Java Part 1: First Steps and Your First Program

Java is a programming language used for business systems and web services. We write it as plain text, and a program turns that text into instructions a computer can run. A free development kit and a free editor are all we need to practise, and every step here can be repeated on any computer. Four steps take us from an empty file to a working program: write, compile, run, and read and fix. In this part, Samir, a logistics analyst at a freight company, uses them to print his first shipment message, and each step answers one practical question.

1. Write

A Java program starts as a plain text file with the extension dot java. The code lives inside a class, and the class name must match the file name when the class is public. Every program begins at a special method called main, and Java starts running there. Inside it, the println statement from System out prints a line of text to the screen. Java is case sensitive, so System and system are different words, and most statements end with a semicolon. Comments, which start with two slashes, are ignored by the compiler and help human readers. The question for this step is: what is the smallest complete program that does something visible?

Example: A trainee at a bank writes a file named Welcome with a class called Welcome. Inside the main method she places one println statement with the words Welcome aboard. The file name and the class name match exactly, even in the capital letter. The whole program is only five lines long.

At work: Samir opens a free editor, such as Eclipse or Visual Studio Code, and creates a file named ShipmentLabel. He writes a class with the same name, a main method and one line that prints the words Shipment ready: `System.out.println("Shipment ready");`. The line sends the text between the quotation marks to the screen. He asks himself the step question, and the answer is these few lines.

2. Compile

A computer cannot run our text directly, so a tool called the compiler translates it first. The compiler is named javac, and it comes with the Java Development Kit, the JDK, which also contains the other tools we need. It reads the source file, checks the rules and the types, and writes a new file with the extension dot class. That file holds bytecode, a compact set of instructions that is not tied to one operating system. If the code breaks a rule, the compiler stops and reports the file, the line and the problem, so mistakes are caught before anything runs. A warning is different from an error: the compiler still produces the class file, but it advises us to look more closely. The question for this step is: does the compiler accept our code?

Example: A finance team writes a small program to add up expenses and compiles it. The compiler complains that a word is missing a capital letter on line seven. Nothing has run yet, and no data has been touched. They fix the letter and compile again, and the dot class file appears.

At work: Samir opens a terminal in the folder of his file and types javac followed by the file name. For a moment nothing is printed, which is good news, because silence means success. A new file named ShipmentLabel dot class now sits next to his source. He answers the step question: yes, the compiler accepts the code.

3. Run

To run the program we use the java command with the class name, and it starts the Java Virtual Machine, the JVM. The JVM loads the bytecode, checks it and executes it on the actual computer. Because the bytecode is the same everywhere, the same class file can run on Windows, macOS or Linux, as long as a Java runtime is installed. This is the idea behind the famous portability of Java. Recent versions of Java can also run a single

This document is a property of Asaseeyat Academy - © Copy Right Protected.

source file directly with one command, which is handy for small experiments, but compiling first shows what really happens. A common first problem is that the java command is not found, which usually means that the system does not yet know where the JDK tools are installed. The question for this step is: which class do we start, and where does the JVM look for it?

Example: A software vendor ships one compiled program to customers who use different operating systems. Each customer installs a Java runtime and runs the same class file with the java command. The JVM on each machine loads the same class file and does the same work. The vendor does not rebuild the program for every system. There is only one build to maintain and explain.

At work: Samir types java followed by ShipmentLabel, without any extension. The JVM finds the class, starts at main and prints the line Shipment ready on the screen. He gets the answer to the step question: the class name, in the folder where the compiled file lives. Then he shows the screen to his manager, and the manager asks for a message with a customer name next.

4. Read the Messages and Fix

Beginners meet errors every day, and reading them calmly is a core skill. A compile error appears before the program runs and names the file and the line. A runtime error appears while the program is running, and Java prints a stack trace, which lists the chain of calls that led to the failure. We read the first message first, because one early mistake often causes many later complaints. Typical first mistakes are a missing semicolon, a missing closing brace, a wrong capital letter, and a file name that does not match the class. Saying the message aloud in plain words, such as on line four this name is unknown, turns a scary block of red text into a simple to-do item. The question for this step is: what does the message say, and on which line?

Example: A support engineer forgets a closing quotation mark and sees several messages at once, each on a different line. She does not try to fix them all. She reads the first message, fixes only that one and compiles again. Most of the other messages disappear, because they came from the same missing mark. The one early mistake was the cause of the whole red list.

At work: Samir mistypes println as printLn, with a capital L, and the compiler says it cannot find the symbol on line four. He reads the line number, spots the capital letter and corrects it. He compiles again and runs the program successfully. Then he repeats the loop with a second line that prints the destination city.

Conclusion

To recap, a Java program goes through four steps: write, compile, run, and read the messages and fix. We write a class with a main method in a file whose name matches the class. The compiler javac turns it into bytecode, and the JVM runs that bytecode on any system with a Java runtime. Errors are not failures but directions, so we read the first message and the line number. Samir now has a working first program, and next time he will give it values to remember and calculate with. A good habit is to run the program after every small change, so that the last change is always the first suspect.

Sources

- Oracle: The Java Tutorials, Getting Started (the Hello World application, compiling and running) – https://docs.oracle.com/javase/tutorial/getStarted/cupojava/index.html
- Oracle: The Java Tutorials, Getting Started (the Java technology, what it is and how it runs) – https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html
- dev.java: Learn Java (official learning site for the Java community) – https://dev.java/learn/
- Java is a trademark of Oracle and/or its affiliates. This video is independent training material and is not affiliated with Oracle.
- OpenJDK: JEP 330, Launch Single-File Source-Code Programs (Java 11) – https://openjdk.org/jeps/330

This document is a property of Asaseeyat Academy - © Copy Right Protected.