3️⃣ Java Program Execution Flow

1. **What happens when you compile a Java program?**
    - When a Java program is compiled, the Java compiler (javac) converts the source code (.java) into bytecode (.class), which can be executed by the JVM on any platform.

2. **What happens when you run a Java program?**
    - When a Java program is run, the JVM loads the .class file, verifies the bytecode, and executes it, converting bytecode into machine code for the system.

3. **Explain Java program execution step by step.**
    - Compilation: javac converts .java source code into .class bytecode.
    - Class Loading: JVM's ClassLoader loads the required .class files.
    - Bytecode Verification: JVM checks bytecode for security and correctness.
    - Execution: Execution Engine (Interpreter/JIT) converts bytecode into machine code and runs it.
    - Memory Management: JVM allocates memory and performs garbage collection.
    - **NOTE: Java program execution involves compile → load → verify → execute → manage memory.**

4. **What is bytecode and where is it stored?**
    - Bytecode is an intermediate code generated by the Java compiler (javac), which is platform-independent.
    - It is stored in .class files and executed by the JVM.

5. **What is classloader and its types?**
    - A ClassLoader is a part of JVM that loads Java classes into memory at runtime.
    - Types of ClassLoader:
        - Bootstrap ClassLoader – Loads core Java classes (java.lang, java.util).
        - Extension ClassLoader – Loads classes from JDK extension libraries.
        - Application ClassLoader – Loads classes from the application classpath.
    - **NOTE: ClassLoader dynamically loads classes into JVM using a hierarchical delegation model.**

6. **What is JIT compiler?**
    - JIT (Just-In-Time) Compiler is a part of the JVM Execution Engine that converts frequently used bytecode into native machine code at runtime, improving performance.

- **NOTE: JIT compiles bytecode into machine code at runtime to make Java faster.**

7. What is runtime data area?

- The Runtime Data Area is the memory area of the JVM created when a Java program runs, used to store data during execution.
- It includes:
    - Method Area
    - Heap
    - Stack
    - PC Register
    - Native Method Stack
- **NOTE: Runtime Data Area is JVM memory used to store and manage data while a Java program is executing.**

8. What is stack memory?

- Stack memory is a part of JVM memory used to store method calls, local variables, and references.
- Key Points:
    - Each thread has its own stack.
    - Memory is allocated and deallocated automatically.
    - Faster than heap memory.
- **NOTE: Stack memory stores method execution details and local variables for each thread.**

9. What is heap memory?

- Heap memory is a part of JVM memory used to store objects and class instances.
- Key Points:
    - Shared among all threads.
    - Memory is managed by Garbage Collector.
    - Larger but slower than stack memory.
- **NOTE: Heap memory stores objects and is managed by the garbage collector.**

10. Difference between stack and heap.

Stack Memory	Heap Memory
Stores method calls and local variables	Stores objects and class instances
Each thread has its own stack	Shared among all threads
Memory allocation is automatic	Memory managed by Garbage Collector
Faster access	Slower than stack
Size is limited	Larger memory size

- **NOTE: Stack is for method execution, Heap is for object storage.**