Differences between JDK, JRE and JVM

**JAVA DEVELOPMENT KIT**

The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) and other tools needed in Java development.

**JAVA RUNTIME ENVIRONMENT**

**JRE** stands for **“Java Runtime Environment”** and may also be written as **“Java RTE.”** The Java Runtime Environment provides the minimum requirements for executing a Java application; it consists of the *Java Virtual Machine (JVM), core classes*, and *supporting files*.

**JAVA VIRTUAL MACHINE**

It is:

• A specification where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Sun and other companies.

• An implementation is a computer program that meets the requirements of the JVM specification

• Runtime Instance Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.



• JDK – Java Development Kit (in short JDK) is Kit which provides the environment to develop and execute(run) the Java program. JDK is a kit(or package) which includes two things

1. Development Tools(to provide an environment to develop your java programs)

2. JRE (to execute your java program).

Note : JDK is only used by Java Developers.

• JRE – Java Runtime Environment (to say JRE) is an installation package which provides environment to only run(not develop) the java program(or application)onto your machine. JRE is only used by them who only wants to run the Java Programs i.e. end users of your system.

• JVM – Java Virtual machine(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 interpreter.

How does JRE and JDK works?

What does JRE consists of?

JRE consists of the following components:

• Deployment technologies, including deployment, Java Web Start and Java Plug-in.

• User interface toolkits, including Abstract Window Toolkit (AWT), Swing, Java 2D, Accessibility, Image I/O, Print Service, Sound, drag and drop (DnD) and input methods.

• Integration libraries, including Interface Definition Language (IDL), Java Database Connectivity (JDBC), Java Naming and Directory Interface (JNDI), Remote Method Invocation (RMI), Remote Method Invocation Over Internet Inter-Orb Protocol (RMI-IIOP) and scripting.

• Other base libraries, including international support, input/output (I/O), extension mechanism, Beans, Java Management Extensions (JMX), Java Native Interface (JNI), Math, Networking, Override Mechanism, Security, Serialization and Java for XML Processing (XML JAXP).

• Lang and util base libraries, including lang and util, management, versioning, zip, instrument, reflection, Collections, Concurrency Utilities, Java Archive (JAR), Logging, Preferences API, Ref Objects and Regular Expressions.

• Java Virtual Machine (JVM), including Java HotSpot Client and Server Virtual Machines.

How does JRE works?

To understand how the JRE works let us consider a Java source file saved as Example.java. The file is compiled into a set of Byte Code that is stored in a “.class” file. Here it will be “Example.class“.



The following diagram depicts what is done at compile time.

The following actions occur at runtime.

• Class Loader

The Class Loader loads all necessary classes needed for the execution of a program. It provides security by separating the namespaces of the local file system from that imported through the network. These files are loaded either from a hard disk, a network or from other sources.

• Byte Code Verifier

The JVM puts the code through the Byte Code Verifier that checks the format and checks for an illegal code. Illegal code, for example, is code that violates access rights on objects or violates the implementation of pointers.

The Byte Code verifier ensures that the code adheres to the JVM specification and does not violate system integrity.



• Intrepreter

At runtime the Byte Code is loaded, checked and run by the interpreter. The interpreter has the following two functions:

• Execute the Byte Code

• Make appropriate calls to the underlying hardware

Both operations can be shown as:



To understand the interactions between JDK and JRE consider the following diagram.  


How does JVM works?

JVM becomes an instance of JRE at runtime of a Java program. It is widely known as a runtime interpreter.JVM largely helps in the abstraction of inner implementation from the programmers who make use of libraries for their programmes from JDK.