1) What is Java?

[Java](https://www.javatpoint.com/java-tutorial) is the high-level, [object-oriented](https://www.javatpoint.com/java-oops-concepts), robust, secure programming language, platform-independent, high performance, Multithreaded, and portable programming language. It was developed by [**James Gosling**](https://www.javatpoint.com/james-gosling-father-of-java) in June 1991. It can also be known as the platform as it provides its own JRE and API.

2) What are the differences between C++ and Java?

The differences between [C++](https://www.javatpoint.com/cpp-tutorial) and Java are given in the following table.

|  |  |  |
| --- | --- | --- |
| **Comparison Index** | **C++** | **Java** |
| **Platform-independent** | C++ is platform-dependent. | Java is platform-independent. |
| **Mainly used for** | C++ is mainly used for system programming. | Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications. |
| **Multiple inheritance** | C++ supports multiple inheritance. | It can be achieved by [interfaces in java](https://www.javatpoint.com/interface-in-java). |
| **Operator Overloading** | C++ supports [operator overloading](https://www.javatpoint.com/cpp-overloading). | Java doesn't support operator overloading. |
| **Pointers** | C++ supports [pointers](https://www.javatpoint.com/cpp-pointers). You can write pointer program in C++. | Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in Java. |
| **Compiler and Interpreter** | C++ uses compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent. | Java uses compiler and interpreter both. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform independent. |
| **Call by Value and Call by reference** | C++ supports both call by value and call by reference. | Java supports call by value only. There is no call by reference in java. |
| **Thread Support** | C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support. | Java has built-in [thread](https://www.javatpoint.com/multithreading-in-java) support. |
| **Documentation comment** | C++ doesn't support documentation comment. | Java supports documentation comment (/\*\* ... \*/) to create documentation for java source code. |
| **Virtual Keyword** | C++ supports virtual keyword so that we can decide whether or not override a function. | Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default. |
| **Inheritance Tree** | C++ creates a new inheritance tree always. | Java uses a single inheritance tree always because all classes are the child of Object class in java. The object class is the root of the [inheritance](https://www.javatpoint.com/inheritance-in-java) tree in java. |
| **Hardware** | C++ is nearer to hardware. | Java is not so interactive with hardware. |
| **Object-oriented** | C++ is an object-oriented language. However, in C language, single root hierarchy is not possible. | Java is also an [object-oriented](https://www.javatpoint.com/java-oops-concepts) language. However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object. |

3) List the features of Java Programming language.

There are the following features in Java Programming Language.

* **Simple:** Java is easy to learn. The syntax of Java is based on C++ which makes easier to write the program in it.
* **Object-Oriented:** Java follows the object-oriented paradigm which allows us to maintain our code as the combination of different type of objects that incorporates both data and behavior.
* **Portable:** Java supports read-once-write-anywhere approach. We can execute the Java program on every machine. Java program (.java) is converted to bytecode (.class) which can be easily run on every machine.
* **Platform Independent:** Java is a platform independent programming language. It is different from other programming languages like C and C++ which needs a platform to be executed. Java comes with its platform on which its code is executed. Java doesn't depend upon the operating system to be executed.
* **Secured:** Java is secured because it doesn't use explicit pointers. Java also provides the concept of ByteCode and Exception handling which makes it more secured.
* **Robust:** Java is a strong programming language as it uses strong memory management. The concepts like Automatic garbage collection, Exception handling, etc. make it more robust.
* **Architecture Neutral:** Java is architectural neutral as it is not dependent on the architecture. In C, the size of data types may vary according to the architecture (32 bit or 64 bit) which doesn't exist in Java.
* **Interpreted:** Java uses the Just-in-time (JIT) interpreter along with the compiler for the program execution.
* **High Performance:** Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++).
* **Multithreaded:** We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.
* **Distributed:** Java is distributed because it facilitates users to create distributed applications in Java. RMI and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet.
* **Dynamic:** Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

4) What do you understand by Java virtual machine?

JVM is a virtual machine that enable the computer to run java program. JVM act as run time engine which call main method prasent in java program and it converted java code to bytecode which is machine independent and close to the native code.

### 5) What is the difference between JDK, JRE, and JVM?

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.

Now we need an environment to make a run of our program. Henceforth, 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.

Now let us discuss JVM, which stands out for java virtual machines. It is as follows:

* A specification where the 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 a java command on the command prompt to run the java class, an instance of JVM is created.

Before proceeding to the differences between JDK, JRE, and JVM, let us discuss them in brief first and interrelate them with the image below proposed.



Don’t get confused as we are going to discuss all of them one by one.

1. JDK (Java Development Kit) is a Kit that provides the environment to develop and execute(run) the Java program. JDK is a kit(or package) that includes two things

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

2. JRE (Java Runtime Environment) 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.

3. [JVM (Java Virtual Machine)](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/) 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](https://www.geeksforgeeks.org/compiler-vs-interpreter-2/).

Now let us discuss the components of JRE in order to understand its importance of it and perceive how it actually works. For this let us discuss components.

The components of JRE are as follows:

1. Deployment technologies, including deployment, Java Web Start, and Java Plug-in.
2. 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.
3. 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.
4. 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).
5. 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.
6. Java Virtual Machine (JVM), including Java HotSpot Client and Server Virtual Machines.

After having an adequate understanding of the components, now let us discuss the working of JDK. In order to understand how JDK works, let us consider an illustration below as follows:

Illustration:

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“.



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

Now let us discuss in brief how JVM works out. It is as follows:

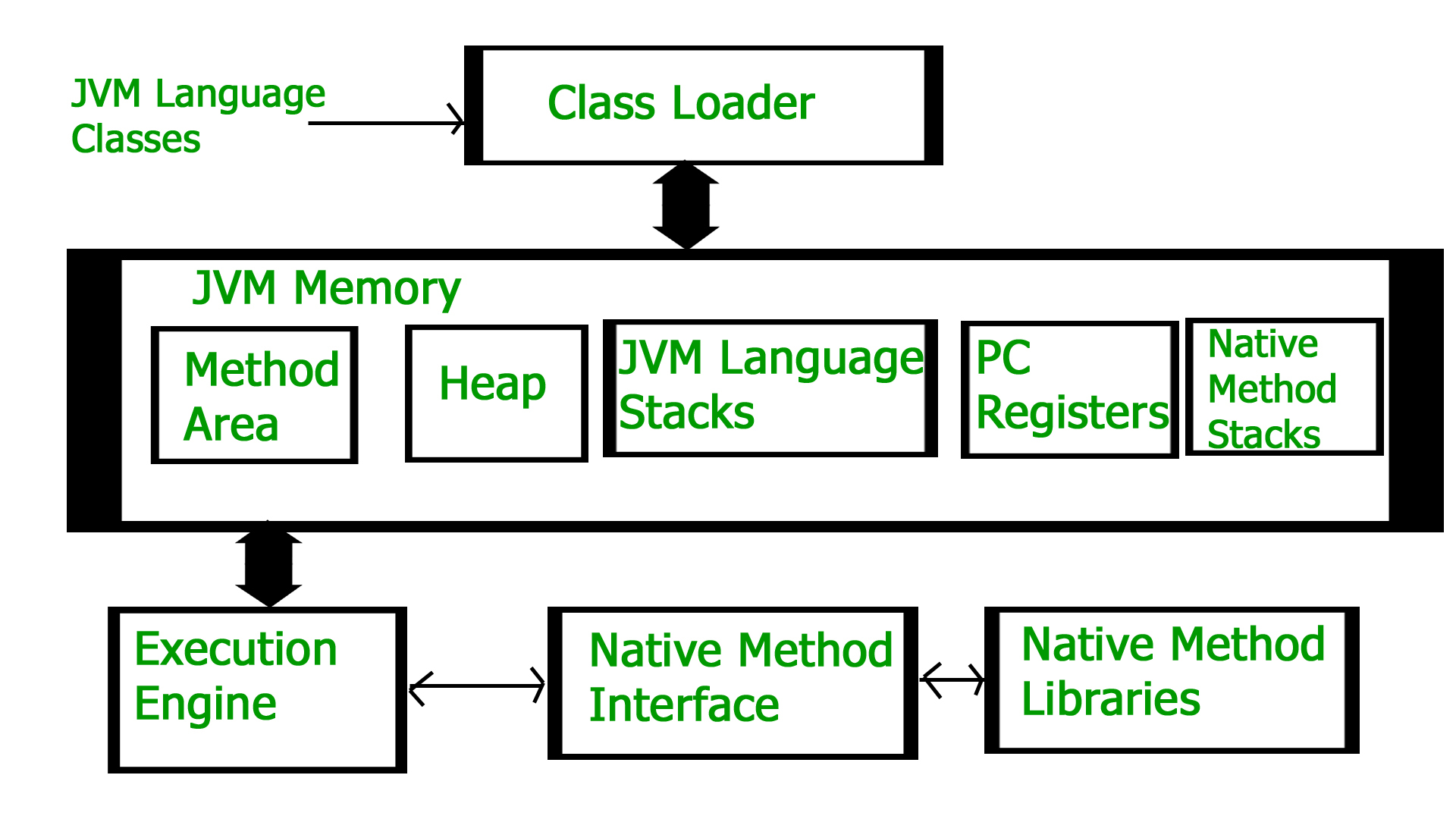
JVM becomes an instance of JRE at the runtime of a Java program. It is widely known as a runtime interpreter.JVM largely helps in the abstraction of inner implementation from the programmers who make use of libraries for their programs from JDK.

It is mainly responsible for three activities.

* Loading
* Linking
* Initialization

Similarly, now let us discuss the working of JRE which is as follows:

* JVM(Java Virtual Machine) acts as a run-time engine to run Java applications. JVM is the one that actually calls the main method present in a java code. JVM is a part of JRE(Java Runtime Environment).
* Java applications are called WORA (Write Once Run Anywhere). This means a programmer can develop Java code on one system and can expect it to run on any other Java-enabled system without any adjustments. This is all possible because of JVM.
* When we compile a .java file, .class files(contains byte-code) with the same class names present in .java file are generated by the Java compiler. This .class file goes into various steps when we run it. These steps together describe the whole JVM.



This article is contributed by Krishna Bhatia. If you like GeeksforGeeks and would like to contribute, you can also write an article using [write.geeksforgeeks.org](http://www.write.geeksforgeeks.org/) or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.