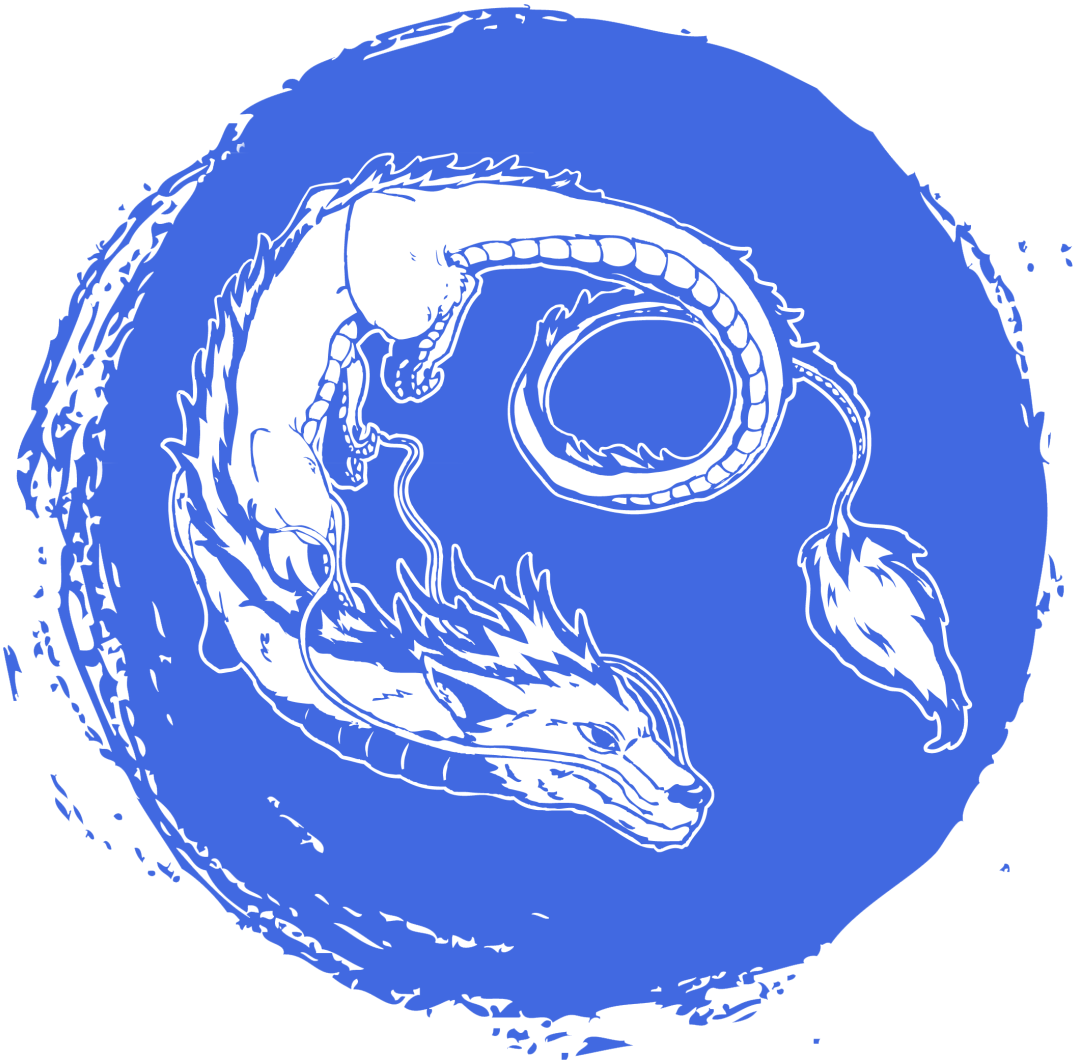
Java Workshop — Tutorial D0

version **#v0.9.2**



Yet Another Kind of Assistants 2022 <yaka@tickets.assistants.epita.fr>

Copyright

This document is for internal use at EPITA (website) only.

Copyright © 2020-2021 Assistants <yaka@tickets.assistants.epita.fr>

The use of this document must abide by the following rules:

- ▷ You downloaded it from the assistants' intranet.*
- ▷ This document is strictly personal and must **not** be passed onto someone else.
- ▷ Non-compliance with these rules can lead to severe sanctions.

Contents

*https://intra.assistants.epita.fr

The goal of this document is to provide a list of tools that will be used during the Java workshop. Make sure all of these prerequisites are installed on your machine and work correctly before the workshop begins. You can also try to get familiar with those tools and experiment with them.

A more detailed presentation of each tool will be given during the first day of the workshop, so do not worry if you do not understand those tools for now.

1 IntelliJ IDEA

1.1 Presentation

IntelliJ IDEA is an IDE (Integrated Development Environment) for projects built around the JVM (Java Virtual Machine). You are of course free to use the IDE of your choice, but it is recommended that you use IntelliJ IDEA Ultimate for your Java projects for the following reasons:

- It is one of the best Java IDEs available so far.
- You can get a free education license for the Ultimate edition.
- This IDE fully supports and integrates Maven, and many other useful tools.

1.2 Installation

You can download IntelliJ IDEA Ultimate from the JetBrains website and follow the instructions on the download page.

When starting the application, you will be asked for a license key or a JetBrains account. You should be able to get an education license when creating an account with your EPITA credentials, and use the application for free.

After opening a Maven project, you may be prompted to select the Java JDK that will be used by the IDE[2] to compile this project. You will need to tell the IDE to use the JDK you downloaded and installed earlier.

2 Java

2.1 Presentation

The Java programming language will be the focus of the coming workshop. A proper presentation will be given on the first day, we will only focus on the prerequisites for now.

[2] https://www.jetbrains.com/help/idea/sdk.html

2.2 Installation

In order to develop and run java applications, you need a Java Development Kit (JDK). We will be using JDK 16 for every Java-based project this year.

Since JDK16 was released recently, it may not be available in your package manager, which means you will have to install it from the IDE.

From IntelliJ's welcome menu, click the `New Project` button. In the `Project SDK` section, scroll down to `Download JDK` then select `OpenJDK` version 16.

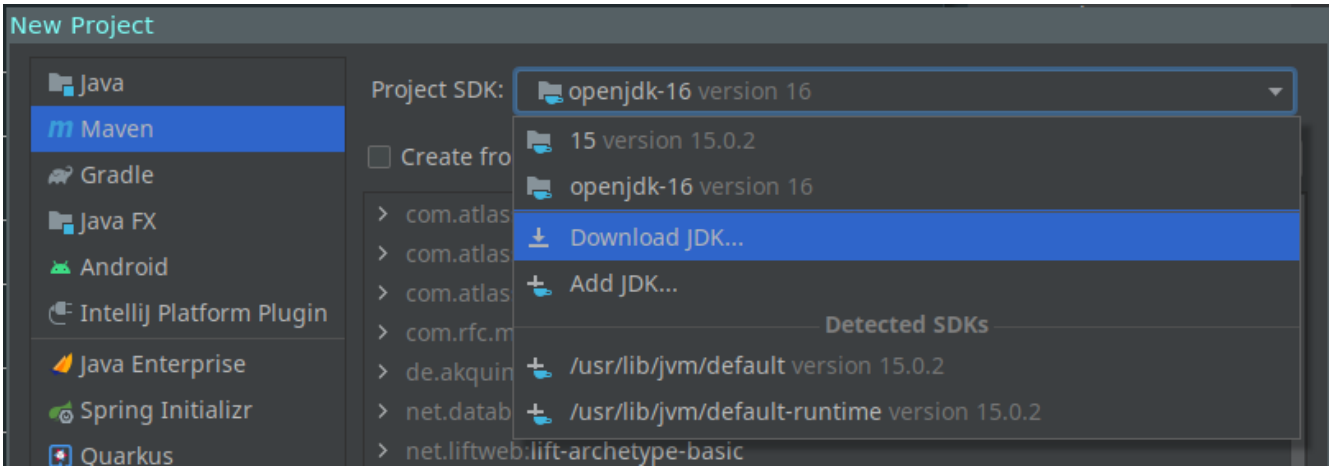


Fig. 1: Home menu

If for some reason one of your project does not have the correct JDK setup, you can access the JDK selection from the `File` > `Project Structure` menu. Make sure `Project Language Level` is set to use the SDK default.

3 Maven

3.1 Presentation

Maven is a project management and build tool for Java-based projects. During the workshop, you will mainly use Maven as a build system for your projects, similar to Cmake/Autotools with C and C++.

3.2 Installation

We will be using Maven version 3.6.*. Maven should be available to download by most package managers. You can find detailed installation instructions and other helpful guides (such as maven phases and project bootstraps, which will also be explained later in this document) on the Maven website[1].

[1] https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

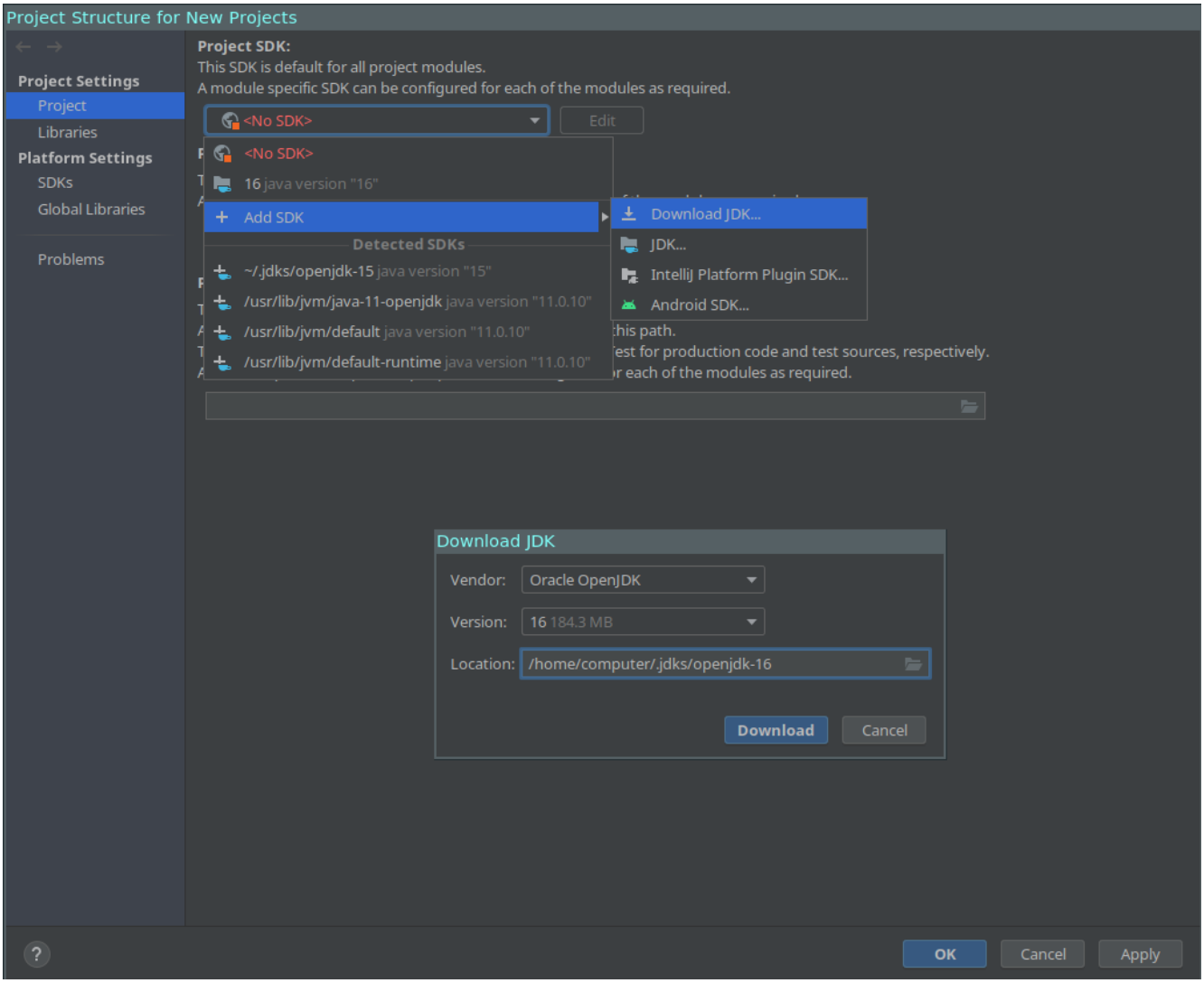


Fig. 2: Project Structure menu

3.3 Cheat sheet

Here are the maven commands you will use the most during the workshop:

- `mvn clean`: This command cleans the maven project by deleting the target directory.
- `mvn compile`: This command compiles the source Java classes of the project.
- `mvn test`: This command runs the test cases of the project.
- `mvn package`: This command builds the maven project and packages it into a JAR. It also runs the test cases of the project.

4 Bootstrap a maven project

4.1 Setup of a Maven project

You can generate a Maven project using the following command:

```
mvn archetype:generate -DgroupId=fr.epita.example \
                       -DartifactId=example-app \
                       -DarchetypeArtifactId=maven-archetype-simple \
                       -DarchetypeVersion=1.4 \
                       -DinteractiveMode=false
```

Here are the key parameters to understand:

- `groupId`: The unique identifier of your project across all projects. By convention, it takes the following form: `<reversed_domain_name>.<project_name>`.
- `artifactId`: The name of your project.

In order to enforce the use of Java 16 as our target Java version you need to modify the `pom.xml` file and change the `properties` fields as follows:

```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>16</maven.compiler.source>
    <maven.compiler.target>16</maven.compiler.target>
</properties>
```

You may also need to change the JUnit version.

If you prefer, it is also possible to setup a Maven project directly from the IDEA IDE. You can learn more from the IntelliJ website[3].

[3] https://www.jetbrains.com/help/idea/maven-support.html

4.2 Structure of a Maven project

Here is a quick overview of the project generated by the above command.

- `example-app/`: The root directory of your project.
- `example-app/pom.xml`: The POM (Project Object Model) is the file that describes your project, and is used by Maven to build and manage it.
- `example-app/src/main/java/`: The root directory of your project's main source files.
- `example-app/src/test/java/`: The root directory of your project's test source files.
- `[...]/fr/epita/example/App.java`: The *App* class from the *fr.epita.example* package. Classes and Packages are core elements of Java projects. You will learn more about them during the first day of the workshop.
- `example-app/src/site/`: You can ignore and even remove this directory.

You can find more information about the Standard Directory Layout on the Maven website[4].

Don't be afraid, I just wanna help you.

[4] https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html