**JVM Architecture Report**
Title: JVM Architecture & Execution Process
Author: Suneel Soni
Subject: Java Fundamentals Assignment

**1. Introduction**
This report explains the internal architecture of the Java Virtual Machine (JVM), the role of major components, and how Java achieves platform independence through the “Write Once, Run Anywhere” principle.

**2. What is JVM?**
The JVM (Java Virtual Machine) is a software engine responsible for executing Java bytecode.
It provides:
Memory management
Security
Platform independence
Execution environment for Java programs

**3. JVM Architecture Overview**
JVM consists of the following core components:
A. Class Loader Subsystem
Responsible for loading Java classes into memory.
Main parts:
Loading – Reads .class files
Linking
Verification
Preparation
Resolution
Initialization – Executes static blocks & static variables
B. Runtime Data Areas

These are the memory areas inside the JVM:
1. Method Area
Stores class-level data
Method info, static variables, constructors
2. Heap Area
Stores object instances
Shared among all threads
3. Stack Area
Each thread has its own stack.
Contains:
Local variables
Method calls
Stack frames
4. PC Register
Program Counter (holds the next instruction address)
5. Native Method Stack
Used for native (non-Java) code like C/C++
C. Execution Engine
1. Interpreter
Reads bytecode instruction by instruction
Slower but immediate
2. JIT Compiler (Just-In-Time)
Compiles frequently-used bytecode into machine code
Greatly improves performance

Interpreter + JIT together = Efficient execution
3. Garbage Collector

Removes unused objects from Heap
Prevents memory leaks
**4. Bytecode Execution Process (Step-by-Step)**
.java  Compiled into .class (bytecode)
Class Loader loads class into JVM
Bytecode is stored in Method Area
Execution Engine interprets or compiles bytecode
Objects created in Heap
Stack stores method-level data
GC cleans unused objects
**5. Write Once, Run Anywhere (WORA)**
This is possible because:
Java compiler generates bytecode, not machine code
JVM for each OS can run the same bytecode
So one .class file runs on Windows, Linux, macOS without modification
Diagram:
Java Code  Bytecode  JVM (Windows)
 JVM (Linux)
 JVM (Mac)
**6. Conclusion**
JVM is the core part of Java's execution model.
Its modular design (Class Loader, Runtime Areas, Execution Engine, GC) ensures portability, performance, and safety across all operating systems.