Differences Between JDK, JRE and JVM

Understanding the **difference between JDK, JRE, and JVM** plays a vital role in understanding how Java works and how each component contributes to the development and execution of Java applications.

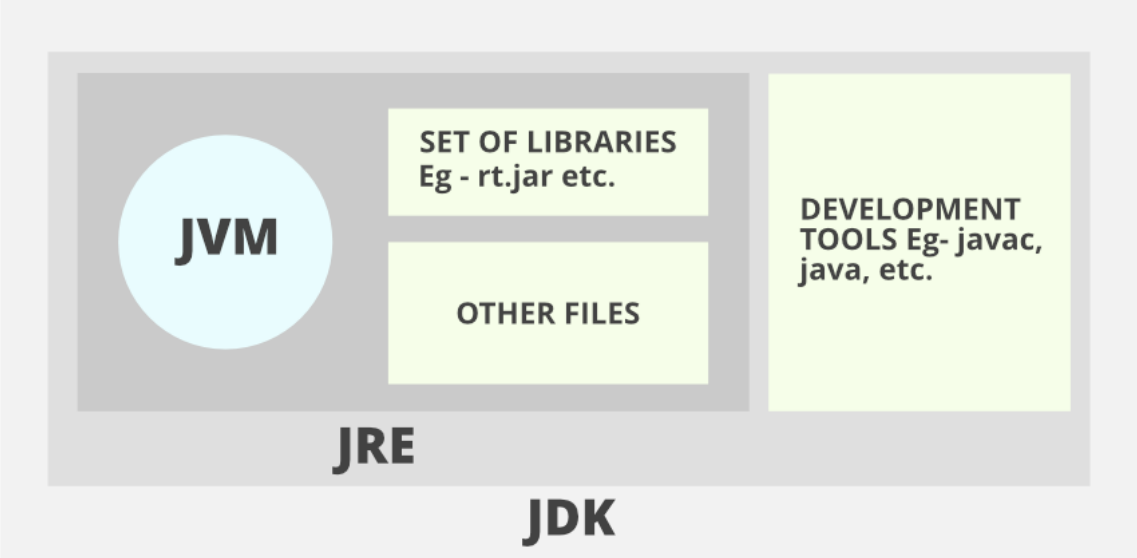
- **JDK:** JDK stands for Java Development Kit. It is a set of development tools and libraries used to create Java programs. It works together with the JVM and JRE to run and build Java applications..
- **JRE:** JRE stands for **Java Runtime Environment,** and it provides an environment to run Java programs on the system. The environment includes Standard Libraries and JVM.
- **JVM:** JVM stands for **Java Virtual Machine.** It's responsible for executing the Java program.

JDK vs JRE vs JVM

Aspect	JDK	JRE	JVM
Purpose	Used to develop Java applications	Used to run Java applications	Executes Java bytecode
Platform Dependency	Platform-dependent (OS specific)	Platform-dependent (OS specific)	JVM is OS-specific, but bytecode is platform-independent
Includes	JRE + Development tools (javac, debugger, etc.)	JVM + Libraries (e.g., rt.jar)	ClassLoader, JIT Compiler, Garbage Collector
Use Case	Writing and compiling Java code	Running a Java application on a system	Convert bytecode into native machine code

Note: *The JVM is platform-independent in the sense that the bytecode can run on any machine with a JVM, but the actual JVM implementation is platform-dependent. Different operating systems (e.g., Windows, Linux, macOS) require different JVM implementations that interact with the specific OS and hardware*

We have discussed the core differences, now let's take a closer look at each component. Let, us discuss them in brief first and interrelate them with the image below proposed.



JDK (Java Development Kit)

The JDK is a software development kit that provides tools to develop and run Java applications. It includes two main components:

- Development Tools (to provide an environment to develop your java programs)
- JRE (to execute your java program)

Note:

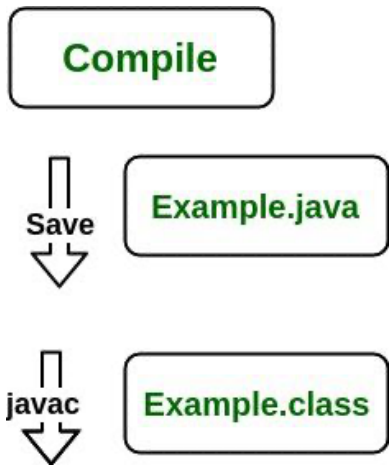
- *JDK is only for development (it is not needed for running Java programs)*
- *JDK is platform-dependent (different version for windows, Linux, macOS)*

Working of JDK

The JDK enables the development and execution of Java programs. Consider the following process:

- **Java Source File (e.g., Example.java)**: You write the Java program in a source file.
- **Compilation:** The source file is compiled by the Java Compiler (part of JDK) into bytecode, which is stored in a .class file (e.g., Example.class).

- **Execution:** The bytecode is executed by the JVM (Java Virtual Machine), which interprets the bytecode and runs the Java program.



Working of JDK

Note: From above, media operation computing during the compile time can be interpreted.

The following actions occur at runtime as listed below:

- Class Loader
- Byte Code Verifier
- Interpreter
    - Execute the Byte Code
    - Make appropriate calls to the underlying hardware

JRE ((Java Runtime Environment)

The JRE is an installation package that provides an environment to **only run(not develop)** the Java program (or application) onto your machine. JRE is only used by those who only want to run Java programs that are end-users of your system.

Note:

- *JRE is only for end-users (not for developers).*
- *JRE is platform-dependent (different versions for different OS)*

Working of JRE

When you run a Java program, the following steps occur:

- **Class Loader**: The JRE's class loader loads the .class file containing the bytecode into memory.

- **Bytecode Verifier**: JRE includes a bytecode verifier to ensure security before execution
- **Interpreter**: JVM uses an interpreter + JIT compiler to execute bytecode for optimal performance
- **Execution**: The program executes, making calls to the underlying hardware and system resources as needed.

JVM (Java Virtual Machine)

The **JVM** is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for executing the java program line by line, hence it is also known as an ***interpreter*****.**

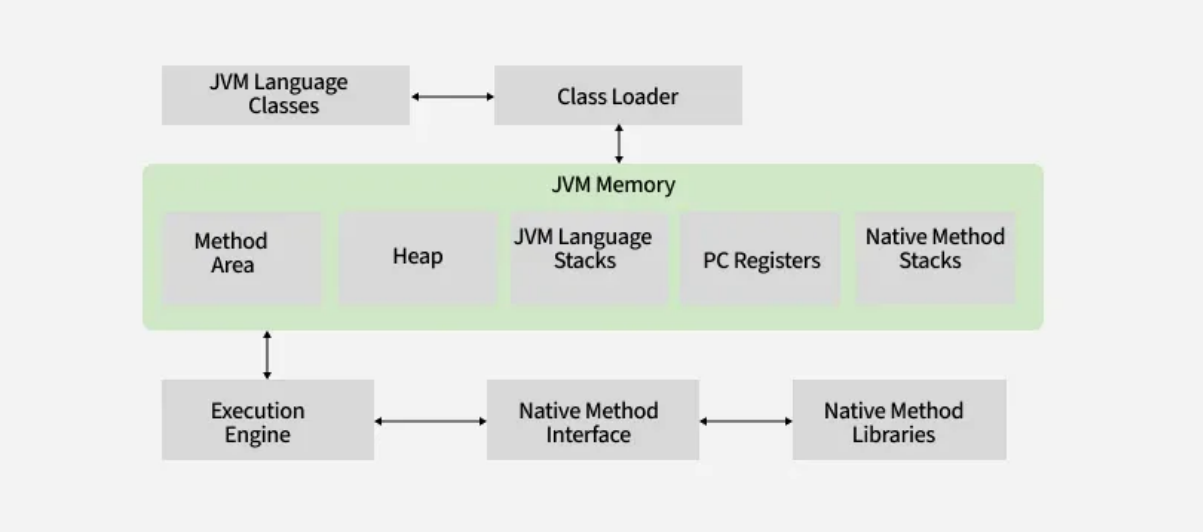
Note:

- *JVM is platform -dependent (different JVMs for window, linux, macOS).*
- *Bytecode (.class files) is platform-independent (same file runs in any JVM).*
- *While JVM includes an interpreter, modern implementations primarily use JIT compilation for faster execution*

Working of JVM

It is mainly responsible for three activities.

- Loading
- Linking
- Initialization



JVM Working