



Android Architecture

Nguyen Tran (Nguyen.TranLeHoang@vn.bosch.com)

Sep 27, 2024

Day 1

Group working



QUIZ (1/10)



QUIZ (2/10)



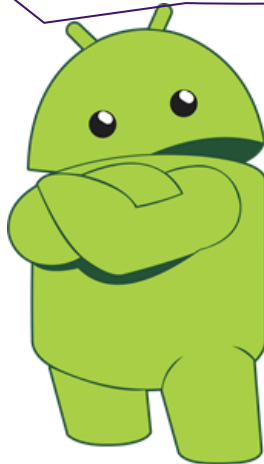
QUIZ (3/10)



QUIZ (4/10)



Which language should be used in Android application? And why?



QUIZ (5/10)



QUIZ (6/10)



Which strategy did Google use to promote Android?



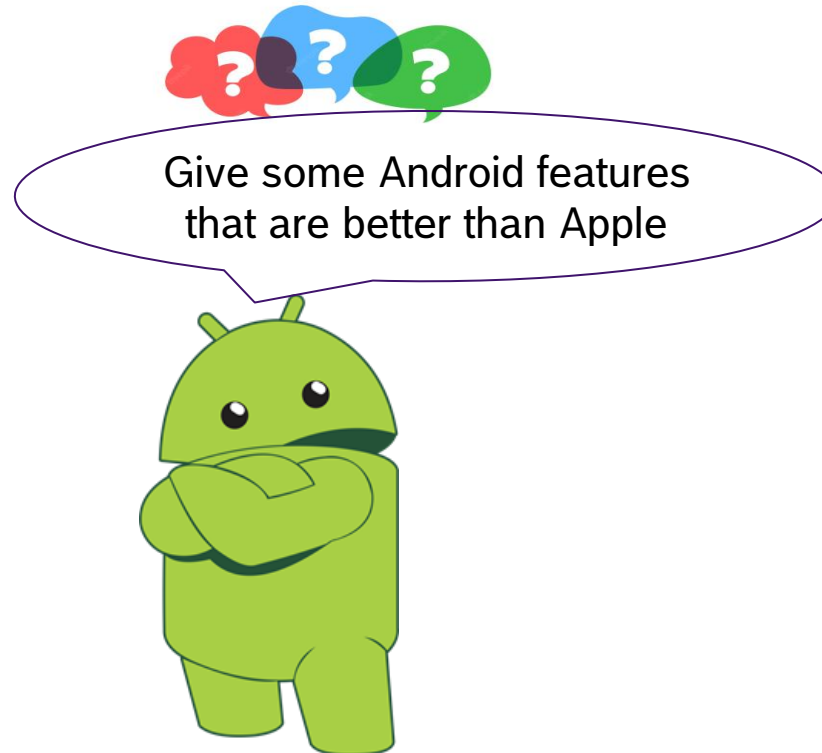
QUIZ (7/10)



Please give some benefits to
learn/develop Android platform?



QUIZ (8/10)



QUIZ (9/10)



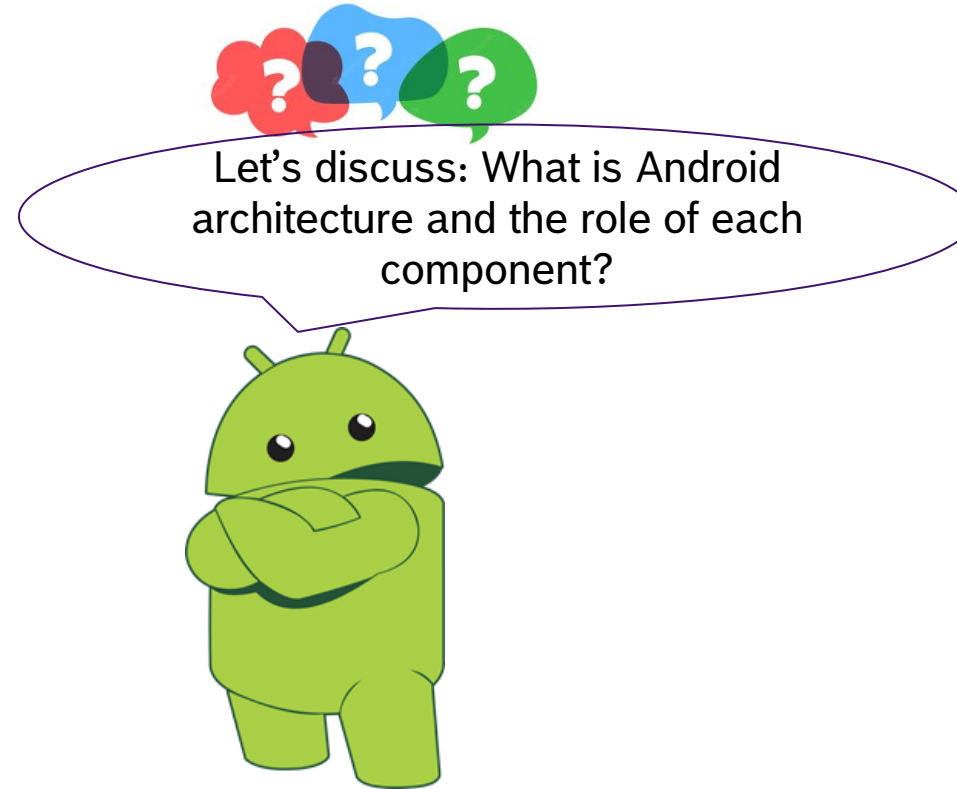
QUIZ (10/10)



What is API level? What will happen if an app having greater API level than device is installed?



Group working



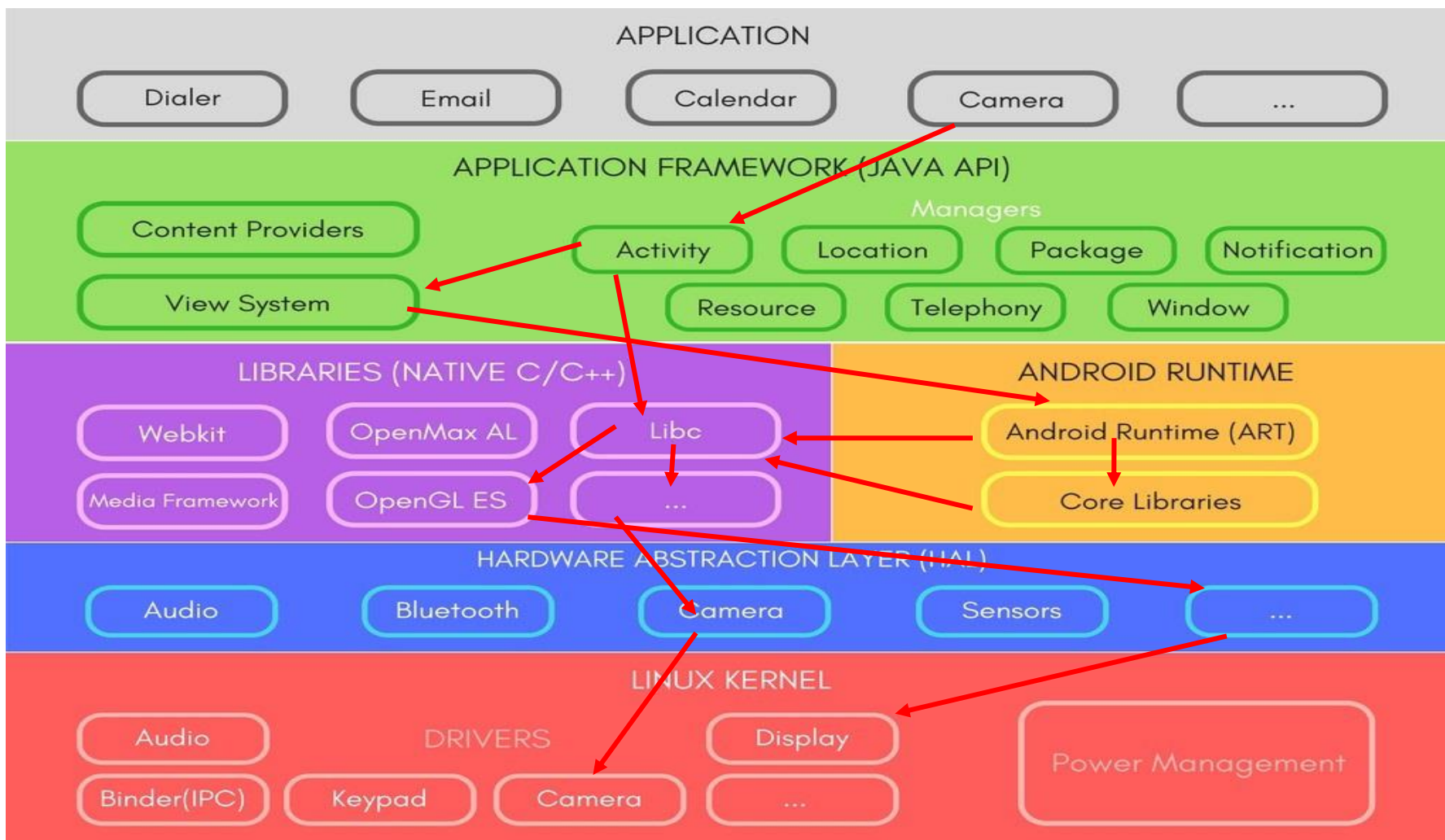
Agenda

- **Architecture**
 - **Overview**
 - **Application**
 - **Application Framework**
 - **Libraries and Android Runtime**
 - **HAL**
 - **Kernel**

Agenda

- **Architecture**
 - **Overview**
 - Application
 - Application Framework
 - Libraries and Android Runtime
 - HAL
 - Kernel

Overview



Agenda

- **Architecture**

- Overview
- **Application**
- Application Framework
- Libraries and Android Runtime
- HAL
- Kernel

Application

- Android comes with a set of core apps for email, SMS messaging, calendars, internet browsing, contacts, etc.
- Apps are included with the platform, but third-party apps can become the user's default.
- This is the place where an application is installed and used.
- Android app developers will develop their application at this layer.



Application

- There are 2 programming languages for Android developers: Java and Kotlin.
- However, Kotlin is recommended for Android app development since May 7th, 2019 by Google.



Application

- Every Android application runs in its own process. Whenever there's a request that should be handled by a particular component, Android will:
 - make sure that the application process of the component is running.
 - starting it if necessary.
 - if an appropriate instance of the component is available, launch it.



Application

- A Linux process encapsulating an Android application is created for the application when some of its code needs to be run, and will remain running until:
 - it is no longer needed, OR
 - the system needs to reclaim its memory for use by other applications.



Application

- An unusual and fundamental feature of Android is that an application process's lifetime is not directly controlled by the application itself. Instead, it is determined by the system through a combination of:
 - the parts of the application that the system knows are running.
 - how important these things are to the user, and
 - how much overall memory is available in the system.

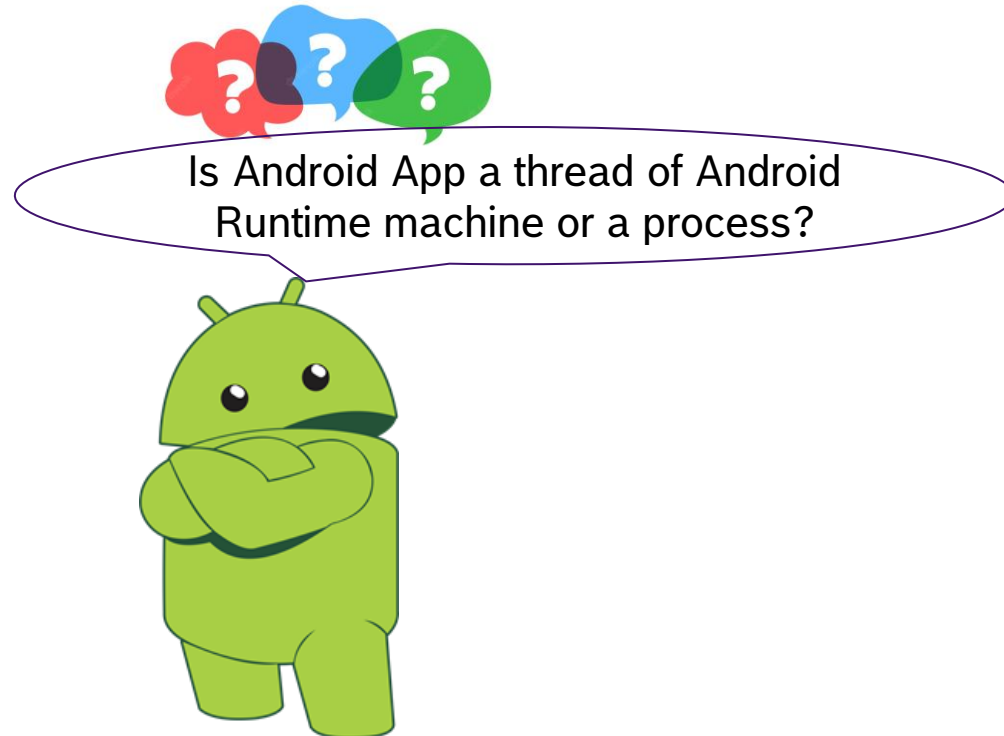


Day 2

Group working



QUIZ (1/5)



QUIZ (2/5)



QUIZ (3/5)



When “Back” button is pressed,
did Android app exit? Why?



QUIZ (4/5)



What will happen if Android
is out of memory?



QUIZ (5/5)



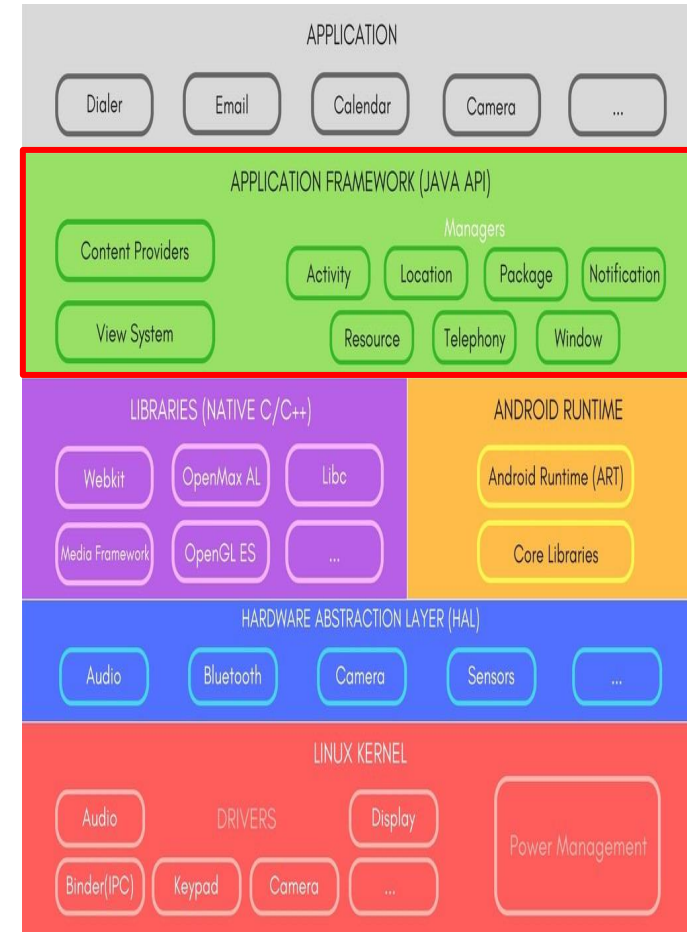
Agenda

- **Architecture**

- Overview
- Application
- **Application Framework**
- Libraries and Android Runtime
- HAL
- Kernel

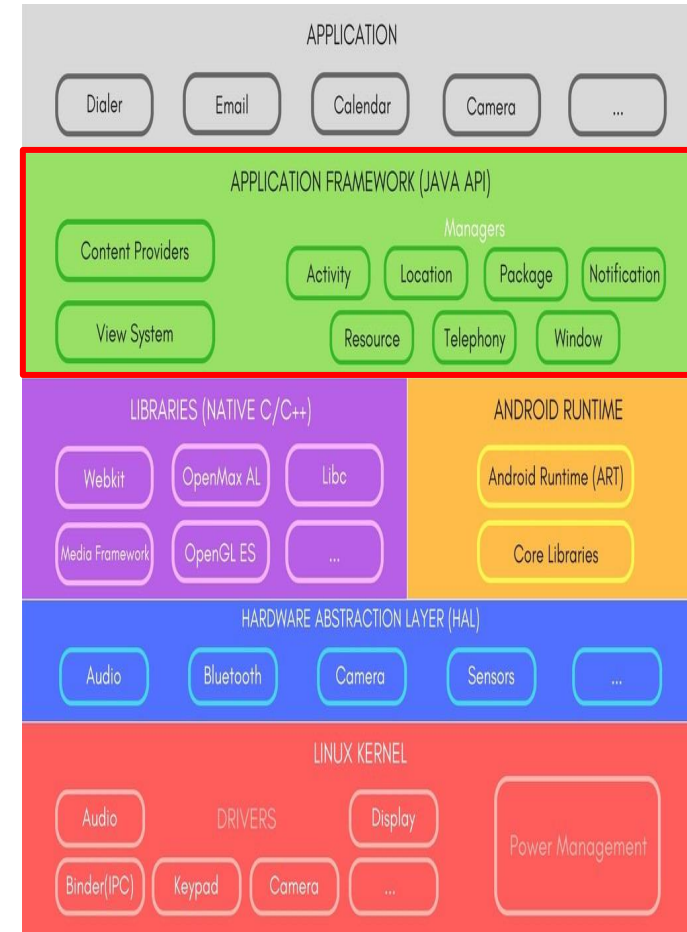
Application Framework

- The Application Framework layer provides many higher-level services to applications in the form of Java classes.
- Application developers are allowed to make use of these services in their applications.



Application Framework

- Through the use of Java package naming, Android frameworks are divided into separate namespaces, according to their functionality.
- Packages in the android.* namespace are available for use by developers.
- Packages in com.android.* are internal.
- Android also supports most of the standard Java runtime packages in the java.* namespace.



Application Framework

Package Name	API	Contents
android.app	1	Application Support
android.content		Content providers
android.database		Database support, mostly SQLite
android.graphics		Graphics support
android.opengl		OpenGL Graphics support
android.hardware		Camera, input and sensor support
android.location		Location support
android.media		Media support
android.net		Network support built over java.net APIs
android.os		Core OS Service and IPC support
android.provider		Built-in Android content-providers
android.sax		SAX XML Parsers
android.telephony		Core Telephony support
android.text		Text rendering
android.view		UI Components (similar to iOS's UIView)
android.webkit		Webkit browser controls
android.widget		Application widgets
android.speech	3	Speech recognition and Speech-to-Text
android.accounts	4	Support for account management and authentication.
android.gesture		Custom gesture support
android.accounts	5	User account support
android.bluetooth		Bluetooth support
android.media.audiofx	9 (G)	Audio Effects support
android.net.sip		Support for VoIP using the Session Initiation Protocol (RFC3261)
android.os.storage		Support for Opaque Binary Blobs (OBB)



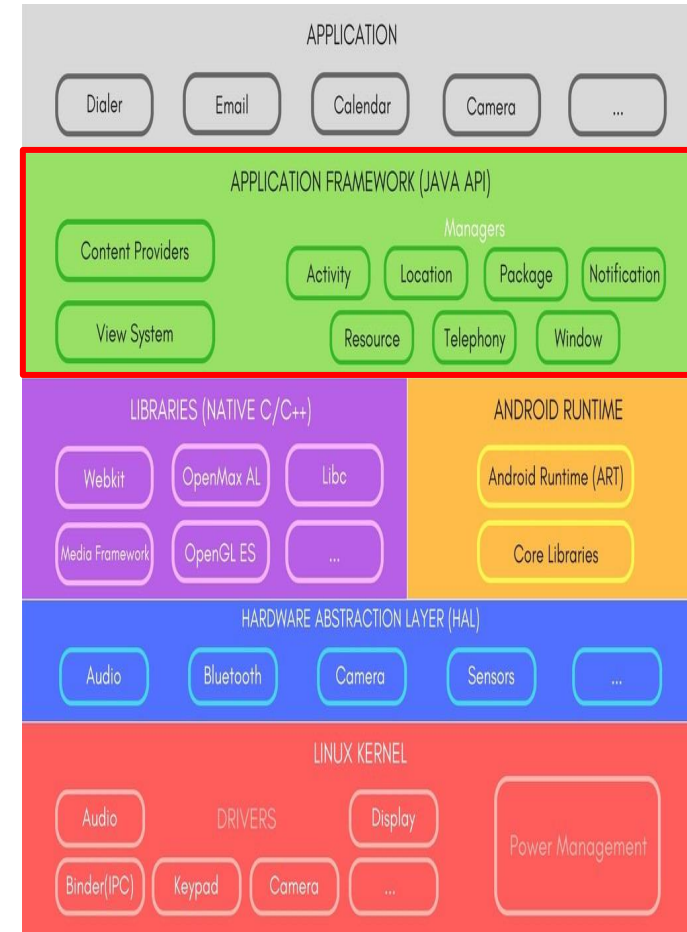
Application Framework

android.nfc		Support for Near Field Communication
android.animation	11 (H)	Animation of views and objects
android.drm		Digital Rights Management and copy protection
android.renderscript		RenderScript (OpenCL like computation language)
android.hardware.usb	12	USB Peripheral support
android.mtp		MTP/PTP support for connected cameras, etc
android.net.rtp		Support for the Real-Time-Protocol (RFC3501)
android.media.effect	14 (I)	Image and Video Effects support
android.net.wifi.p2p		Support for Wi-Fi Direct (Peer-To-Peer)
android.security		Support for keychains and keystores
android.net.nsd	16 (J)	Neighbor-Service-Discovery through Multicast DNS (Bonjour)
android.hardware.input		Input device listeners
android.hardware.display	17	External and virtual display support
android.service.dreams	19 (K)	"Dream" (screensaver) support
android.graphics.pdf		PDF Rendering
android.print[.pdf]		Support for external printing
android.app.job	21 (L)	Job scheduler
android.bluetooth.le		Bluetooth Low-Energy (LE) support
android.hardware.camera2		The new camera APIs
android.media.[browse/projection/session/tv]		Media browsing and TV support
android.service.voice		Activation by "hot words" (e.g. "OK Google")
android.system	22	uname(), poll(2) and fstat[vfs] (2)
android.service.carrier		SMS/MMS support (CarrierMessagingService)
android.hardware.fingerprint	23 (M)	Fingerprint sensor support
android.security.keystore		Cryptographic key generation & storage
android.service.choser		Application deep linking



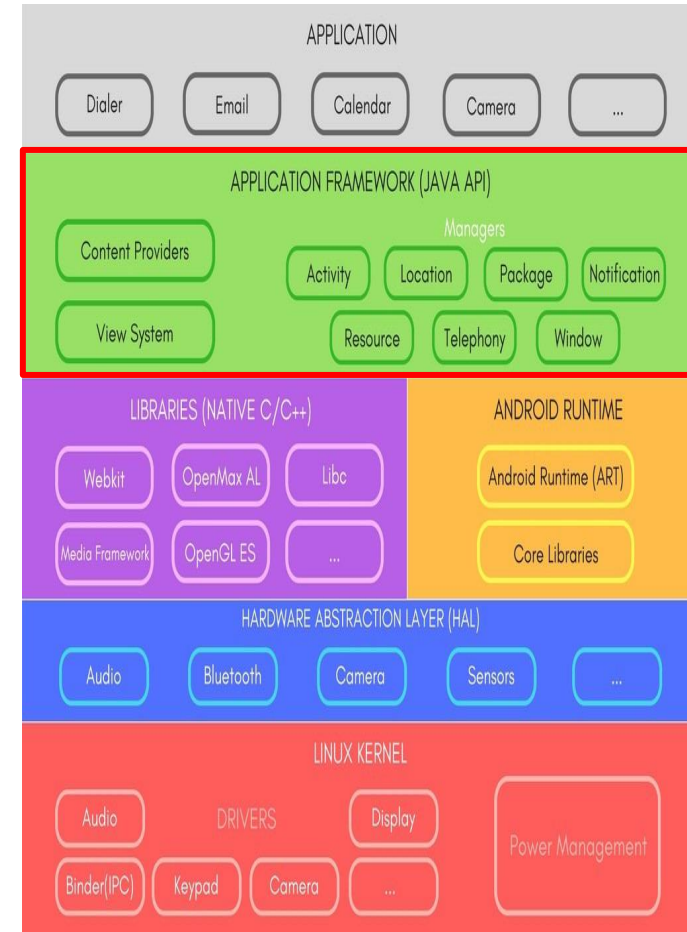
Application Framework

- There are 5 key services in Android framework:
 - Activity managers.
 - Content providers.
 - Resource managers.
 - Notification managers.
 - View system.



Application Framework

- There are 5 key services in Android framework:
 - Activity managers.
 - Content providers.
 - Resource managers.
 - Notification managers.
 - View system.



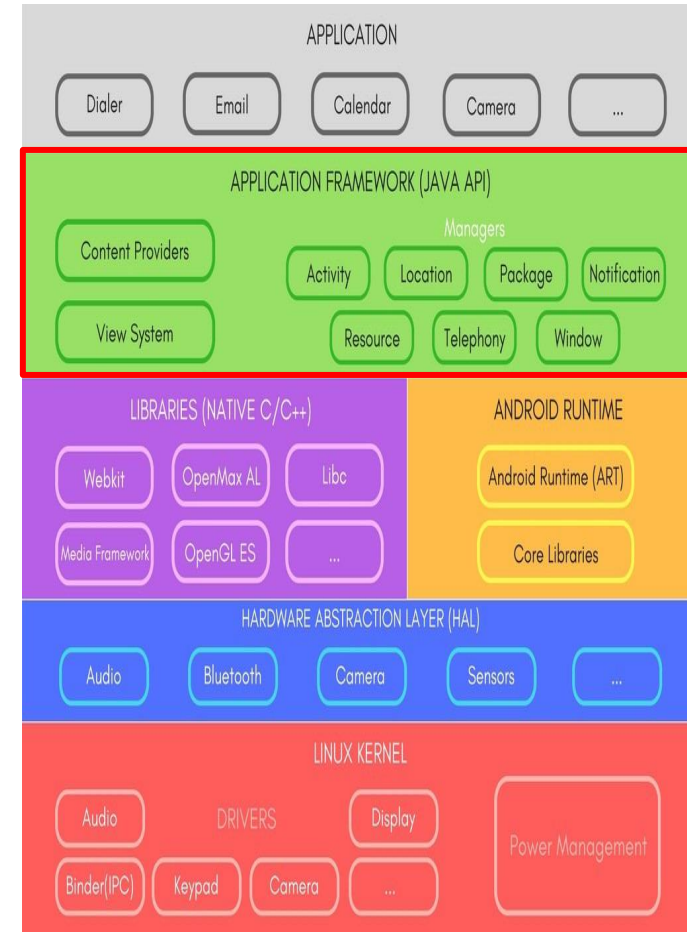
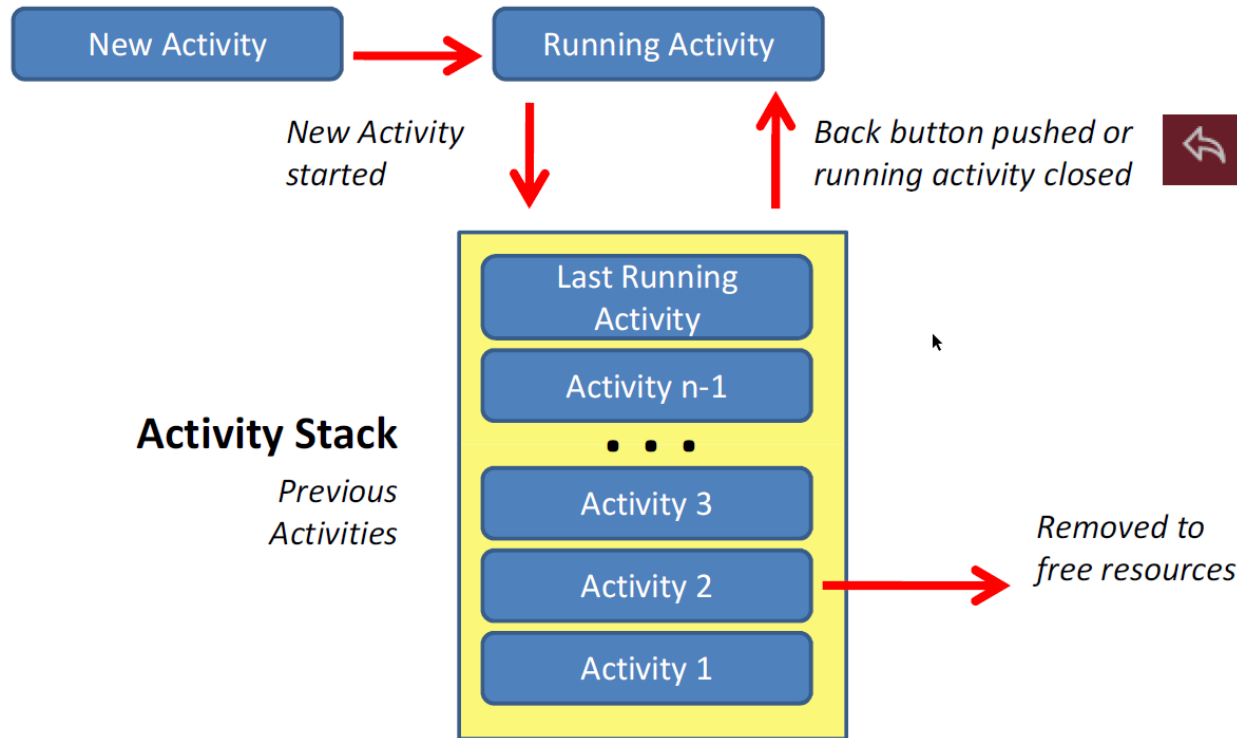
Application Framework

- Activity managers:
 - Controls all aspects of the application lifecycle and activity stack.
 - Activity stack is the one managing all activities in the system.
 - When a new activity is started , it is placed on the top of the stack and becomes the running activity -- the previous activity always remains below it in the stack, and will not come to the foreground again until the new activity exits.
 - If the user presses the Back Button the next activity on the stack moves up and becomes active.



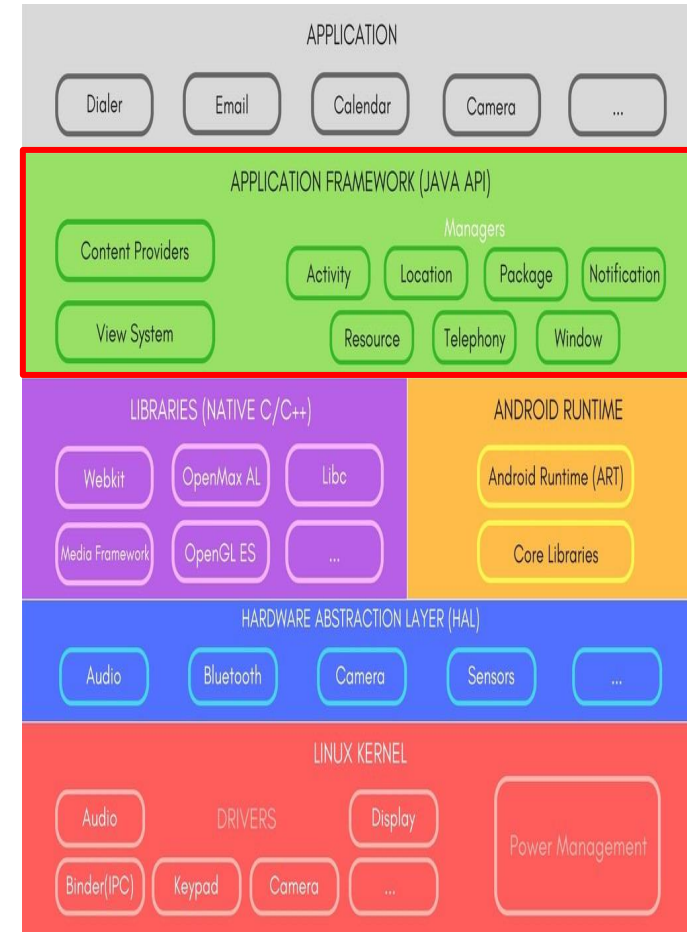
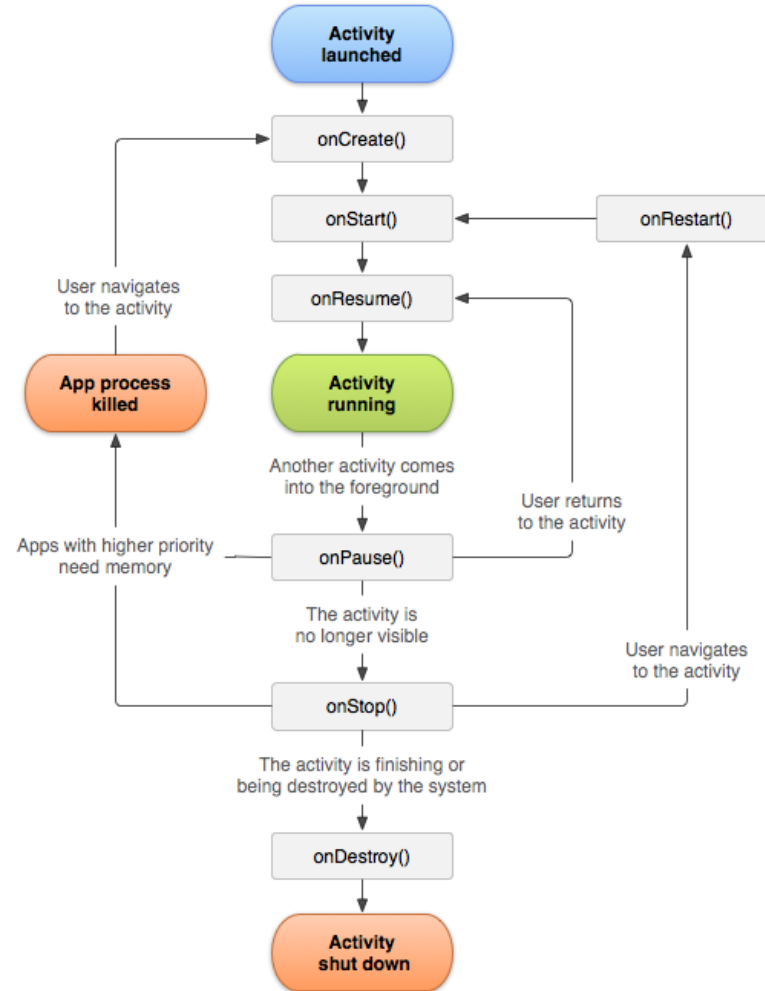
Application Framework

- Activity managers:



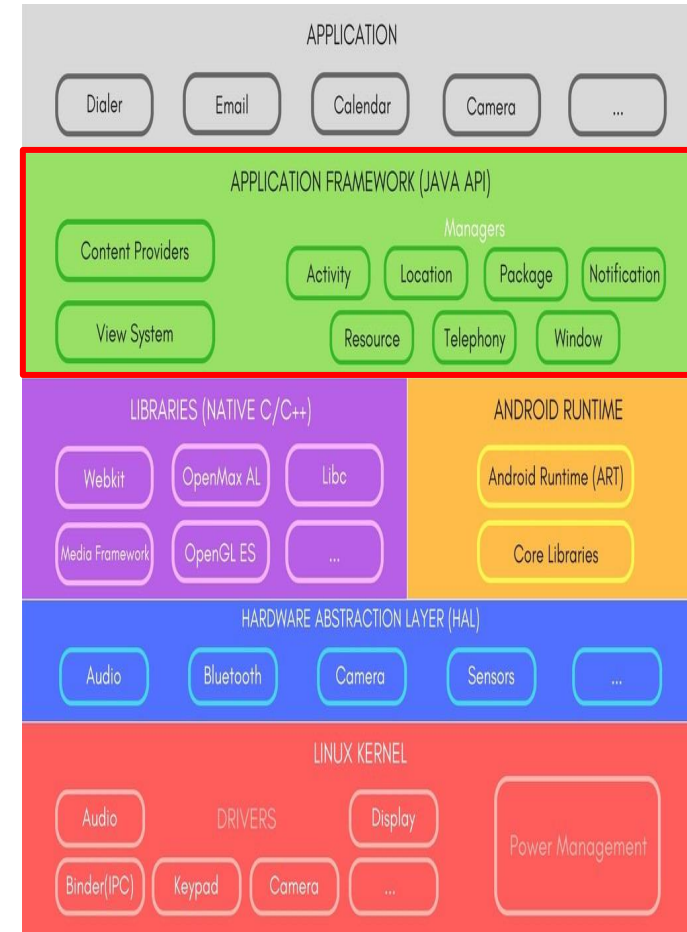
Application Framework

- Activity managers:



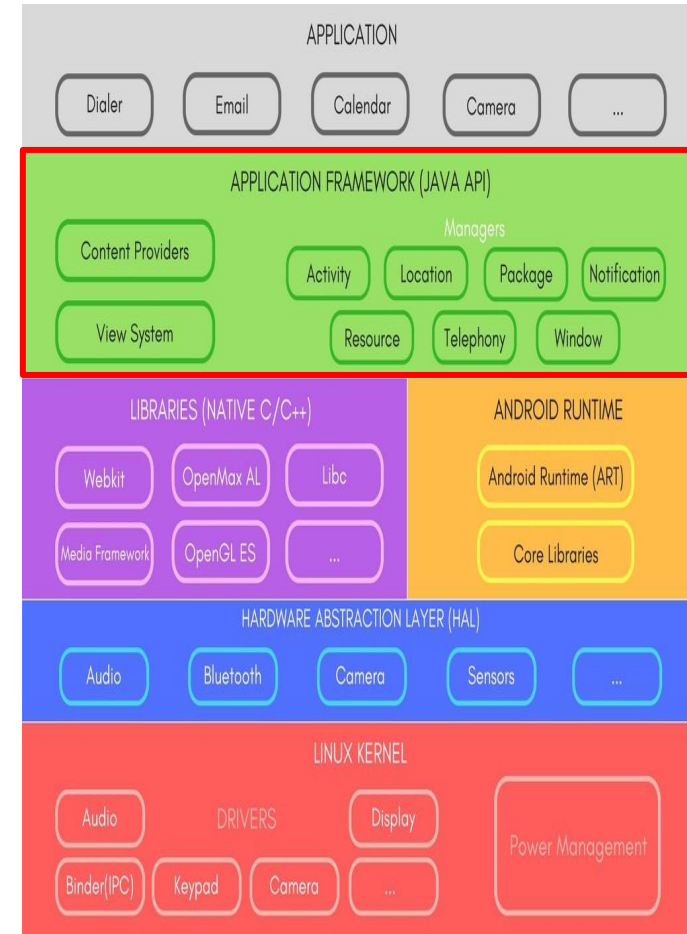
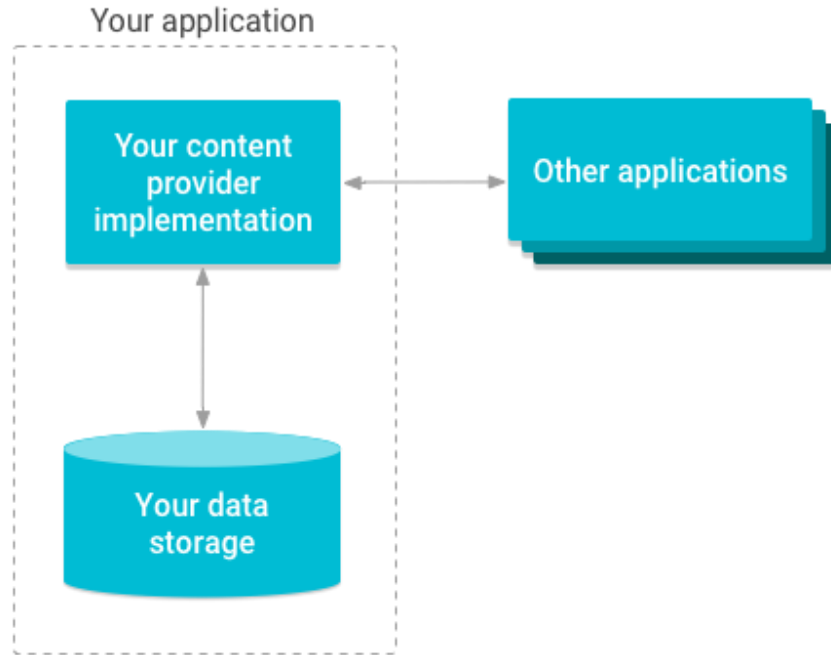
Application Framework

- There are 5 key services in Android framework:
 - Activity managers.
 - Content providers.
 - Resource managers.
 - Notification managers.
 - View system.



Application Framework

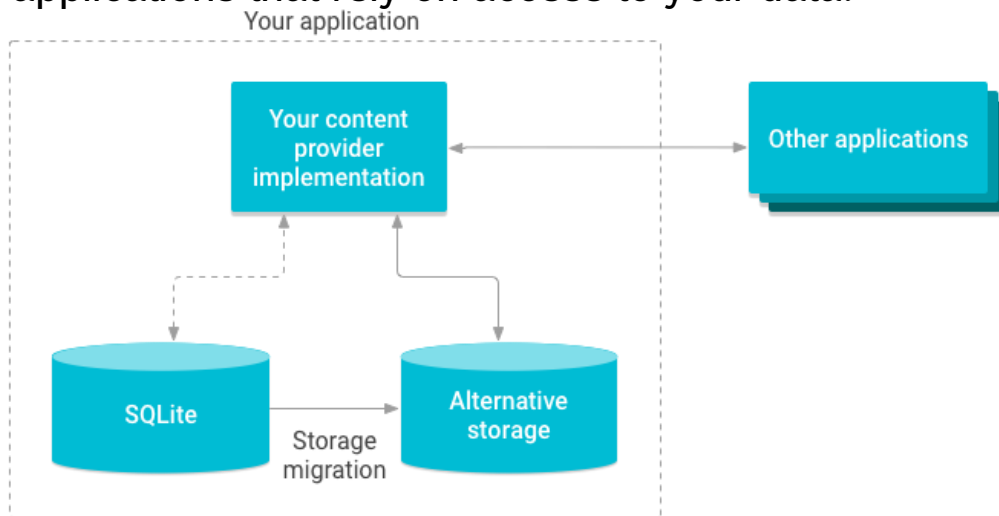
- Content Providers:
 - Allows applications to publish and share data with other applications (eg: Contacts app, etc.)



Application Framework

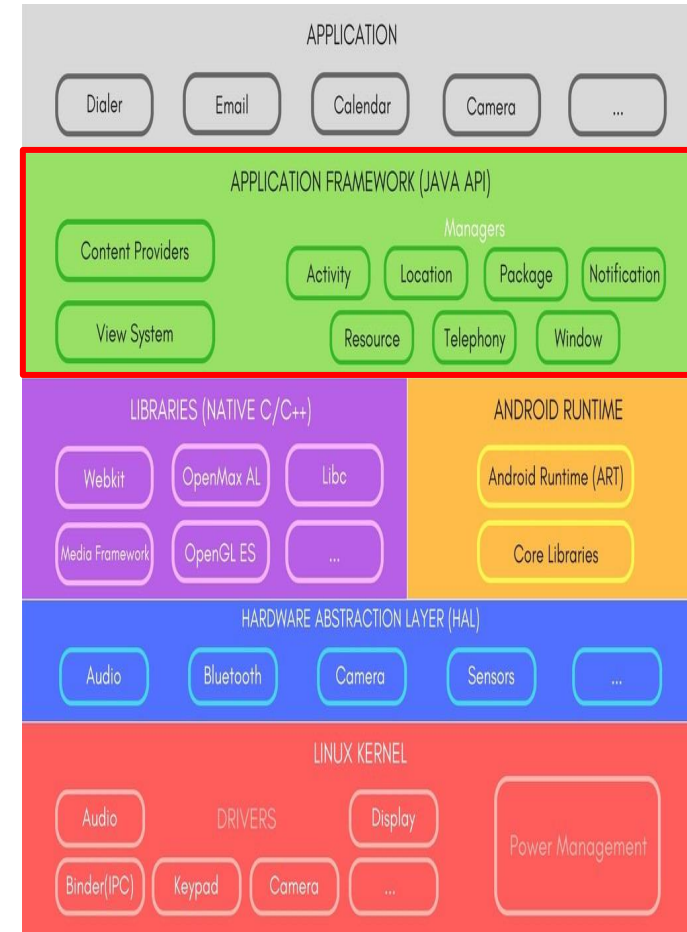
- Content Providers:

- Use content providers if you plan to share data. If you don't plan to share data, you may still use them because they provide a nice abstraction, but you don't have to.
- This will ensure that your application data is modified without affecting other existing applications that rely on access to your data.



Application Framework

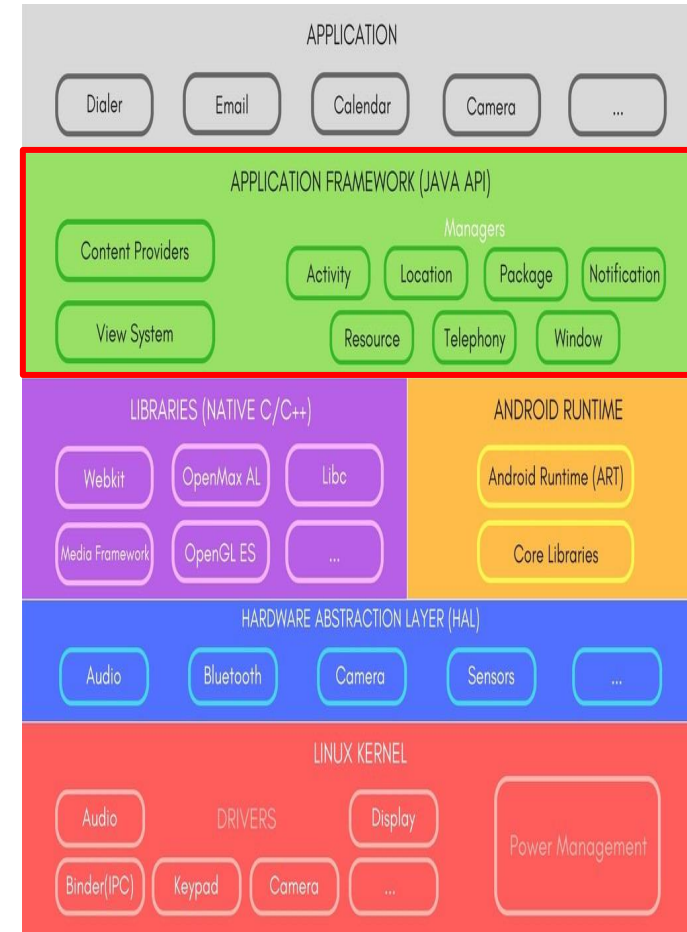
- There are 5 key services in Android framework:
 - Activity managers.
 - Content providers.
 - Resource managers.
 - Notification managers.
 - View system.



Application Framework

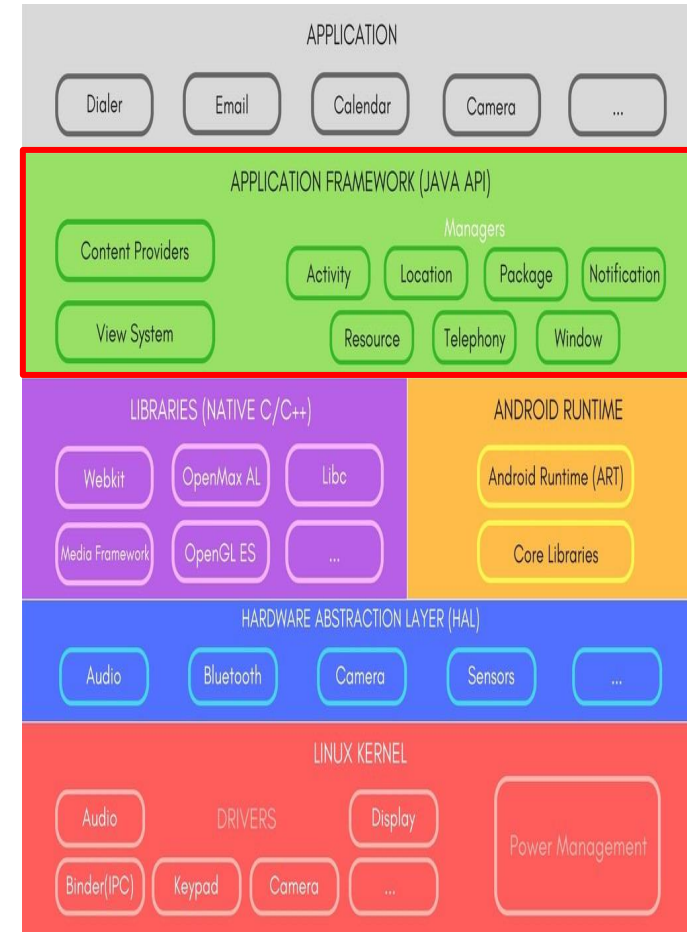
- Resource Manager:
 - Provides access to non-code embedded resources such as strings, color settings, user interface layouts, etc.
 - Each type of resource is placed in a specific subdirectory of project's res/ directory.

```
MyProject/  
  src/  
    MyActivity.java  
  res/  
    drawable/  
      graphic.png  
    layout/  
      main.xml  
      info.xml  
    mipmap/  
      icon.png  
    values/  
      strings.xml
```



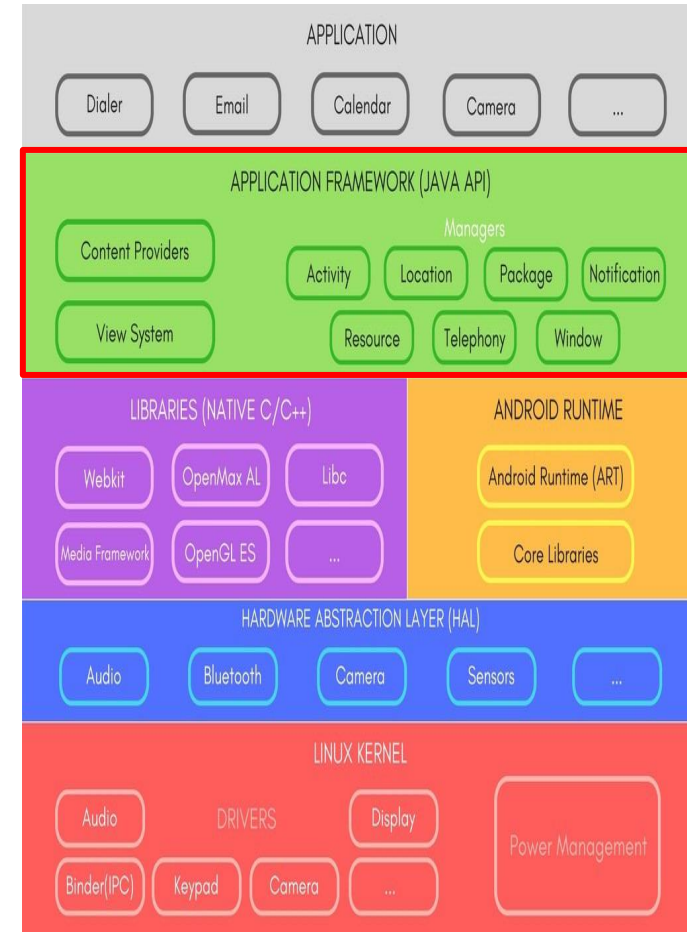
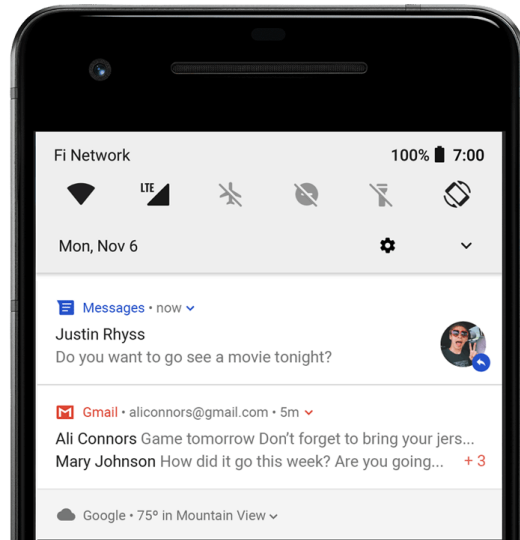
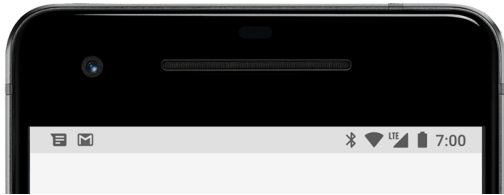
Application Framework

- There are 5 key services in Android framework:
 - Activity managers.
 - Content providers.
 - Resource managers.
 - Notification managers.
 - View system.



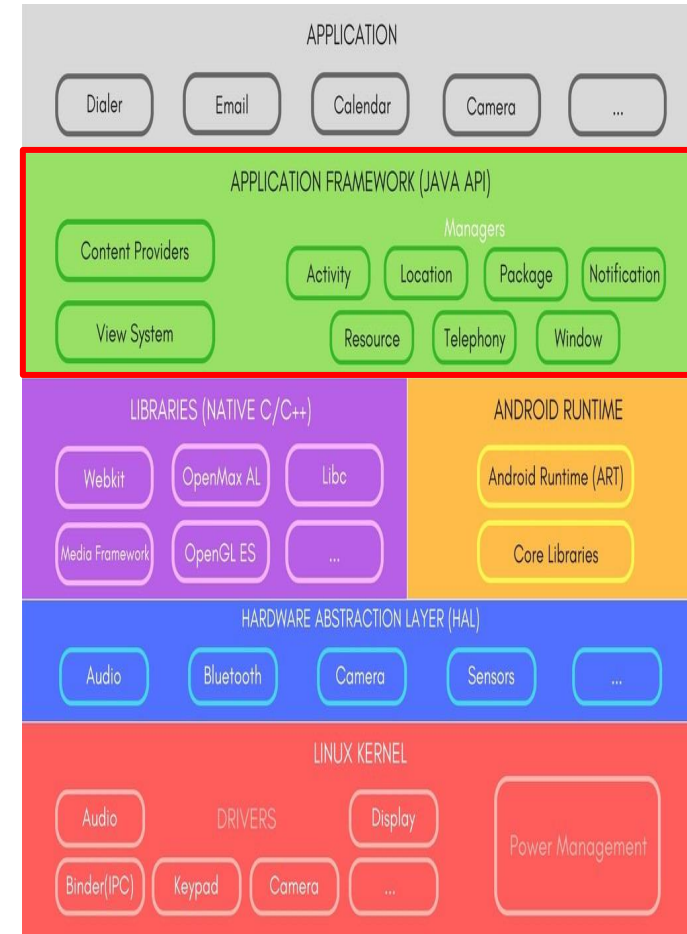
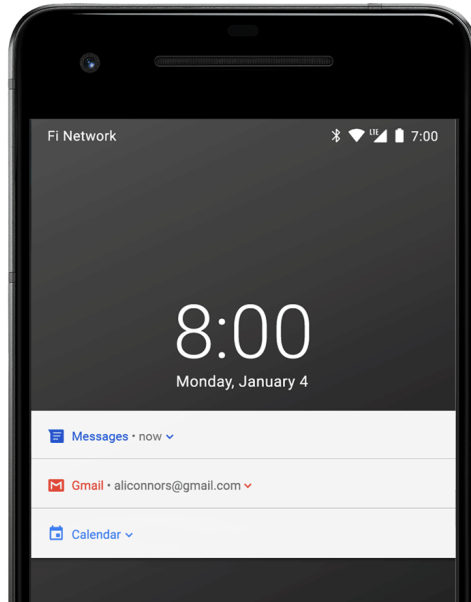
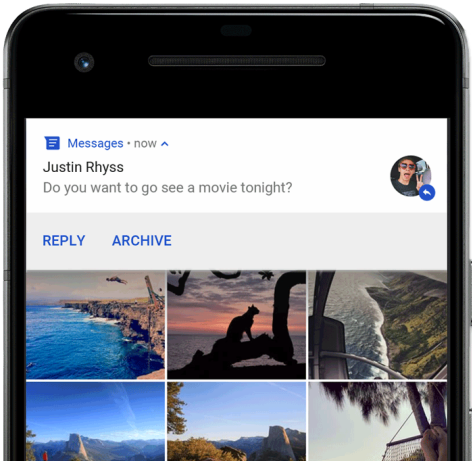
Application Framework

- Notifications Manager:
 - Allows applications to display alerts and notifications to the user.
 - Users can tap the notification to open your app or take an action directly from the notification.



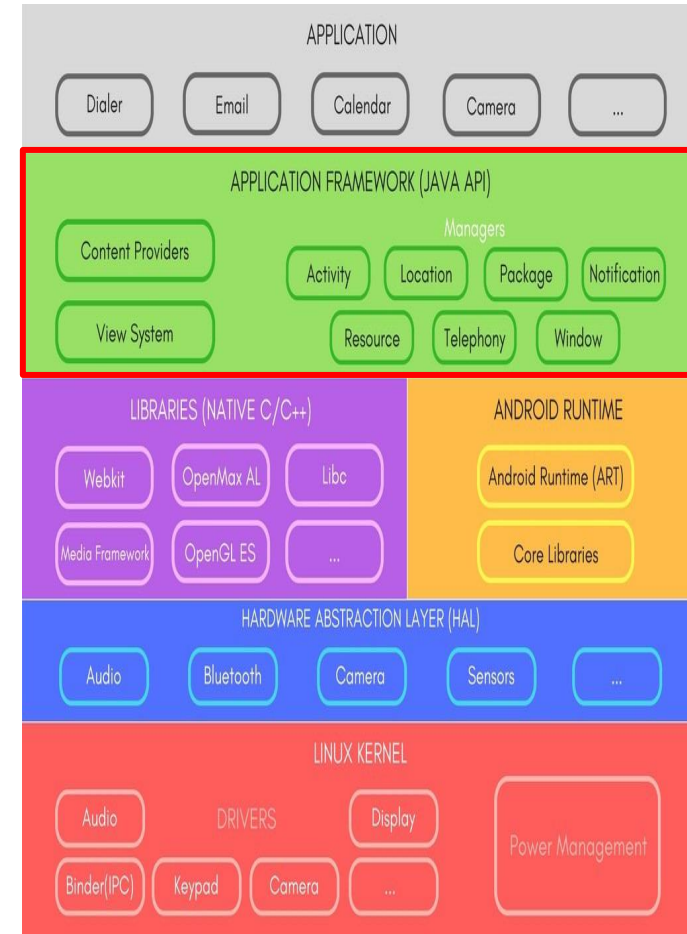
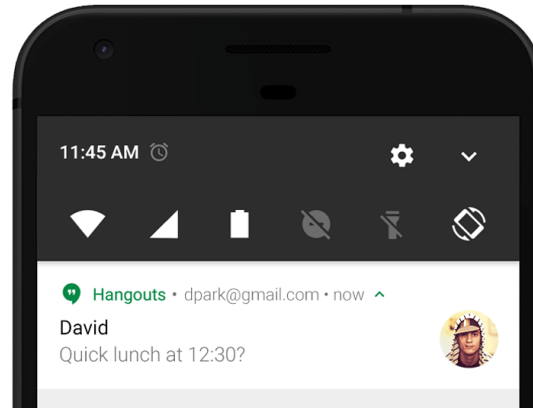
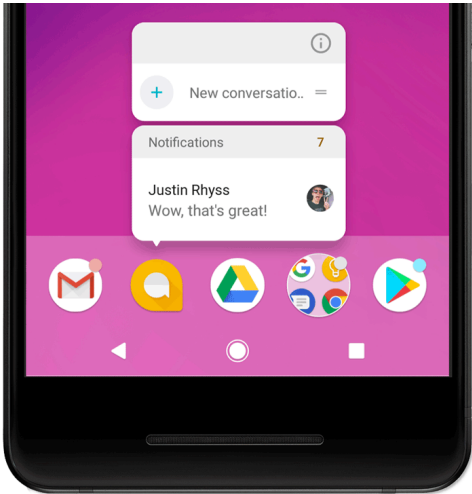
Application Framework

- Notifications Manager:
 - Allows applications to display alerts and notifications to the user.
 - Users can tap the notification to open your app or take an action directly from the notification.



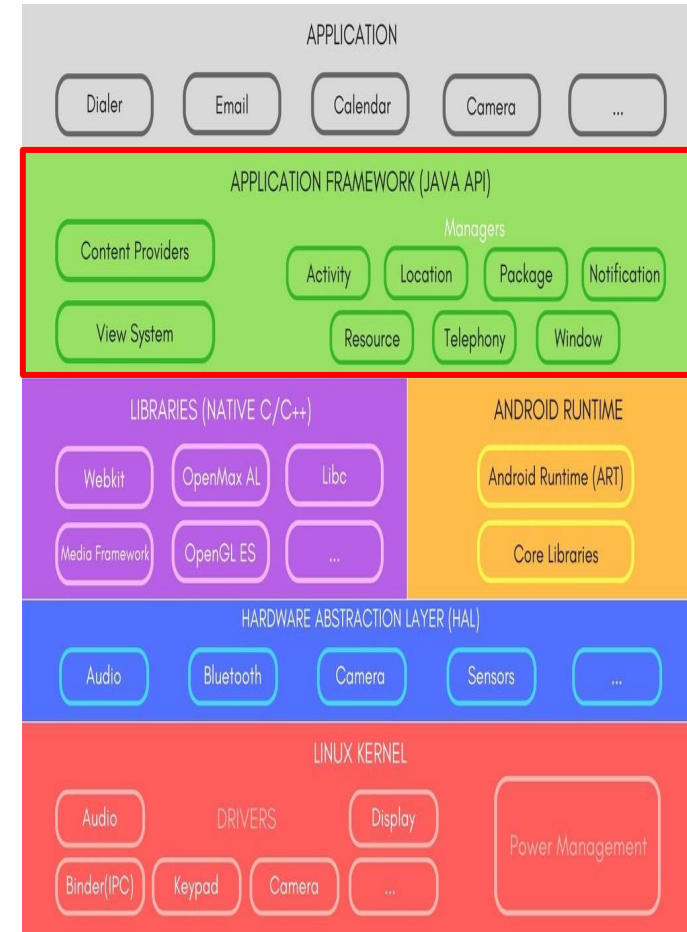
Application Framework

- Notifications Manager:
 - Allows applications to display alerts and notifications to the user.
 - Users can tap the notification to open your app or take an action directly from the notification.



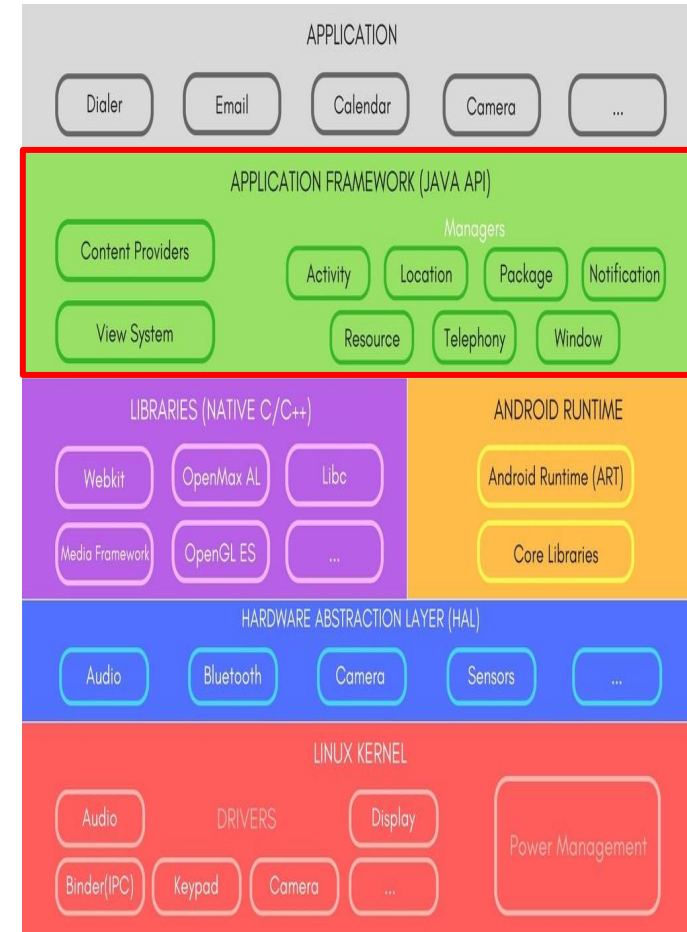
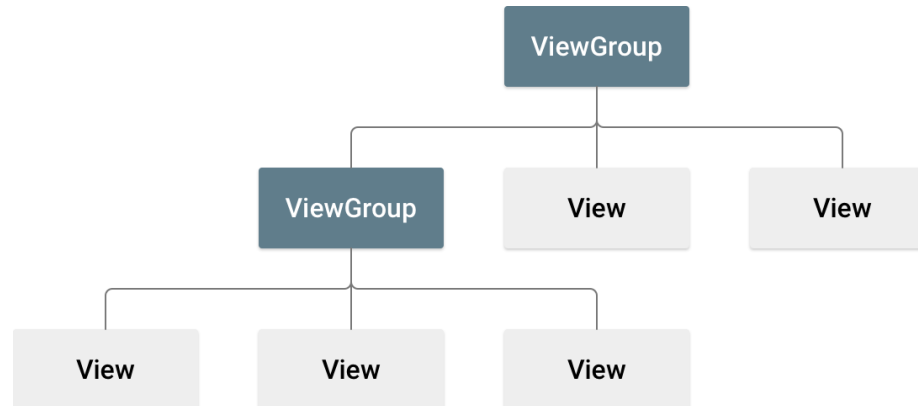
Application Framework

- There are 5 key services in Android framework:
 - Activity managers.
 - Content providers.
 - Resource managers.
 - Notification managers.
 - View system.



Application Framework

- View system:
 - An extensible set of views used to create application user interfaces.
 - All elements in the layout are built using a hierarchy of View and ViewGroup objects.
 - A View usually draws something the user can see and interact with.
 - A ViewGroup is an invisible container that defines the layout structure for View and other ViewGroup objects.



Application Framework

- View system:
 - The View objects are usually called "widgets" and can be one of many subclasses, such as Button or TextView.
 - The ViewGroup objects are usually called "layouts" can be one of many types that provide a different layout structure (eg: LinearLayout, ConstraintLayout).
 - Layout can be declared either in XML file or at runtime.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
```

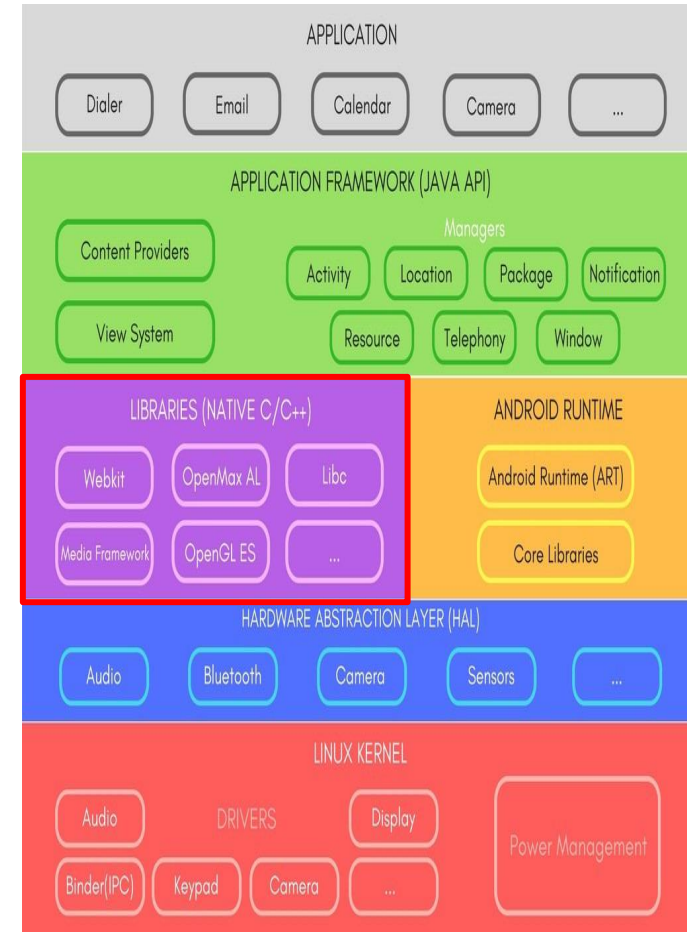


Agenda

- History
- Application field
- Market share
- **Architecture**
 - Overview
 - Application
 - Application Framework
 - **Libraries and Android Runtime**
 - HAL
 - Kernel

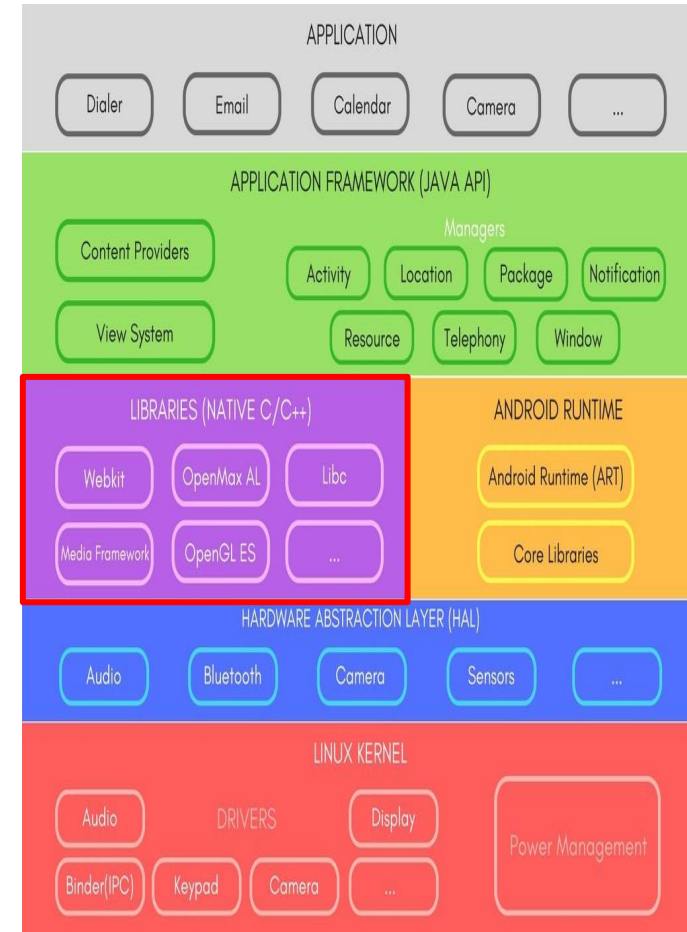
Libraries

- Gather a lot of Android specific libraries to interact at a low level with the system, but third parties libraries as well.
- Bionic is the C library, SurfaceManager is used for drawing surfaces on the screen, etc.
- The libraries mostly provide wrappers over kernel Androidisms, or implement additional functionality in user-mode.
- The functionality of these native libraries were exposed to Android application via Java framework APIs.



Libraries

- Binaries are usually located in `/system/bin`, and `/xbin` (with a few critical binaries located in `/sbin`).
- All of them are ELF binaries.
- Most binaries are usually the same across all devices, being part of the AOSP.



Android Runtime

- Handle the execution of Android applications.
- Almost entirely written from scratch by Google.
- Contain ART, the virtual machine that executes every application that you run on Android, and the core library for the Java runtime, coming from Apache Harmony Project.
- Also contain system daemons, init executable, basic binaries, etc.



Android Runtime

- Each app runs in its own process and with its own instance of the Android Runtime (ART).
- ART is written to run multiple virtual machines on low-memory devices by executing DEX files, a bytecode format designed specially for Android that's optimized for minimal memory footprint.



JNI

- A Java framework to call and be called by native applications written in other languages. Mostly used for:
 - Writing Java bindings to C/C++ libraries
 - Accessing platform specific features
 - Writing high performance sections
 - It is used extensively across the Android user space to interface between the Java Framework and the native daemons.
- Since Gingerbread, Java methods can be called through JNI.



Agenda

- **Architecture**


- Overview
- Application
- Application Framework
- Libraries and Android Runtime
- **HAL**
- Kernel

Hardware Abstraction Layer (HAL)

- Provide standard interfaces that expose device hardware capabilities to the higher-level Java API framework.
- Consist of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the camera or bluetooth module.
- When a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component.



Day 3



Thank for your listening!