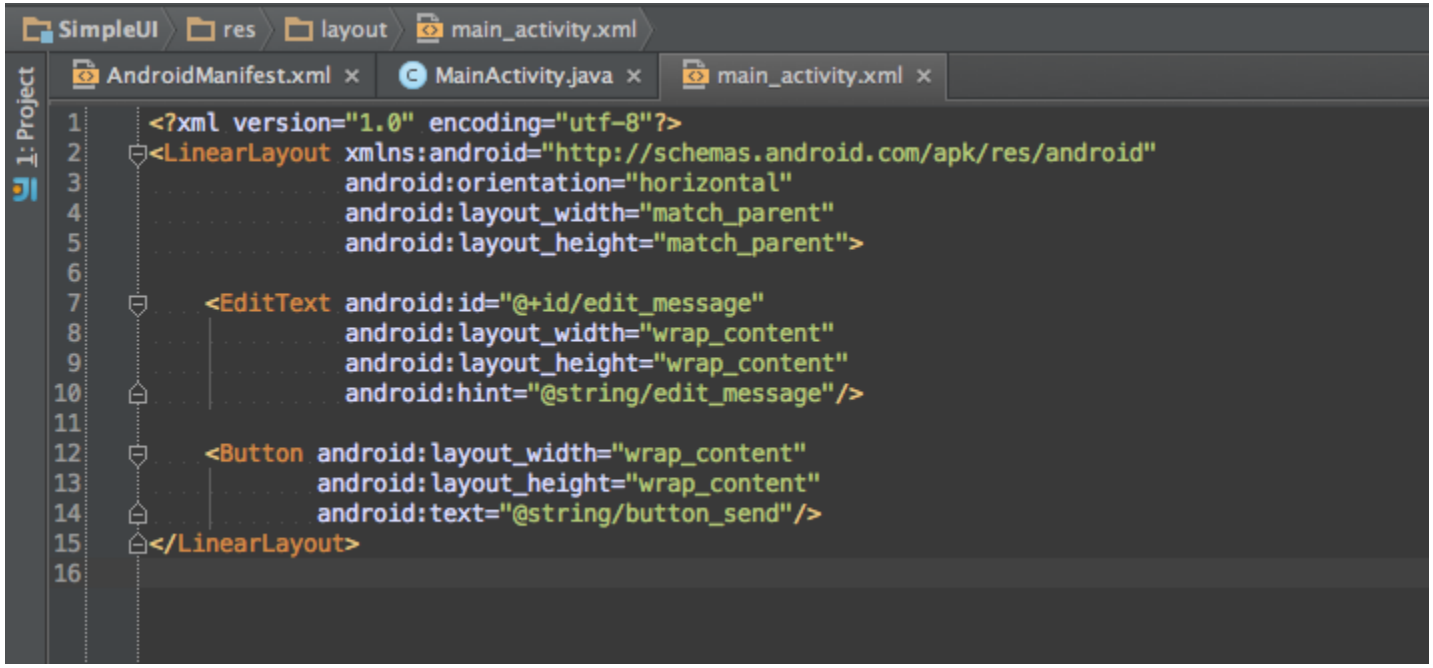


Add Button View

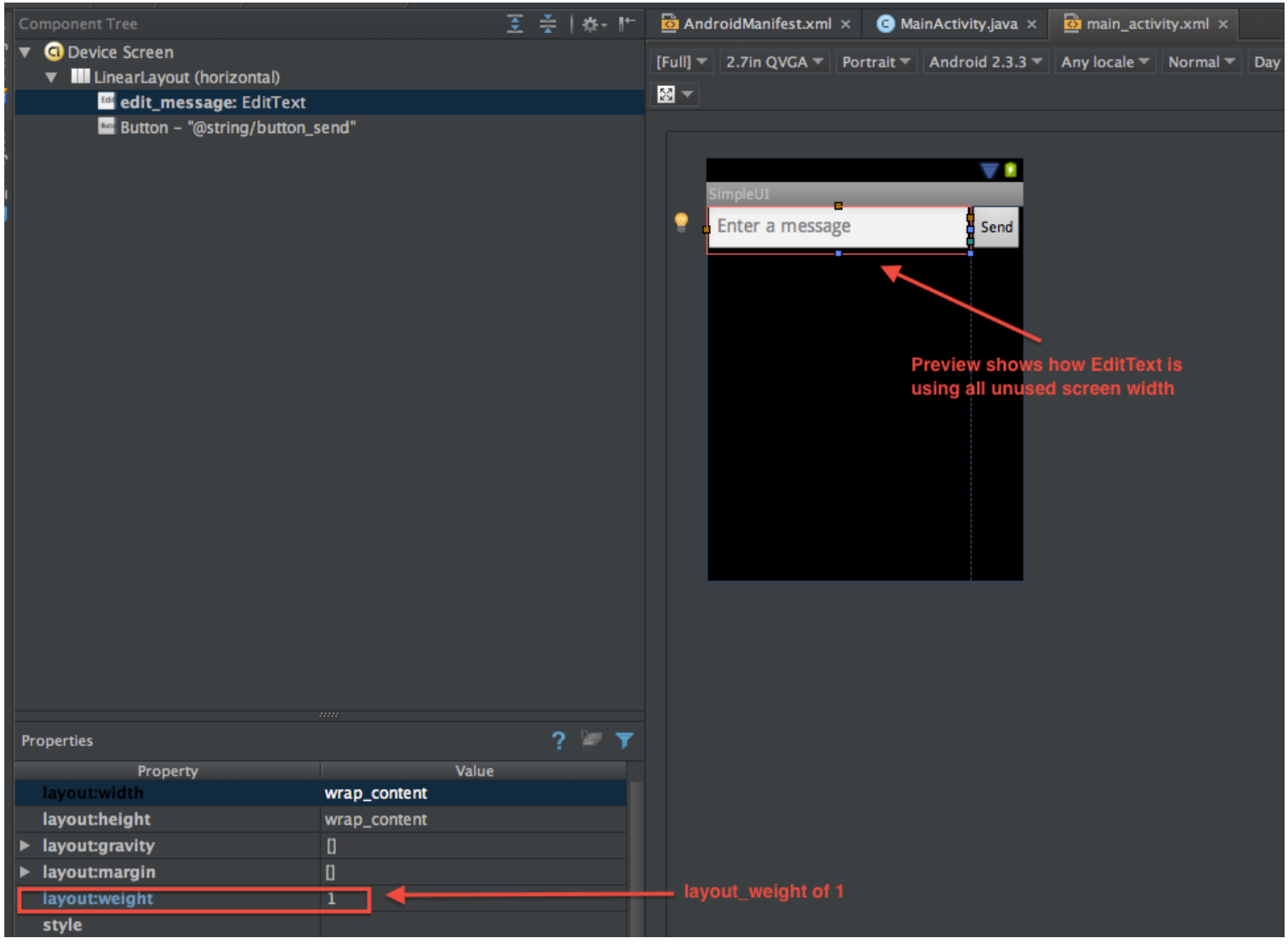


Button in XML

The updated XML after adding Button:



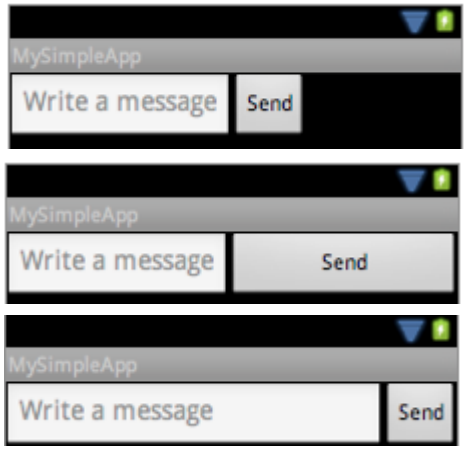
View weight


Component Tree
Device Screen
LinearLayout (horizontal)
edit_message: EditText
Button – "@string/button_send"
AndroidManifest.xml
MainActivity.java
main_activity.xml
[Full]
2.7in QVGA
Portrait
Android 2.3.3
Any locale
Normal
Day
SimpleUI
Enter a message
Send
Preview shows how EditText is using all unused screen width
Properties
Property
Value
layout:width
wrap_content
layout:height
wrap_content
layout:gravity
[]
layout:margin
[]
layout:weight
1
style
layout_weight of 1

layout_weight

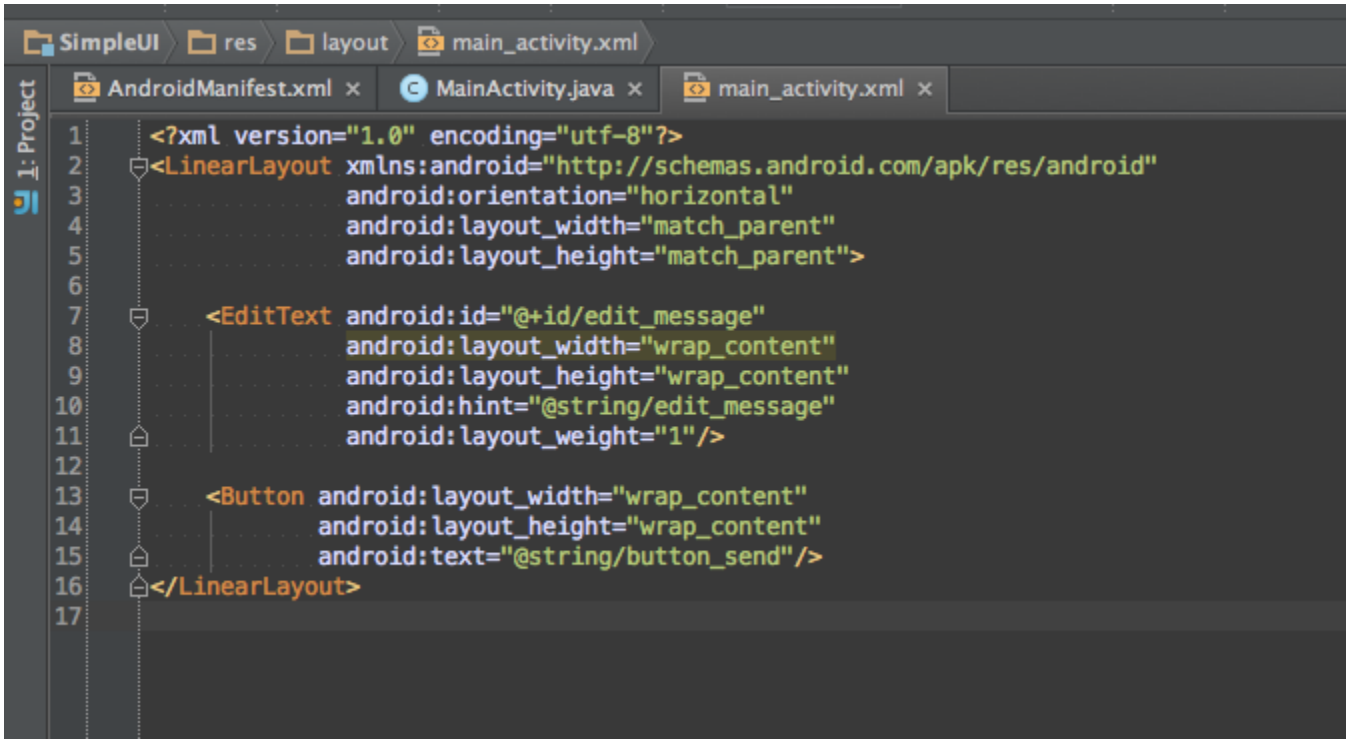
What does `layout_weight` do?

- Assigns the relative "importance" of a child view
- Determines which view gets any extra space available in a layout



EditText Weight	Button Weight
0	**0**
0	**1**
5	**1**

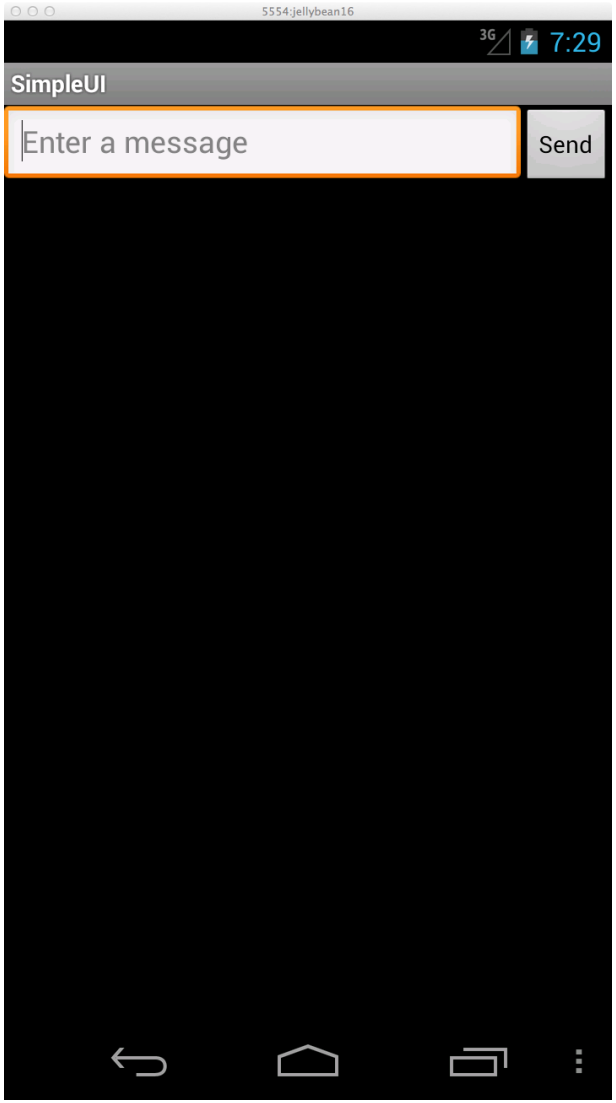
Completed UI (XML)



Loading UI resource in Activity

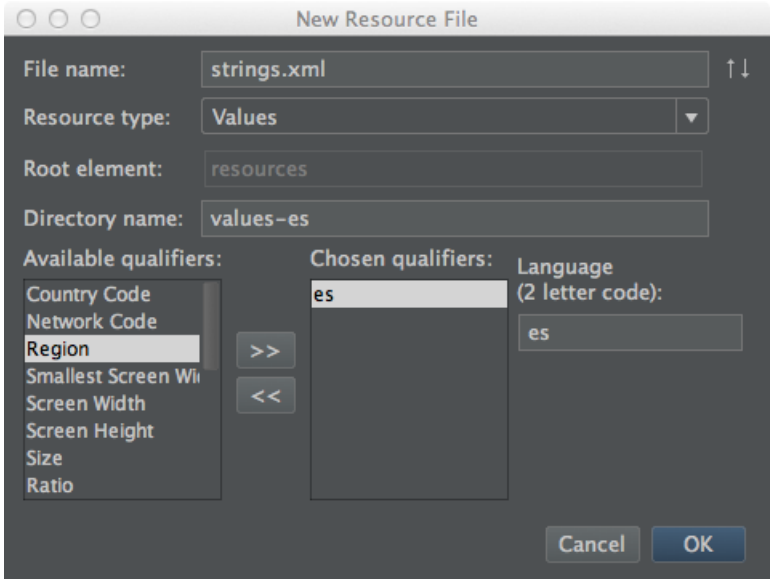
- Set layout in Activity (MainActivity.java, Command+N shortcut)

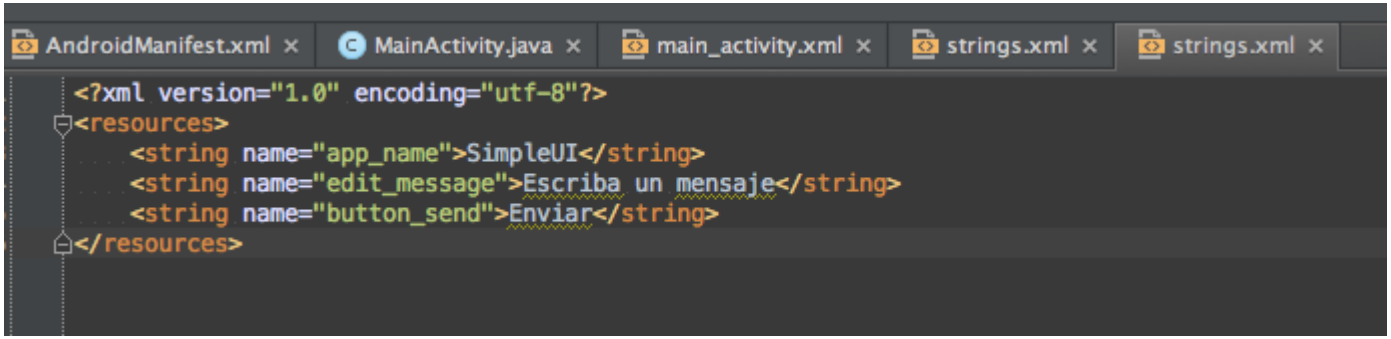
App running in emulator



Supporting multiple languages

- Create new Android resource file for Spanish





Test Spanish localization

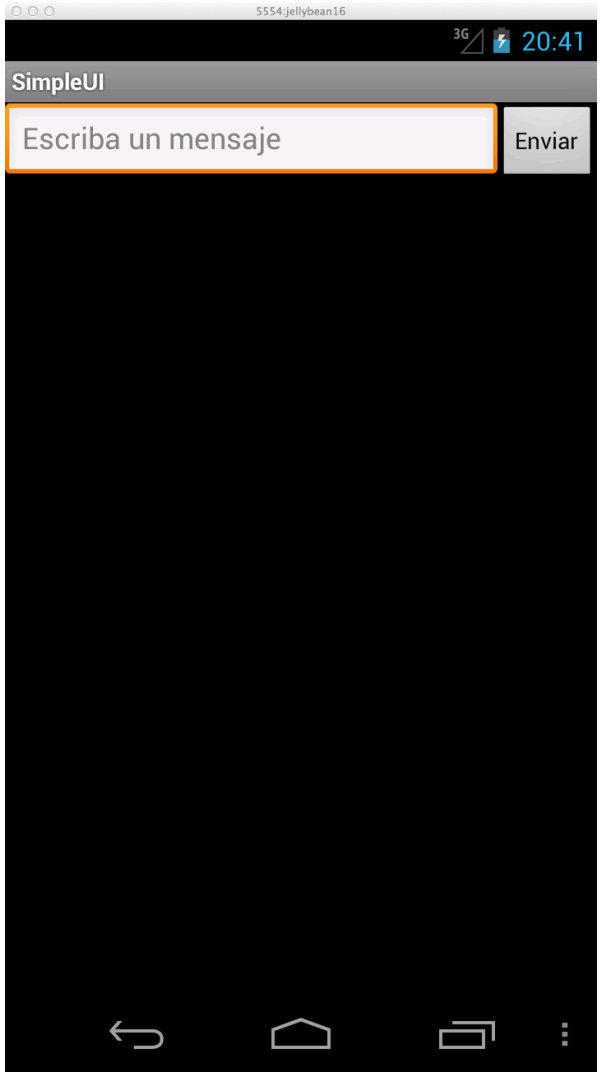
- Switch system language to Spanish

(Settings->Language & Input->Language)

- Build and install app (Shift+F10)

- To switch back to English:

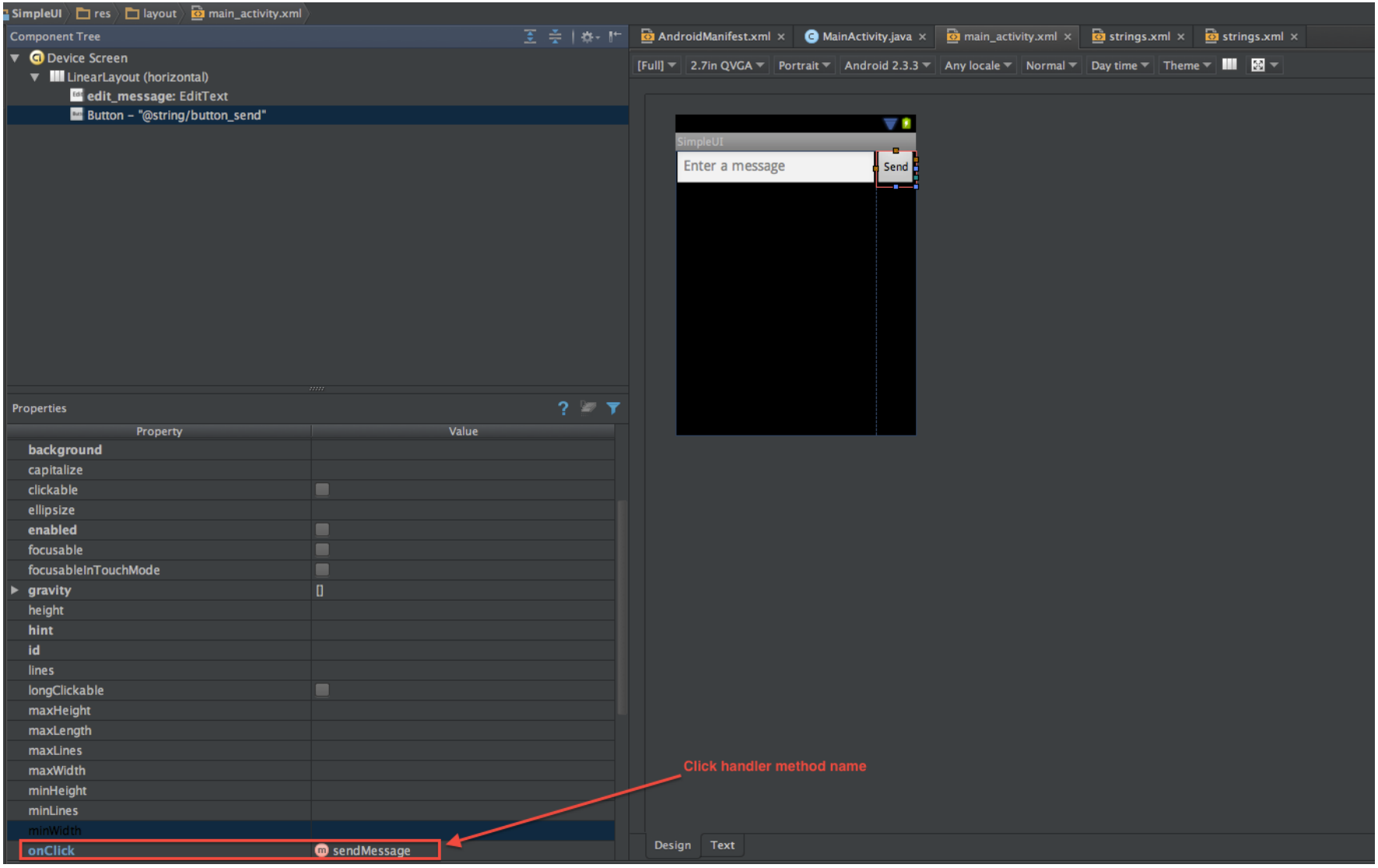
(Configuration -> Idioma)



Hello Android!

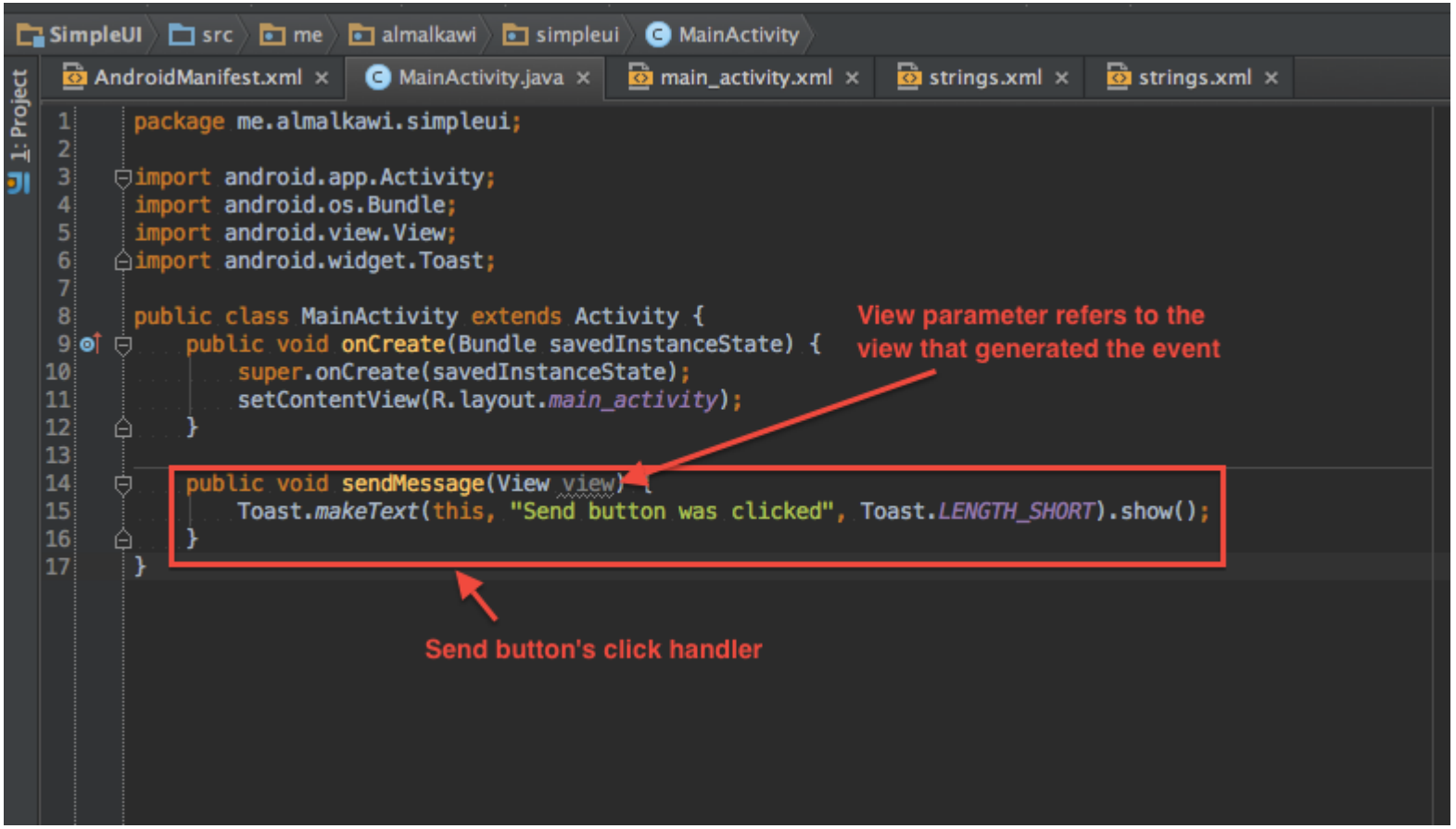
Event handlers

Specify click handler for Button

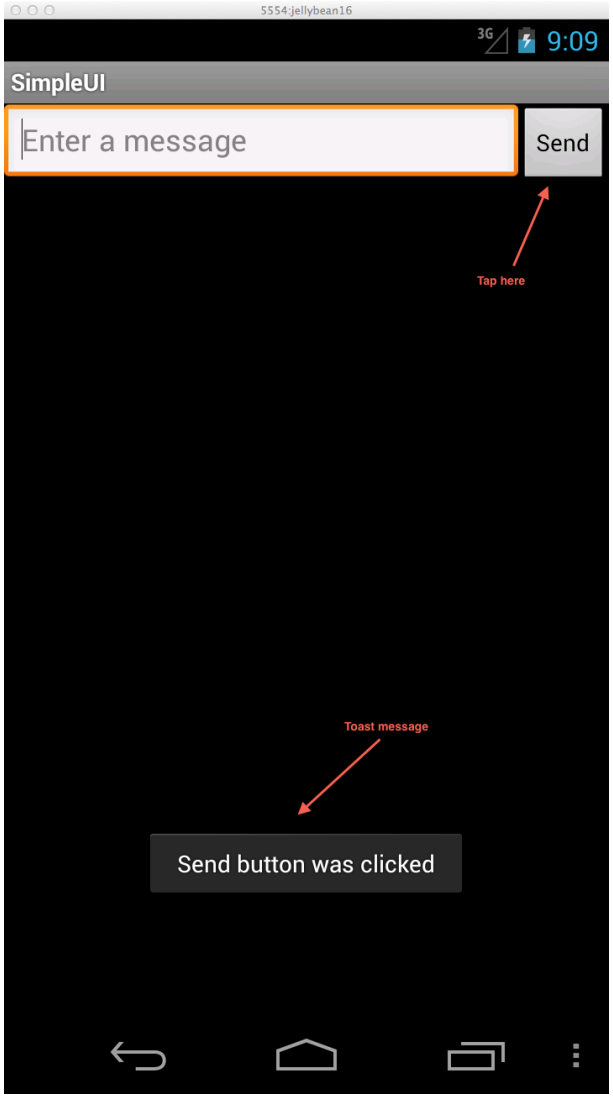


Implement click handler

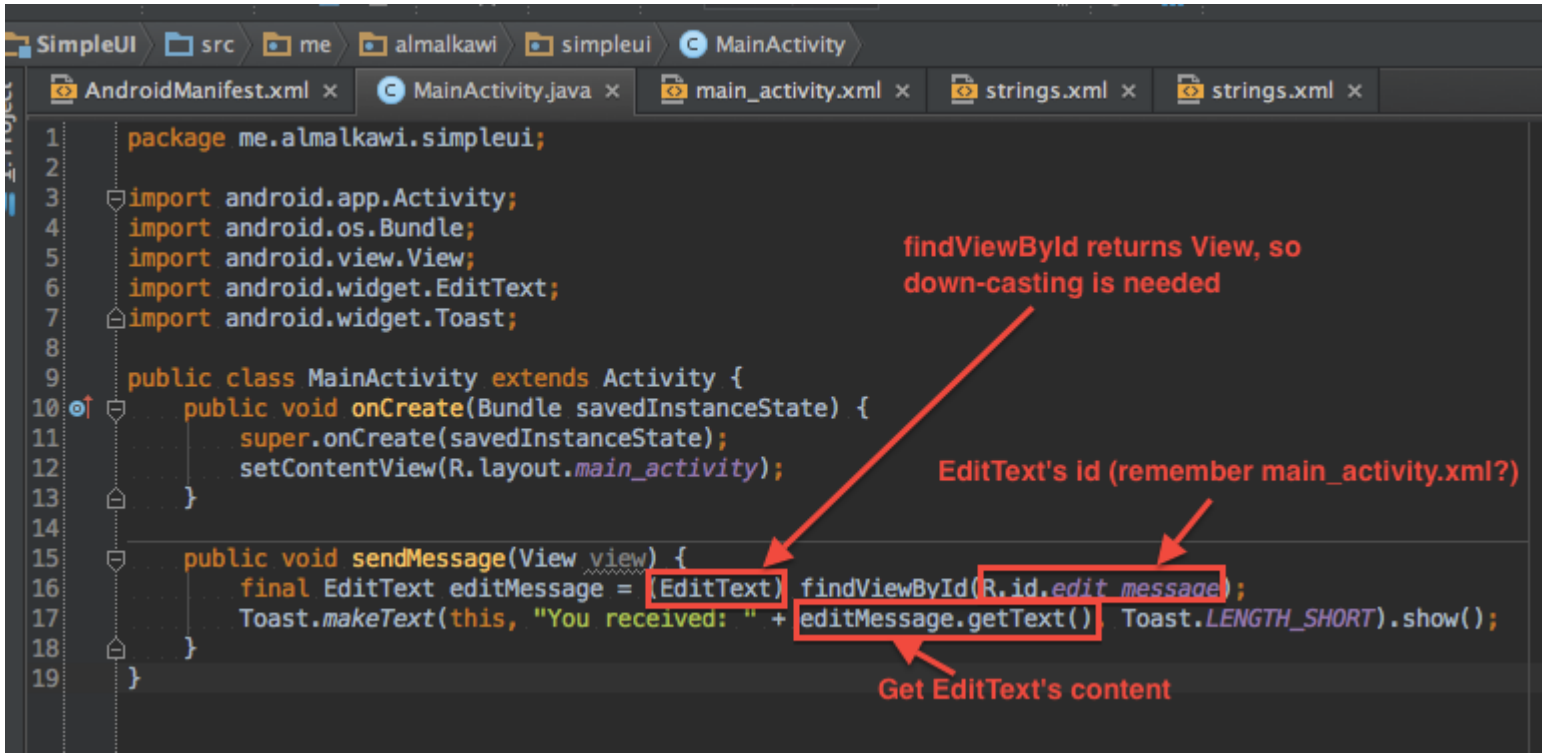
Add click handler to MainActivity.java



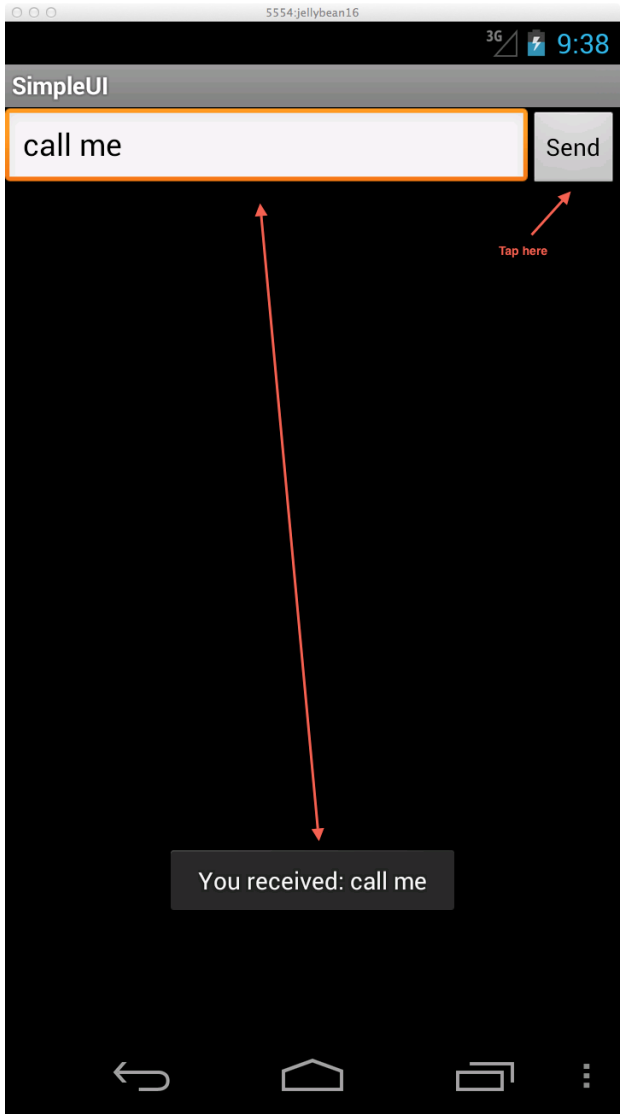
Test click handler



Show EditText content in Toast



Test reading EditText content

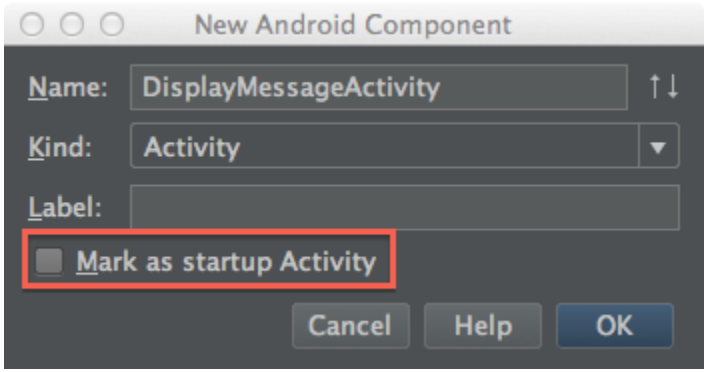


Hello Android!

Intents

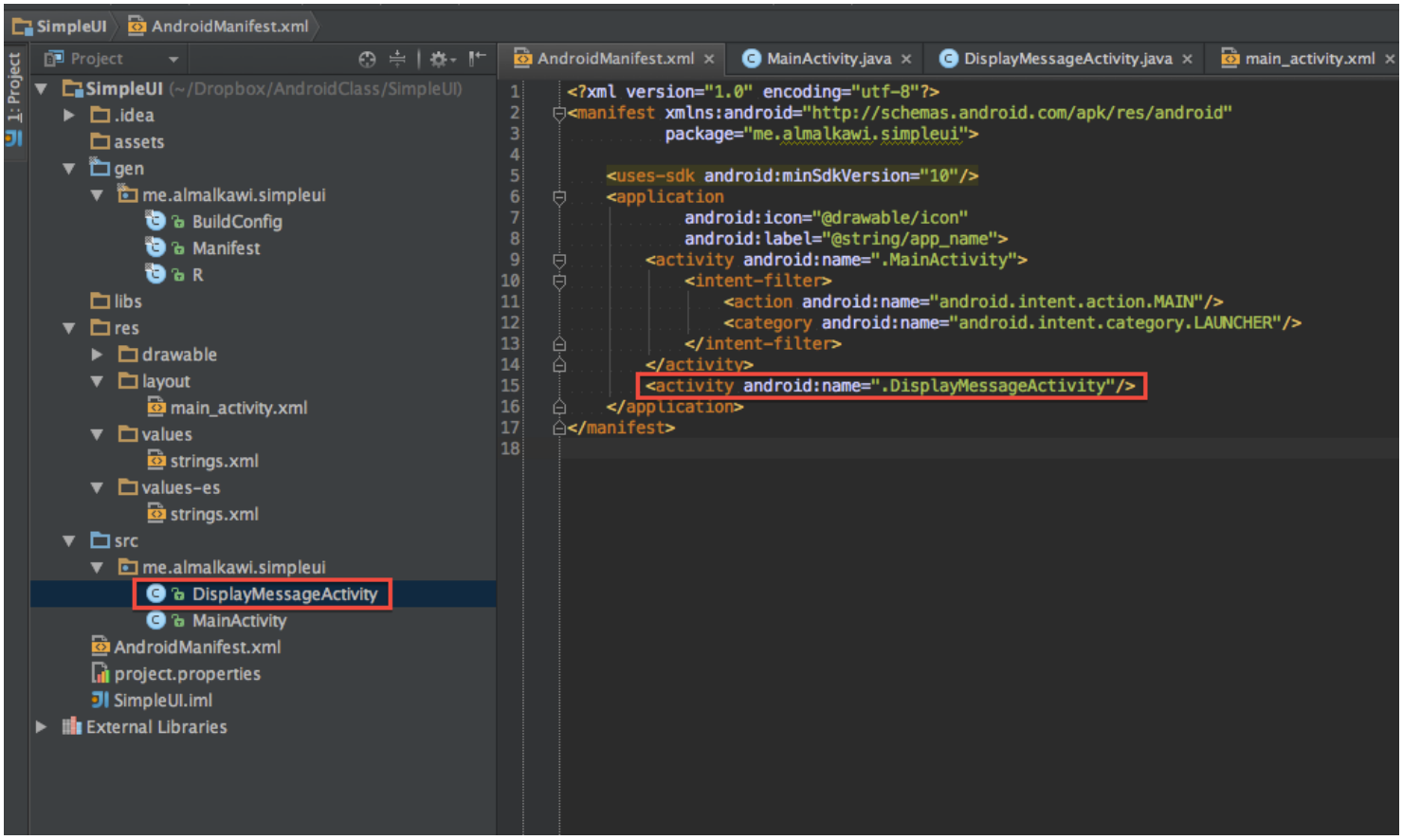
Add second Activity to SimpleUI

- Create a second Activity (DisplayMessageActivity)



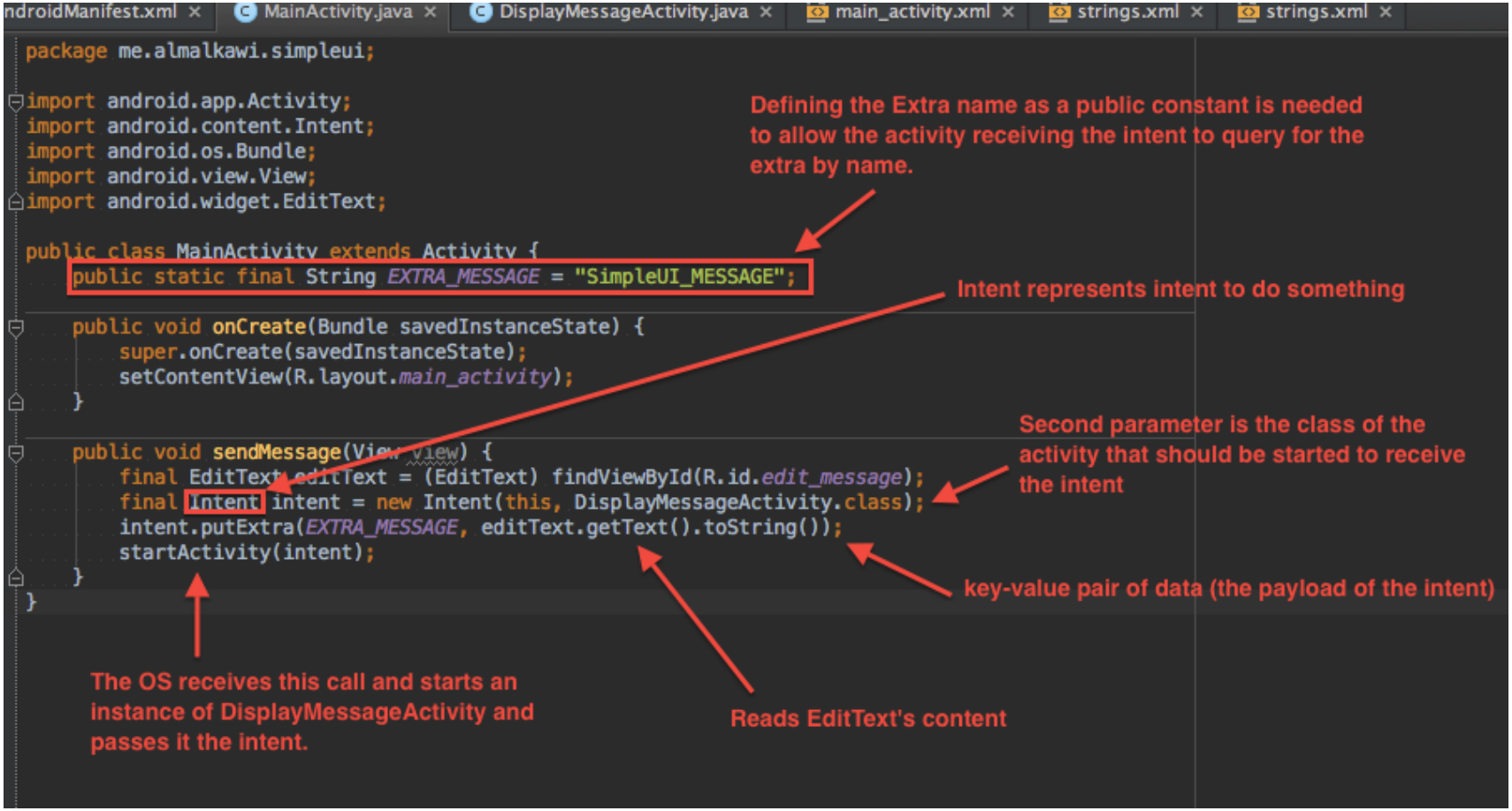
Second Activity declared in manifest

New Activity is automatically added to the manifest.



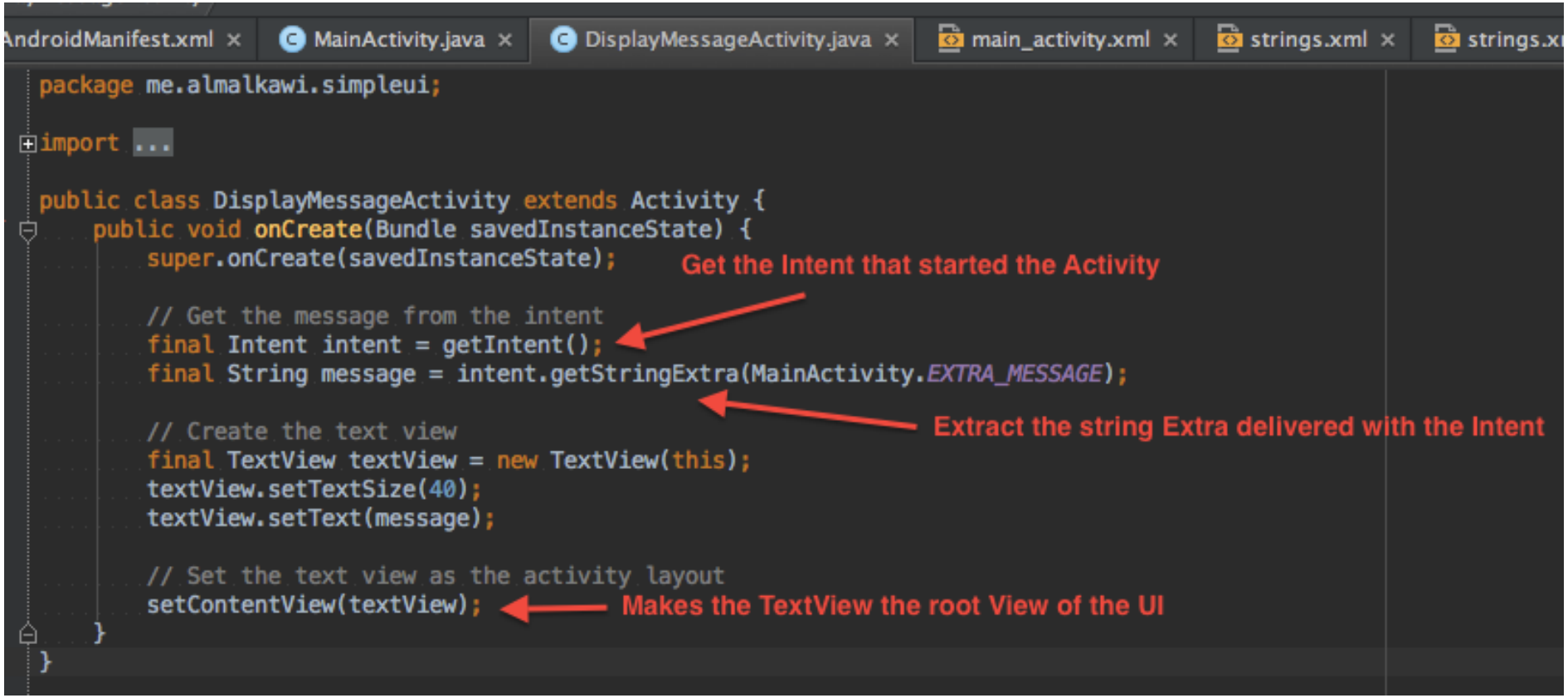
Create Intent

Create Intent to start the DisplayMessageActivity activity

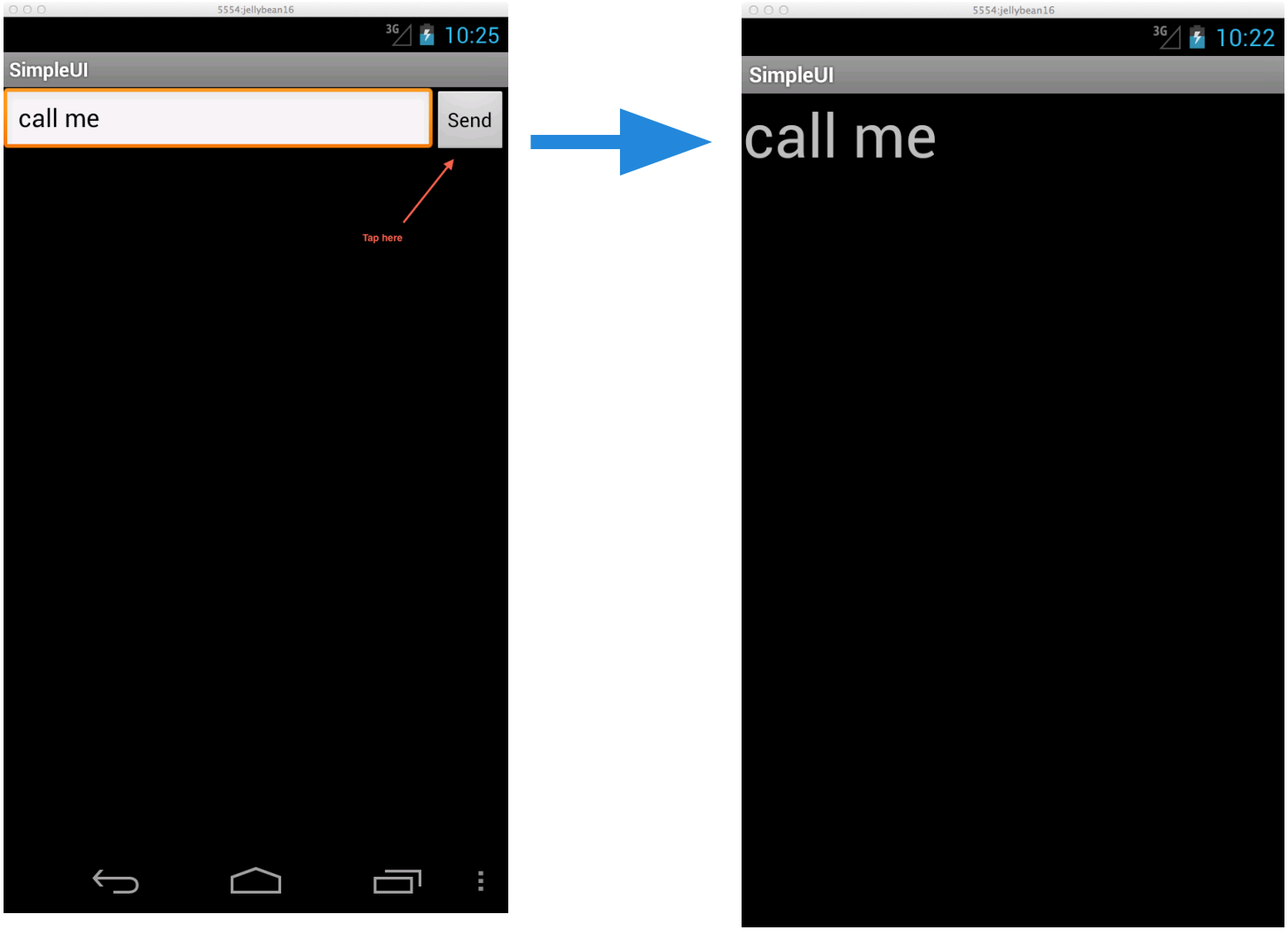


Second Activity's Implementation

Activity receiving the Intent Extra



First Activity starting Second Activity

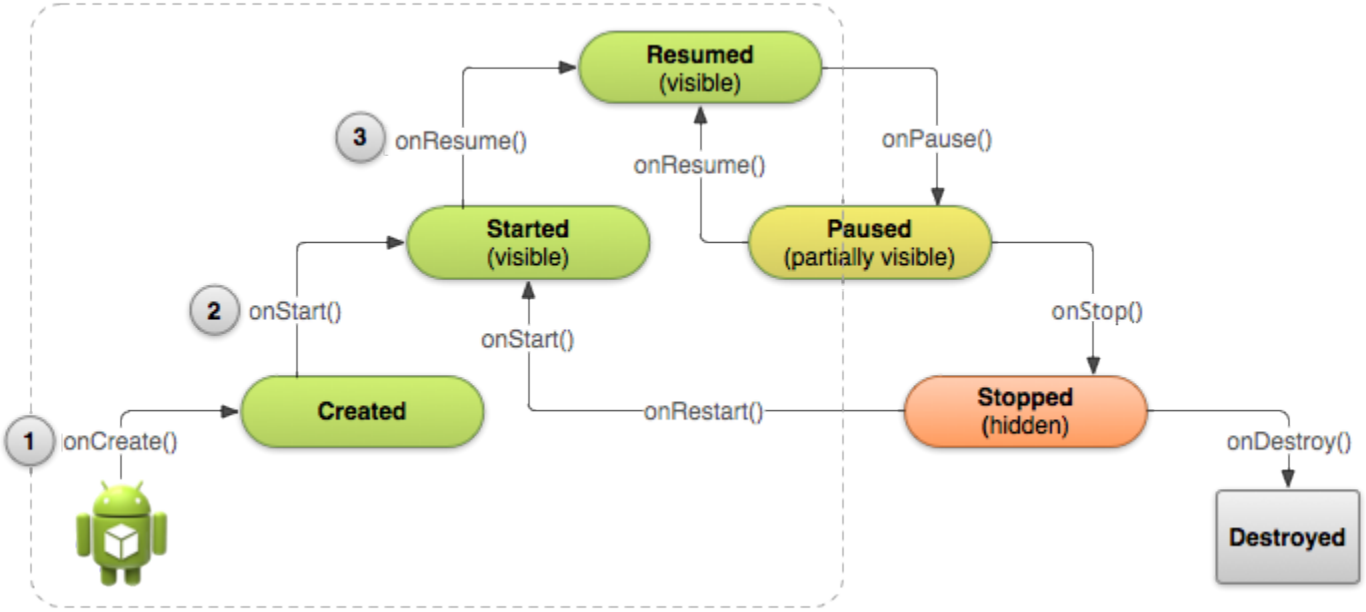


Hello Android!

Activity lifecycle

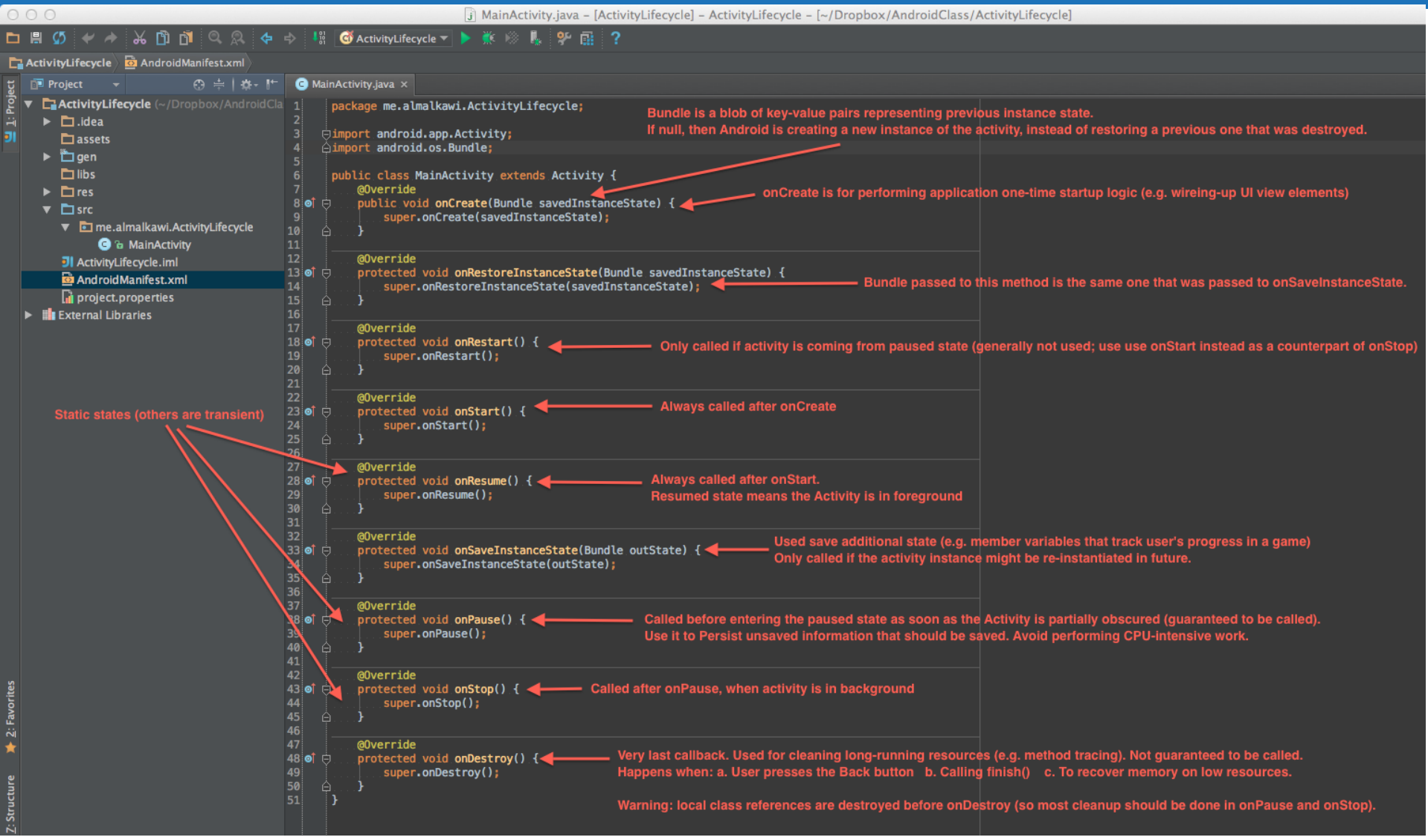
Activity Lifecycle

Activity instances transition between different states in their lifecycle



Source: http://developer.android.com/images/training/basics/basic-lifecycle-create.png

Activity lifecycle callbacks



Activity lifecycle demo

Code: https://github.com/almalkawi/AndroidClass/tree/master/ActivityLifecycle

Run the following scenarios in emulator:

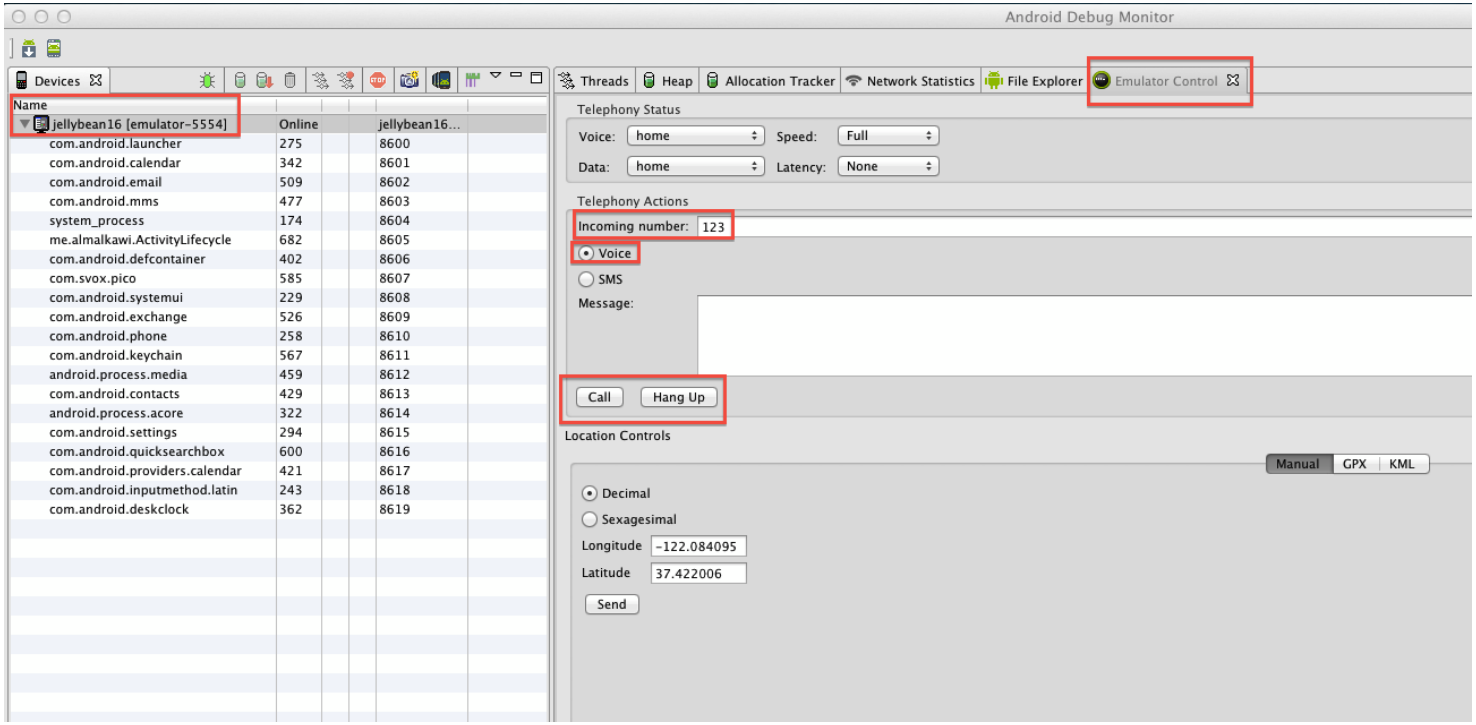
1. Start app for first time: onCreate (null Bundle), onStart, onResume
2. Stop app by clicking on home button: onSaveInstanceState, onPause, onStop
3. Re-start the app again (from Launcher): onRestart, onStart, onResume
4. Destroy the app by clicking on Back: onPause, onStop, onDestroy
5. Start the app again (from Launcher): onCreate (null Bundle), oStart, onResume
6. Press the “Finish” button: onPause, onStop, onDestroy
7. Rotate phone

onSaveInstanceState, onPause, onStop, onDestroy, onCreate (non-null bundle), onStart, onRestoreInstanceState, onResume

Activity lifecycle demo (phone calls)

Two more scenarios:

1. Receive call, then hangup quickly: onSaveInstanceState, onPause, onResume
2. Receive call, then answer: onSaveInstanceState, onPause (+ onStop when answered)



Hello Android!

AdapterViews and Adapters

AdapterViews and Adapters

AdapterView:

- View whose children are determined by an Adapter.
- Populated with data by binding it to an Adapter

Adapter:

- Retrieves data from external source and creates a View for each entry.
- Common external sources of data are arrays in memory and databases

ArrayAdapter and SimpleCursorAdapter

Android provides subclasses of Adapter for retrieving data and building views. Common ones:

- ArrayAdapter
- SimpleCursorAdapter

ArrayAdapter is used when data comes from an array.

SimpleCursorAdapter is used when data comes from a Cursor (DB).

Demo

ListView is an AdapterView that displays a list of scrollable items.

Example of ListView is tweets home timeline.

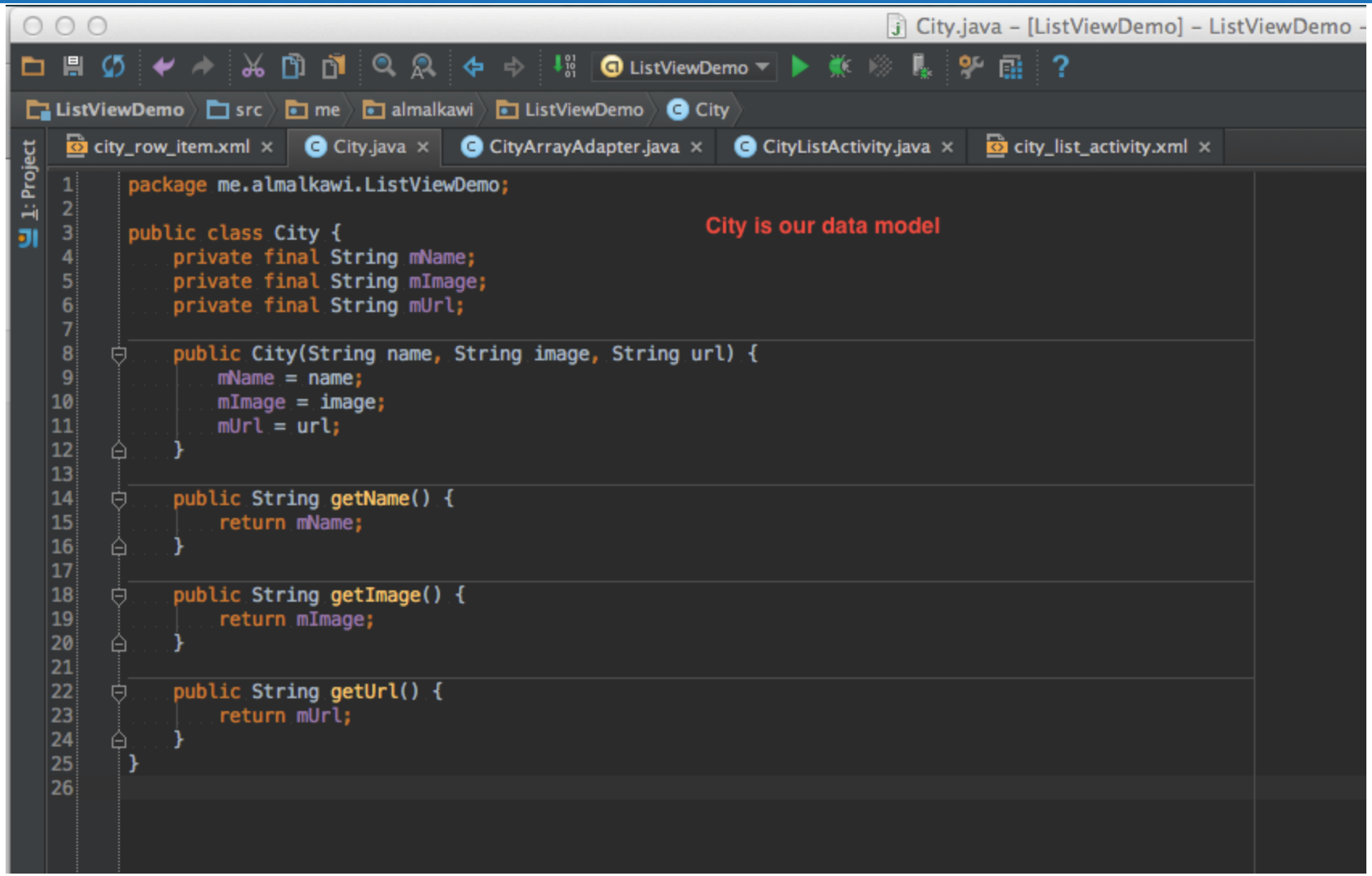
Create new Project (ListViewDemo)

Code Walkthroughs is in next set of slides

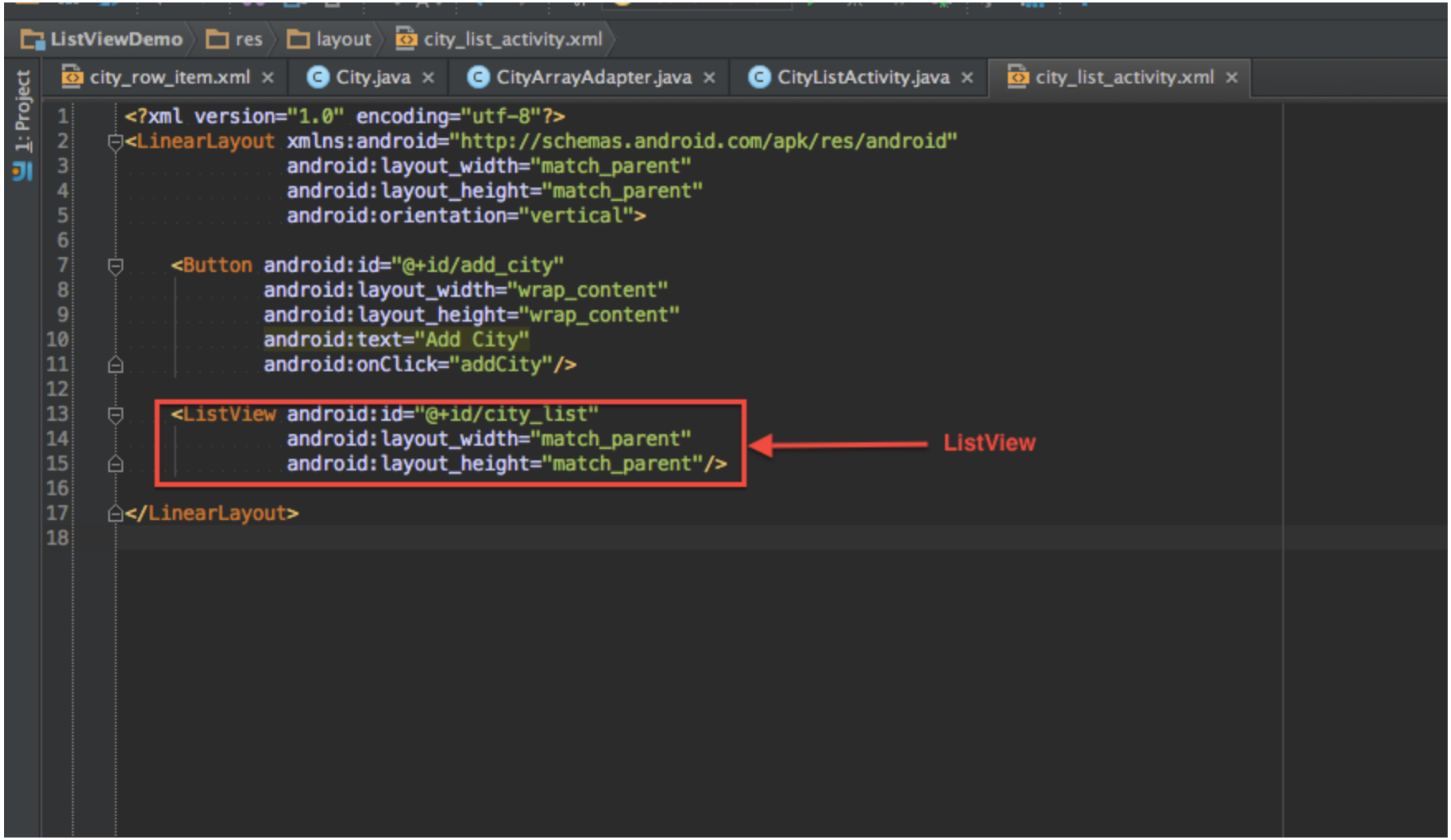
Images: https://github.com/almalkawi/AndroidClass/tree/master/ListViewDemo/res/drawable

Complete Code: https://github.com/almalkawi/AndroidClass/tree/master/ListViewDemo

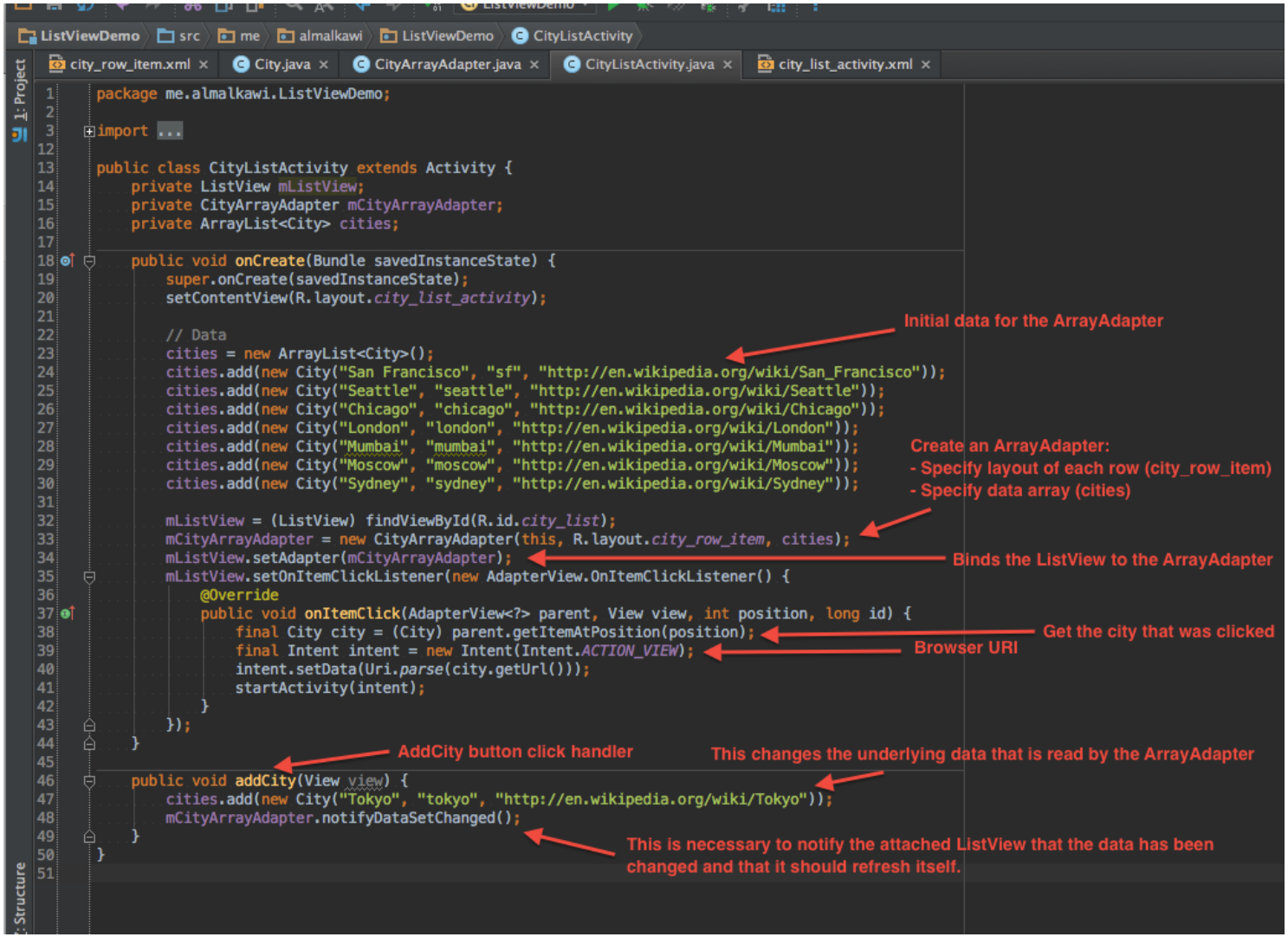
City Data Model (City.java)



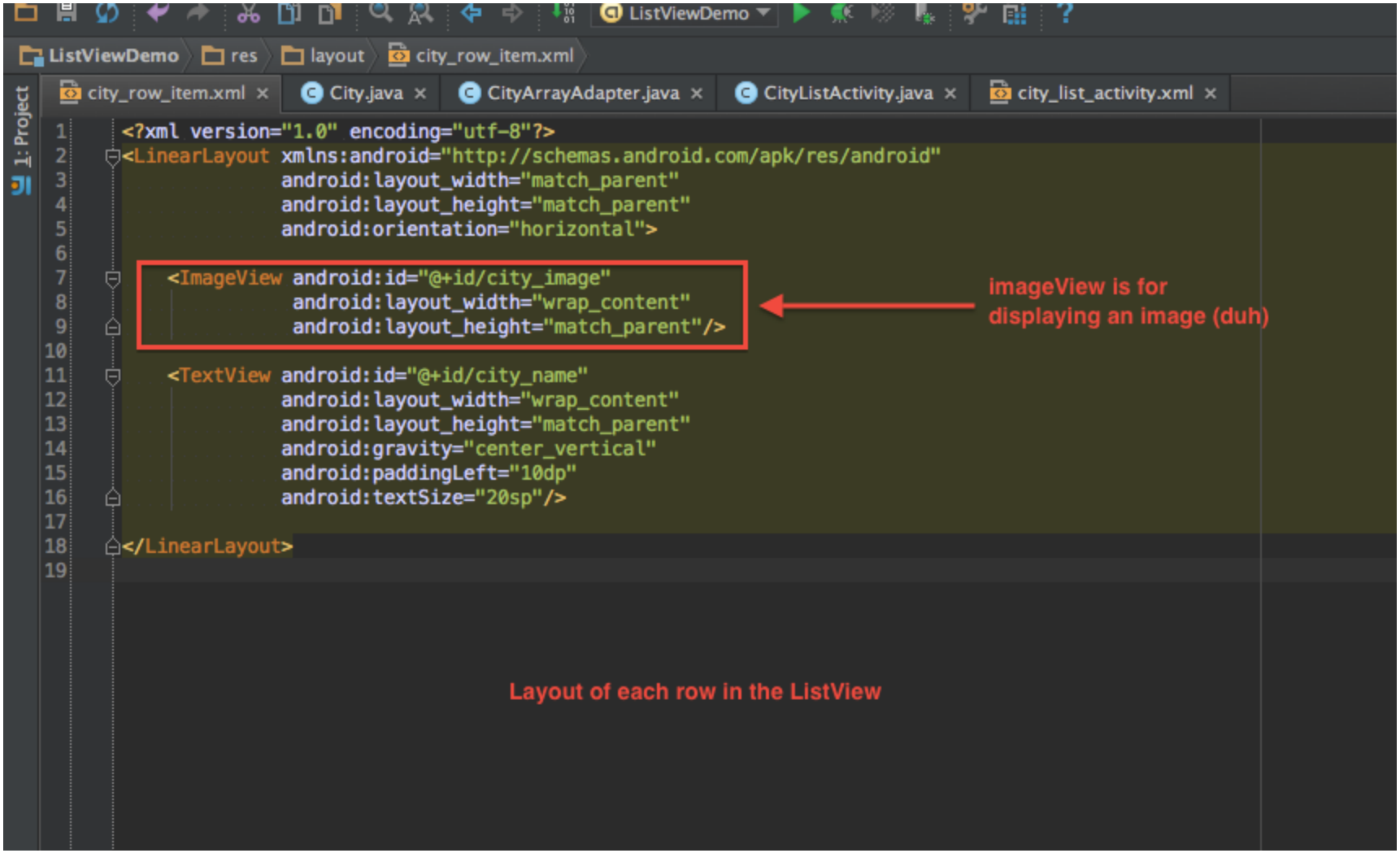
ListView in XML (city_list_activity.xml)



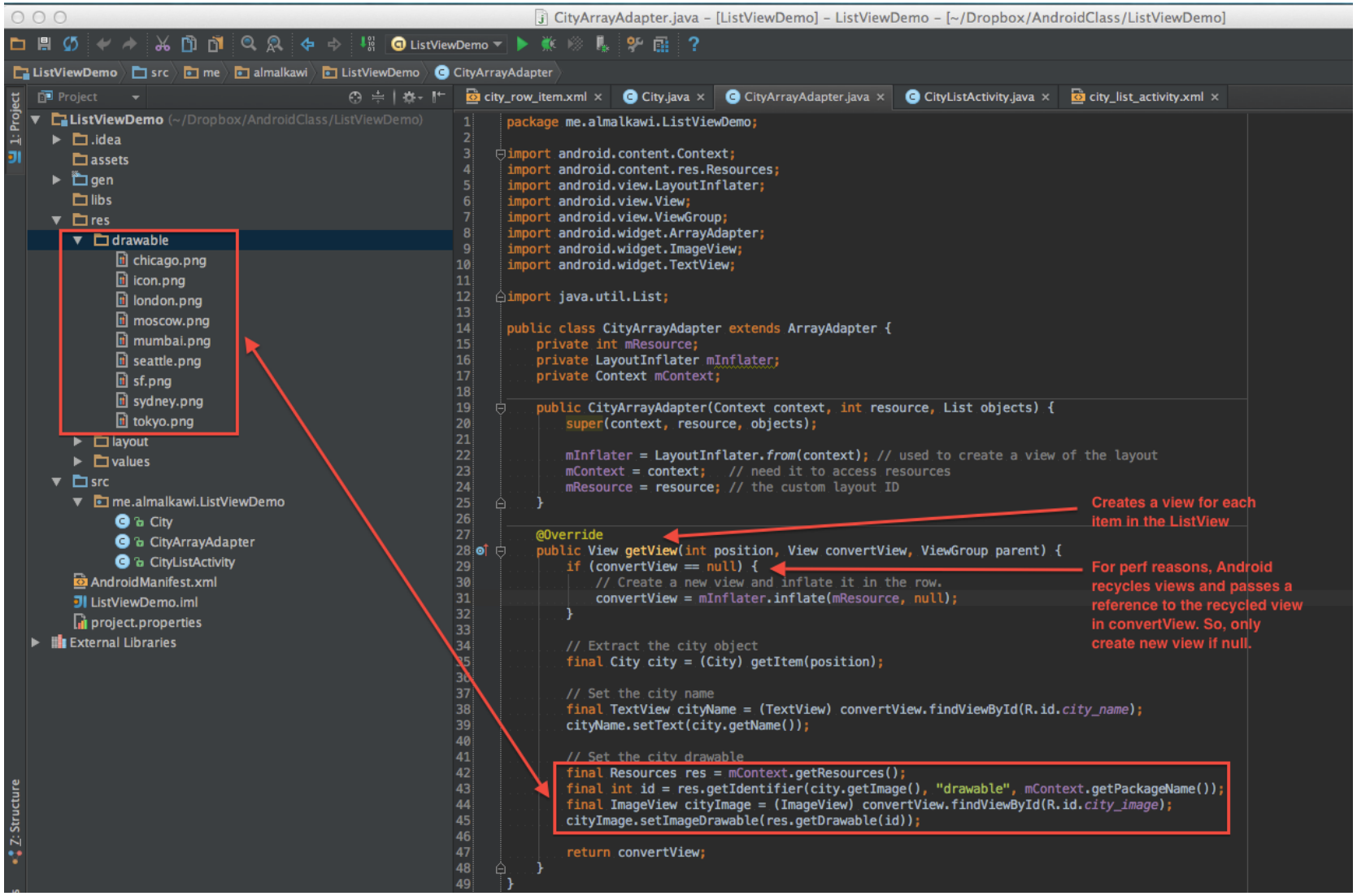
City ListView Activity (CityListActivity.java)



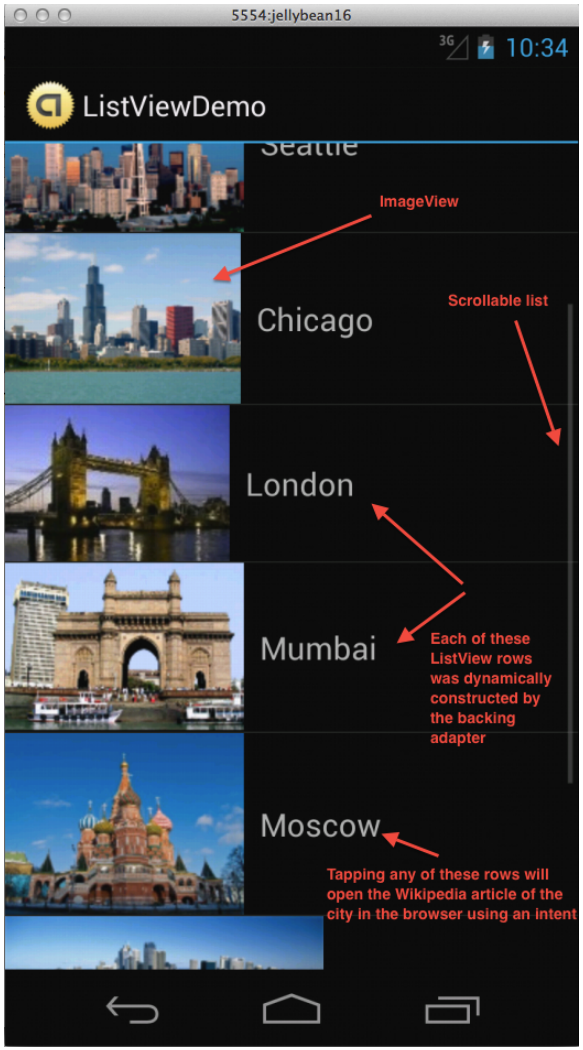
Layout of ListView rows (city_row_item.xml)



City ArrayAdapter (CityArrayAdapter.java)



ListView Demo in emulator



Hello Android!

Lab exercise

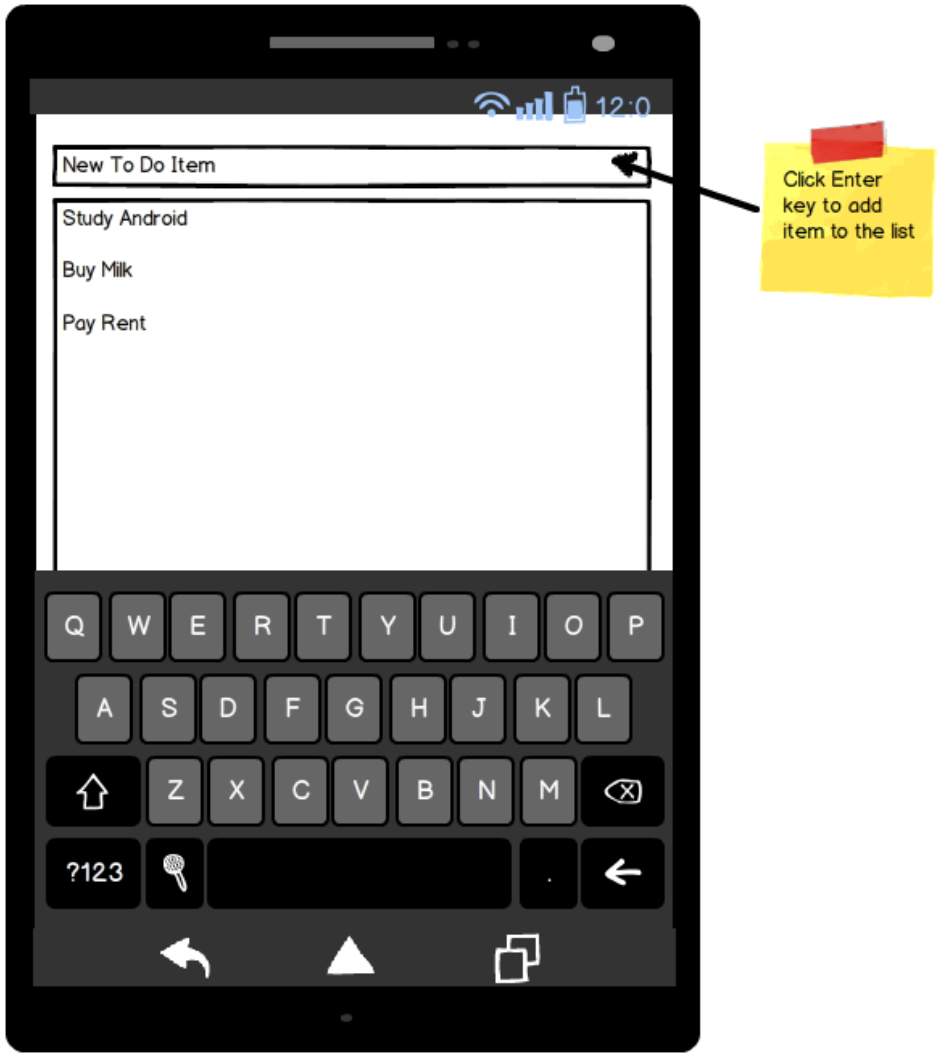
TO-DO List

- Create a TO-DO list app.

- Concepts to incorporate:
  - Activity
  - EditText and KeyListener
  - ListView
  - ArrayAdapter

- UI Mockup is on next slide

TO-DO App UI Mockup



Hello Android!

Dynamic UIs using Fragments

Fragments

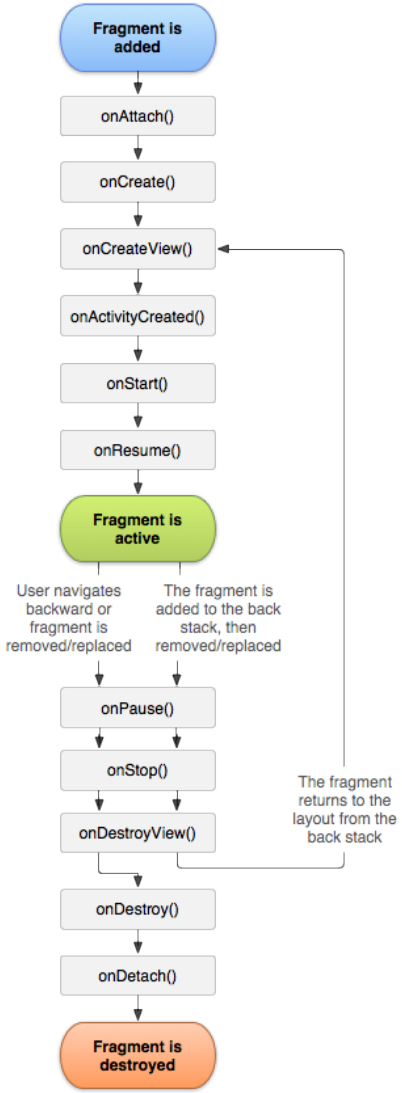
Fragments are:

- Reusable, modular UI components.
- For creating dynamic and multi-pane interfaces.
- Can be swapped into/out of activities based on things like screen size.

Each instance of a Fragment class must be associated with a parent Activity.

If Activity is a controller, then Fragments are mini-controllers.

Fragment Lifecycle



Associating Fragment with Activity

Fragment can be associated with parent activity:

- Using the activity's layout XML file.
- Added dynamically at run-time

Fragments do nt need to be registered in the manifest. (because they can ***not*** exist on their own. They are embedded in other activities).

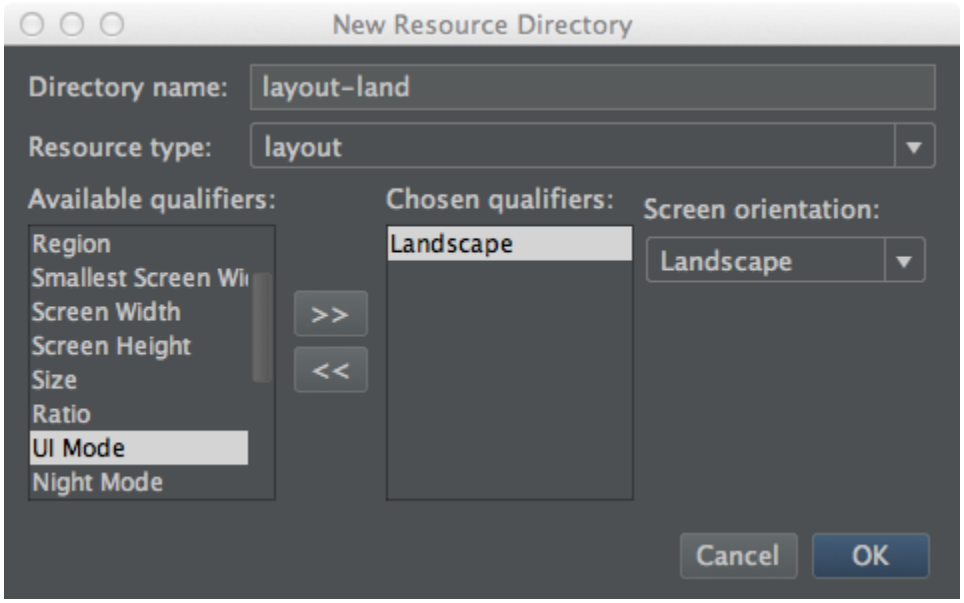
Demo: Fragments using XML

Source: https://github.com/almalkawi/AndroidClass/tree/master/FragmentsXML

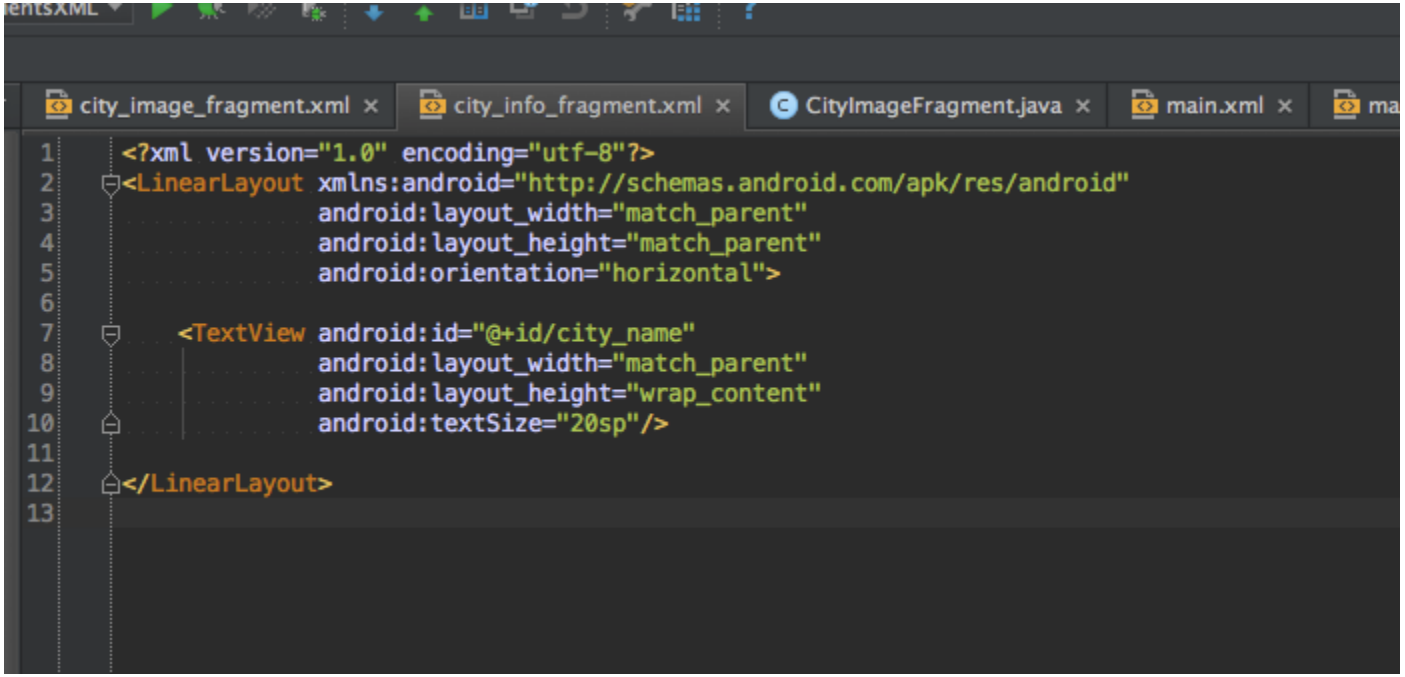
Demo demonstrates handling different screen sizes/orientation.

Run project in emulator and observe lifecycle of Activity and Fragment through Toast messages.

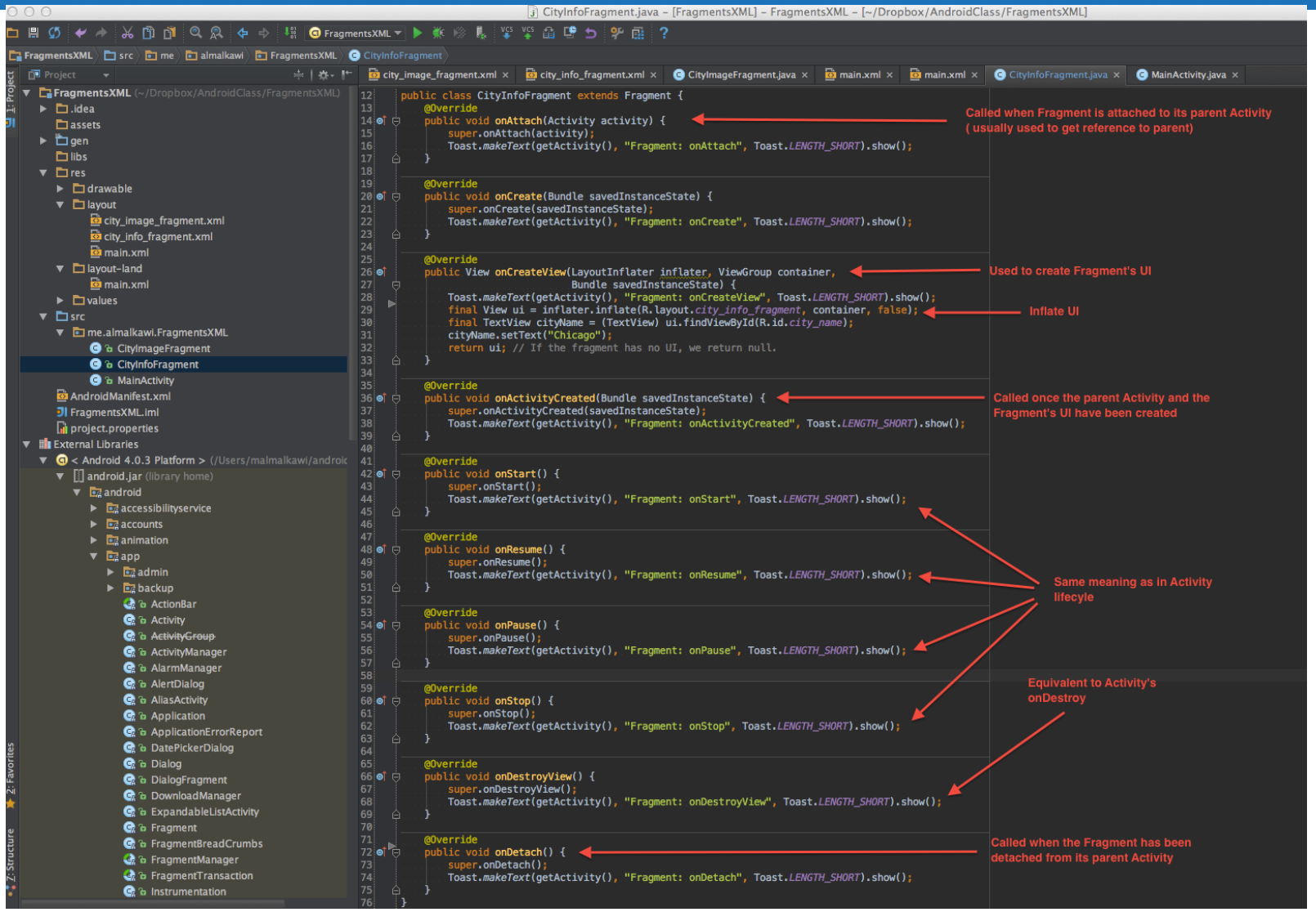
Create landscape layout



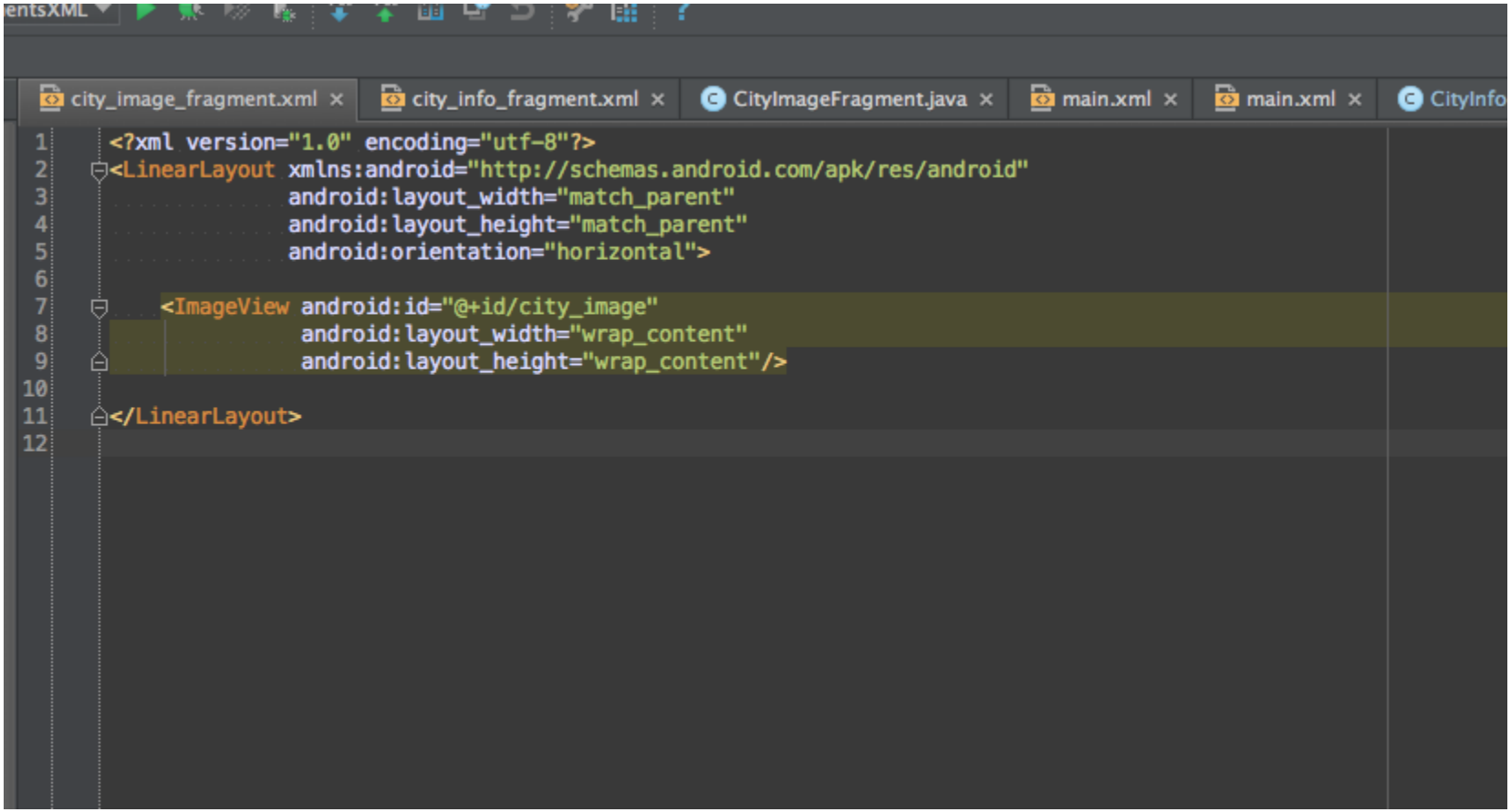
CityInfo Fragment UI (city_info_fragment.xml)



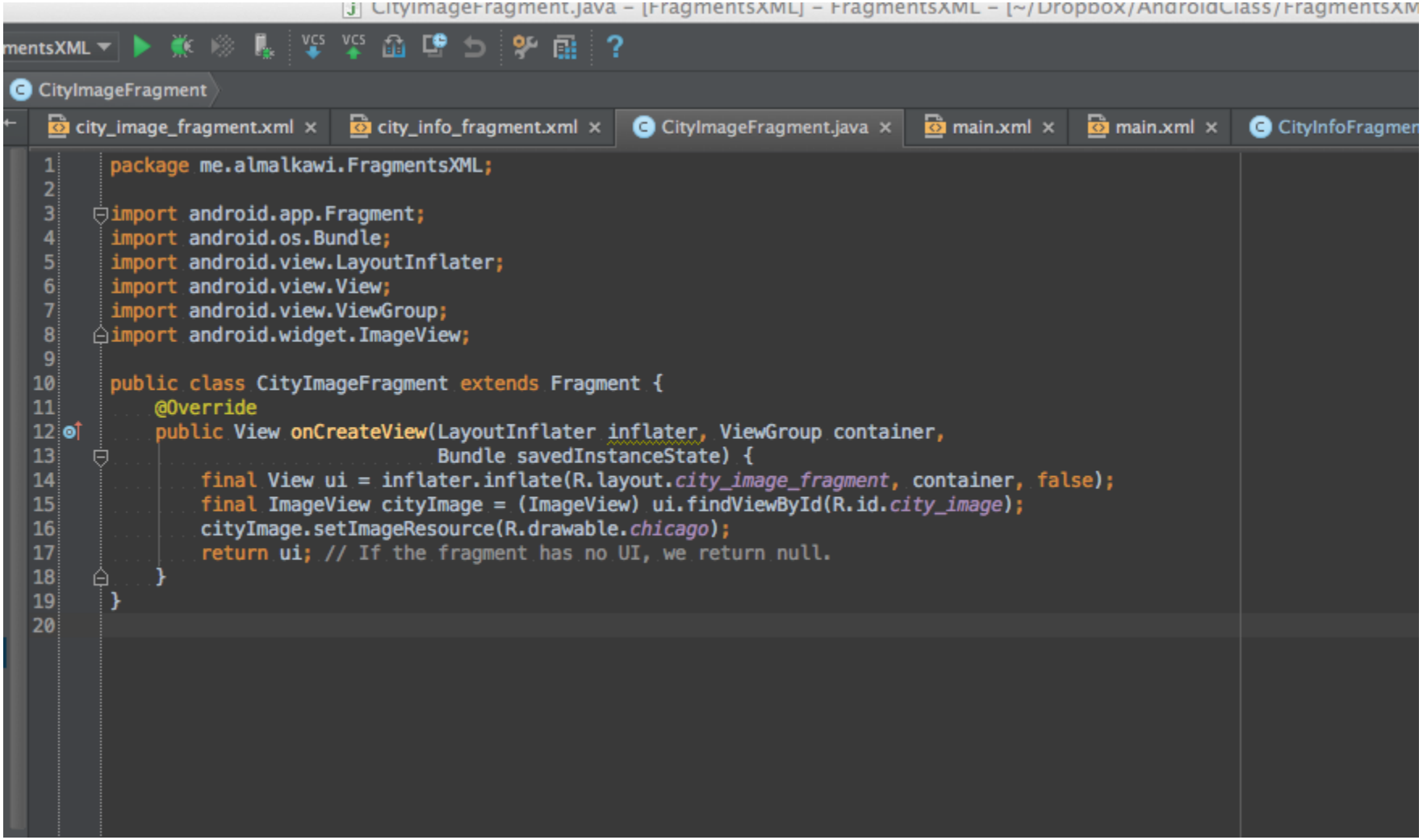
CityInfo Fragment (CityInfoFragment.java)



CityImage Fragment UI (city_image_fragment.xml)



CityImage Fragment (CityImageFragment.java)



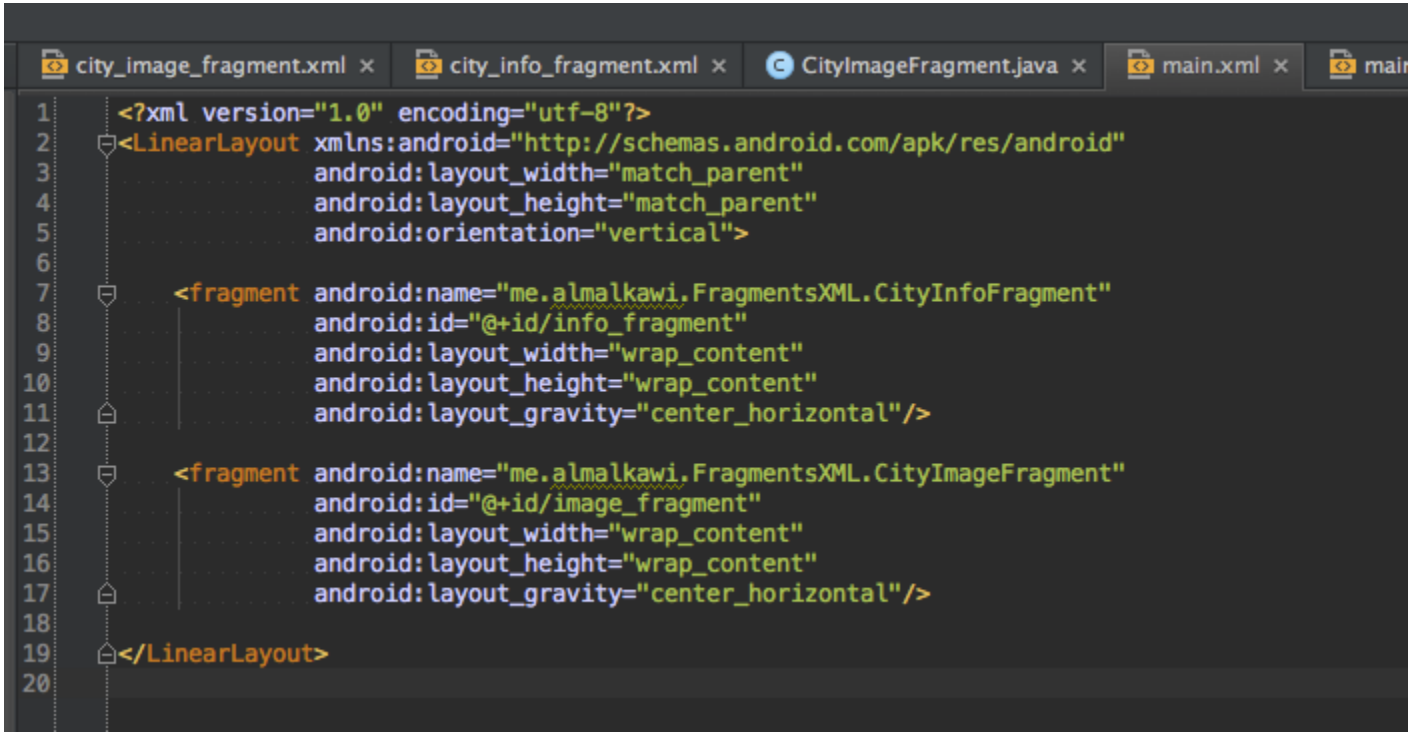
MainActivity.java

MainActivity

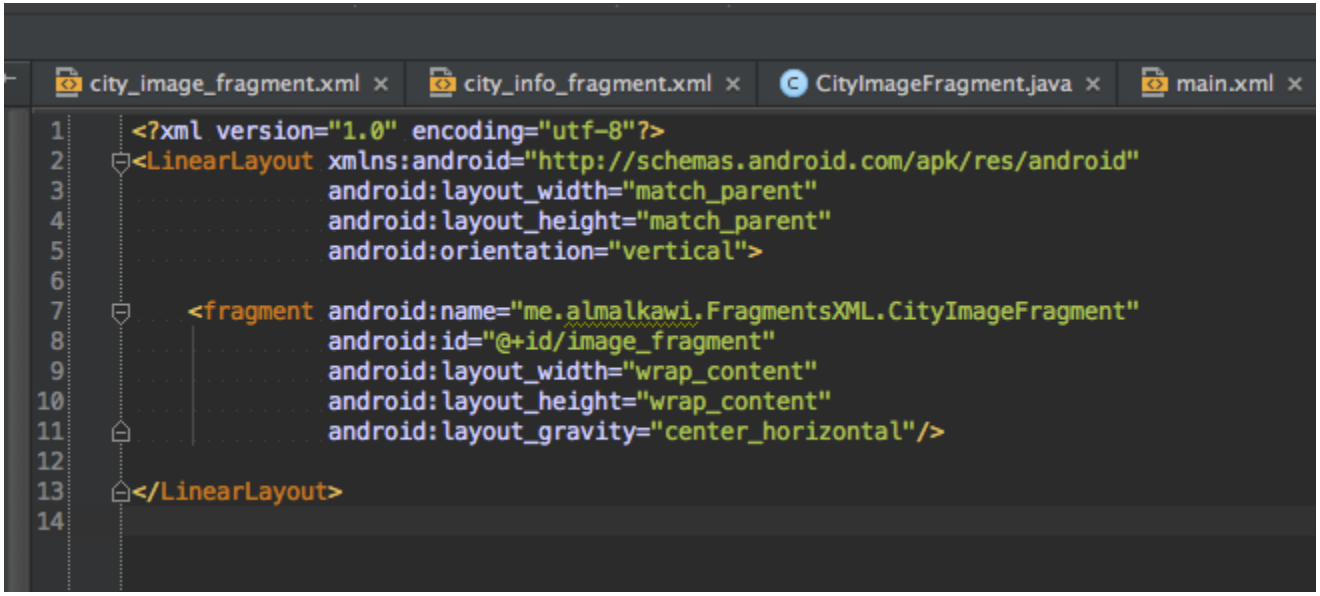
city_image_fragment.xml × | city_info_fragment.xml × | CityImageFragment.java × | main.xml ×

```java
package me.almalkawi.FragmentsXML;

import android.app.Activity;
import android.os.Bundle;
import android.widget.Toast;

public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Toast.makeText(this, "Activity: onCreate", Toast.LENGTH_SHORT).show();
        setContentView(R.layout.main);
    }

    @Override
    protected void onStart() {
        super.onStart();
        Toast.makeText(this, "Activity: onStart", Toast.LENGTH_SHORT).show();
    }

    @Override
    protected void onRestart() {
        super.onRestart();
        Toast.makeText(this, "Activity: onRestart", Toast.LENGTH_SHORT).show();
    }

    @Override
    protected void onResume() {
        super.onResume();
        Toast.makeText(this, "Activity: onResume", Toast.LENGTH_SHORT).show();
    }

    @Override
    protected void onPause() {
        super.onPause();
        Toast.makeText(this, "Activity: onPause", Toast.LENGTH_SHORT).show();
    }

    @Override
    protected void onStop() {
        super.onStop();
        Toast.makeText(this, "Activity: onStop", Toast.LENGTH_SHORT).show();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Toast.makeText(this, "Activity: onDestroy", Toast.LENGTH_SHORT).show();
    }
}
```

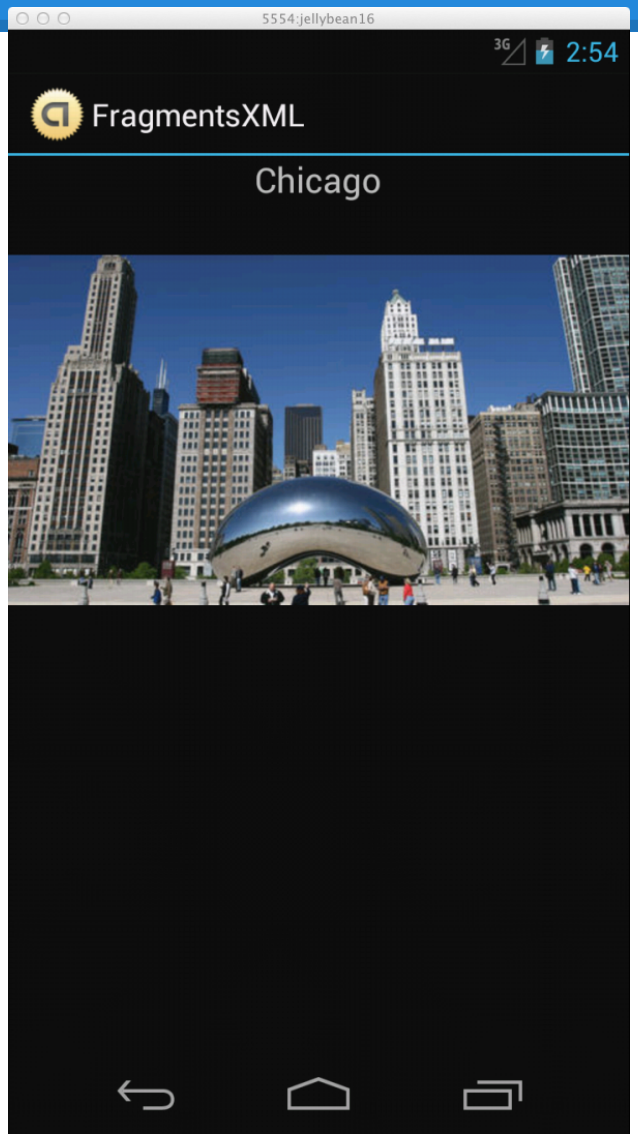
res/layout/main.xml



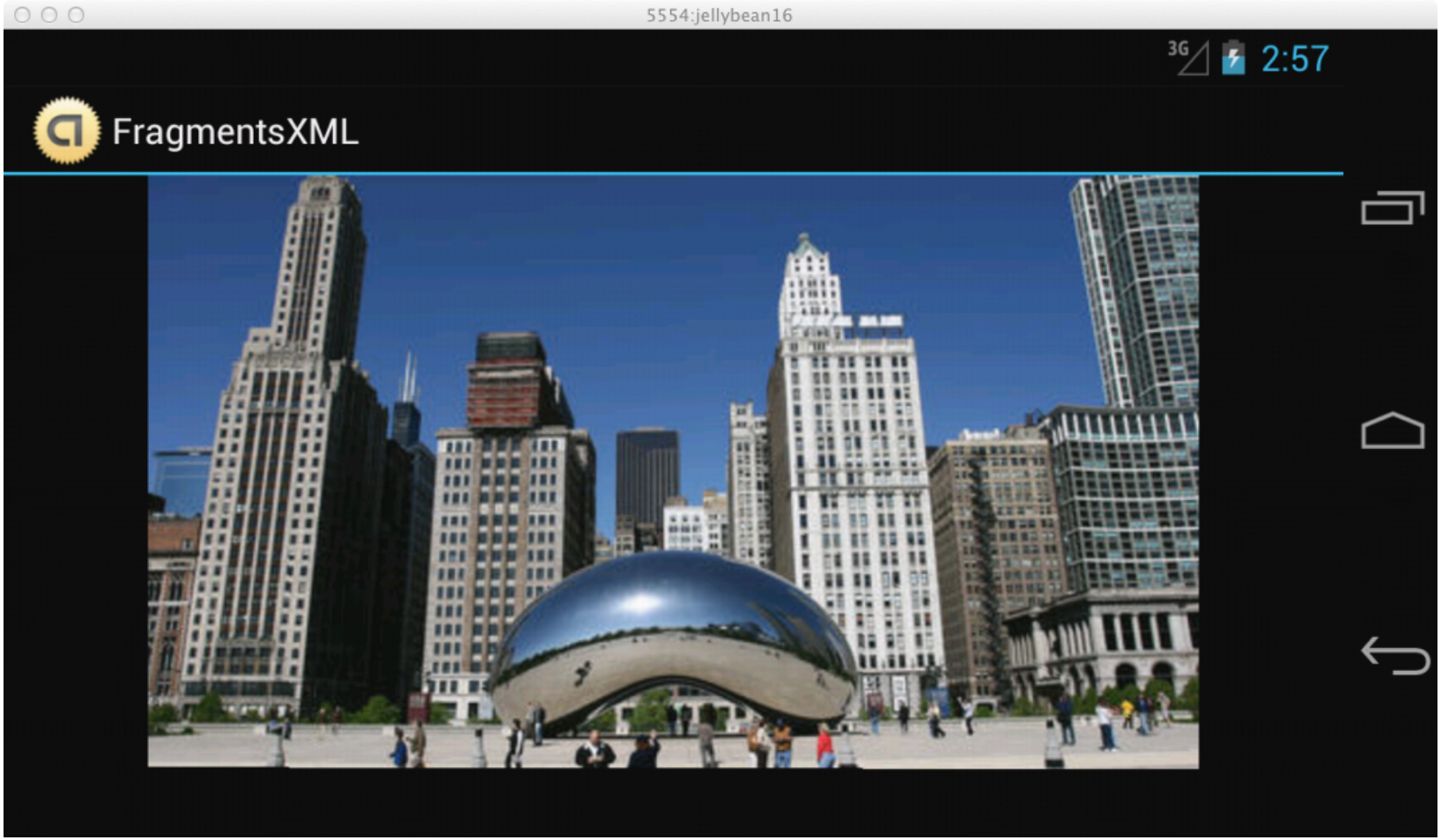
res/layout-land/main.xml



Running project (Portrait)



Running project (Landscape)



Demo: Adding/Removing Fragments at runtime

Source: https://github.com/almalkawi/AndroidClass/tree/master/DynamicFragments

These are the same as in previous demo:

CityInfoFragment.java

city_image_fragment.xml

CityImageFragment.java

city_image_fragment.xml

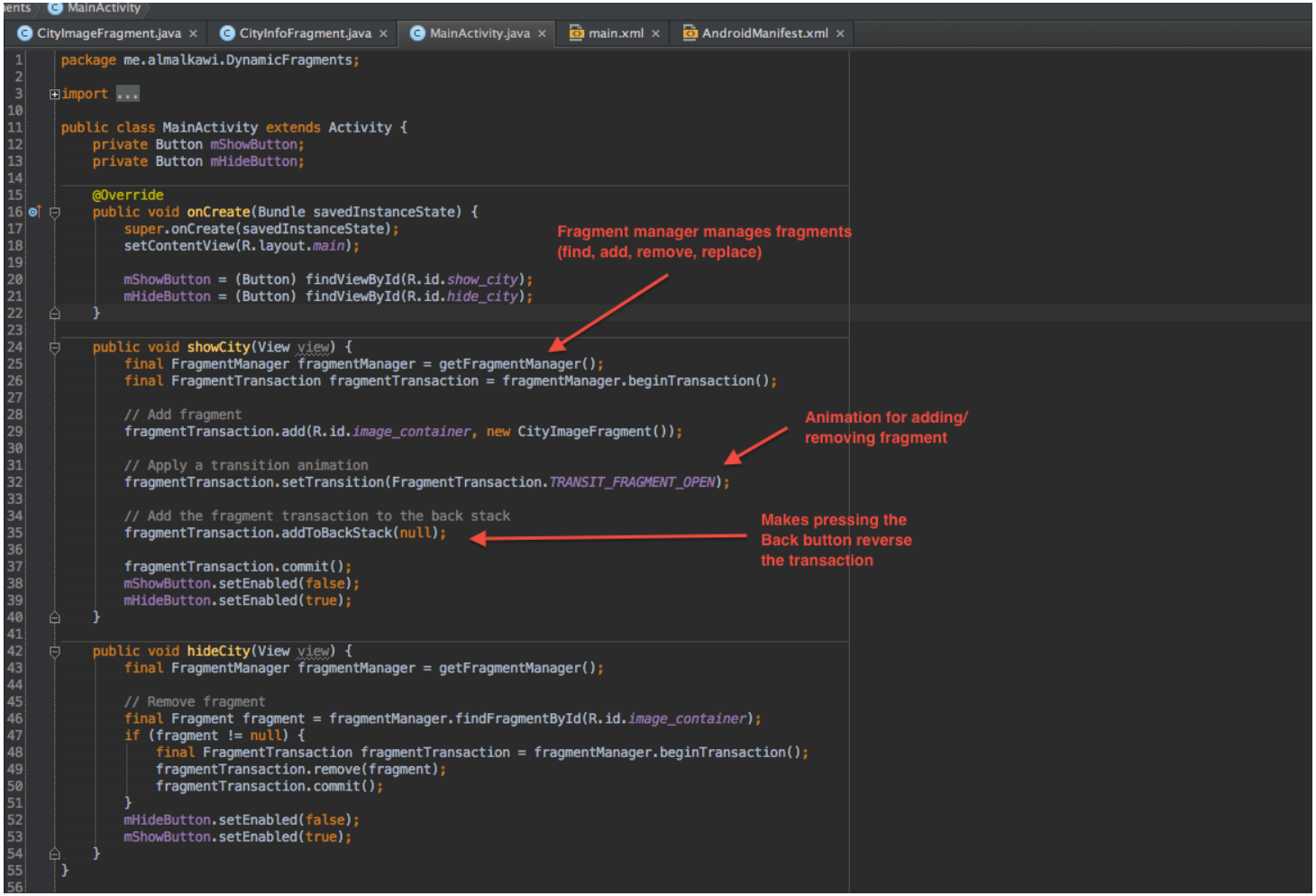
main.xml

CityImageFragment.java | CityInfoFragment.java | MainActivity.java | main.xml | AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical">

    <fragment android:name="me.almalkawi.DynamicFragments.CityInfoFragment"
              android:id="@+id/image_fragment"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_gravity="center_horizontal"/>

    <LinearLayout android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:orientation="horizontal"
                  android:layout_gravity="center_horizontal">

        <Button android:id="@+id/show_city"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:text="Add fragment"
                android:enabled="true"
                android:onClick="showCity"/>

        <Button android:id="@+id/hide_city"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:text="Del fragment"
                android:enabled="false"
                android:onClick="hideCity"/>

    </LinearLayout>

    <FrameLayout android:id="@+id/image_container"
                 android:layout_width="wrap_content"
                 android:layout_height="wrap_content"
                 android:layout_gravity="center_horizontal"/>

</LinearLayout>
```

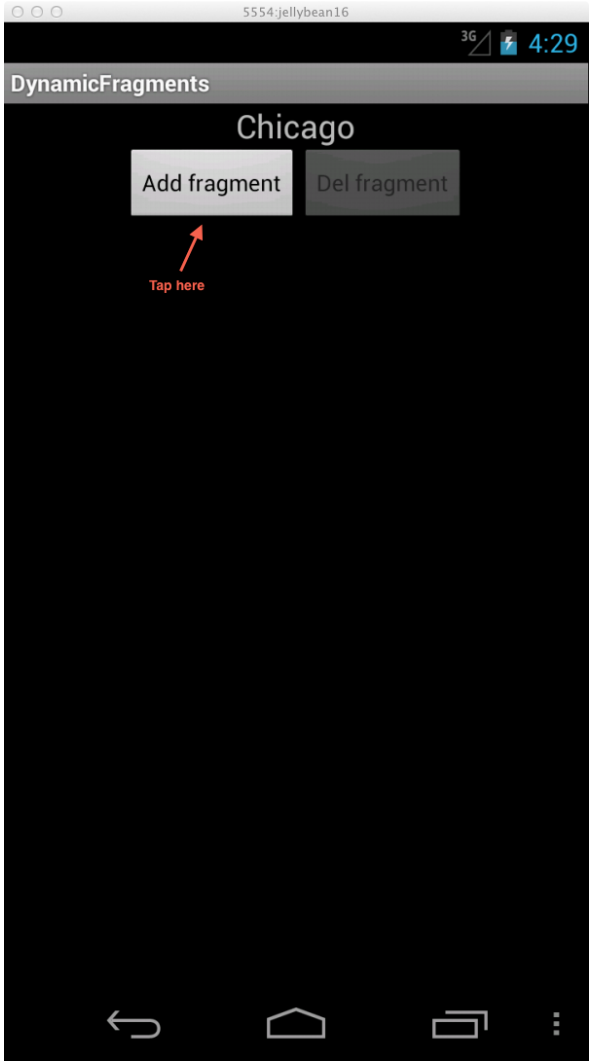
Container in which fragment can be placed at runtime

MainActivity.java

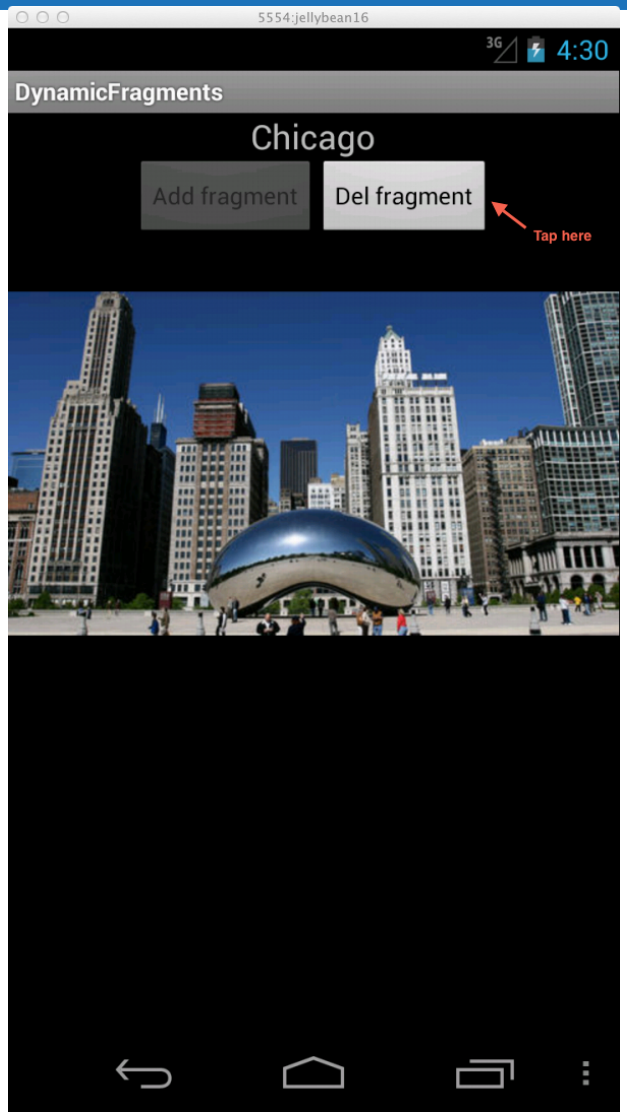



```java
package me.almalkawi.DynamicFragments;

import ...

public class MainActivity extends Activity {
    private Button mShowButton;
    private Button mHideButton;

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

        mShowButton = (Button) findViewById(R.id.show_city);
        mHideButton = (Button) findViewById(R.id.hide_city);
    }

    public void showCity(View view) {
        final FragmentManager fragmentManager = getFragmentManager();
        final FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

        // Add fragment
        fragmentTransaction.add(R.id.image_container, new CityImageFragment());

        // Apply a transition animation
        fragmentTransaction.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);

        // Add the fragment transaction to the back stack
        fragmentTransaction.addToBackStack(null);

        fragmentTransaction.commit();
        mShowButton.setEnabled(false);
        mHideButton.setEnabled(true);
    }

    public void hideCity(View view) {
        final FragmentManager fragmentManager = getFragmentManager();

        // Remove fragment
        final Fragment fragment = fragmentManager.findFragmentById(R.id.image_container);
        if (fragment != null) {
            final FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
            fragmentTransaction.remove(fragment);
            fragmentTransaction.commit();
        }
        mHideButton.setEnabled(false);
        mShowButton.setEnabled(true);
    }
}
```

Running project (initial and after removing fragment)



Running project (after tapping "Add fragment")



Hello Android!

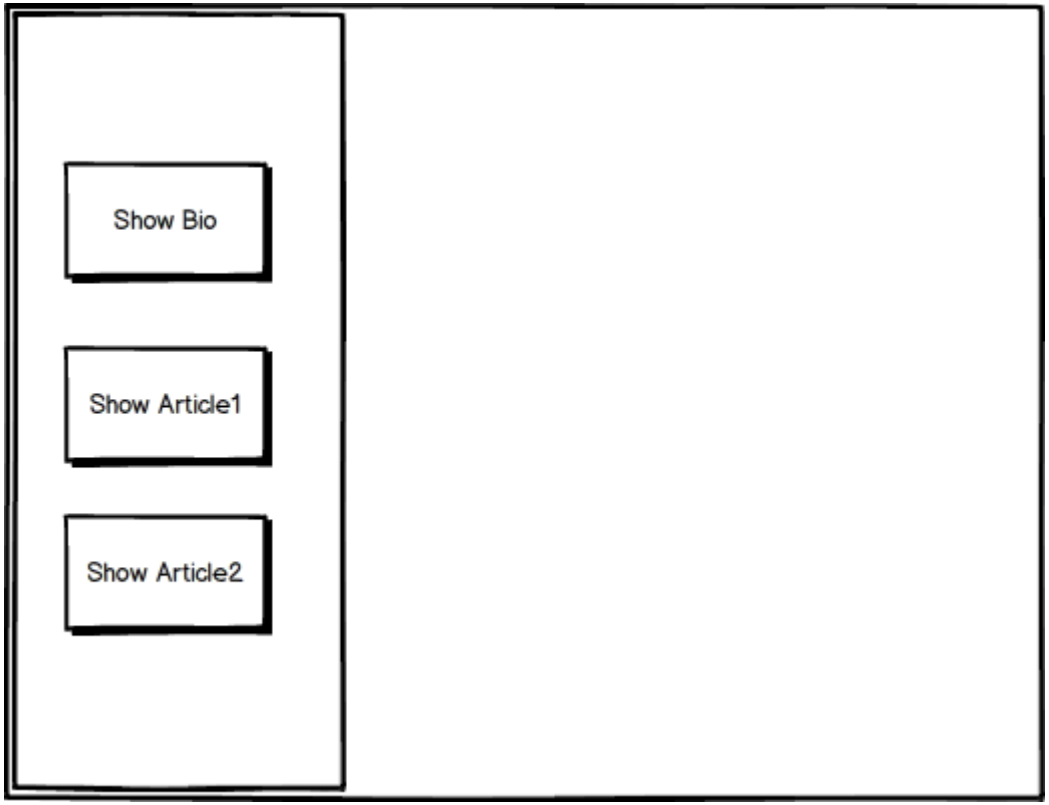
Fragments Exercise

Exercise: Reader App

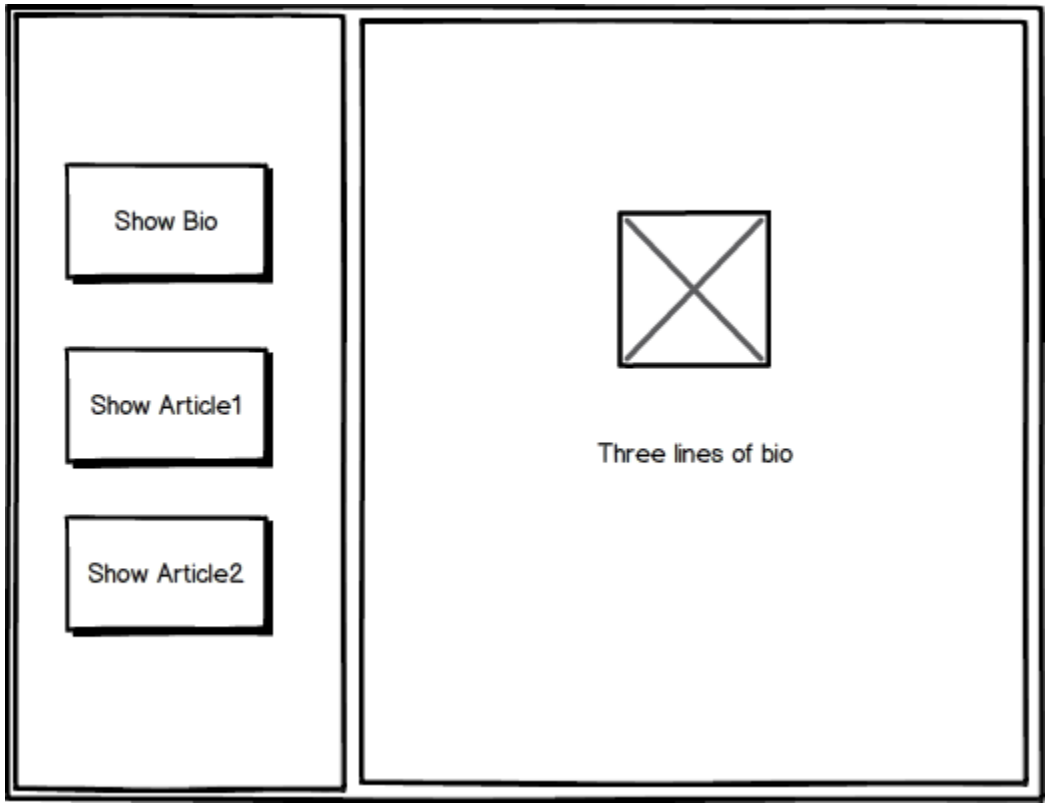
- Use one static Fragment for left pane
- Use one dynamic Fragment for bio
- Use one dynamic Fragment for article

- Mockups are in next slides

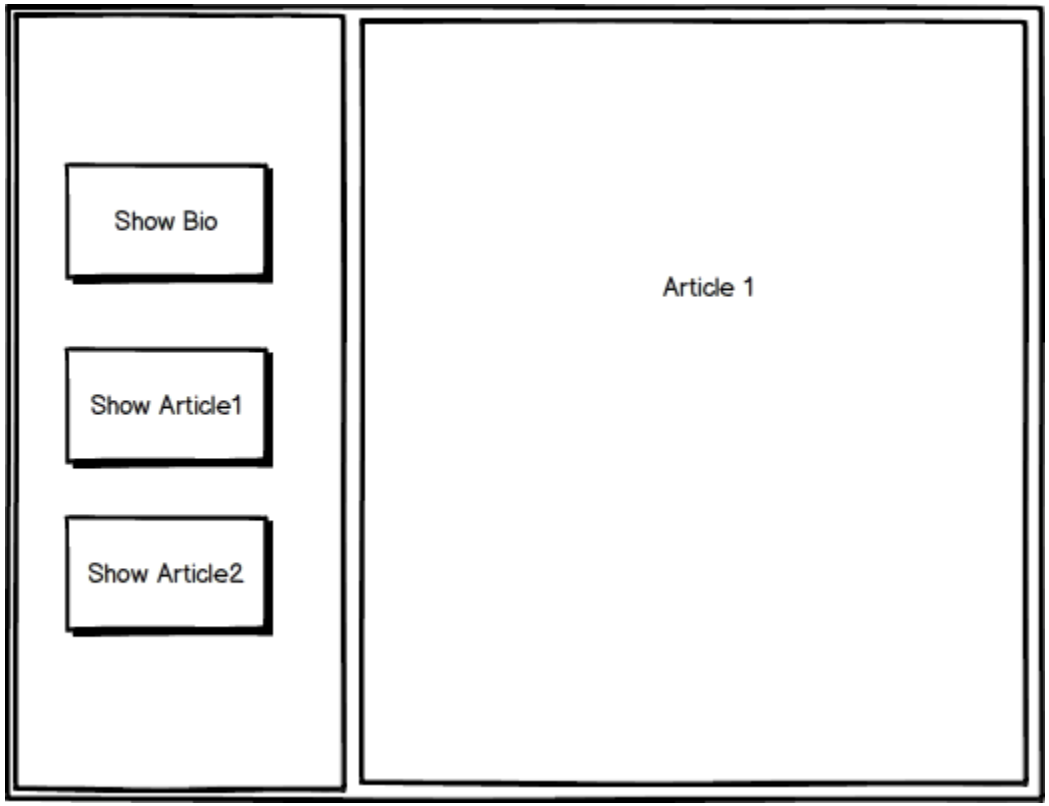
Exercise: Initial State



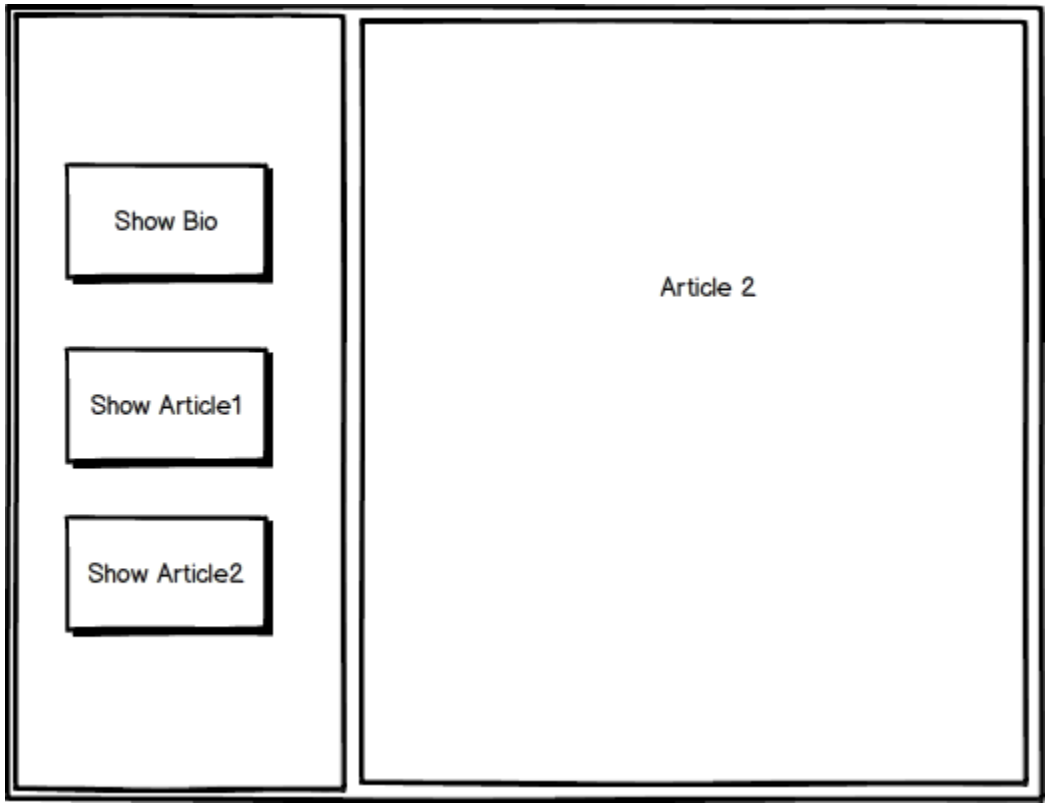
Exercise: Showing Bio Fragment



Exercise: Showing Article1



Exercise: Showing Article2



Hello Android!

Data Persistence

SQLite tools

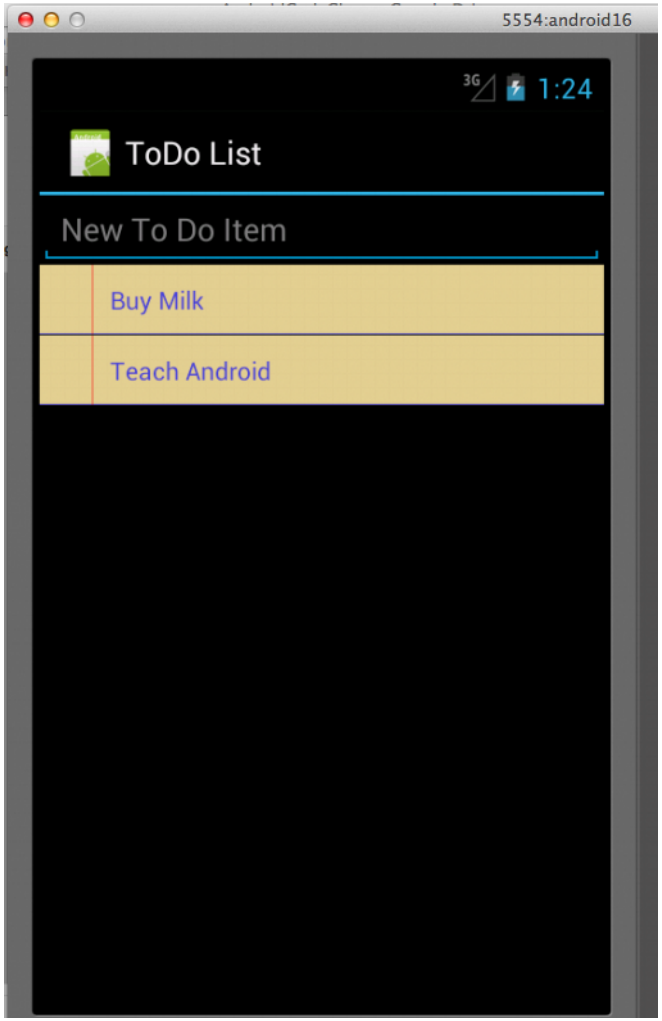
Command line:

```
> sqlite3
```

GUI:

- SQLite Manager Firefox plugin *free*
- Base 2 (http://menial.co.uk/base/) *favorite*
- Navicat for SQLite

Enhanced TODO App



Demo

GitHub Repo: https://github.com/almalkawi/AndroidClass.git

Project: PersistedToDoList

(https://github.com/almalkawi/AndroidClass/tree/master/PersistedToDoList)

Pull DB from phone

- App data: /data/data/com.example.todolist/databases

- Copy DB to /sdcard

- adb pull the db

- Open DB using Base2

Demo

GitHub Repo: https://github.com/almalkawi/AndroidClass.git

Project: ContactsList

(https://github.com/almalkawi/AndroidClass/tree/master/ContactsList)

Hello Android!

Data Persistence Exercise

Exercise

- Add Checkbox next to every item in ToDo App

- Add handler to the checkboxes to remove checked items from the DB and ListView once checkbox is checked.