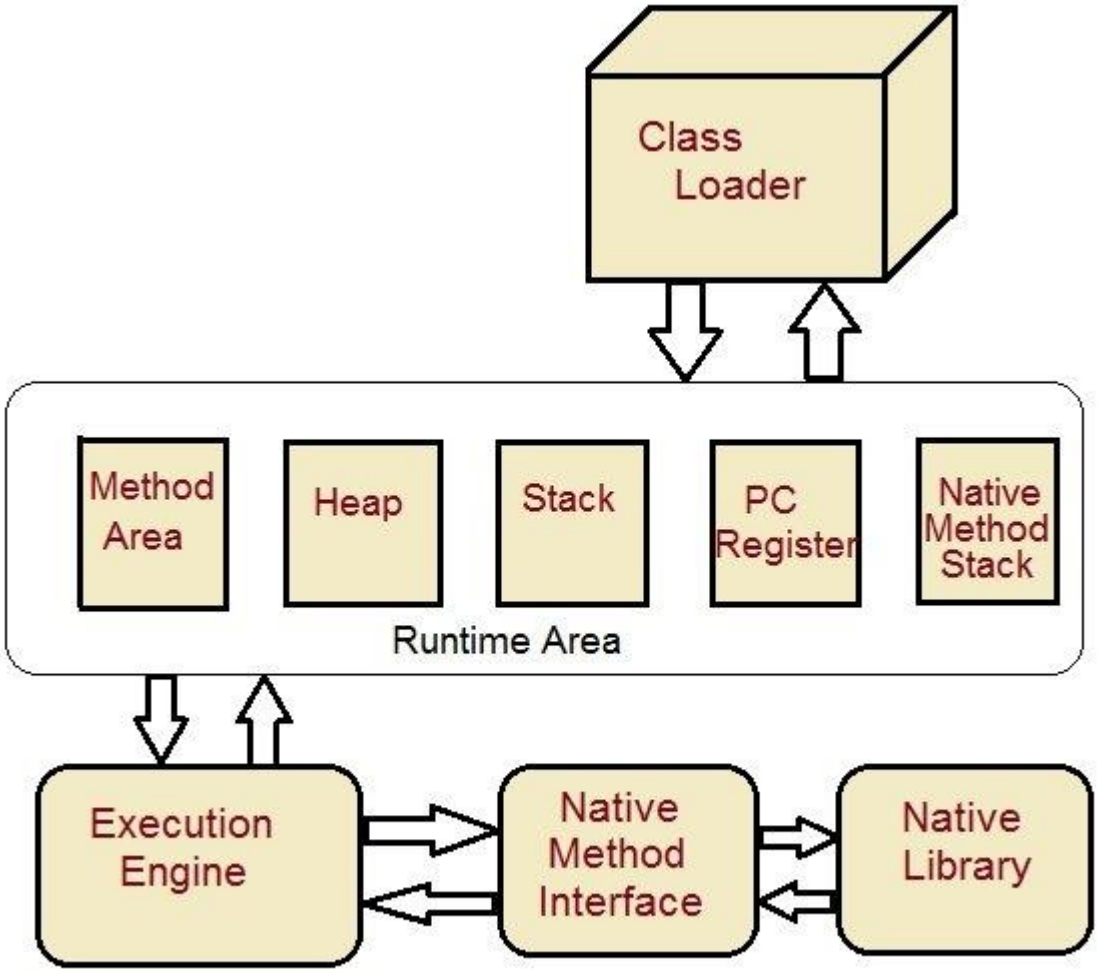
JVM, JDK and JRE

 # Java virtual Machine (JVM) is a virtual Machine that provides runtime environment to execute java byte code. The JVM doesn't understand Java typo, that's why you compile your *.java files to obtain *.class files that contain the bytecodes understandable by the JVM.

JVM controls the execution of every Java program. It enables features such as automated exception handling, Garbage-collected heap.

 # JVM Architecture

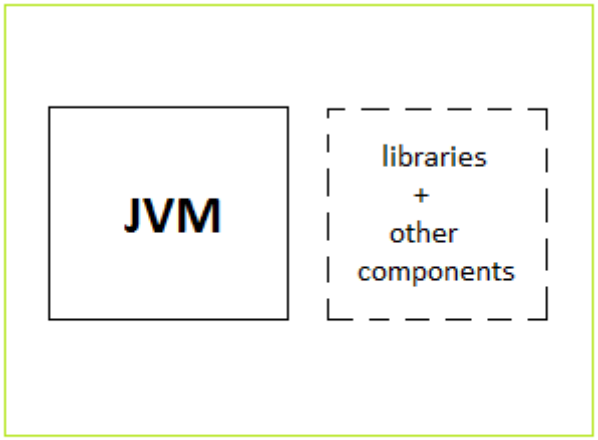
JVM architecture in Java



- Class Loader : Class loader loads the Class for execution.
- Method area : Stores pre-class structure as constant pool.
- Heap : Heap is a memory area in which objects are allocated.
- Stack : Local variables and partial results are stored here. Each thread has a private JVM stack created when the thread is created.
- Program register : Program register holds the address of JVM instruction currently being executed.
- Native method stack : It contains all native used in application.
- Executive Engine : Execution engine controls the execute of instructions contained in the methods of the classes.
- Native Method Interface : Native method interface gives an interface between java code and native code during execution.
- Native Method Libraries : Native Libraries consist of files required for the execution of native code.

 # Difference between JDK and JRE

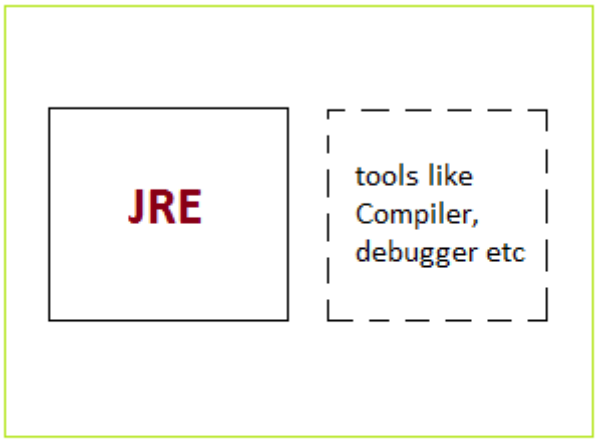
- JRE : The Java Runtime Environment (JRE) provides the libraries, the Java Virtual Machine, and other components to run applets and applications written in the Java programming language. JRE does not contain tools and utilities such as compilers or debuggers for developing applets and applications.



JRE - Java Runtime Environment

EQUALITY EDUCATION

- JDK : The JDK also called Java Development Kit is a superset of the JRE, and contains everything that is in the JRE, plus tools such as the compilers and debuggers necessary for developing applets and applications.



JDK - Java Development Kit

EQUALITY EDU ION