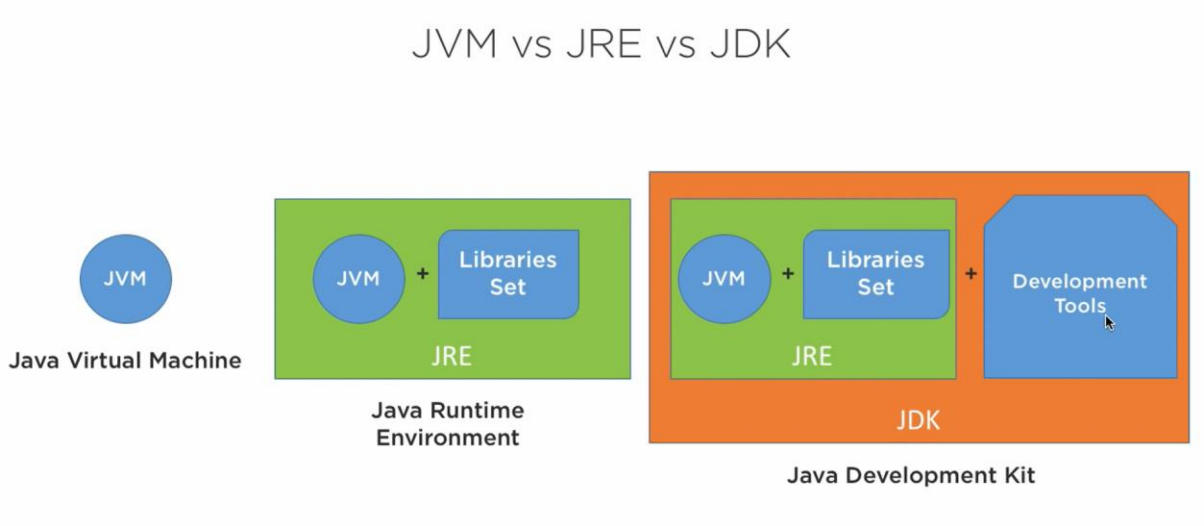
Day 1 task:

JDK,JRE,JVM



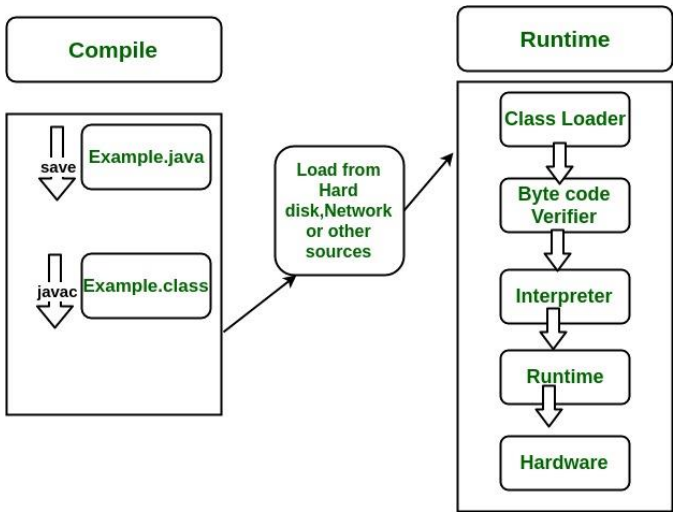
JDK – Java Development kit

JDK is Kit which provides the environment to **develop and execute(run)** the Java program. JDK is a kit(or package) which includes two things

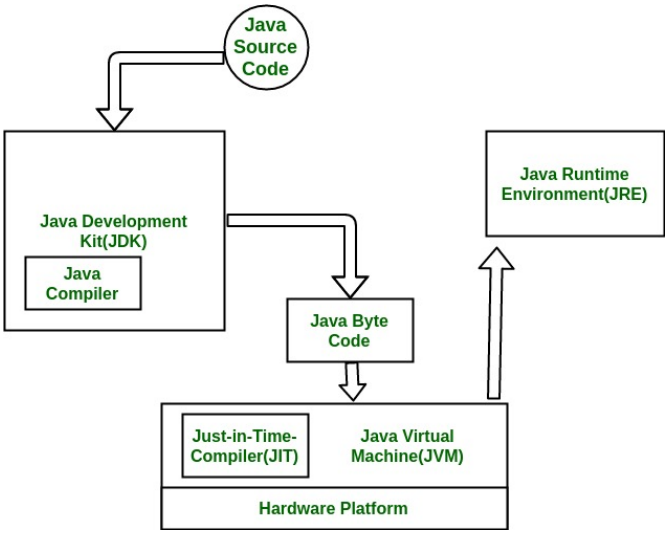
1. Development Tools(to provide an environment to develop your java programs)
2. JRE (to execute your java program).

JRE – Java Runtime Environment

JRE is an installation package which provides environment to **only run the java program**(or application)onto your machine.



How program runs:

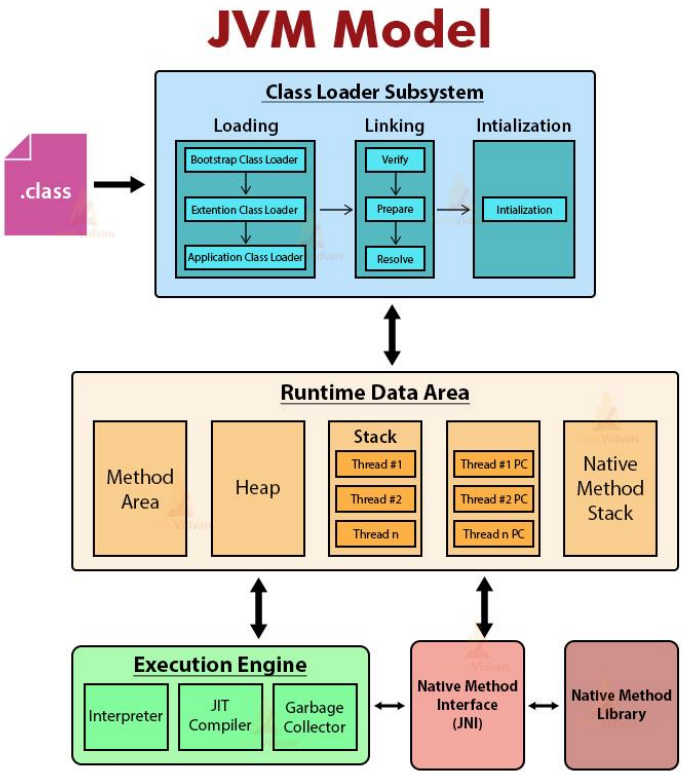


JVM - java virtual machine

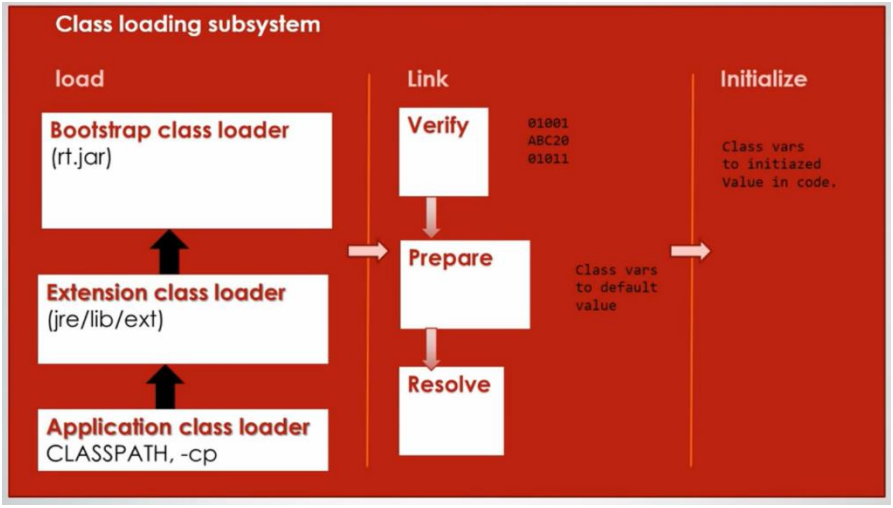
- JVM is a part of JRE(Java Runtime Environment).
- JVM is a platform dependent.
- Java applications are called WORA (Write Once Run Anywhere).

This means a programmer can develop Java code on one system and can expect it to run on any other Java-enabled system without any adjustment.

- When we compile a .java file, .class files(contains bytecode) with the same class names present in .java file are generated by the Java compiler.

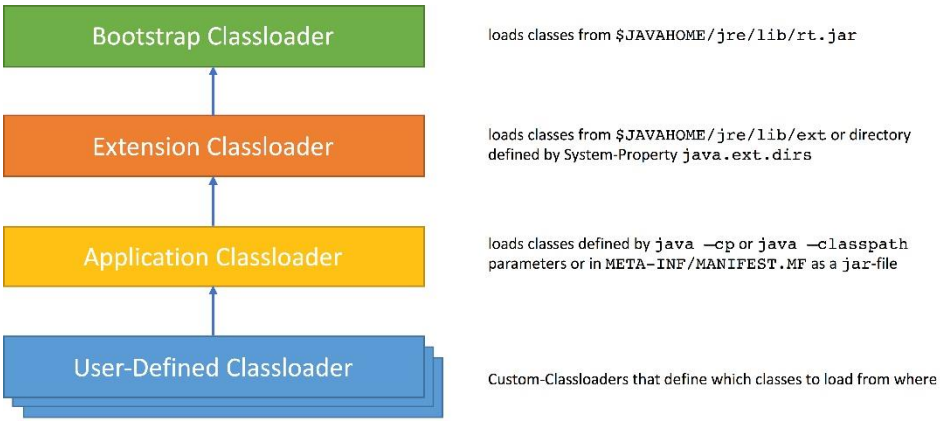


Class Loader Subsystem:

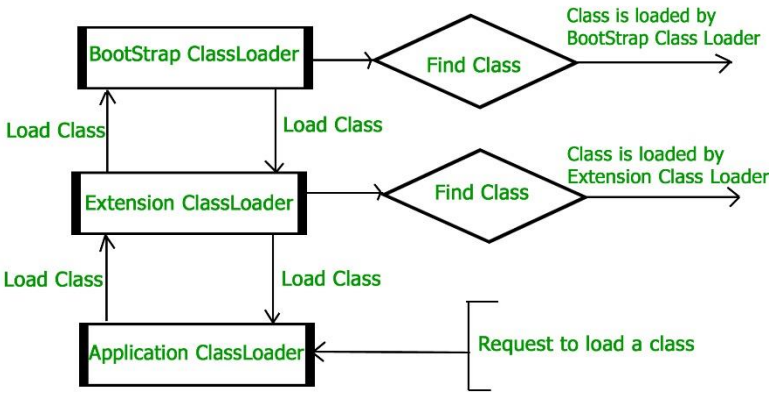


Loading:

The Class loader reads the "*.class"* file, generate the corresponding binary data and save it in the method area



JVM follows the Delegation-Hierarchy principle to load classes.



- System class loader delegate load **request to extension class loader** and
- extension class loader delegate **request to the bootstrap class loader**.
- If a class found in the boot-strap path, the class is loaded otherwise request again transfers to the extension class loader and then to the system class loader.
- At last, if the system **class loader fails to load class**, then we get run-time exception *java.lang.ClassNotFoundException*.

Linking:

- Performs verification,
- preparation,
- (optionally) resolution.

Verification: it checks whether this file is properly formatted and generated by a valid compiler or not. If verification fails, we get **run-time exception java.lang.VerifyError**.

Preparation: JVM allocates memory for class variables and initializing the memory to default values.

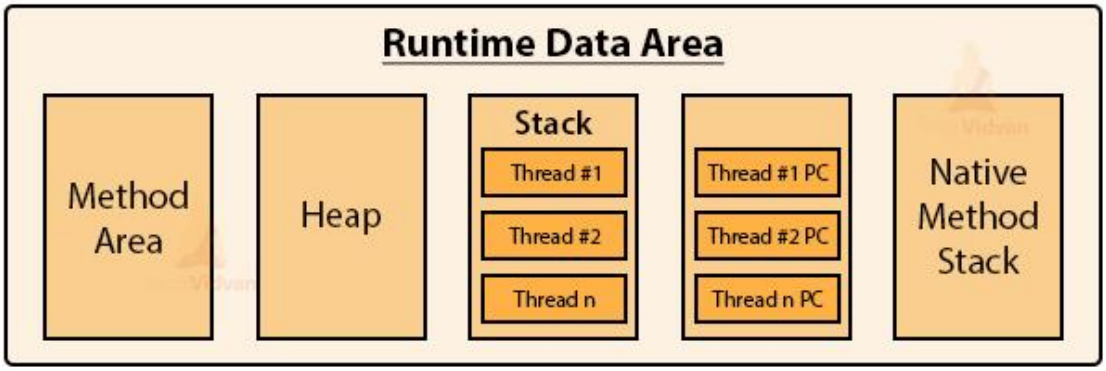
Resolution: It is the process of replacing symbolic references from the type with direct references.

Initialization:

In this phase, all static variables are assigned with their values defined in the code and static block(if any).

This is executed from top to bottom in a class and from parent to child in the class hierarchy.

JVM Memory

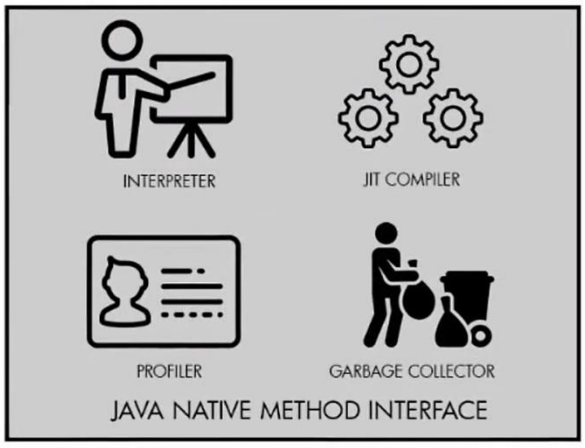


1. **Method area:** all class level information like class name, immediate parent class name, methods and variables information etc. are stored, including static variables.
2. **Heap area:** Information of all objects is stored in the heap area.
3. **Stack area:** For every thread, JVM creates one run-time stack which is stored here. Every block of this stack is called activation record/stack frame which stores methods calls.

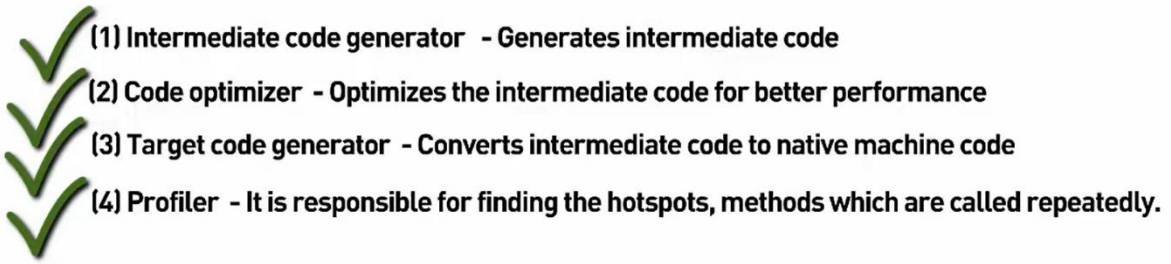
After a thread terminates, its run-time stack will be destroyed by JVM.

4. **PC Registers:** Store address of current execution instruction of a thread.
5. **Native method stacks:** For every thread, a separate native stack is created. It stores native method information.

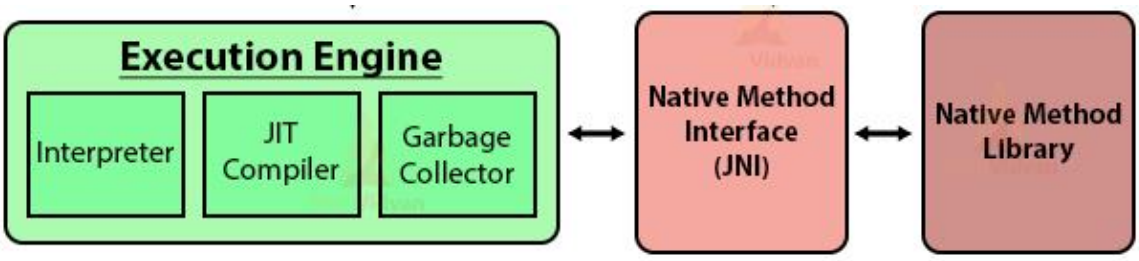
Execution Engine



- *Interpreter*: It interprets the bytecode line by line and then executes. The disadvantage here is that when one method is called multiple times, every time interpretation is required.
- *Just-In-Time Compiler(JIT)* : It is used to increase the efficiency of an interpreter. It compiles the entire bytecode and changes it to native code so whenever the interpreter sees repeated method calls, JIT provides direct native code for that part so re-interpretation is not required, thus efficiency is improved.



- *Garbage Collector*: It destroys un-referenced objects. For more on Garbage Collector, refer Garbage Collector



Java Native Interface (JNI) :

It is an interface that interacts with the Native Method Libraries and provides the native libraries required for the execution.

Native Method Libraries :

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