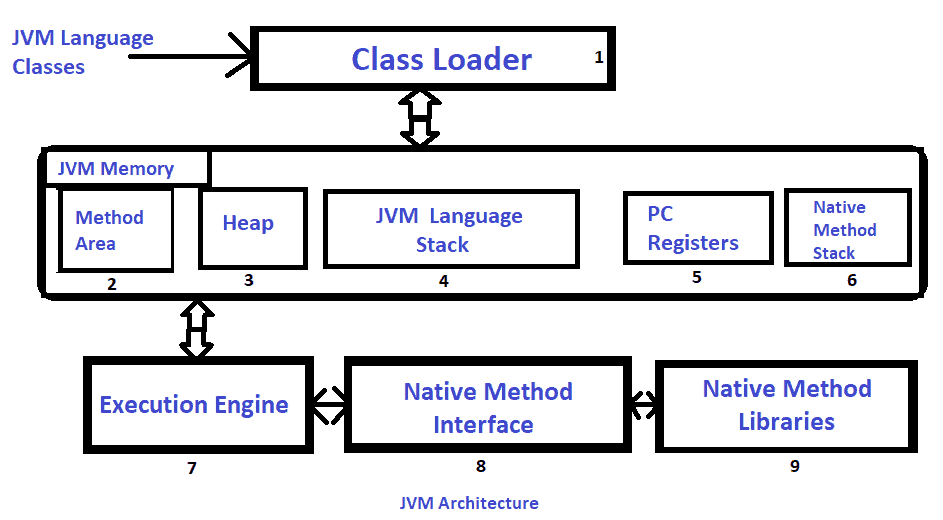
JVM Architecture:



Java applications are called WORA (Write Once Run Anywhere).

Class loader mainly works 3 responsibility

Loading Linking & Initialization

Loading: The Class loader reads the “.class” file, generate the corresponding binary data and save it in the method area.

After loading the “.class” file, JVM creates an object of type Class to represent this file in the heap memory.

Linking: Performs verification, preparation, and (optionally) resolution.

Verify the correctness

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

It is done by searching into the method area to locate the referenced entity.

**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.

Java ClassLoader is an abstract class. It belongs to a **java.lang** package. It loads classes from different resources. Java ClassLoader is used to load the classes at run time.

There are 3 class loader

Bootstrap, extension and System/Application class loader

Bootstrap class loader: It loads standard JDK class files from rt.jar and other core classes. It is a parent of all class loaders. It doesn't have any parent. When we call String.class.getClassLoader() it returns null, and any code based on it throws NullPointerException. It is also called Primordial ClassLoader. It loads class files from jre/lib/rt.jar. For example, java.lang package class.

**Extensions Class Loader:** It delegates class loading request to its parent. If the loading of a class is unsuccessful, it loads classes from jre/lib/ext directory or any other directory as java.ext.dirs. It is implemented by sun.misc.Launcher$ExtClassLoader in JVM.

**System Class Loader:** It loads application specific classes from the CLASSPATH environment variable. It can be set while invoking program using -cp or classpath command line options. It is a child of Extension ClassLoader. It is implemented by sun.misc.Launcher$AppClassLoader class. All Java ClassLoader implements java.lang.ClassLoader.

**Method Area** is a part of the heap memory which is shared among all the threads. It creates when the JVM starts up. It is used to store class structure, superclass name, interface name, and constructors.

**Stack Area** generates when a thread creates. It can be of either fixed or dynamic size. The stack memory is allocated per thread. It is used to store data and partial results. It contains references to heap objects. It also holds the value itself rather than a reference to an object from the heap. The variables which are stored in the stack have certain visibility, called scope.

**Native Method stack** It is also known as C stack. It is a stack for native code written in a language other than Java. Java Native Interface (JNI) calls the native stack. The performance of the native stack depends on the OS.

There are five types of **garbage collection** are as follows:

* **Serial GC:** It uses the mark and sweeps approach for young and old generations, which is minor and major GC.
* **Parallel GC:** It is similar to serial GC except that, it spawns N (the number of CPU cores in the system) threads for young generation garbage collection.
* **Parallel Old GC:** It is similar to parallel GC, except that it uses multiple threads for both generations.
* **Concurrent Mark Sweep (CMS) Collector:** It does the garbage collection for the old generation. You can limit the number of threads in CMS collector using **XX:ParalleCMSThreads=JVM option**. It is also known as Concurrent Low Pause Collector.
* **G1 Garbage Collector:** It introduced in Java 7. Its objective is to replace the CMS collector. It is a parallel, concurrent, and CMS collector. There is no young and old generation space. It divides the heap into several equal sized heaps. It first collects the regions with lesser live data.

JRE is the part of the JDK and a stand-alone JVM.

Flow control statement

Java class

Oops

Modifiers

Interface

Garbage Collector

String

Wrapper class

Exception Handling

Multithreading

Nested class

Annotations

Enumeration

Arrays

Collection Framework

Generics

Networking

AWT (Abstract Window ToolKit)

Swings

Applet in Java

Internationalization

Assertions

Other Documents:

Hibernate

Spring framework

Programming Questions: