Izvorni '.java' file se na početku kompajlira. To znači da se izvorni kod „prevodi“ u strojni jezik; Bytecode. Sadržaji svake klase iz '.java' filea se spremaju u zasebni '.class' file.

Tijekom kompajliranja u Bytecode, izvorni kod prolazi kroz naredno objašnjene korake:

1. Raščlanjivanje (Parse) – 'čita' '.java' izvorni file i označava sve u AST
2. Unos (Enter) – Unosi simbole iz tablice sa simbolima gdje je potrebno po AST onzaci
3. Obrađivanje (Process Annotations) – Ako je zatraženo, obrađuju se bilješke u specifičnim kompajliranim jedinicama
4. Karakteriziranje (Attribute) – Karakterizira poredak sintakse (syntax tree). Ovaj korak uključuje provjeravanje naziva i tipova
5. Tok (Flow) – Analiza toka podataka iz prethodnog koraka.
6. Desugar – Obavlja se drukčiji zapis AST-a i prevode se neki sintaktični 'dodatci'
7. Generate (Generiranje) – Generira se '.class' file

'.class' file koji je generiran kompajliranjem je neovisan o stroju ili operativnom sustavu, stoga se može pokrenuti na bilo kojem uređaju. Da se izvrši '.class' file, koji u sebi ima 'main' metodu, se prosljeđuje do JVM (Java Virtual Machine) i prolazi kor tri glavna koraka prije nego se izvrši strojni kod. Ta tri koraka su:

1. ClassLoader
2. ByteCode Verifier
3. Just-In-Time Compiler

ClassLoader:

Glavna klasa se učitava u memoriju zaobilazeći '.class' file kroz JVM, pozivanjem na JVM. Sve druge klase koje se spominju u programu se očitavaju u 'class loader'. 'Class loader' ,koji je i sam objekt, stvara prazno ime i prazna tijela klasa ({}) koja su pozvana String nazivom. Postoje dvije vrste 'class loadera' : iskonski (primordial) i ne-iskonski (non-primordial). Iskonski 'class loader' se nalazi u svim JVM-ima i to je zadani 'class loader'. Ne-iskonsi 'class loader' je definiran od strane korsinika, isprogramiran. Programiranjem ne-iskonskog 'class loadera' se postiže prilagođivanje učitavanja klase i ,ako postoji, preferiran je 'class loader' za učitavanje klasa.

ByteCode Verifier:

Nako što se ByteCode klase učita od strane 'class loadera' mora biti pregledan 'ByteCode Verifierom'. 'ByteCode Verifier' za zadaću ima da provjeri da instrukcije ne izvode opasne radnje. Naredno su neke provjere koje se obavljaju:

* Varijable se inicijaliziraju prije upotrebe
* Pozivi metoda odgovaraju tipovima referenci objekata
* Pravila za pristupanje privatnim podatcima i metodama nisu prekršena
* Pristup lokalnim varijablama spada u 'runtime stack'
* 'Runtime stack' se ne smije 'preliti'

Ako išta od navedenog ne prođe provjeru klasama nije dopušteno učitavanje!

Just-In-Time Verifier:

Završni korak bilo kojeg Java programa, u kojem se 'bytecode' pretvara u strojni kod. Korištenjem JIT kompajlera hardware može izvesti autohton kod, za razliku od JVM koji interpretira iste sekvence 'bytecodea' uzastopno, a to za kaznu ima duže trajanje kompajliranja.

Izvor 1: [Geeks for Geeks](https://www.geeksforgeeks.org/compilation-execution-java-program/)

Izvor 2: [Oracle.com](https://www.oracle.com/java/technologies/compile.html)

\*Note: Većina informacija pronađena je na Geeks for geeks stranici, Oracle je bio korišten za provjeru nekih podataka i tvrdnji.