Java Virtual Machine (JVM) architecture

If you start learning Java, you would want to learn full details of how JVM really functioning. In this section, I'm going to explain JVM (Java Virtual Machine) architecture.

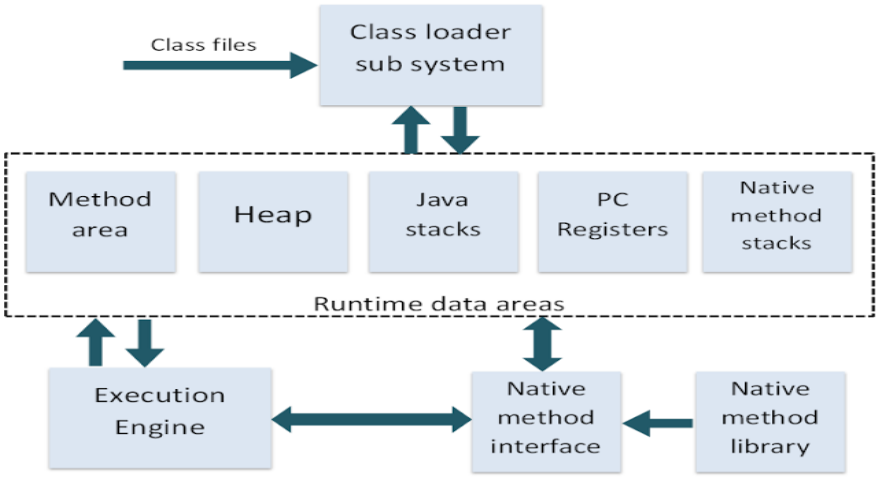

Figure: JVM Architecture

JVM has various sub components internally. You can see all of them from the above diagram.

1. Class loader sub system: JVM's class loader sub system performs 3 tasks

a. It loads .class file into memory.

b. It verifies byte code instructions.

c. It allots memory required for the program.

2. Run time data area: This is the memory resource used by JVM and it is divided into 5 parts

a. Method area: Method area stores class code and method code.

b. Heap: Objects are created on heap.

c. Java stacks: Java stacks are the places where the Java methods are executed. A Java stack contains frames. On each frame, a separate method is executed.

d. Program counter registers: The program counter registers store memory address of the instruction to be executed by the microprocessor.

e. Native method stacks: The native method stacks are places where native methods (for example, C language programs) are executed. Native method is a function, which is written in another language other than Java.

3. Native method interface: Native method interface is a program that connects native methods libraries (C header files) with JVM for executing native methods.

4. Native method library: holds the native libraries information.

5. Execution engine: Execution engine contains interpreter and JIT compiler, which covert byte code into machine code. JVM uses optimization technique to decide which part to be interpreted and which part to be used with JIT compiler. The HotSpot represents the block of code executed by JIT compiler.