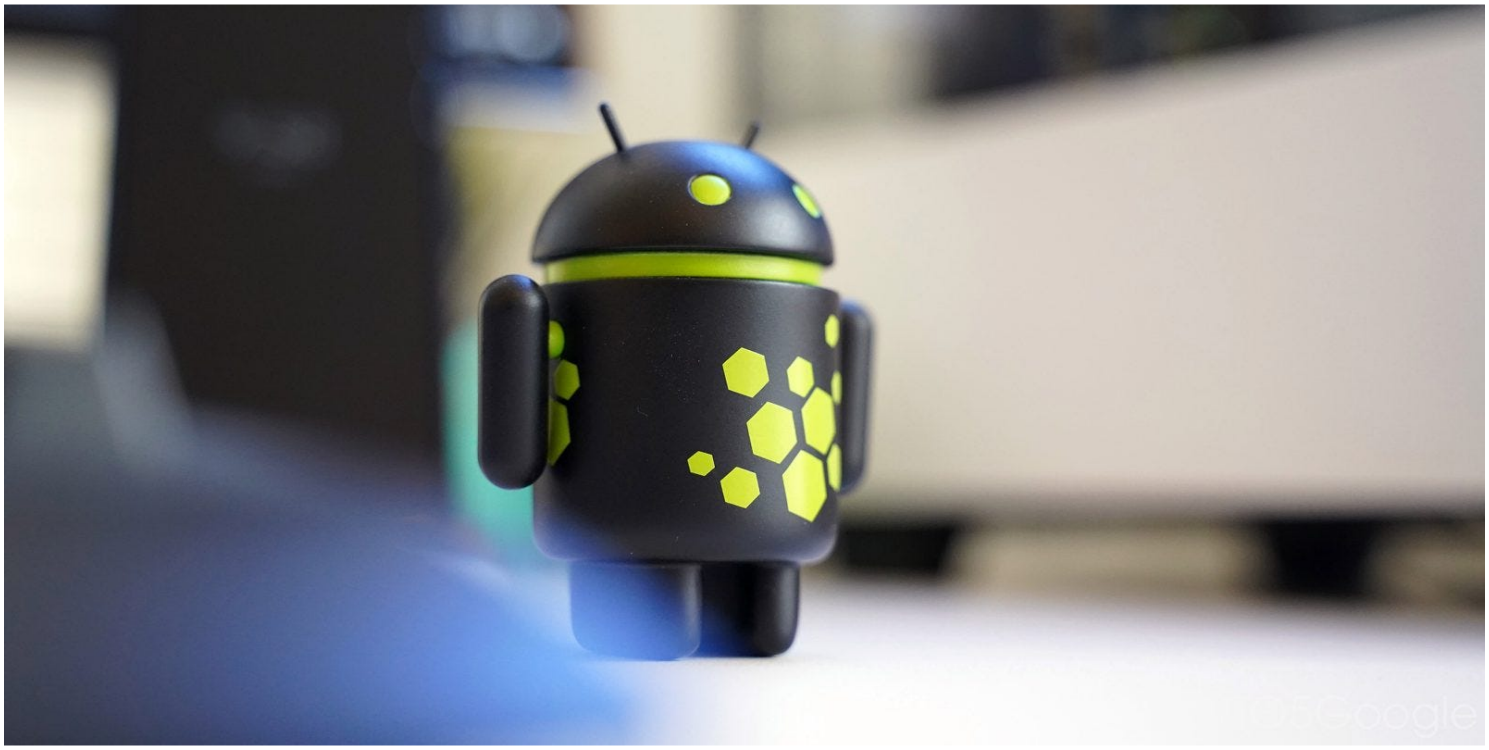
Virtual Machine in Android: Everything you need to know

Tam H. Doan · Follow
Published in AndroidPub
6 min read · Aug 4, 2019

Listen Share More

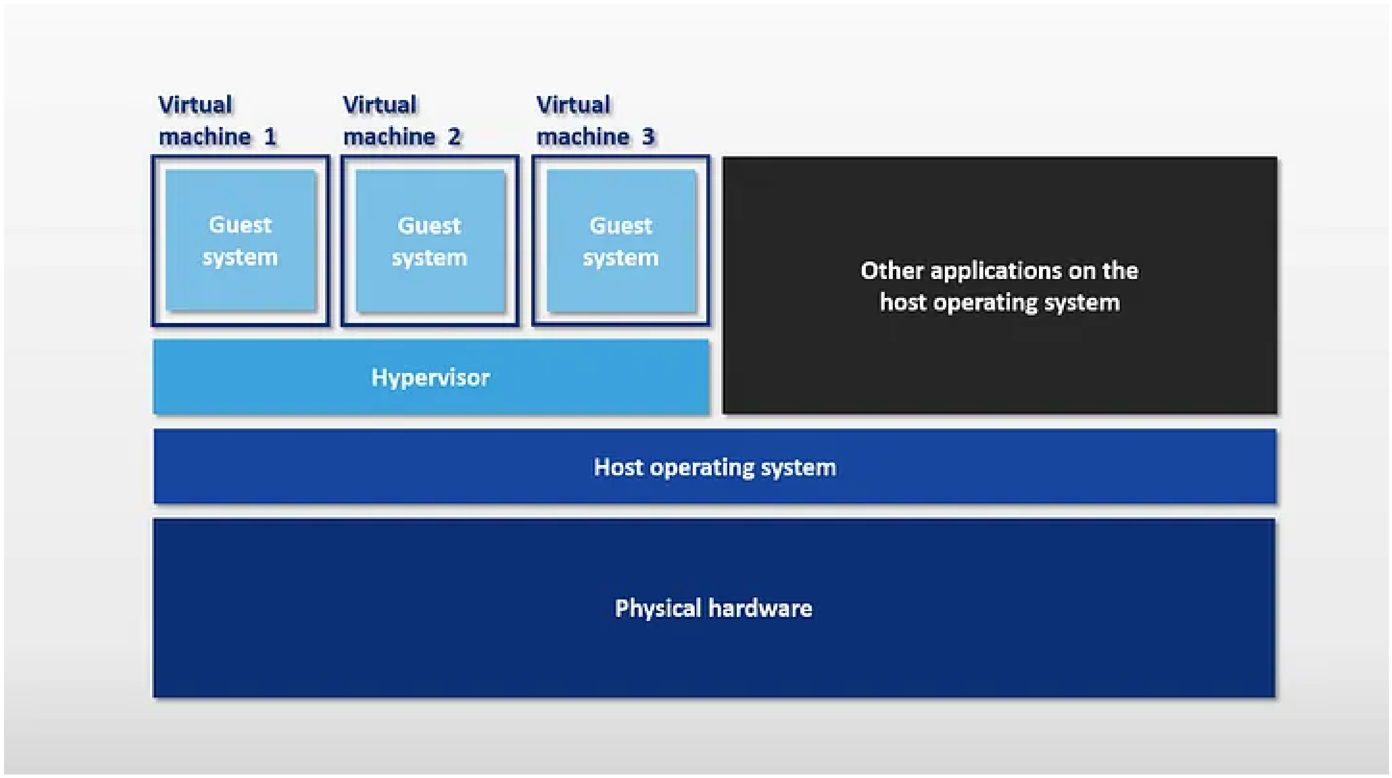

Welcome to the Android world!

Virtual Machine?

A virtual machine is based on computer architectures to provide functionality of a computer.

There are 2 main types of Virtual Machine (VM):

- **System virtual machines** (full virtualization VMs) provide a substitute for a real machine.

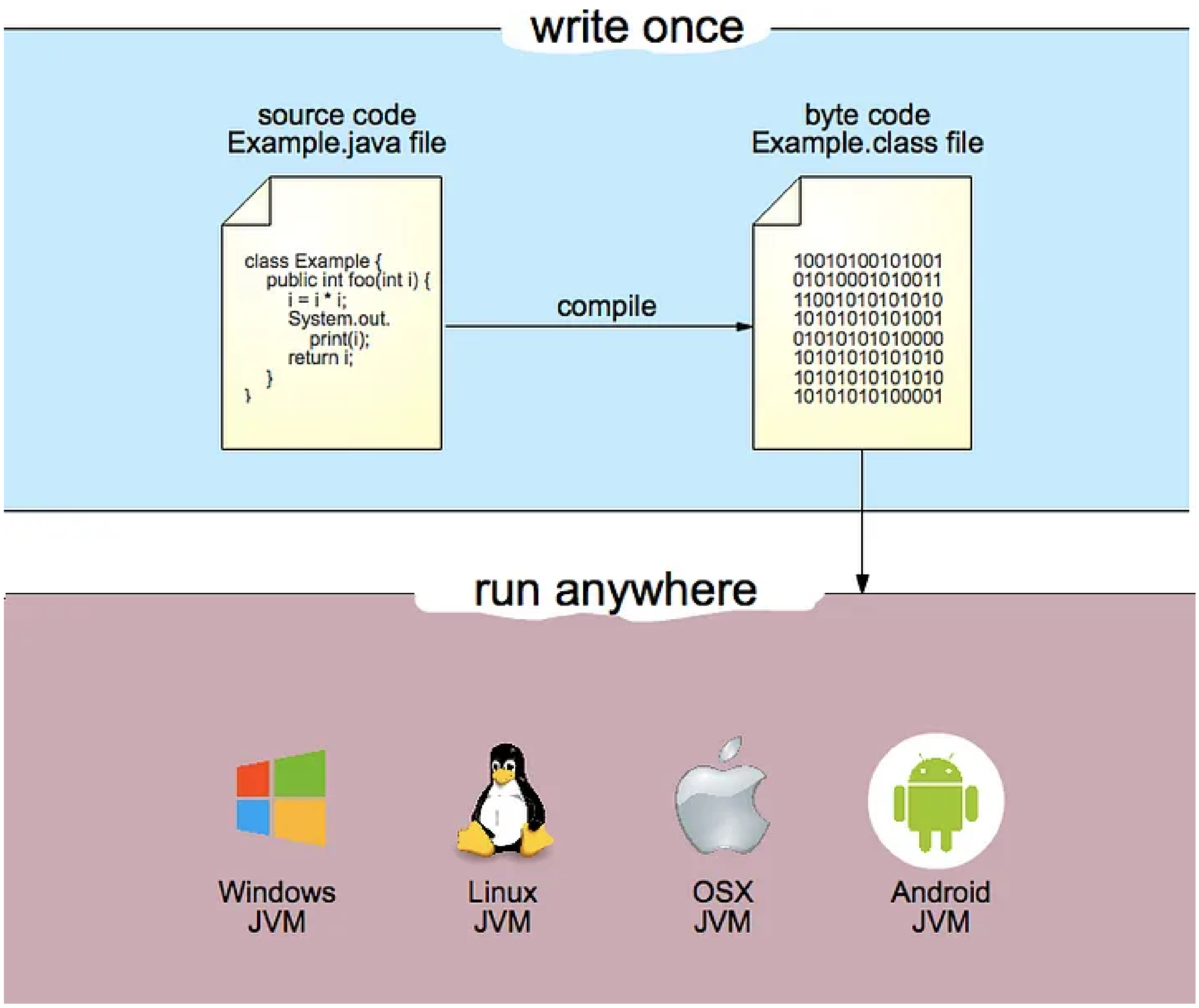


Virtual Machine architecture. Source: IONOS.

Java Virtual Machine

Java Virtual Machine (JVM) is a **Process virtual machine.**

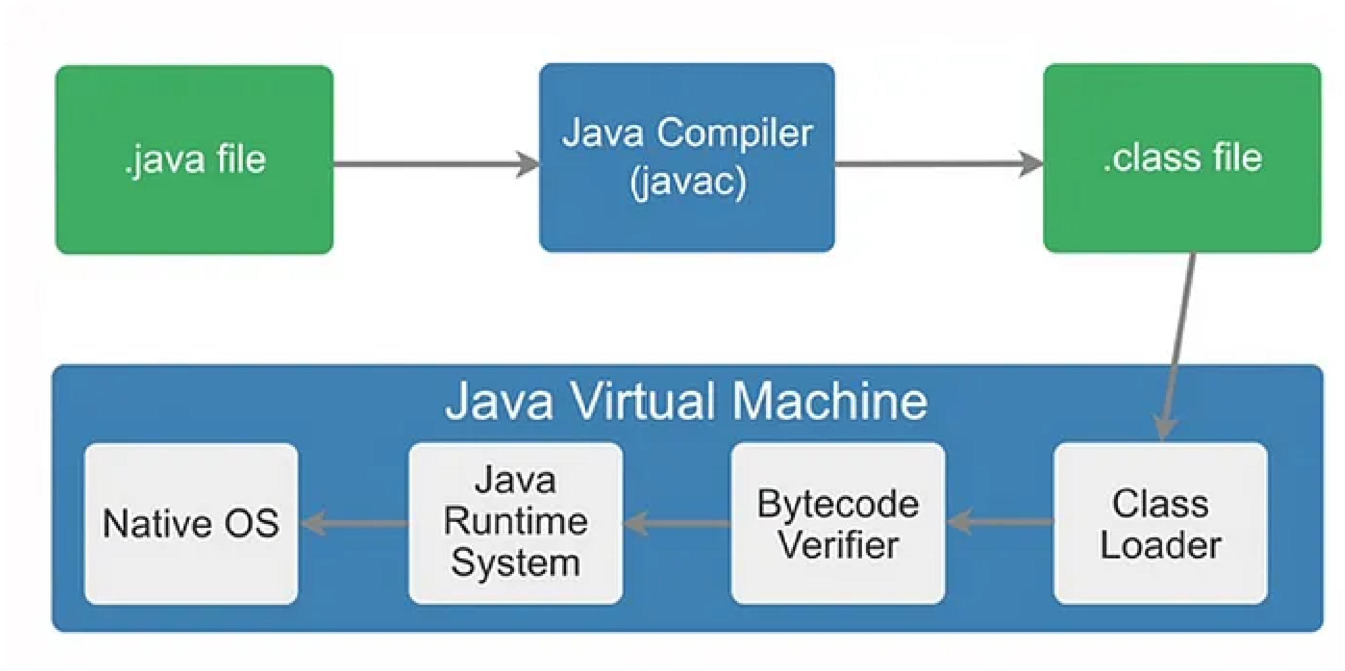
Follow "write once, run anywhere" principle, JVM allows programs — written in Java/Kotlin — to run on any device or operating system.



Write once, Run anywhere! Source: JavaTutorial

How it works?

IDEs — like IntelliJ / Android Studio — use Java Compiler (`javac`) to compile Java code (`.java`) into Java bytecode (`.class`).



JVM internals.

Then on runtime environments (Web, PC, Android), **JVM translates it into machine specs instruction set that this environment can understand.**

- Start JVM.
- Start main thread. Load `.class` into memory using **Classloader**.
- Verify bytecode is valid and not violate Java's security restrictions.
- Translate bytecode into machine instructions and execute them.
- Unload classes. Terminate main thread.
- Shutdown JVM.

Note that: It's possible to compile Java code to native code ahead-of-time, then run. And it's also possible to interpret Java code directly.

Android OS

Android is an open source, **Linux-based** software stack created for a wide array of devices and form factors. For example, the **Android Runtime (ART)** relies on the Linux kernel for underlying functionalities: networking, multi-threading and low-level memory management,..

Why Android use virtual machine?

There are many reason that Google engineers decide to use Android with VM, but two main reason is:

- **Security**: In theory, app code is totally isolated by the VM and cannot even “see” the host OS. So app code that contains malware cannot affect system directly, make app and system more robust and reliable.
- **Platform independent**: Android platform can run on different devices with different architectures (ARM, MIPs, x86). To abstract out the need to compile binaries for each architecture, VM comes into play.

Dalvik Virtual Machine

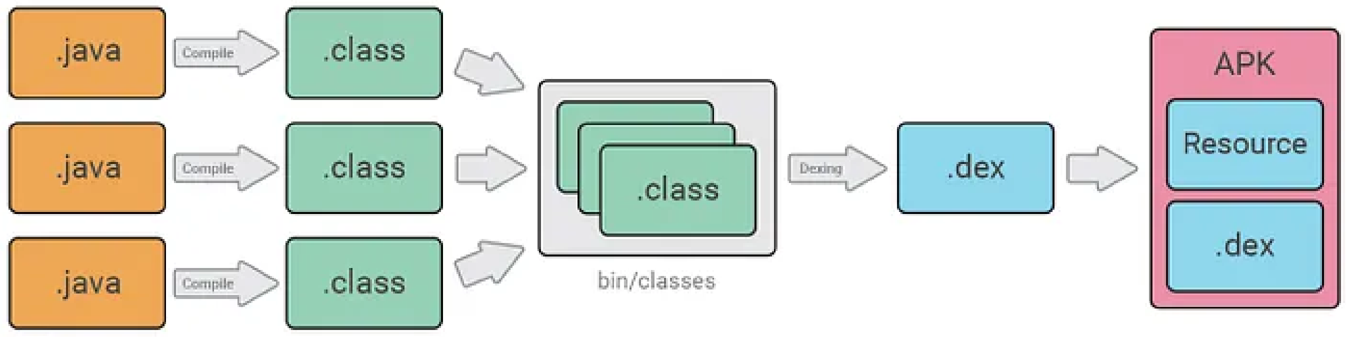
Dalvik Virtual Machine (DVM) is a **Process virtual machine** for Android.

> DVM is a virtual machine that optimized for mobile environment (memory, battery life, performance,..)

Dan Bornstein named it after the Dalvík village in Iceland. Source: Android Developer Blog.

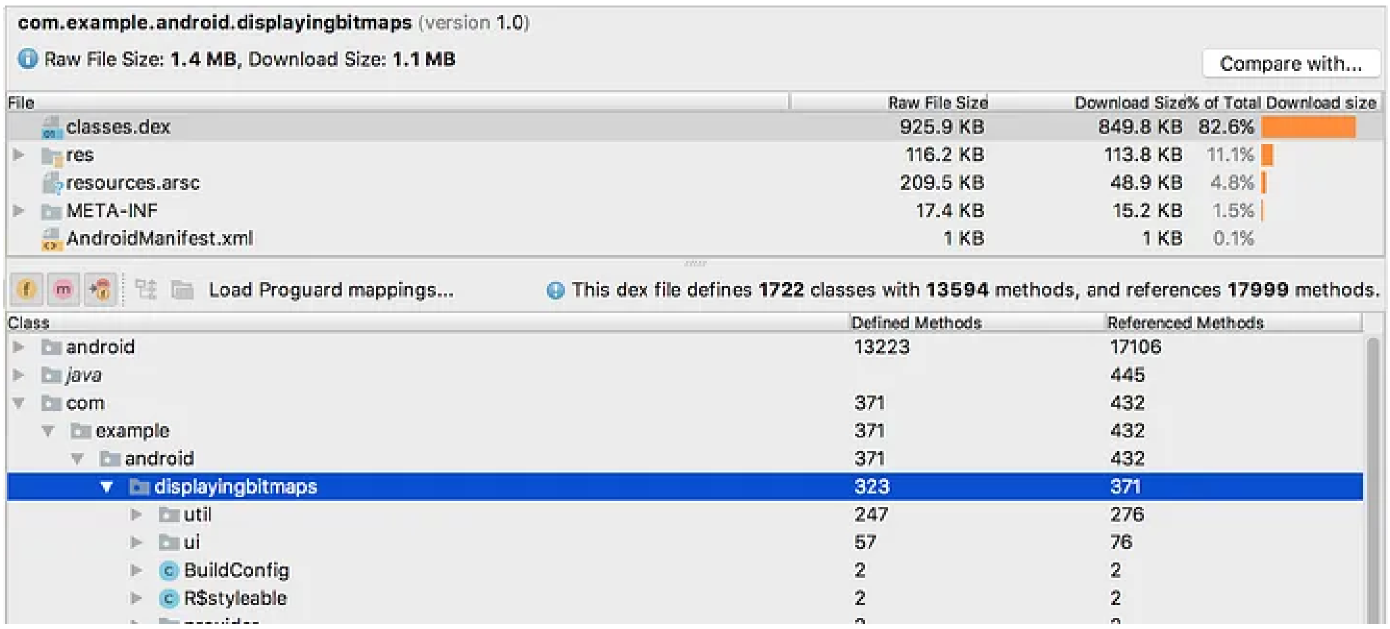
Dalvik EXecutable

Dalvik EXecutable (`.dex`) is Dalvik bytecode which **translated from Java bytecode** using dx tool (replaced by D8 tool since API 28). This format was designed for systems that have constrained memory and processor speed.



The process of generate new APK

Multiple classes are included in a single `.dex` file:


com.example.android.displayingbitmaps (version 1.0)

Raw File Size: 1.4 MB, Download Size: 1.1 MB

File	Raw File Size	Download Size	% of Total Download size
classes.dex	925.9 KB	849.8 KB	82.6%
res	116.2 KB	113.8 KB	11.1%
resources.arsc	209.5 KB	48.9 KB	4.8%
META-INF	17.4 KB	15.2 KB	1.5%
AndroidManifest.xml	1 KB	1 KB	0.1%

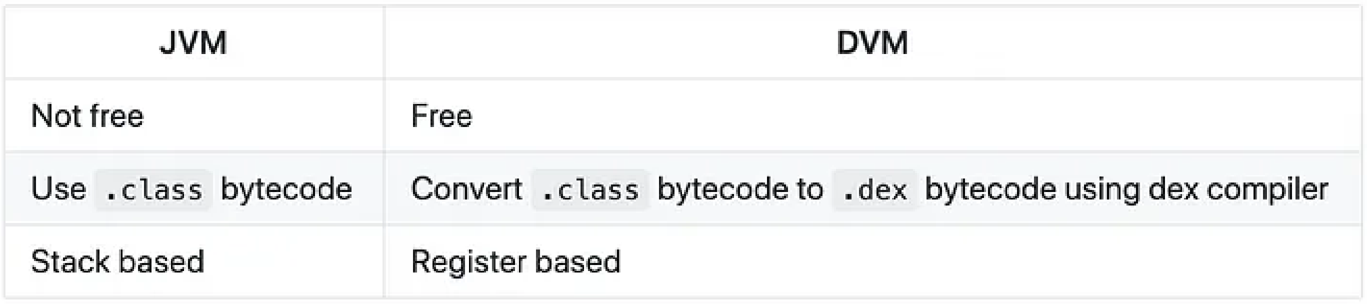
Load Proguard mappings... This dex file defines 1722 classes with 13594 methods, and references 17999 methods.

Class	Defined Methods	Referenced Methods
android	13223	17106
java		445
com	371	432
example	371	432
android	371	432
displayingbitmaps	323	371
util	247	276
ui	57	76
BuildConfig	2	2
R$styleable	2	2

You can use APK Analyzer to see what's inside a DEX file. Source: Android Developers.

Compare with Java VM

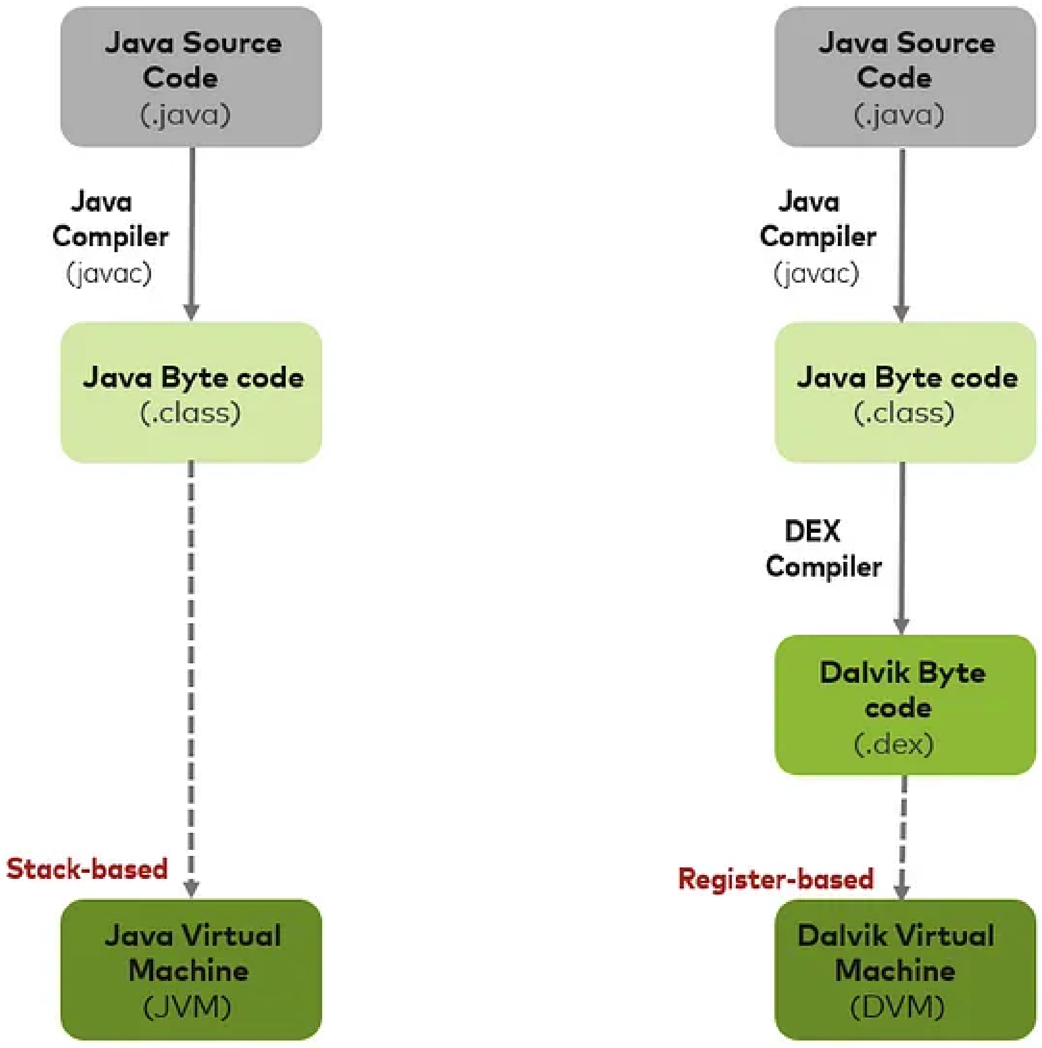
The DVM was **built specifically for Android, to address the battery life, processing speed.** Moreover, the Java tools are free but the JVM is not, so Google engineers made their own VM and made it as **free**.



JVM	DVM
Not free	Free
Use `.class` bytecode	Convert `.class` bytecode to `.dex` bytecode using dex compiler
Stack based	Register based

JVM vs Dalvik VM

Unlike JVM, which are simple stack machines, the DVM uses a register-based — which requires fewer instructions, fewer code units, avoid unnecessary memory access — resulting in better performance code.



JVM and Dalvik VM flows

Android Runtime

> Android Runtime (ART) is the managed runtime used by apps and system services on Android. Replacing the predecessor Dalvik, ART performs the translation of the app's bytecode into native instructions that are later executed by the device's runtime environment.

The ART is written to **run multiple VMs on low-memory devices.** To maintain backward compatibility, **ART also uses the same input bytecode as Dalvik** — the standard Dalvik EXecutable (`.dex`) files — which also designed for Android to minimize memory footprint.

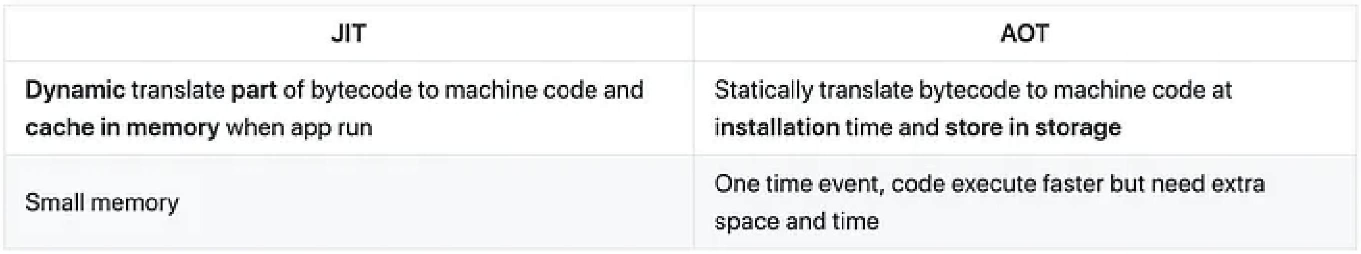
Android Runtime (ART).

From Android 5.0, each app runs in its own process, with its own instance of ART. But prior to this, it use Dalvik. If your app runs well on ART, then it should work on Dalvik as well, but the reverse may not true.

Just-In-Time vs Ahead-Of-Time

Just-In-Time (JIT) added in Android 2.2. It **dynamically** compiles bytecode into native machine code **at run-time** to make app runs faster. It does all this while app is running, and that's where the "Just-In-Time" comes from.

Ahead-Of-Time (AOT) added in Android 4.4. It **statically** compiles bytecode into machine code **at install-time** — using on-device dex2oat tool — and stores data in the device's storage, to **improve run-time performance**.



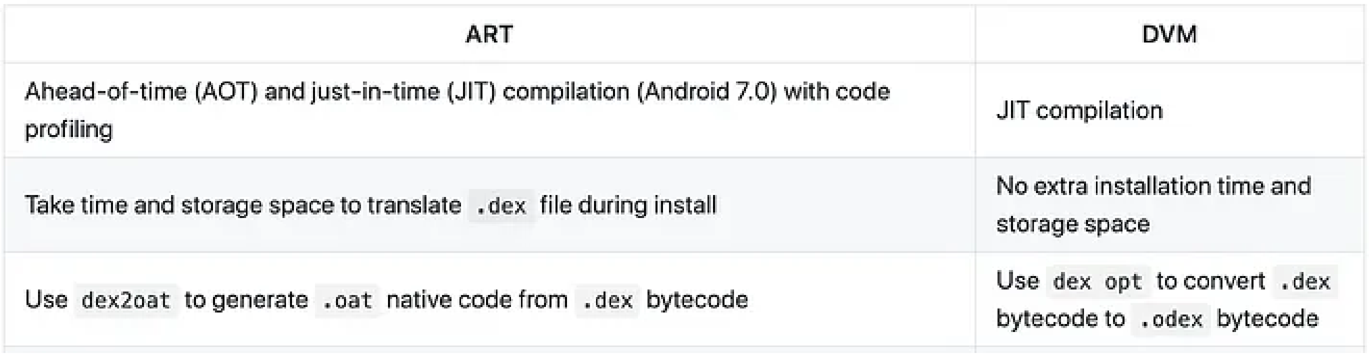
JIT	AOT
Dynamic translate **part** of bytecode to machine code and **cache in memory** when app run	Statically translate bytecode to machine code at **installation** time and **store in storage**
Small memory	One time event, code execute faster but need extra space and time

JIT vs AOT

Compare with Dalvik VM

Dalvik is a JIT compilation based engine.

The ART includes a JIT compiler (with code profiling). The JIT compiler complements new AOT compiler and improves runtime performance, speed up apps and system updates.

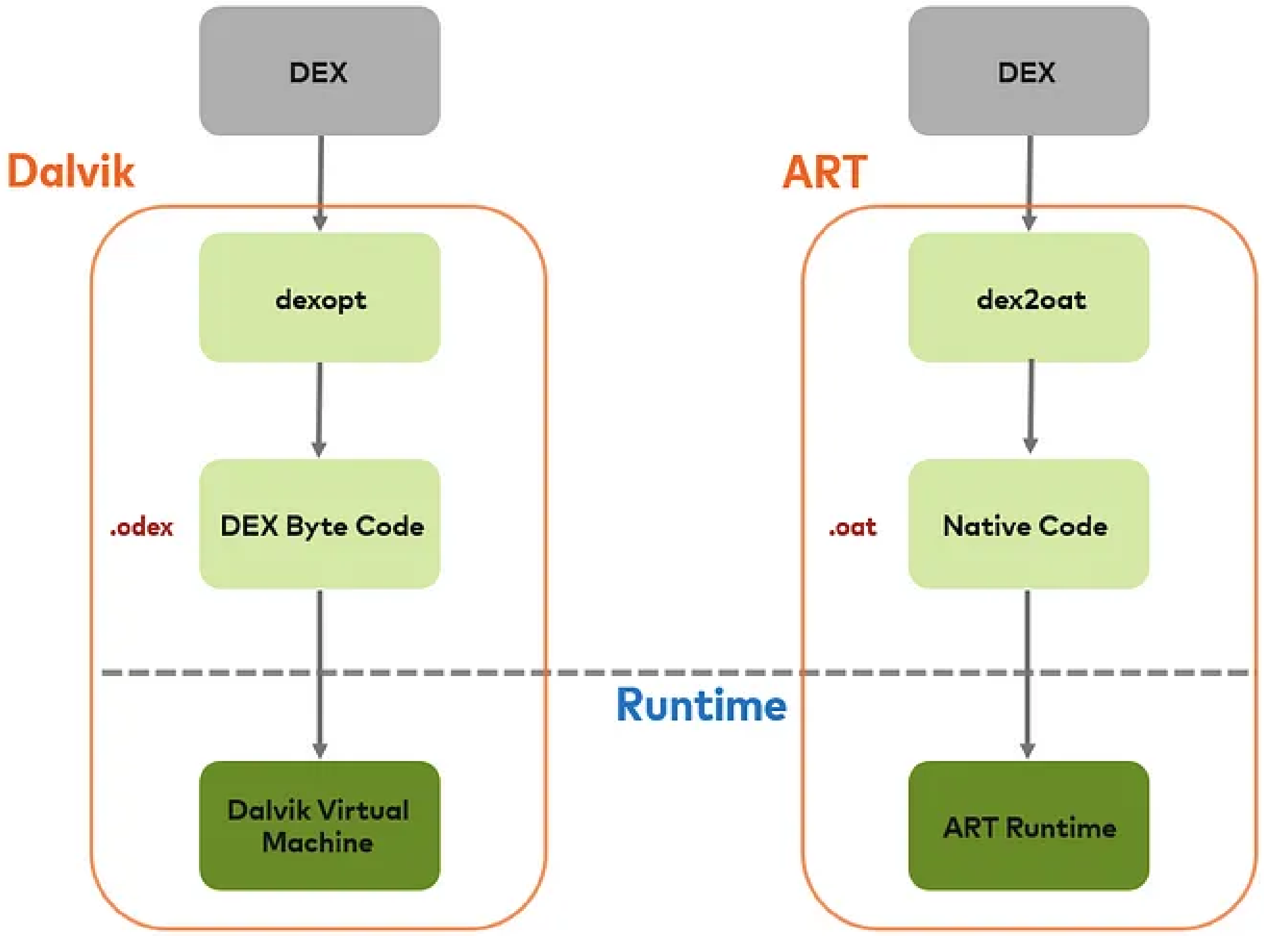


ART	DVM
Ahead-of-time (AOT) and just-in-time (JIT) compilation (Android 7.0) with code profiling	JIT compilation
Take time and storage space to translate `.dex` file during install	No extra installation time and storage space
Use `dex2oat` to generate `.oat` native code from `.dex` bytecode	Use `dex opt` to convert `.dex` bytecode to `.odex` bytecode

ART vs Dalvik VM

Furthermore, ART has many more advantage:

- **Optimized garbage collector**: one GC pause instead of two.
- Loop optimizations: Bounds check, Induction variable is eliminated.
- Faster native calls using `@FastNative` and `@CriticalNative` annotations.
- **Improve battery life.**
- Reduce startup time as native code is directly executed.
- Faster runtime performance because AOT compile at install-time.
- From API 28, convert APK’s DEX files to more compact machine code.
- Better debugging support (dedicated sampling profiler, detailed exceptions reporting, and ability to set watchpoints to specific fields).



Dalvik VM vs ART flows

Optimized Garbage Collector

Garbage Collector (GC) can impact app's performance with "stop-the-world" events, which resulting in frozen frames, poor UI responsiveness. The default GC plan is the CMS (concurrent mark sweep).

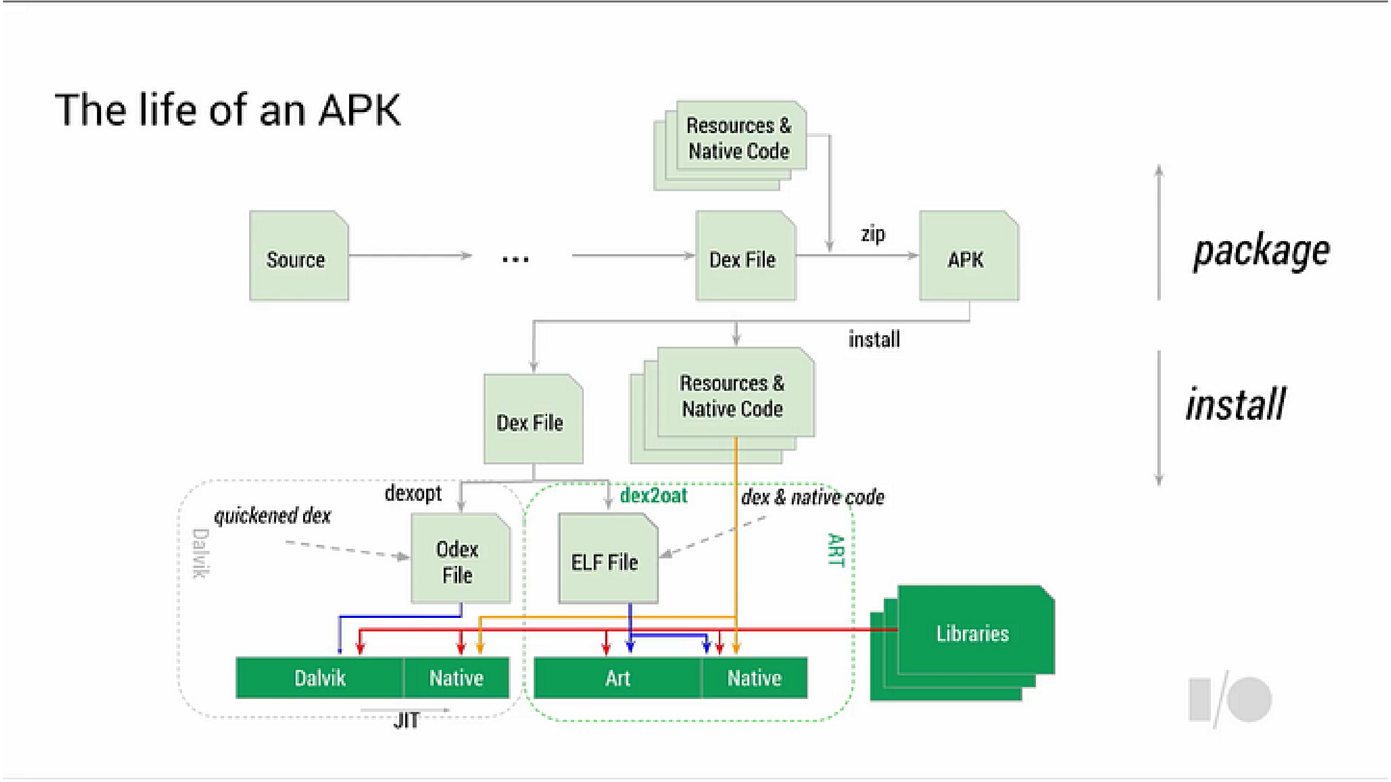
The ART improves GC in several ways:

- **The number of pauses is reduced from 2 to 1 compared to Dalvik.** Dalvik's first pause — which mostly root Marking — is done concurrently in ART by getting the threads to mark their own roots.
- Parallelized processing in the second pause (before Sweeping phase).
- Increased GC throughput enabled by the sticky CMS collector.

- Reduce total time when clean up recently-allocated, short-lived objects.
- Performs heap compaction — when app changes process state to background or cached — to reduce background memory usage.

In Conclusion

Before Android 5.0 (API 21), Android use **Dalvik Virtual Machine (DVM)** — a Process virtual machine — that optimized for mobile environment (memory, battery life, performance,..).



Source: Google I/O

After that, each Android app runs in its own process and with its own instance of the **Android Runtime (ART)** — an app runtime environment used by Android OS. Replacing Dalvik, ART performs the translation of the app's bytecode into native instructions that are later executed by the device's runtime environment.