Does JVM generate bytecode or run bytecode?

Asked 6 years, 3 months ago Active 1 year, 11 months ago Viewed 14k times

5

I'm little confused here, does the JVM represents the bytecode (generate it) or it's just it load the compiled .class files (bytecode) into memory?! or JVM is just specifications to run the bytecode in a platform independent way?! thank you very much.

java compiler-construction jvm javac bytecode

1

Share Follow

edited May 17 '17 at 11:32

asked Aug 14 '15 at 13:16

Mohamed Seif
364 1 2 13

2 JDK produces bytecode with the compiler (javac), JVM runs it. – BackSlash Aug 14 '15 at 13:17

9 Answers

Active | Oldest | Votes

7

The Java compiler (javac) turns your human-readable code into bytecode, which is then running in a JVM.

From the oracle docs:

> A program has to be converted to a form the Java VM can understand so any computer with a Java VM can interpret and run the program. Compiling a Java program means taking the programmer-readable text in your program file (also called source code) and converting it to bytecodes, which are platform-independent instructions for the Java VM.

Share Follow

answered Aug 14 '15 at 13:21

appleseedexm
116 2

7

1. JVM = JIT Compiler + Java Interpreter + Garbage Collector
2. JRE = JVM + Library Classes
3. JDK = JRE + Development Tool

Sun JVM is written in C and Oracle JVM is written in C++

Java compiler javac converts source code into bytecode. JIT Compiler and Java Interpreter inside JVM convert the bytecode into corresponding machine code.

In java, only the source code(.java files) and bytecodes(.class files) are available. And we can't save the machine codes(.exe files) as because .exe files can only be formed at Run Time and vanished from RAM as soon as the program is executed completely.

In our system both the javac.exe(for compiling java source code, eg: javac HelloWorld.java) and java.exe(for executing java bytecode by JVM, eg: java HelloWorld) are called, which are available in .exe formats only(javac.exe and java.exe). So the Java Compiler javac and JVM were not written in Java.If those were written in Java then those might have been available in javac.class and java.class format.

javac comes under JDK and not under JVM.Remember,JVM only works during Run Time means after the compilation of Source Code into Byte Code..but before that javac compiles the source code into byte code. JVM converts bytecode into corresponding machine code by JIT Compiler and Java Interpreter.

For different Operating Systems different JDK and JRE softwares are available by Oracle Corporation; So both the JVM(coming under JRE) and javac Compiler(coming under JDK) are Platform Dependent.So it is confirmed that javac Compiler and JVM are not written in Java.Because Java language is always Platform Independent.

Share Follow

answered Apr 7 '18 at 15:07

Sasmita Nayak

71 1 3

i think this answer is the exact response to the question – divine Nov 7 '19 at 4:28

4

When you say `javac file`, the Java Compiler (called javac) will convert your code into an intermediate form (bytecode). It does not convert it directly to machine language, which is platform specific, so that you can give the class files (bytecode) to anyone on any platform. This is how "Write once, Run anywhere" works. Instead of compiling to platform specific machine language, it compiles to a generic bytecode.

When you say `java file`, the JVM will take the bytecode and convert it to native machine language in chunks "on the fly" (during runtime) and execute them. It does this using a JIT compiler (which might be a source of confusion, since this is NOT the same as javac). By the way, the JDK is not the same as javac. The JDK is an SDK (software development kit) that contains everything in Java, including the JVM and javac.

Share Follow

answered Aug 17 '15 at 21:50


M3579
870 2 11 22

1

JVM runs the bytecode, Java compiler generates it.

However, applications can generate the bytecode while they run, but the generated bytecode is again executed by the JVM.

Share Follow

edited May 23 '17 at 12:10

Community Bot
1 1

answered Aug 14 '15 at 13:20

Dragan Bozanovic
22k 4 39 104

1

JVM, depending on platform , convert the byte code to m/c code. More precisely, **JIT (just-in-time) compiler** inside JVM does this. Byte code is generate by javac.exe. And java.exe converts this byte code to m/c code with the help of **jvm.dll** (in windows).

Share Follow

edited Aug 15 '15 at 17:25

answered Aug 14 '15 at 13:43

Rohit sharma
71 4

2 Minor nitpick: a JIT compiler typically generates machine code, not assembly code. Assembly code is a textual representation of (something very close to) machine code. – Martin Törnwall Aug 15 '15 at 16:52

0

JDK(javac) generates the byte code(.class files). Now this byte code can be run on any platform by the JVM of that platform.

Share Follow

answered Aug 14 '15 at 13:47

abhiabhi
56 5

0

In short **Java Virtual Machine runs / interprets / translates Bytecode into native machine code. It does not generate the bytecode.** And we can consider JVM as an interpreter.

Below will help to understand the above,

Bytecode, also termed portable code or p-code, is a form of instruction set designed for efficient execution by a software interpreter. It is something in between a human readable source code and a machine readable machine code. A bytecode program may be executed by parsing and directly executing the instructions, one at a time. This kind of bytecode interpreter is very portable. Some systems, called dynamic translators, or just-in-time (JIT) compilers, translate bytecode into machine code as necessary at runtime. This makes the virtual machine hardware-specific, but doesn't lose the portability of the bytecode. Refer this.

A **compiler is computer software that transforms computer code written in one programming language (the source language) into another programming language** (the target language). A Java compiler is a compiler for the programming language Java. The most common form of ***output from a Java compiler is Java class files containing platform-neutral Java bytecode***.

For Java it has a dedicated compiler called Javac. **Javac Compiles your Java code into Bytecode**. And then we have the **JVM which uses that bytecode**.

Share Follow

answered Apr 4 '18 at 20:00

prime
12.1k 12 80 116

0

enter image description hereThere is an inbuilt tool now available called Java Visual VM(after JDK version 6, update 7) which you will find in bin, the directory where javac.exe exists. Just open this tool which monitors processes that use JVM (profiler). It will show javac,eclipse etc as a Java process running on JVM if you execute it while Visual VM is open.

Share Follow

edited Nov 16 '18 at 17:44

answered Nov 16 '18 at 15:56

Navanee Subburaj
41 3

0

Jdk generates the byte code for a given file with .java extension. JVM converts the byte code into machine language and then executes it.

Share Follow

answered Dec 13 '19 at 10:11

Vrushali Shirodkar
1