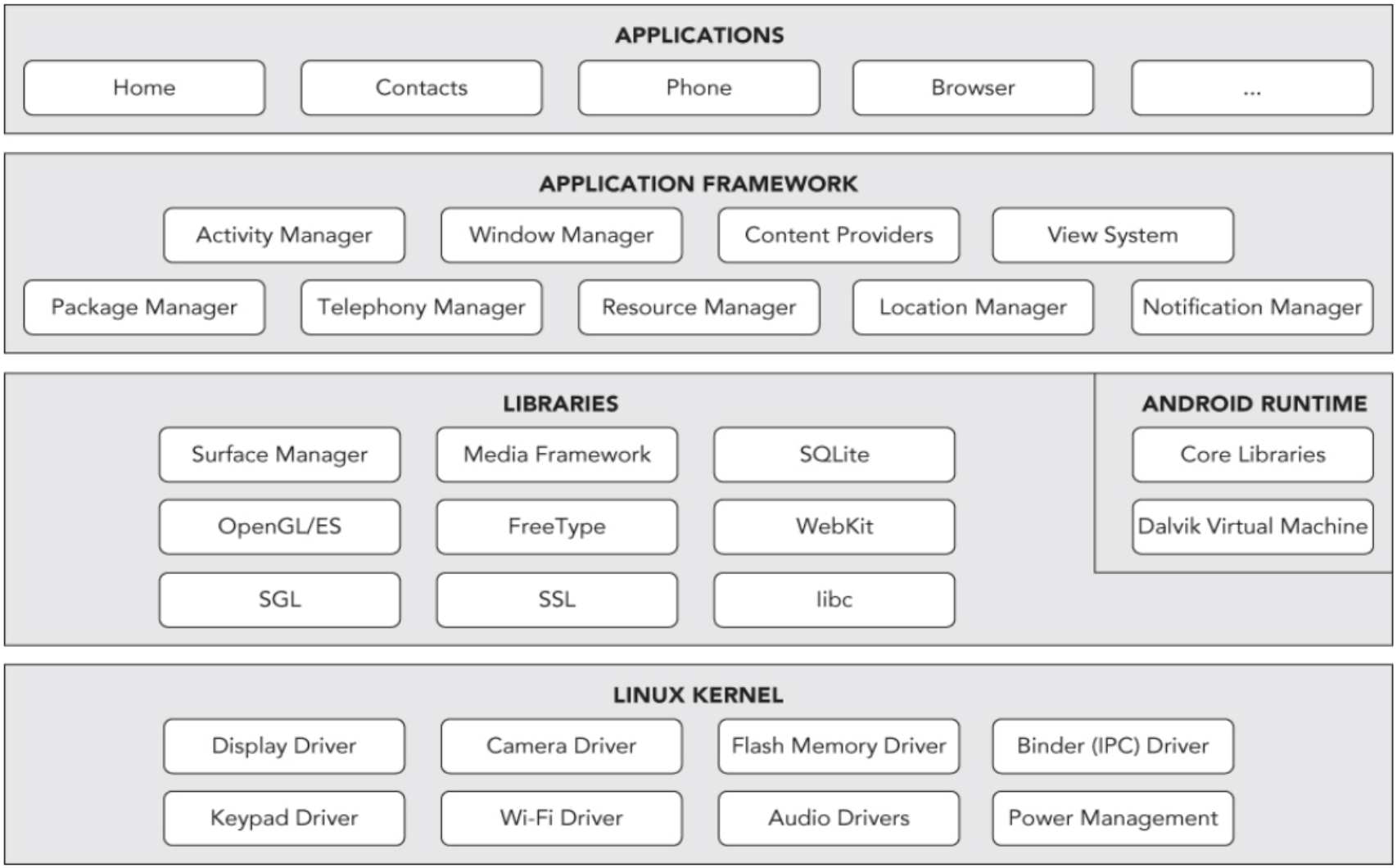
Summer 2, 2019 - CS 4520/CS5520 – Mobile Application Development

Pratheep Kumar Paranthaman, Ph.D.,

Today's topics

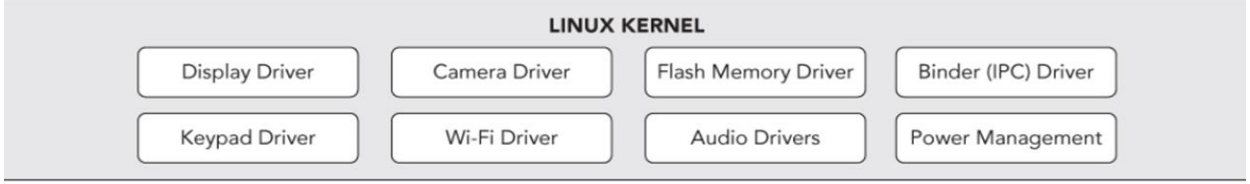
- Android Architecture
- Project structure
- User interaction

How Android works?



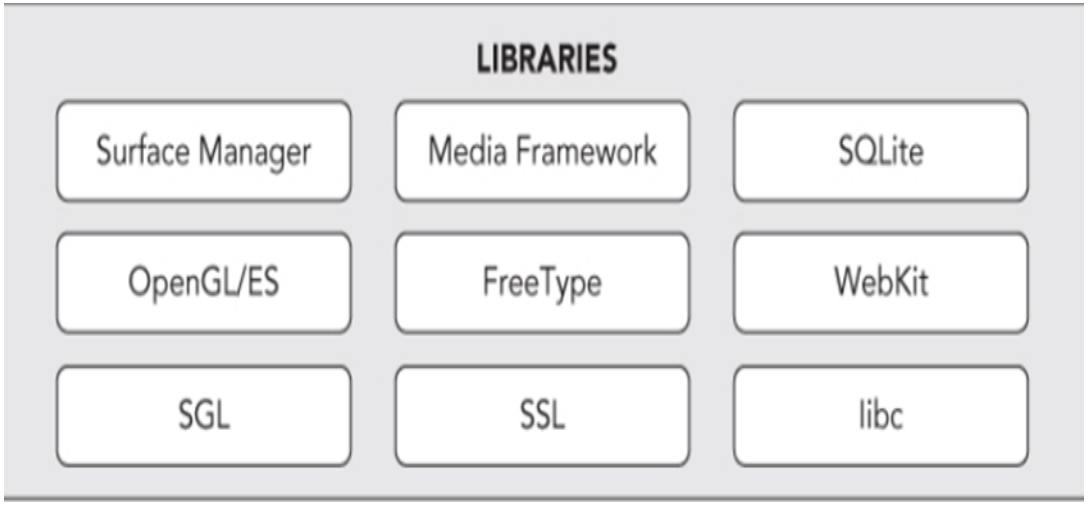
Linux Kernel

- Provides Hardware Abstraction layer
- Holds the low-level device drivers for various components(Example: keypad driver, wi-fi driver, camera driver, and lots more) in an Android device.
- Internally Android uses Linux for:
  - Device management
  - Memory management
  - Process management
  - Networking
  - OS services

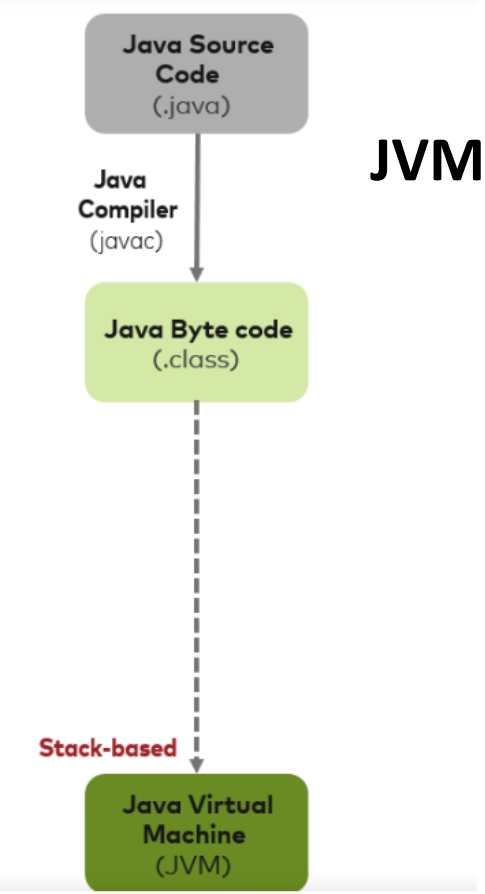


Native Libraries (Kernel -> Native Libraries)

- The shared libraries are written in C or C++
- Contains the units that provide the main features of an Android OS.
- Native libraries:
  - Surface Manager
  - 2D and 3D graphics
  - Media Codecs
  - SQL database
  - Browser Engine

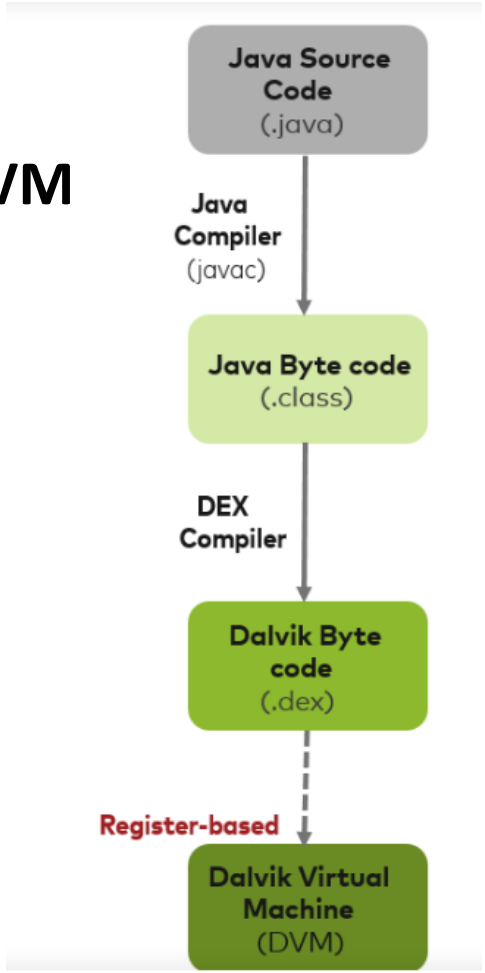


Runtime (Kernel -> Android Runtime)



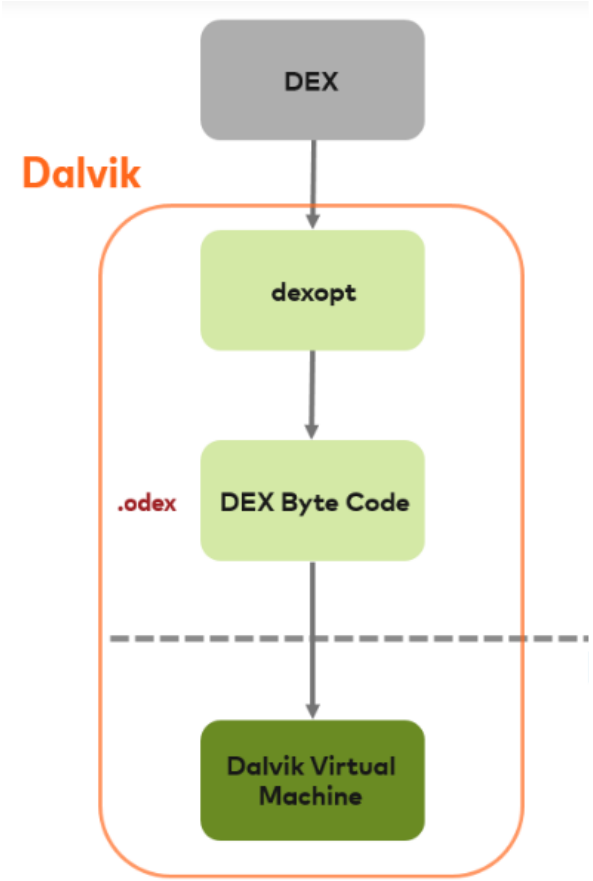
JVM

DVM



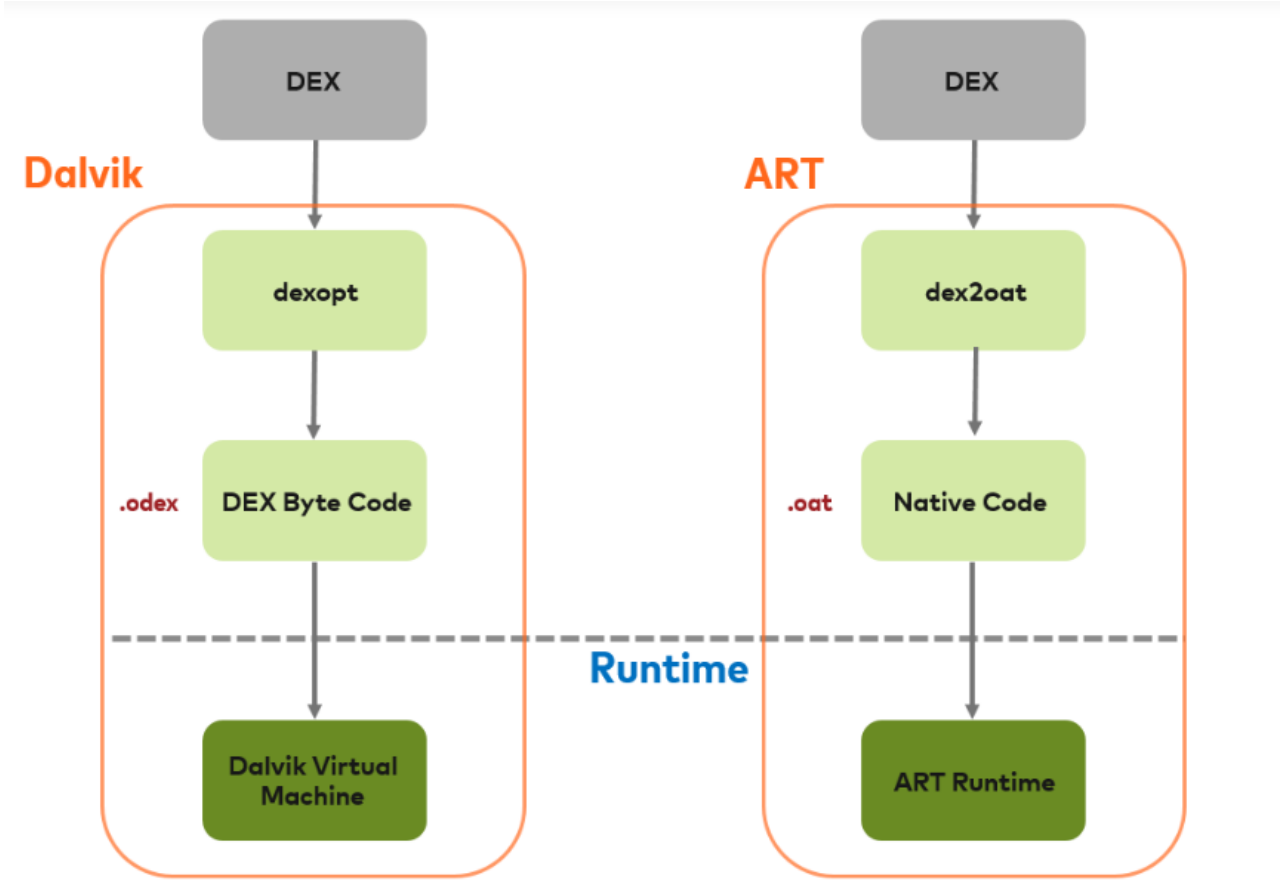
[android.jlelse.eu/closer-look-at-android-runtime-dvm-vs-art-1dc5240c3924]

Dalvik Vs. ART



[android.jlelse.eu/closer-look-at-android-runtime-dvm-vs-art-1dc5240c3924]

Dalvik Vs. ART



[android.jlelse.eu/closer-look-at-android-runtime-dvm-vs-art-1dc5240c3924]

Android Runtime ART (Kernel -> Android Runtime)

- Introduced in Android 4.4(KitKat)
- Replaced Dalvik in Android 5.0(Lollipop)
- Compiles an application into machine code, while it's installed onto your Android device.
- Makes programs run faster at the expense of a longer install time.

Long story Short!

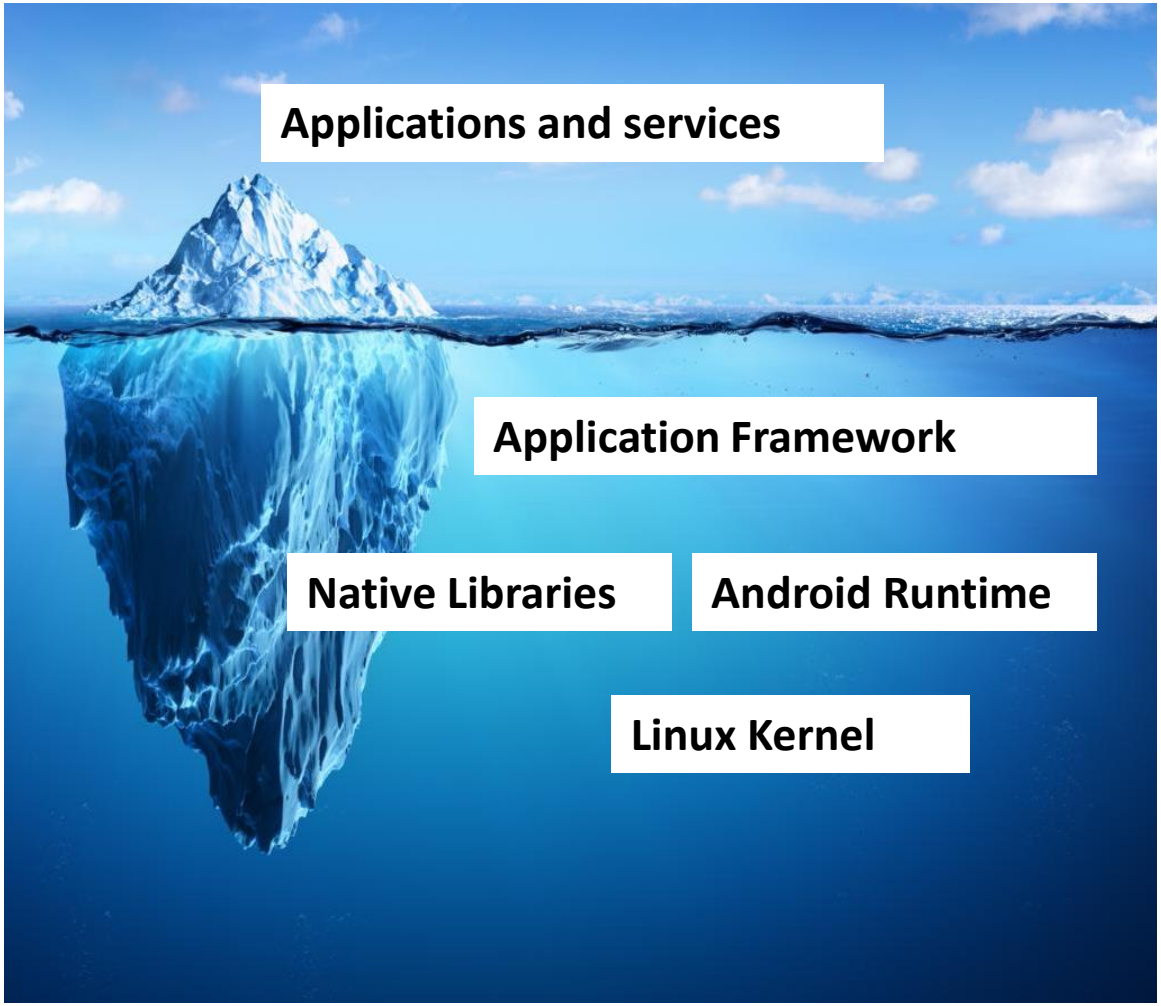
Code will be written in Java and run by Dalvik or ART

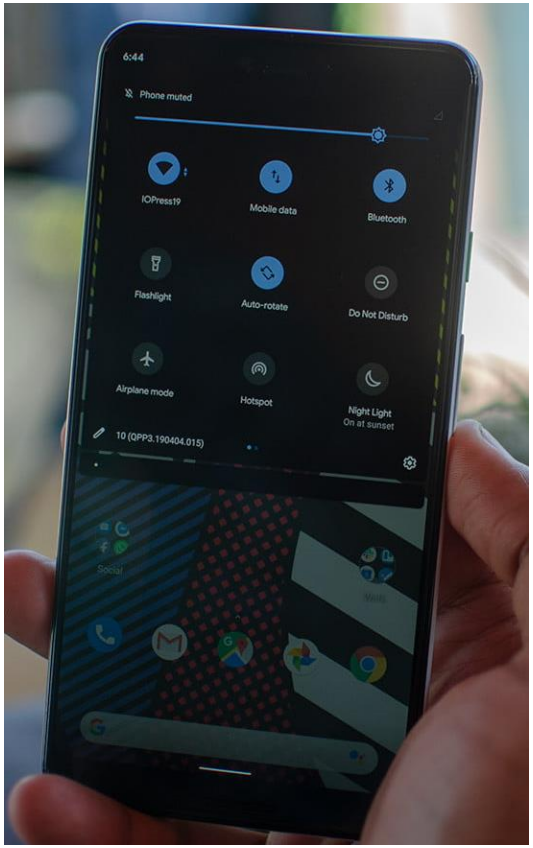
Application Framework

- Exposes various capabilities of Android OS to Application developers

- Application Framework features:
  - Activity Manager
  - Content providers
  - Resource manager
  - Location Manager
  - Notification Manager

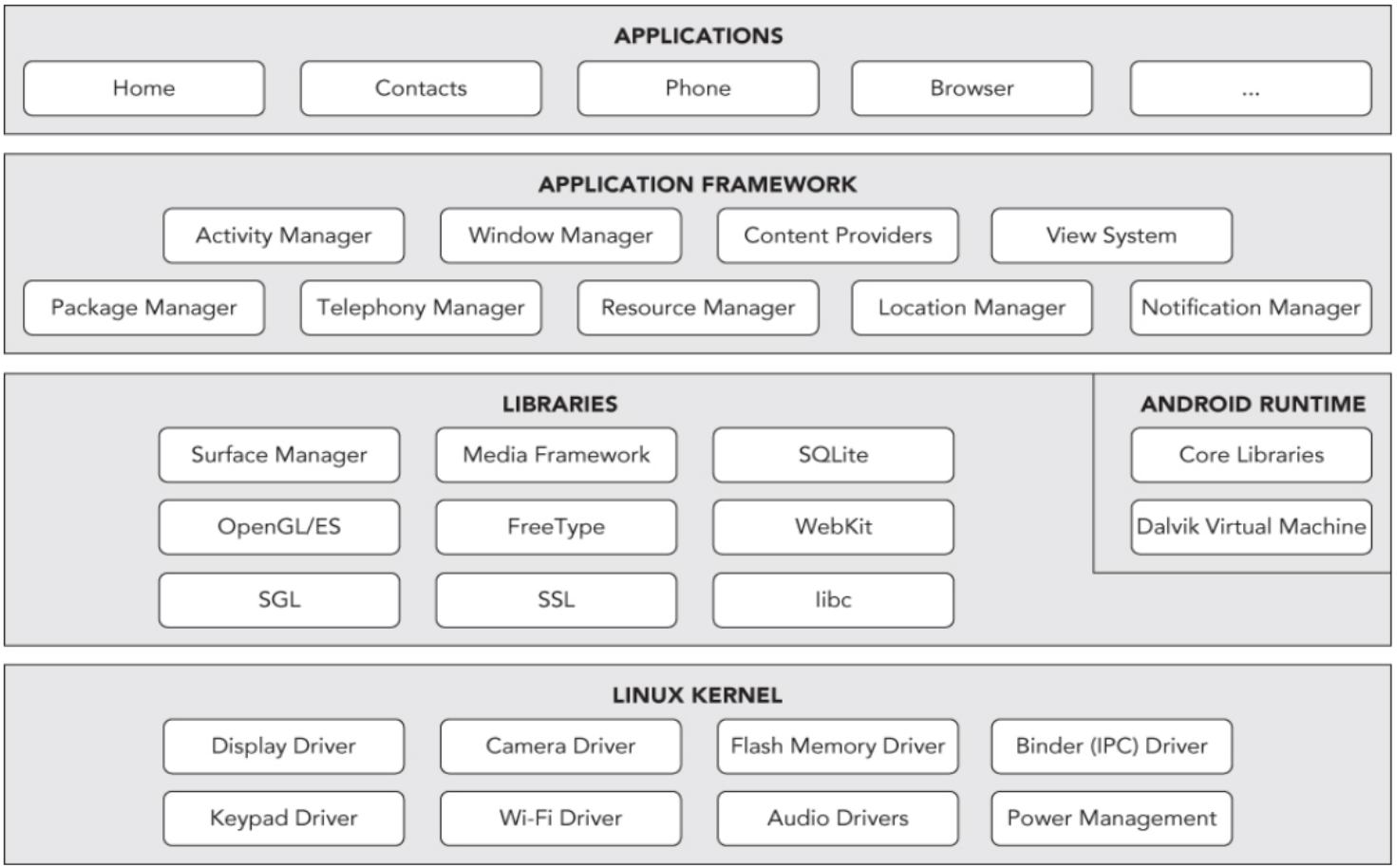
Applications and Services



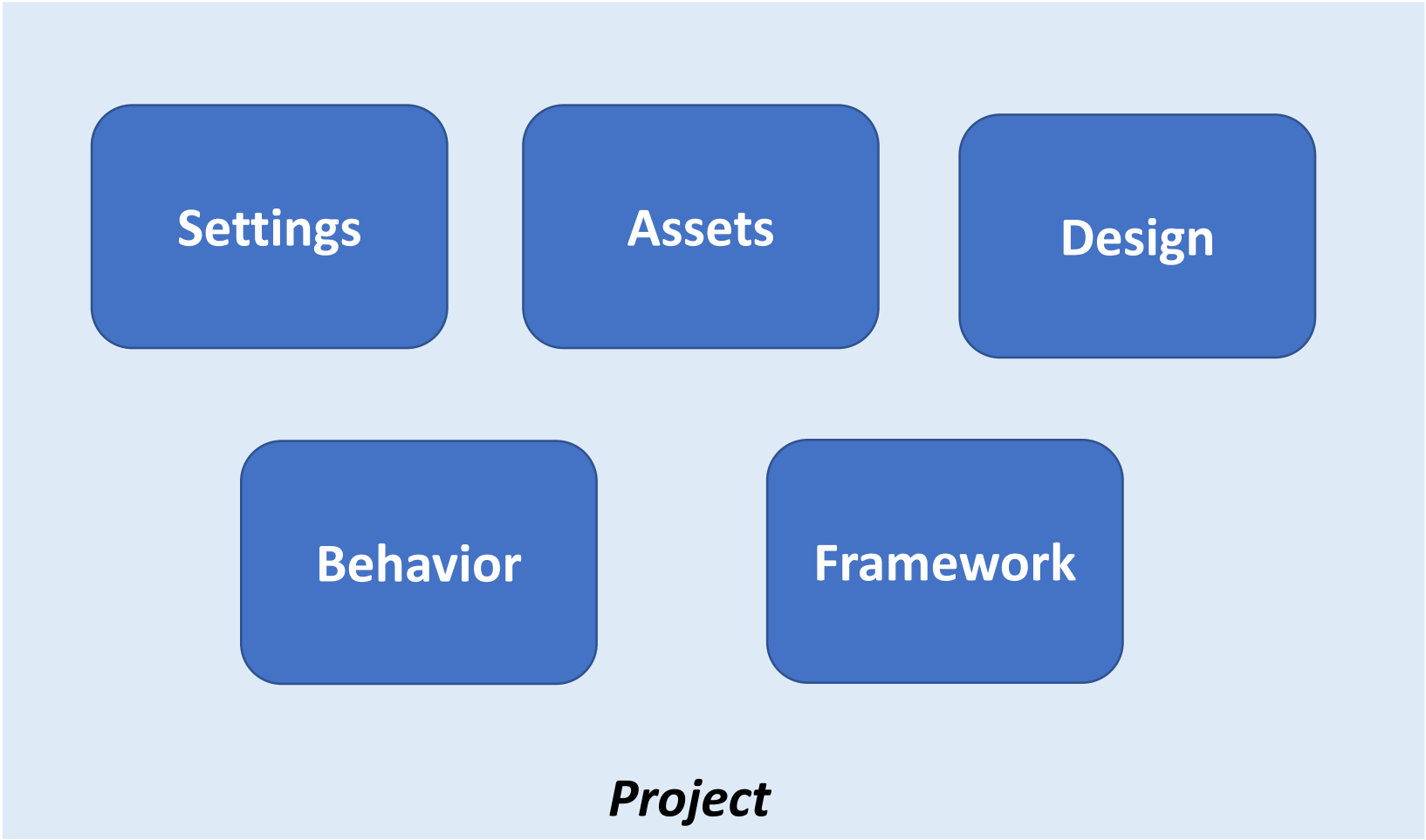


[digitaltrends.com/mobile/android-10-q-review/]

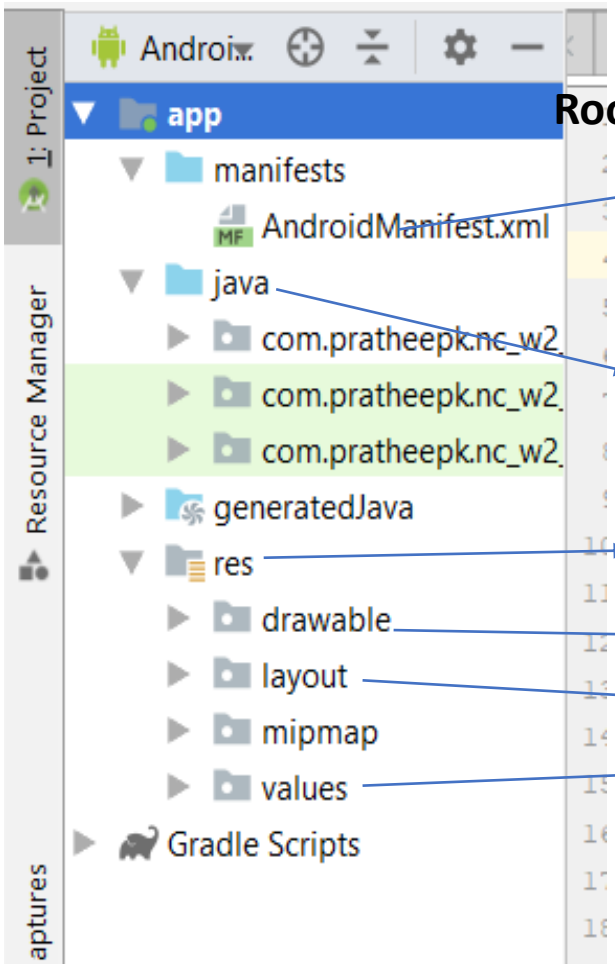
How Android works?



Android Project Structure



Android Project Structure



Root folder

Stores all information of your app – one of the crucial files

Stores all in the code that you write

All project resources

Images

App layouts with XML definitions

App Styling features(Themes)

Android Building blocks

Activity

Layout

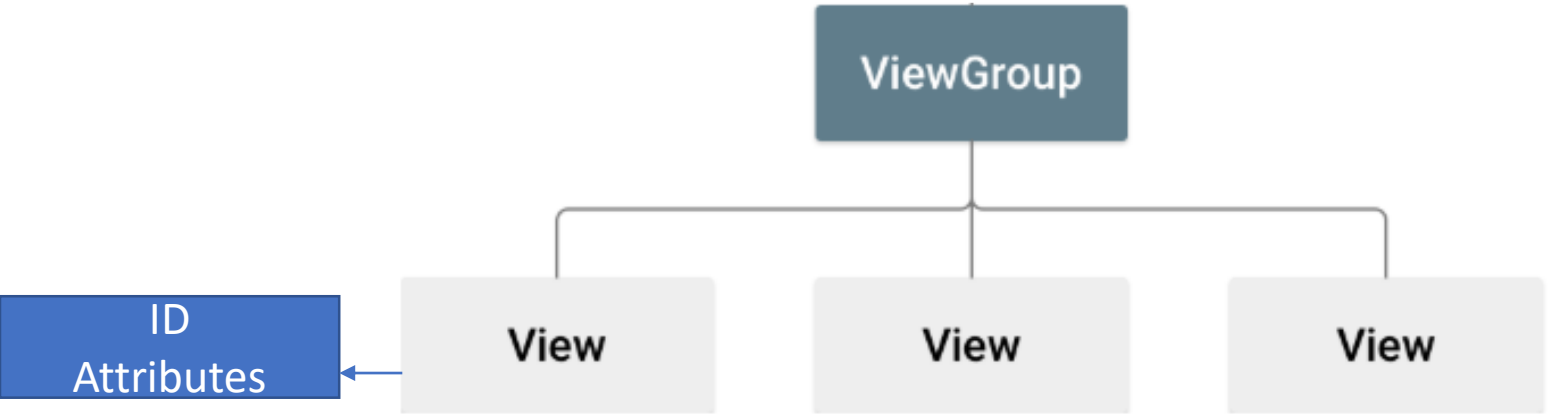
Resources

And lots more….

Layout

UI Layout with view Hierarchy

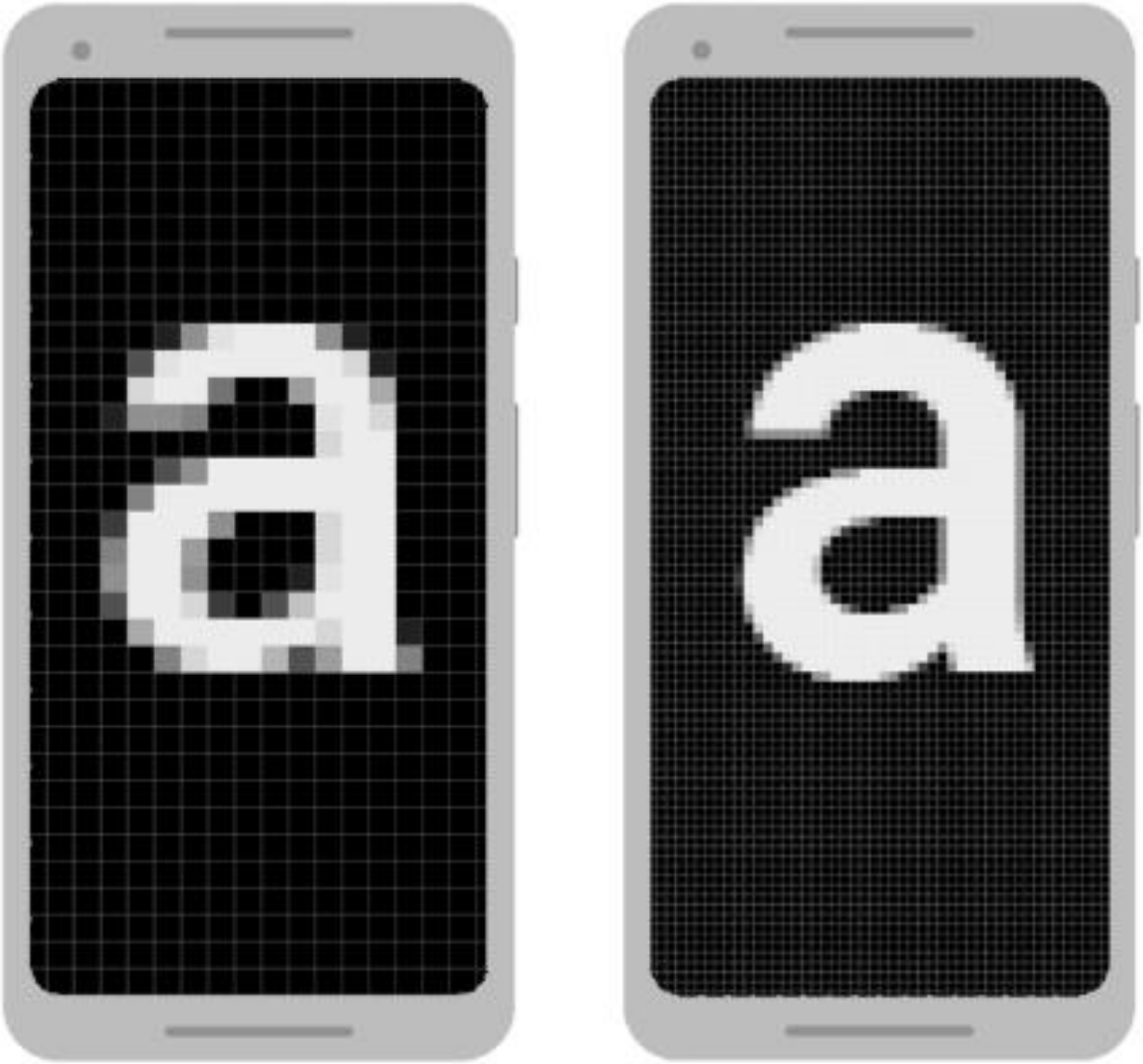
- Create them using the XML
- Instantiate during the runtime



[developer.android.com/guide/topics/ui/declaring-layout]

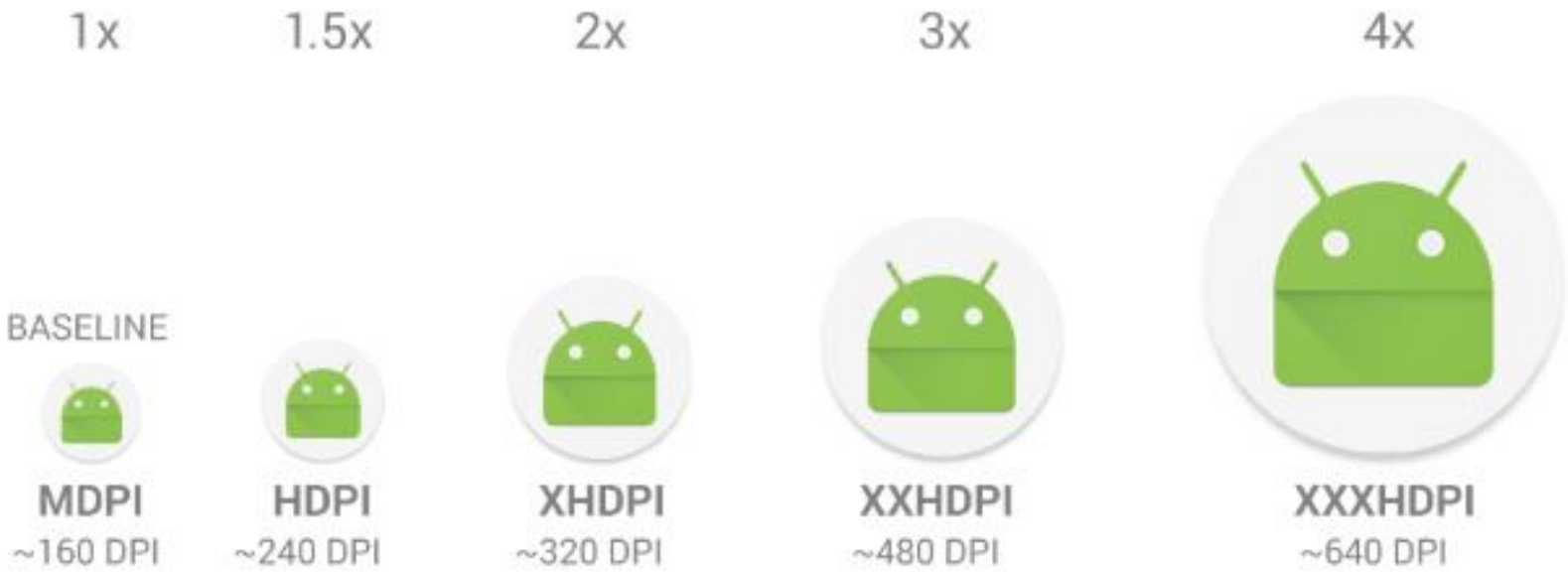
Activity

- Crucial part of user interaction
- Extends the AppCompatActivity

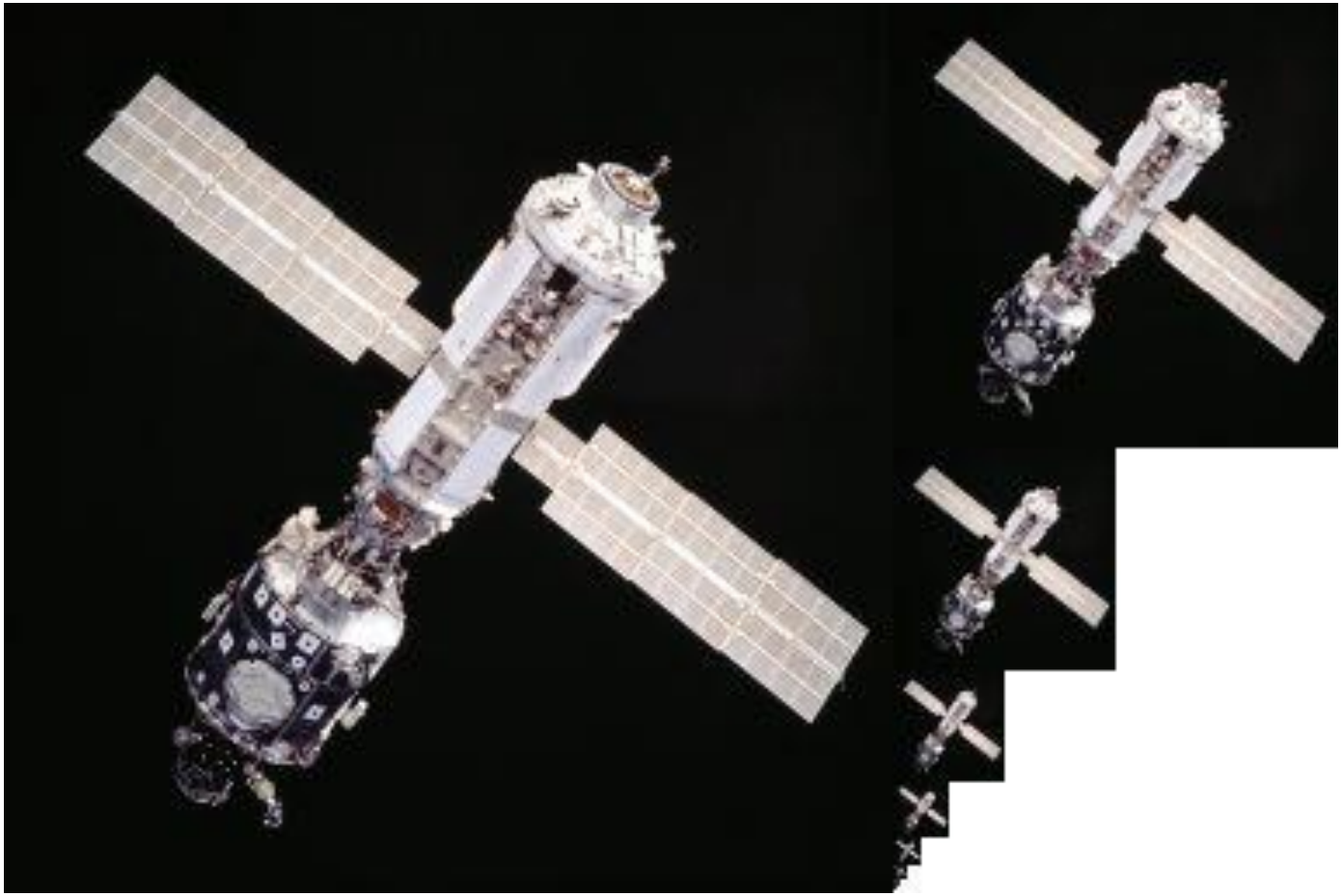


Mipmaps?? What are they

[developer.android.com/training/multiscreen/screendensities]

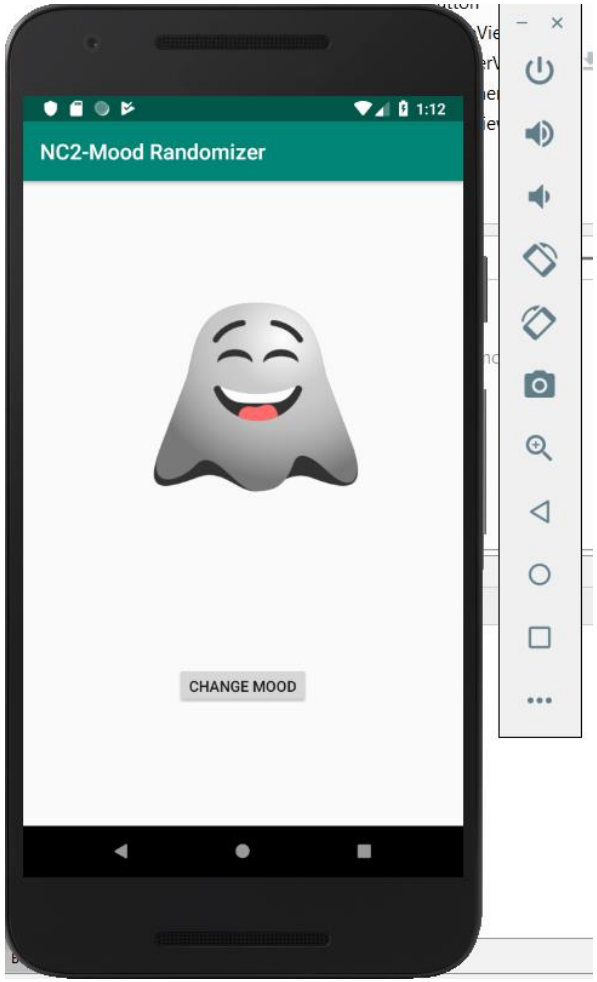

1x
1.5x
2x
3x
4x
BASELINE
MDPI
~160 DPI
HDPI
~240 DPI
XHDPI
~320 DPI
XXHDPI
~480 DPI
XXXHDPI
~640 DPI

Mipmaps



Let’s build an App!

- Mood randomizer
  - Updates the image based on user clicks
  - We’ll explore the following
    - Store external images
    - Handling events
    - Updating UI on run time



In class exercise – 1 (Mood Randomizer)

- Create a project with an EmptyActivity
- Download the Ghost Mood images from Blackboard (**Course Material -> Course Documents -> Media Files for in-class exercises**)
- Import the images(**5 png files**) to your Android Studio Project.
  - Drag and drop the images to **res -> Drawable in your project**

In class exercise – 1 (Mood Randomizer)

- **Layout**
- Open the layout.xml and insert a **VideoView** and **Button** Component in it
  - Set an id for the VideoView Component on the Layout.
  - Set an id for the button.

Activity

- Create an Array of type ***int (Resource ID)*** and store the 5 images of the drawable folder in it.
- Include an OnClick Event for the button and include the code for selecting a random index in the array
- Grab the Imageview component and set the randomized resource ID by using

ImageView.setImageResource(array[randomNumber]);