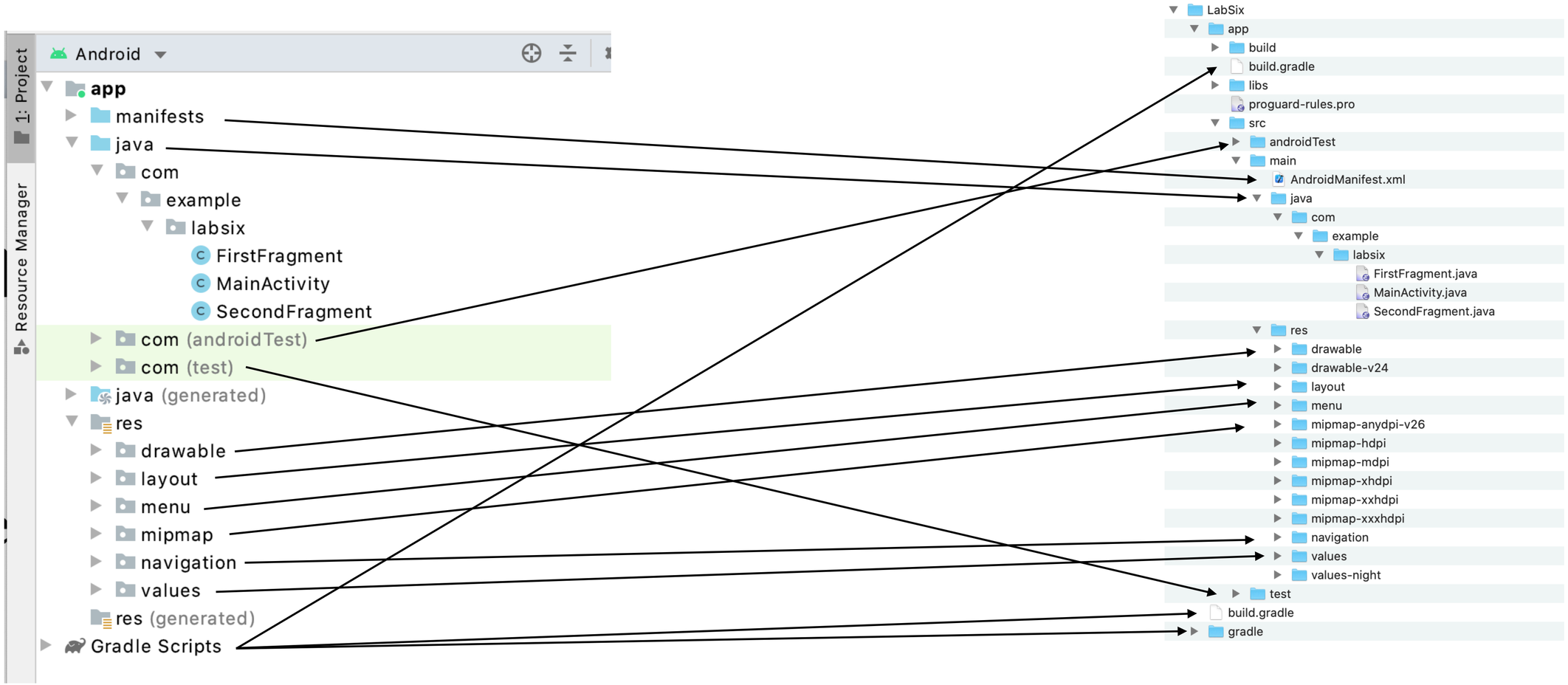
File layout in Android Androiding

cs380
Nov 12

The src/main subdirectory

Almost everything you do is here!

- More importantly, the structure visible in Android Studio echoes this directory (plus some)



More in res

- Layout
  - Design screens in app
    - Almost total separation presentation from preparation
      - In any but very small shops, there a people who never work outside of the res directory and its files — in particular the layout system
        - demo changing background color
- Menu:
  - organizes dropdown and slideout menus
    - demo add menu item to 3 dots
- Navigation
  - In a fragment-based system, (like the basic app) this can define a set of transitions
  - This is not always in use.

Measuring/positioning things

Android

- px (Pixels) -
  - Actual pixels or dots on the screen.
- in (Inches) -
  - Physical size of the screen in inches.
- mm (Millimeters) -
  - Physical size of the screen in millimeters.
- pt (Points)
  - 1/72 of an inch.
- dp (Density-independent Pixels) -
  - An abstract unit that is based on the physical density of the screen. These units are relative to a 160 dpi screen, so one dp is one pixel on a 160 dpi screen. The ratio of dp-to-pixel will change with the screen density, but not necessarily in direct proportion.
    - “dip" and "dp" are same.
- sp (Scale-independent Pixels)
  - Similar to dp unit, but also scaled by the user's font size preference

In many functions, the only units actually available is pt, so if you want others you have to do it yourself

HTML

- cm, mm, in
  - centimeters, millimeters, inches (1in = 96px = 2.54cm)
- px, pt, pc
  - pixels (1px = 1/96th of in), points (1pt = 1/72 of in), picas (1pc = 12 pt = 1/844 of in)
- em
  - Relative to the font-size of the element (2em means 2 times the size of the current font)
- ex
  - Relative to the x-height of the current font (rarely used)
- ch
  - Relative to the width of the "0" (zero)
- rem
  - Relative to font-size of the root element
- vw (vh)
  - Relative to 1% of the width, height of the viewport*
- vmin (vmax)
  - Relative to 1% of viewport's* smaller (larger) dimension
- %
  - Relative to the parent element

Files in Practice

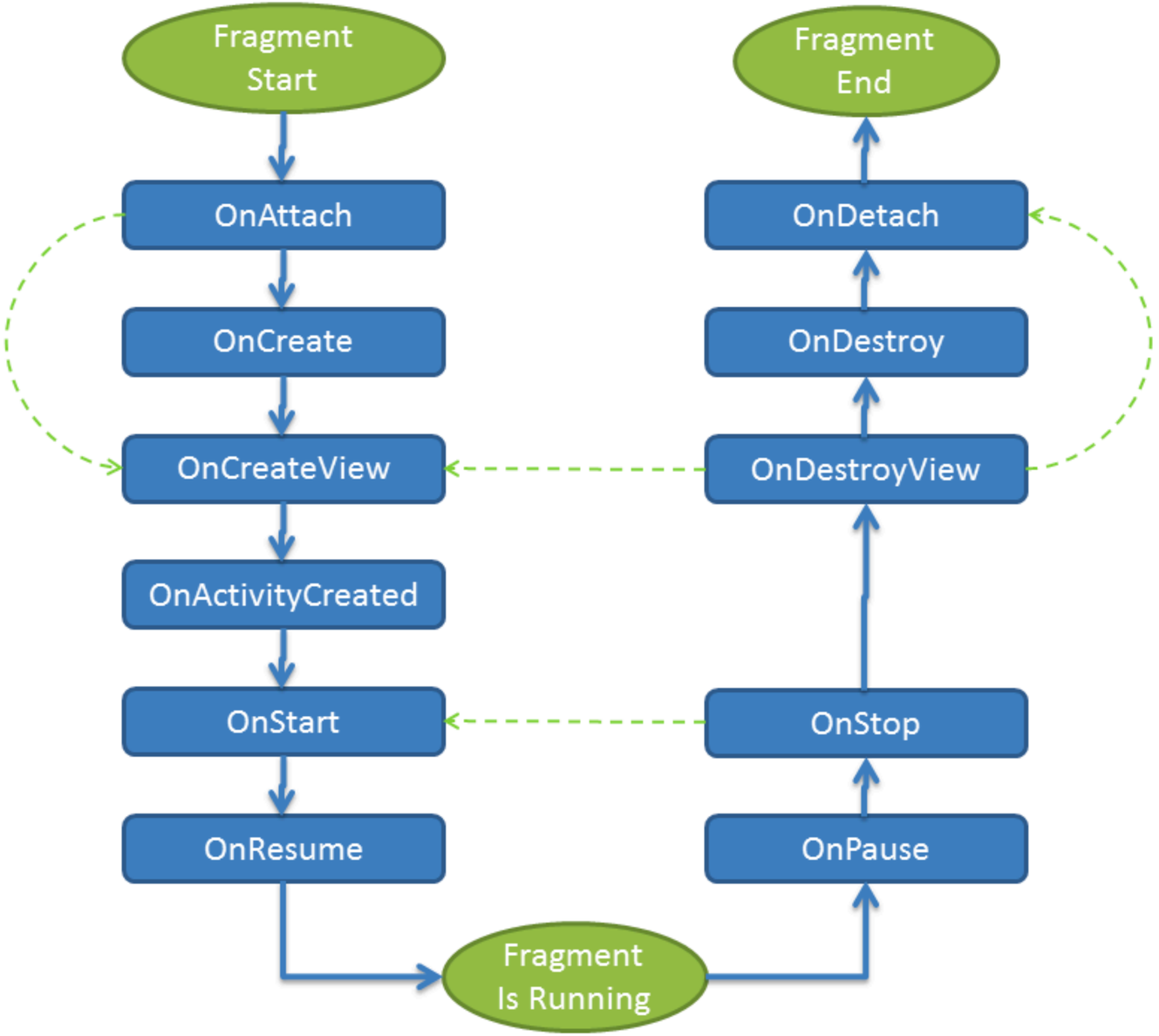
Looking at and mucking with the base app

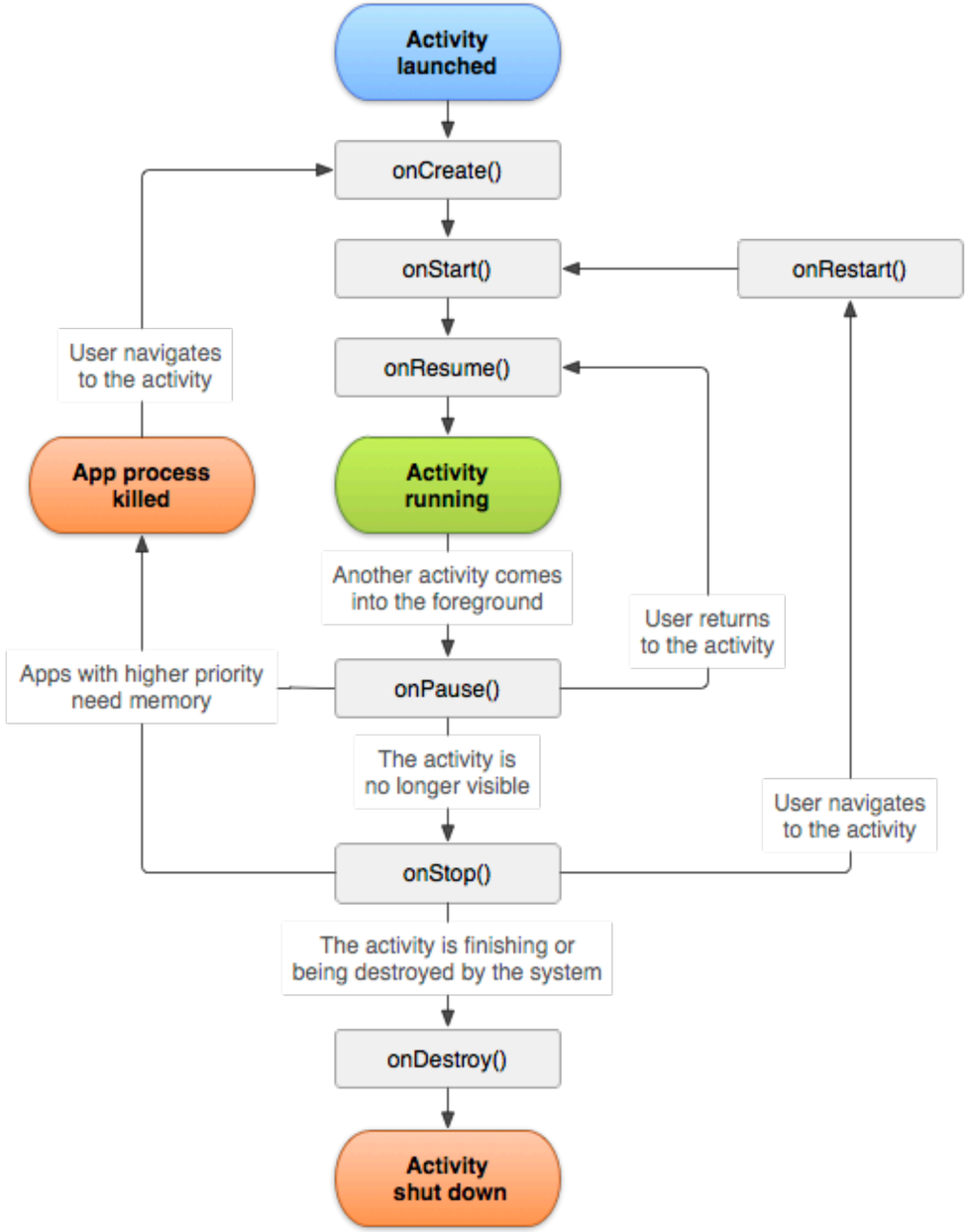
- Changing the text — what happened?
  - layout/fragment_first.xml
    - android:text=....
- Xml gets rendered out into Java code which then shows in app
  - Looking at documentation you can see all of the options for customizing
  - Google expects you will use XML to develop screens, not required.
    - Why?
    - Why Not?
- Change text, change text color, change background color, textSize, all caps in button ...
  - links to values/strings.xml, color.xml
    - Why the indirection?

Backing up — Fragments and Activities

- What is a fragment?
  - “A fragment represents a behavior or a portion of user interface”
  - You can think of a fragment as a modular section of an activity, which has its own lifecycle, receives its own input events,
    - oncreate, onCreateView, onPause
- What is an Activity?
  - Every App has one — any/most have more than one
  - Each activity has its own lifecycle
    - onCreate, onStart, onResume, onPause, onStop, onRestart, onDestroy
    -

Fragment Lifecycles Activity





Getting back to Fragments

- How again does that “rendering out”? when?
- FragmentFour comes in two pieces:
  - FragmentFour.java
  - fragment_four.xml
    - only real linkage between java and xml is in “return inflater.inflate ...”
    - Otherwise names are just names but it is good to use related names
- navigation.xml
  - specifies the transitions between fragments
  - Want fragment 4 to be started from fragment 2, and can be either 3 or 4 from 2
  - From fragment 4 transition to fragment 1

Just because Google wants you to does not mean you have to

- Doing everything in code (almost)!!!

- For transitions between fragments:

```
FragmentManager fragmentManager = this.getSupportFragmentManager();
FragmentTransaction transaction = fragmentManager.beginTransaction();
transaction.replace(R.id.main_actv, new FirstFragment(), null);
transaction.addToBackStack(null);
transaction.commit();
```

  - Instead of “inflater.inflate .... ” in onCreateView just return a LayoutManager of fragments

```
LinearLayout ll = new LinearLayout(getContext());
ll.setId(FirstFragment.FIRST_FRAG_ID);
ll.setOrientation(LinearLayout.VERTICAL);
return ll;
```

  - In either onCreateView or onViewCreated add components to the layout

More doing it Programatically

- Make the “Layout Manager” a LinearLayout (more on layout managers later)

```
LinearLayout ll = new LinearLayout(getContext());
ll.setId(FirstFragment.FIRST_FRAG_ID);
ll.setOrientation(LinearLayout.VERTICAL);
```

- Then create and add items

```
TextView v = new TextView(getContext());
v.setTextColor(Color.WHITE);
v.setTextSize(48);
v.setGravity(Gravity.CENTER);
v.setBackgroundColor(Color.rgb(128, 0, 25));
ll.addView(v, new LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, 0, 2));
```

-