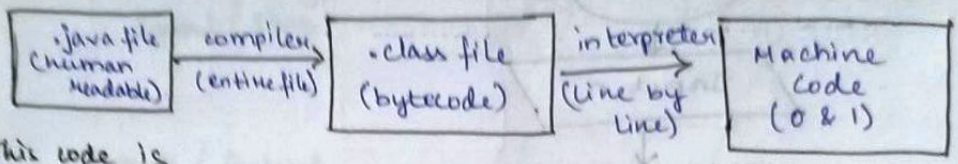
Day 8: Java

How does the Java code execute?



This code is the source code.

1. this code will not directly run on a system
2. We need JVM to run this
3. Reason why Java is platform independent.

→ because the system only knows machine code.

→ It is an interpreter that compiles the code line by line.

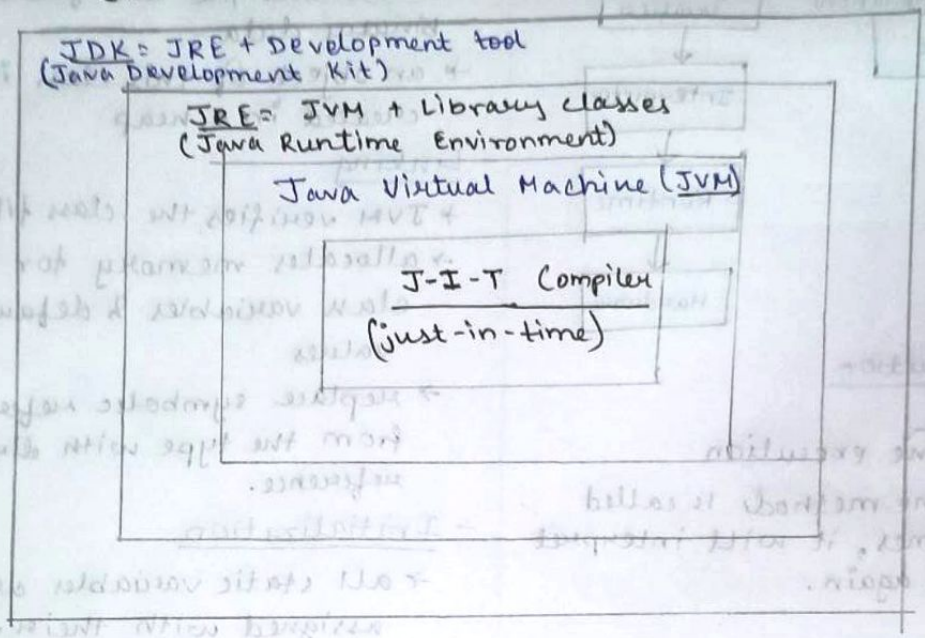
→ because java compiler firstly converts the source code to bytecode. This bytecode is converted to machine code by a software called JVM (Java Virtual Machine), which recognizes the platform it is on and converts the bytecode to "native" machine code.

More on Platform Independence

- Platform independence means that the bytecode can run on all operating system.
- After compiling the c/c++ source code in any platform we get .exe file which can only be executed in that platform. This makes these languages platform dependent.
- In Java, after compiling we get bytecode which can be converted to machine code by JVM on any platform.
- There are different JVM for different platforms. This makes JVM platform dependent. But, it does the same work on every platform. That means, it can convert the same bytecode on every platform to the native machine code.

ARCHITECTURE OF JAVA

JDK vs. JRE vs. JVM vs JIT



What is JDK?

- It is a package that we can download from internet. It provides an environment to develop and run the java program.
- It consists of → development tools
  - → JRE - to execute program.
  - → a compiler - javac
  - → archiever - jar
  - → docs generator - javadoc
  - → interpreter / loader.

What is JRE?

- It is an installation package (a set of files) that provides an environment to only run the program.
- It consists of → Deployment technologies
  - → User interface toolkits
  - → Integration libraries
  - → Base libraries
  - → JVM
- After we get the .class file (bytecode), the next things that happen at runtime are:-
  - → class loader loads all the classes needed to execute the program.
  - → JVM sends the code to Bytecode Verifier to check the format of the code.

Compile time

.java file

javac (compilation)

.class file

Runtime

class loader

Byte code Verifier

Interpreter

Runtime

Hardware

JVM execution

- Interpreter
  - → Line by line execution
  - → when one method is called many times, it will interpret again and again.
- JIT
  - → those methods that are repeated, JIT provides direct machine code so re-interpretation is not required.
  - → makes execution faster
- Garabage Collector.

(How JVM works)

class loader:

- Loading
  - → reads .class file and generates binary data
  - → an object of this class is created in a heap
- Linking
  - → JVM verifies the .class file
  - → allocates memory for class variables & default values
  - → replace symbolic references from the type with direct reference.
- Initialization
  - → all static variables are assigned with their values defined in the code and static block.

JVM contains the stack and Heap memory allocations.

Collectively performance of execution