Android

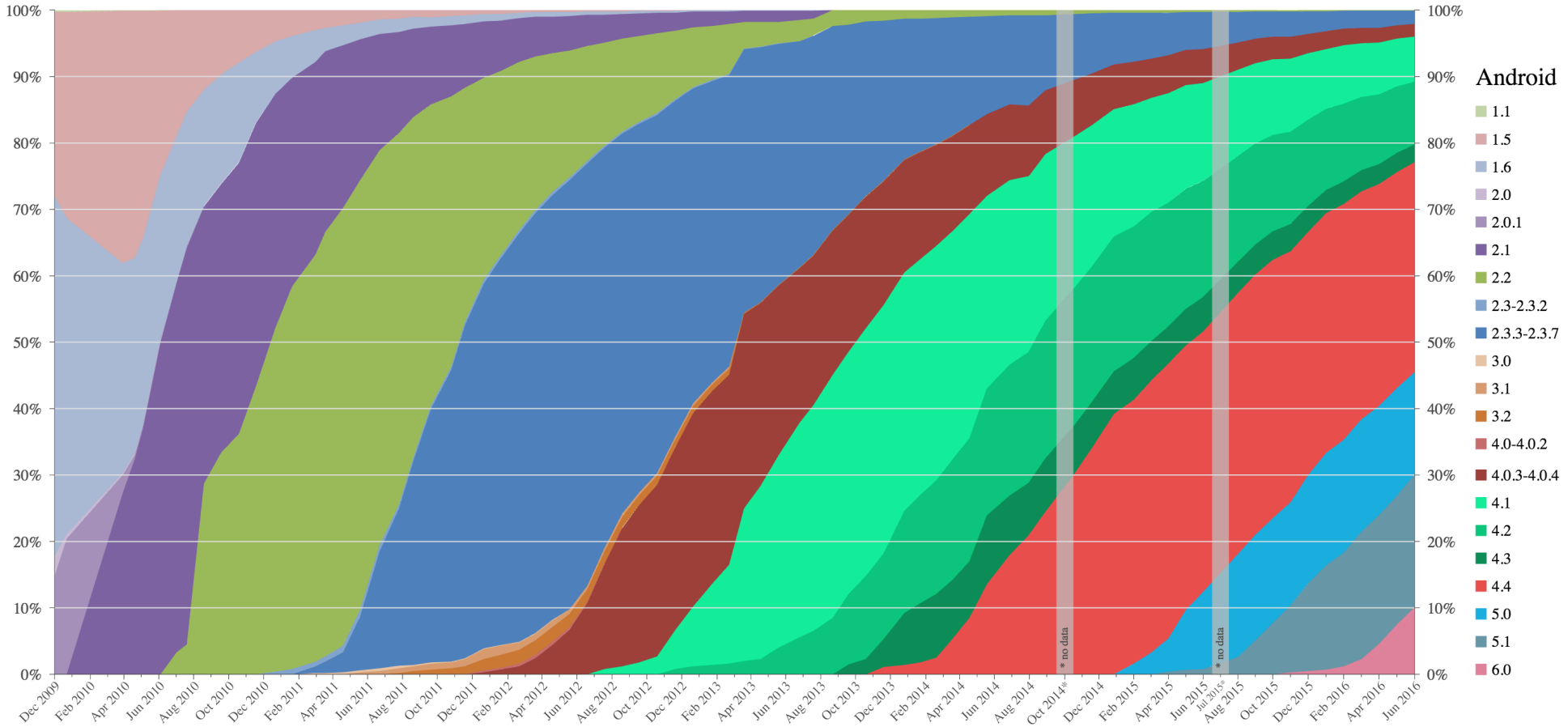
History

- Andy Rubin
  - worked for Apple, WebTV, Danger Inc.
  - in 2003, start a company Android, Inc.
  - Google acquired Android in 2005
- in 2007, Google announced (by Andy)
  - Open source project
  - Formed Open Handset Alliance
- in 2008
  - Android developer challenge ($1M)
  - Android market launched
  - AOSP (Android Open Source Project) launched
  - T-Mobile G1 phone

Android versions

Code name	Version number	Initial release date	API level	Support status
N/A	1.0	September 23, 2008	1	Discontinued
	1.1	February 9, 2009	2	Discontinued
Cupcake	1.5	April 27, 2009	3	Discontinued
Donut	1.6	September 15, 2009	4	Discontinued
Eclair	2.0 – 2.1	October 26, 2009	5–7	Discontinued
Froyo	2.2 – 2.2.3	May 20, 2010	8	Discontinued
Gingerbread	2.3 – 2.3.7	December 6, 2010	9–10	Discontinued
Honeycomb	3.0 – 3.2.6	February 22, 2011	11–13	Discontinued
Ice Cream Sandwich	4.0 – 4.0.4	October 18, 2011	14–15	Discontinued
Jelly Bean	4.1 – 4.3.1	July 9, 2012	16–18	Discontinued
KitKat	4.4 – 4.4.4	October 31, 2013	19	Security updates only
Lollipop	5.0 – 5.1.1	November 12, 2014	21–22	Supported
Marshmallow	6.0 – 6.0.1	October 5, 2015	23	Supported
Nougat	**7.0 – 7.1**	**August 22, 2016**	**24–25**	Supported

Versions on Market




Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Notification Manager
Package Manager
Telephony Manager
Resource Manager
Location Manager
XMPP Service
Libraries
Surface Manager
Media Framework
SQLite
OpenGL|ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Bluetooth Driver
Flash Memory Driver
Binder (IPC) Driver
USB Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Installation Overview

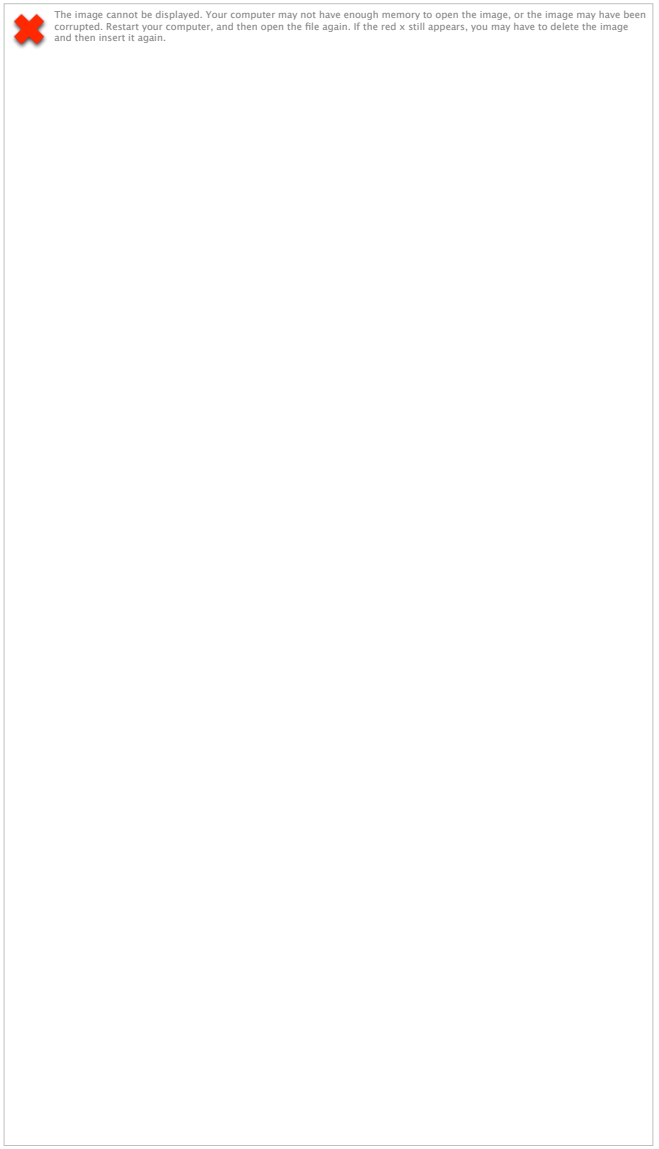
- Install JDK
  - Set JAVA_HOME
- Install Android Studio
  - Run SDK manager to install
- Install Genymotion
  - Install VirtualBox

Create the First App: Hello World

- Create a new Android project with empty Activity
  - App name: Hello World
    - Shown in Play Store
  - App domain: mp2016f.mju.ac.kr
  - Package Name: kr.ac.mju.mp2016f.helloworld
    - Reverse domain name + app name
    - Follows Java's package name convention
    - Why reverse order?

Project Structure

- App/src
  - Source files
- App/src/main/java
  - Activity codes
- App/src/main/res (resources)
  - /Drawable: images for layouts
  - /Layout: user interface defined in XML
  - /Values: various global values defined in XML
- App/build.gradle
  - Building information



Binding between Layout & Value

- app/src/main/res/layout/activity_main.xml
  - <TextView …. android:text="@string/app_name" />
- app/src/main/res/values/strings.xml
  - <string name="app_name">Hello World</string>

Binding between Activity & Layout

- Activity calls sets content view to a layout
  - setContentView(R.layout.<layout_name>)

  →sets the layout of this activity to layout/<layout_name>.xml

- ```
  public class MainActivity extends AppCompatActivity {
      @Override
      protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.activity_main);
      }
  }
  ```
  ```

@Override

- Java annotation, providing meta data for code
  - Tells compiler about class, interface, method, fields, local variables, …
- Built-in Java annotations
  - Disappear in compiled java code
  - @Deprecated
  - @Override
  - @SuppressWarnings
- Custom Java annotations
  - Defined by user
  - Used for Java Reflection
- Why @Override is useful?
  - If used, compiler checks if it actually overrides
  - Gets error if the overriding method signature doesn't match any superclass method
  - Easy to understand the code

Manifest file

- app/src/main/res/ AndroidManifest.xml
- Summary of the App
  - User permissions
  - Declare Activities, Services, Content Provider, Broadcast Receiver, …
  - API Level
  - HW/SW components to be used

-
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://
schemas.android.com/apk/res/android"
   package="kr.ac.mju.mp2016f.helloworld">

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">
      <activity android:name=".MainActivity">
         <intent-filter>
            <action
android:name="android.intent.action.MAIN" />

            <category
android:name="android.intent.category.LAUNCHE
R" />
         </intent-filter>
      </activity>
   </application>

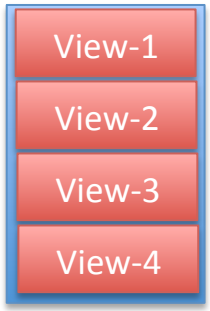
</manifest>
```

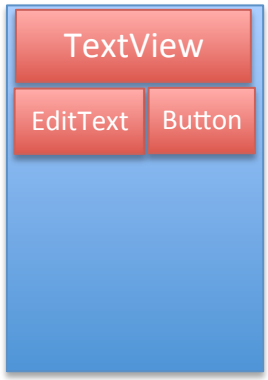
UI: Overview

- All UI elements are either
  - View
    - Base class for widgets: Button, TextView, EditText, CheckBox, RadioButton
  - ViewGroup
    - Subclass of View
    - Invisible container of Views or VieGroups
- Layouts
  - Subclasses of ViewGroup
  - Linear Layout
  - Relative Layout
  - Frame Layout
  - List View
  - Grid View
  - Table Layout

Linear Layout

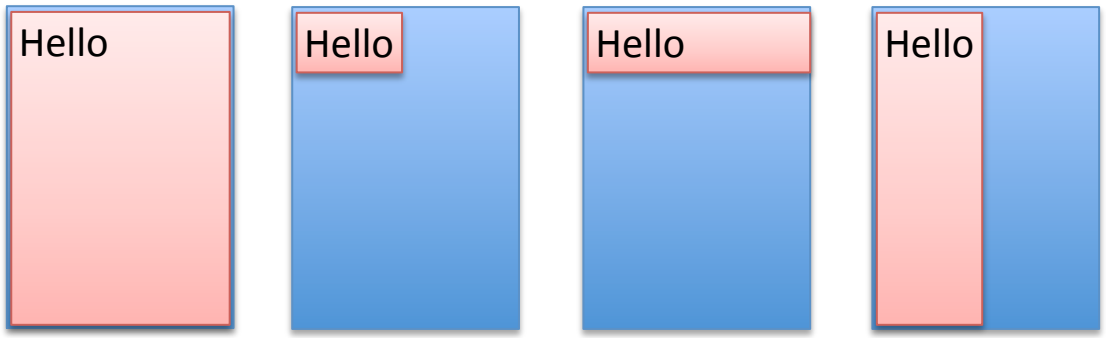
- Puts child views one after another, vertically or horizontally
  - <LinearLayout android:orientation="horizontal"|"vertical">
    <*view-1*…/>
    …
    <*view-N*…/>
    </LinearLayout>
- Nested LinearLayouts for complex a layout
  - <LinearLayout *vertical*>
    <TextView … />
    <LinearLayout *horizontal*>
    <EditText … />
    <Button …/>
    </LinearLayout>
    </LinearLayout>





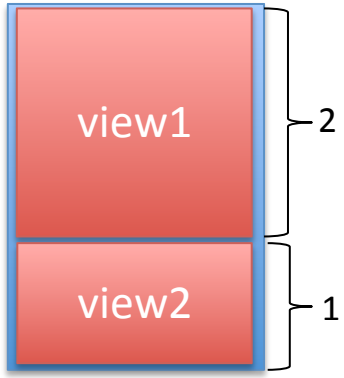
Controlling View(Group) Size

- <*viewname*
  android:layout_width=*size*
  android:layout_height=*size*
  android:layout_weight=*weight* />
- *size*= match_parent (fill up the parent's area)
  | wrap_content (only as much as needed for the content)
  | # px (# pixels)
  | # dp (# density-independent pixel (dip)
  1 dp = 1 px for 160 dpi,
  thus 160 dp=1 inch for any screen density)

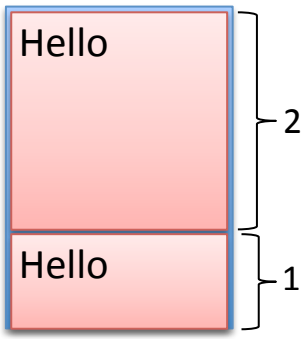


Proportional View(Group) Size

- *weight*: number defining the proportion of the view relative to sibling views' weights
  - If view1's weight=2, view2's weight=1
    then view1:view2 = 2:1, thus view1 is 2/3 wide of the parent size
  - <LinearLayout ..orientation=vertical>
    <*view1* ..height=wrap_cont/>
    <*view2* ..height=wrap_cont/>
    </LInearLayout>



View1: hei=wrap
View2: hei=wrap



View1: weight=2
View2: weight=1

View1: weight=1
View2: [weight=0]

LogCat & Log class

- LogCat
  - Command line tool to print system & app log messages
  - App can output log messages to LogCat via Log class
- LogCat window in Android Studio
  - Shows logcat output to a dedicated window
- Log class
  - Log.v (…) Less priority
  - Log.d (...)
  - Log.i (...)
  - Log.w (...)
  - Log.e (...) Higher priority
- Output format
  - Call Log.<priority>(<tag>, <message>) outputs:
    <date> <time> <PID>-<TID>/<package> <priority>/<tag>: <message>