1Z0-808 Exam Topic Reviewer

TopicId: 1000
Topic: Java Environment and Fundamentals

August 5, 2025

Introduction: Thinking Like the Compiler

Welcome, future Java certified professionals! Our journey begins with the absolute fundamentals. As you heard from my colleague, the 1Z0-808 exam isn't just about writing code; it's about deeply understanding the rules of the Java language. Many questions will present code that looks strange but is perfectly valid. Your job is to act like the Java compiler and JVM—to analyze, not just assume. This module lays that critical foundation. Let's master the environment and the core mechanics of a Java program.

1 The Java Ecosystem: JDK, JRE, and JVM

The terms JVM, JRE, and JDK are often used interchangeably, but for the exam, you must know their precise differences.

1.1 Java Virtual Machine (JVM)

The JVM is an *abstract machine.* It's a specification that provides a runtime environment in which Java bytecode can be executed.

- **Role:** It loads, verifies, and executes bytecode. It also manages memory (Garbage Collection).
- **Platform-Dependent:** While your Java code is platform-independent, the JVM itself is not. There are different JVM implementations for Windows, macOS, and Linux.
- **Core Concept:** This is what enables Java's "Write Once, Run Anywhere" (WORA) principle. You compile your code once into universal bytecode, and the specific JVM on any machine knows how to run it.

1.2 Java Runtime Environment (JRE)

The JRE is the *implementation* of the JVM. It provides the minimum requirements for **running** a Java application.

- **Contents:** It contains the JVM, core Java class libraries (like `java.lang`, `java.util`), and other supporting files.
- **Purpose:** If you only want to run a pre-compiled Java program (e.g., a `.jar` file), you only need the JRE. You cannot compile code with just a JRE.

1.3 Java Development Kit (JDK)

The JDK is a superset of the JRE. It contains everything needed to **develop**, compile, and run Java applications.

- **Contents:** It includes the entire JRE plus development tools. The most important ones for the exam are:
  - `javac`: The Java compiler.

- `java`: The Java launcher (which starts the JVM).
- `jar`: The archiver, for packaging files.
- `javadoc`: The documentation generator.

Hierarchical Relationship: The relationship is simple: **JDK** $\supset$ **JRE** $\supset$ **JVM**. You develop with a JDK. You run with a JRE. The JRE uses a JVM.

2 The Life of a Java Program: From Source to Execution

Understanding this two-step process is fundamental.

(a) **Step 1: Writing Source Code**
You write your code in a plain text file with a `.java` extension. This file contains human-readable Java statements. For a public class named `MyApp`, the file must be named `MyApp.java`.

(b) **Step 2: Compilation**
You use the Java compiler, `javac`, which is part of the JDK. The compiler reads your `.java` file, checks for syntax errors, and translates it into **bytecode**.

```
javac MyApp.java
```

If successful, this creates a new file named `MyApp.class`. This file contains platform-independent bytecode, which is not human-readable.

(c) **Step 3: Execution**
You use the Java launcher, `java`, which is part of the JRE. This command starts the JVM.

```
java MyApp
```

Crucial Exam Tip: Notice there is **no** `.class` extension in the run command. Providing it (`java MyApp.class`) will result in an error. The JVM then performs several actions: loading the bytecode, verifying it for security, and finally executing it by converting the bytecode to native machine code.