Set Up Your JDK and IDE

Two installs make you able to run Java: OpenJDK 17 and an IDE. Then create the project you will type -- not paste -- your first automated check into.

Automation Foundations · ashishautomationlabs.store

Two installations, and you are ready to write Java the way automation teams actually work. If a PATH issue eats an evening later, the fix is usually two lines -- do not lose the day to it.

1. Install the JDK

Download an **OpenJDK 17** build. Eclipse Temurin (from adoptium.net) is a widely used free choice; Oracle's builds also work. Run the installer and accept the defaults. On Windows, tick any option like "Set JAVA_HOME" or "Add to PATH."

Verify it worked. Open a terminal (Command Prompt on Windows, Terminal on Mac) and type:

```
java -version
```

If a version number prints, your machine can run Java.

2. Install an IDE

You _can_ write Java in Notepad and compile by hand in a terminal. You can also execute a 500-case regression with a printed checklist and a stopwatch. We use professional tools.

An **IDE** -- Integrated Development Environment -- is your toolbench: editor, compiler connection, error highlighting, and later a debugger, in one window. **Eclipse IDE for Java Developers** is free, and automation teams everywhere use it. Its most useful feature for you right now: compiler errors show as red underlines while you type -- a spell-checker for code.

Download it from eclipse.org and install with the defaults. If your team uses IntelliJ IDEA instead, that is completely fine -- everything here works the same way; only menu names differ.

Create your first project

In Eclipse: **File -> New -> Java Project** -> name it `java-hero` -> make sure your installed JDK 17 is selected -> **Finish**.

If Eclipse offers to create a `module-info.java` file, click **Don't Create** -- you do not need it for this series.

Eclipse gives you a folder structure. The only part that matters today is the `src` folder. Your source code lives there.

The habit that separates fluent from familiar

Type every program yourself. Do not copy-paste.

Copy-paste teaches your clipboard; typing teaches your hands. You will make small mistakes, and the compiler will report them. Reading those reports is the real course. The typing mistakes are not in the way of the learning. They are the learning.

You can prove Java runs on this machine, and you have a project ready for code. The next lesson is not Hello World. Your first program will be a real -- if small -- automated check: compare expected against actual, then announce PASS or FAIL.