
JAVA
www.schuilenburg.
tel.: +32(0)52/49 90

Introduction to Java: A Beginner's Guide

After 28 years, Java is still a popular choice when developing applications.

In this tutorial, we'll explore the fundamentals of Java, its history, key features, and its role in the world of software development.

Java belongs to object-oriented programming languages. Object-oriented programming (or OOP for short) is a paradigm that organizes software around classes and objects. Unlike the other concepts, like procedural or functional programming, in OOP, classes represent the main building blocks. Moreover, we write our code inside some classes.

Additionally, we can make every class "alive" by instantiating it (just a fancy word for creating an object). Objects communicate with each other, sharing information, changing states, and performing actions.

Usually, when creating classes, we bring them close to real life and the way we think.

For instance, let us imagine we want to create a calculator. Instead of a physical calculator, we would have the Calculator class. Additionally, each action (addition, subtraction, multiplication, division, etc.) would be represented by methods.

Now, let us take a brief overview of the Java history.

1. Historia Magistra Vitae est

Once upon a time, there was a company called Sun Microsystems. In 1991 the company decided to be different than all the other companies. Thus, they started a project named The Green Project. And no, it was not about the environment and saving the Mother Earth. The main goal of this project was to create a programming language that would be able to run on different operating systems.

James Gosling (let us call him the Father), worked in his office on this new super magic cool programming language. Now, back then he had a problem – he did not know how to name the language. And then, one sunny day, it came up to him. He looked at his window and saw the big oak. Therefore, he decided to name the language Oak.

Nevertheless, the marketing team from Sun noticed there was already an IT company named Oak. So, they were back at square one. However, they decided to discuss about new name over a nice cup of coffee. On this coffee, they agreed the name of the new language would be Java.

The first Java version was published in 1995.

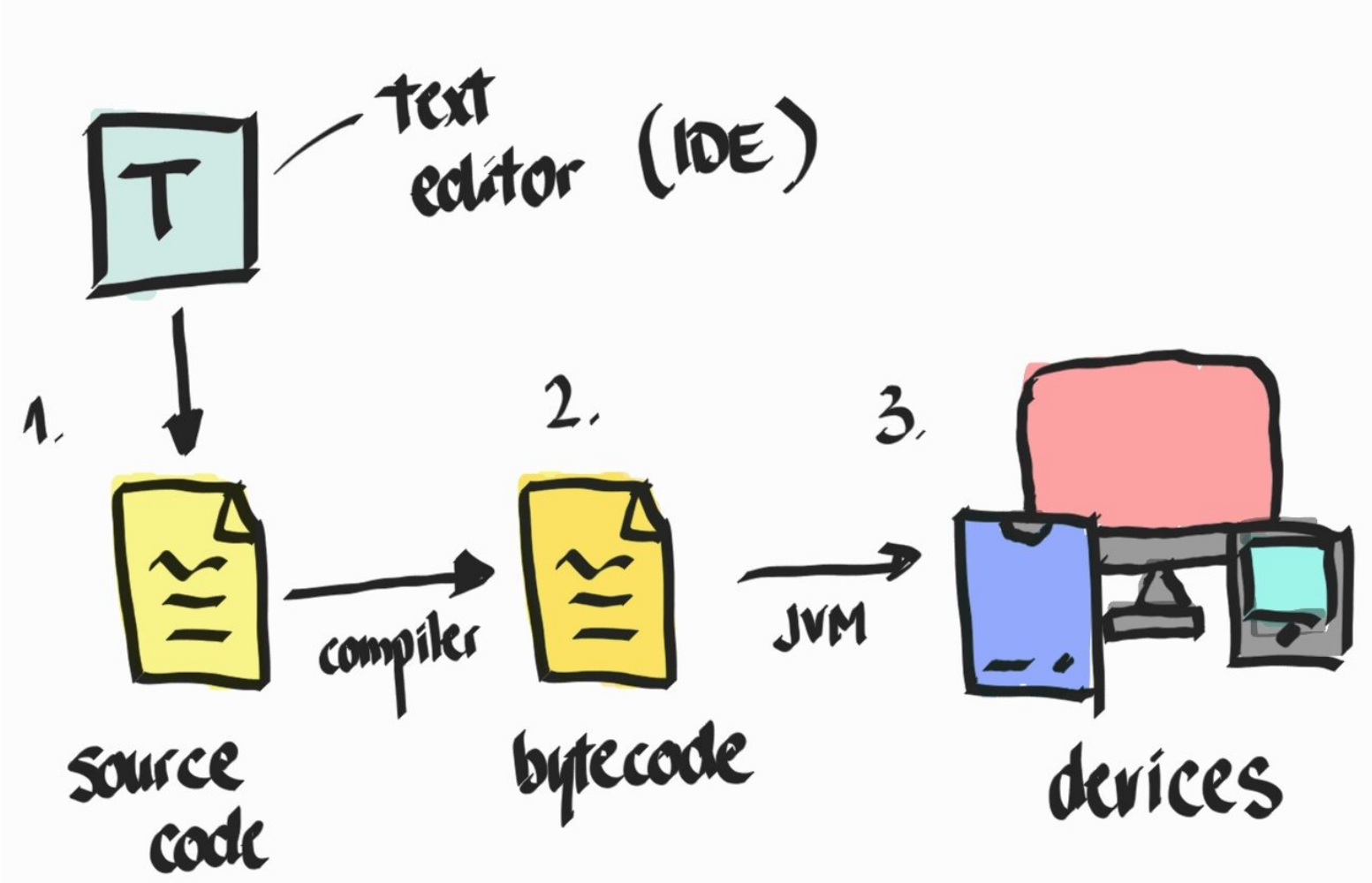
After 15 years, Oracle decided to purchase Sun Microsystems will all its products (Java included). And here we are, more than 10 years later. At the time this book was written, the newest Java version was Java 21. Oracle tries to release a new Java version once a year.

2. How Java Work?

We, developers, represent only one piece of a big puzzle. Our job is to write commands that will perform desired actions. And this is where our job finishes. After, Java says "I am the captain now." and takes control. Java has a best (although very strict) friend called the compiler. The compiler translates our commands into the commands Java Virtual Machine (JVM) understands. JVM is responsible for running

our application.

The simplified process looks like the following:



1. Firstly, we need to create a file to put our code. This file is known as a source code and uses the .java extension. In this file, we write a code using Java programming language syntax. We can write code inside a text editor or use something called IDE (Integrated Development Environment). This is where our job ends and Java takes control instead of us.
2. As mentioned earlier, Java has a buddy named compiler. You can think of the compiler as a Grammar Nazi. The main task of the compiler is to check your Java syntax and scream if something is not correct. Furthermore, when you write code, the compiler will not check if the code makes sense (logically). Thus, it will only check if you made some syntax mistakes. The compiler will transform our code into the code Java Virtual Machine can understand. This code is called the bytecode. He will create a new file with a .class extension.
3. Java Virtual Machine, or JVM in short, is another Java friend. The JVM has one significant job to do – run our application.

3. Key Features of Java

Next, let us look at the core features that make Java a perfect choice when developing applications.

1. Platform Independence

As already mentioned, Java does not depend on the operating system. This platform independence is possible because of the JVM. Our code is translated into bytecode, which can run on any system with a compatible JVM. Therefore, it simplifies cross-platform development.

> Write once, run everywhere.

2. Object-Oriented

Java is a pure object-oriented language, emphasizing the use of classes and objects. It encourages modular and reusable code through inheritance, encapsulation, and polymorphism.

3. Garbage Collection

Java manages its own memory. When objects are created, they take up a certain amount of space in memory. That space should be deallocated once an object is no longer needed.

Java uses a technique called Garbage Collection. It will remove unused objects from memory. It will help prevent memory leaks.

Languages such as C do not use Garbage Collection. We need to deallocate a memory by ourselves.

4. Pointers No More

Unlike the C programming language, Java does not use pointers. When it comes to objects, Java becomes very protective and, just like any mother, does not allow anywhere near them. To put it differently, we cannot access objects directly.

The only way to access them is by using references.

5. Multithreading

Java supports multithreading, allowing developers to create concurrent applications that can take advantage of modern multi-core processors.

4. Getting Started with Java

Finally, to start programming in Java, you'll need to set up your environment. Let us go through the steps.

4.1. Install Java

Firstly, you need to download and install the Java Development Kit (or JDK) from the official Oracle website. Alternatively, you can use OpenJDK which is an open-source.

Once you finish downloading JDK, you will need to create the JAVA_HOME environment variable. It

should contain the path to your Java installation directory.

For instance, on the Windows operating system, it would be something like the following:

C:\Program Files\Java\jdk21

4.2. Choose an IDE

Secondly, you need an application where you will write your code. You can use a text editor like Notepad, or install Integrated Development Environments (IDEs) such as IntelliJ IDEA, Eclipse, or Visual Studio Code.

IDE offers powerful tools for Java development. If you ask me, my weapon of choice would be IntelliJ.

4.3. Your First Program

Lastly, let us write a simple “Hello World” program.

Firstly, we need to create a file named Demo.java. Next, let us put the code that prints “Hello World” in the console:

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

Now, we can use the javac compiler to compile Java code into bytecode:

javac Demo.java

If we do not get any error, we can run our code with java command:

java Demo

And this is it. It should print “Hello World” into the console.

5. Key Takeaways

In this tutorial, we got an introduction to Java programming language.

Java has a large and active developer community, which means there are ample resources, libraries, and frameworks available for various applications, such as web development (e.g., Spring Framework), mobile app development (e.g., Android), and more.

Read More