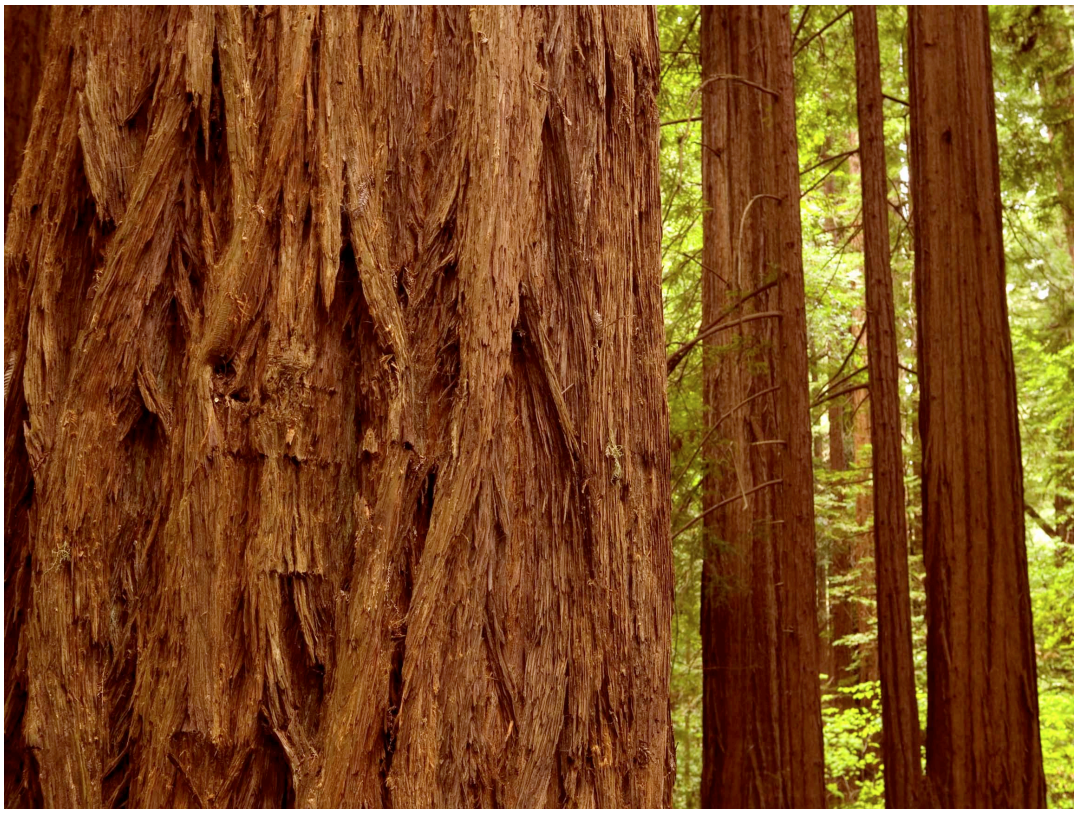
Section 1.4. Java's Magic: Bytecode, Java Virtual Machine, JIT,



JRE and JDK

This section clearly explains the Java's revolutionary features in the programming world. Java basic terminology and concept is explained in this section.

Java's Magic: Bytecode, Java Virtual Machine, JIT, JRE and JDK

Objective:

This section clearly explains the Java's revolutionary features in the programming world. Java basic terminology and concept is explained in this section.

Overview:

Lets say, if Java is a human speaking Spanish language and we ask a person who does not speak Spanish to read this language. It would be impossible for him to read, write or understand this language because he cannot speak this language.

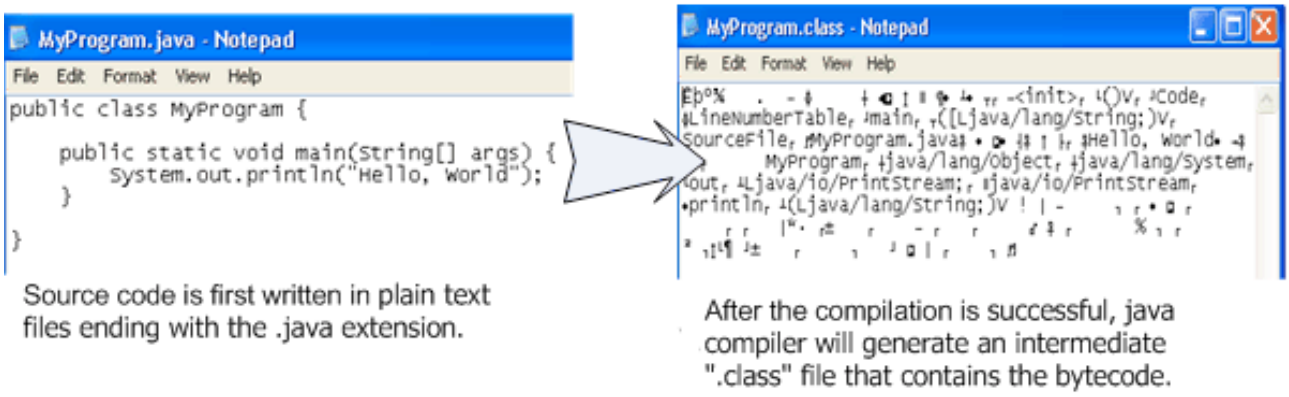
In order to make this language understandable to any person we need a interpreter who helps to interpret this Spanish language into that person understandable language.

Similarly, to make a programming language understandable to any platform, device or operating system there is a need for such interpreter which can interpret the developers code into machine specific instructions.

Unlike, C and C++ which generated the compiled code into machine specific instructions made the program impossible for various platform to understand or execute.

The key advantages of Java programming language is that the compiled code of java is not machine specific instructions but rather an intermediate code called as **ByteCode**.

Bytecode is the code generated after the Java programs are compiled. This is the intermediate representations of Java programs. Bytecode is NOT executable code like ".exe" but this is java propriety intermediate code. Bytecode is NOT machine understandable language.



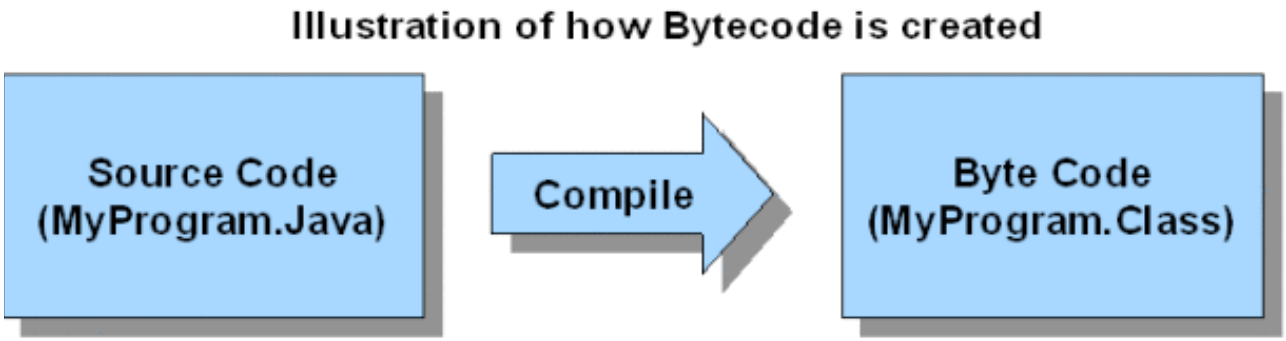
Bytecode is a highly optimized set of instructions designed to be executed by the Java run-time system (interpreter) which is called the Java Virtual Machine (JVM). This provides greater level of flexibility for the developers to implement the logic specific to JVM rather to any platform or device.

Job of JVM is to read this bytecode and convert into machine dependent instructions. So, JVM's needs to be platform specific but not the developers code. JVM is an interpreter for Bytecode

So, for any system to support java programs it is mandatory that JVM needs to be pre-installed on that machine. With every new device or platform in the market, Oracle provide JVM release which open the way for java programs to execute on that device. JVM implementation will differ from platform to platform but all interpret the same bytecode.

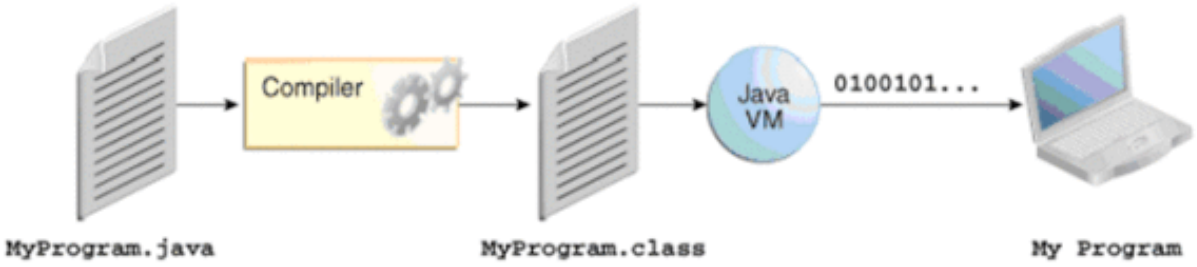
This key concept makes the java program secure as the bytecode must be executed under the control of JVM and the implementation of JVM is done by the product team. This way JVM does not allow any security breach to happen from the java programs and can monitor the illegal behavior of the programs.

ByteCode Platform Independent



Bytecode generated after compiling in Mac, Windows, Linux or Unix will be same which makes Bytecode **platform independent**. So, Bytecode compiled in one platform can be executed into another platform.

Java Virtual Machine (JVM) Definition:



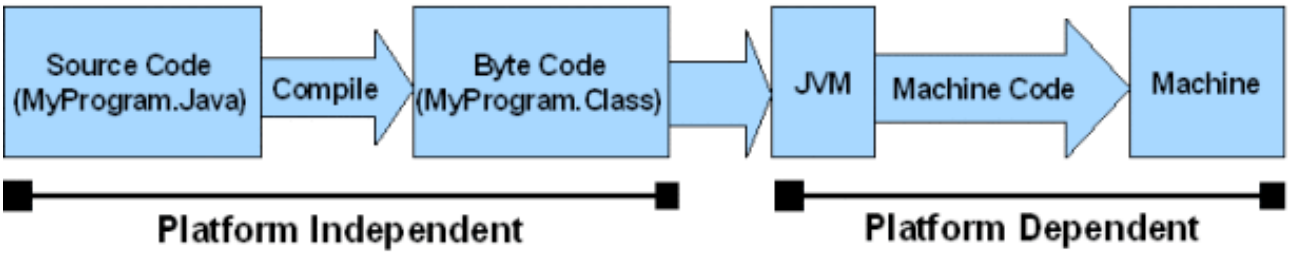
The Java Virtual Machine (JVM) is an abstract computer, on which the byte code can be executed.

Or

Java Virtual Machine (JVM) interprets the byte code into the machine code depending upon the underlying operating system and hardware combination.

JVM does not know anything about Java program rather it reads the bytecode, interprets the code and executes the code.

Bytecode Platform Independent vs Java Virtual Machine (JVM) Platform Dependent:



JVM is platform dependent that means there are different implementation of JVM on different OS.

Java code / Bytecode is always the same on different OS.That makes java program as platform independent.

JVM Implementation:

JVM is platform dependent that means there are different implementation of JVM on different OS.

Type of JVM implementations:
The primary reference Java VM implementation is HotSpot, produced by Oracle Corporation.

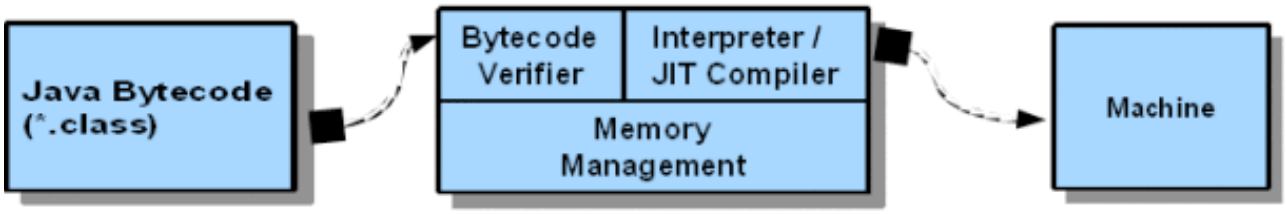
Other list of JVM for various O/S and hardware:

- Hewlett-Packard, Java for HP-UX, OpenVMS, Tru64 and Reliant (Tandem) UNIX platforms
- J9 (IBM), for AIX, Linux, MVS, OS/400, Pocket PC, z/OS
- JBlend, (Aplix) is a Java ME implementation
- JRockit (originally from Appeal Virtual Machines) acquired by Oracle for Linux, Windows and Solaris
- Mac OS Runtime for Java (MRJ)
- Microsoft Java Virtual Machine (discontinued in 2001)

SAPJVM (SAP) is a licensed and modified SUN JVM ported to all supported platforms of SAP NetWeaver, started as Java 5, in the meantime Java 6 compatible (Windows i386, x64, IA-64; Linux x86, IA-64, PowerPC; AIX PowerPC; HP-UX SPARC IA-64; Solaris SPARC x86-64; i5/OS PowerPC)

Java's Magic – Just In Time (JIT):

Just in time (JIT) is a part of Java Virtual Machine (JVM) architecture. The job of JIT inside JVM is to compile bytecode into machine executable code in real time, on a piece-by-piece, demand basis.

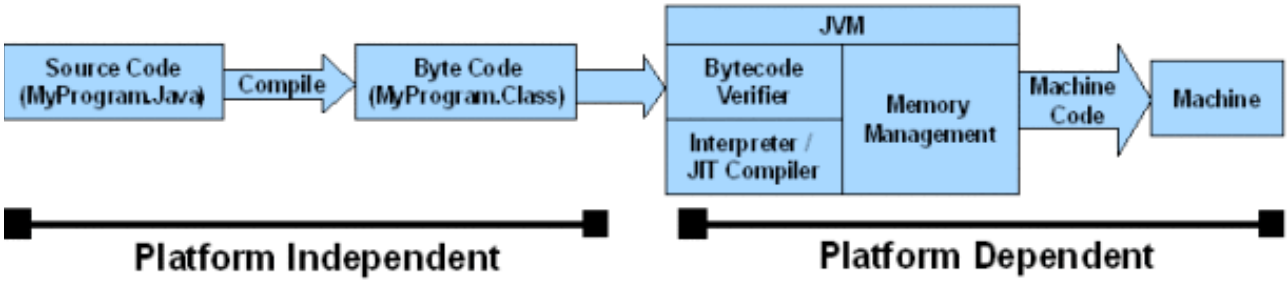


JVM Internal Architecture

When Java programs are executed, JVM does not read the entire Bytecode and converts it into machine instructions. If JVM tries to do this approach then the program execution time will be delayed for hours. Java has overcome the latency of program execution time by interpreting the required bytecode and keep the rest of the code aside.

Just in time (JIT) helps to compile code that is only needed and at the same time boost the program performance. Whether the Java programs are interpreted traditional way or on the fly the functionality and features like portability and security remains the same.

Illustration of how Bytecode is verified and interpreted by JIT

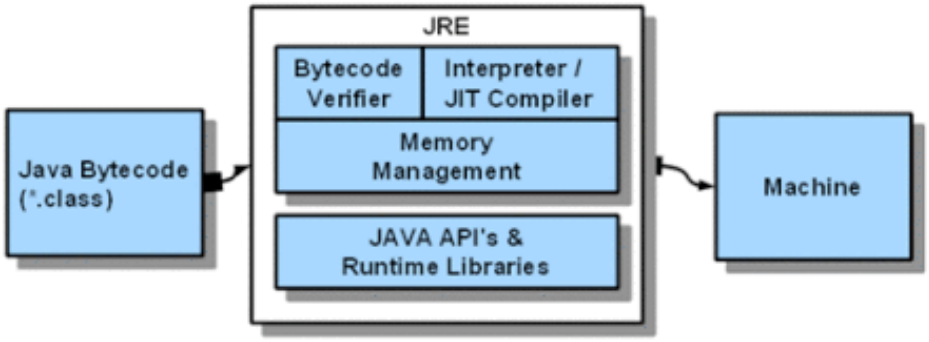


Summary: Java compiler converts the Java source code that you write into a binary program consisting of bytecodes. Bytecodes are machine instructions for the Java Virtual Machine.

When you execute a Java program, a program called the Java interpreter (JIT) inspects and deciphers the bytecodes into machine executable language.

Java's Magic – Java Runtime Environment (JRE):

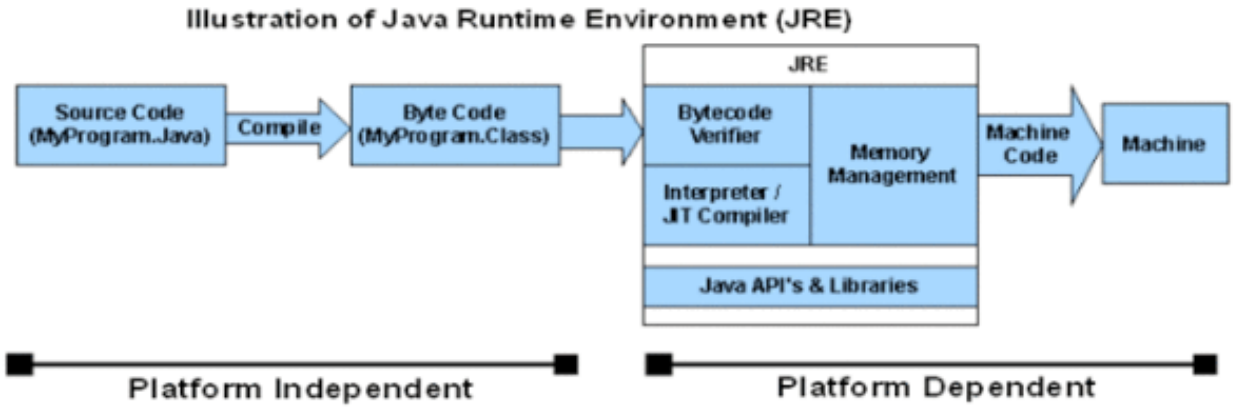
As we discussed above, Java programs cannot be executed on machine without the JVM installed on machine. Java Runtime Environment (JRE) is a software which we can download and install on the any operating system like Windows, Mac or Linux.



JRE is combination of JVM and Java Application Programming Interface (Java API). Java API are set of tools and libraries that is required by the JVM to execute the java programs.

Thus, Java Runtime Environment provides an environment to execute java programs on the computer.

JRE = JVM + Java API's (like util, math, lang, awt, swing etc) + Runtime libraries.

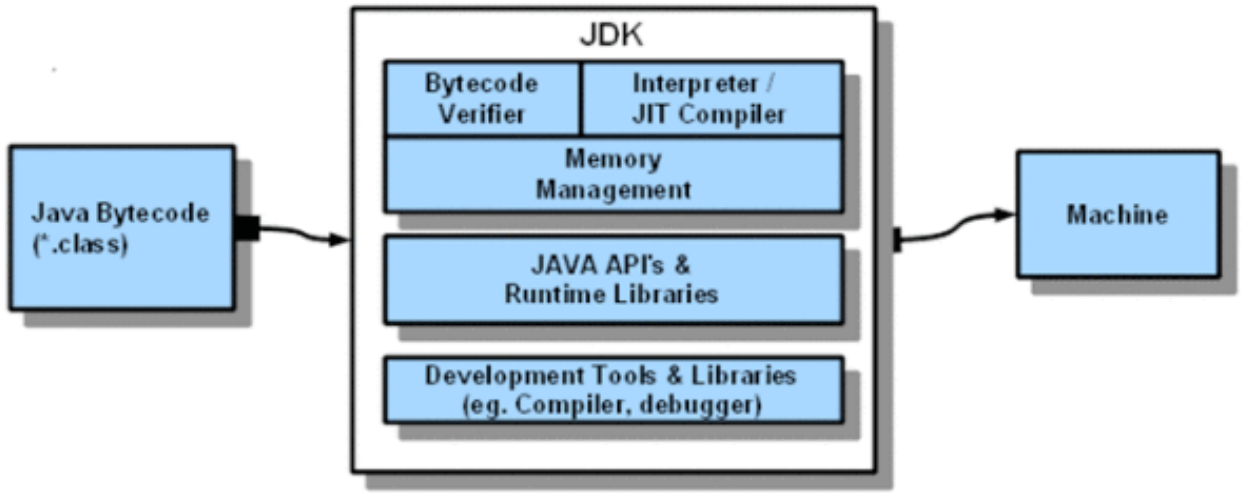


JRE does NOT contain any development tools such as compiler, debugger, etc. and it is NOT for development purpose.

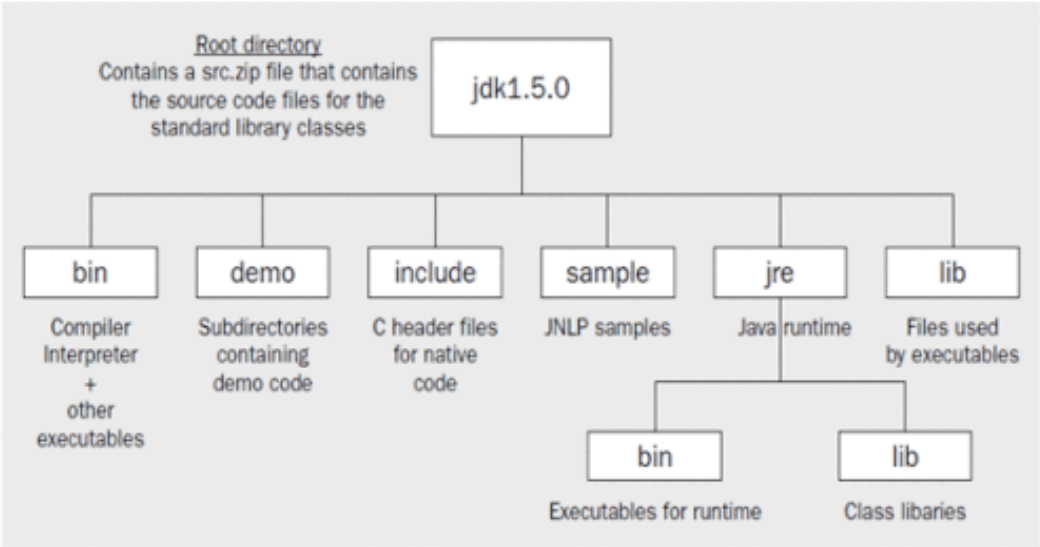
Java's Magic – Java Development Kit (JDK):

Java Development Kit (JDK) is a set of development tools installed on the local machine to write and compile Java programs.

JDK = JRE + Java Development Tools + Libraries



Java Folder structure and usage:



Summarize:

- Java programs are written in ".java" file.
- Bytecode: Bytecode is the code generated after the java program is compiled.
- Java Virtual Machine (JVM): This is virtual machine which reads the bytecode and interprets into machine code depending upon the underlying operating system and hardware combination.
- Just In Time (JIT): Just in time compiler is part of the Java Virtual Machine (JVM) and it compiles bytecode into executable code in real time, on a piece-by-piece, demand basis.
- Java Runtime Environment (JRE): Java Runtime Environment provides an environment to execute java programs on the computer.
- Java Development Kit (JDK): Java development Kit is the development tools and libraries that are required to develop java programs.
- Java Compiler: This is the compiler tool that compiles and convert the ".java" code into ".class" bytecode.
- Java Interpreter: The job of interpreter is to read the bytecode and convert into machine dependent instructions to execute.