# Java Programming Terminology:

Java programming is built on several layers of terminology, from core architectural components to object-oriented principles.

## 1. The Java Environment (Execution & Tools):

These terms describe how Java code is transformed from text into a running application:

*JDK (Java Development Kit)*: The full software package used to develop Java apps. It includes the JRE and development tools like the compiler.

*JRE (Java Runtime Environment):* The part of the JDK that allows Java programs to run but does not include development tools.

*JVM (Java Virtual Machine):* The engine that executes Java Bytecode, making Java "platform-independent".

*Bytecode:* The intermediate code (.class files) generated by the Java compiler that the JVM understands.

*JIT (Just-In-Time) Compiler:* A component of the JVM that improves performance by compiling bytecode into native machine code at runtime.

*Garbage Collector (GC):* An automatic memory management program that deletes or reclaims memory from objects no longer in use.

## 2. Core Structural Elements:

The building blocks you use to write any Java program:

*Class:* A blueprint or template used to create objects; it defines attributes (data) and methods (behavior).

*Object:* An "instance" of a class. If "Car" is the class, "your Honda" is the object.

*Method:* A block of code that performs a specific action, similar to a function in other languages.

*Variable:* A named container for storing data values (e.g., int speed = 60;).

*Package:* A namespace or folder used to group related classes and interfaces together.

*Constructor:* A special method called when an object is first created to initialize its state.

## 3. Object-Oriented (OOP) Pillars:

Java is strictly object-oriented, relying on these four (or five) key concepts:

*Encapsulation:* Hiding internal data and requiring all interaction to happen through public methods.

*Inheritance:* The process where one class (subclass) acquires the properties and methods of another (superclass) using the extends keyword.

*Polymorphism:* The ability of an object to take "many forms"—for example, a start() method behaving differently for a Car vs. a Boat.

*Abstraction:* Hiding complex implementation details and showing only the essential features of an object.

## 4. Technical Keywords & Logic:

*Interface:* A "contract" that defines methods a class must implement, but does not provide the code for them.

*Static:* Indicates that a member (variable or method) belongs to the class itself rather than to a specific object.

*Final:* Used to make a variable unchangeable, a method un-overridable, or a class un-extendable. (CONSTANTS)

*Exception:* An error event that occurs during program execution. These are handled using Try-Catch blocks to prevent crashes.