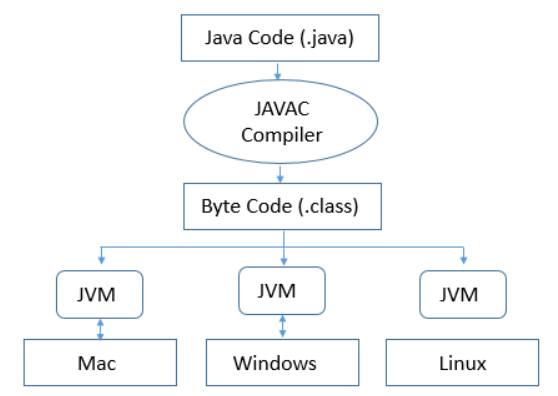
**ABOUT JAVA VIRTUAL MACHINE**

1. The JVM is a virtual machine that runs Java class files in a portable way. Being a virtual machine means that JVM is an abstraction of an underlying, actual machine--such as the server that your program is running on.
2. A Java virtual machine (JVM) is a virtual machine that **enables a computer to run Java programs as well as programs written in other languages that are also compiled to** Java bytecode.
3. When you run a Java program, it runs as a thread within the JVM’s responsibility to load your class files, verify code, interpret them and execute them.



1. **Technical definition :-** The JVM is the specification for a software program that execute code and provides the runtime environment for that code.
2. **Everyday definition :-** The JVM is how we run our Java program. We configure the JVM’s settings and then rely on it to manage program resource during execution.
3. In Java, program memory is managed by JVM. The JVM manage memory through a process called **garbage collection**, which continuously identified and eliminates unused memory in Java programs.
4. The Java class loader is the part of the JVM that loads classes into memory and makes them available for execution. Class loader use techniques like lazy-loading and caching to make class loading as efficient as it can be. That said, class loading isn’t the epic brain-teaser that portable runtime memory management is, so techniques are comparatively simple.
5. **The Java class loader in the JVM :-** Everything in Java is a class, and all Java applications are built from classes. An application could consist of one class or thousands. In order to run a Java application, a JVM must load compiled (dot) .class file into a context, such as a server, where they can be accessed. A JVM depends on its class loader to perform this function.

Q1) Why JAVA is platform independent ?

Q2) JIT Compiler

Q3)

import java.util.Scanner;

100 % ABSTRACTION

interface clonable,appendable

{

void TERMINATOR()

}

class INNOVATE implements INSAN

{

void TERMINATOR()

{

// 1 eye

// dekh data collect

// fly

}

}

filterStream -> PrintStream