



西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

Object Oriented Programming

Chapter 2 The Java Programming Environment

Dr. Helei Cui

4 Mar 2025

Slides partially adapted from lecture notes by Cay Horstmann

Before setting up your Java development environment

Java jargon

Name	Abbreviation	Explanation
Java Virtual Machine	JVM	Interprets the bytecode into code that will run on actual computer hardware
Java Development Kit	JDK	The software for programmers who want to write Java programs
Java Runtime Environment	JRE	The software for consumers who want to run Java programs
Standard Edition	SE	The Java platform for use on desktops and simple server applications
Enterprise Edition	EE	The Java platform for complex server applications
Micro Edition	ME	The Java platform for use on cell phones and other small devices
Update	u	Oracle's term for a bug fix release

E.g., JDK 8u281 - Java™ SE Development Kit 8, Update 281

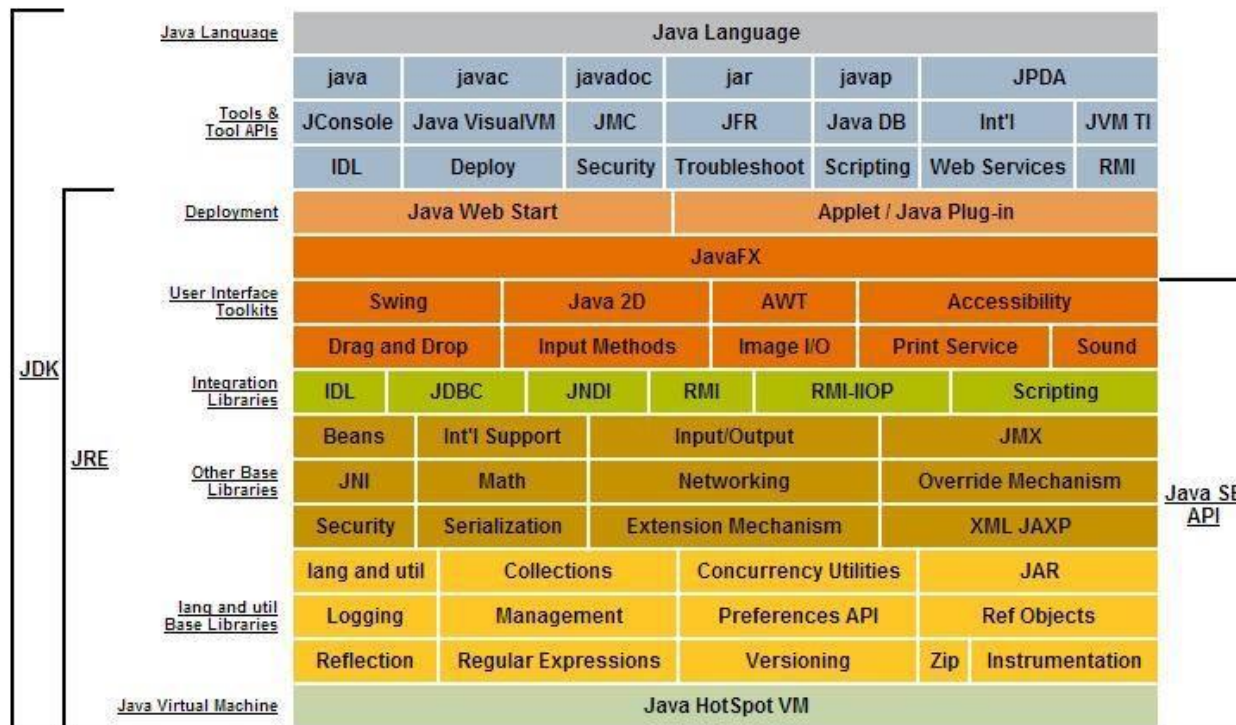
Quick question 1

Which one is the best for you to learn Java?

- A. JVM
- B. JDK
- C. JRE

What is JDK?

- JDK is a software development environment used for making applets and Java applications.
 - Java developers can use it on Windows, macOS, Solaris, and Linux. JDK helps them to code and run Java programs.
 - It is possible to install different JDK versions on the same computer.



What is JRE?

- JRE is a piece of a software which is designed to run other software.
 - **It contains the class libraries, loader class, and JVM.**
 - In simple terms, if you want to run Java program you need JRE.
 - If you are not a programmer, you don't need to install JDK, but just JRE to run Java programs.
- All JDK versions comes bundled with JRE.



What is JVM?

- JVM is an engine that provides a runtime environment to drive the Java Code or applications.
 - It converts Java bytecode into machine language.
- JVM is a part of JRE.
 - It cannot be separately downloaded and installed.
 - To install JVM, you need to install JRE.

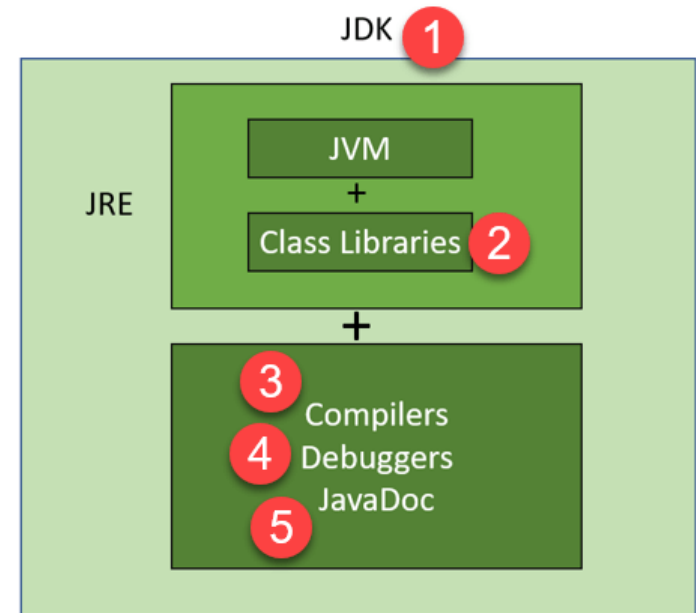


Key differences

JDK	JRE	JVM
a software development kit	a software bundle that allows Java program to run	an environment for executing bytecode
platform dependent	platform dependent	platform-independent
contains tools for developing, debugging, and monitoring java code	contains class libraries and other supporting files that JVM requires to execute the program	software development tools are not included in JVM
enables developers to create Java programs that can be executed and run by the JRE and JVM	is the part of Java that creates the JVM	is the Java platform component that executes source code
superset of JRE	subset of JDK	subset of JRE
JDK comes with the installer	JRE only contain environment to execute source code	JVM bundled in both software JDK and JRE

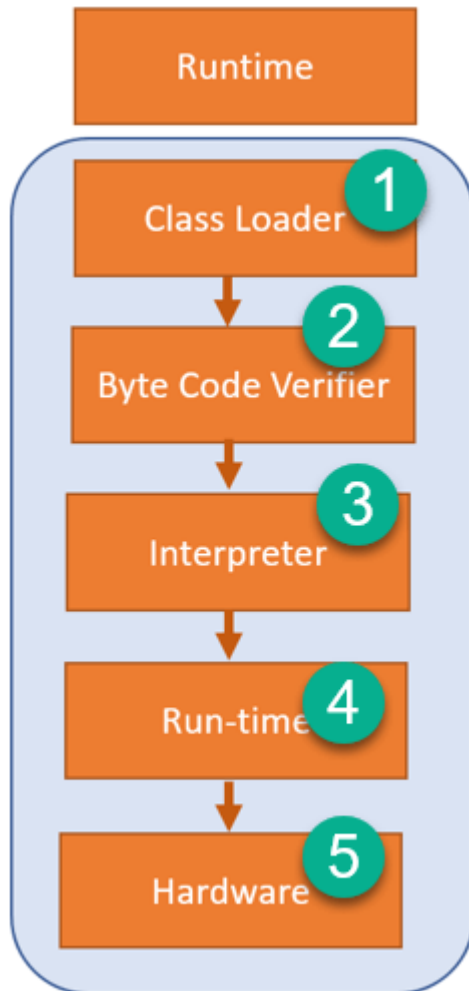
How JDK functions?

- ① **JDK and JRE:** The JDK enables programmers to create core Java programs that can be run by the JRE, which included JVM and class libraries.
- ② **Class Libraries:** It is a group of dynamically loadable libraries that Java program can call at run time.
- ③ **Compilers:** It is a Java program that accepts **text file** of developers and compiles into Java **class file**.
 - It is the common form of output given by compiler, which contains Java byte code.
 - In Java, the primary compiler is **Javac**.
- ④ **Debuggers:** It is a Java program that lets developers test and debug Java programs.
- ⑤ **JavaDoc:** It is documentation made by Sun Microsystems for the Java. It can be used generating API documentation in HTML file from the source program.



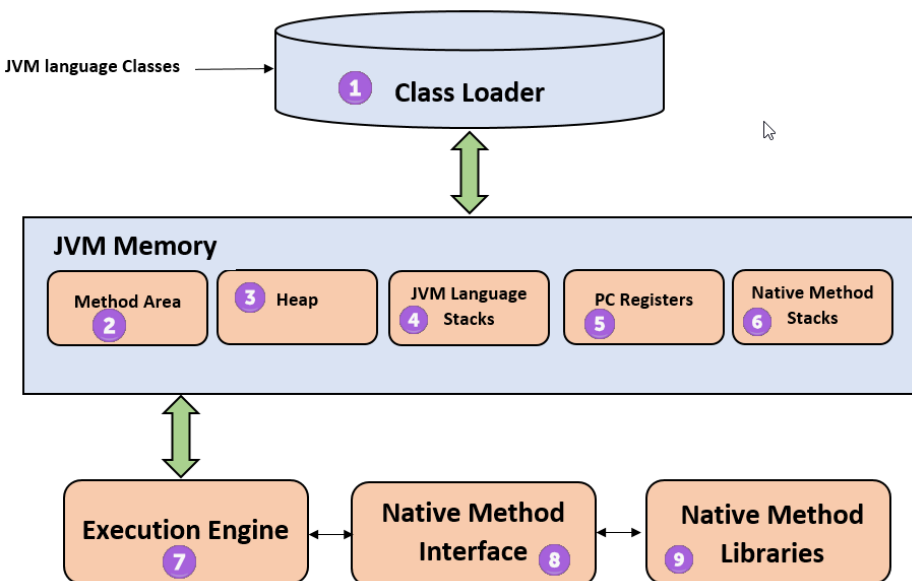
<https://www.guru99.com/difference-between-jdk-jre-jvm.html>

How JRE functions?



- ① **Class loaders:** It loads various classes that are necessary for running a Java program.
 - JVM uses three class loaders called the bootstrap class loader, extensions class loader, and system class loader.
- ② **Byte code verifier:** It verifies the bytecode so that the code doesn't disturb the interpreter.
- ③ **Interpreter:** Once the classes get loaded, and the code is verified, the interpreter reads the code line by line.
- ④ **Run-time:** It is a system used mainly in programming to describe time period during which a particular program is running.
- ⑤ **Hardware:** Once you compile Java native code, it runs on a specific hardware platform.

How JVM functions?



<https://www.guru99.com/java-virtual-machine-jvm.html>

- ④ **JVM language Stacks:** They store local variables, and partial results.
- ⑤ **PC Registers:** They store the address of JVM instruction which is currently executing.
- ⑥ **Native Method Stacks:** They hold the instruction of native code depends on the native library.
- ⑦ **Execution Engine:** It is a type of software used to test hardware, software, or complete systems.
- ⑧ **Native Method Interface:** It is a programming framework. It allows Java code, which is running in a JVM to call by libraries and native applications.
- ⑨ **Native Method Libraries:** is a collection of the Native Libraries (C, C++), which are needed by the Execution Engine.

- ① **Class Loader:** It is a subsystem used for loading class files.
- ② **Method Area:** It stores class structures like metadata, the constant runtime pool, and the code for methods.
- ③ **Heap:** All the **Objects**, their related instance variables, and arrays are stored in the heap.

How JVM works? (2 min)



<https://www.youtube.com/watch?v=G1ubVOI9IBw>

Installing the Java Development Kit

Download JDK

- Download Oracle JDK
 - <https://www.oracle.com/java/technologies/downloads/>

Java 23, Java 21, and earlier versions available now

JDK 23 is the latest release of the Java SE Platform.

[Learn about Java SE Subscription](#)

JDK 21 is the latest *Long-Term Support (LTS)* release of the Java SE Platform.

Earlier JDK versions are available below.

JDK 23 **JDK 21** GraalVM for JDK 23 GraalVM for JDK 21

Java SE Development Kit 21.0.6 downloads

JDK 21 binaries are free to use in production and free to redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions \(NFTC\)](#).

JDK 21 will receive updates under the NFTC, until September 2026, a year after the release of the next LTS. Subsequent JDK 21 updates will be licensed under the [Java SE OTN License \(OTN\)](#) and production use beyond the [limited free grants](#) of the OTN license will [require a fee](#).

Linux **macOS** **Windows**

Product/file description	File size	Download
x64 Compressed Archive	185.92 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip (sha256)
x64 Installer	164.31 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe (sha256)
x64 MSI Installer	163.06 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi (sha256)

Oracle JDK vs OpenJDK

- **The biggest difference is the **license**.**
 - OpenJDK is **completely open-source Java** with a GNU General Public License.
 - Oracle JDK requires **a commercial license** under the Oracle Binary Code License Agreement.
- **Are they the same?**
 - In brief, OpenJDK has **the same code** as Oracle JDK, depending on what provider you're using.
- **Performance?**
 - Historically, Oracle JDK has had better performance than OpenJDK. However, the performance of OpenJDK is growing.
- **Can I use OpenJDK for commercials?**
 - You can use OpenJDK for commercial software projects.

Choose a correct version

- For this course, you can select JDK 21.0.6.

Java 23, Java 21, and earlier versions available now

JDK 23 is the latest release of the Java SE Platform.

[Learn about Java SE Subscription](#)

JDK 21 is the latest *Long-Term Support (LTS)* release of the Java SE Platform.

Earlier JDK versions are available below.

JDK 23 **JDK 21** GraalVM for JDK 23 GraalVM for JDK 21

Java SE Development Kit 21.0.6 downloads

JDK 21 binaries are free to use in production and free to redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions \(NFTC\)](#).

JDK 21 will receive updates under the NFTC, until September 2026, a year after the release of the next LTS. Subsequent JDK 21 updates will be licensed under the [Java SE OTN License \(OTN\)](#) and production use beyond the [limited free grants](#) of the OTN license will [require a fee](#).

Linux **macOS** **Windows**

Product/file description	File size	Download
x64 Compressed Archive	185.92 MB	https://download.oracle.com/java/21/latest/jdk-21-windows-x64.zip
x64 Installer	164.31 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe (sha256)
x64 MSI Installer	163.06 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi (sha256)

Windows
64-bit OS

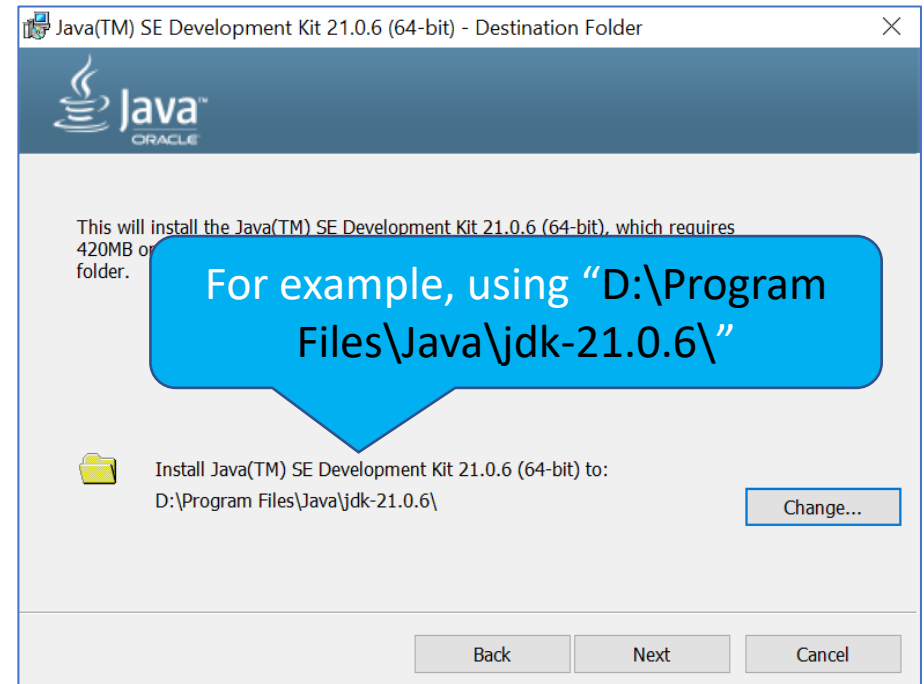
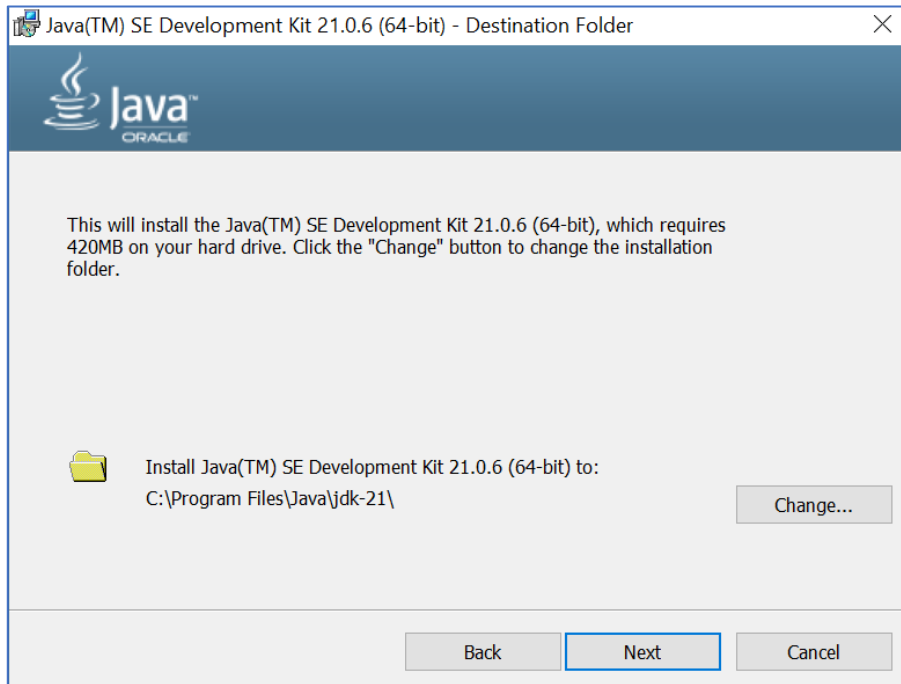
Install JDK

- Double click JDK package you just downloaded and start installing.



Choose installation folder

- You can change the directory during the installation.



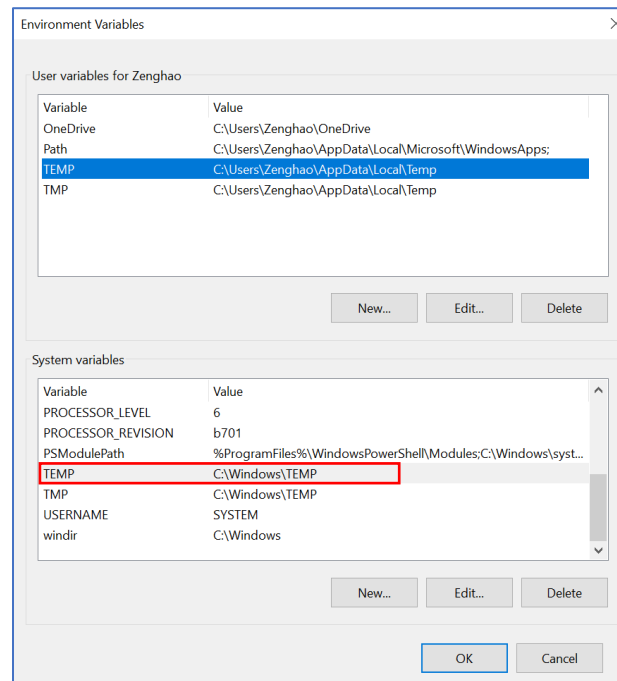
Finish installation

- Almost done!



What is Environment Variable?

- An environment variable is a **dynamic-named value** that can affect the way running processes will behave on a computer.
 - They are part of the environment in which a process runs.
 - *For example, a running process can query the value of the **TEMP** environment variable to discover a suitable location to store temporary files.*



Java Environment Variables

- **JAVA_HOME**

- This means the **JDK installation root path**, e.g., “D:\Program Files\Java\jdk-21.0.6”.
- It is used by the launcher for finding the JDK/JRE to use. Programs like **Tomcat** needs it to be set in order to execute error-free.



- **PATH**

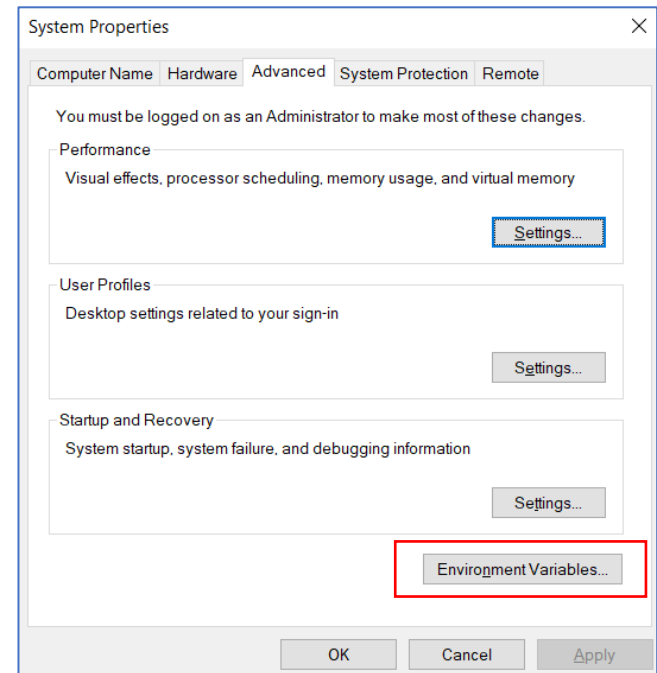
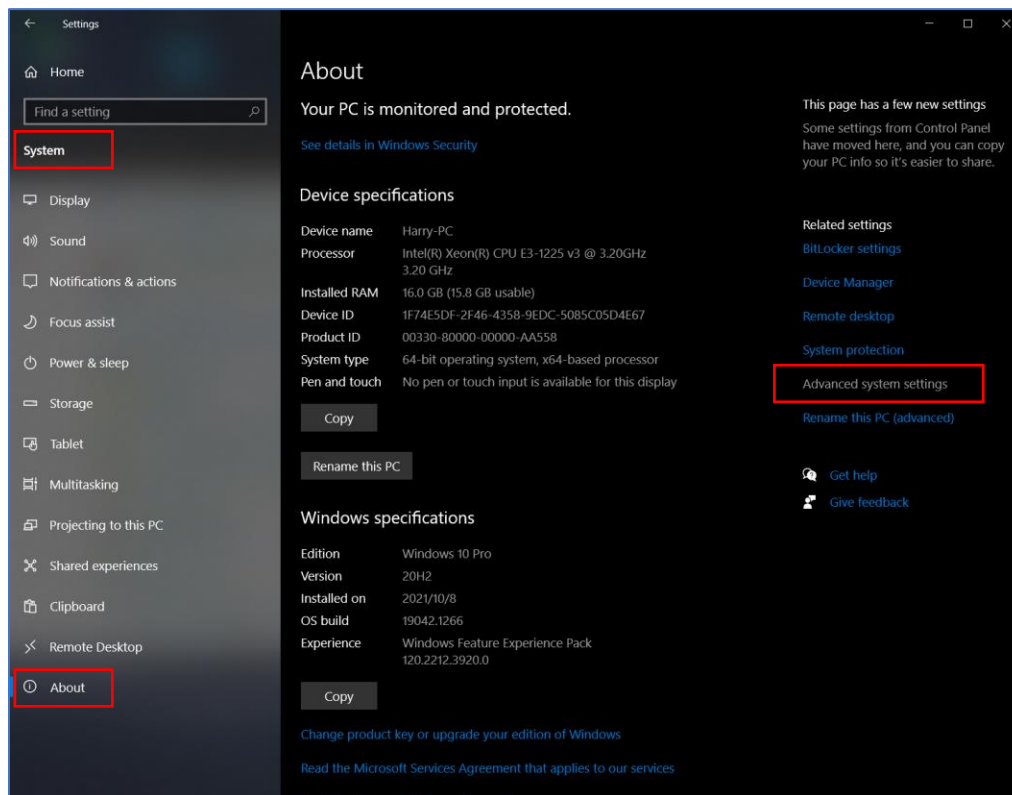
- If you want to be able to run the executables (**javac**, **java**, **javadoc**, etc.) from any directory without having to type the **FULL PATH** of the command, **PATH** needs to be set.
- If you don't set **PATH** variable, you need to specify the full path to the executable every time you run it, e.g., *D:\Program Files\Java\jdk-21.0.6\bin\javac HelloWorld.java*.

- **CLASSPATH**

- This is the path where to find the java class when your application executes.
- “It is not a good idea to set this variable globally, but some poorly written software installers in Windows do just that.”

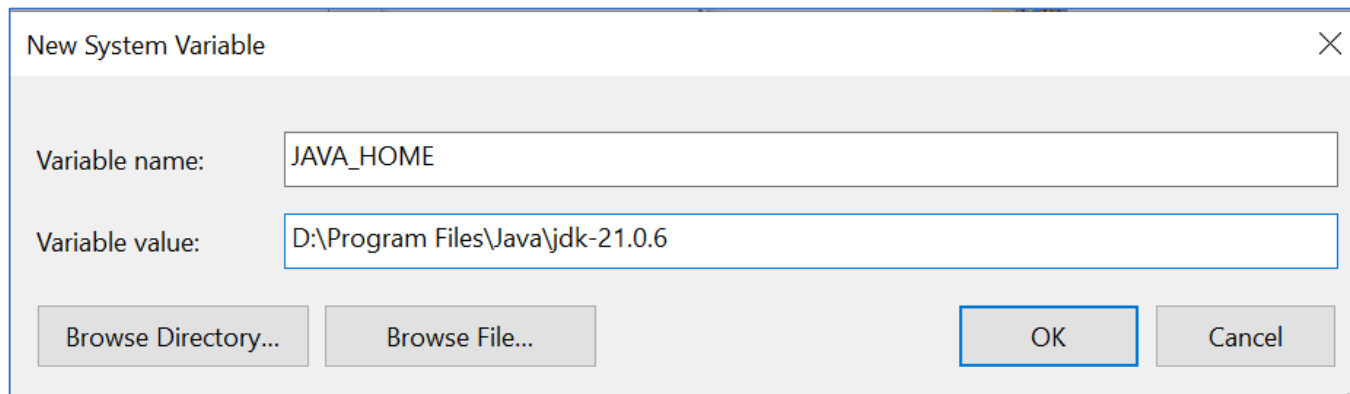
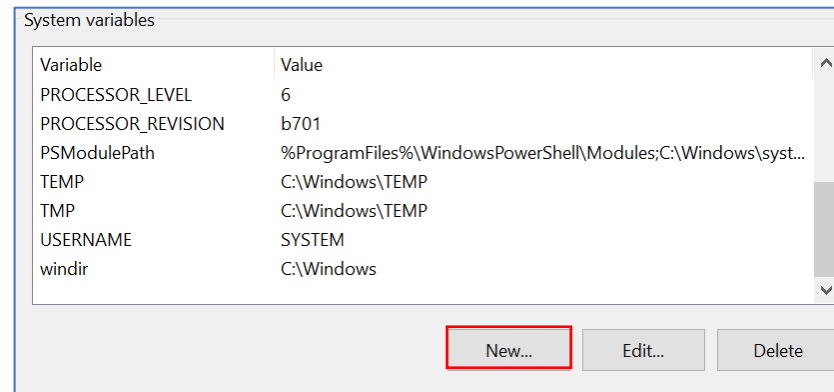
Find “Environment Variables”

- It’s a little tricky to find **Path** variable. First search “Settings” in Windows **Start** and then click “System” – “About” - “Advanced system settings”.



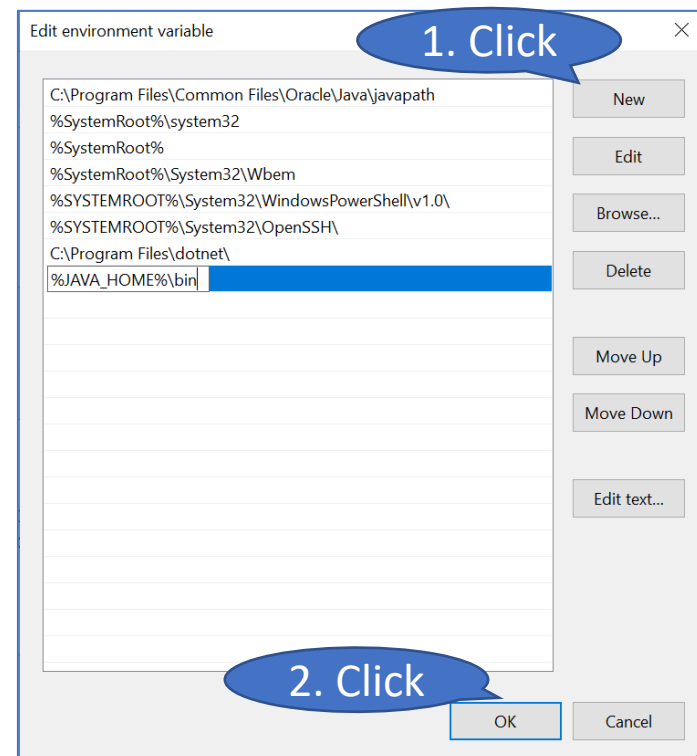
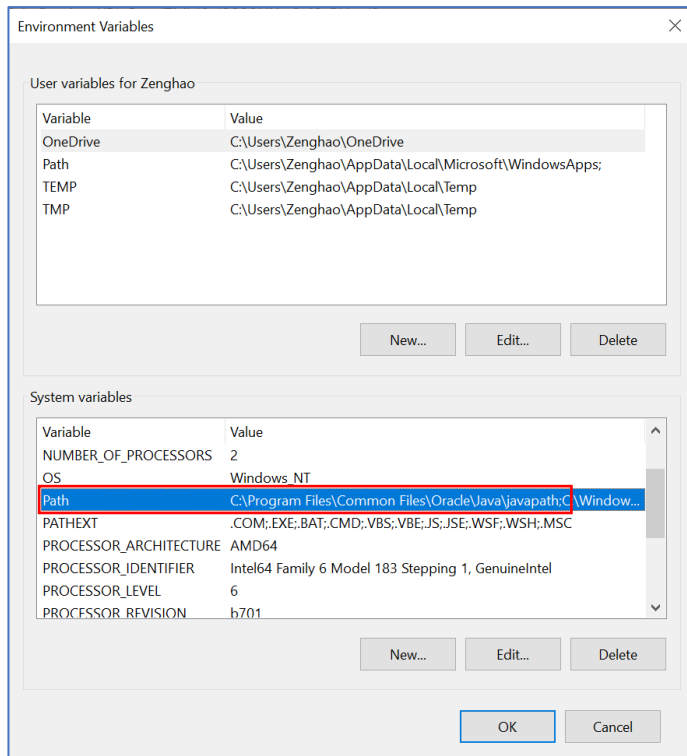
Set “JAVA_HOME”

- If “JAVA_HOME” is not present in “System variables”, you have to create it by yourself.



Set “Path”

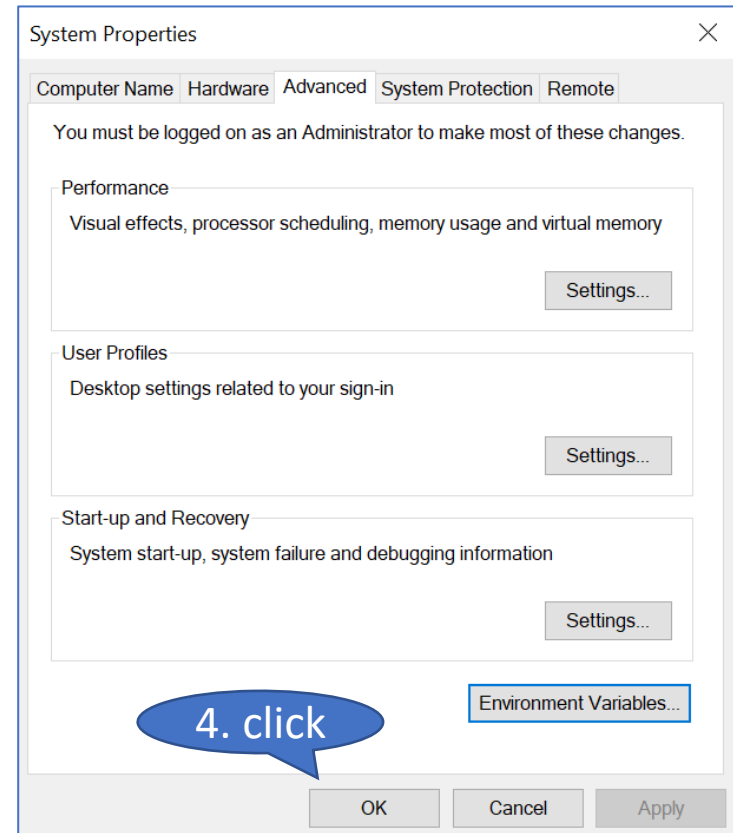
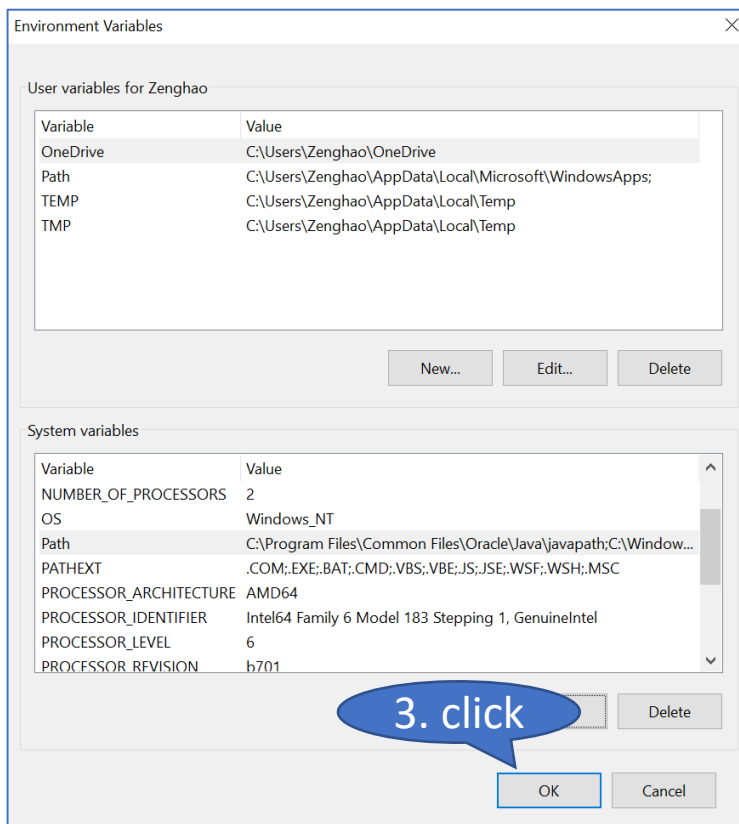
- In the “environment variables”, there is a variable named **Path**, you need to double click and edit it.



- If your “**JAVA_HOME**” has been added, you can simply insert the following value, i.e., “**%JAVA_HOME%\bin**”.

Finish “environment variable” setting

- After editing **Path** variable, click “OK” to close current dialog, and click “OK” till all the windows are closed.



Test Java environment

- Open your *Terminal* and try to run “*java -version*” or “*java --version*”

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19042.508]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Zenghao>java -version
java version "21.0.6" 2025-01-21 LTS
Java(TM) SE Runtime Environment (build 21.0.6+8-LTS-188)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.6+8-LTS-188, mixed mode, sharing)
```

Using the Command-Line Tools

HelloWorld.java

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

- Copy the above code to a text file, and name it to **HelloWorld.java**
- Move it to **D:\oop\ch02**

Using the Command-Line Tools

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19042.508]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Zenghao>D:

D:\>cd .\oop\ch2

D:\oop\ch2>javac HelloWorld.java

D:\oop\ch2>java HelloWorld
Hello World!

D:\oop\ch2>
```

- Open the *Terminal*
 - Search “cmd” on Win10/Win11
- Change to the directory
 - D:
 - cd .\oop\ch02\
 - javac HelloWorld.java
 - java HelloWorld

Congratulations! You have just compiled and run your first Java program!

Exploring HelloWorld.java

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

- The first line declares a class named **HelloWorld**, which is **public**, that means that any other class can access it.
 - Notice that when we declare a public class, we must declare it inside a file with the same name (i.e., **HelloWorld.java**), otherwise we'll get an error when compiling.

Exploring HelloWorld.java

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

- The second line is **the entry point** of our Java program.
 - “**public**” again means that anyone can access it.
 - “**static**” means that you can run this method without creating an instance of **HelloWorld**.
 - “**void**” means that this method doesn't return any value.
 - “**main**” is the name of the method.
 - “**String[] args**” are the arguments we get inside the method when running the program with parameters. It's an array of strings.

Exploring HelloWorld.java

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

- The third line calls a **standard output function** to print a line to the console.
 - **“System”** is a pre-defined class that Java provides us and it holds some useful methods and variables.
 - **“out”** is a static variable within System that represents the output of your program (stdout).
 - **“println”** is a method of out that can be used to print a line.

Things to take away

1. Every valid Java Application must have a class definition (that matches the filename).
2. The main method must be inside the class definition.
3. The compiler executes the codes starting from the main function.

```
public class HelloWorld {  
    public static void main(String[] args) {  
        // Write your code here  
    }  
}
```

This is a valid Java program that does nothing.

Using an Integrated Development Environment (IDE)

Using an IDE

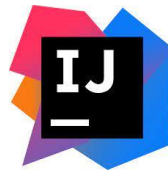
- Why?
 - More productive;
 - Several choices;
 - Free;
 - ...
- Most popular Java IDE:
 - **Eclipse** (Opensource and free)
 - **IntelliJ IDEA** (Community version is also free)
 - **NetBeans**
 - ... see more comparison on <https://hackr.io/blog/best-java-ides>



Quick question 2

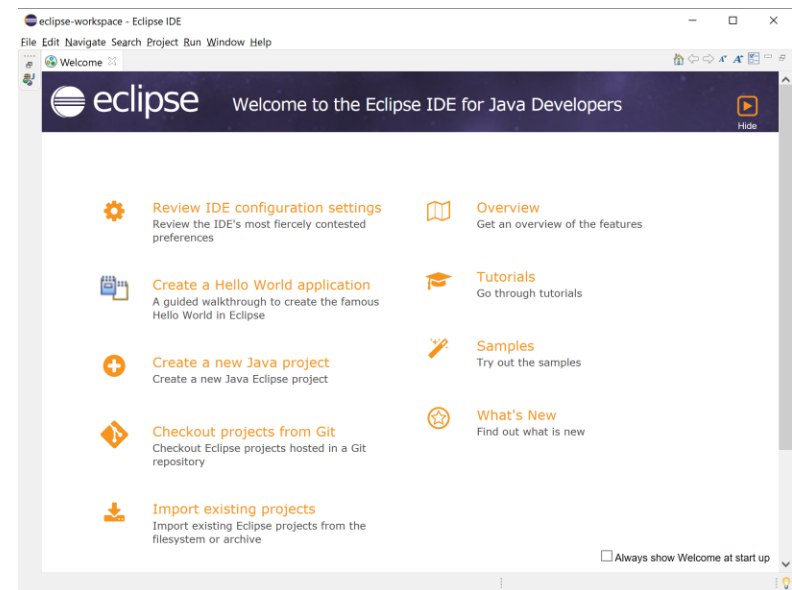
Which one have you ever used?

- A. Eclipse
- B. IntelliJ IDEA
- C. NetBeans
- D. Others like BlueJ, JDeveloper, etc.



More about Eclipse

- A dedicated Java IDE as one of the big three of Java IDEs
- First Released - November 2001
- Cross Platform - Linux/macOS/Windows
- Features:
 - Code completion
 - Refactoring
 - Syntax checking
 - Source code formatting
 - ...



Eclipse is constantly assisting while coding!

Let's try Eclipse

- Download Eclipse installer from <https://www.eclipse.org/downloads/>



[Projects](#) [Supporters](#) [Collaborations](#) [Resources](#) [The Foundation](#)



Download Eclipse Technology that is right for
you

Register for FOSSASIA Summit 2025

Join us for three days of open source collaboration and knowledge exchange from 13-15 March at this hybrid event based in Bangkok.

[Register Here](#)



Install your favorite desktop IDE pack

[Learn More](#)

[Download](#)

[Downl](#)

Download x86_64

Download AArch64

Click here



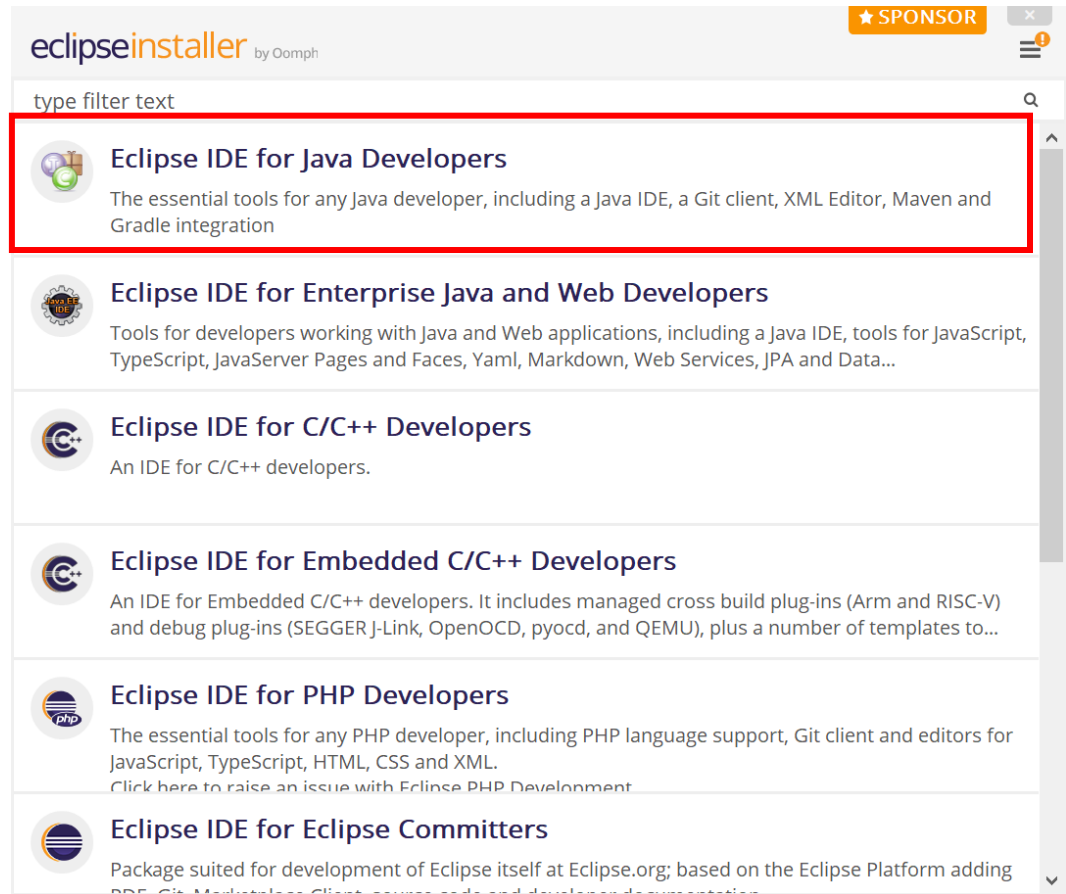
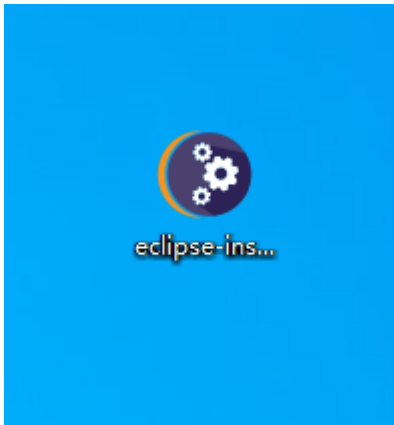
The Eclipse Temurin™ project provides high-quality, TCK certified OpenJDK runtimes and associated technology for use across the Java™ ecosystem.

[Learn More](#)

[Download](#)

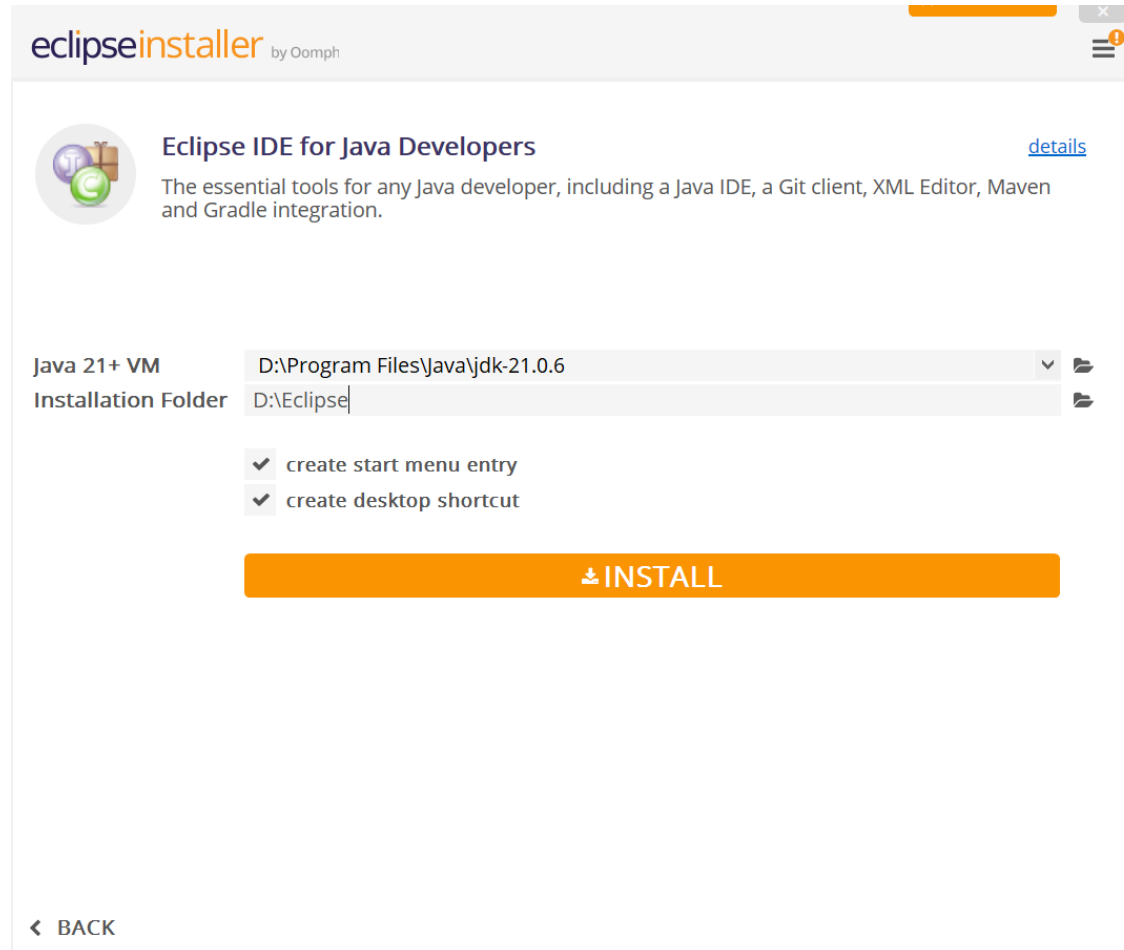
Install Eclipse

- Open the Eclipse-installer and choose the first one, i.e., “Eclipse IDE for Java Developers”.



Choose the installation folder

- Find the path to the above installed JDK



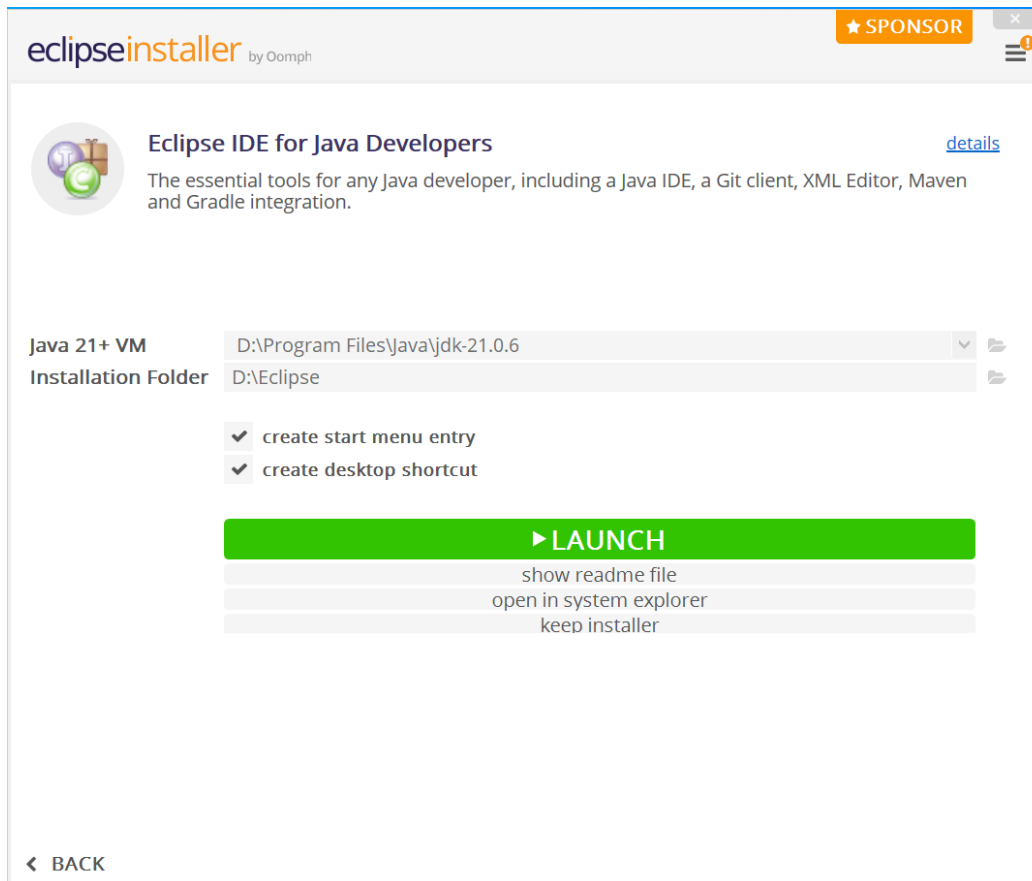
Begin installation

- Choose “Accept Now” to agree with the user terms and begin installation.



Launch Eclipse

- After a while, when you see the “Launch” button, you click it and run Eclipse.



If the installer fails

- Try <https://www.eclipse.org/downloads/packages/>

Eclipse IDE 2024-12 R Packages

Eclipse IDE for Enterprise Java and Web Developers

544 MB 296,361 DOWNLOADS



Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces, Yaml, Markdown, Web Services, JPA and Data Tools, Maven and Gradle, Git, and more.



Windows | x86_64 | AArch64
macOS x86_64 | AArch64
Linux x86_64 | AArch64 | riscv64

Click [here](#) to raise an issue with the Eclipse Web Tools Platform. Maintainers will move opened issues to the right place.
Click [here](#) to raise an issue with the Eclipse Platform.
Click [here](#) to raise an issue with Maven integration for web projects.
Click [here](#) to raise an issue with Eclipse Wild Web Developer (incubating).



Eclipse IDE for Java Developers

340 MB 256,314 DOWNLOADS

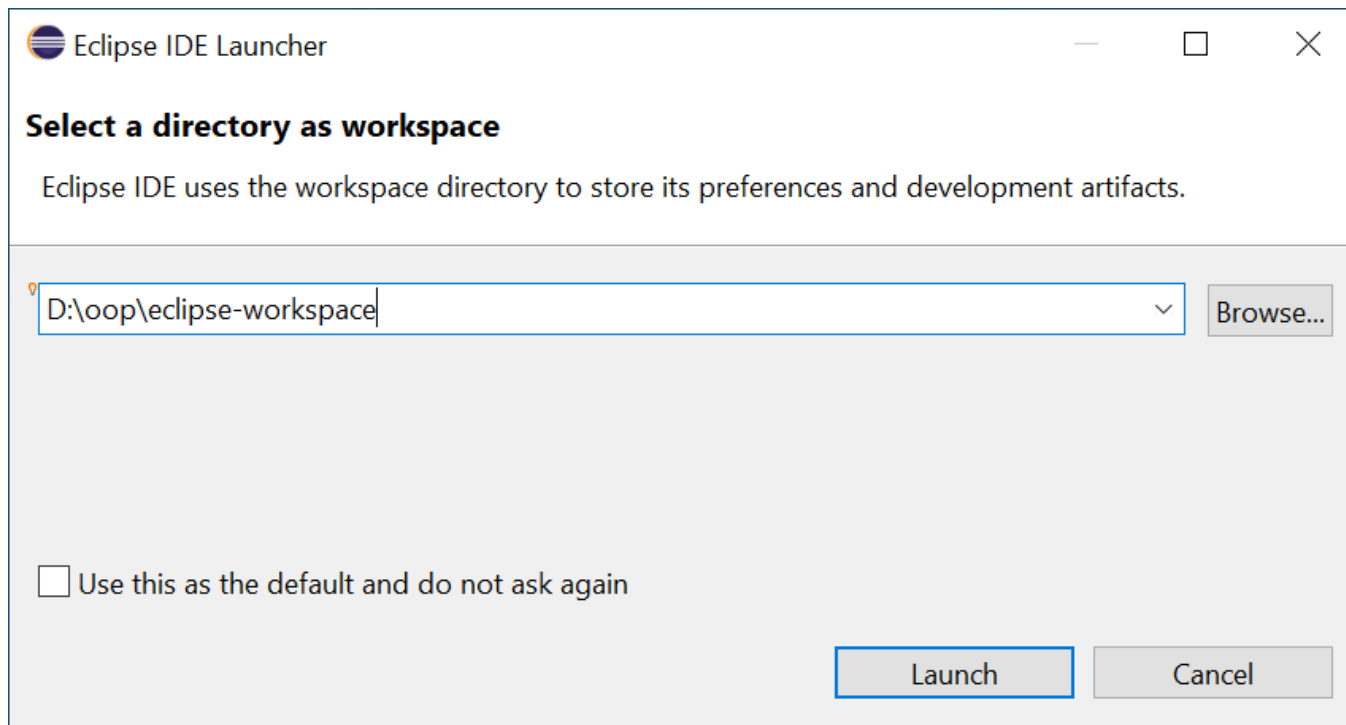


Windows | x86_64 | AArch64
macOS x86_64 | AArch64
Linux x86_64 | AArch64 | riscv64

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration

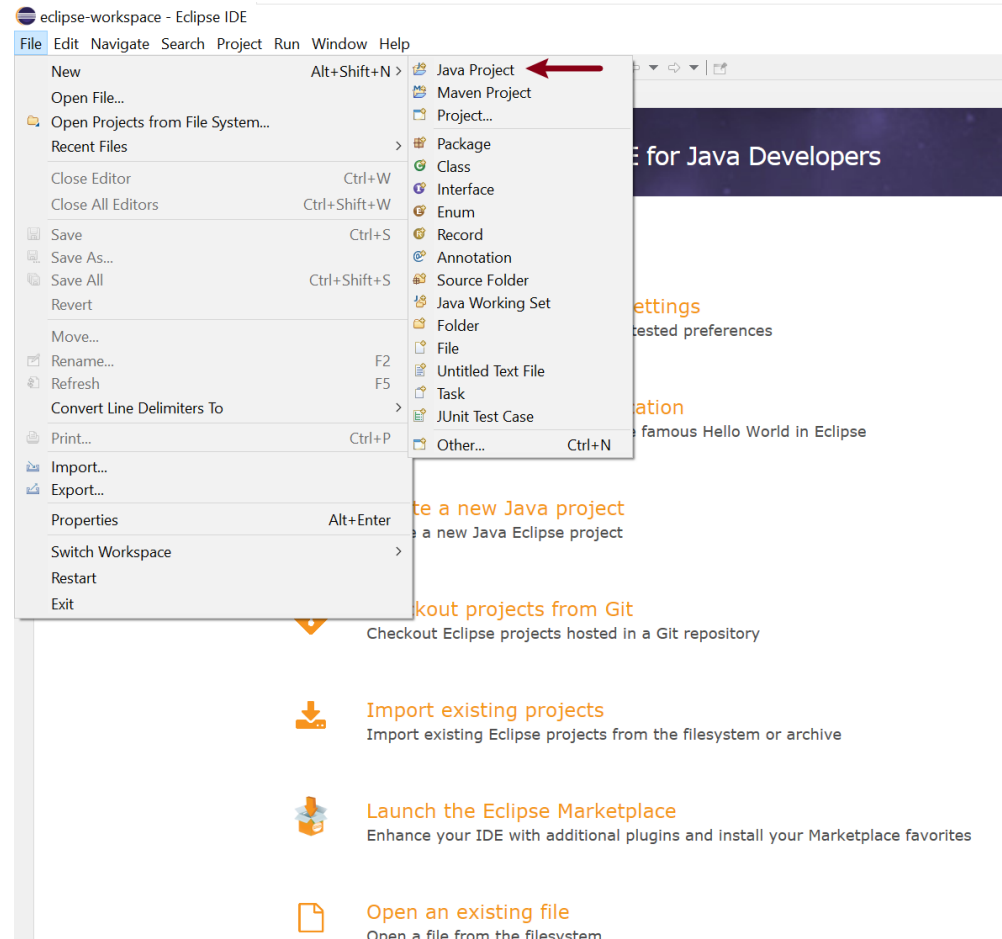
Setup Eclipse workspace

- Workspace is the location where you store preferences of Eclipse and your development artifacts.



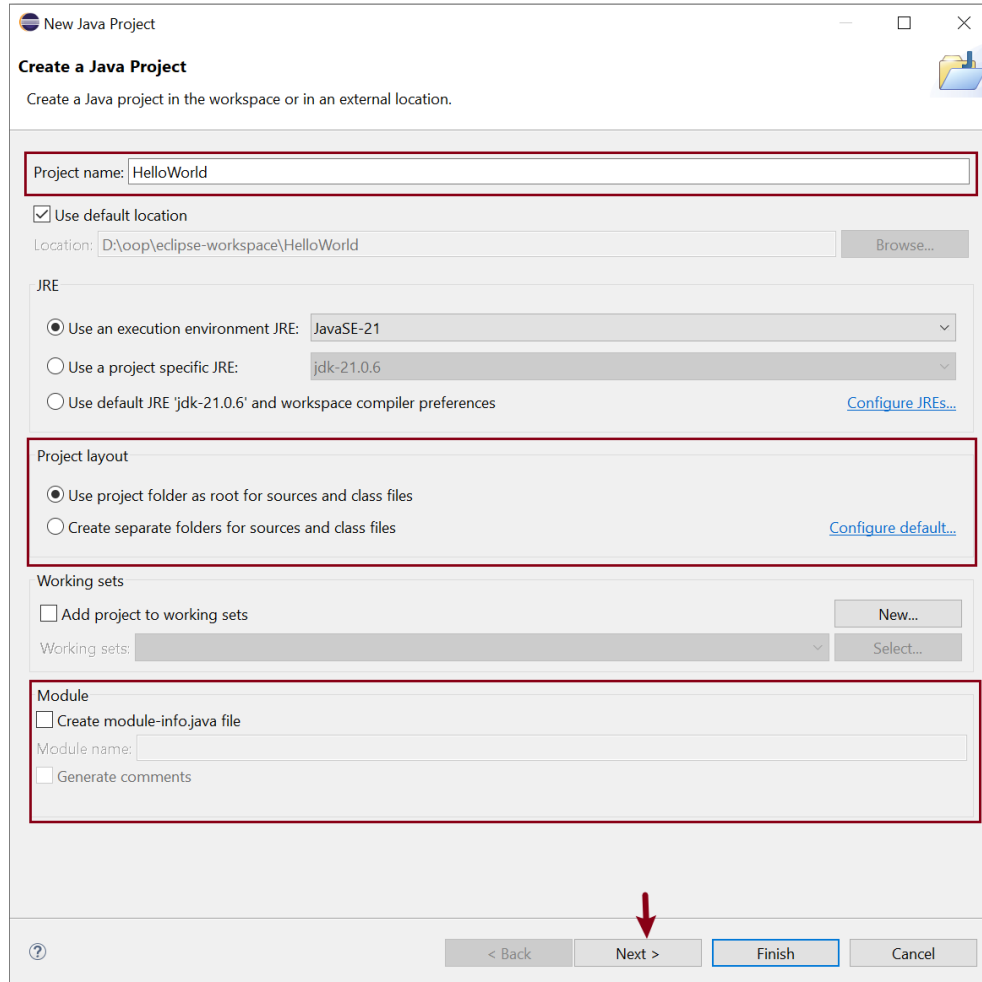
Create a Java project

- File -> New -> Java Project



Java project pre-configuration

- You should specify the “Project name”, change the “Project layout” from second choice to the first, and uncheck the “Module” option.



New Java Project

Create a Java project in the workspace or in an external location.

Project name: HelloWorld

☒ Use default location
Location: D:\oop\eclipse-workspace\HelloWorld [Browse...](#)

JRE

☒ Use an execution environment JRE: JavaSE-21 [Configure JREs...](#)
☐ Use a project specific JRE: jdk-21.0.6
☐ Use default JRE 'jdk-21.0.6' and workspace compiler preferences

Project layout

☒ Use project folder as root for sources and class files [Configure default...](#)
☐ Create separate folders for sources and class files

Working sets

☐ Add project to working sets [New...](#)
Working sets: [Select...](#)

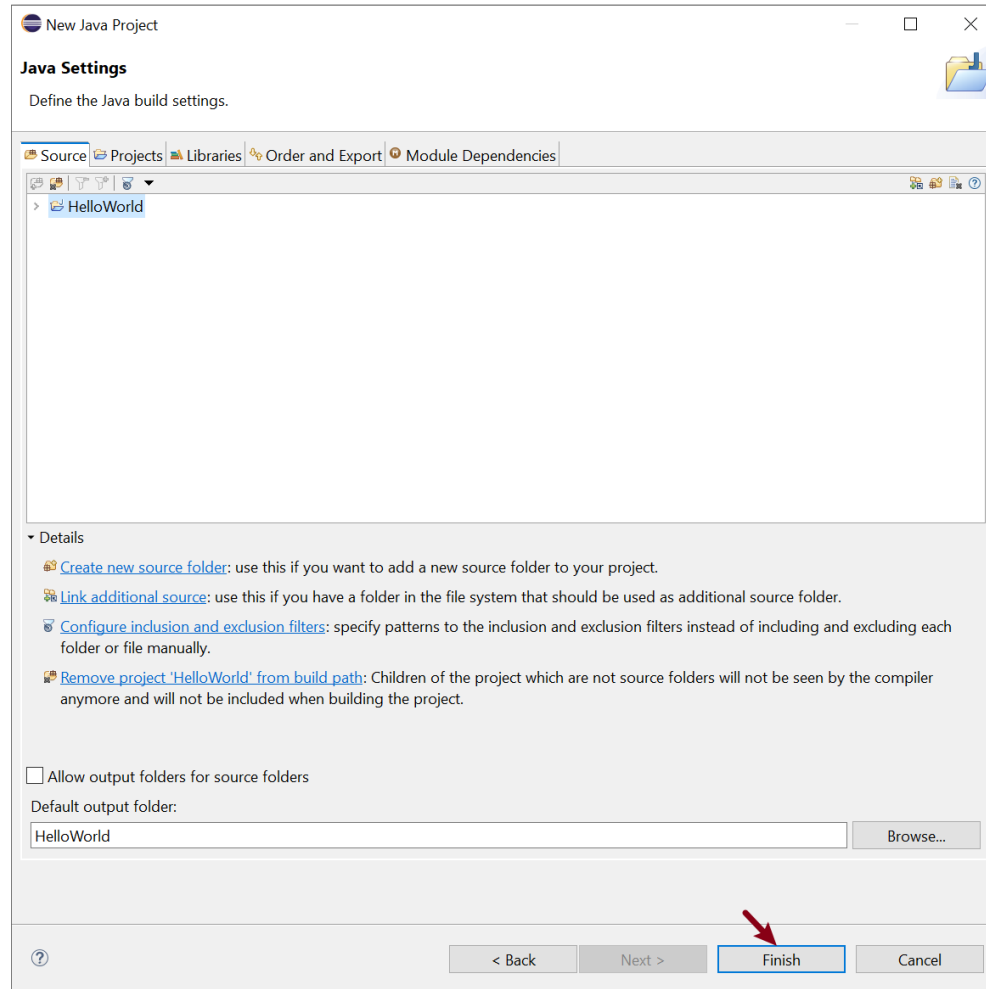
Module

☐ Create module-info.java file
Module name:
☐ Generate comments

[? < Back](#) [Next >](#) [Finish](#) [Cancel](#)

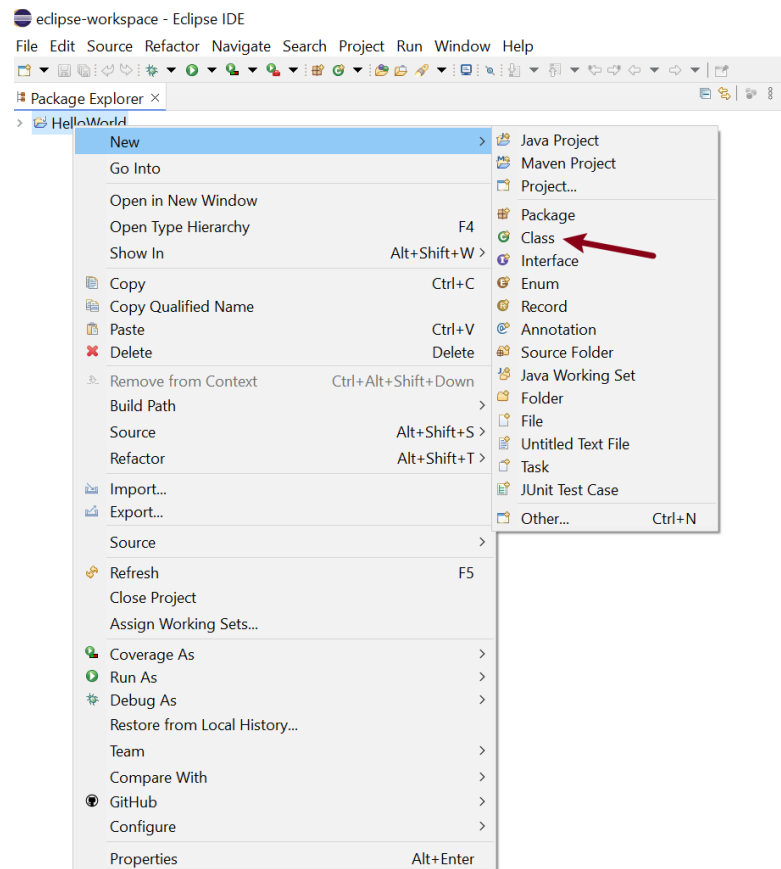
The last step of project setup

- Click “Finish” to finish setup.



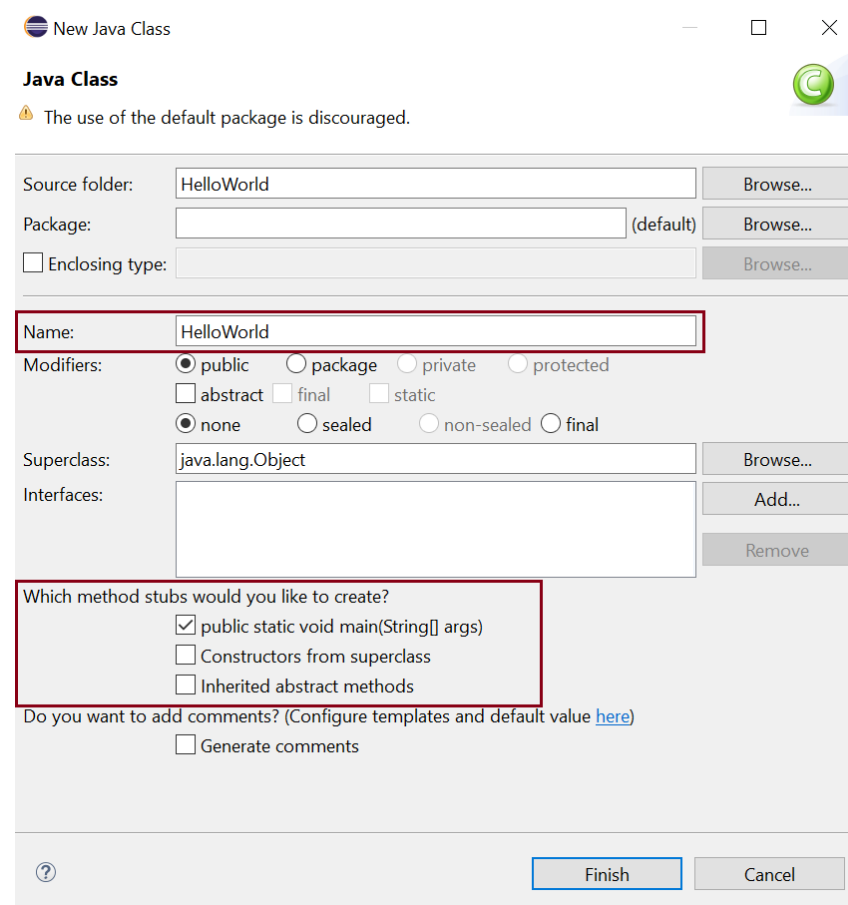
Create Java class

- After the Java project has been set up, you need to create a class in .java file. Right click the “HelloWorld” name, select “New” -> “Class”, a java file will be created.



Configure Java Class

- In this step, you need to specify the name of Java class, and check box before “public static void main(String[] args)”.



New Java Class

Java Class

The use of the default package is discouraged.

Source folder: HelloWorld Browse...

Package: (default) Browse...

☐ Enclosing type: Browse...

Name: HelloWorld

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static
☒ none ☐ sealed ☐ non-sealed ☐ final

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

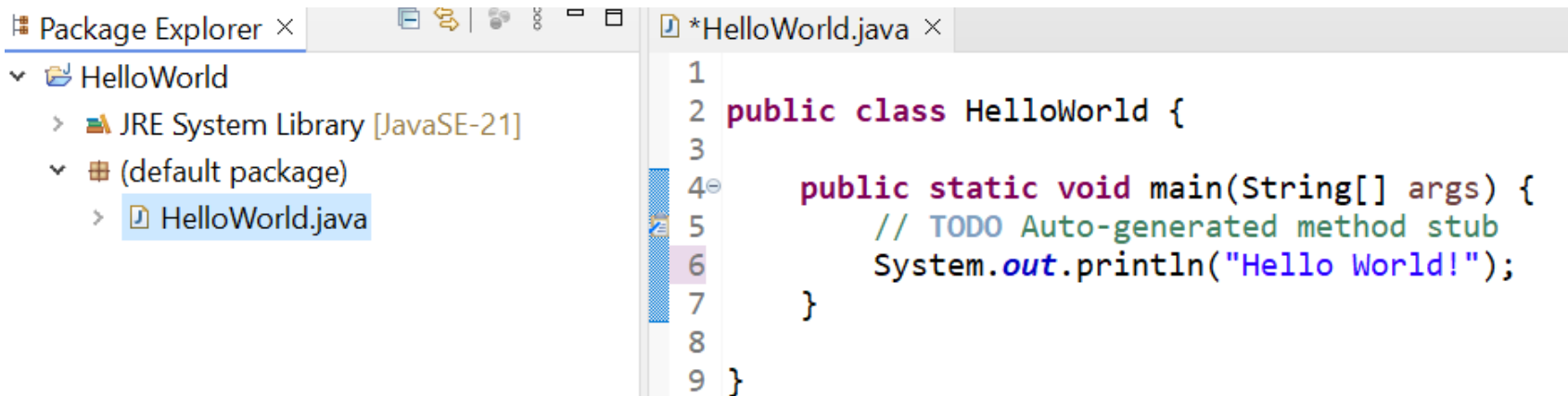
☒ public static void main(String[] args)
☐ Constructors from superclass
☐ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Finish Cancel

Write our first line of code

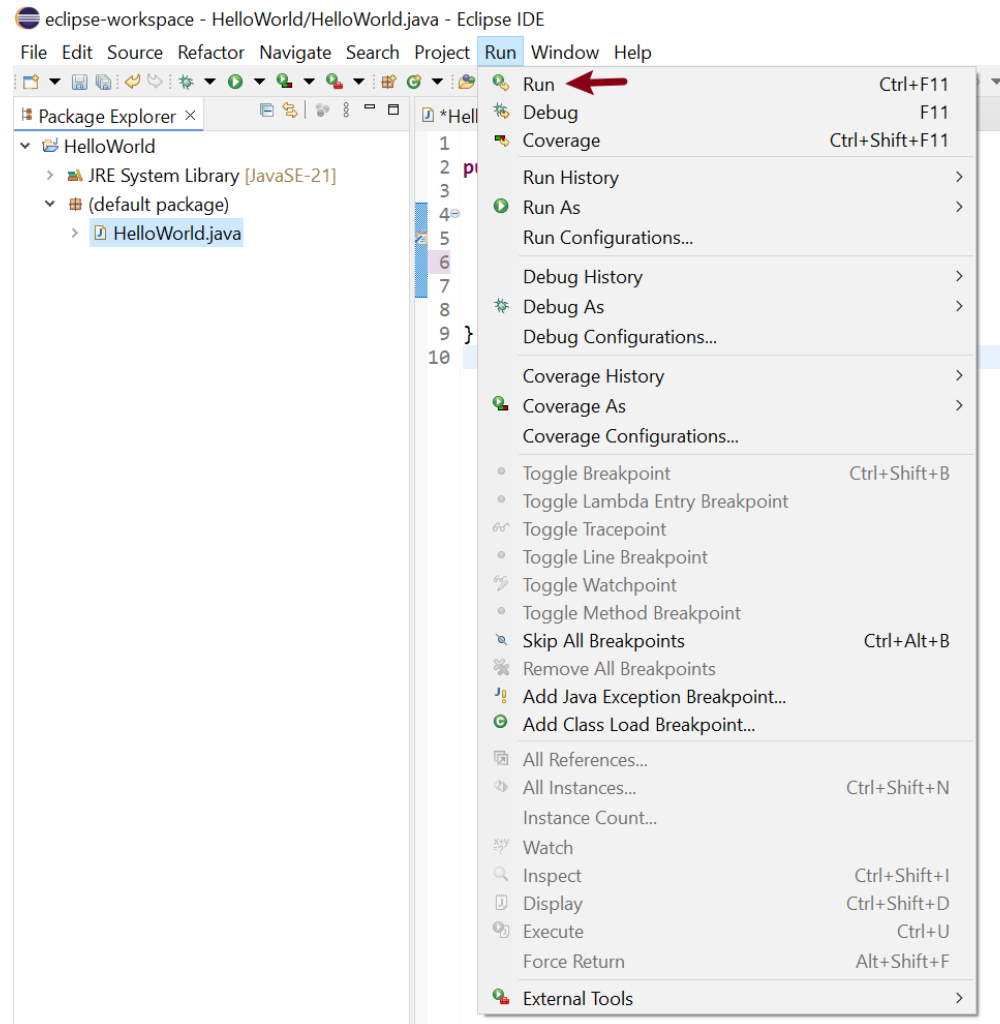
- Inside the main method of HelloWorld, you can write our conventional first line of code “`System.out.println("Hello World!");`”.



```
1
2 public class HelloWorld {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         System.out.println("Hello World!");
7     }
8
9 }
```

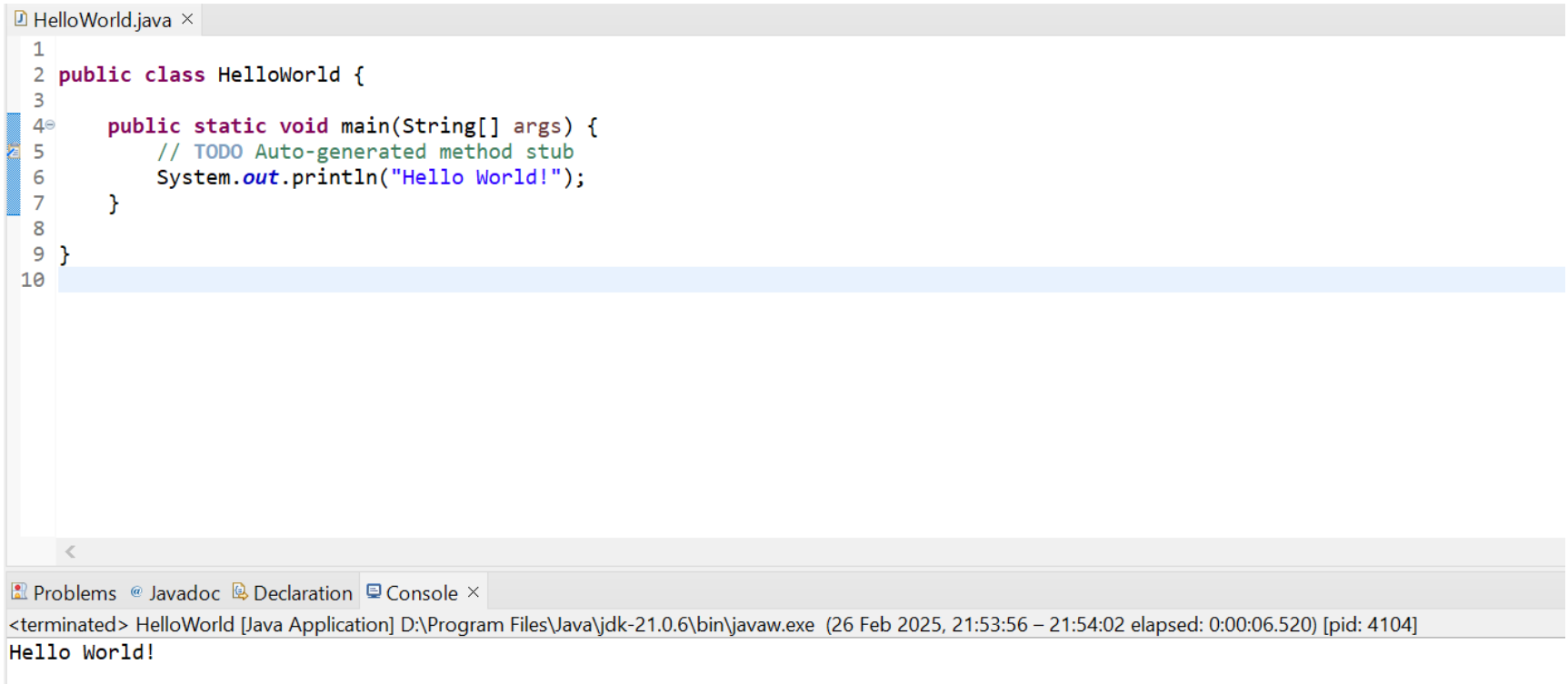
Run HelloWorld in IDE

- Save the file and click “run” button to execute our program.



Console outputs “Hello World”

- Once the execution completes, the console will print out “Hello World!”.



```
1  
2 public class HelloWorld {  
3  
4     public static void main(String[] args) {  
5         // TODO Auto-generated method stub  
6         System.out.println("Hello World!");  
7     }  
8  
9 }  
10
```

Problems @ Javadoc Declaration Console ×

<terminated> HelloWorld [Java Application] D:\Program Files\Java\jdk-21.0.6\bin\javaw.exe (26 Feb 2025, 21:53:56 – 21:54:02 elapsed: 0:00:06.520) [pid: 4104]
Hello World!

Welcome.java

```
/**
 * This program displays a greeting for the reader.
 * @version 1.30 2014-02-27
 * @author Cay Horstmann
 */
public class Welcome
{
    public static void main(String[] args)
    {
        String greeting = "Welcome to Core Java!";
        System.out.println(greeting);
        for (int i = 0; i < greeting.length(); i++)
            System.out.print("=");
        System.out.println();
    }
}
```

Learn to use Violet UML editor

Violet UML Editor

- Violet is a UML editor with these benefits:
 - Very easy to learn and use.
 - Draws nice-looking diagrams.
 - Completely free. Cross-platform.
 - Violet is intended for developers, students, teachers, and authors who need to produce simple UML diagrams quickly.



Violet UML Editor

What is UML?



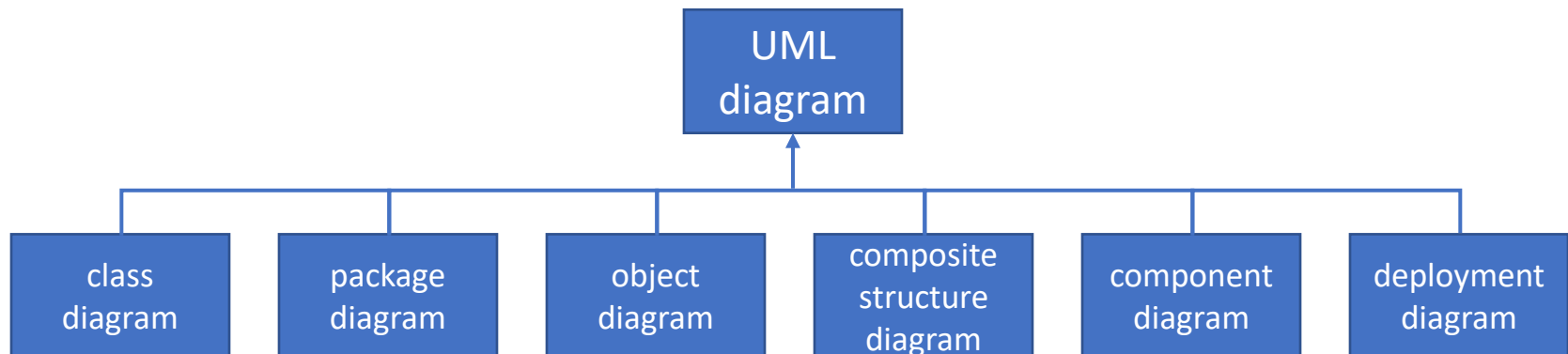
- UML is short for **Unified Modeling Language**, a language intended to help to communicate upon object programming concepts that are not intuitive.
 - UML was born in the middle 90's by merging three methods: OMT, Booch and OOSE.
 - This is a graphical language composed by 13 diagrams (UML 2.0).
 - UML is powerful because these diagrams are always very friendly.
 - It's an efficient way to communicate between programmers and end users.
 - It can also be very precise and helps to avoid ambiguities when you define your system.
- **UML is not a method.**
 - It's a language. So, UML will not help you to organize your project. It will just help you to describe it, to communicate over it, to clarify its features.

Introduction to diagram

- A diagram is a partial graphical representation of a system model.
- There are 13 types of them, which can be categorized into 2 types:
 - the static view
 - the dynamic view

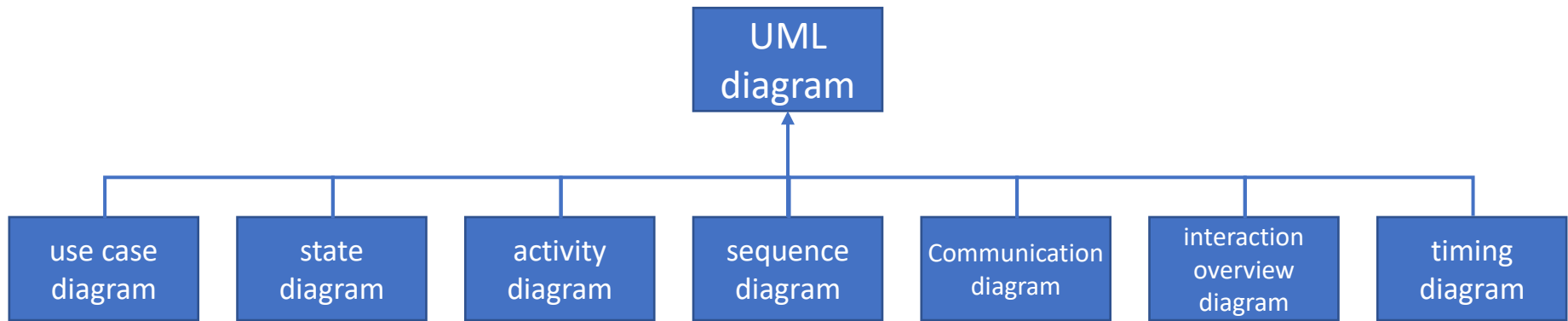
The static view

- The static view contains diagrams showing all the entities and behaviors of your system.
 - Think of an object as an entity. These diagrams would expose all your system entities, define their rules and how they will interact all together.



The dynamic view

- The dynamic view contains diagrams showing scenarios.
 - This time, you see how entities works, not only how they are made.





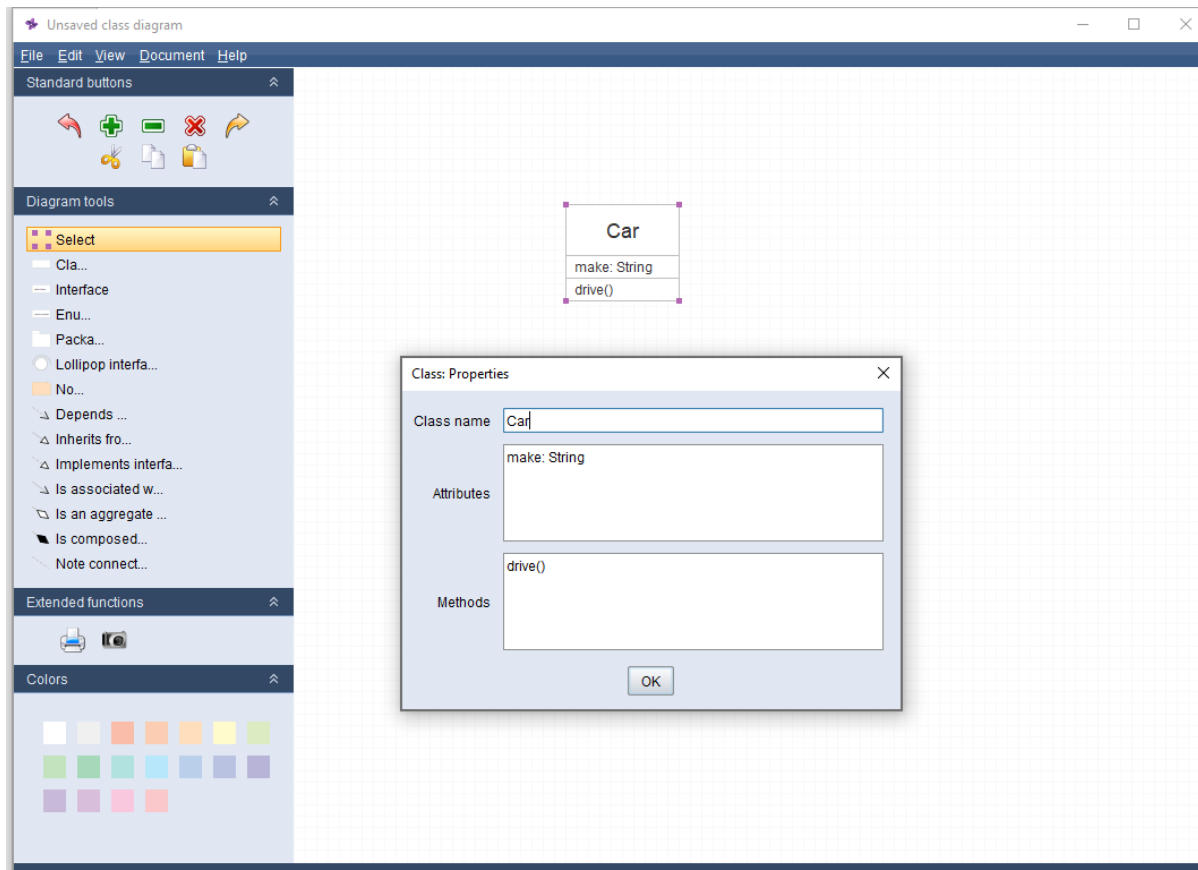
The UML class diagram (3 min)

Class Diagrams

<https://www.youtube.com/watch?v=qmqIwAdSLpQ>

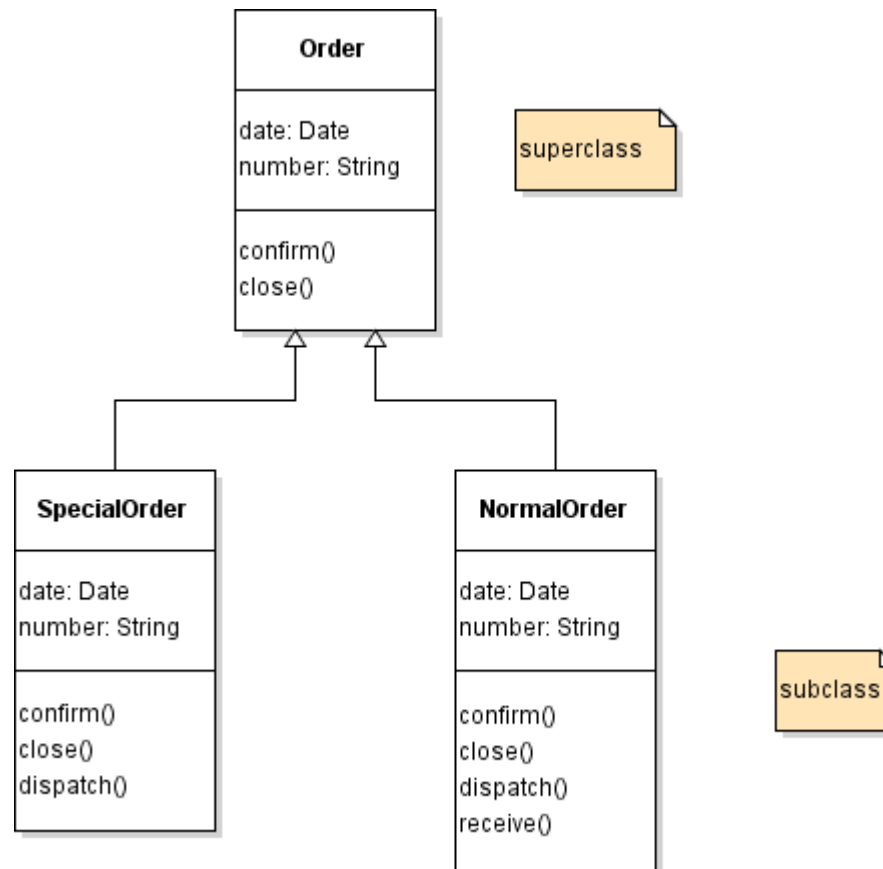
Download violet UML editor

- <http://alexdp.free.fr/violetumleditor/page.php?id=en:download>
- In order to run violet UML editor, you need to install JRE.



An example

- Inheritance illustration in violet UML editor.



Alternatives

- <https://www.diagrams.net/>

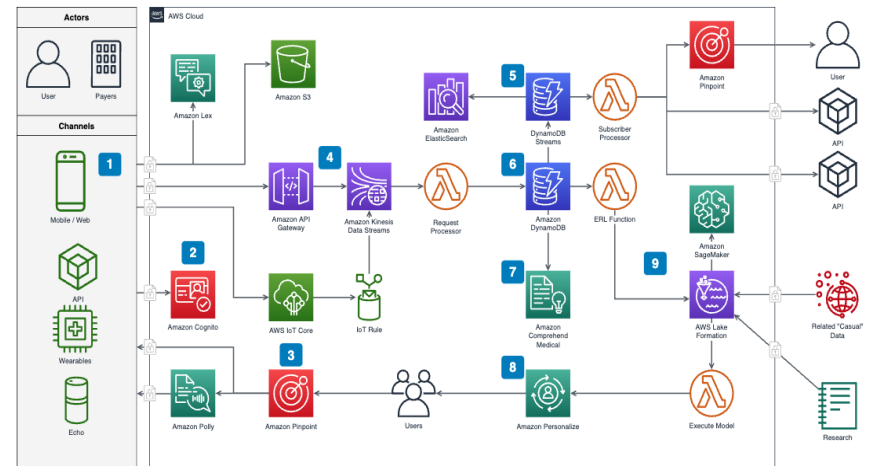
[Blog](#)[Start Now](#)

Security-first diagramming for teams.

Bring your storage to our online tool, or go max privacy with the desktop app.

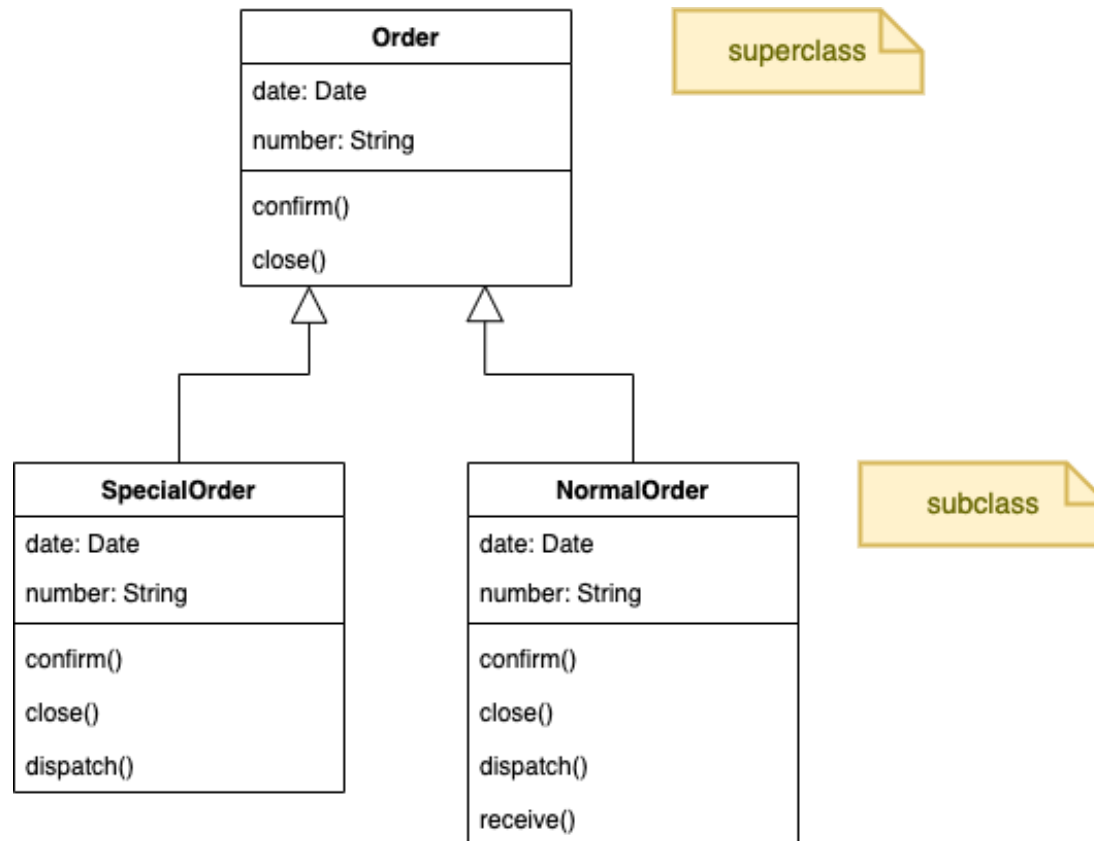
[Start](#)[Download](#)

No login or registration required.



The example via diagrams.net

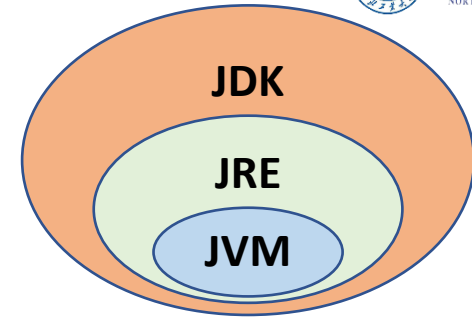
- <https://www.diagrams.net/>



Recap

➤ JDK, JRE, JVM?

- JDK is a software development kit
- JRE is a software bundle that allows Java program to run
- JVM is an environment for executing bytecode



➤ Run **HelloWorld.java**

- Using Command-Line Tools
- Using an IDE, e.g., Eclipse

➤ Use UML editor

➤ Violet or diagrams.net

