Difference Between JDK, JRE, and JVM

JDK Vs. JRE Vs. JVM: Explore What is the Difference Between JDK, JRE, and JVM?

Out of all the programming languages, Java is the most common one for the purpose of development today. Most developers use it for mobile and desktop computing, game development, back-end development, etc. Here, JVM, JRE, and JDK play a crucial role in the development process with Java. Before getting a detailed knowledge of this language, we must know the significant difference between JDK, JRE, and JVM. In this article, we will discuss the same. But first, let's get a brief overview of them individually.

What is JDK?

JDK is an abbreviation for Java Development Kit. It is an environment of software development used for developing applets and Java applications. JDK has a physical existence, and it contains JRE + development tools. One can easily install more than one version of JDK on the same computer. The Java developers can make use of it on macOS, Windows, Linux, and Solaris. JDK assists them in coding and running the Java programs.

It is an implementation of any of the given Java Platforms that the Oracle Corporation released:

- Micro Edition
- Enterprise Edition
- Standard Edition

The JDK consists of a private JVM (Java Virtual Machine) along with a few other resources, java (a loader/interpreter), like javac (a compiler), Javadoc (a documentation generator), jar (an archiver), etc., for completing the process of Java application development.

What is JRE?

JRE stands for Java Runtime Environment- also written as Java RTE. It is a set of software tools designed for running other software. It is an implementation of JVM, and JRE provides a runtime environment. In short, a user needs JRE to run any Java program. If not a programmer, the user doesn't need to install the JDK- JRE alone will help run the Java programs.

All the versions of JDK come bundled up with the JRE (Java Runtime Environment). This way, a user doesn't have to download and install JRE on their PC separately. The JRE also exists physically. It consists of a library set + a few more files that the JVM (Java Virtual Machine) deploys at the runtime.

What is JVM?

JVM stands for Java Virtual Machine. It provides a runtime environment for driving Java applications or code. JVM is an abstract machine that converts the Java bytecode into a machine language. It is also capable of running the programs written by programmers in other languages

(compiled to the Java bytecode). The JVM is also known as a virtual machine as it does not exist physically.

JVM is essentially a part of the JRE (Java Run Environment). You cannot separately download and install it. You first need to install the JRE to install the JVM. It is available for many software and hardware platforms. In various distinct programming languages, the compiler functions to produce machine code for specific systems. However, only the Java compiler produces code for a virtual machine- also known as JVM.

All three, JDK, JRE, and JVM, are dependent. It is because each Operating System's (OS) condition is different from one another. But Java is independent of the platform. The JVM has three notions: *implementation*, *instance*, and *specification*.

JVM primarily performs the following tasks:

- Provides runtime environment
- Verifies code
- Loads code
- Executes code

Difference Between JDK, JRE, and JVM

Parameter	**JDK**	**JRE**	**JVM**
Full-Form	The JDK is an abbreviation for Java Development Kit.	The JRE is an abbreviation for Java Runtime Environment.	The JVM is an abbreviation for Java Virtual Machine.
Definition	The JDK (Java Development Kit) is a software development kit that develops applications in Java. Along with JRE, the JDK also consists of various development tools (Java Debugger, JavaDoc, compilers, etc.)	The Java Runtime Environment (JRE) is an implementation of JVM. It is a type of software package that provides class libraries of Java, JVM, and various other components for running the applications written in Java programming.	The Java Virtual Machine (JVM) is a platform-independent abstract machine that has three notions in the form of specifications. This document describes the requirement of JVM implementation.
Functionality	The JDK primarily assists in executing codes. It primarily functions in development.	JRE has a major responsibility for creating an environment for the execution of code.	JVM specifies all of the implementations. It is responsible for providing all of these implementations to the

			JRE.
Platform Dependency	The JDK is platform-dependent. It means that for every different platform, you require a different JDK.	JRE, just like JDK, is also platform-dependent. It means that for every different platform, you require a different JRE.	The JVM is platform-independent. It means that you won't require a different JVM for every different platform.
Tools	Since JDK is primarily responsible for the development, it consists of various tools for debugging, monitoring, and developing java applications.	JRE, on the other hand, does not consist of any tool- like a debugger, compiler, etc. It rather contains various supporting files for JVM, and the class libraries that help JVM in running the program.	JVM does not consist of any tools for software development.
Implementation	**JDK** = Development Tools + JRE (Java Runtime Environment)	**JRE** = Libraries for running the application + JVM (Java Virtual Machine)	**JVM** = Only the runtime environment that helps in executing the Java bytecode.
Why Use It?	Why use JDK? Some crucial reasons to use JDK are: • It consists of various tools required for writing Java programs. • JDK also contains JRE for executing Java programs. • It includes an Appletviewer, Java application launcher, compiler, etc. • The compiler helps in converting the code written in Java into bytecodes. • The Java application launcher helps in opening a JRE. It	Why use JRE? Some crucial reasons to use JRE are: • If a user wants to run the Java applets, then they must install JRE on their system. • The JRE consists of class libraries along with JVM and its supporting files. It has no other tools like a compiler or a debugger for Java development. • JRE uses crucial package classes like util, math, awt, lang, and various runtime libraries.	Why use JVM? Some crucial reasons to use JVM are: • It provides its users with a platform-independent way for executing the Java source code. • JVM consists of various tools, libraries, and multiple frameworks. • The JVM also comes with a Just-in-Time (JIT) compiler for converting the Java source code into a low-level machine language. Thus, it ultimately

	then loads all of the necessary details and then executes all of its main methods.		runs faster than any regular application. • Once you run the Java program, you can run JVM on any given platform to save your time.
Features	Features of JDK • Here are a few crucial features of JDK: • It has all the features that JRE does. • JDK enables a user to handle multiple extensions in only one catch block. • It basically provides an environment for developing and executing the Java source code. • It has various development tools like the debugger, compiler, etc. • One can use the Diamond operator to specify a generic interface in place of writing the exact one. • Any user can easily install JDK on Unix, Mac, and Windows OS (Operating Systems).	Features of JRE • Here are a few crucial features of JRE: • It is a set of tools that actually helps the JVM to run. • The JRE also consists of deployment technology. It includes Java Plug-in and Java Web Start as well. • A developer can easily run a source code in JRE. But it does not allow them to write and compile the concerned Java program. • JRE also contains various integration libraries like the JDBC (Java Database Connectivity), JNDI (Java Naming and Directory Interface), RMI (Remote Method Invocation), and many more. • It consists of the JVM and virtual machine client for	Features of JVM Here are a few crucial features of JVM: • The JVM enables a user to run applications on their device or in a cloud environment. • It helps in converting the bytecode into machine-specific code. • JVM also provides some basic Java functions, such as garbage collection, security, memory management, and many more. • It uses a library along with the files given by JRE (Java Runtime Environment) for running the program. • Both JRE and JDK contain JVM. • It is easily customizable. For instance, a user can feasibly allocate a

		Java HotSpot.	maximum and minimum memory to it. • JVM can also execute a Java program line by line. It is thus also known as an interpreter. • JVM is also independent of the OS and hardware. It means that once a user writes a Java program, they can easily run it anywhere.