What I have learnt about Java compilation process:

The simple way to explain it is that when we compile the source code/the .java file, It is going to be passed to an OS independent compiler to be encoded into a bytecode before then is passed to the Java Virtual Machine to be converted into machine code to then be readable by the machine.

To further explain it we need to look back at the process of encodement of the source code to the bytcode. When the source code is passed to the system independent compiler, each class in the source code is going to be stored into a separate .Class files. While it is converting the source code to the byte code, the compiler would follow these following steps.

It will first **Parse** the source code and map the resulting token into AST (Abstract Syntax Tree)-Nodes. Then it will **enter symbols** for each of the definitions into a symbol table. It will then **process annotations** that is requested in the specified compilation units continued by attributing the Syntax trees that includes name resolution, type checking, and constant folding. Next, it would **performs dataflow analysis** on the trees from the Syntax Trees attribution that includes checks for assignments and reachability. Finally after it rewrites the AST and translates away some syntactic sugar by the process that’s called **Desugar**, it will then **generate** it into .Class files.

After those .Class files are generated, it will be then passed into the Java Virtual Machine and then goes through three main stages before the final machine code is executed. Those stages are:

1. Class Loader

As the name suggests, This process involves loading the classes either with the default order through JVM’s default setting (Primordial class loader) or with the used defined class loader to customize which class is loaded first (non-primordial class loader). The latter is the preferred way of doing it.

1. Bytecode Verifier  
   The bytecode verifier is going to check the instruction that is generated by the class loader to be problem free. If it detects that something could be potentially damaging to the machine, the verifier won’t allow the class to be loaded.
2. JIT (Just-In-Time) Compiler

The JIT compiler is going to be converting the loaded the bytecode into machine code to then be executed by the machine.