**Java Compilation Process**

Java code must go through compilation before its contents are executed. The source .java code must be encoded into machine-independent encoding known as bytecode before being converted into machine code. In simple terms, bytecode is a set of instructions for the Java virtual machine that will be held within a .class file. The transformation in itself follows a step-by-step process.

Here are the steps are taken in the compilation process of java:

1. Parse: .java source file is read and mapped into AST (Abstract Syntax Tree) Nodes.
2. Enter: Symbols for definitions are entered into the symbol table.
3. Process Annotations: If requested, specified compilation units will process annotations.
4. Attribute: The syntax tree is attributed to check for errors such as name resolution and type checking.
5. Flow: Dataflow analysis is checked in this stage. This includes assignments and reachability.
6. Desugar: AST is rewritten and translated into syntactic sugar.
7. Generate: .class file is generated.

Once the .class file has been made, it has to go through a class loader and a bytecode verifier before it could be transformed into machine code.

In a class loader, the main class and the referenced classes will be called. This is where all the required classes will be readied to execute the program.

The bytecode verifier acts as a final checkpoint before the bytecode is converted into machine code. It makes sure that the code does not have anything that could damage the system if ran. If it finds something that is potentially damaging, the classes will not be loaded. Some of the things it checks for are:

* Variables are initialized before they are used
* Method calls match the type of object referenced
* Access parameters (e.g private, public, etc.) are not violated
* Run time stack does not overflow

Once the above process is done, the program will be converted into machine code via the Just-in-Time (JIT) compiler. The machine code is then used to execute the program.