JVM (Java Virtual Machine) - Detailed Explanation

1. What is JVM?

JVM (Java Virtual Machine) is a software engine that runs Java bytecode on any operating system.

It makes Java platform-independent.

2. How JVM Works (Simple Steps)

- You write Java code (.java file).

- Java compiler (javac) converts it into Bytecode (.class file).

- JVM loads the .class file and executes the bytecode.

3. JVM Architecture

a) Class Loader: Loads .class files into JVM.

b) Runtime Data Areas (Memory):

- Method Area: Stores class info & static data.

- Heap: Stores objects.

- Stack: Stores method calls & local variables.

- PC Register: Keeps track of current executing instruction.

- Native Method Stack: Runs native (C/C++) code if needed.

c) Execution Engine:

- Interpreter: Reads and executes bytecode line by line.

- JIT Compiler: Converts frequently used code to native machine code.

d) Garbage Collector: Removes unused objects automatically.

4. Example Code (JVM Execution Example)

```
public class JVMExample {
```

```
    public static void main(String[] args) {
        int x = 10;
        int y = 20;
        int sum = x + y;
        System.out.println("Sum = " + sum);
    }
}
```

5. Step-by-Step Execution

- Class Loader loads JVMExample.class into Method Area.
- Memory Allocation:
    * Stack: x=10, y=20, sum=30
    * Heap: System.out object
- Execution Engine runs bytecode, JIT optimizes repeated code.
- Output: Sum = 30

6. Bytecode Example (via javap -c JVMExample)

```
0: bipush 10       -> Push 10 to stack
2: istore_1        -> Store in variable x
3: bipush 20       -> Push 20 to stack
5: istore_2        -> Store in variable y
6: iload_1         -> Load x
7: iload_2         -> Load y
8: iadd            -> Add x and y
9: istore_3        -> Store in sum
10: getstatic #2   -> Get System.out
13: invokevirtual #4 -> Call println()
```

19: return

7. Why JVM is Important?

- Platform Independent (Write once, run anywhere)

- Automatic Memory Management (Garbage Collection)

- Security (Bytecode verification)

- Performance (JIT optimization)