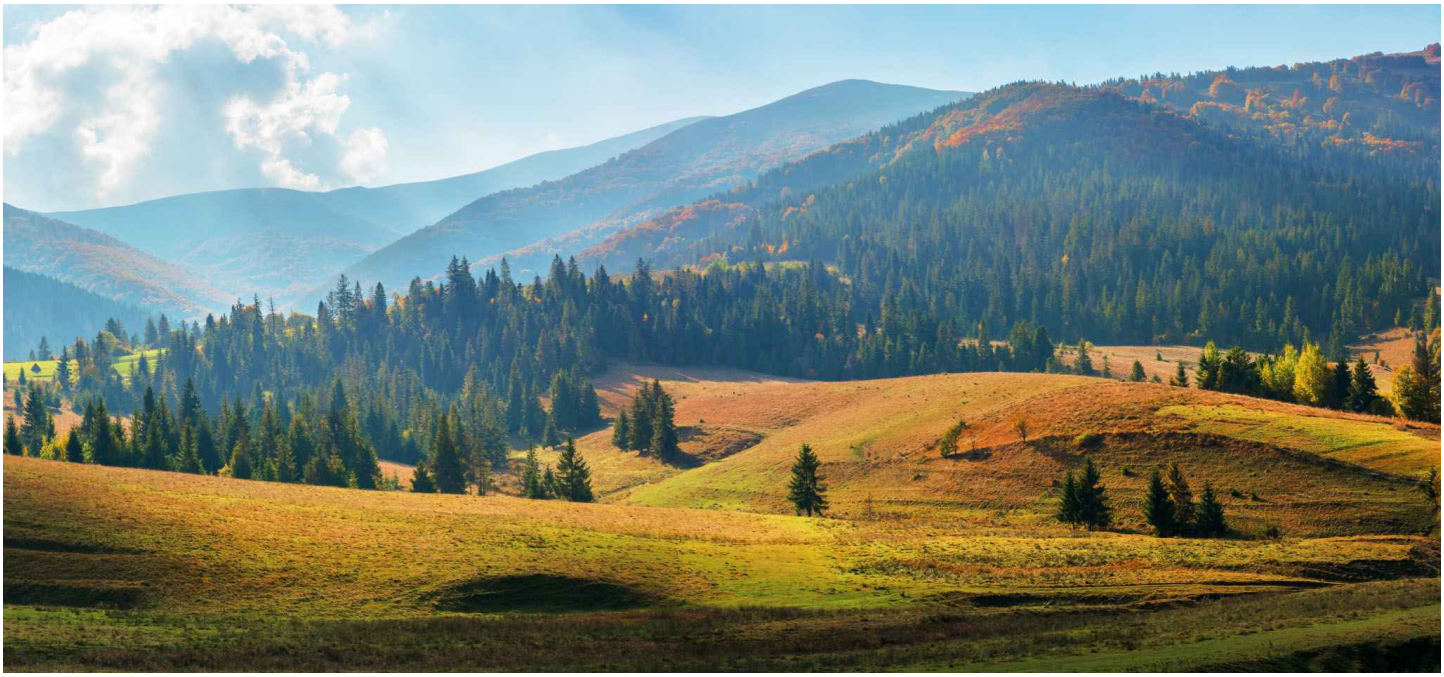


1-Introduction to Java

Author: Vincent Lau

Learning Objectives

- Describe the difference between compiled and interpreted languages
- Explain the purpose of the JVM, JRE and JDK
- Explain how Java is platform independent
- Illustrate the Java delivery process
- Compile and execute Java programs in the command line

The History of Java

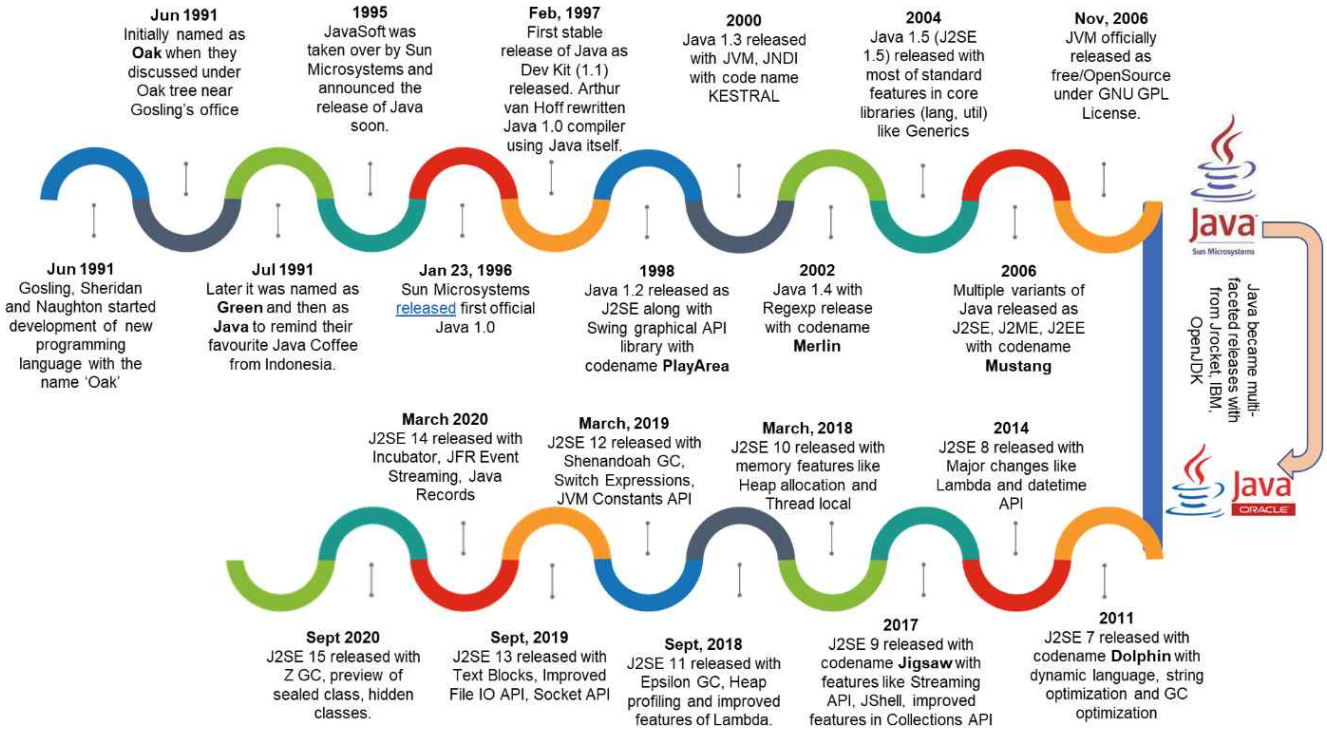


Figure: History and Evolution Timeline of Java.

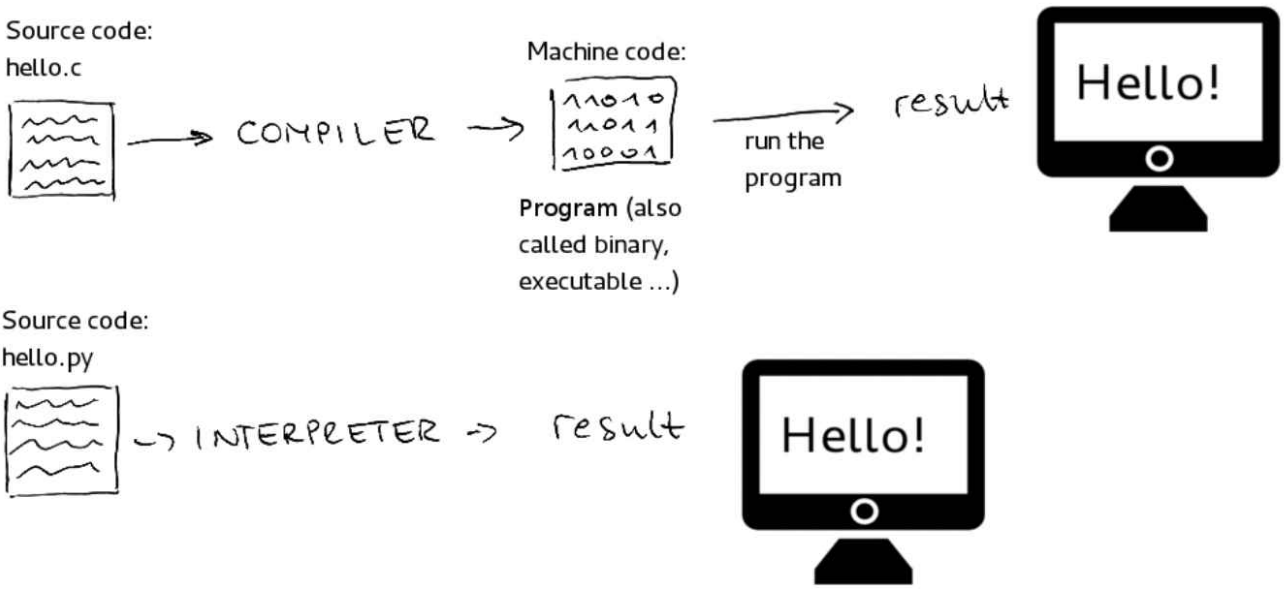
Java Version

- *Java* was developed by *James Gosling* at *Sun Microsystems Inc* in the year 1991, later acquired by *Oracle Corporation* in 2010. Java is the name of an **island** in Indonesia where the first coffee (named java coffee) was produced.

Oracle Java SE Support Roadmap*†				
Release	**GA Date**	**Premier Support Until**	**Extended Support Until**	**Sustaining Support**
7 (LTS)	July 2011	July 2019	July 2022*****	Indefinite
8 (LTS)**	March 2014	March 2022	December 2030*****	Indefinite
9 (non-LTS)	September 2017	March 2018	Not Available	Indefinite
10 (non-LTS)	March 2018	September 2018	Not Available	Indefinite
11 (LTS)	September 2018	September 2023	September 2026	Indefinite
12 (non-LTS)	March 2019	September 2019	Not Available	Indefinite
13 (non-LTS)	September 2019	March 2020	Not Available	Indefinite
14 (non-LTS)	March 2020	September 2020	Not Available	Indefinite
15 (non-LTS)	September 2020	March 2021	Not Available	Indefinite
16 (non-LTS)	March 2021	September 2021	Not Available	Indefinite
17 (LTS)	September 2021	September 2026****	September 2029****	Indefinite
18 (non-LTS)	March 2022	September 2022	Not Available	Indefinite
19 (non-LTS)***	September 2022	March 2023	Not Available	Indefinite
20 (non-LTS)***	March 2023	September 2023	Not Available	Indefinite
21 (LTS)***	September 2023	September 2028	September 2031	Indefinite

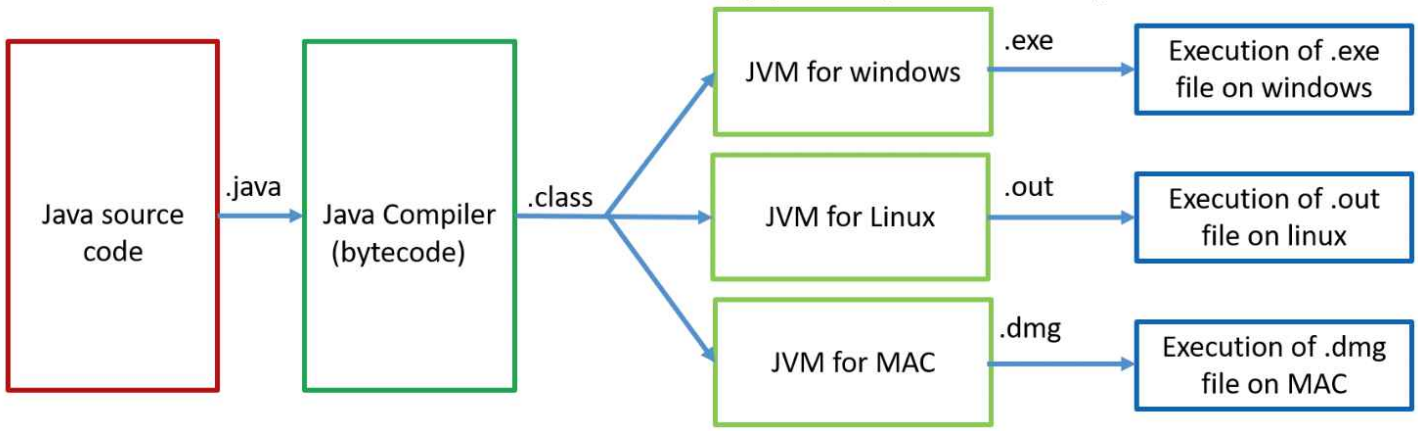
- Oracle has revamped Java starting from release control movement from 1-2 years to 6 months once. There have been tremendous changes, new feature additions.
- `Java 8 LTS` is one of the remarkable versions to let developers regain the market space.
- `Java 11 LTS` was released on Sep 2018, while `Java 17 LTS` was released on Sep 2021.

Compiled Language vs Interpreted Languages

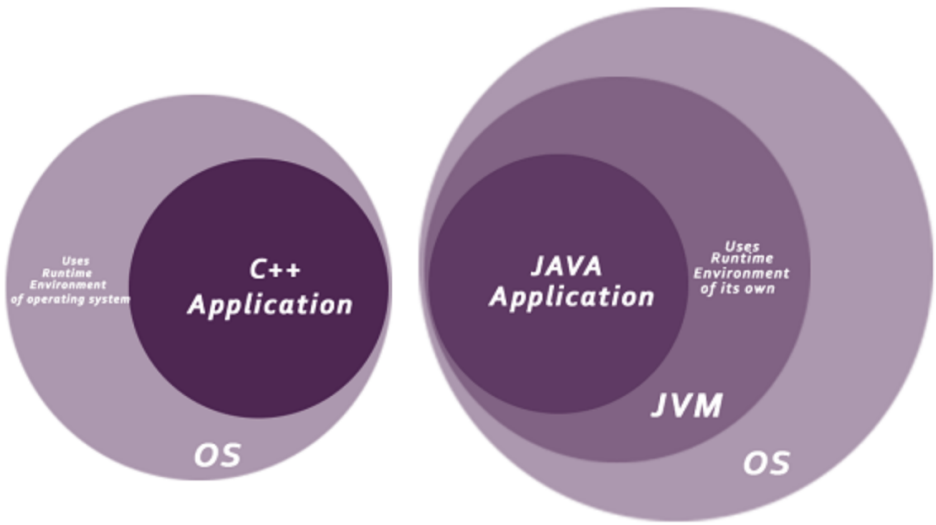


`Compiled Language`	`Interpreted Language`
The code of compiled languages can be executed directly by the CPU after compilation.	A program written in an interpreted language is interpreted by the interpreter line by line at runtime.
The source code must be transformed into machine readable instructions prior to execution.	It does not compile the source code into machine language prior to running the program.
Compiled programs cannot be modified during runtime in production.	Interpreted programs can be modified while the program is running.
Delivers better performance, in terms of runtime speed.	Delivers relatively slower performance. But with the help of Just-In-Time (JIT) Compilation, the gap is shrinking.
Examples: C, C++, Java, Go, etc	Examples: PHP, Ruby, Python, JavaScript, etc

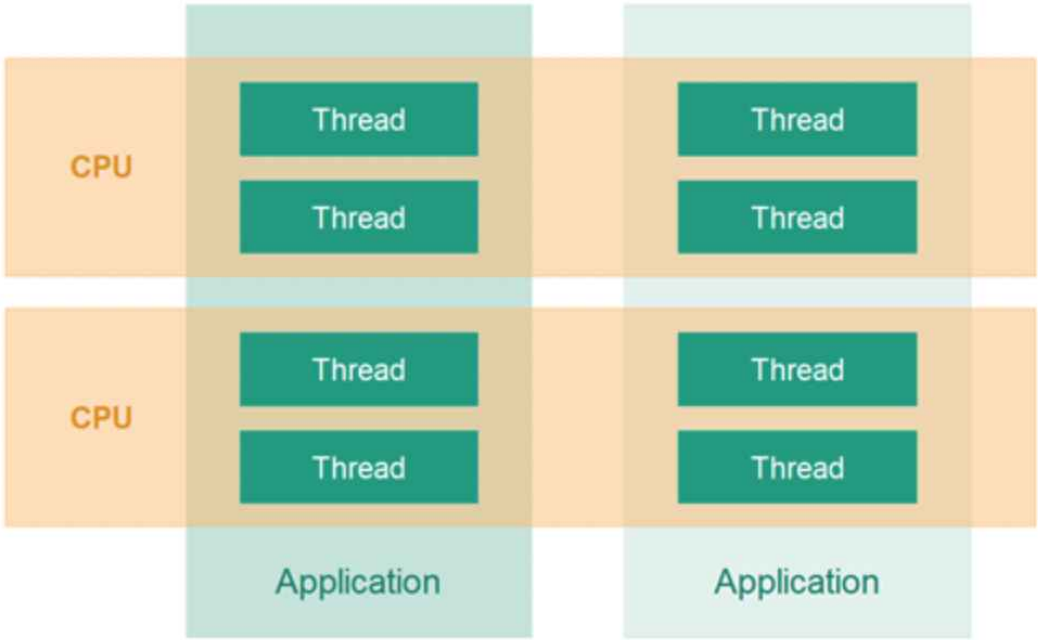
Main Characteristics



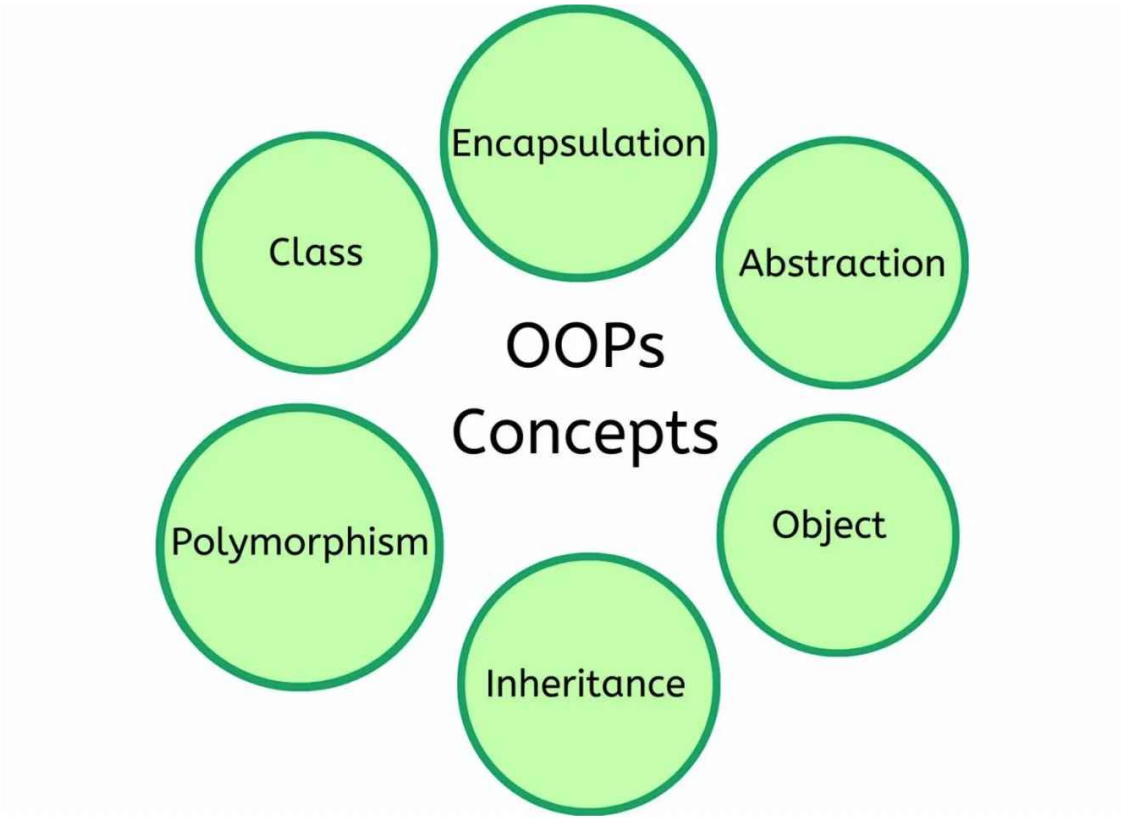
- **Platform independent** - ***Write once, run anywhere - with JVM***.
- **Simple** - Compared to the C/C++ language, Java does not have complex features such as pointers handling explicitly, operator overloading, multiple inheritance or explicit memory allocation.



- **Robust and Secure** - Features such as strong type checking, compile-time syntax checking, exception handling, **automatic memory allocation** and garbage collection. They all contribute to the robustness and security of the Java language.
- **Distributed** - Java applications are built to be distributed such that it is easy to connect multiple Java applications via Internet with the help of Remote Method Invocation (RMI). Networking capability is inherently integrated into Java, so writing network programs is like sending and receiving data to and from a file.

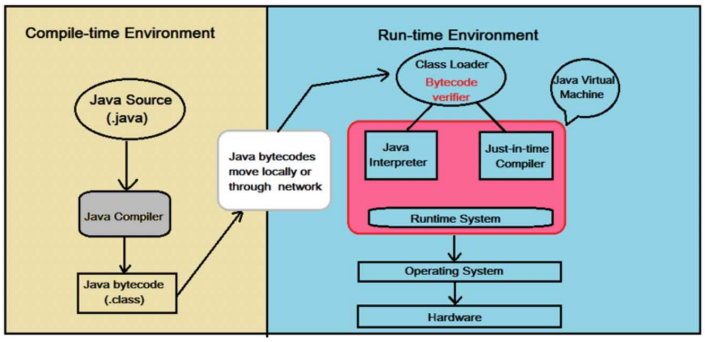


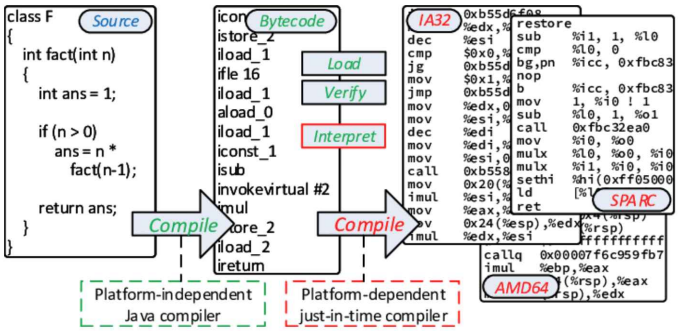
- **Multithreading** - Java allows concurrent execution of multiple threads to maximize CPU utilization.
- **High Performance** The execution of Java programs is optimized by the use of Just-In-Time (JIT) compiler.



- **Object-Oriented** - The Java language is all about objects. An object is made of attributes and behaviors. Modeling real world business logic with objects allow for modularity and reusability.

Delivery Process (Compile-time to Run-time)



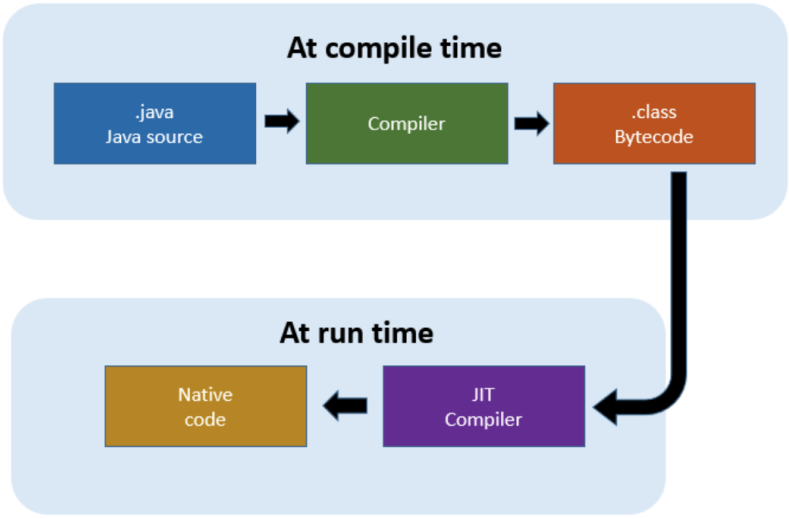


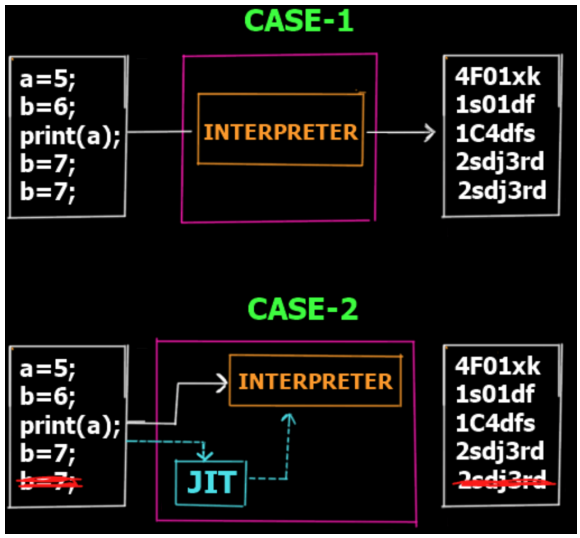
- Java is considered **both a compiled and an interpreted language**.
- `.java` files which contain source code are **compiled** into `.class` files which contain *bytecodes,* by the Java compiler (**javac**). The JVM (Java Virtual Machine) then **interprets** the Java *bytecodes,* and executes them at runtime.
- Through the JVM, the same is capable of running on multiple operating systems *(build once, run anywhere)*.

The Java Platform

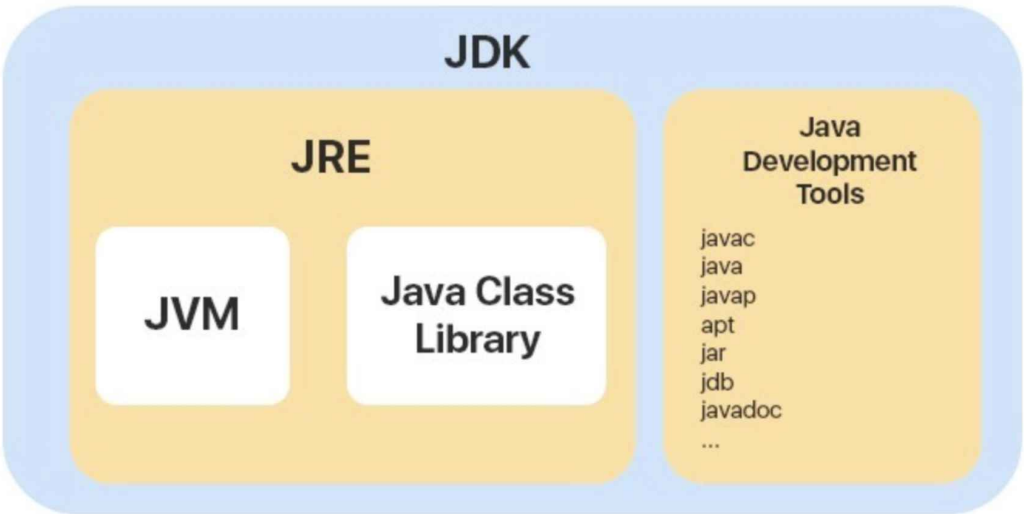
- A ***platform*** is the hardware or software environment in which a program runs.
- The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.
- The Java platform has two components:
  - The *Java Virtual Machine (JVM)*
  - The *Java Application Programming Interface* (API)
- The API and Java Virtual Machine insulate the program from the underlying hardware.

Just-In-Time (JIT) Compiler





- JIT compiler compiles the bytecode of the frequently-called methods into native code at run-time. Also, it would remove duplicated codes, which are unnecessary. Hence it is responsible for the optimization of the Java programs.
- JVM automatically monitors which methods are being executed. Once a method becomes eligible for JIT compilation, it is scheduled for compilation into machine code. This method is then known as a *hot method*. This compilation into machine code happens on a separate JVM thread.
- As a result, it does not interrupt the execution of the current program. After compiling the frequently-executed code into machine code, it runs faster.



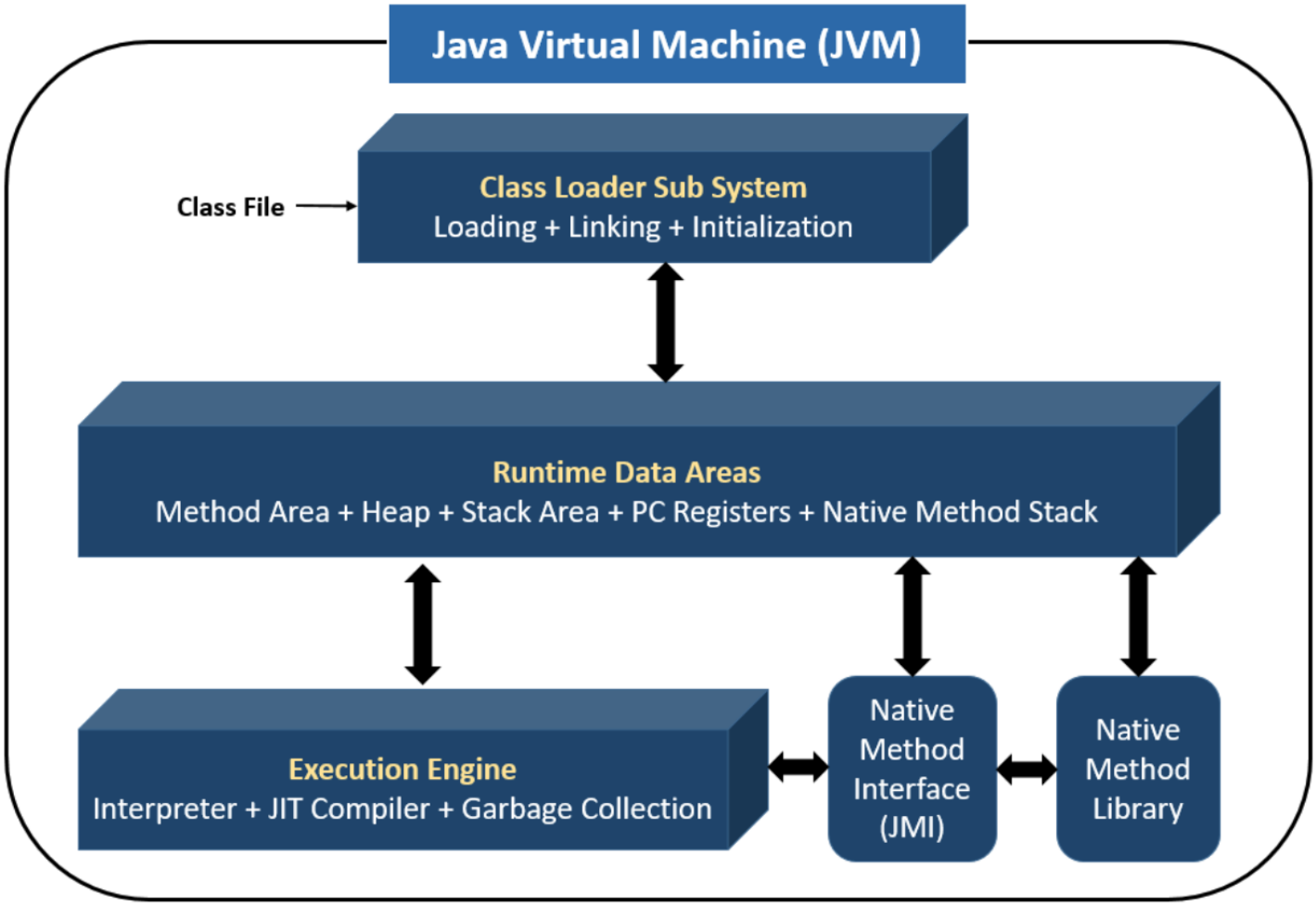
JVM = Only Runtime environment for executing the Java byte code.

JRE = Java Virtual Machine (JVM) + Libraries to run the application.

JDK = Java Runtime Environment (JRE) + Development tools.

JVM

= Virtual Machine (JVM) is an abstract machine responsible for compiling and executing Java code. It is a part of the Java Runtime Environment (JRE).



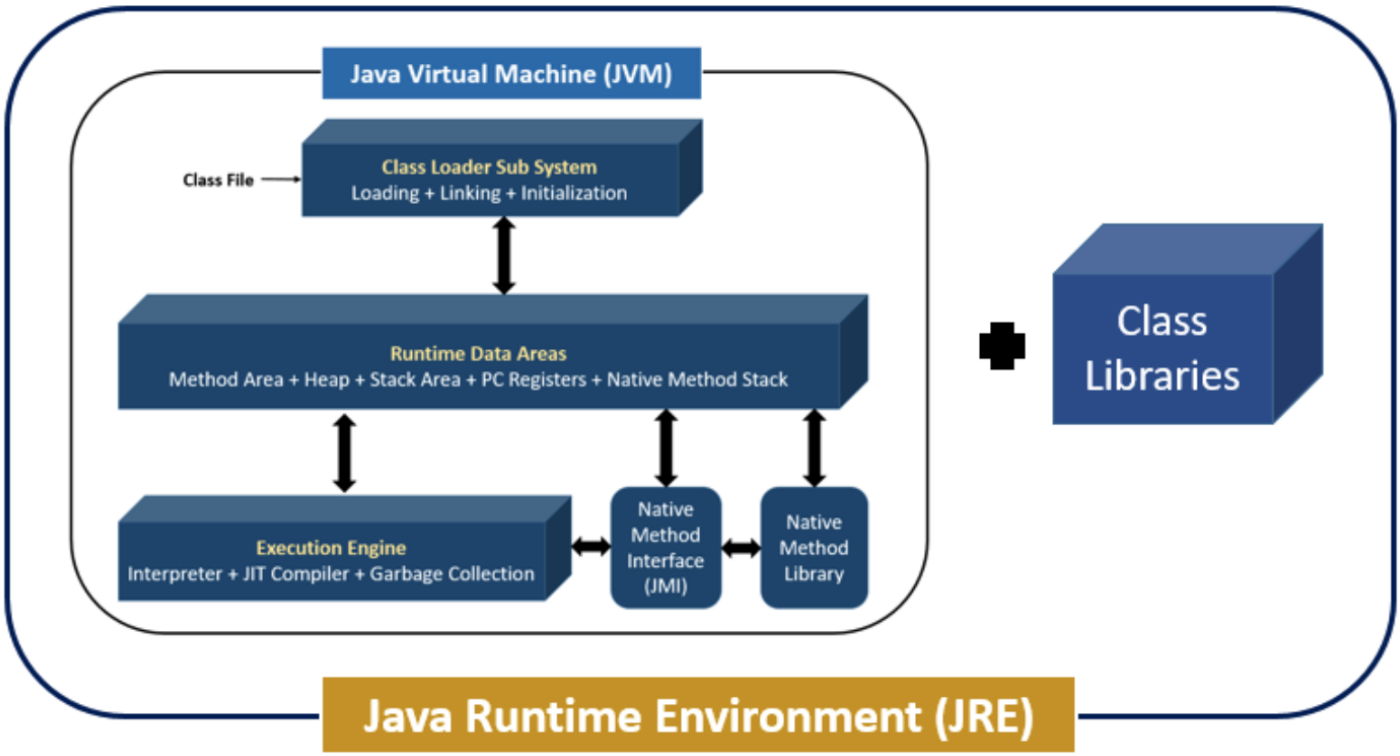
Features of JVM

- Java Virtual Machine is an abstract machine responsible for compiling and executing Java code.
- It has a class loader, runtime data area, execution engine, and libraries.
- JVM comes with JIT(Just-in-Time) compiler that converts Java source code into machine code.
- JVM provides basic java functions like memory management, security, garbage collection, etc.
- JVM is an integral part of JRE. Runs the program by utilizing JRE’ s libraries and files.
- It can execute the java program line by line. Therefore, it is also known as an interpreter.

JRE

= Java Runtime Environment

= JVM + Class Libraries (For Running Java Applications)

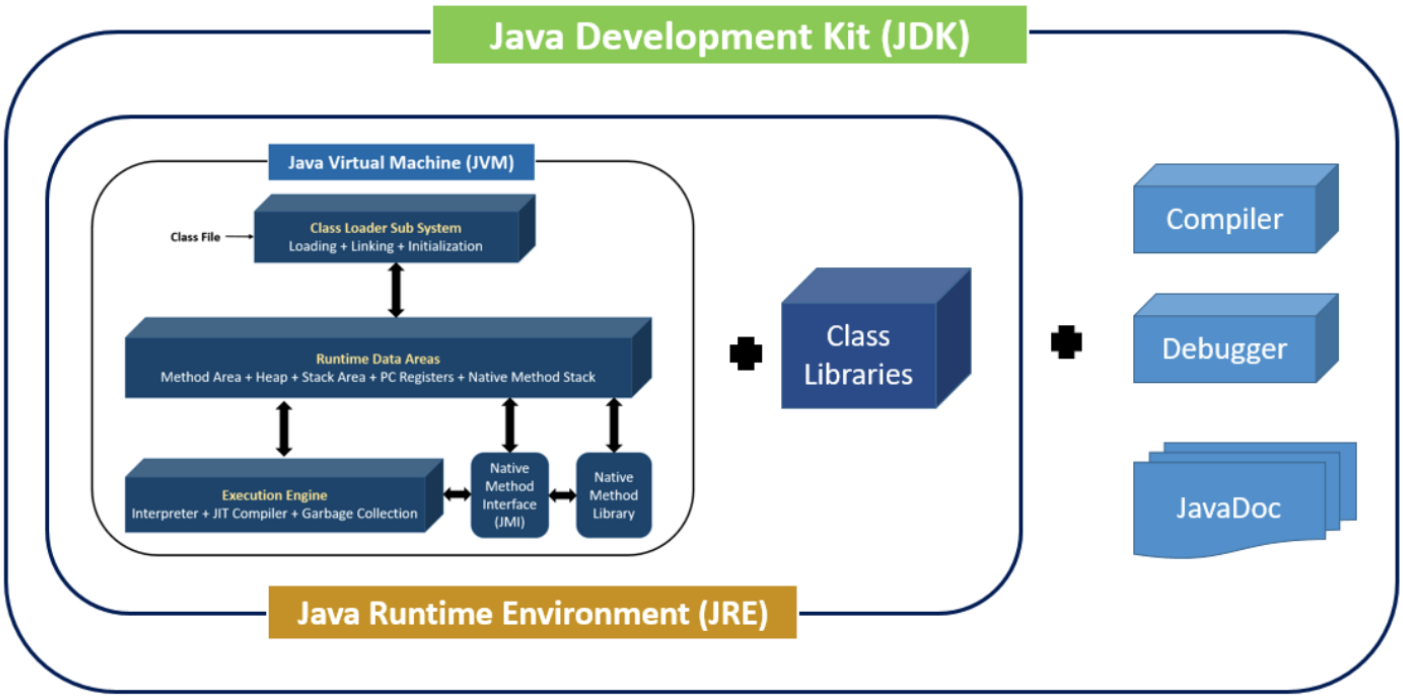


Features of JRE

- JRE consists of a set of tools to help the JVM run. In addition, it includes a few deployment tools such as Java Plug-in and Java Web Start.
- A User can efficiently run java code with JRE only. However, JRE doesn’ t allow writing the program.
- JRE appends various integration libraries like the JDBC (Java Database Connectivity), JNDI (Java Naming and Directory Interface), RMI (Remote Method Invocation), etc.

JDK

Development Tools + JRE (Java Runtime Environment)



Features of JDK

- JDK provides an environment for developing and executing Java source code.
- It includes all the functionalities of JRE and JVM.
- JDK helps developers to handle the exceptions using multiple extensions in a single catch block.
- It has various other development tools like the debugger, compiler, etc.
- It is platform-dependent. Any user can easily install JDK on Operating systems like Unix, Mac, Windows, etc.

Questions

- What are the differences between compiled and interpreted languages?
- Is Java a compiled or interpreted language?
- Why is Java considered a platform-independent language?
- What is the purpose of the JVM, JRE and JDK?
- How do we compile and execute a Java program?