**How the Java Compilation Process Works**

Java is a programming language that uses a two-step compilation process. First through an OS compiler and then a virtual machine (JVM). The JVM, or Java Virtual Machine is custom-built for every operating system, hence making Java a platform independent programming language.

**Compilation**

First, the source code is passed through the compiler that encodes the source code into a code for machine independent encoding known as a Bytecode. Converting the source code into bytecode involves several steps. The compiler first parses the source code and maps the resulting token sequence into AST-Nodes, then the compiler enters the symbols for the definitions into the symbol table. After that the compiler then attributes the syntax trees, including name resolution, type checking, and constant folding. Next, the compiler performs dataflow analysis on the trees from the previous step. This includes checks for assignments and reachability. Finally, the compiler rewrites the AST and translates away some syntactic sugar and generates the class file.

**Execution**

The class files generated by the compiler are independent of the machine or the OS, which allows them to be run on any system. To run the program, the main class file is passed to the JVM, then it is loaded into the memory, and then verified by the bytecode verifier. The verifier checks if the variables are initialized before they are used, method calls match the types of object references, rules for accessing private data and methods are not violated, local variable accesses fall within the runtime stack, and finally checks if the run time stack does not overflow. If any of the checks fail, the verifier does not allow the class to be loaded. The compiler then executes the final step which converts the bytecode into machine code, ready to be used.