* **JDK, JRE & JVM**

**A diagram of a rectangular object

AI-generated content may be incorrect.**

**1. JVM (Java Virtual Machine)**

* Definition: An abstract machine that executes Java bytecode.
* Purpose:
  + Converts bytecode → machine code (via Interpreter or JIT compiler).
  + Provides memory management (Garbage Collection).
* Contains:
  + Class Loader → Loads .class files into memory.
  + Bytecode Verifier → Ensures bytecode is safe & valid.
  + Execution Engine → Runs the bytecode.
* Platform Dependent: JVM implementation is OS-specific.

Think of it as: The engine that runs Java programs.

**2. JRE (Java Runtime Environment)**

Think of it as: JVM + libraries = Java runner.

* Definition: A package that provides everything needed to run Java programs, but cannot compile them.
* Contains:
  + JVM
  + Core Java libraries (Collections, IO, Networking, etc.)
  + Supporting files
* Does NOT contain:
  + Java Compiler (javac)
  + Development tools (debuggers, profilers)

**3. JDK (Java Development Kit)**

* + Think of it as: JRE + Compiler & tools = Java developer toolkit.
* Definition: Full package to develop and run Java programs.
* Contains:
  + JRE (and thus JVM)
  + Development tools:
    - Java Compiler (javac)
    - Java Debugger (jdb)
    - Java Documentation Tool (javadoc)
* Types:
  + JDK Standard Edition (SE) → Core Java
  + JDK Enterprise Edition (EE) → Java EE APIs (Servlets, JSP, etc.)
  + JDK Micro Edition (ME) → Embedded/mobile devices

**1. Interpreter in JVM**

* Reads Java bytecode (.class) line-by-line and executes it immediately.
* Advantage: Starts running program quickly (no need to compile the whole thing first).
* Disadvantage: Slower execution because each line is translated every time it runs.

Think of it as:  
 Reads script → Runs it instantly → But repeats translation every time.

**2. JIT (Just-In-Time) Compiler in JVM**

* Compiles bytecode → native machine code at runtime.
* Stores the compiled code in memory so it can run directly on the CPU without re-interpreting.
* Advantage: Much faster after first run (no repeated interpretation).
* Disadvantage: Initial delay when compiling hot code.

Think of it as: Translates script into a native executable once → Reuses compiled version → Runs much faster.