**Java:**

Java is high level programming language. A program written in high level language cannot be run on any machine directly. First it needs to be translated into that particular machine language. The javac compiler does this thing[ machine code is referred as bytecode]

**JVM:**  Java Virtual machine reside in the real machine and machine language for JVM IS bytecode. JVM executes bytecode.



**Java Virtual Machine( JVM ) :**

We write a program, then compile the program and at last we run the program.

1. We write program.
2. Complication of program is done by javac complier. Complier takes program has input and generates bytecode output.
3. JVM executes the bytecode generated by compiler.

Each operating system has different JVM, however the output they produce after execution of bytecode is same across all the operating system.

**JDK** contains Java Runtime Environment, compilers and various tools like JavaDoc, Java debugger etc.

**JRE** is **part** of **JDK**. When JRE is installed on system we can a java program however we cannot compiler it. **JRE includes JVM**.

* Java is platform independent language.
* Java is an Object Oriented language.
* Robust language – means reliable.
* Secure – means several security flaws like stack corruption, buffer overflow is impossible
* Distributed – means we can create distributed applications (objects one JVM can execute procedures on a remote JVM.
* Multithreading – means java allows concurrent execution of two or more parts of program.
* Portable – means java code written on one machine can run on another machine.

**JVM Architecture**

* 

**Class Loader:** the class loader read the .class file and save the bytecode in the **method area.**

**Method Area:** this holds the class level information [ JVM has only one method area which is shared among all the classes].

**Heap:** is a part of JVM memory where objects are allocated. JVM creates a class object for each .class file.

**Stack:** is also a part of JVM memory but unlike Heap, it is used for storing temporary variables.

PC Register: this keeps the track of which instructions has been executed and which one is going to be executed. Since instructions are executed by threads, each thread has a separate PC register.

Native Method Stack: native method can access the runtime data areas of the virtual machine.

Native Method Interface: it enables java code to call or be called by native applications. Native applications are programs that are specific to the hardware and OS of a systm.