− Introduction to Java and Features

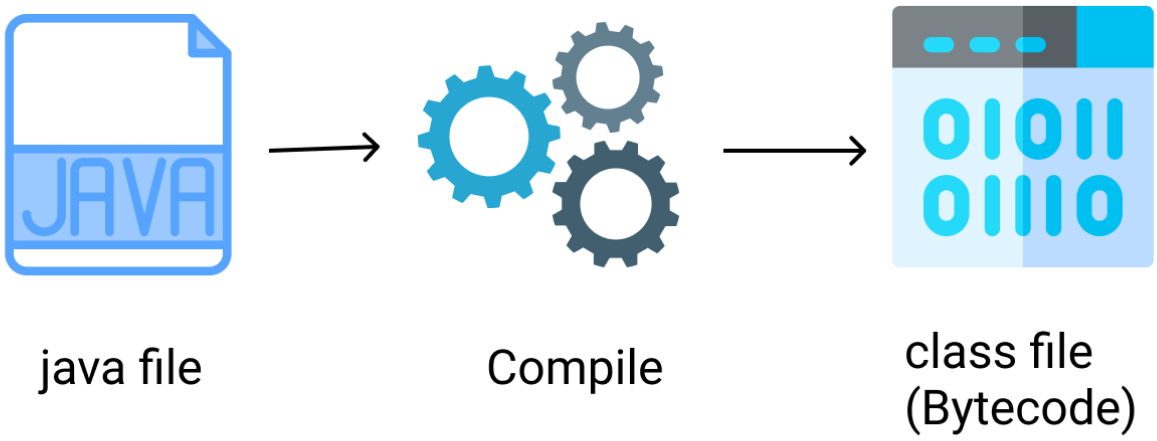
Getting started with any programming language should be purposeful, we must understand the **features and power** of the programming language to develop a deep interest in it.
Java was developed after the C and C++ around 1991. We can consider Java as a **complete package** for software developers to play with it for a variety of applications they want to build.

Java is currently the most used programming language worldwide. The biggest tech giants such as **Amazon, Google, TCS, Wipro, etc.,** have a huge percentage of code in Java.

Java is power-packed with ample of features. Here are some of the most important features of Java Programming Language:

- **Simple -** Java was designed for professional developers to learn and use easily . If you are aware of basic programming principles, then Java is not a hard nut to crack.
- **Secure -** Java achieves security by confining Java Programs to Java Execution Environment and not allowing Java programs to access other parts of the computer system.
- **Portable -** Portable code must run on a variety of operating systems, CPU's, and browsers. When a Java program is compiled, a Bytecode is generated, this bytecode can be executed on any platform using JVM (Java Virtual Machine). We will discuss JVM details in the latter part of the tutorial.
- **Object-Oriented -** Java provides a clean, usable, and pragmatic approach to objects. It was designed on the principle that "Everything under the sun is an Object."
- **Robust -**The robustness of the program is the reliable execution on a variety of systems. To better understand this, consider one of the main reasons for program failure is memory management. In C/C++, a programmer must manually allocate and free dynamic memory. There are cases where programmers forgot to free up the dynamic memory which may cause program failure. Java virtually solves this problem as unused objects are garbage collected automatically.
- **Architecture Neutral -** The central issue for Java Designers was that of code longevity and portability. One of the main problems for programmers was that there was no guarantee that the program running perfectly fine today will run tomorrow because of the system upgrades, processor upgrades, or changes in core system resources. Java Designers made several hard decisions and designed the Java Virtual Machine to solve this problem. Their goal was "write once, run anywhere, any time, forever".
- **Distributed -** Java is designed for distributed environment because it handles TCP/IP protocol. It is very much useful in building large scale distributed computing based applications. Java also supports Remote Method Invocation. This feature enables a program to invoke methods across a network.
- **Dynamic -** Java programs have run time meta information to verify and resolve accesses of the objects at runtime. This makes it possible to dynamically link code in a safe and secure environment. It also gives us the feature to update small fragments of bytecode, dynamically updated on a running system.

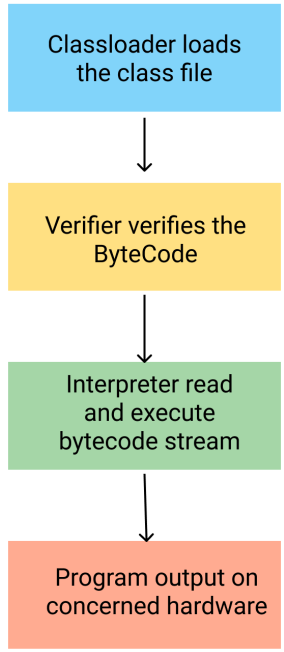
Java Program Compilation - Java program compilation converts a java program file to a highly optimized class file which is called bytecode.



Java Program Execution - Java program execution is completed with the help of the following tools.

- **Classloader -** It loads the class file for JVM execution.
- **Bytecode Verifier -** It verifies the bytecode and restrict the objects for illegal access of other parts of the system.
- **Interpreter -** It read the bytecode instructions and execute them line by line.

Java program Execution Flow:



Java Virtual Machine (JVM)

JVM is called an abstract virtual machine because it does not exist physically, but it is a kind of specification that provides a secure runtime environment to execute the bytecode generated through the compiler, JVM actually invokes the main( ) method present in the Java program.

JVM is a part of the 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 adjustment. This is all possible because of JVM.

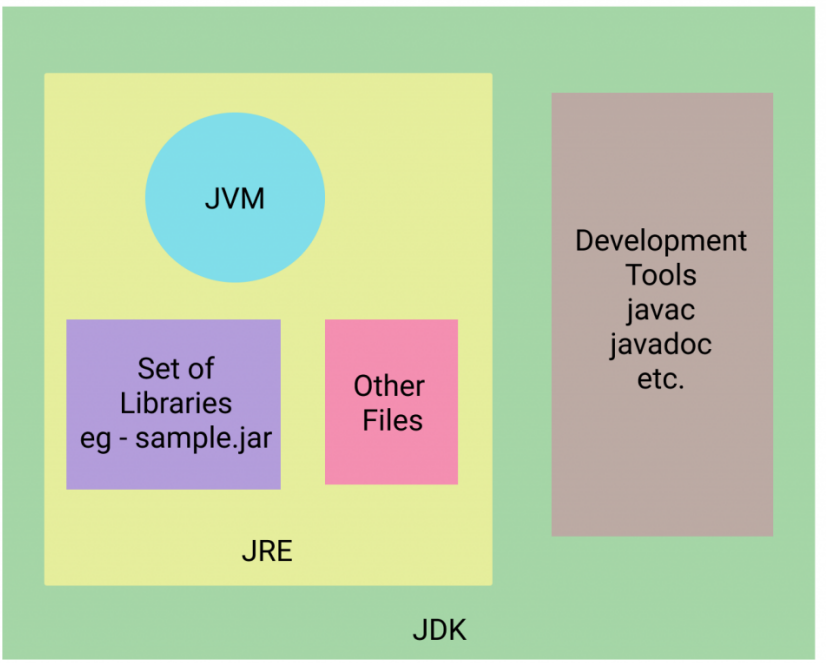
Java Runtime Environment (JRE)

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.

- 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 a java command on the command prompt to run the java class, an instance of JVM is created.

Java Development Kit (JDK)

Java Development Kit (in short JDK) is a Kit that provides the environment to develop and execute. The JDK contains a private Java Virtual Machine (JVM), interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.



Difference between Java and C++

Comparison Metric	C++	Java
Domain and Usage	C++ is mainly used for system programming.	Java is mainly used for building desktop, web, and Mobile Applications.
Compiler and Interpreter	In C++ program is only compiled and converted into machine code. That is why it is platform dependent.	In Java , program once compiled converted into bytecode ,then interpreter read the bytecode stream to execute line by line through JVM. That is why it is platform independent.
Pointers	C++ supports pointers for memory allocation , You can write pointer programs.	Java supports pointer internally , however you can not write pointer programs.
Structure and Union	C++ have Structures and Unions.	Java don’t have these , however you can implement this using classes.
Thread Support	C++ don’t have inbuilt thread support , you have to use third party library.	Java has built in thread support.
Documentation Comment	C++ don’t have documentation comment.	Java have documentation comment , which is useful for maintaining the code.
Call by value and reference	C++ supports both call by value and call by reference.	Java supports only call by value.
Operator Overloading	C++ supports operator overloading.	Java don’t have support for operator overloading.