Inside Android

Mobile Software
2019 Fall

What is Android?

- **Android** is the **software platform** from Google and the Open Handset Alliance (OHA) that has the potential to **revolutionize the global cell phone market**.
  - It has the capability to make inroads in many other **(non-phone) embedded application markets**.

Android

- **Features**
  - **재사용 가능한** Application Framework
  - Dalvik **가상 머신**
  - WebKit **기반 내장 웹 브라우저**
  - 2D and 3D graphics APIs with HW
  - **내장 데이터베이스**(SQLite)
  - **각종 오디오 및 비디오 규격 지원**
  - Bluetooth, EDGE, 3G/4G, and Wi-Fi
  - Camera, GPS, compass, and accelerometer
- **Resources**
  - **Android's web page** http://www.android.com
  - **Open Source Project** http://source.android.com
  - **Android Developers** http://developer.android.com

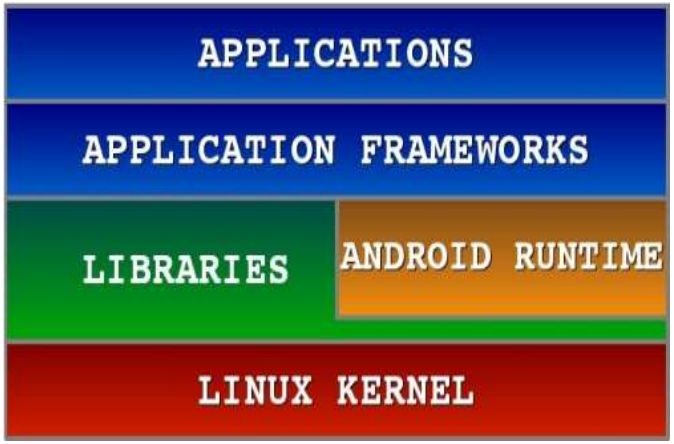
Main differences between each **version of the android platform**

Version name	Key user features added
Pie (9)	**User interface updates** Richer message notification The power option has a "screenshot" button
Oreo (8.1)	Emoji updates Neural networks API for artificial intelligence
Oreo (8.0)	PIP Picture-in-Picture with resizable windows Android instant apps
Nougat (7.1.2)	**Battery usage alerts** **Long press on the app icon enable new launch action** **New set of emoji**
Nougat (7.0)	Unicode 9.0 emoji **Better multitasking** **Multi-window mode** (**PIP, Freeform window**) Better performance and code size thanks to new JIT Compiler

출처: http://socialcompare.com/en/comparison/android-versions-comparison

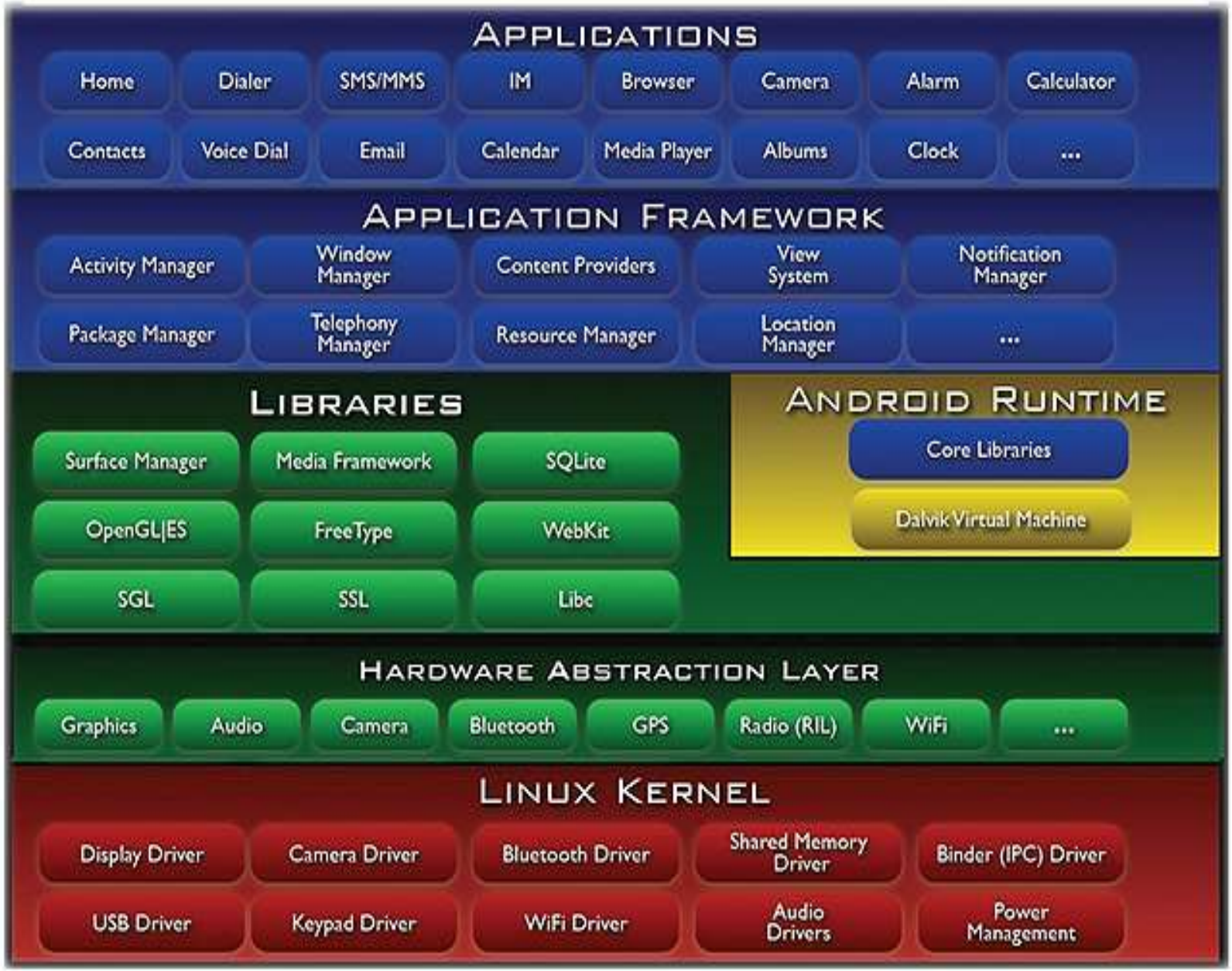
Android Platform

- Android is a software environment built for mobile devices.
  - It is **not** a hardware platform.

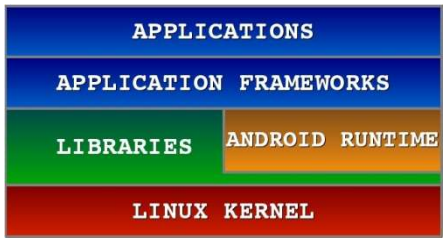


- 계층 구조(layered architecture)
  - 최하위 계층 **: Linux Kernel**
  - 라이브러리 : java가 아닌 **c로 작성**
  - Runtime : JVM이 아닌 별도의 가상 머신(Dalvik Virtual Machine)
  - Frameworks : Android API
    - android에서 제공하는 주요 기능(java 로 구현)

Android Architecture(Android Stack)



Linux Kernel



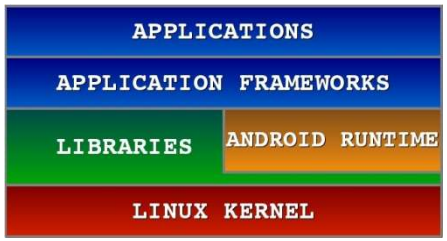
- Relying on Linux Kernel 2.6 for core system services
  - Memory and Process Management
  - Network Stack
  - Driver Model
  - Security
- Providing an abstraction layer between the H/W and the rest of the S/W stack

Why use Linux for a phone?

- **Linux** kernel is a **proven core platform**.
  - Reliability is more important than performance when it comes to a mobile phone.

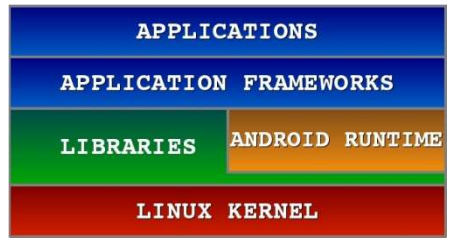
- Linux provides a **hardware abstraction layer**.
  - 기술 변화에 따라 하위 계층의 하드웨어가 바뀌어도
    - 상위 계층은 영향을 받지 않는다.
  - As new accessories appear on the market,
    - drivers can be written at the Linux level to provide support, just as on other Linux platforms.

하드웨어 추상 계층 (Hardware Abstraction Layer)



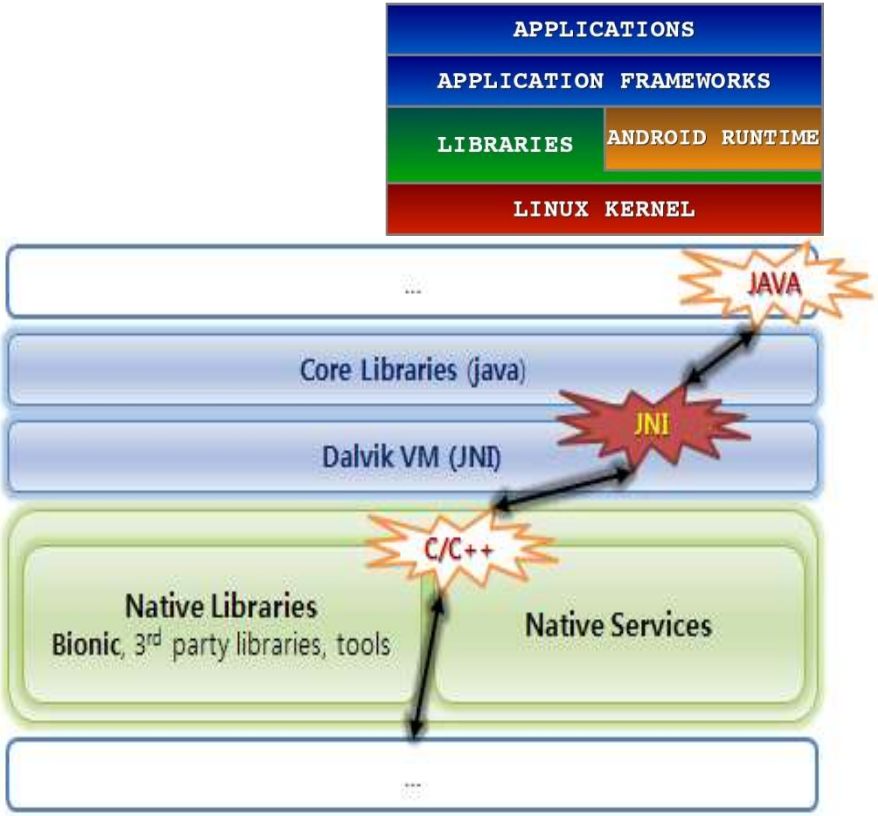
- C/C++ library layer
  - Defines the interface that Android requires hardware "drivers" to implement
  - Separates the Android platform logic from the hardware interface

Libraries



- Android application framework에서 library 사용 가능
- libc : c++ library support
- SGL : Scalable Graphics Library – 2D 그래픽 라이브러리
- SSL : Secure Socket Layer – 서버와 클라이언트 간 인증
  - 예: https://...

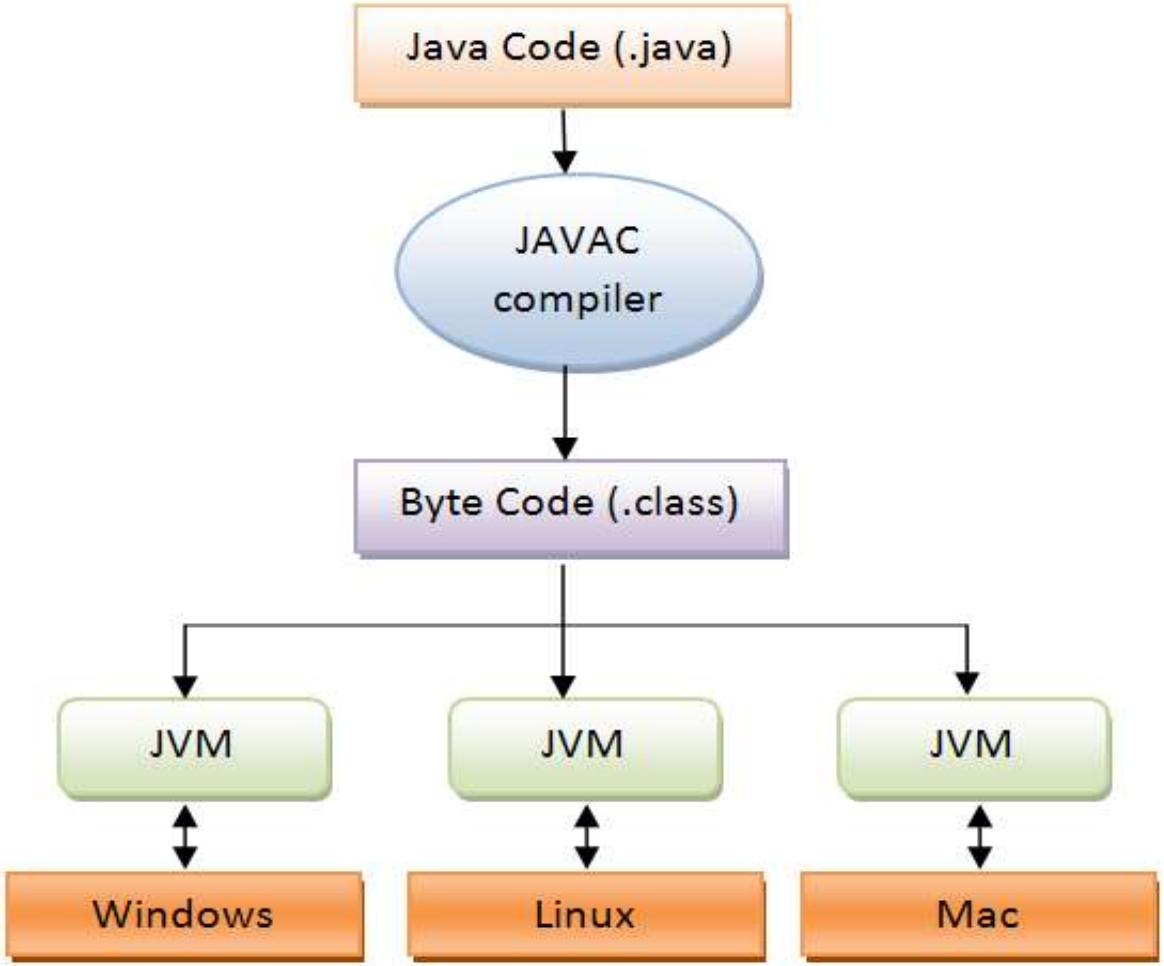
Runtime



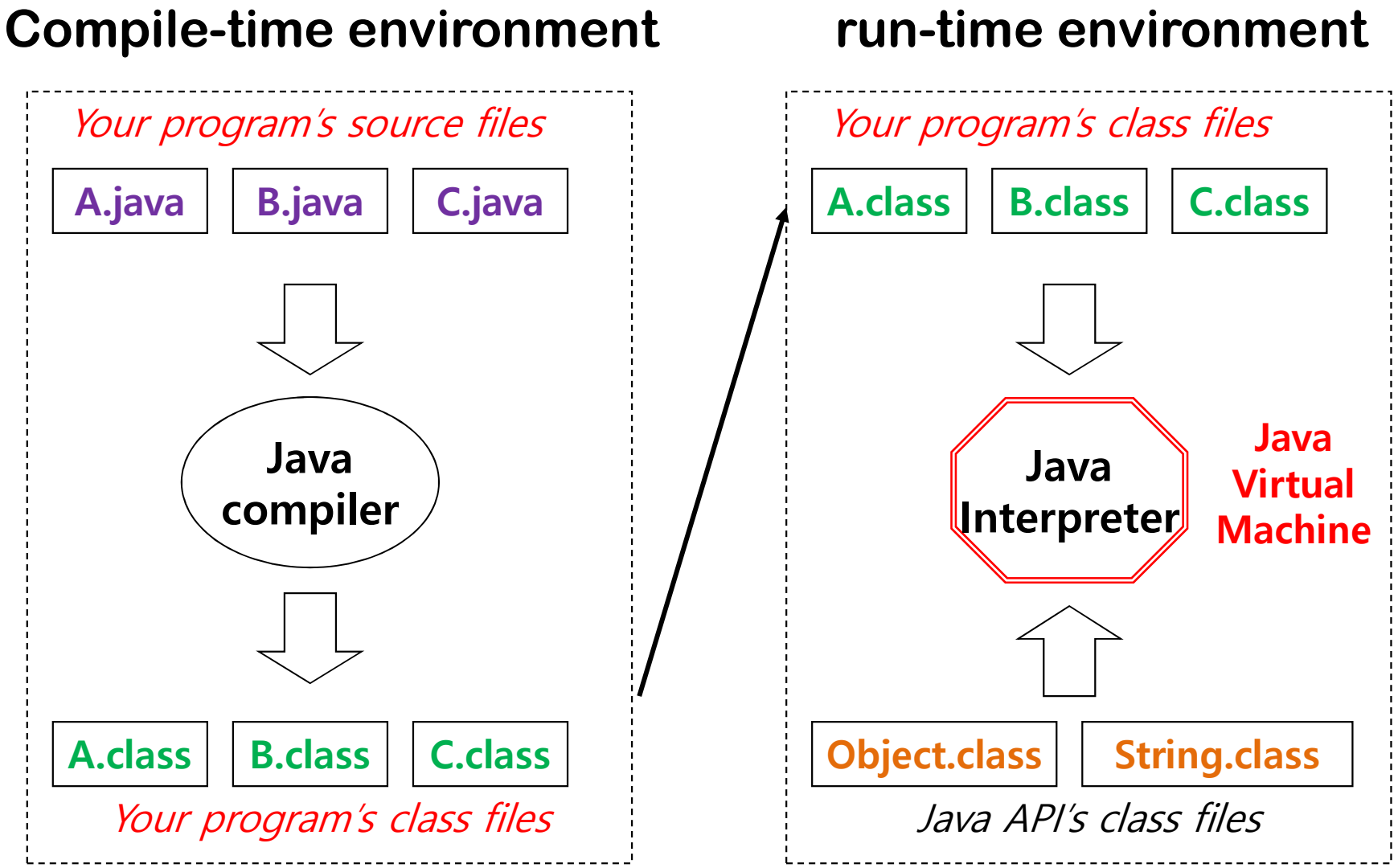
- Core Libraries
  - Providing most of the functionality available in the core libraries of the Java language
- 응용프로그램은 Java, H/W 드라이버는 C/C++
  - JNI를 통해 C/C++ native libraries 호출

JNI: Java Native Interface

Java Virtual Machine (JVM)



Java Programming Environment



Dalvik VM

- **배경**
  - Dan Bornstein을 중심으로 개발
    - Dalvik : 아이슬란드 마을 이름

  - Java 라이선스 문제를 해결할 목적
    - How google routed around Sun's IP-based licensing restriction on Java ME
- 특징
  - Register-based VM : 적은 메모리에 최적화
  - Bytecode가 아닌 dex 포맷 사용
    - dex : Dalvik Executable
    - dx : xxx.**class** → xxx.**dex**

Dalvik과 ART (1/2)

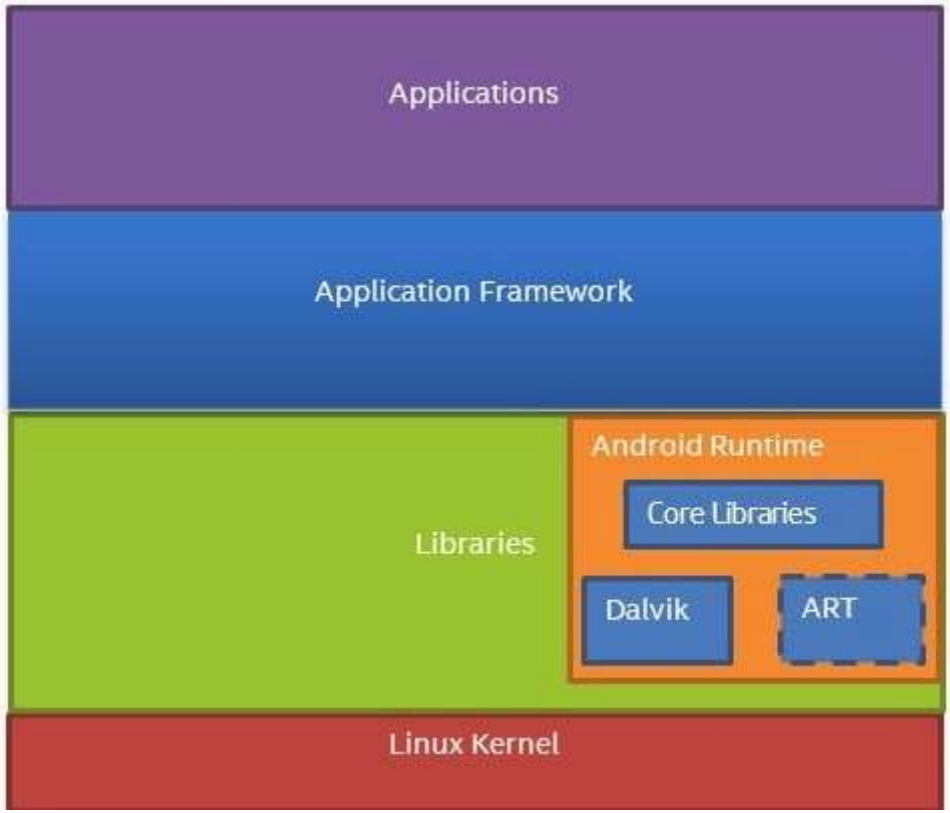
- **JIT** (*Just-In-Time*) Compilation
  - **JVM**
    - interpreter
      - 한 번에 한 줄씩 bytecode 를 읽어 와서 실행
  - **JIT**
    - 전체 byte code 중에서 실행 빈도가 높은 함수(*hotspot*)의 경우만
      - target platform에서 실행 가능한 native code로 번역
    - **Android 2.2** (Froyo) 버전부터 포함
    - 장점
      - 속도가 빨라짐.
    - 단점
      - 배터리 소모가 크고, 화면 전환 시에 속도가 느려짐.

Dalvik과 ART (2/2)

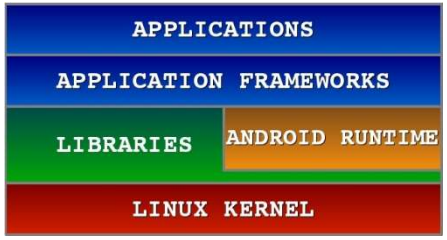
- **AOT** (*Ahead of Time*) compilation 과 ART
  - **ART** : Android Run Time
    - VM이 아니라 runtime library
  - 특징
    - 전체 byte code 를 모두 target platform 에서 실행 가능한 native code로 변환해서, 메모리에 저장
      - 이 과정은 App. 을 설치할 때만 딱 한 번 필요.
      - App.을 실행할 때는 interpret없이 native code를 실행
    - Android 4.4(KitKat)부터 도입
      - Android 5.0(Lollipop)부터는 Dalvik VM을 대체.
  - 장점
    - 처리 속도가 빨라지고, RAM 사용량이 적어짐.
  - 단점
    - 메모리를 더 많이 필요로 하고, 설치 속도가 느려짐.

Dalvik과 ART

- ART as the runtime executes the Dalvik Executable format and Dex bytecode specification.
- ART and Dalvik are **compatible runtimes** running Dex bytecode, so apps developed for Dalvik should work when running with ART.
- However, some techniques that work on Dalvik do not work on ART.

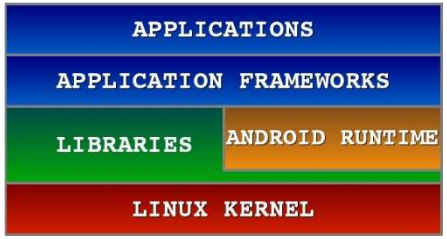


Application Framework



- Enabling and simplifying the **reuse** of **components**
  - Developers have full access to the same framework APIs used by the core applications.
  - Users are allowed to replace components.

Applications



- Android provides a set of core applications:
  - Email Client, SMS, Calendar, Maps, Browser, Contacts, …
- All applications are written using **Java**.