JVM Architecture

The **Java Virtual Machine (JVM)** is an abstract machine that enables Java programs to run on any platform. It provides **platform independence** by converting Java bytecode into machine-specific code.

Main Components of JVM Architecture

1. Class Loader Subsystem

The **Class Loader** loads class files (.class) into the JVM memory.

Types of Class Loaders:

- **Bootstrap Class Loader** – Loads core Java classes (java.lang, java.util)
- **Extension Class Loader** – Loads extension classes
- **Application Class Loader** – Loads user-defined classes

Functions:

- Loading
- Linking
- Initialization

---

2. Runtime Data Areas

These are memory areas used during program execution.

a) Method Area

- Stores class metadata
- Stores method code, static variables, constant pool

b) Heap Area

- Stores objects and class instances
- Shared among all threads
- Garbage Collection happens here

c) Stack Area

- Stores method calls and local variables
- Each thread has its own stack
- Follows LIFO (Last In First Out)

d) PC Register

- Stores address of the currently executing instruction

e) Native Method Stack

- Used for native (non-Java) methods written in C/C++

---

3. Execution Engine

Executes bytecode instructions.

Components:

- **Interpreter** – Executes bytecode line by line
- **JIT Compiler (Just-In-Time)** – Converts bytecode to native code for better performance
- **Garbage Collector** – Automatically removes unused objects from heap

---

4. Java Native Interface (JNI)

- Acts as a bridge between Java code and native libraries
- Allows Java programs to call C/C++ code

---

5. Native Method Libraries

- Contains native libraries required by native methods

JVM Architecture Diagram