SNS COLLEGE OF TECHNOLOGY
(AN AUTONOMOUS INSTITUTION)
COIMBATORE – 35
DEPARTMENT OF COMPUTER SIENCE AND ENGINEERING

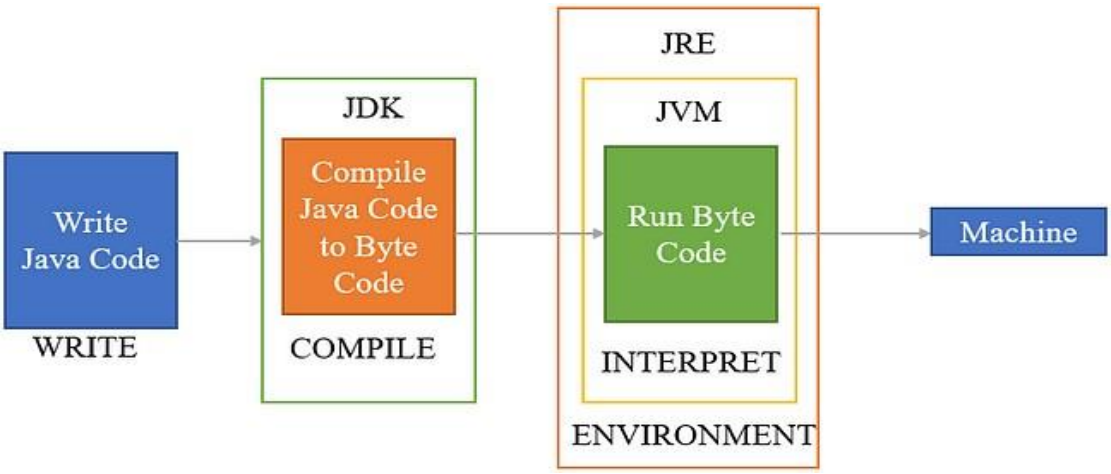
UNIT I- INTRODUCTION TO OOP

Object Oriented Programming concepts
Evolution of java
Java Architecture
Data Types
Variables and Operators
Environment setup
Command Line Arguments, Comments

Java Architecture

There are two processes in Java — Compilation and Interpretation.

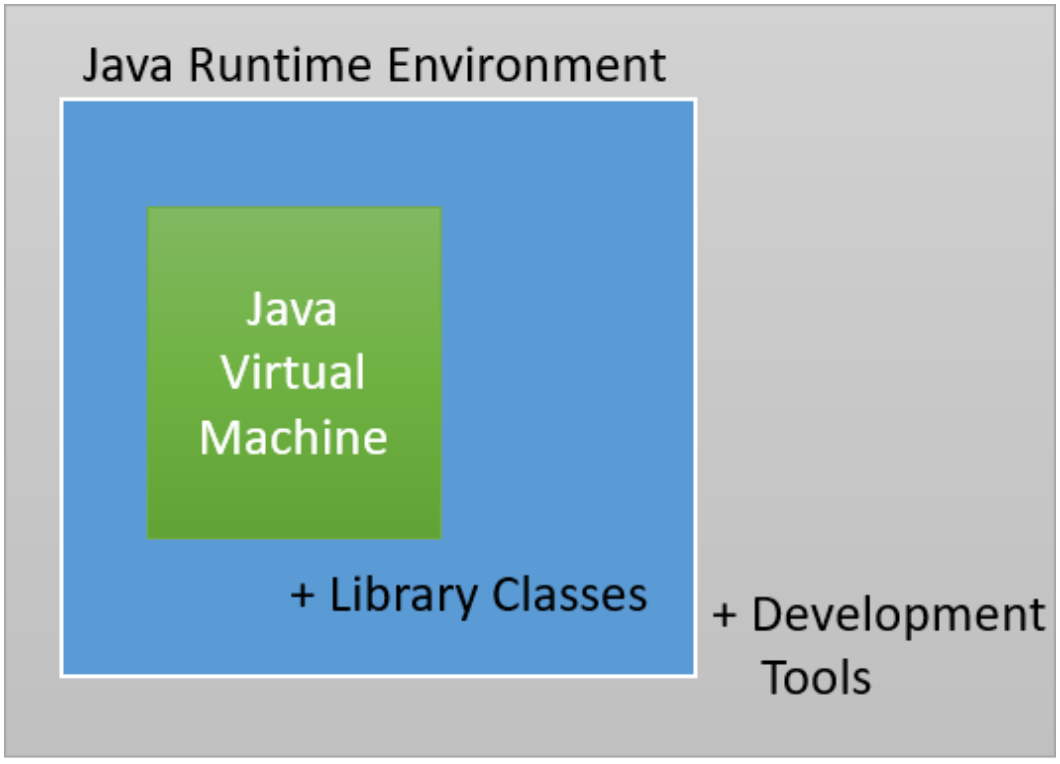
- The Java source code goes to the compiler.
- The Java Compiler converts it into byte codes
- The bytes codes are then converted into machine by the JVM
- The Machine code is executed directly by the machine(Operating System)



Components of Java Architecture

The different components of Java architecture are

- JRE — Java Runtime Environment
- JDK — Java Development Kit
- JVM — Java Virtual Machine



JDK = JRE + Development Tools
JRE = JVM + Library Classes

Java Runtime Environment(JRE)

Java Runtime Environment provides an platform where all the applications like JVM and other runtime libraries linked together to run your Java Program. It builds a runtime environment where you can execute the Java program. The JRE also initiates the JVM for its execution. JRE has the required software and libraries to run the programs.

Java Development Kit (JDK)

Java Development Kit is the set of libraries, compiler, interpreter and other set of programs that will help you to build the Java program. Once JDK installed on machine then start developing, compile and run the Java program. You cannot compile Java program without JDK installed on machine. Once you compile the code with JDK tools, you can get an Byte Code File.

- **java** : it is the launcher for all the java applications.
- **javac** : compiler of the java programming languages.
- **javadoc**: it is the API documentation generator.
- **jar**: creates and manage all the JAR files.

Java Virtual Machine(JVM)

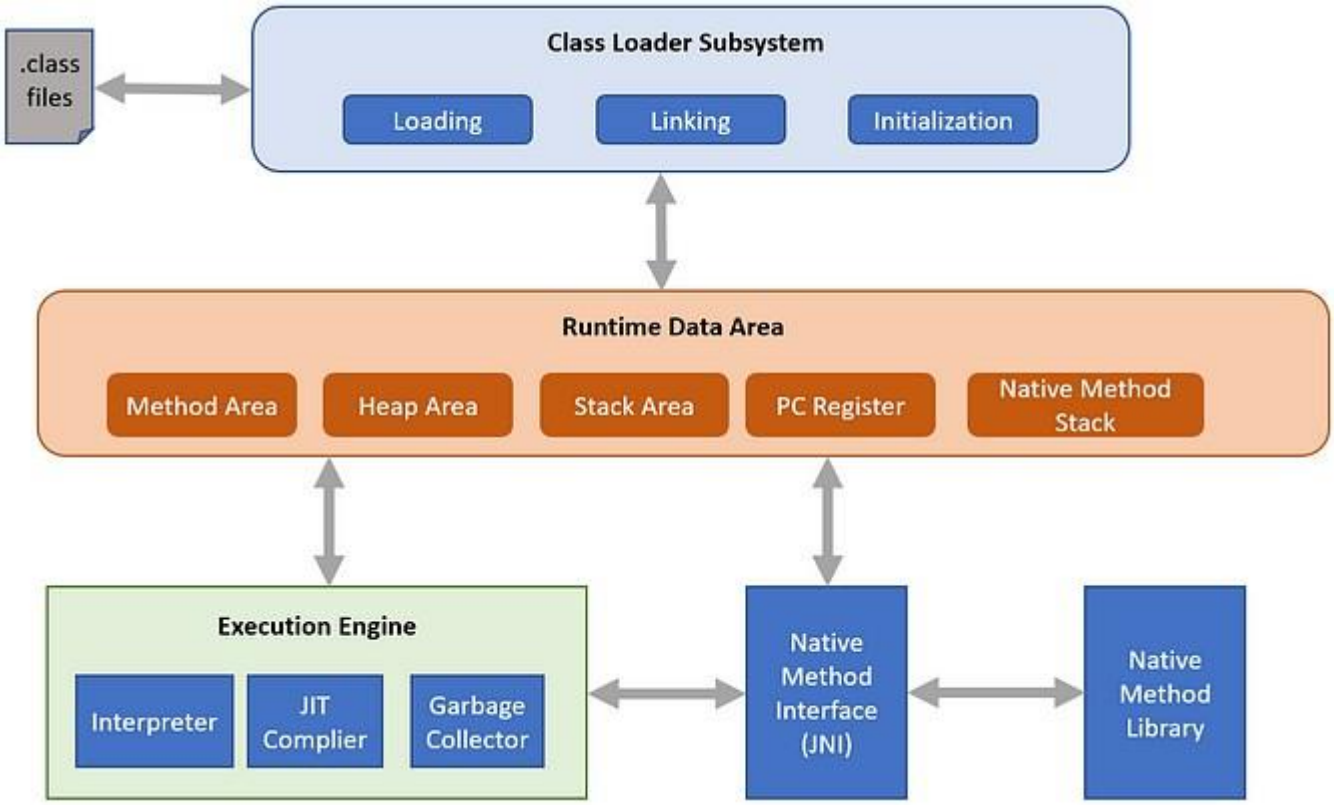
JVM is the interpreter that executes the byte codes into Machine code(instructions). The beautiful quality is WORA(Write Once Run Anywhere). This means code can runs its applications on any platform. This makes Java as Platform Independent.

In a nutshell, JVM performs the following functions:

- Loads the code

- Verifies the code
- Executes the code
- Provides runtime environment

The below architecture depicts the architecture of the JVM. Let's see the each element in detail:



Components of JVM

Class Loader Subsystem

Class Loader is a subsystem of the Java Virtual Machine loads class files. It is the first component of the architecture as it loads the program so other tasks can take place. It also links and initialize the class files. It has three components — Loading, Linking and Initialization

Loading — This component loads the class. It has

- Bootstrap Classloader — Loads the classes belonging to the bootstrap classpath. It loads core java API classes present in the "*JAVA_HOME/jre/lib"* directory
- Extension Classloader — Loads classes situated inside the extension folders"*JAVA_HOME/jre/lib/ext"*(Extension path)
- Application Classloader — Loads classes from path mentioned Environment Variable (mapped to java.class.path) or similar files It is also implemented in Java by the *sun.misc.Launcher$AppClassLoader* class.

Linking — The subsystem has a verifier to verify if the byte code is correct or not. It generates the verification error if the byte code isn't proper. The linking allocates all static variables memory and assigns the default values and replaces the symbolic references of memory with original ones.

Initialization — The system assigns the static variables to the original ones and executes the static block.

Runtime Data Areas

Class Method Area -

- It stores all the class -level data.
- Static Variables, Static Blocks, Static Methods, Instance Methods are stored in this area.
- Every JVM has only one method area.

Heap Area -

- A heap is created when the JVM starts up.
- It may increase or decrease in size while the application runs.
- It stores all the objects and their instance arrays and variables.
- Only one heap area per JVM.

Stack Area -

- JVM stack is known as a thread stack.
- It is a data area in the JVM memory which is created for a single execution thread.
- The JVM stack of a thread is used by the thread to store various elements i.e.; local variables, partial results, and data for calling method and returns.
- It creates unique runtime stacks for every thread and makes an entry for every method call in the stack memory(knows as stock frame).
- It is a Local Variable Array which is related to the method, operand stack and the frame data, where all symbols related to the method remain stored.
- The frame data maintains the catch block information unless there's an exception.

PC Registers -

- Every thread has separate PC Registers which hold the address of the running instructions.
- Once an instruction has completed execution, the PC register updates itself with the next one.

Native Method Stacks -

- It subsumes all the native methods used in your application.
- It creates a unique native method stack for every thread.

Note that method area and heap area are shared resources while the stack area is not.

Execution Engine

The Execution Engine executes the bytecode. It reads and executes and has different components:

Interpreter -

- Interprets the bytecode quickly but is a little slow in execution
- It has a significant drawback as when the system calls one method multiple times, and it requires a new interpretation every time.
- This drawback of the interpreter damages the efficiency of the process substantially.

JIT Complier -

- The Just-In-Time Compiler is part of the runtime environment
- It helps in improving the performance of Java applications by compiling byte codes to machine code at runtime
- The JIT Compiler has the intermediate code generator for producing intermediate code and the code optimizer for optimizing the same.
- It also has a target code generator that produces the native doe and a profile that finds hotspots.

- It is enabled by default. The JIT compiler compiles the bytecode of that method into machine code, compiling it “just in time” to run
- The JIT Compiler doesn’t have the drawback the interpreter has.When the Execution Engine finds repeated code, it uses the JIT Compiler instead of the interpreter.

Garbage Collector -

- Gathers and gets rid of unreferenced objects.
- It tracks each and every object available in the JVM heap space and removes unwanted ones.
- Garbage collector works in two simple steps known as Mark and Sweep:

Mark — it is where the garbage collector identifies which piece of memory is in use and which are not

Sweep — it removes objects identified during the “mark” phase.

Apart from these components the JVM also has the JNI (Java Native Interface) and the Native Method Libraries.

Java Native Interface (JNI) :

It is an interface that interacts with the Native Method Libraries and provides the native libraries(C, C++) required for the execution. It enables JVM to call C/C++ libraries and to be called by C/C++ libraries which may be specific to hardware.

Native Method Libraries :

It is a collection of the Native Libraries(C, C++) which are required by the Execution Engine.

Video link:

JVM (Java Virtual Machine)
https://www.youtube.com/watch?v=5Bp6GLU6HKE

JRE and JDK in Java
https://www.youtube.com/watch?v=KYogNWbjZIU