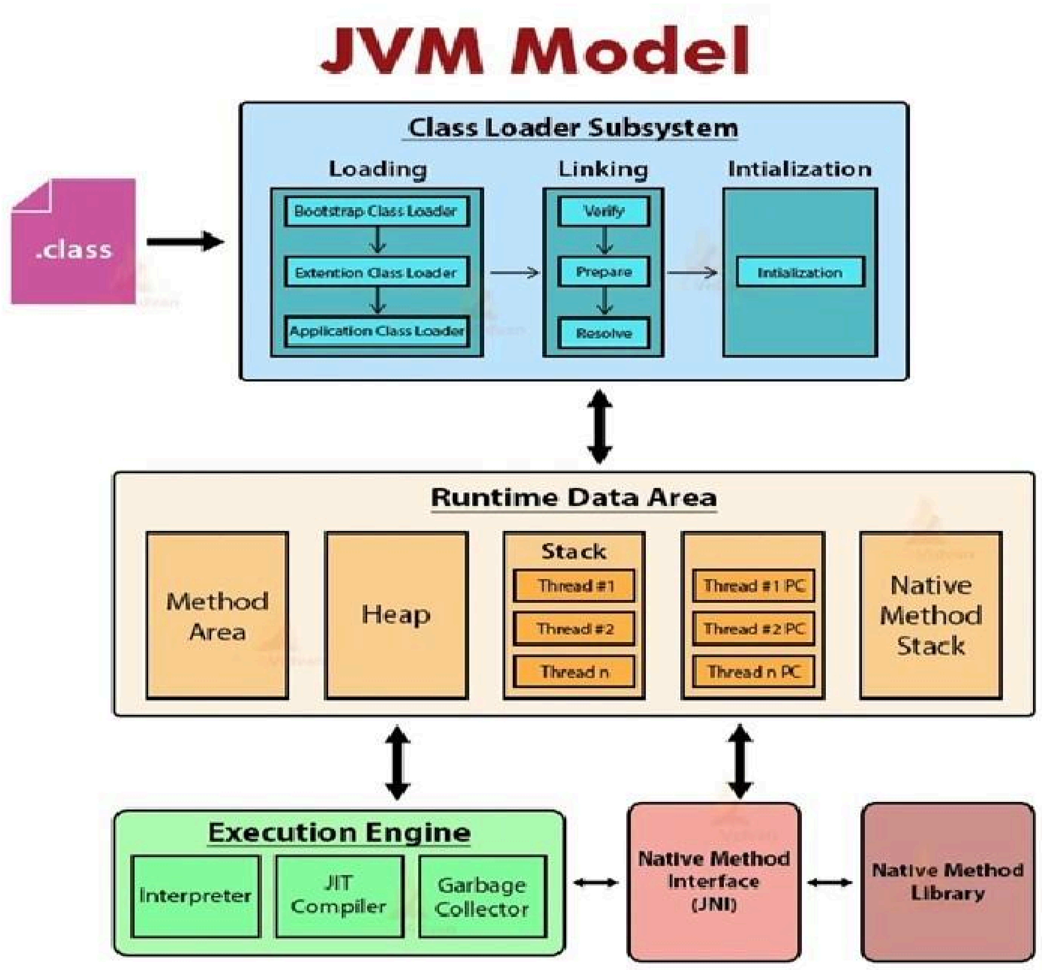
JVM_Architecture Short Notes:

The Java Virtual Machine (JVM)

- The Java Virtual Machine (JVM) is a runtime engine that **loads**, **verifies**, and **executes** Java bytecode. It provides platform independence, memory management, security, and performance optimization.



Main Components of JVM

- The JVM architecture consists of five major components:

1. Class Loader Subsystem

- Responsible for loading `.class` files into memory.

Phases:

1. **Loading** – Reads class file and stores class metadata in Method Area.
2. **Linking:**
    - **Verification** – Ensures bytecode validity and security.
    - **Preparation** – Allocates memory for static variables and assigns default values.
    - **Resolution** – Replaces symbolic references with direct references.
3. **Initialization** – Assigns actual values to static variables and executes static blocks.

Types of Class Loaders:

- **Bootstrap** – Loads core Java classes.
- **Extension** – Loads extension libraries.
- **Application** – Loads user-defined classes.

2. Runtime Data Areas (Memory Areas)

- JVM memory is divided into **five areas**:

1. **Method Area (Metaspace)**

    - Stores **class metadata**, **method code**, **static variables**..
    - One per JVM, shared by all threads.

2. **Heap Area**
    - Stores objects, instance variables, and arrays.
    - One per JVM, shared and managed by the Garbage Collector.
3. **Stack Area**
    - One stack per thread.
    - Stores method calls and local variables as stack frames.
    - Thread-safe.
4. **PC Register**
    - One per thread.
    - Holds the address of the currently executing instruction.
5. **Native Method Stack**
    - Stores native (C/C++) method calls.

3. Execution Engine

- Executes bytecode instructions.

  Components:

    - **Interpreter** – Executes bytecode line by line.
    - **JIT Compiler** – Converts frequently used bytecode (hotspots) into native machine code for better performance.
    - **Garbage Collector** – Automatically frees unused heap memory.

4. Java Native Interface (JNI)

- Acts as a bridge between Java code and native libraries written in C/C++.
  Used for system-level operations and performance-critical tasks.

5. Native Method Libraries

- Platform-dependent libraries accessed through JNI.

JVM loads class files, allocates memory, executes bytecode, manages garbage collection, and provides security and performance optimization.

The **String Constant Pool (SCP)** is a special memory area in the **Heap** used to store **unique String literals** in Java. Its main purpose is to **save memory and improve performance** by avoiding duplicate String objects.

Memory Location

- **Java 7 and later:** String Constant Pool is stored in the **Heap**
- **Java 6 and earlier:** Stored in **PermGen (Method Area)**