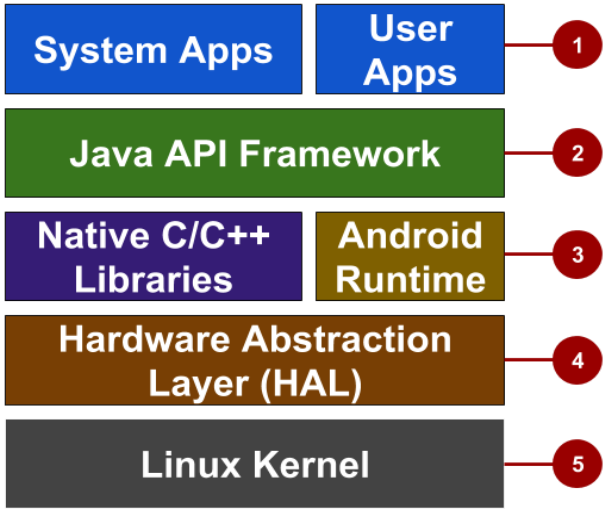
Android Platform Architecture

Android stack

https://developer.android.com/guide/platform

1. System and user apps
2. Android OS API Java framework
3. Expose native APIs; run apps
4. Expose device hardware capabilities
5. Linux Kernel

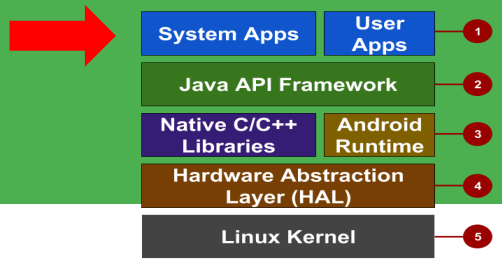


Layers

These slides are partially based on the material that Google provides for the course ***Android Developer Fundamentals***

https://developer.android.com/courses/fundamentals-training/overview-v2

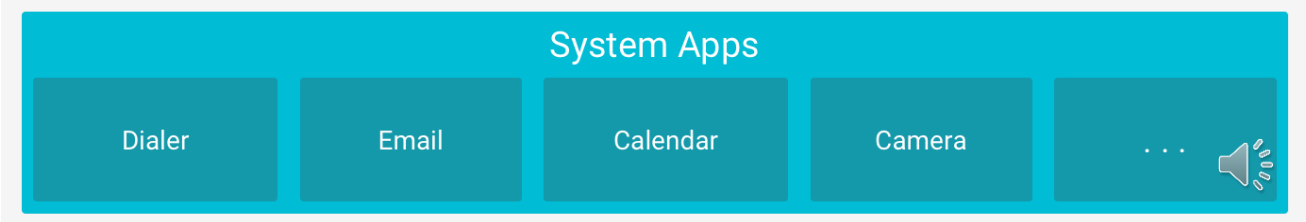
System and user apps



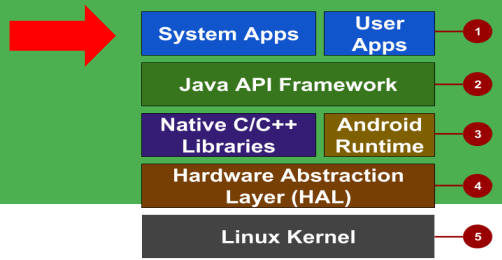
- System apps have ***no special status***
- System apps ***provide key capabilities*** to app developers
  - email, SMS messaging, calendars, internet browsing, contacts

Example:

Your app can use a system app to deliver a SMS message



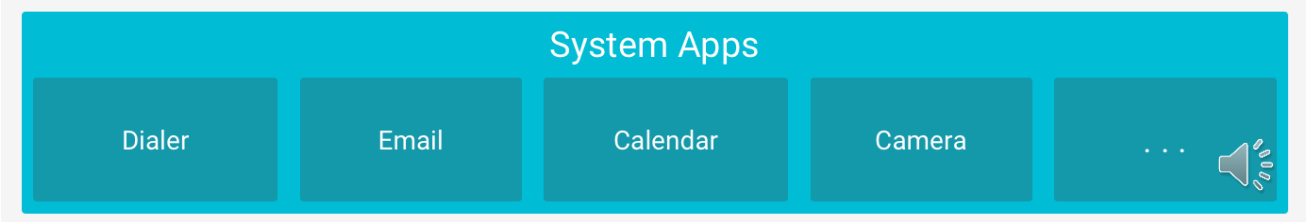
System and user apps



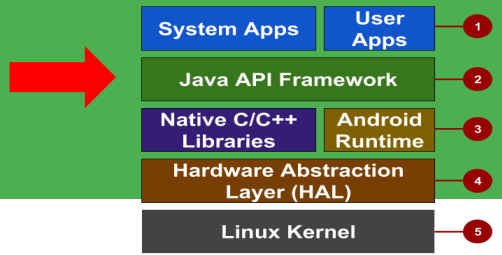
- System apps have ***no special status***
- System apps ***provide key capabilities*** to app developers
  - email, SMS messaging, calendars, internet browsing, contacts

Example:

Your app can use a system app to deliver a SMS message

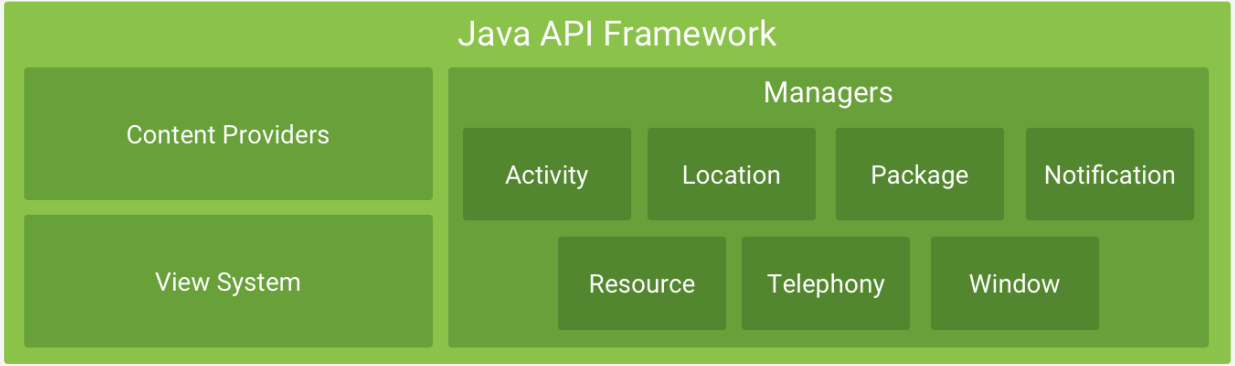


Java API Framework

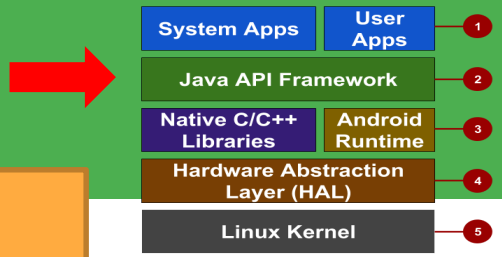


The entire **feature**-set of the **Android OS** is **available** to you through APIs written in the Java language.

- **View system:** to create UI screens
- **Notification manager**: to display custom alerts in the status bar
- **Activity manager**: for app life cycles and navigation
- ....

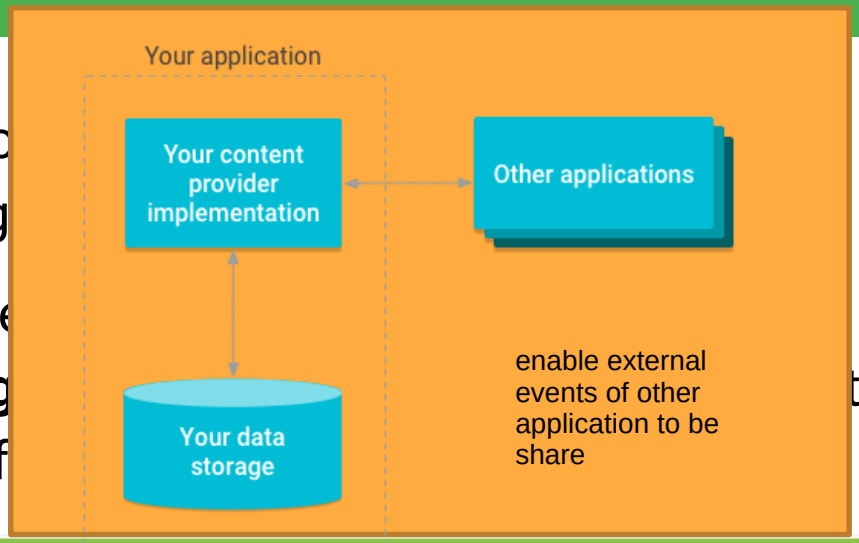


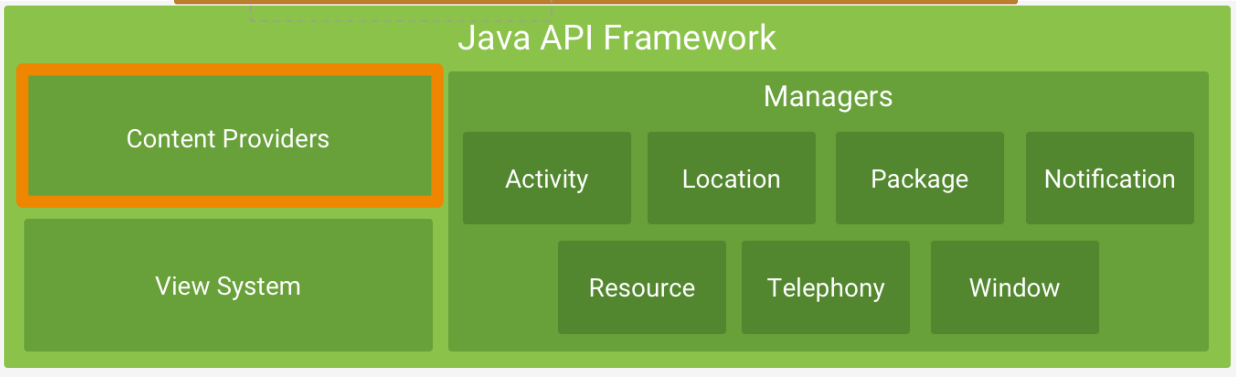
Java API Framework



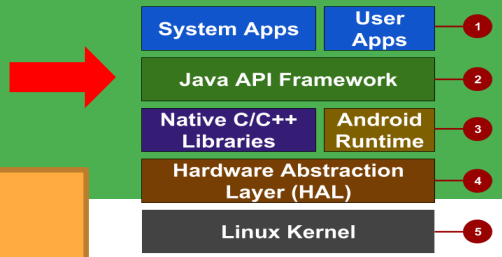
The entire **feature**-set c through APIs written in the Java lang

- **View system:** to cre
- **Notification manag** tatus bar
- **Activity manager**: f
- ....



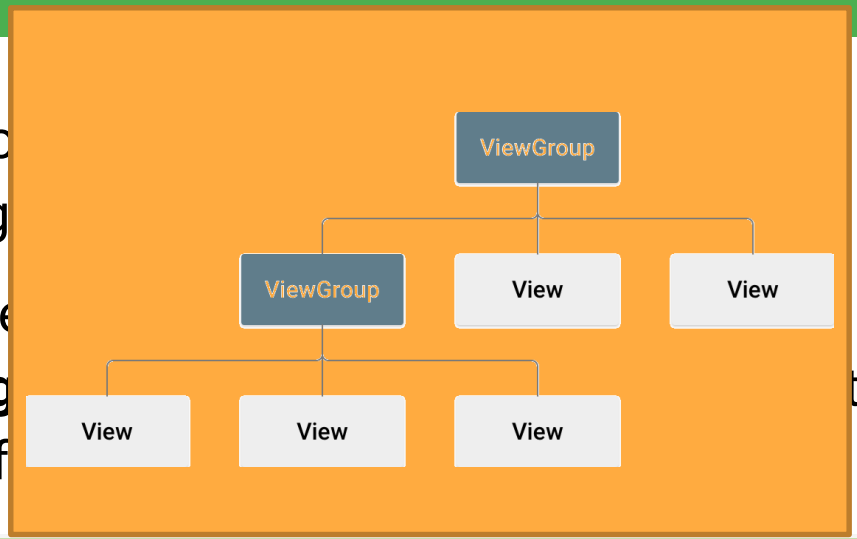


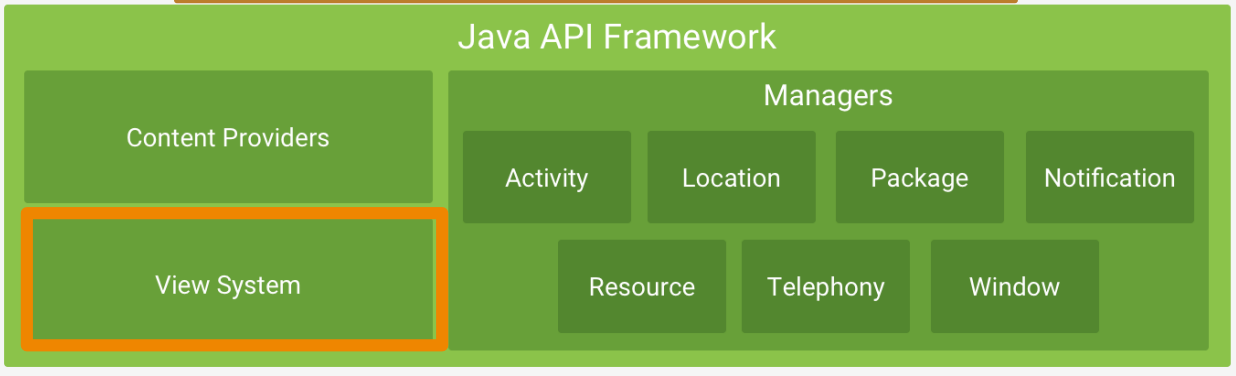
Java API Framework



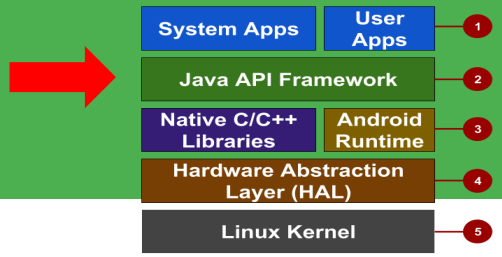
The entire **feature**-set c through APIs written in the Java lang

- **View system:** to cre
- **Notification manag** tatus bar
- **Activity manager**: f
- ....



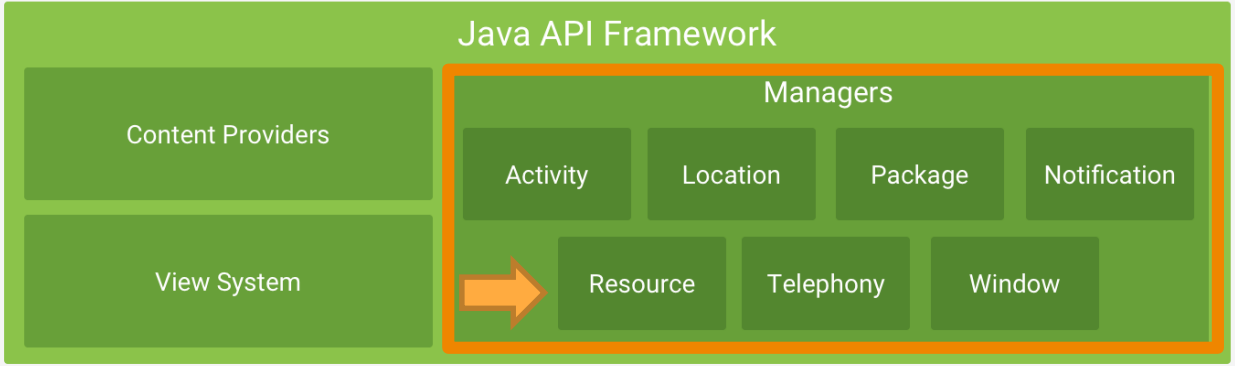


Java API Framework



The entire **feature**-set of the **Android OS** is **available** to you through APIs written in the Java language.

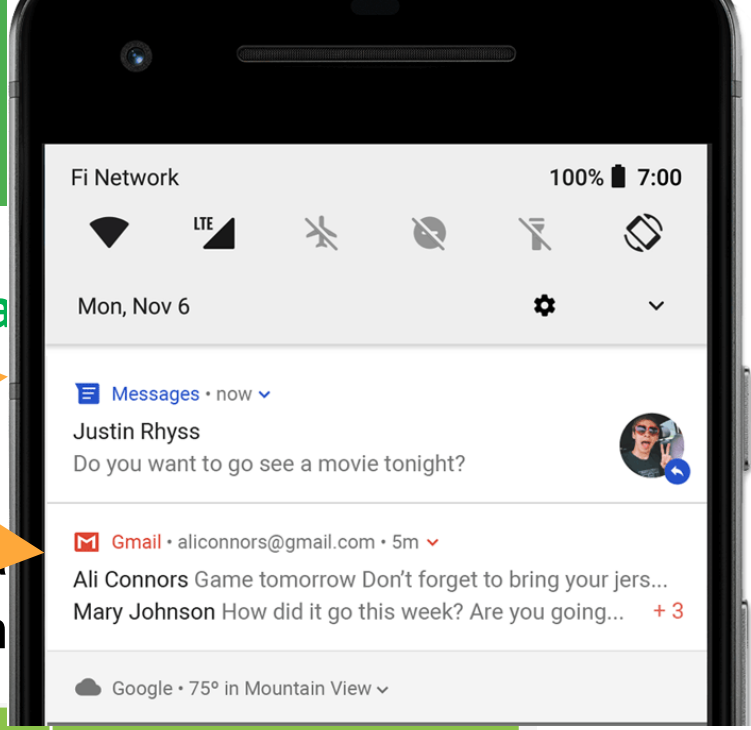
- **View system:** to create UI screens
- **Notification manager**: to display custom alerts in the status bar
- **Activity manager**: for app life cycles and navigation
- ....

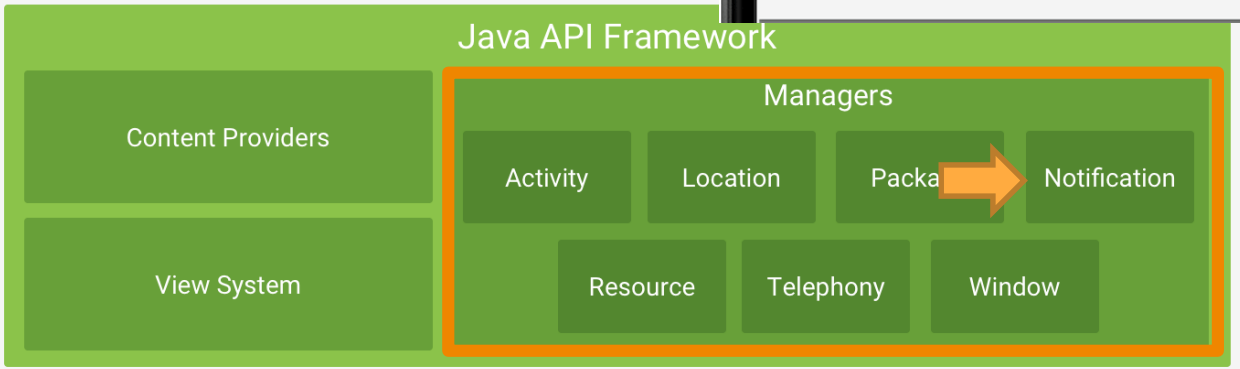


Java API Framework

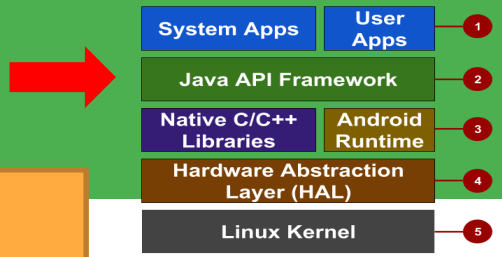
The entire f… written in th…

- **View sy…**
- **Notification manager**: to display custom a…
- **Activity manager**: for app life cycles and n…
- ….



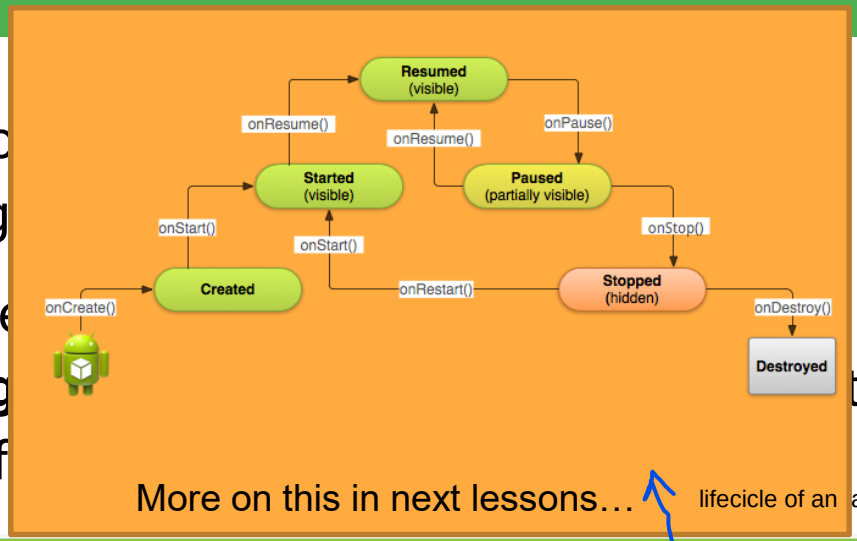


Java API Framework



The entire **feature**-set c… through APIs written in the Java lang…

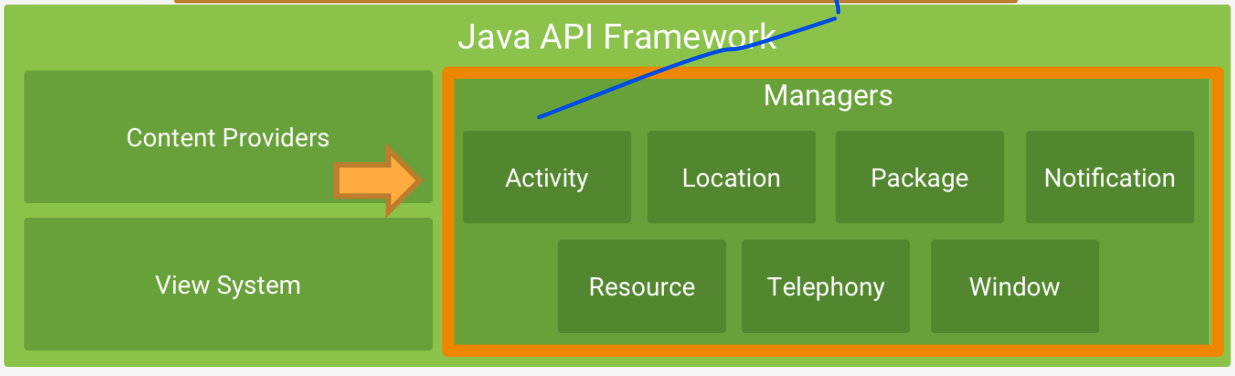
- **View system:** to cre…
- **Notification manag…** tatus bar
- **Activity manager**: f…
- ….



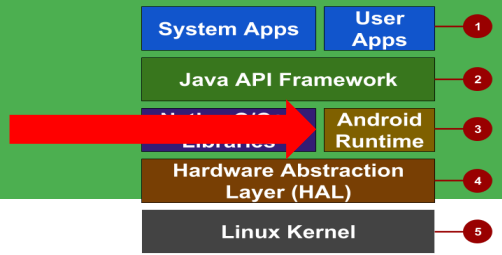
activities move in different case depending on which time is in it lifecicle

More on this in next lessons…

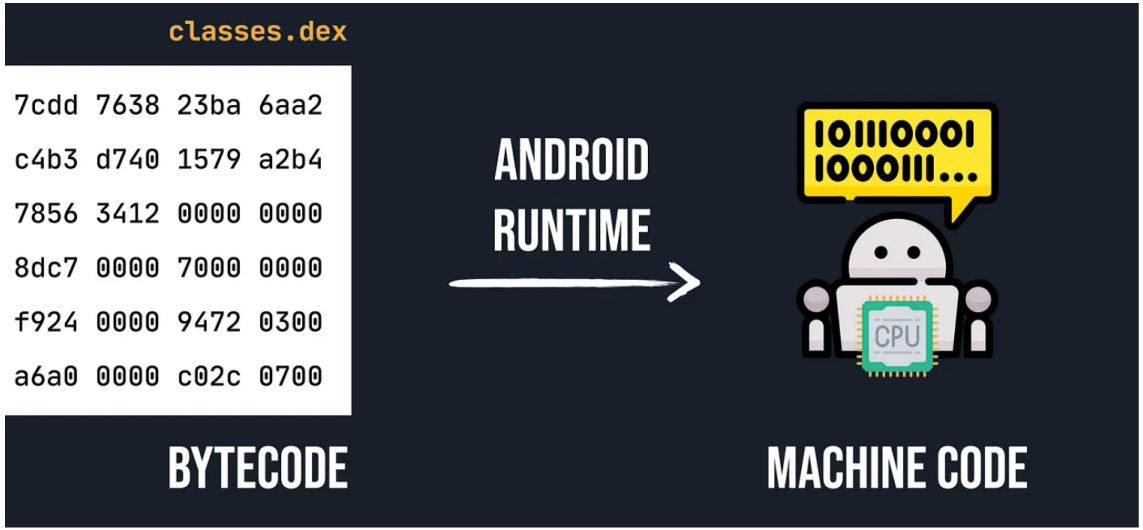
lifecicle of an activity

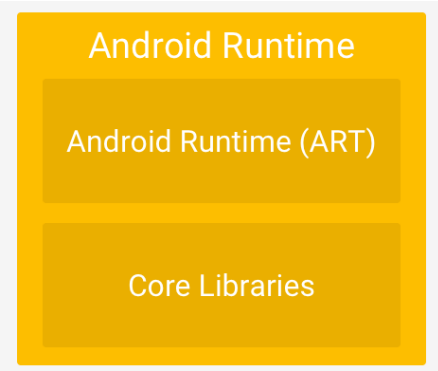


Android runtime



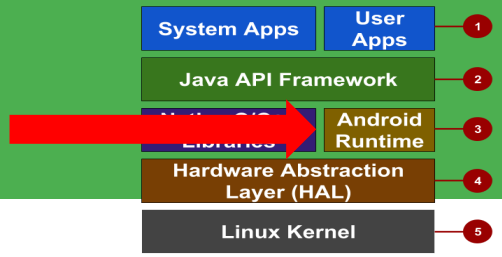
When we build our app and generate APK, part of that APK are **.dex files**.





When a user runs our app the bytecode written in **.dex files is translated by Android Runtime into the machine code** which is a set of instruction that can be processed by the CPU that we have in our machine

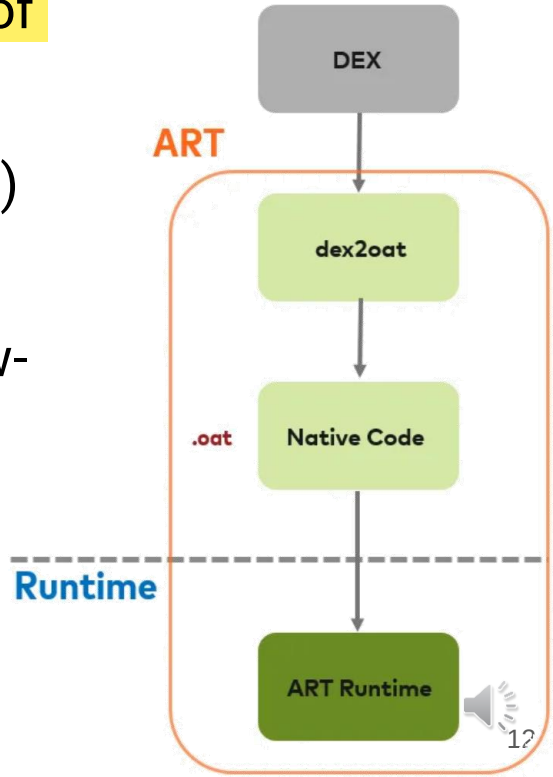
Android runtime



Each app runs in ***its own process*** with its own instance of the Android Runtime (ART)

- For devices running Android version 5.0 (API level 21) or higher
- ART is written to run multiple virtual machines on low-memory devices

by excecuting those text file and this is the main feature because this make occupied a little bit of space

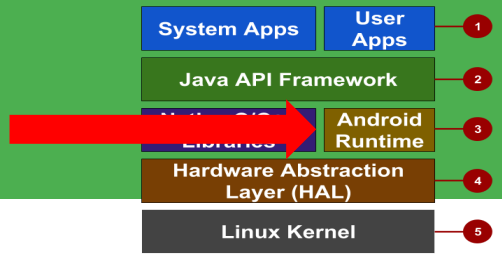


Just an overview, more precise info here:

https://source.android.com/docs/core/runtime

https://source.android.com/docs/core/runtime/configure

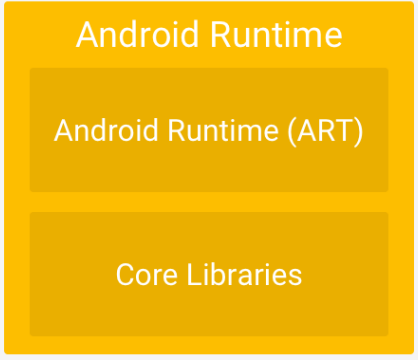
Android runtime



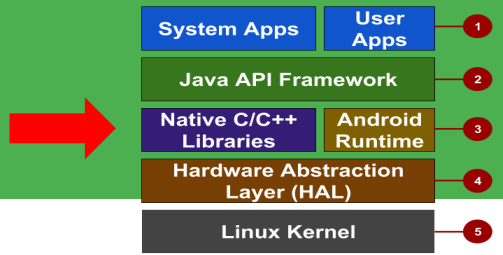
Each app runs in ***its own process*** with its own instance of the Android Runtime (ART)

- For devices running Android version 5.0 (API level 21) or higher
- ART is written to run multiple virtual machines on low-memory devices

Android also includes a set of **core runtime libraries** that provide most of the functionality of the Java programming language But those isn't the only libreries that we have

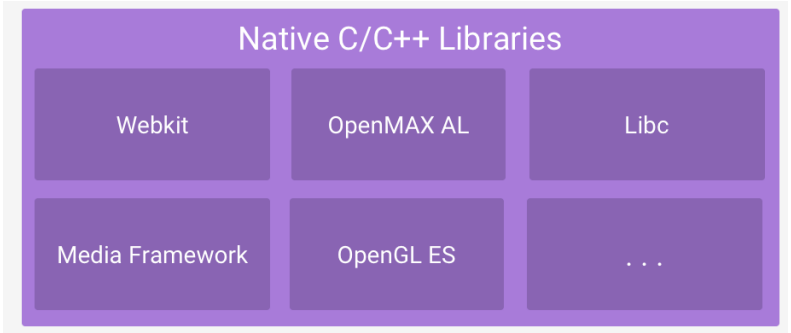


C/C++ libraries

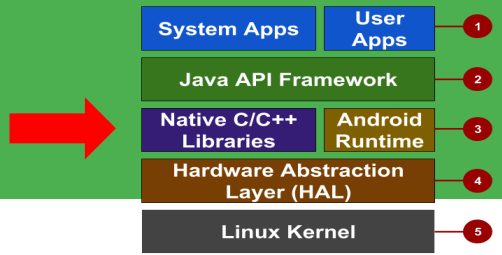


Core C/C++ Libraries give **access** to **core native Android system** components and services

For example, **it is possible to access OpenGL ES** through the Android framework's Java OpenGL API ***to add support for drawing and manipulating 2D and 3D graphics*** in your app

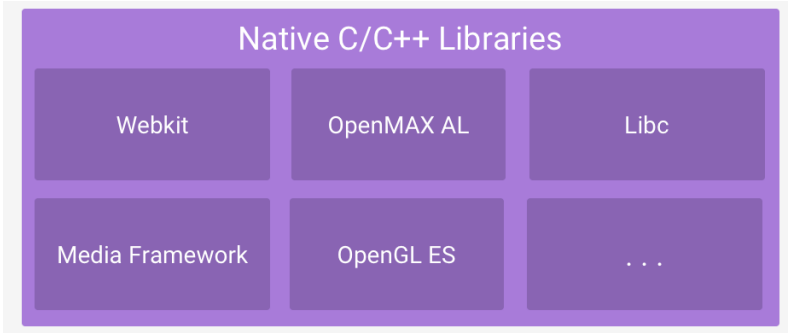


C/C++ libraries

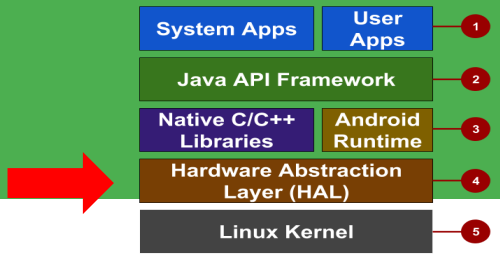


Core C/C++ Libraries give **access** to **core native Android system** components and services

For example, **it is possible to access OpenGL ES** through the Android framework's Java OpenGL API ***to add support for drawing and manipulating 2D and 3D graphics*** in your app



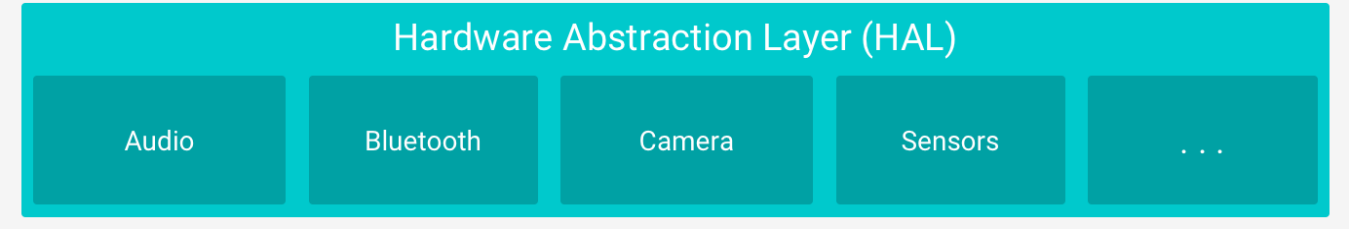
Hardware Abstraction Layer



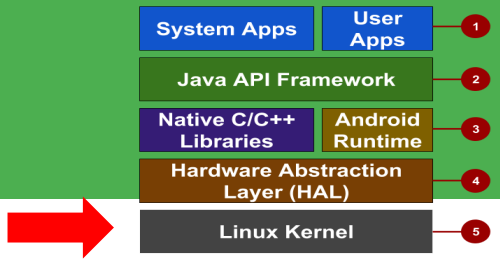
- **Standard interfaces** that **expose** device **hardware capabilities** as libraries

Examples: Camera, Bluetooth module, sensors ….

- when a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component



Linux Kernel



The **foundation** of the **Android platform** is the **Linux kernel.** Features include:

- **Threading** and **low-level memory management**
  - The Android Runtime (**ART**) relies on the Linux kernel for underlying functionalities such as threading and low-level memory management
- **Security**
  - Using a Linux kernel allows Android to take advantage of **key security features**
- **Drivers**
  - Using a Linux kernel allows device manufacturers to **develop hardware drivers for a well-known kernel**