- How Java Code Actually Runs

1. Writing Source Code

You write code in .java files.

Closer to human logic.

2. Compilation (JDK → javac)

.java to Bytecode (.java → .class (Bytecode)).

Bytecode = Intermediate language, not machine-specific.

3. JVM Starts

JVM loads bytecode into memory.

Class Loader brings in your class + library classes.

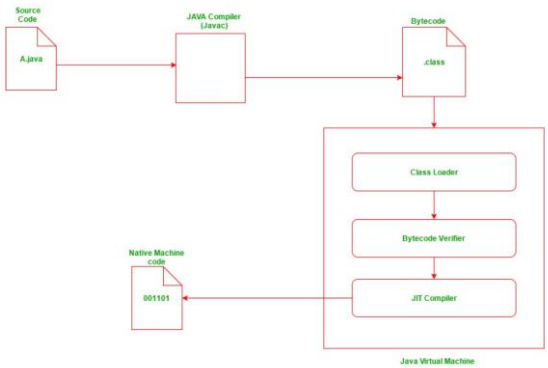
Bytecode Verifier checks for security

4. Execution in JVM

Interpreter: Reads bytecode → converts line-by-line into native(system specific code based on hardware and os)instructions.

JIT Compiler: Frequently used code (hotspot) is compiled into native machine code and cached.

5. Hardware Execution



🔹 What is Native Code?

Machine code that a specific CPU + OS can directly execute.

It's hardware-specific.

🔹 In Java Context.

Java source → compiled into bytecode (not native).

JVM → converts bytecode into native code at runtime (via interpreter or JIT).

⚡ Shortcut Memory:

👉 Bytecode = JVM language, Native Code = CPU language.

Source Code → Bytecode → Native/Machine Code → CPU executes

Java is both compiiler and Interprted language

- Compile (using javac) →

  .java → .class (bytecode).

- Run (inside JVM) →

  First the interpreter reads bytecode line by line and executes.

  While running, if some part of code is used very often (hotspot) → JIT compiler kicks in and translates that bytecode into native machine code.

  Next time, JVM runs that native code directly, which is much faster than interpreting again.

Hotspot -

👉 Some code is being run again and again (like a loop or a frequently called method) to avoid this and runs fast .

- JIT (Just-In-Time) Compiler kicks in 🚀
- JVM detects "hot spots" (code used repeatedly).
- JIT converts that bytecode into native machine code once.
- Next time, instead of interpreting, it directly runs the machine code → much faster

Memory Areas of JVM:

Method Area → class info & static

Heap → objects

Stack → local variables, references

PC Register → current instruction

Native Method Stack → non-Java code