Mobile Application Design and Development

CS 5520

Adrienne Slaughter

Fall 2021

Northeastern University
College of Computer and Information Science

Administrivia

- Sign up for topics:
- **Rendering on Android/how Android renders**
- **Using Fragments well**
- **Testing**
- **Understanding memory usage**
- **Power-aware Android apps**
- **Improving app performance with app profilers**
- **Keeping sensitive data private**
- **Activity recognition**
- **Jetpack Compose**

Android Application Architecture

- **Activity Manager** and **Fragment Manager:** Contain the UI; controls the lifecycle of Activities and Fragments
- **Views:** Constructs the UI (within Activities and Fragments)
- **Notification Manager:** Provides a consistent way to notify the user
- **Content Provider:** Allows your application to share data
- **Resource Manager:** Externalizes non-code resources
- **Intents:** Enables transferring data between applications and components

A1: Recap

- Android Studio: Layout UI AND via XML
- Layouts: Linear, Relative, Constraint
- String resources

Today's Topics

- Layouts
- Views and compound views
- Activity Lifecycle

What is Android?

Android basics: What is it?

- Open source OS and development platform
  - In theory, you can change anything
  - In practice….
- Hardware reference design
- Linux OS kernel
- Open-source libraries for app development

  E.g., SQLite, Webkit, OpenGL, media manager

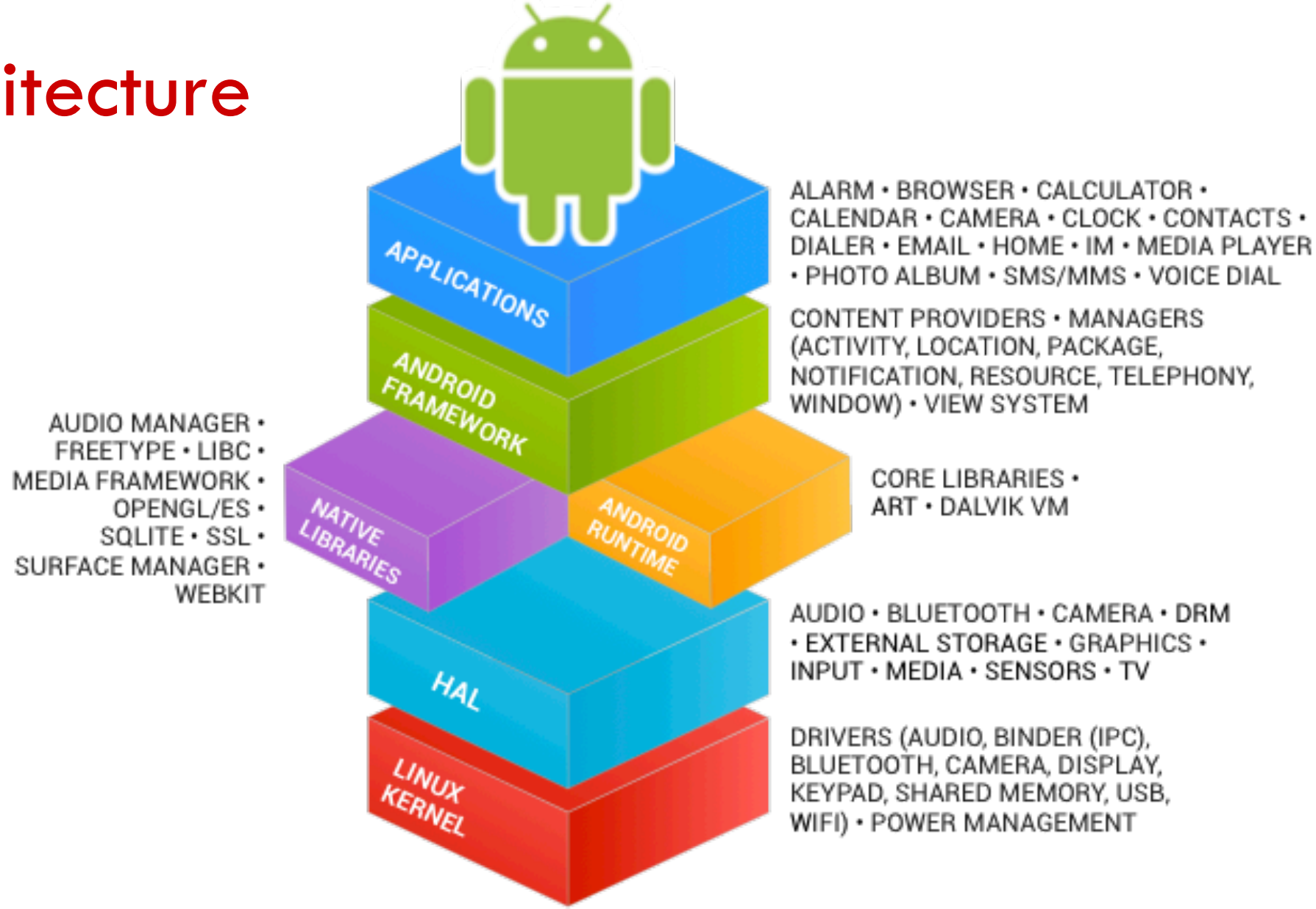
Android basics: what is it?

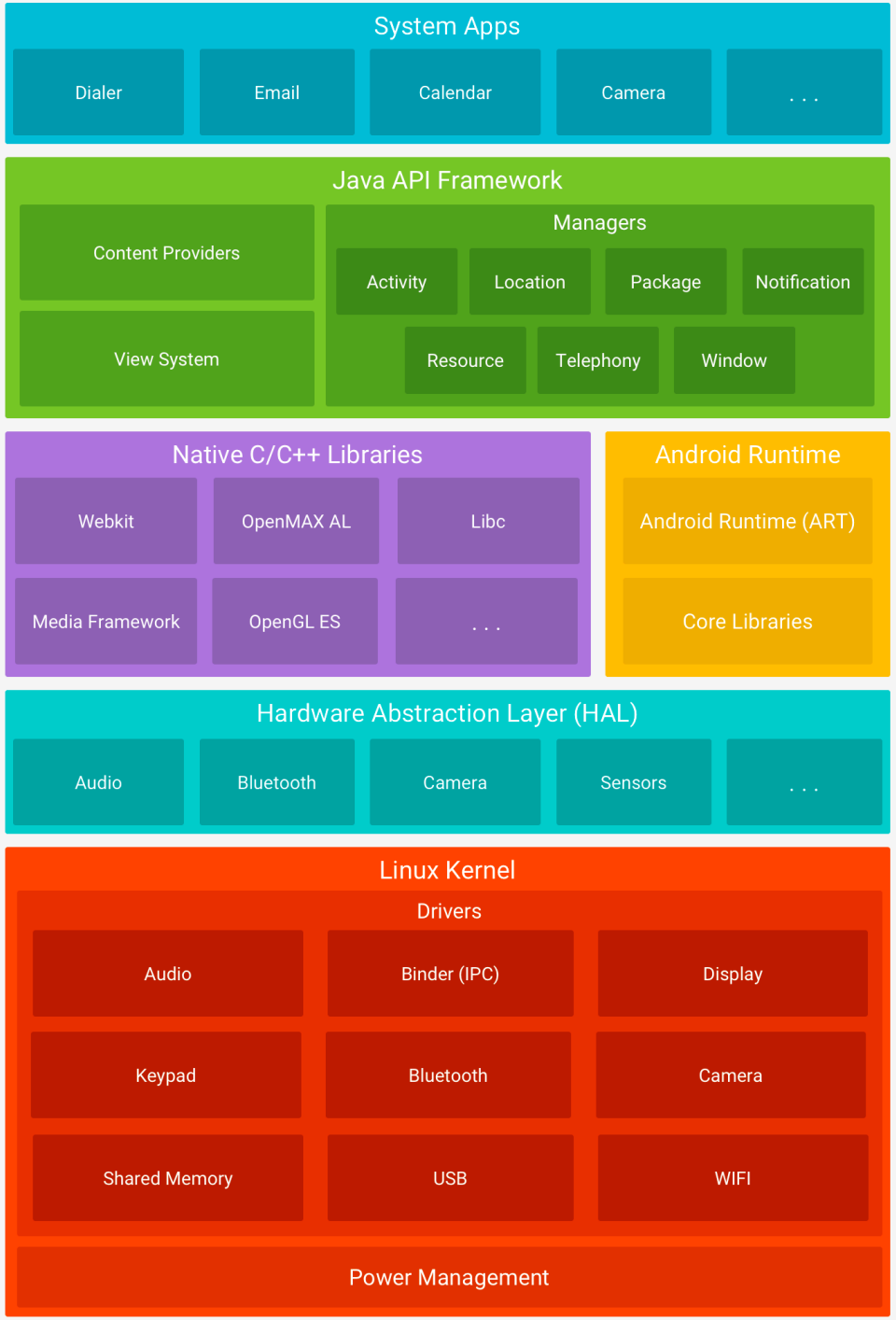
- Dalvik virtual machine (new: ART)
  - Not Java ME
  - Use Java, but compiled to bytecode for small and efficient and multiple VMs
- Application framework
  - Window manager, location manager, telephony, sensors
- UI framework to run and launch apps
- Preinstalled apps

Android basics: what is it?

- SDK and tools
- Wild west of app stores: The Play Store
- Extensions for new tablets
- Extensions for Google Glass
- Extensions for Android Wear, TV, etc.

System Architecture




System Apps
Dialer
Email
Calendar
Camera
. . .
Java API Framework
Content Providers
View System
Managers
Activity
Location
Package
Notification
Resource
Telephony
Window
Native C/C++ Libraries
Webkit
OpenMAX AL
Libc
Media Framework
OpenGL ES
. . .
Android Runtime
Android Runtime (ART)
Core Libraries
Hardware Abstraction Layer (HAL)
Audio
Bluetooth
Camera
Sensors
. . .
Linux Kernel
Drivers
Audio
Binder (IPC)
Display
Keypad
Bluetooth
Camera
Shared Memory
USB
WIFI
Power Management

Notables

- "Open philosophy"
- Application framework that encourages reuse of application components
- Access to (much) hardware (sometimes even without bugs!)
- Release and fix mentality (or it seems like it, sometimes)
- Great standard apps: Google Maps, location services, quick search box
- Background services

Notables

- SQLite (with app data sandboxed)
- Shared data and interprocess communication
  - Notifications (via UI)
  - Intents
  - Content providers (managed access to app private data)
- All apps created equal
- Widgets and livewallpaper and livefolders

What’s different about designing and developing for **mobile** versus **desktop/web**?

Designing and Developing for mobile requires

a new way of thinking.

New way of thinking #1

- Expect…
  - Limited processing power
  - Limited RAM
  - Limited permanent storage capacity
  - Small screen and low resolution
  - High cost of data transfer
  - Slow data transfer rates with high latency
  - Unreliable data connections
- Moore's law less impactful…

- Tip: design and test for older phone/SDK

New way of thinking #2

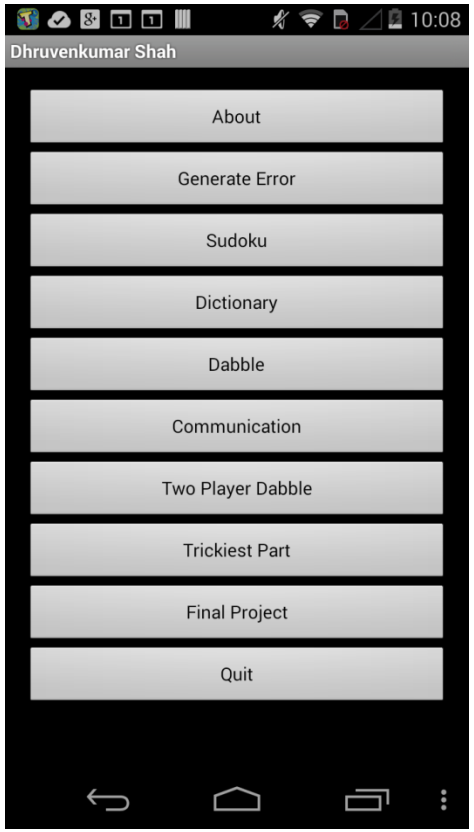
- Application framework that encourages reuse of application components
- "Screens" are Activities that are chained with lightweight exchange of data between them
- OS can handle stack of Activities if you want it to (e.g., back functionality)

New way of thinking #3

- OS Manages process lifetime **(app assassin)**
  - App responsiveness
  - Setting priority to interaction
- You MUST
  - Ensure that your app is ready for swift death
  - Yet, it must remain responsive and/or restart in the background
  - Must come to the foreground quickly

New way of thinking: example

What happens when this screen is rotated?



Types of applications

- Foreground
  - Useful when being used
  - Suspended otherwise
- Background

  Apart from when being configured, spends most of lifetime hidden (e.g., call screening app)
- Intermittent

  Some interaction but mostly in the background (e.g., media player)
- Widget

  Home screen status update

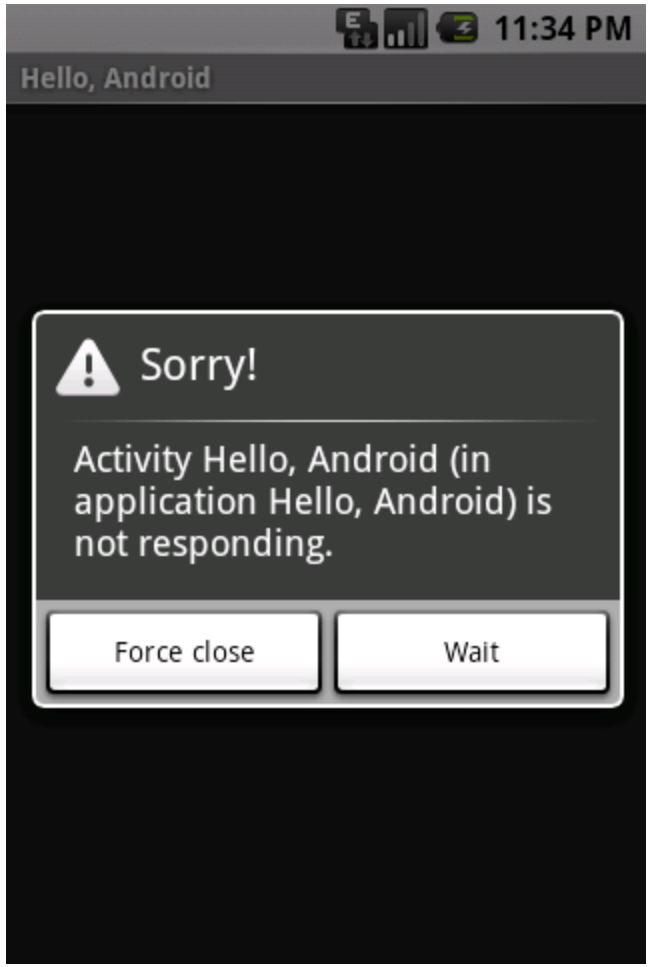
What makes a mobile app *good*?

Good app behavior

- Is well behaved
- Switches seamlessly from background to foreground
- Is polite (e.g., doesn't steal focus)
- Presents a consistent user interface
- Is responsive
- Uses minimal resources
- Uses minimal permissions

Behavior police: process assassin

- Two conditions monitored
  - Must respond to any user action (e.g., key press) within 5s
  - A BroadcastReceiver must return from its OnReceive handler within 10s
- Worst case, not goal! Users notice .5s



What does a mobile app consist of?

What is an application?

Package: an archive file (.apk)

- edu.neu.numad.killerapp
  - edu.neu.numad.killerapp.uttt
  - edu.neu.numad.killerapp.scroggle
- All code in single .apk file one “application”

What is an Application?

- Some code
- A manifest file

Application Components

- Activities
  - Fragments
- Services
- Content Providers
- Broadcast receivers

(See AndroidStudio)

man·i·fest[2]

/ˈmanəˌfest/

noun

noun: **manifest**; plural noun: **manifests**

1. a document giving comprehensive details of a ship and its cargo and other contents, passengers, and crew for the use of customs officers.
   - a list of passengers or cargo in an aircraft.
   - a list of the cars forming a freight train.

verb

verb: **manifest**; 3rd person present: **manifests**; past tense: **manifested**; past participle: **manifested**; gerund or present participle: **manifesting**

1. record in a manifest.
   "every passenger is manifested at the point of departure"

It's a metaphor:

- Android OS is the customs officer
- The app is the train
- The stuff ("components") in the app are the train cars

- Ideally, the customs officer can ***combine*** train cars in ***different*** ways to re-use the code you wrote.

Parts of a project: Manifest

AndroidManifest.xml

- Tell system what components exist
- Set intent filters
- Set permissions
- Declare the minimum API Level (now in Gradle)
- Declare hardware & software features *used or required*, such as a camera, Bluetooth services, or a multitouch screen
- Specify link to API libraries other than the Android framework APIs (e.g., Google maps)
- Set icons

A necessary evil: API levels.

Version and API level

Code name	Version	API level
Nougat	7.1	API level 25
Nougat	7.0	API level 24
Marshmallow	6.0	API level 23
Lollipop	5.1	API level 22
Lollipop	5.0	API level 21
KitKat	4.4 - 4.4.4	API level 19
Jelly Bean	4.3.x	API level 18
Jelly Bean	4.2.x	API level 17
Jelly Bean	4.1.x	API level 16
Ice Cream Sandwich	4.0.3 - 4.0.4	API level 15, NDK 8
Ice Cream Sandwich	4.0.1 - 4.0.2	API level 14, NDK 7
Honeycomb	3.2.x	API level 13
Honeycomb	3.1	API level 12, NDK 6
Honeycomb	3.0	API level 11
Gingerbread	2.3.3 - 2.3.7	API level 10
Gingerbread	2.3 - 2.3.2	API level 9, NDK 5
Froyo	2.2.x	API level 8, NDK 4
Eclair	2.1	API level 7, NDK 3
Eclair	2.0.1	API level 6
Eclair	2.0	API level 5

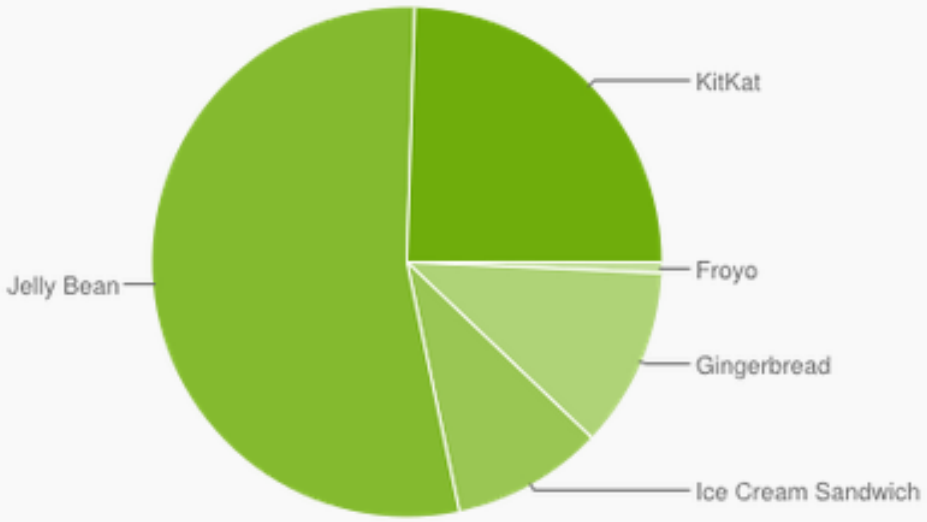
https://source.android.com/source/build-numbers.html

To consider

- Screen size and density
- Input configurations
- Device features
- Platform version…

Sep 2014

Version	Codename	API	Distribution
2.2	Froyo	8	0.7%
2.3.3 - 2.3.7	Gingerbread	10	11.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	9.6%
4.1.x	Jelly Bean	16	25.1%
4.2.x		17	20.7%
4.3		18	8.0%
4.4	KitKat	19	24.5%



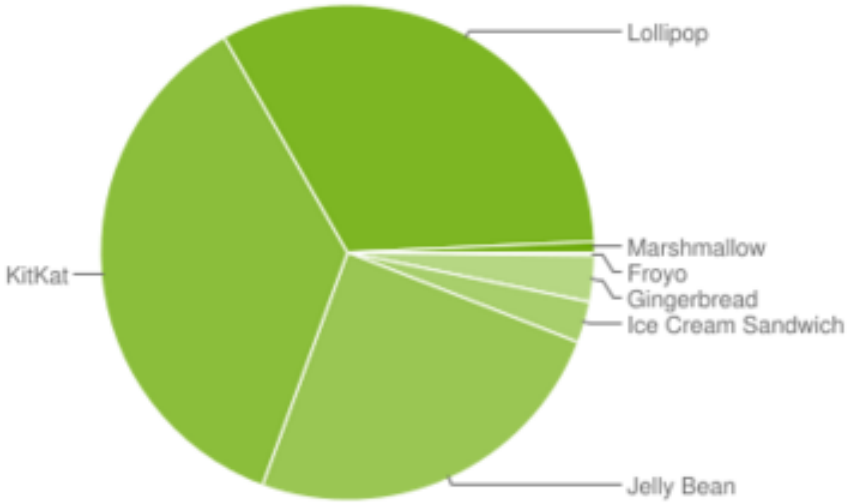
Data collected during a 7-day period ending on September 9, 2014.
Any versions with less than 0.1% distribution are not shown.

> **Note:** Because this data is gathered from the new Google Play Store app, which supports Android 2.2 and above, devices running older versions are not included. However, in August, 2013, versions older than Android 2.2 accounted for about 1% of devices that *checked in* to Google servers (not those that actually visited Google Play Store).

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

API targeting - 2016

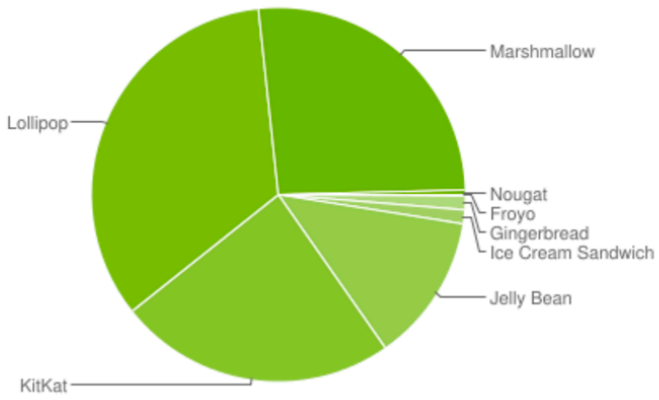
Version	Codename	API	Distribution
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	3.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.7%
4.1.x	Jelly Bean	16	9.0%
4.2.x		17	12.2%
4.3		18	3.5%
4.4	KitKat	19	36.1%
5.0	Lollipop	21	16.9%
5.1		22	15.7%
6.0	Marshmallow	23	0.7%



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

API Targeting– Jan 2017

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.2%
4.1.x	Jelly Bean	16	4.5%
4.2.x		17	6.4%
4.3		18	1.9%
4.4	KitKat	19	24.0%
5.0	Lollipop	21	10.8%
5.1		22	23.2%
6.0	Marshmallow	23	26.3%
7.0	Nougat	24	0.4%

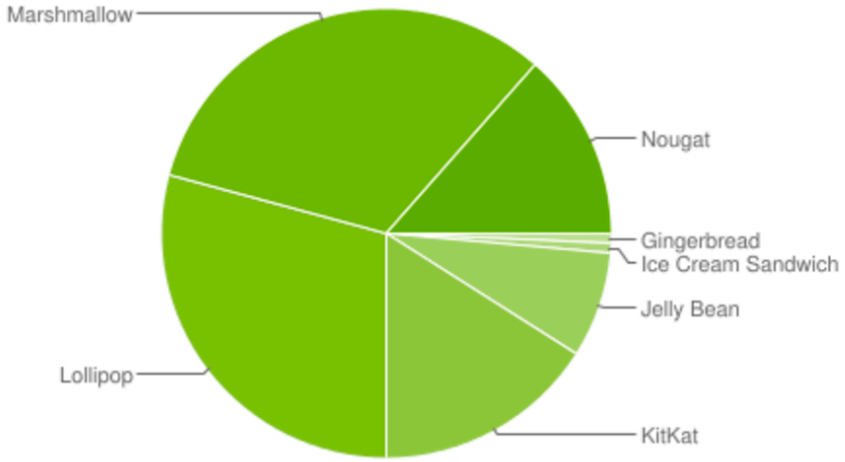


Data collected during a 7-day period ending on December 5, 2016.

https://developer.android.com/about/dashboards/index.html

API Targeting– Sept 2017

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.7%
4.1.x	Jelly Bean	16	2.7%
4.2.x		17	3.8%
4.3		18	1.1%
4.4	KitKat	19	16.0%
5.0	Lollipop	21	7.4%
5.1		22	21.8%
6.0	Marshmallow	23	32.3%
7.0	Nougat	24	12.3%
7.1		25	1.2%

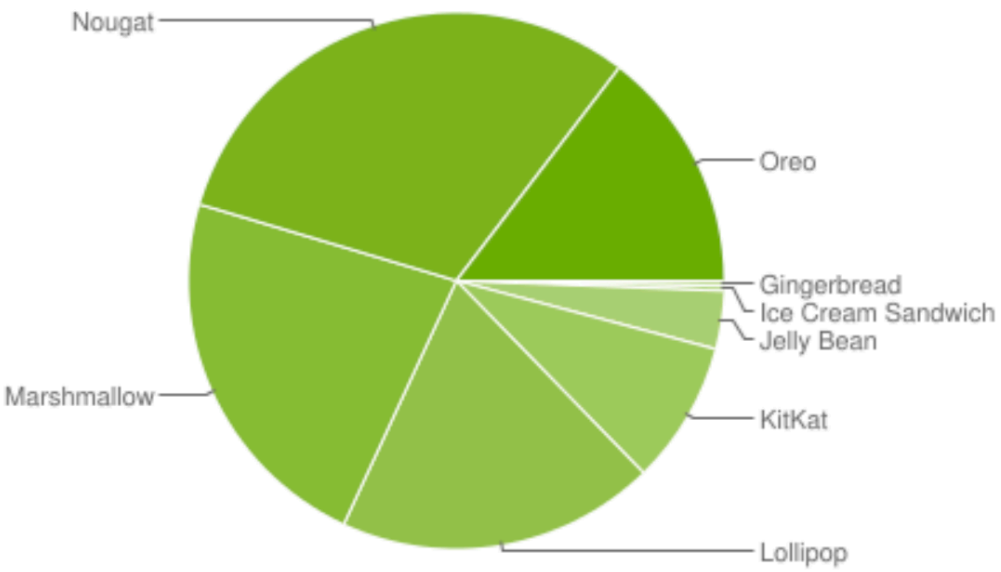


Data collected during a 7-day period ending on August 8, 2017.

Any versions with less than 0.1% distribution are not shown.

API Targeting– Sept 2018

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.8%
4.3		18	0.5%
4.4	KitKat	19	8.6%
5.0	Lollipop	21	3.8%
5.1		22	15.4%
6.0	Marshmallow	23	22.7%
7.0	Nougat	24	20.3%
7.1		25	10.5%
8.0	Oreo	26	11.4%
8.1		27	3.2%

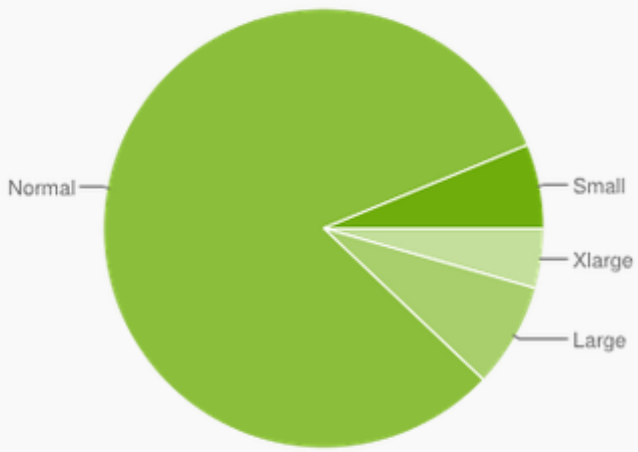


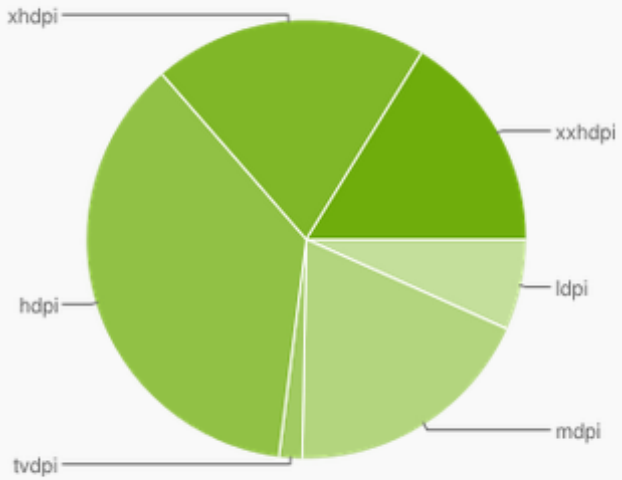
Data collected during a 7-day period ending on August 31, 2018.
Any versions with less than 0.1% distribution are not shown.

What next?

Sep 2014

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	6.2%						6.2%
Normal		10.6%		35.7%	19.2%	16.2%	81.7%
Large	0.5%	4.3%	1.7%	0.6%	0.6%		7.7%
Xlarge		3.7%		0.3%	0.4%		4.4%
Total	6.7%	18.6%	1.7%	36.6%	20.2%	16.2%	



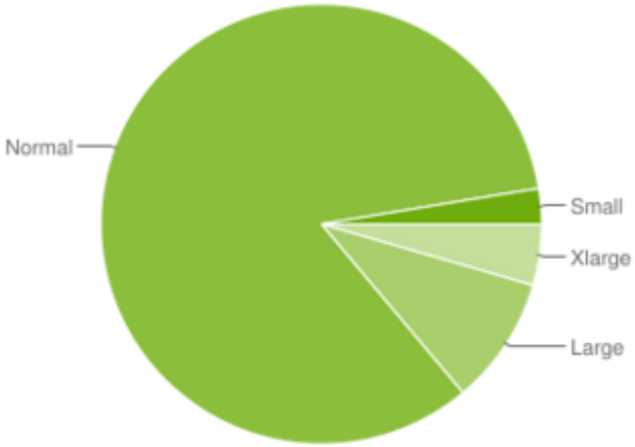


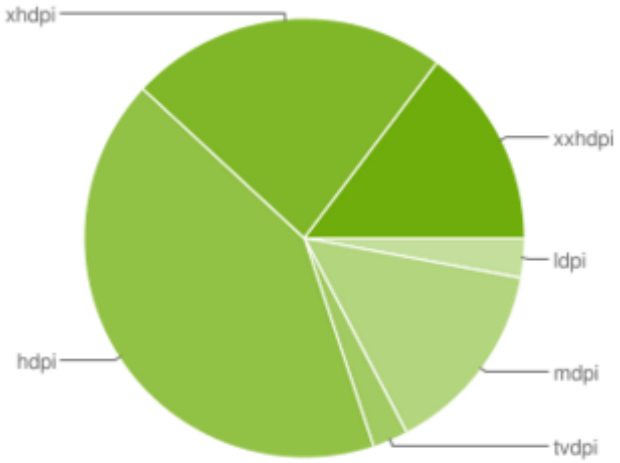
Data collected during a 7-day period ending on September 9, 2014.
Any screen configurations with less than 0.1% distribution are not shown.

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

Jan 2016

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	2.6%						2.6%
Normal		5.5%	0.1%	41.1%	22.1%	14.7%	83.5%
Large	0.3%	5.4%	2.5%	0.6%	0.6%		9.4%
Xlarge		3.5%		0.3%	0.7%		4.5%
Total	2.9%	14.4%	2.6%	42.0%	23.4%	14.7%	



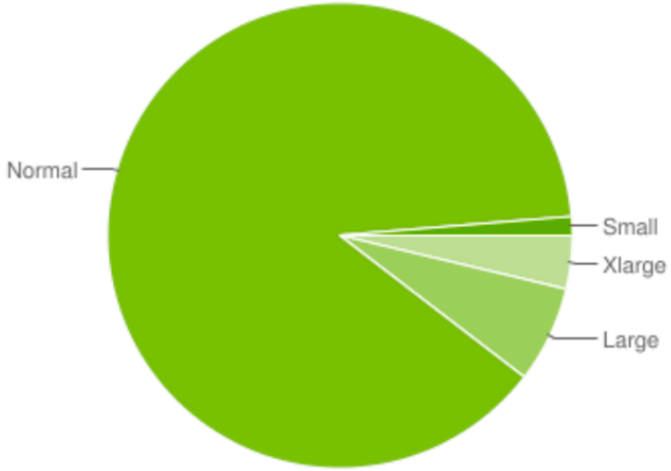


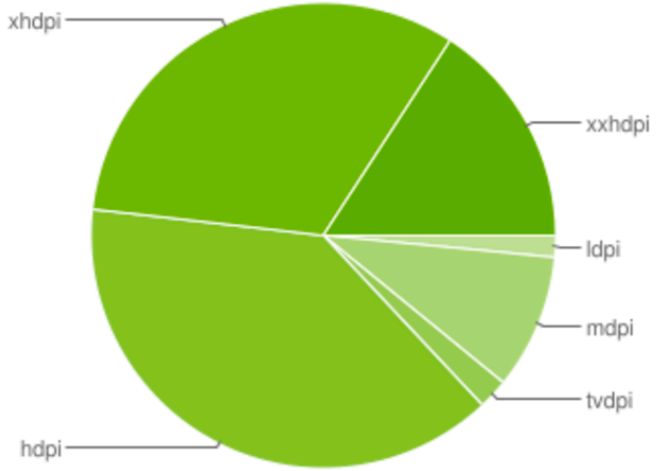
Data collected during a 7-day period ending on January 4, 2016.

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

Jan 2017

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	**Total**
Small	1.3%						1.3%
Normal		2.9%	0.2%	38.0%	31.4%	15.8%	88.3%
Large	0.2%	3.8%	1.9%	0.4%	0.4%		6.7%
Xlarge		2.7%		0.4%	0.6%		3.7%
Total	1.5%	9.4%	2.1%	38.8%	32.4%	15.8%	

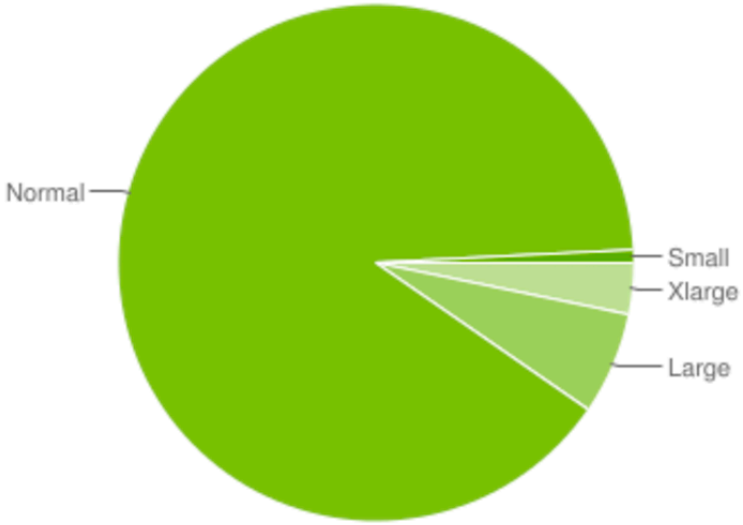


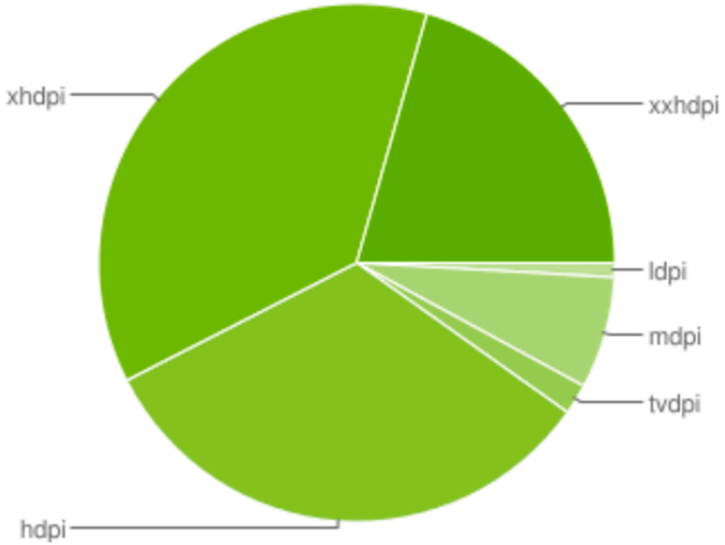


Data collected during a 7-day period ending on December 5, 2016.

Sept 2017

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	0.8%						0.8%
Normal		1.7%	0.2%	31.7%	35.5%	20.5%	89.6%
Large	0.1%	3.1%	1.7%	0.5%	0.9%	0.1%	6.4%
Xlarge		2.2%		0.5%	0.5%		3.2%
Total	0.9%	7.0%	1.9%	32.7%	36.9%	20.6%	

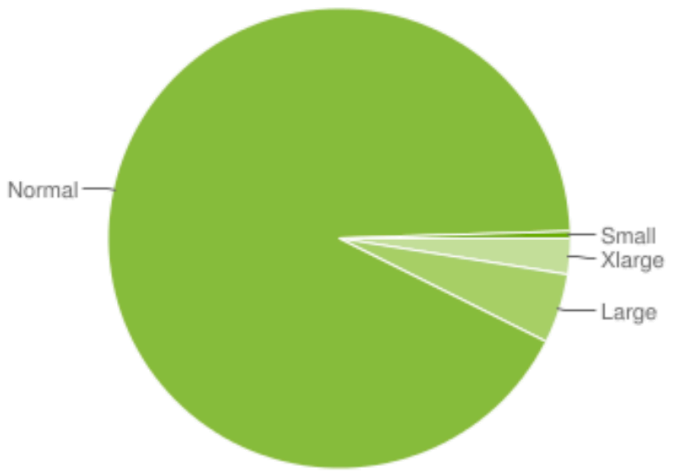


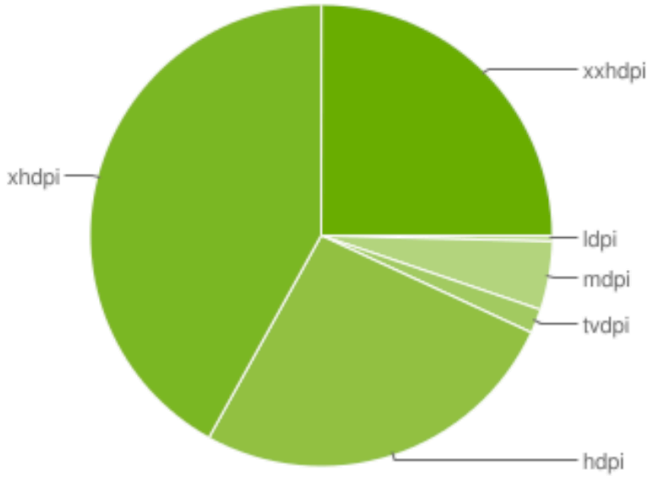


Data collected during a 7-day period ending on August 8, 2017.

Sept 2018

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	0.4%					0.1%	0.5%
Normal		0.8%	0.3%	25.4%	41.2%	24.4%	92.1%
Large		2.4%	1.4%	0.3%	0.3%	0.5%	4.9%
Xlarge		1.6%		0.5%	0.4%		2.5%
Total	0.4%	4.8%	1.7%	26.2%	41.9%	25.0%	





Data collected during a 7-day period ending on August 31, 2018.
Any screen configurations with less than 0.1% distribution are not shown.

API targeting

- "it's a good practice to support about 90% of the active devices, while targeting your app to the latest version"
- To do this, you can use the Android Support Library

  Be compatible w/ devices w/1.6+ (API level 4)

  http://developer.android.com/tools/support-library/index.html

We know why to target an API;

Now, *HOW* do we target an API?

The Manifest file!

ticTacToev6 × | MainActivity.java ×

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 22
    buildToolsVersion "22.0.1"

    defaultConfig {
        applicationId
        minSdkVersion 14
        targetSdkVersion 22
        versionCode 1
        versionName "1.0"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
```

minSDKVersion: Android 4.0.3 – API Level 15

SDK targeting

```
defaultConfig {
   applicationId
   minSdkVersion 14
   targetSdkVersion 22
   versionCode 1
   versionName "1.0"
}
```

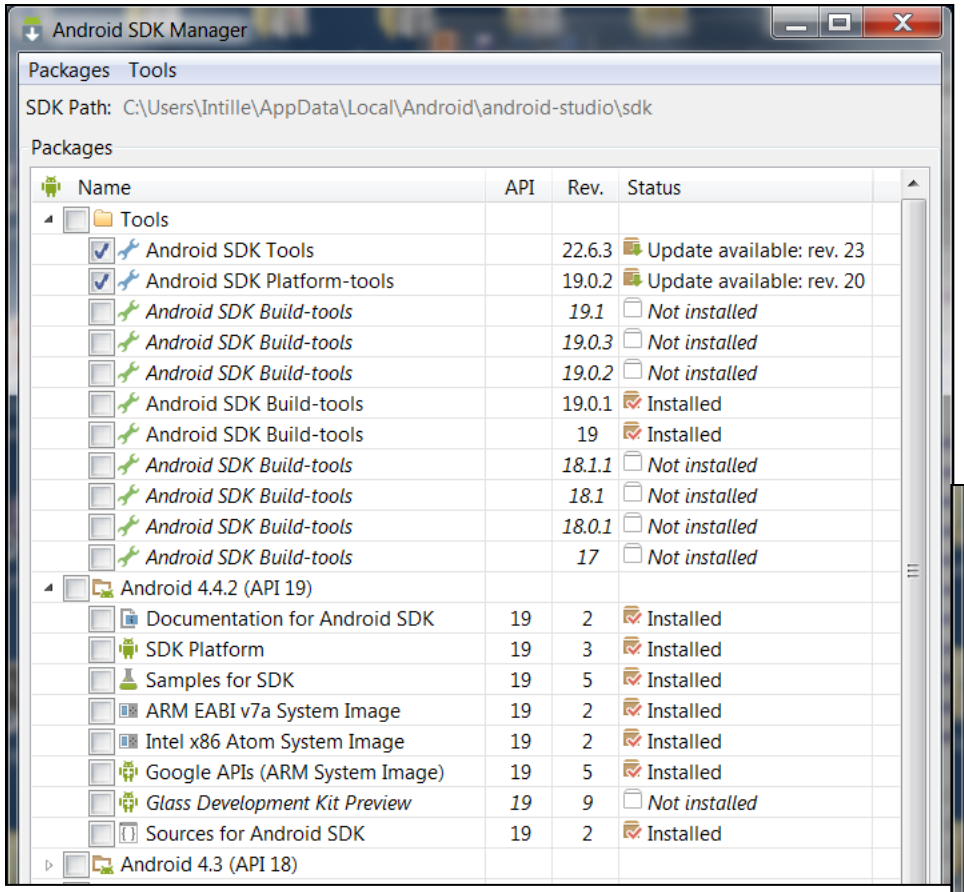
minSdkVersion: The minimum API Level required for the application to run (prevents install otherwise). You NEED to set this number, otherwise it will assume a level of 1 (which you don't want!).

SDK targeting

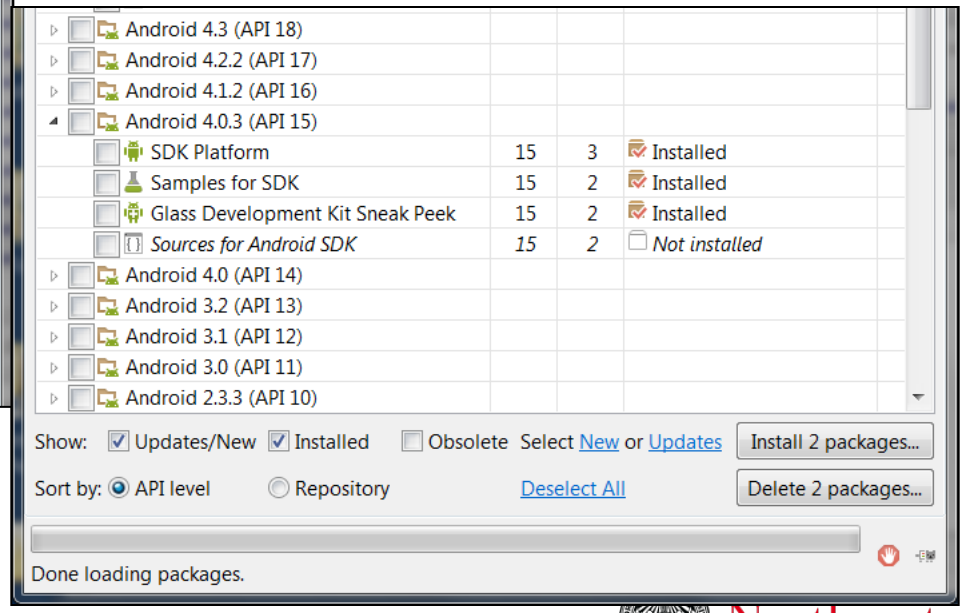
targetSdkVersion: You have tested against the target version and the system should not enable [illegible] forward-compatibility with t[illegible] own to minSdkVersion). Se[illegible] sting with (a recent one, most lik[illegible]

The important thing is, "if the API level of the platform is higher than the version declared by your app's targetSdkVersion, the system may enable compatibility behaviors to ensure that your app continues to work the way you expect." What Google wants you to do is, "To maintain your application along with each Android release, you should increase the value of this attribute to match the latest API level, then thoroughly test your application on the corresponding platform version."

Android SDK manager



Make sure you have the right SDKs and tools installed



- <See Android SDK Manager live>

Manifest warning!!

Activities, services, and content providers in code but not manifest won't run!

Parts of a project: Resources

- Resources separate from the source code
  - XML layout: animations, menus, styles, colors, layout of UI
  - Images, audio files
  - Strings
  - Every resource has unique resource ID
    - E.g., logo.png (saved in res/drawable/ dir); SDK tools generate ID named R.drawable.logo
- Designed to allow qualifiers (e.g., language)

DDMS and debugging

Sudokuv4 - [E:\NUMAD15S\NUMAD15S-AmeyaPandilwar] - [app] - ...\app\src\main\java\edu\neu\madcourse\ameyapandilwar\sudoku\

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

NUMAD15S-AmeyaPandilwar > app > src > main > java > edu > neu > madcourse > ameyapandilwar > sudoku > Sudoku

Project

- src
  - main
    - assets
    - java
      - edu.neu.madcourse.ameyapandilwar
        - hellomad
          - AboutMe
          - MainMenu
        - sudoku
          - About
          - Game
          - Keypad
          - Music
          - Options
          - Prefs
          - PuzzleView
          - Sudoku
    - res
    - AndroidManifest.xml

Sudoku.java

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

    // Set up click listeners for all the buttons
    View continueButton = findViewById(R.id.continue_button);
    continueButton.setOnClickListener(this);
    View newButton = findViewById(R.id.new_button);
    newButton.setOnClickListener(this);
    View aboutButton = findViewById(R.id.about_button);
    aboutButton.setOnClickListener(this);
    View exitButton = findViewById(R.id.exit_button);
    exitButton.setOnClickListener(this);
}

@Override
protected void onResume() {
    super.onResume();
    Music.play(this, R.raw.main);
}
```

Android DDMS

Devices | ADB logs | logcat

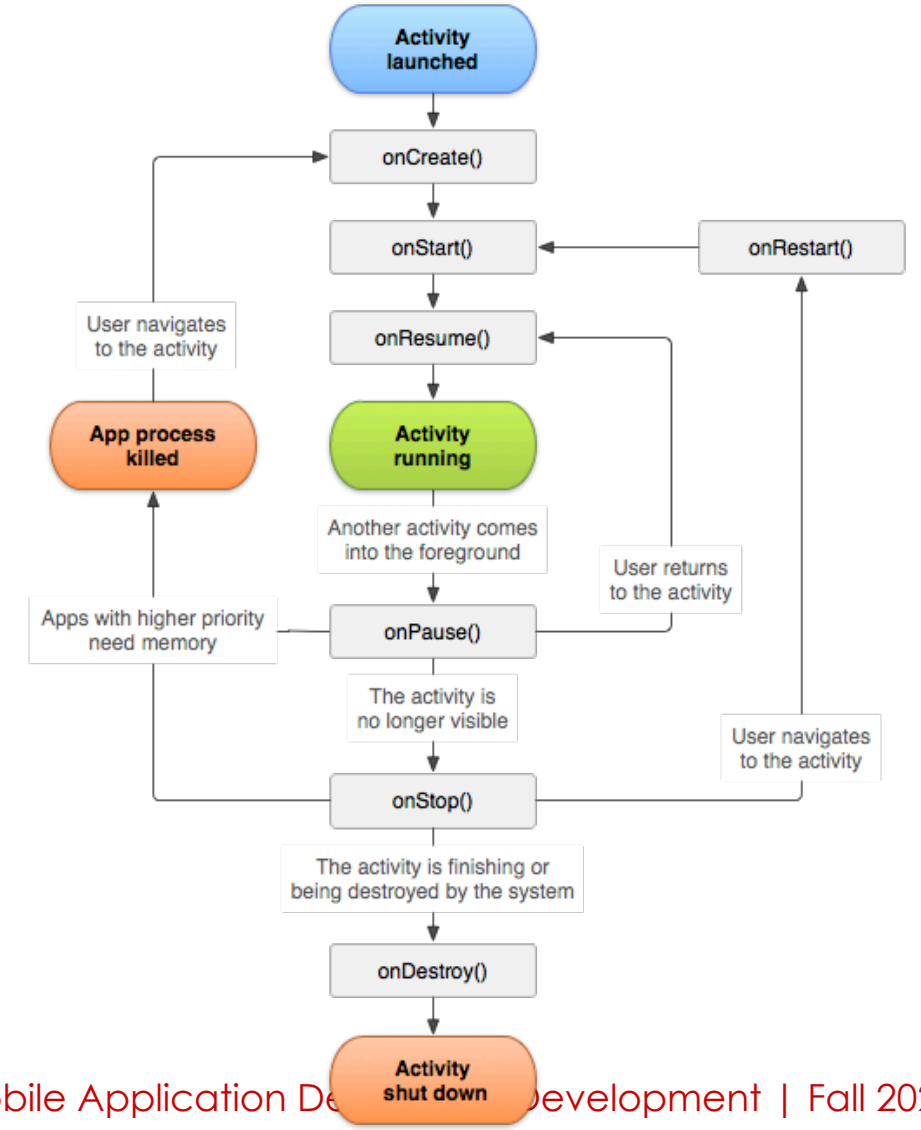
```
01-26 11:06:37.541    328-328/edu.neu.madcourse.ameyapandilwar D/dalvikvm:  GC_EXTERNAL_ALLOC freed 64K, 52% free 2589K/5379K, exte
01-26 11:06:40.981    328-328/edu.neu.madcourse.ameyapandilwar W/KeyCharacterMap:  No keyboard for id 0
01-26 11:06:40.991    328-328/edu.neu.madcourse.ameyapandilwar W/KeyCharacterMap:  Using default keymap: /system/usr/keychars/qwert
01-26 11:06:46.000    328-328/edu.neu.madcourse.ameyapandilwar D/Sudoku:  clicked on 0
01-26 11:06:46.060    328-328/edu.neu.madcourse.ameyapandilwar D/Sudoku:  onCreate
01-26 11:06:46.371    328-328/edu.neu.madcourse.ameyapandilwar D/Sudoku:  onSizeChanged: width 35.555557, height 53.666668
01-26 11:06:46.421    328-328/edu.neu.madcourse.ameyapandilwar D/Sudoku:  selRect=Rect(0, 0 - 35, 53)
```

4: Run | TODO | 6: Android | Terminal | 0: Messages

Session 'app': running (2 minutes ago)

- Read messages carefully
- Use debugger
- Use logcat
- Ask others to help you find bugs
- Take breaks if you get frustrated

Activity Lifecycle (…a bit more!)



- Paused

  Lost focus, but visible (e.g. transparent); retains state

- Stopped

  Completely obscured; retains state; often killed

Important: In either case, system can drop the activity by asking it to finish or by killing the process

Activity lifecycle

- Lifetime
  - onCreate (setup “global” state)
  - onDestroy (release all major resources)
- Visible lifetime (“behind glass door”)
  - onStart (register BroadcastReceivers)
  - onStop (unregister)
- Foreground lifetime (frequent switching)
  - onResume (user interacting; lightweight code)
  - onPause (commit changes;lightweight code)

Typical activity

```
public class Activity extends ApplicationContext {
     protected void onCreate(Bundle savedInstanceState);

     protected void onStart();

     protected void onRestart();

     protected void onResume();

     protected void onPause();

     protected void onStop();

     protected void onDestroy();
 }
```

THE UI

What is the User Interface?

- A set of components that:
  - present information to users,
  - allow users to input information,
- It's the thing that sits between the user and your logic

Big Picture View of Creating a UI

- Have an idea in mind of what it should look like and how it should act
- Decide on a layout/Layout Manager
- Decide on widgets
- Decide which widgets need to respond to user input
  - Which widgets the user will touch/interact directly with
- Write the code that does something when a widget does something
- Link the widget to the code
- Test, iterate.

Example: Assignment 1

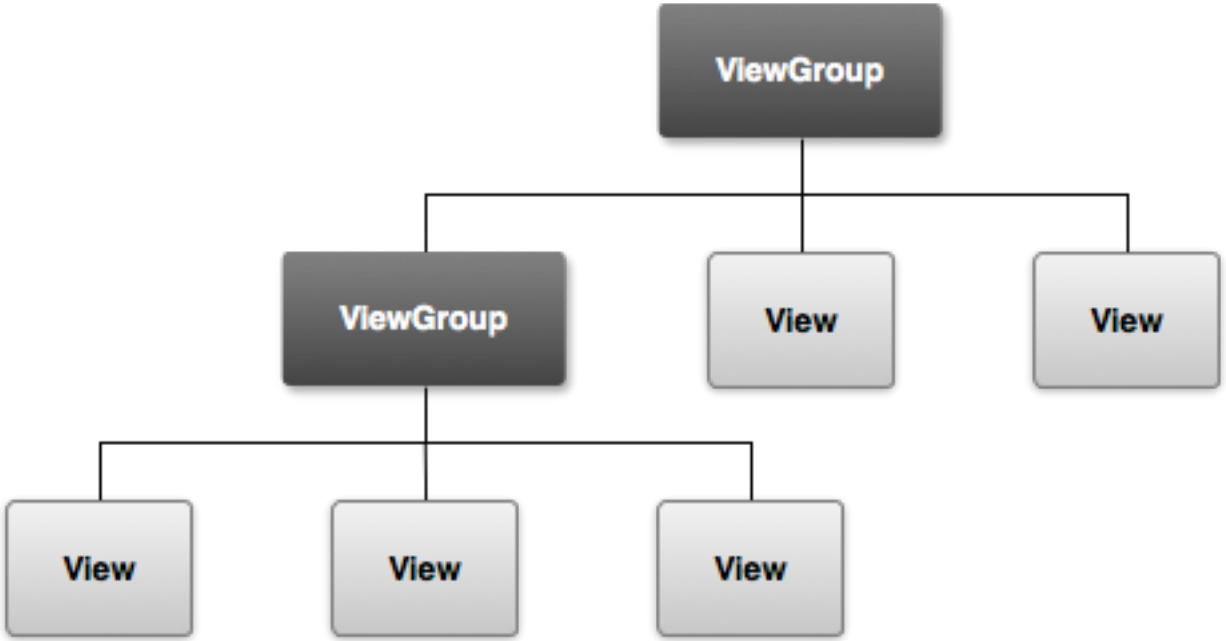
- Sketch of app
- Layout: LinearLayout
- Widgets: Buttons
- OnClick(): Show a new activity
- Set the onClick listener on the buttons
- Run

UI Components

- Layouts
- Input Controls
- Input Events
- Dialogs
- Notifications
- Toasts
- App Bar
- …

Views & View groups

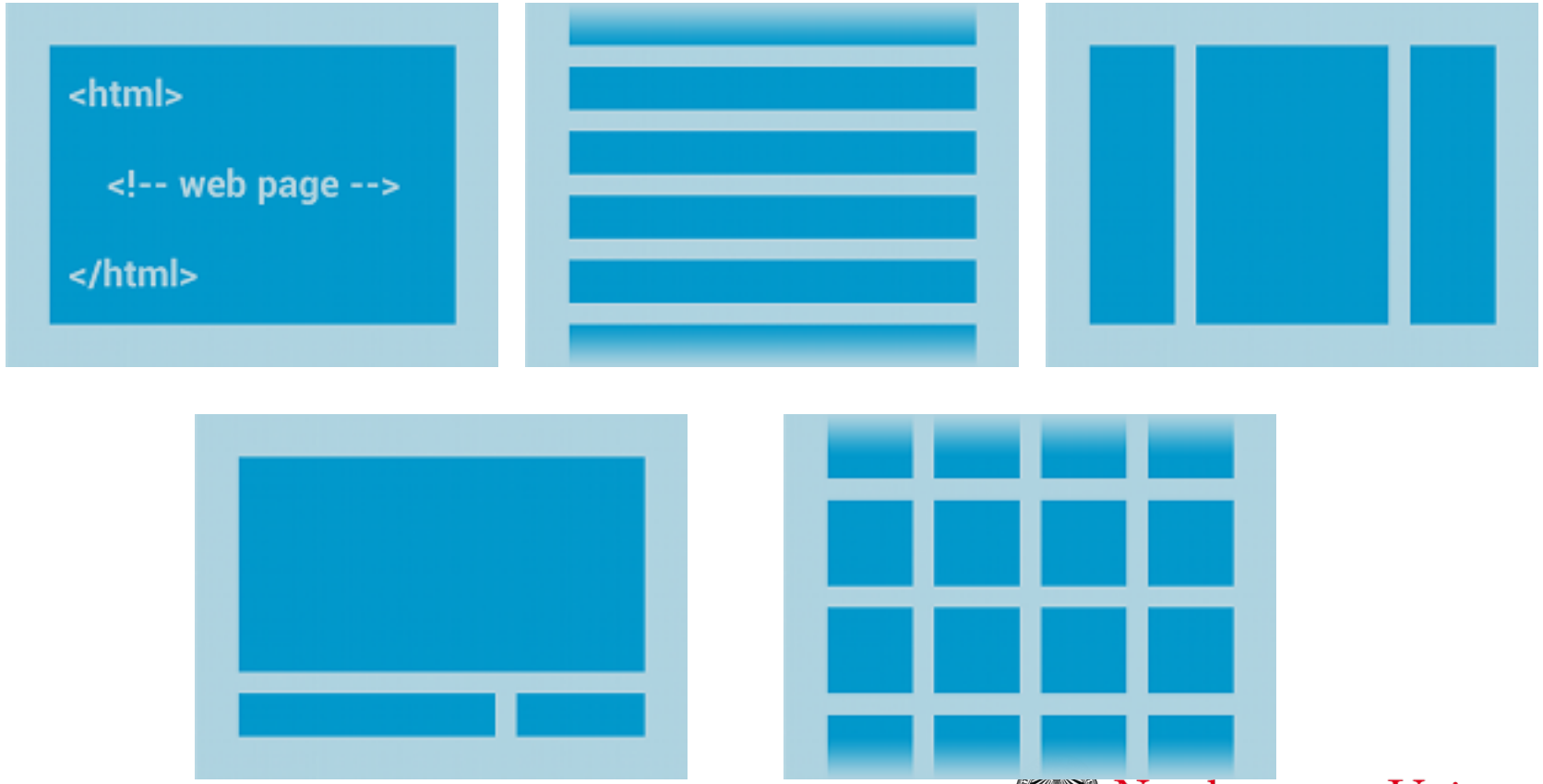
- All UI components extend a View or ViewGroup
  - Views: a UI component
  - ViewGroup: a UI component that holds other UI components



https://developer.android.com/guide/topics/ui/overview.html

Layouts

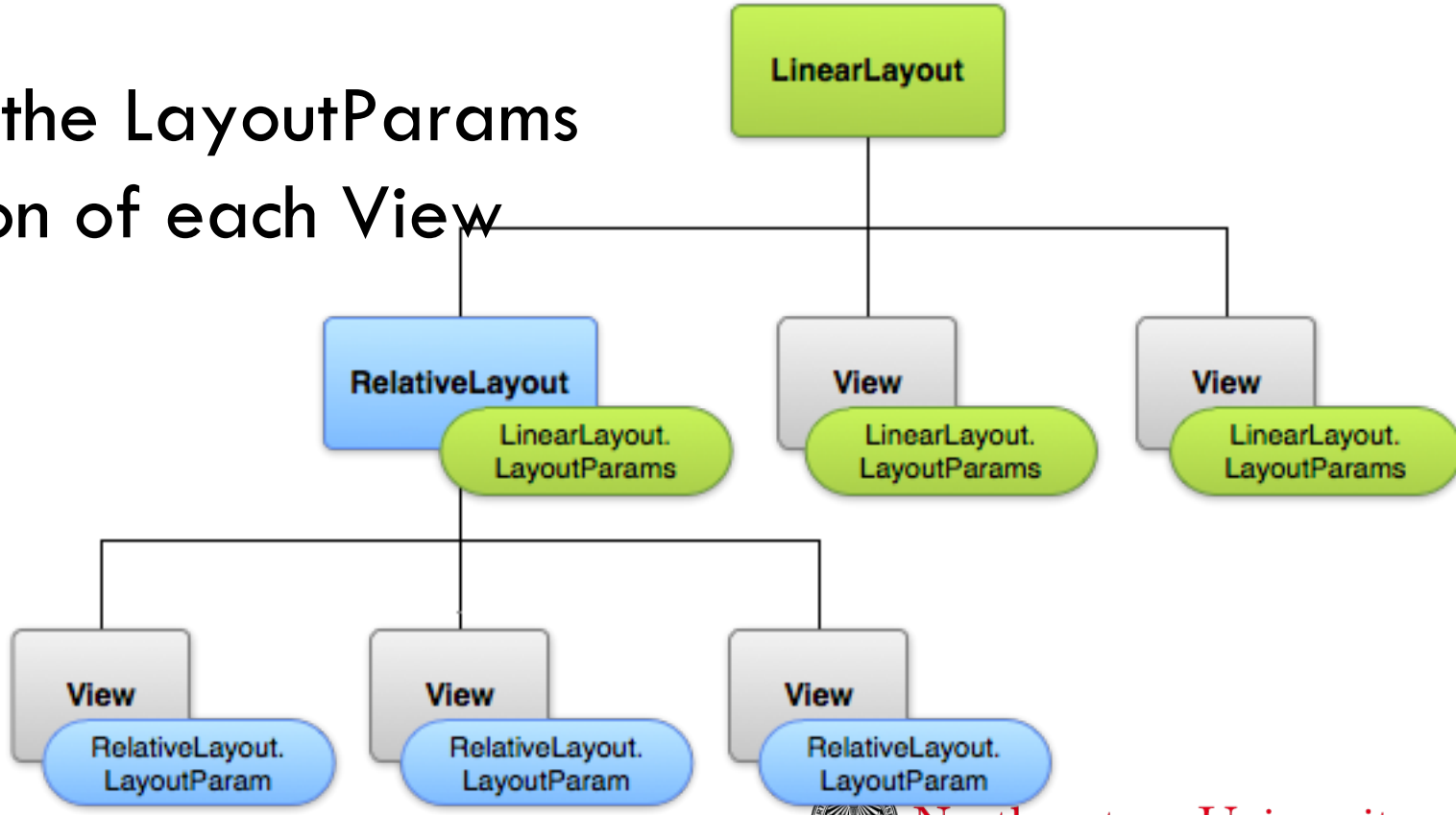
- Presenting information and *stuff* to the user
  - Grid
  - Frame
  - Linear
  - Relative
  - Constraint
  - Web



https://developer.android.com/guide/topics/ui/declaring-layout.html

Layout Parameters

- Specifies how each View relates to other Views
- The Layout Manager uses the LayoutParams to determine size & position of each View



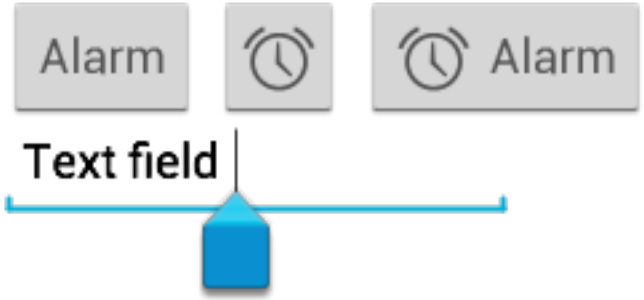
https://developer.android.com/guide/topics/ui/declaring-layout.html

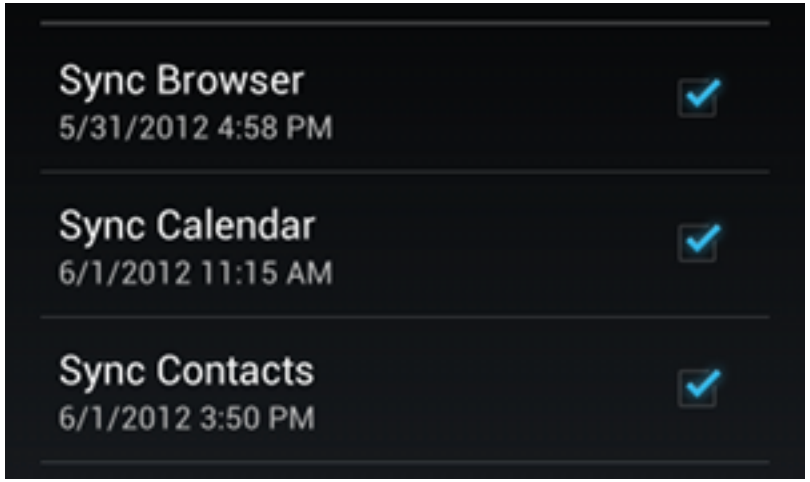
Northeastern University
College of Computer and Information Science

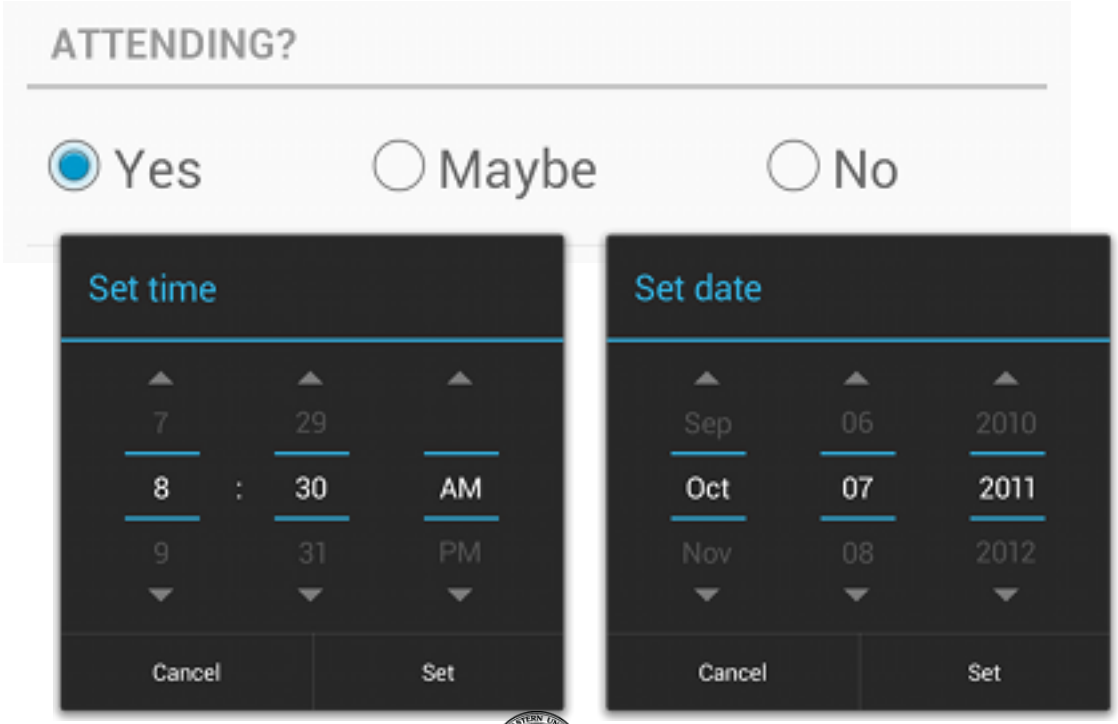
Building a Layout with an Adapter

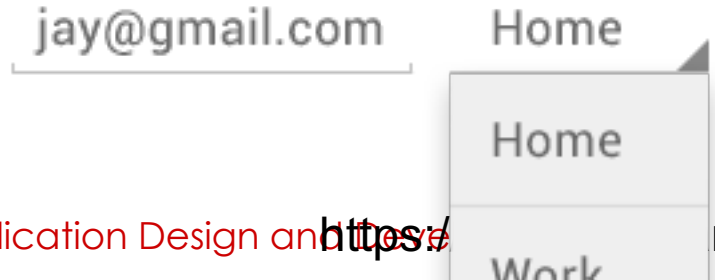
Input Controls

- Getting input from the user
  - Button
  - Text field
  - Checkbox
  - Radio button
  - Toggle button
  - Spinner
  - Picker









https://developer.android.com/guide/topics/ui/controls.html#CommonControls

Input Events

- The process of getting input from users
- Input controls generate Input Events.
- Event Listeners:
  - onClick()
  - onLongClick()
  - onFocusChange()
  - onKey()
  - onTouch()
  - onCreateContextMenu()

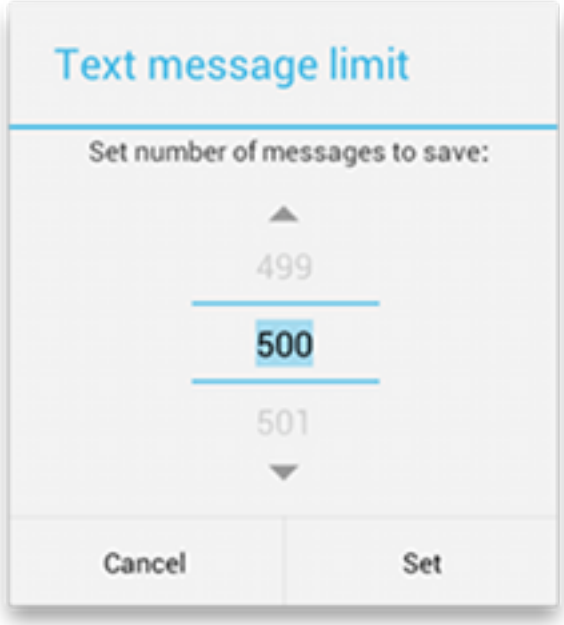
Input Events, cont.

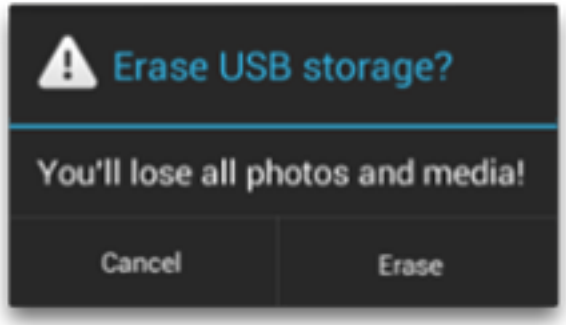
- Event Handlers:
  - onKeyDown()
  - onKeyUp()
  - onTrackballEvent()
  - onTouchEvent()
  - onFocusChanged()

- You write code that does something, overriding an event listener
- Create a UI component
- Assign an event listener to the UI component
- When that event happens on that UI component, your code is notified and run.

Dialogs

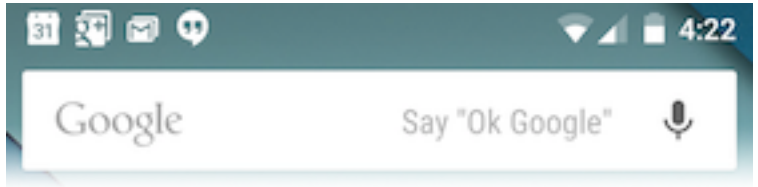
- Provide non-blocking information to the user
- Good for short feedback, errors, helpful info, etc.
- Example:

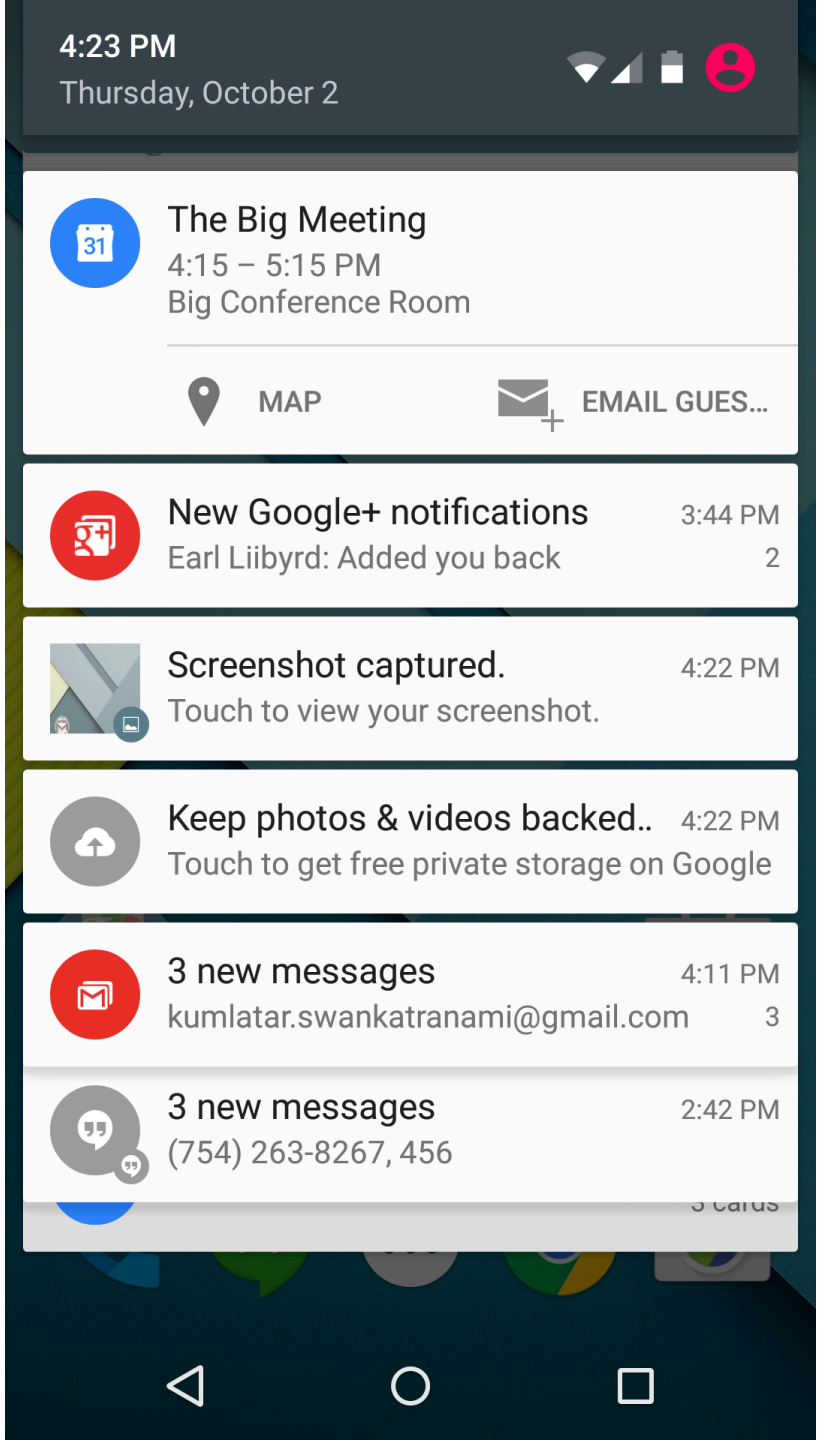




Notifications

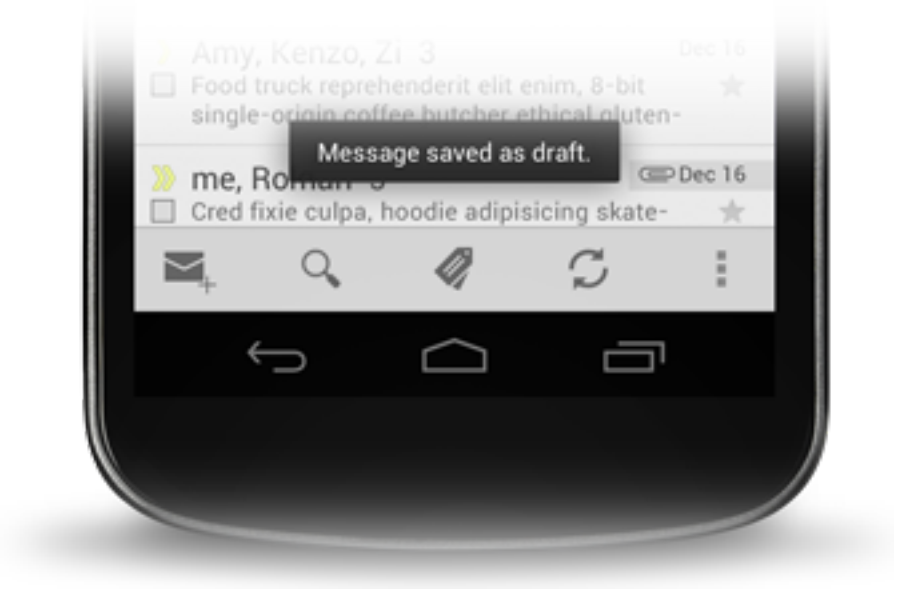
- Provide information to the user outside of your app
- Examples:





Toasts

- Small message
- Only as big as it needs to be
- Fades in & out fairly quickly



Building the UI

- Decide on your Activities
- For each Activity, define your Layout & UI Components
- Build the UI in XML (or code)
- Write the Activity code that uses the layout defined in XML

Building the UI with XML

Inflating a layout (XML option)

```
@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);

  setContentView(R.layout.main);
  TextView myTextView =
       (TextView)findViewById(R.id.myTextView);
}
```

Building the UI with Java

Inflating a layout (manual option)

```
@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);

  TextView myTextView = new TextView(this);
  setContentView(myTextView);

  myTextView.setText("Hello, Android");
}
```

- Combining controls
- Adapting for different screen sizes and shapes

Resolution and density

- UIs running on a variety of screen sizes and densities
- Use resource qualifiers to deal with this “floodgate” of devices
- Potential hazard!

(They also have small changes in hardware and other quirks to keep in mind!)

Resolution and density

- Screen size
  - Small (smaller than 3.2” screen)
  - Medium (“Typical” size)
  - Large
- Pixel density
  - ldpi (100-140 dpi)
  - mdpi (140-180 dpi)
  - hdpi (190-250 dpi)
  - nodpi (no scaling regardless screen density)

Resolution and density

Aspect ratio

- long (significantly wider in landscape)
- notlong (typical ratio)

```
res/layout-small-long/ // Layouts for small, long
screens.
res/layout-large/      // Layouts for large screens.
res/drawable-hdpi/     // Drawabless for high density
screens.
```

To force screen size

- Use manifest

```
<supports-screens
  android:smallScreens="false"
  android:normalScreens="true"
  android:largeScreens="true"
  android:anyDensity="true"
/>
```

- anyDensity = false forces Android to scale

Tips

- Don't make assumptions about screens
- Avoid hard-coded pixel values
  - wrap_content
  - fill_parent
  - dp and sp
- Avoid AbsoluteLayout class
- RelativeLayout for complex UIs is probably what you need

Tips 2

Use Drawable resources rather than bitmaps

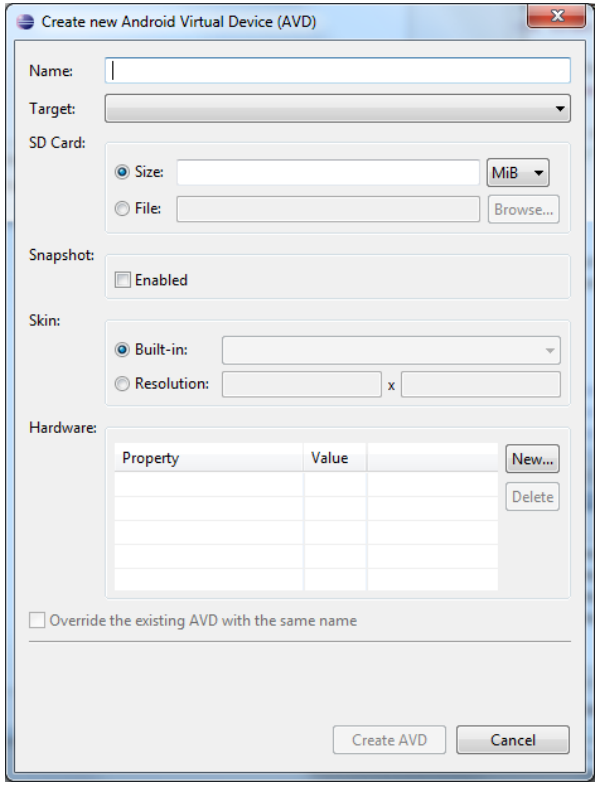
- NinePatches
- Shape Drawables
- Gradient Drawables
- Composite and transformable drawables
  - Rotate and Scale Drawables
  - LevelListDrawables
  - StateListDrawables

Asset that can't be scaled

- Image assess for each category to avoid aliasing/artifacts
  - res/drawable-ldpi
  - res/drawable-mdpi
  - res/drawable-hdpi
- Layouts for different phones
  - res/layout-small
  - res/layout-normal
  - res/layout-large

Test, test, test…

Using AVD you can define arbitrary screen resolutions and pixel densities



Android Application Components

- Activities
- Services
- Broadcast Receivers
- Content providers
- Intents
- Widgets
- Notifications