**Java Class Loading Mechanism**

**1. Introduction**

Java is an Object oriented programming language. In Java, all code needs to be encapsulated in a class for execution. When a program’s source code is prepared, the Java compiler compiles the source code and creates a **.class** file which contains **byte code**. This .class file is sent as input to **JVM** which translates it into machine code for execution. The JVM is responsible for loading the class and getting the program ready for execution.

Before we dive into how classes are loaded, we need to understand the components which drive the loading mechanism.

**2. Components of JVM**

**2.1 Class Loading Sub System:**

This component is mainly responsible for loading of .class files. All java classes, interfaces, enum etc, have their own .class files. When a .class file is received as input, this component performs three steps which are as follows:

**2.1.1 Loading:**

During loading, a **Class** (java.lang.Class) object is created for all .class files. Since Object is super class of all classes, an additional class object is also created for **Object** (java.lang.Object) class.

Each of these Class (java.lang.Class) objects store information about each of these .class files. These Class (java.lang.Class) objects contain methods and variables of actual class, their parent class and are stored in **method area of memory**. These class objects help developers to access members of classes and obtain their prototypes.

There are three types of class loaders: Bootstrap Loader, Extension Loader and Application Loader which will be explained later.

**2.1.2 Linking:**

At this stage, the following steps are performed:

**a) Verfication:**

It ensures whether the byte code is formatted properly and correctly. It ensures security and safety.

**b) Preparation:**

During this step, all static variables are initialised to their default values. For example, default values for **int** and **double** are **0** and **0.0** respectively.

**c) Resolution:**

Source code is human-understandable. Computers cannot directly execute nothing but machine code. Therefore, all class words are replaced by their corresponding references in memory to translate the code to machine code.

**2.1.3 Initialization:**

During this phase, all static fields are initialized to their original values.

**2.2 Memory Areas:**

There are five areas in this section:

**2.2.1 Method area:**

These contain Class (java.lang.Class) objects (which contain information about classes).

**2.2.2. Heap area:**

The memory pool which contains actual objects.

**2.2.3 Stack area:**

For every thread of execution a stack exists. All these stacks reside in the stack area.

**2.2.4 PC Registers:**

Each register is for a thread of execution which contains the address of the next instruction.

**2.2.5 Native method area:**

It contains stacks for each thread of execution for native methods.

**2.3 Execution Engine:**