The execution of a Java program involves several steps, including class loading, interpretation, Just-In-Time (JIT) compilation, and execution. Let's break down the process step by step:

1. **Source Code:**
   * You start with your Java source code, typically saved in **.java** files.
2. **Compilation:**
   * Use the Java Compiler (**javac**) to compile the source code into bytecode. Bytecode is a set of instructions for the Java Virtual Machine (JVM) rather than platform-specific machine code.
3. **Bytecode:**
   * The compilation generates **.class** files containing bytecode. Bytecode is platform-independent, allowing it to run on any system with a compatible JVM.
4. **Class Loading:**
   * The Java ClassLoader comes into play. It loads classes (bytecode) into memory as they are referenced in the program.
   * The ClassLoader has a hierarchy: Bootstrap ClassLoader (loads core Java classes), Extension ClassLoader (loads classes from extensions), and System ClassLoader (loads user-defined classes).
5. **Bytecode Verification:**
   * Before execution, the JVM performs bytecode verification to ensure it adheres to Java's safety and security rules.
6. **Interpreter Execution:**
   * Initially, the JVM may use an interpreter to execute the bytecode line by line. While this is slower, it allows for quick startup.
7. **JIT Compilation:**
   * As the program runs, the JVM monitors the frequently executed methods. If a method is deemed "hot" (executed frequently), the JVM may decide to apply JIT compilation.
   * JIT compilation translates bytecode into native machine code for the specific platform, optimizing performance.
   * The generated native code is cached, reducing the need for recompilation.
8. **Execution:**
   * The JVM executes the optimized native code, resulting in faster and more efficient program execution.
9. **Runtime Services:**
   * During execution, the Java Runtime Environment (JRE) provides various runtime services, including memory management, garbage collection, and thread management.

In summary, the process involves compiling Java source code into bytecode, loading classes into memory, verifying bytecode, interpreting initially, JIT compiling hot methods for optimization, and finally executing the native code on the JVM. The ClassLoader, JIT compiler, JRE, and JVM work collaboratively to execute Java programs efficiently.