**ISSUE NAME:** Overview of **JVM, DVM, ART, JIT, AOT**

**GIRLSCRIPT WINTER OF CONTRIBUTING’21**

**OVERVIEW**

1. **What is Virtual Machine?**

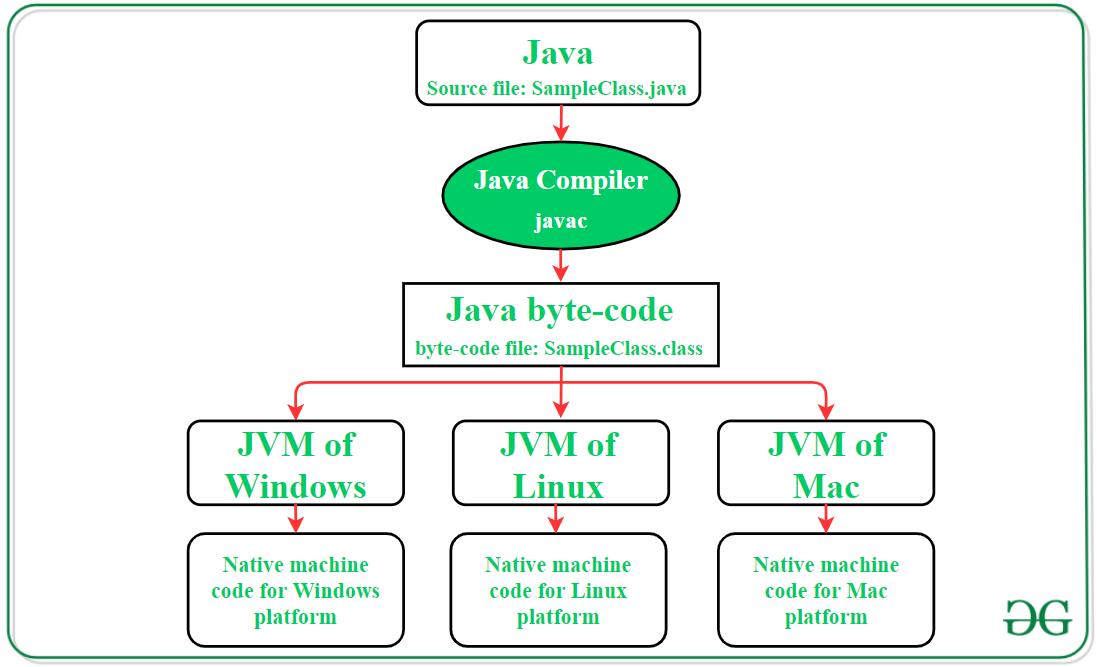
A virtual machine, commonly shortened to just VM, is no different than any other physical computer like a laptop, smart phone or server. It has a CPU, memory, disks to store your files and can connect to the internet if needed. While the parts that make up your computer (called hardware) are physical and tangible, VMs are often thought of as virtual computers or software-defined computers within physical servers, existing only as code.

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

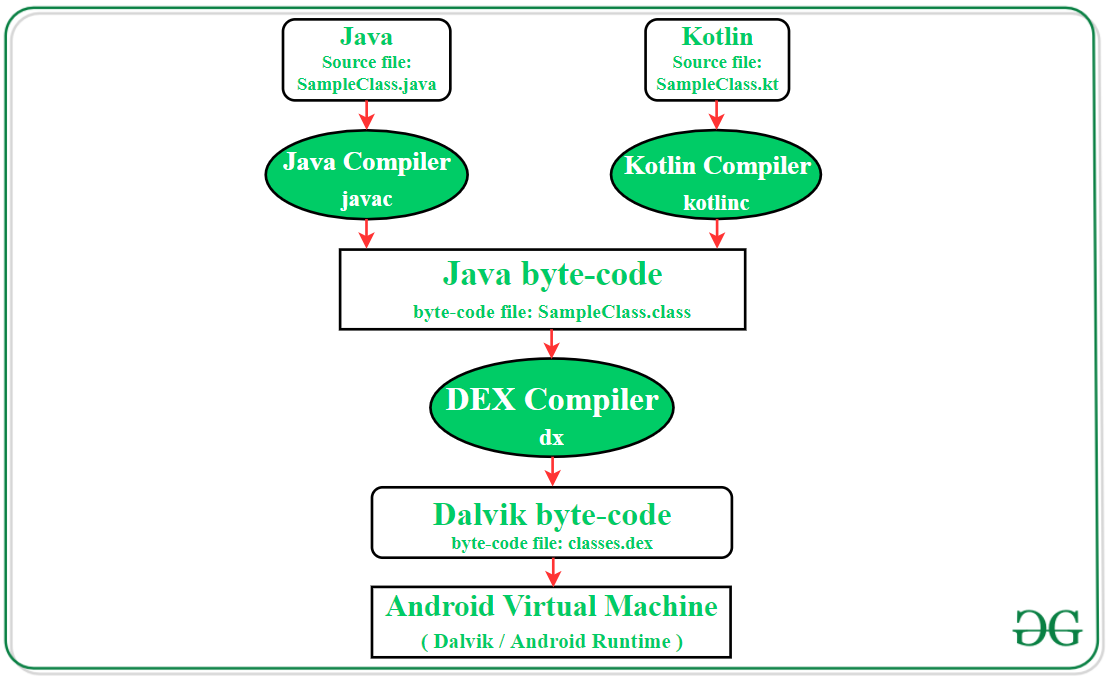
* System virtual machines ([full virtualization](https://en.wikipedia.org/wiki/Full_virtualization) VMs) provide a substitute for a real machine.
* Process virtual machines are designed to execute computer programs in a platform-independent environment.

1. **Give some overview of JVM, DVM, ART, JIT,AOT**

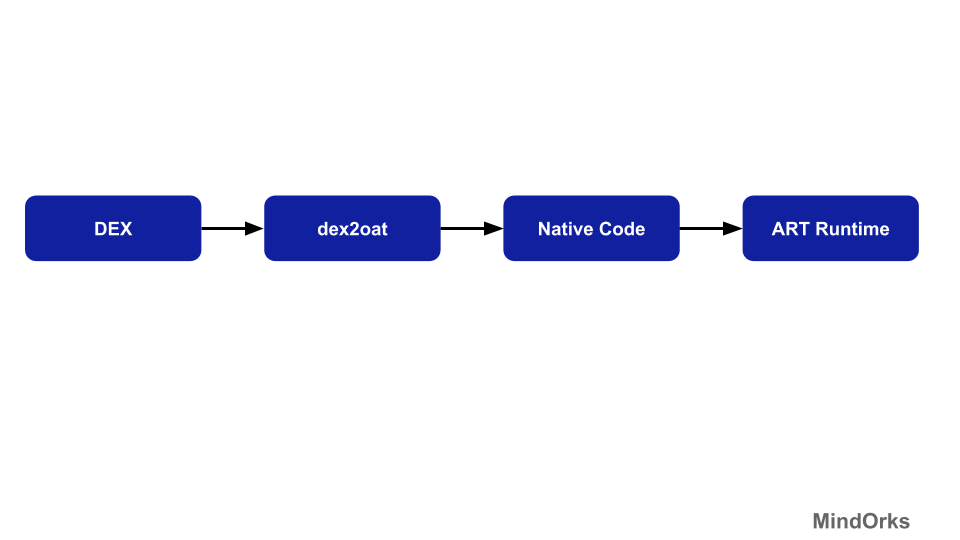
**a.) JVM:** JVM is the virtual machine that runs java code on different platforms. It acts as an abstract layer between the program and the platform on which the java code is running. The portability of Java code is possible only because of the JVM. The javac compiler converts the source code file(.java file) into an intermediate java bytecode format which is machine/platform independent. This intermediate file is then provided to the target machine/platform where it gets translated into machine code. JVM supports multiple host architecture and it is the reason why Java applications are called WORA(Write Once Run Anywhere).



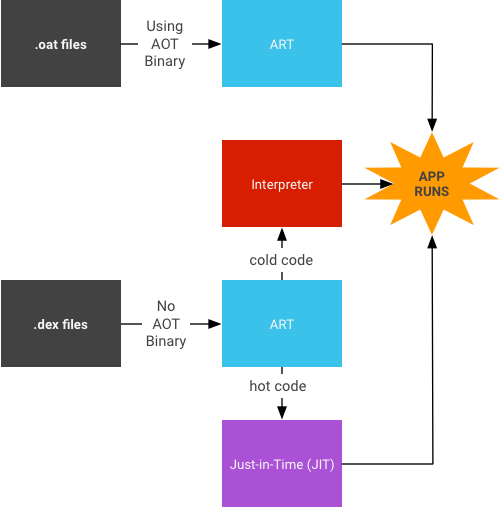
**b.) DVM: DVM** stands for “Dalvik Virtual Machine”. It is a Register-Based virtual machine. It was designed and written by Dan Bornstein with contributions of other Google engineers as part of the Android mobile phone platform. The Dalvik virtual machine was named after Bornstein after the fishing village “Dalvík” in Iceland, where some of his ancestors used to live.



**c.) ART:** ART or Android Runtime is an Android runtime that uses Ahead Of Time (AOT). By using AOT, what is does is it converts or compiles the whole High-level language code into Machine level code and at the time of installation of the app and not dynamically as the application runs(like in case of Dalvik). By compiling the whole code during installation results in no lag that we see when we run our app on our device. By doing so, the compilation becomes very faster.



**d.) JIT:** JIT stands for "Just In Time," and we use it to describe a Dalvik JIT compiler, which was added to Android with the 2.2 release. It compiles bytecode into native machine code at runtime. Essentially it takes the code for an app, analyzes it and converts it into something that runs faster. It does all this while the application is running, and that's where the "just in time" tag comes from. The JIT compiler designed for Android also can do this with a very short "warm up" time, meaning it doesn't take very long to analyze the code before it starts working. It stores information in a cache in your phone's RAM, which means it's not an ideal solution for devices with low memory.



**e.) AOT:** AOT stands for Ahead of Time compilation, and compiles your code, to the native platform, dependent upon the architecture. The reason you would use AOT, is because it has a drastically reduces startup time, and app performance.

**3. JVM V/s DVM**

**Difference Table**

| JVM(Java Virtual Machine) | DVM(Dalvik Virtual Machine) |
| --- | --- |
| Stack-based VM that performs arithmetic and logic operations through push and pop operands. The result of operations is stored in stack memory. | Register-based VM that uses registers located in the CPU to perform arithmetic and logic operations. |
| Java source code is compiled into Java bytecode format(.class file) that further translates into machine code. | Source code files are first of all compiled into Java bytecode format like JVM. Further, the **DEX compiler (dx tool)** converts the Java bytecode into Dalvik bytecode(classes.dex) file that will be used to create the **.apk file**. |
| More information is required to the VM for data loading and manipulation as well as method loading in the stack data structure. | Instruction size is larger as it needs to encode the source and destination register of the VM. |
| Compiled bytecode size is compact because the location of the operand is implicitly on the operand stack. | Compiled bytecode size is larger as each instruction needs all implicit operands. |
| The executable file for the device is **.jar file**. | The executable file for the device is **.apk file**. |
| A single instance of JVM is configured with shared processes and memory space in order to run all deployed applications. | The device runs multiple DVM instances with a separate process in shared memory space to deploy the code of each application. |

**4. Why Android OS used DVM instead of JVM ?**

One of the main reasons of using DVM in android is because it follows the register based model and it is much faster than stack based model while JVM follows the stack based model which takes a lot of memory and also slower than DVM.

There are couple of reasons why Google not choose JVM over DVM, so let’s understand each one of them one by one.

* Though JVM is free, it was under GPL license, which is not good for Android as most the Android is under Apache license.
* JVM was designed for desktops and it is too heavy for embedded devices.
* DVM takes less memory, runs and loads faster compared to JVM.

**5. Why Android deprecated DVM & started to use ART?**

In Android Lollipop, DVM has been replaced by ART. Because DVM converts bytecode every time you launch a specific app. But ART converts it just once at the time of app installation.

**ART is extremely fast and smooth**. Apps are very snappy and responsive. Any comparison between Dalvik and ART, will surely make the ART device win by a significant margin.

**Dalvik used Just-In-Time (JIT) approach** in which the compilation was done on demand. All the dex files were converted into their respective native representations only when it was needed.

But **ART uses the Ahead-Of-Time (AOT) approach**, in which the dex files were compiled before they were demanded. This itself massively improves the performance and battery life of any Android device.

**6. DVM V/s ART**

### Difference Between DVM and ART

| **DALVIK VIRTUAL MACHINE** | **ANDROID RUN TIME** |
| --- | --- |
| Faster Booting time | Rebooting is significantly longer |
| Cache builds up overtime | The cache is built during the first boot |
| Occupies less space due to JIT | Consumes a lot of storage space internally due to AOT |
| Works best for small storage devices | Works best for Large storage devices |
| Stable and tested virtual machine | Experimental and new – not much app support comparatively |
| Longer app loading time | Extremely Faster and smoother Faster and app loading time and lower processor usage |
| Uses JIT compiler(JIT: Just-In-Time) Thereby resulting in lower storage space consumption | Uses AOT compiler(Ahead-Of-Time) thereby compiling apps when installed |
| Application lagging due to garbage collector pauses and JIT | Reduced application lagging and better user experience |
| App installation time is comparatively lower as the compilation is performed later | App installation time is longer as compilation is done during installation |

**What’s Better?**

**It depends on the device and one’s own device and preferences: in the long run, ART is better, but the apps do get bigger, requiring large storage space over time, unlike flappy bird which was just 1MB and required less space. Therefore, the devices in the future need to expand storage if ART is going to dominate the market in the next few years.**

**7. Why Android is reintroducing JIT with ART?**

This is done in order to tackle problems such as initial installation time and memory. Using the Hybrid Runtime, there won't be any compilation during install, and applications can be started right away, the bytecode is interpreted. Now with ART and the new JIT the application installation is faster.

**8. JIT V/s AOT**

JIT compilers have way more information at their disposal than AOT compilers. Static analysis is impossible in the general case (nearly everything interesting you would want to know about a program can be reduced to either the Halting Problem or Rice's Theorem), and hard even in the special case. JIT compilers don't have this problem: they don't have to statically analyze the program; they can observe it dynamically at runtime.

Plus, a JIT compiler has techniques at its disposal that AOT compilers don't, the most important one being de-optimization. Now, you might think, we is de-optimization important for performance? Well, if you can de-optimize, then you can be over-aggressive in making optimizations that are actually invalid (like inlining a method call that may or may not be polymorphic), and if it turns out that you are wrong, you can then de-optimize back to the un-inlined case (for example).

However, there's the problem of resources: an AOT compiler can take as much time as it wants, and use as much memory as it wants. A JIT compiler has to steal its resources away from the very program that the user wants to use right now.

References:

1. <https://azure.microsoft.com/en-in/overview/what-is-a-virtual-machine/#overview>

2. <https://medium.com/programming-lite/android-core-jvm-dvm-art-jit-aot-855039a9a8fa>

3. <https://medium.com/android-news/virtual-machine-in-android-everything-you-need-to-know-9ec695f7313b>

4. <https://www.geeksforgeeks.org/what-is-dvmdalvik-virtual-machine/>

5. <https://www.geeksforgeeks.org/difference-between-jvm-and-dvm/>

6. <https://blog.mindorks.com/what-are-the-differences-between-dalvik-and-art>

7. <https://www.journaldev.com/23464/android-runtime-dvm-vs-art-aot-vs-jit>

8. <https://towardsdatascience.com/jvm-vs-dvm-b257229d18a2>

9. <https://stackoverflow.com/questions/31957568/what-is-difference-between-dvm-and-art-why-dvm-has-been-officially-replaced-wi>

10. <https://stackoverflow.com/questions/40336455/difference-between-aot-and-jit-compiler-in-the-art>