
MAHARISHI INTERNATIONAL UNIVERSITY
Engaging the Managing Intelligence of Nature
Computer Science Department
CS390 Fundamental Programming Practices (FPP)
Prof. Paul Corazza and
Prof. Ankhtuya Ochirbat

Lecture 1:
Introduction to Java And the IntelliJ IDE

Pulling the Arrow Back to Hit the Target

Wholeness of the Lesson

Java is an object-oriented highly portable programming language that arose as an easy alternative to the once dominant, but error-prone, C++ language.

IntelliJ is a modern, powerful, and user-friendly IDE known for its intelligent code assistance and seamless support for Java and related technologies.

Working from deeper levels of intelligence allows one to accomplish more with fewer mistakes and less effort.

About Java

- ***Brief History.*** The Java language began as a language for programming electronic devices, though the original project was never completed. Its creator was **James Gosling**, of Sun Microsystems. The language was developed privately starting in 1991 and was made publicly available in 1994. In 2009, Oracle bought the rights to Java from Sun Microsystems.

- ***Java Is an OOP Language.*** Java is an object-oriented programming language. This means that Java programs are designed by defining software objects that interact with each other (mirroring the way things get done in the real world).

- In many enterprise environments, especially banking, insurance, and large-scale **enterprise systems**, Java remains a leading language.

The Java API Docs

- Oracle provides online documentation of all the Java library classes. Full documentation of each class in the Java libraries is provided. For Java 24, the link is

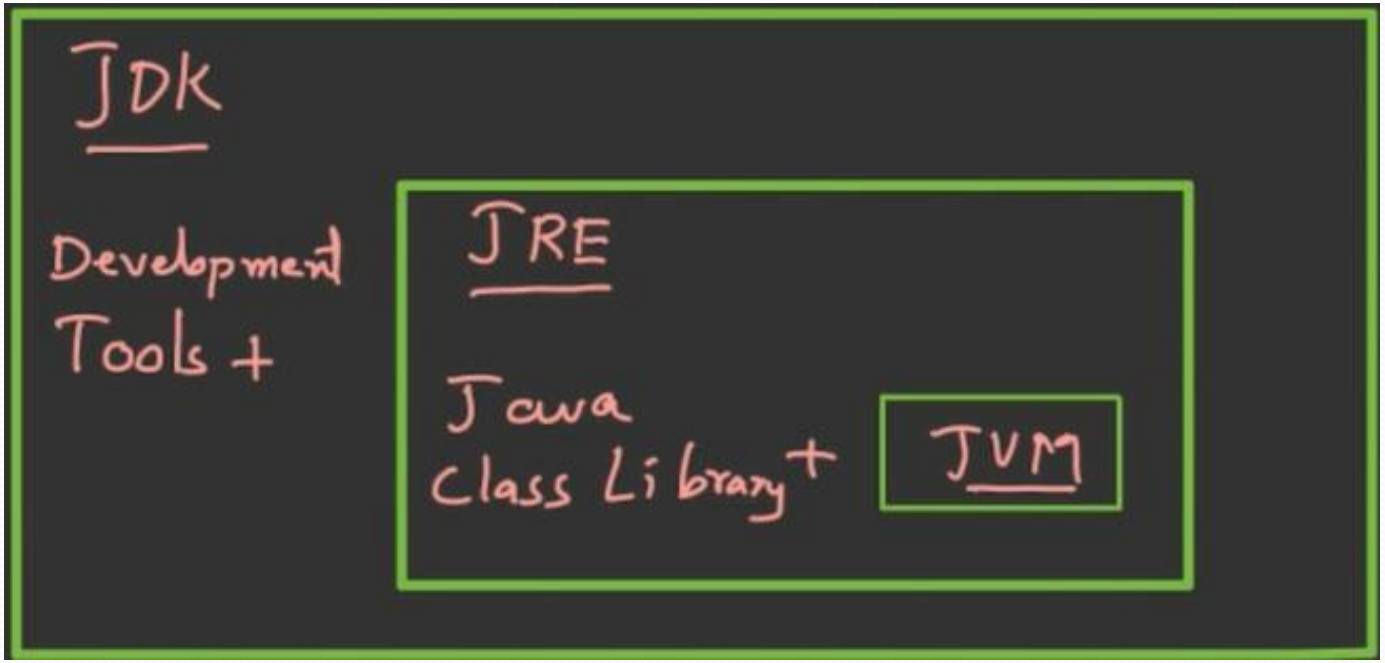
  https://docs.oracle.com/javase/specs/jls/se24/html/index.html

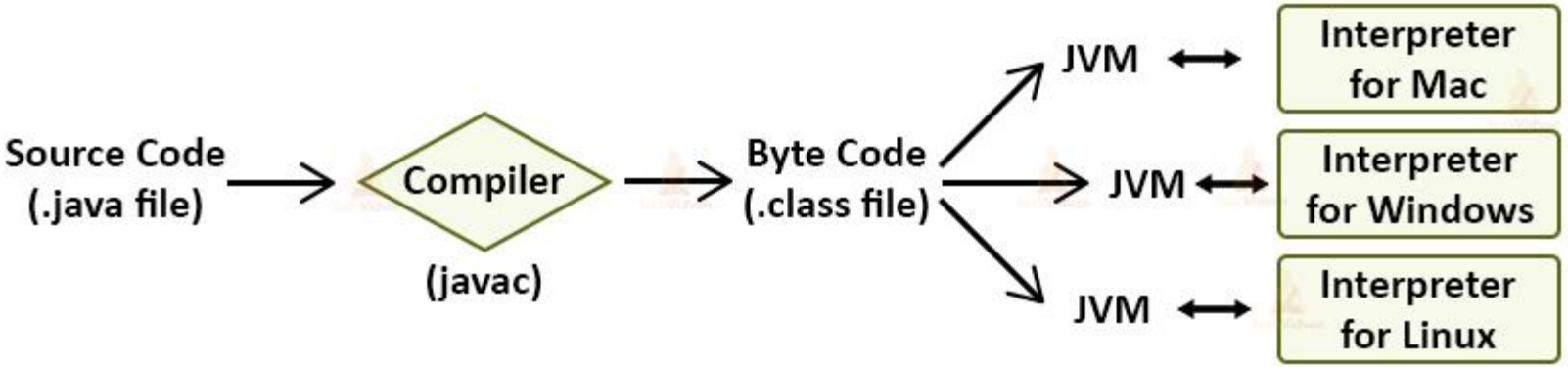
Viewing Java Library Source Code

1. Open any Java project in IntelliJ IDEA.
2. Inside any .java file, type the class name (e.g., String) and place your cursor on it.
3. Press **Ctrl + B** (or Cmd + B on macOS) Or right-click the class name → Go to > Declaration or Usages

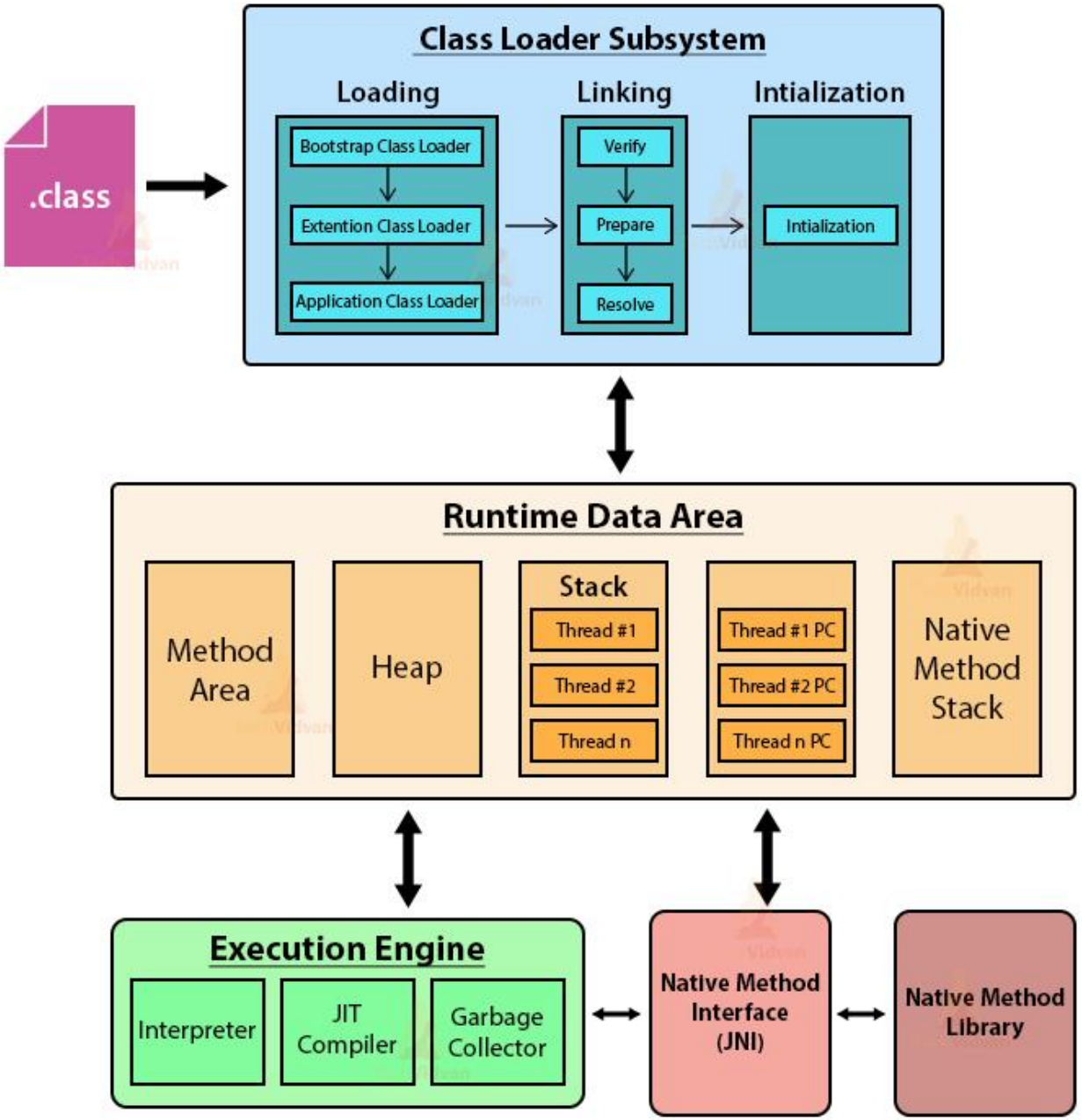
- This will take you to the source code of that class (not just the compiled .class file), if the src.zip is available.

JDK, JRE, and JVM

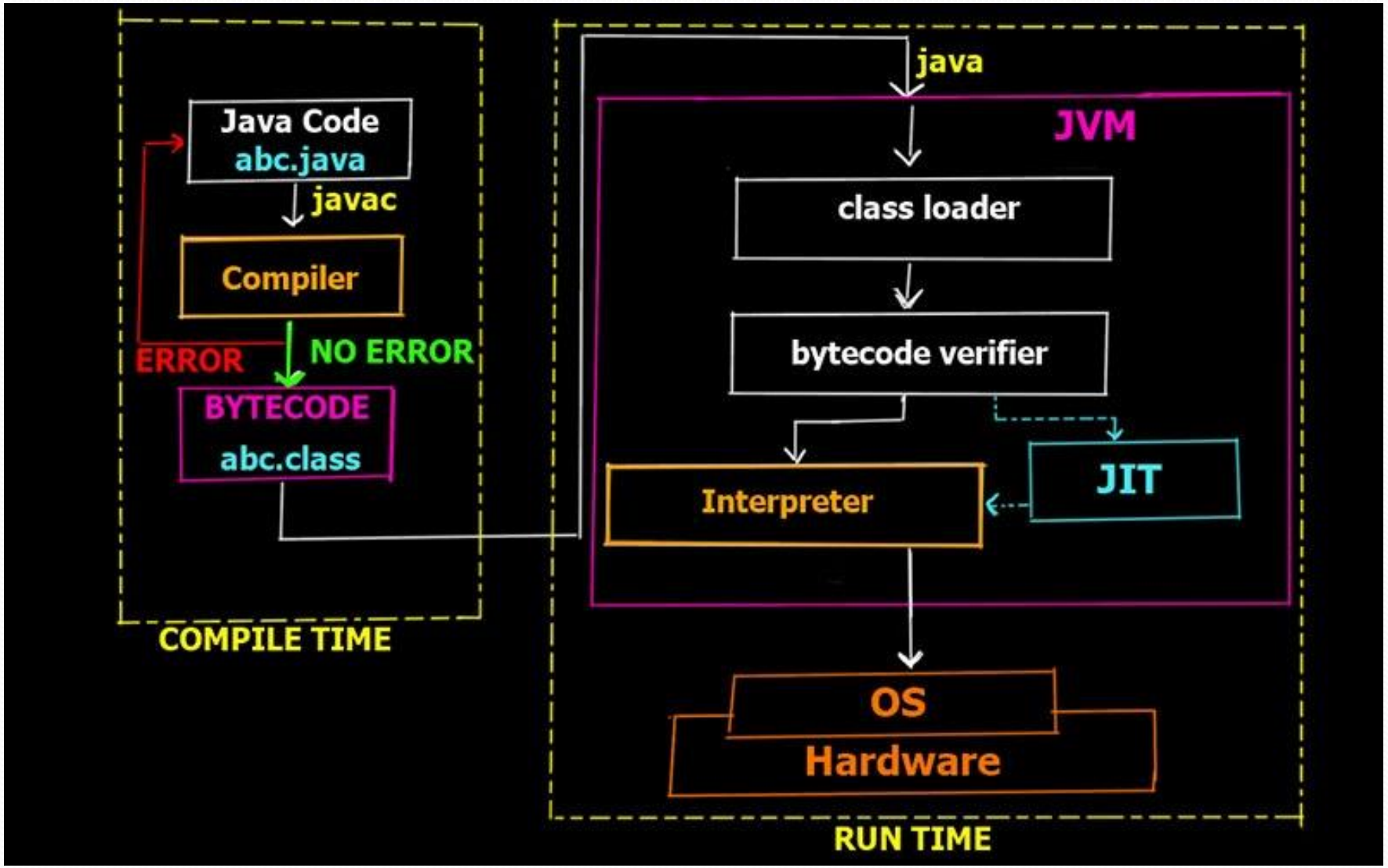




JVM Architecture



[Ref]


Java Code
abc.java
javac
Compiler
ERROR
NO ERROR
BYTECODE
abc.class
COMPILE TIME
java
JVM
class loader
bytecode verifier
Interpreter
JIT
OS
Hardware
RUN TIME

Integrated Development Environments (IDE)

- A good IDE supports compiling, running, and debugging code with tools that are integrated and typically easy to use. For a large Java project, an IDE is indispensable.
- Good choices of IDE are JetBrains' IntelliJ, Eclipse, Apache NetBeans, VS Code, Oracle JDeveloper so on.
- We will use **IntelliJ** in this course.

The intelliJ IDE

- *Getting started*. For this course, you will need one of the recent versions of the JDK and of IntelliJ IDEA (Community Edition is free and sufficient for most Java courses).

- Features of the intelliJ IDE
  - Project > Module > Package > Class
  - Setting preferences: File > Settings (Windows/Linux) or IntelliJ IDEA > Preferences (macOS)
  - Save / compile / run / debug: IntelliJ auto-saves and builds by default. Run and Debug using green arrows or Shift+F10 / Shift+F9
  - Auto-formatting: Code > Reformat Code or press Ctrl+Alt+L (Windows/Linux), Cmd+Option+L (macOS)

"Hello World"

- Create the simplest Java program. This involves creating a package and then defining a class inside that package.

```
package lesson1.hello;

public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

"Hello World" – the class file

When you create an IntelliJ project, IntelliJ creates a src/ folder (which will contain your source code) and an out/ folder (which will contain the compiled files – the .class files).

The .class file is an intermediate file consisting of *bytecode*. The bytecode is interpreted by the operating system into native code at runtime.

```
YourProject/
├── .idea/                  ← IntelliJ project config
├── out/
│   └── production/
│       └── YourProject/    ← Compiled .class files (bytecode)
├── src/
│   └── Main.java           ← Your source code
└── YourProject.iml         ← Module file
```

In-Class Exercise 1.1

- In this exercise you will run a Java application SampleClass, available in the InClassExercises project.
- The comments written in the class SampleClass explain many details about basic Java code. The goal is to study the code and run it to get a feeling for how it works and then try to use what you have learned to implement the requirements in a second Java class MyClass.

Connecting the Parts of Knowledge With the Wholeness of Knowledge

1. Using Java, highly functional applications can be built more quickly and with fewer mistakes than is typically possible using C or C++.
2. To optimize the use of Java's features, IDEs such as IntelliJ ease the work of the developer by handling in the background many routine tasks.

---

3. **Transcendental Consciousness:** To be successful, action must be based on the field of pure intelligence, which is located at the source of thought.

4. **Wholeness moving within itself**: In Unity Consciousness, the pure intelligence located in TC is found pervading all of creation, from gross to subtle.