Lec 7 Compilation/JDK

This lesson explains how Java programs are compiled and run, focusing on the roles of JDK (Java Development Kit), JRE (Java Runtime Environment), and JVM (Java Virtual Machine). It highlights how Java achieves platform independence by using its own runtime environment instead of relying on the operating system, and how compiled Java code (bytecode) can run on any system with the appropriate JRE.

Understanding Java Program Structure

The basics of Java program syntax are introduced, including keywords like class, method, string, and static. These are essential building blocks for writing Java programs.

Introduction to Java Architecture

Java architecture is compared to building architecture, where understanding the components and materials is crucial. In Java, knowing what makes up the environment is necessary to understand how programs run.

Components of JDK (Java Development Kit)

JDK is described as a toolkit containing the compiler, debugger, and other tools needed for Java development. The compiler (javac) translates Java code into bytecode.

Role of JRE (Java Runtime Environment)

JRE is a part of JDK that provides the environment needed to run Java programs, including standard libraries and the JVM. Unlike C or C++, Java does not use the operating system directly as its runtime environment.

C and C++ programs are platform dependent because they rely on the operating system to run. In contrast, Java uses its own runtime (JRE), making Java programs platform independent.

The Role of JRE in Platform Independence

Java programs do not run directly on Windows or any OS but require a specific JRE for each platform. This JRE acts as a bridge between the Java program and the system, ensuring the same Java code runs everywhere.

Inside the JRE: Libraries and JVM

JRE contains standard libraries (pre-written code) and the JVM, which is crucial for executing Java programs. The JVM is responsible for managing the control flow of the program and calling the main method.

JVM: The Heart of Java Execution

The JVM loads, verifies, interprets, and executes the bytecode. It ensures security and proper execution by checking the bytecode before running it, and uses a Just-In-Time (JIT) compiler to convert bytecode to machine code.

Platform Independence of Java Programs and Platform Dependence of JRE

While Java programs (bytecode) are platform independent, the JRE is platform dependent because each operating system requires a specific implementation of the JRE to run the bytecode.

The Compilation and Execution Process

Java source code is compiled by the JDK's compiler into a .class file containing bytecode. This bytecode can be sent to others and run on any system with the appropriate JRE, without needing recompilation.

Step-by-Step Java Program Execution

The process starts with compiling the .java file into bytecode (.class file). The JVM then loads, verifies, interprets, and compiles this bytecode into machine code for the specific platform, finally producing output.

Bytecode: Universal, Intermediate, or Magic Code

Bytecode is also called universal or intermediate code because it can run on any platform with the right JRE. Its universality is key to Java's platform independence.

Internal Steps of JVM During Execution

The JVM loads the bytecode, verifies it, interprets it, and then uses the JIT compiler to convert it into native machine code, which is then executed to produce the program's output.

Recap: JDK, JRE, and JVM Roles

The JDK is needed for development and compilation, the JRE for running programs, and the JVM for executing bytecode. The process ensures that Java programs can run on any system with the correct JRE.

Moving Ahead in Java Programming

After understanding the architecture and internal working, the focus will shift to practical programming, applying these concepts in real code.