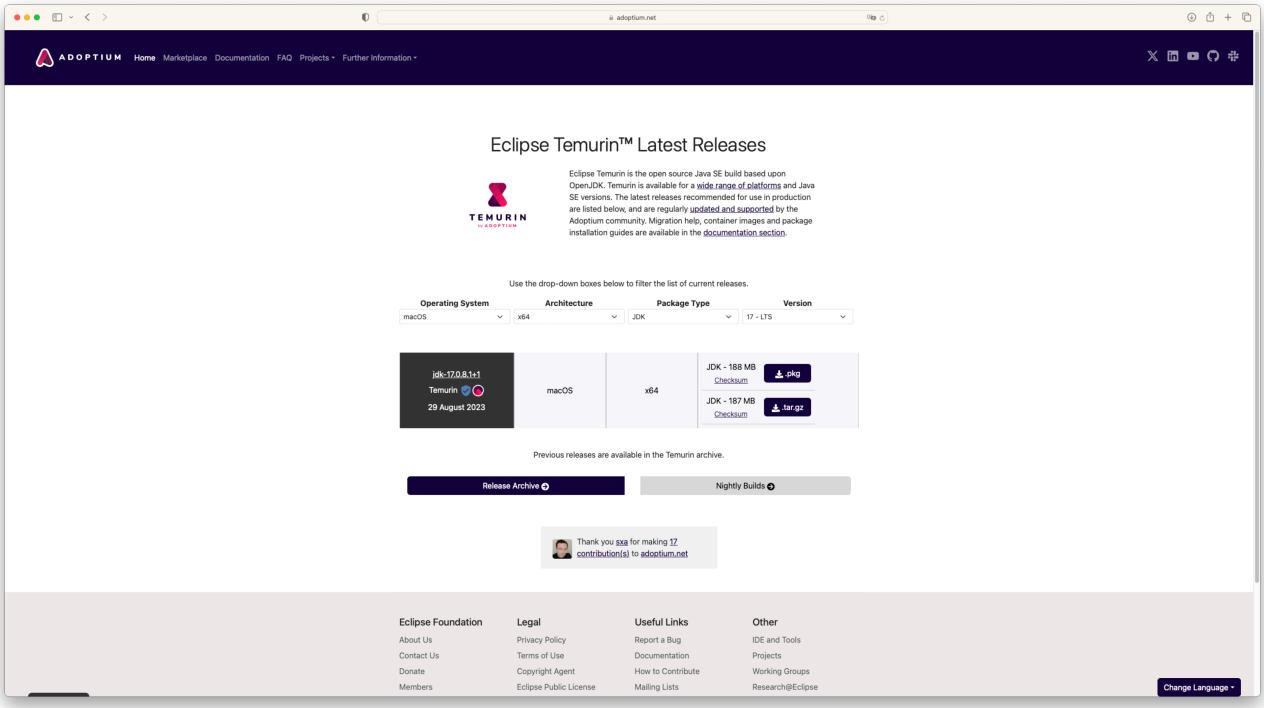
Introduction rapide à Java, IntelliJ et Maven[1]

1. Installation de Java

Pour développer des applications en Java, il est nécessaire d'installer le kit de développement pour le langage Java (Java développement kit, abrégé JDK). Les programmes réalisés dans le cadre des laboratoires de POO devront pouvoir être compilés et exécutés au moyen de l'environnement JDK 17 (Java SE Standard Development Kit) distribué en version open source (adoptium.net/temurin/releases) ou propriétaire (https://www.oracle.com/java/technologies/downloads).



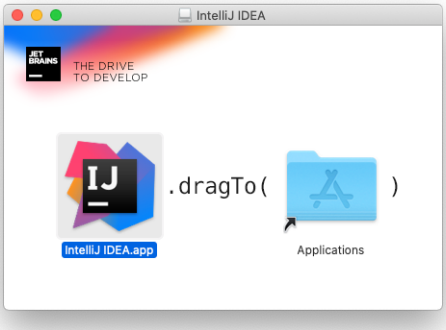
La variable d'environnement PATH doit référencer sur le répertoire bin où est installé Java (typiquement sur C:\Program Files\Java\jdk17.x.x.x\bin sous Windows).
Ce répertoire doit être listé avant celui d'autres installations Java qui pourraient exister dans le système (pour éviter les confusions, le mieux est encore de les désinstaller si elles ne sont pas utilisées).

2. Installation de IntelliJ

IntelliJ est un environnement de développement intégré (abrégé IDE) développé par Jetbrains. La version Community de IntelliJ est open source et téléchargeable gratuitement[2]. Une licence éducative de IntelliJ Ultimate (avec profiler) peut être obtenue en remplissant une demande sur le site Internet de Jetbrains avec votre email de la Heig-vd[3].

[1] Ce document fait référence au versions macOS des programmes. Toutefois, les exemples sont probablement réalisables sur une version légèrement différente du même programme.
[2] https://www.jetbrains.com/idea/download/
[3] https://www.jetbrains.com/community/education/#students

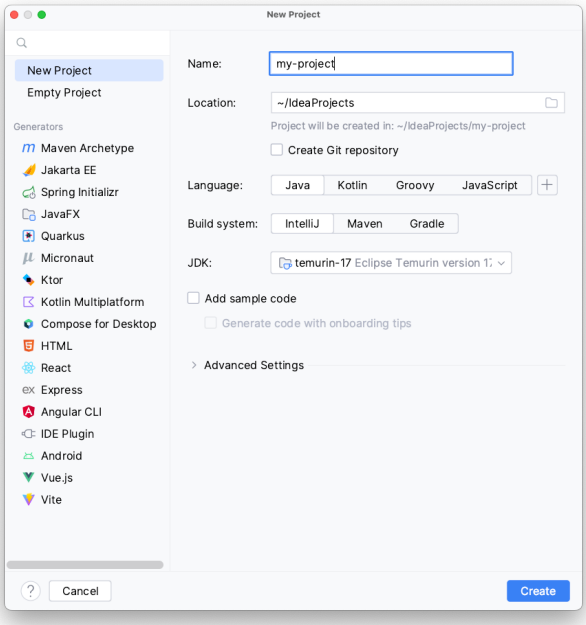
Comme illustré ci-dessous, l'installation d'IntelliJ est triviale sur la majorité des plateformes. En cas de difficultés lors de l'installation, se référer à la documentation[4] d'IntelliJ en sélectionnant la bonne plateforme.



3. Création d'un projet Java

Dans le menu « Fichier » ou dans la fenêtre d'accueil, sélectionner « Nouveau », puis « Projet… ». Ensuite, choisir la catégorie « New Project » et s'assurer que le langage est Java et le build système IntelliJ. Finalement, nommer le projet « my-project », comme illustré ci-dessous.

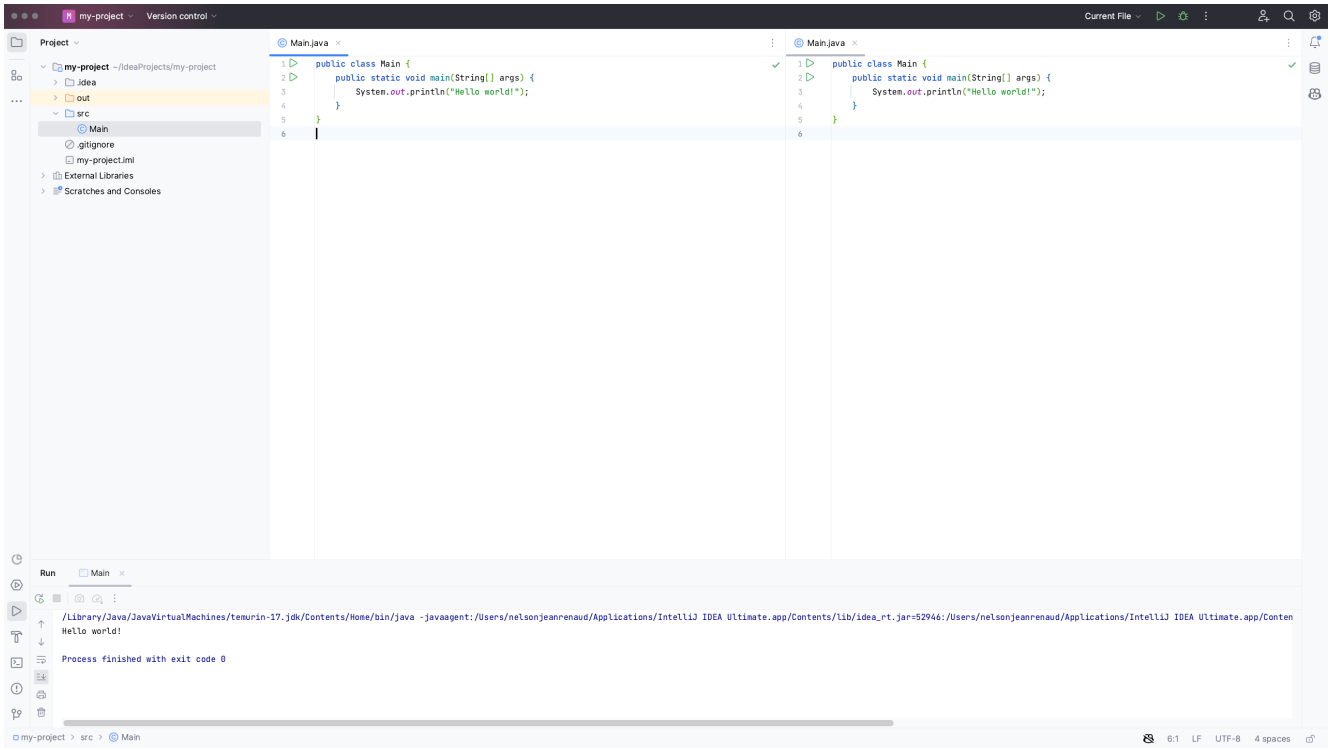
Attention, il est possible qu'une autre version du JDK, soit installé sur votre machine. De manière à utiliser la version 17, assurez-vous de l'avoir sélectionné dans la liste « JDK ».



[4] https://www.jetbrains.com/help/idea/installation-guide.html#standalone

4. Présentation de l'interface

Les différents onglets permettent de gérer le projet, et les fichiers en cours d'édition. L'onglet « Projet » offre une vue des fichiers et des dossiers du projet. L'onglet « Structure » offre une vue logique du projet. Par exemple, si plusieurs classes sont codées dans le même fichier, elles apparaissent de manière distincte dans l'onglet « Structure ». Le menu contextuel (clic droit) du fichier ouvert permet de subdiviser l'espace selon ses besoins, par exemple pour afficher plusieurs fichiers en même temps.



A l'aide du menu contextuel du dossier « src » (clic droit), créer une classe Java nommée « Main » et introduire le code suivant dans le fichier Main.java :

```
public class Main {

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

}
```

Pour compiler puis exécuter le code, il existe plusieurs solutions :

- Raccourci clavier CTRL-R
- Cliquer sur la flèche verte en haut de l'interface.
- Menu « Exécuter » : « Run 'Main'»
- Clic droit sur la classe Main de l'onglet « Projets », puis « Exécuter »

A l'aide d'un clic droit sur le dossier « src » dans l'onglet « Projets », créer une nouvelle classe Java, nommée « Car », et introduire le code suivant :

```
public class Car {
  private boolean started;

  public void start(){
    if (!started)
      System.out.println("Car started.");

    started = true;
  }

  public void stop(){
    if (started)
      System.out.println("Car stopped.");

    started = false;
  }
}
```

Modifier également le code de « Main.java » comme suit, puis compiler et tester le projet.

```
public class Main {

    public static void main(String[] args) {
        Car myCar = new Car();
        myCar.start();
        myCar.start();
        myCar.stop();
    }

}
```

5. Quelques astuces pour gagner du temps :

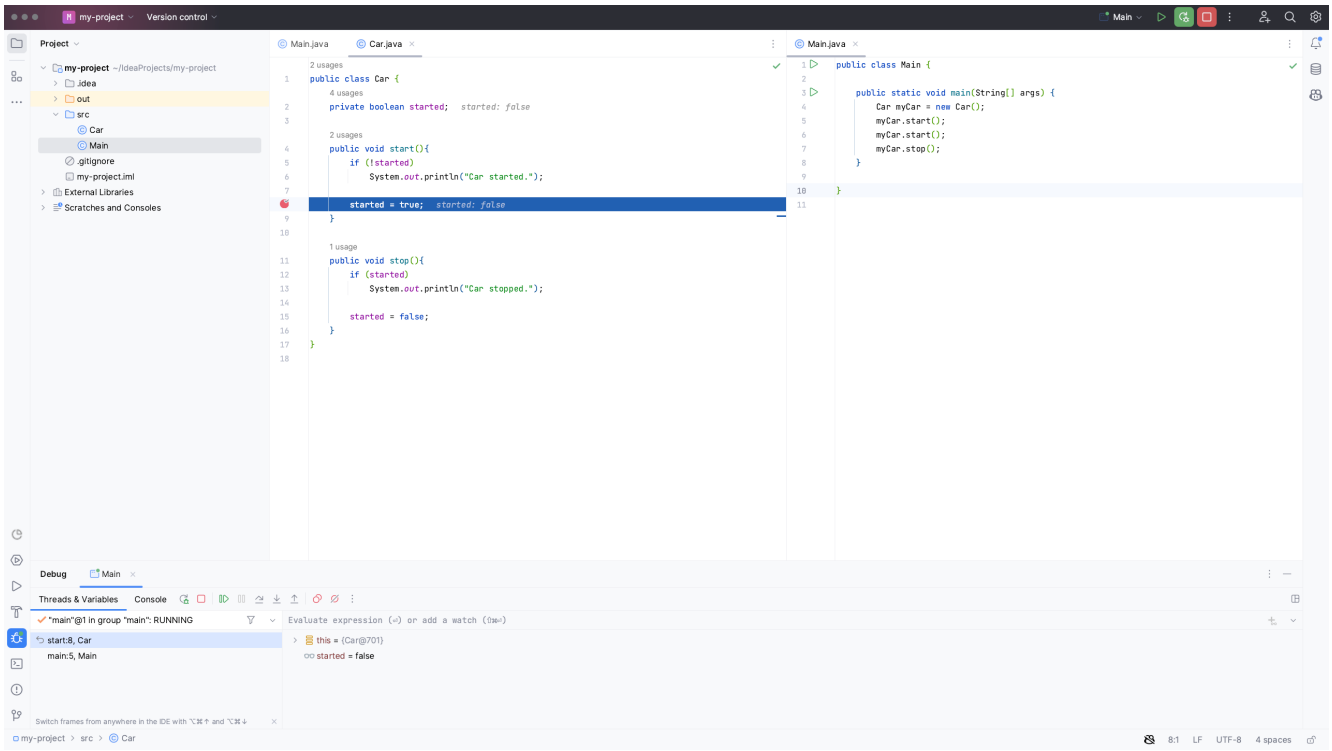
- L'utilitaire « Trouver une action » accessible depuis le menu d'aide ou le raccourci CMD+SHIFT+A est votre ami.
- L'installation du plugin « IDE Features Trainer »[5] et les vidéos d'entrainement[6] permettent d'augmenter rapidement la productivité dans l'IDE de manière interactive.

[5] https://plugins.jetbrains.com/plugin/8554-ide-features-trainer/
[6] https://www.youtube.com/user/intellijideavideo

6. Utilisation du *debugger*

Un simple clic dans la partie à gauche du code permet d'ajouter un point d'arrêt. Pour lancer l'exécution de l'application en mode *debug*, il existe plusieurs possibilités :

- Bouton « Debug » dans l'interface utilisateur
- Raccourci clavier « CTRL + D »
- Menu de l'application « Run » : « Debug 'Main' »
- Clic droit sur la classe Main dans l'onglet « Projet » ou dans l'éditeur de la classe « Main »



A tester :

- Placer un point d'arrêt dans le fichier « Main.java », sur la première ligne **`myCar.start() ;`**
- Lancer l'exécution en mode *debug.* L'exécution débute et l'interface de IntelliJ se modifie pour afficher également l'onglet « Threads & Variables », ainsi que les boutons de navigation du mode « Debug ». Ce menu permet également l'affichage d'autres onglets utiles, comme l'état de la « Mémoire ».
- Lorsque l'exécution arrive au point d'arrêt défini, effectuer un pas à pas (bouton de navigation de l'onglet « Debug » : « Pas à pas » ou F7) pour entrer dans la méthode (fonction) **`start()`**.

Consulter les onglets spécifiques au mode *debug,* puis poursuivre pas à pas en observant les changements dans chaque onglet. Essayer également les commandes de *debug* « Sortir » (remonte la chaine d'appel) et « Step Over Expression » (Passe à l'expression suivante).

7. Utilisation du Profiler

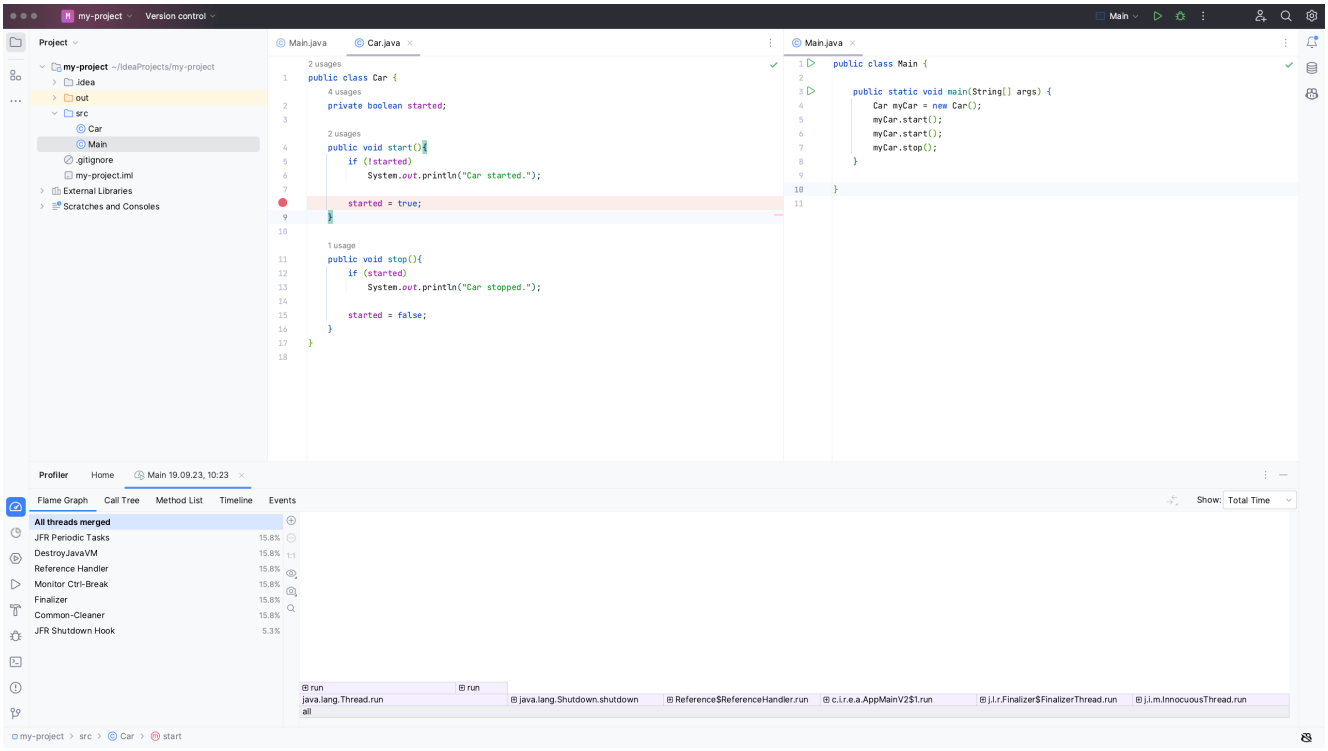
Premature optimization is the root of all evil – Donald Knuth

L'optimisation d'un programme est souvent contre-intuitive. En d'autres termes, ce n'est pas toujours ce qui semble prendre du temps lors de l'écriture d'un programme qui en prendra lors

de son exécution. Pour cette raison, il est souvent préférable d'implémenter la solution naïve d'un problème, puis de l'optimiser si cela s'avère nécessaire. Le profiler (bouton situé dans la liste déroulante « trois petit points » sur la droite du bouton d'exécution) permet d'examiner l'exécution d'un programme et d'identifier les parties qui doivent être optimisées.

A tester :

- Exécuter le programme à l'aide du Profiler
- Dans l'onglet « Profiler », examiner les différentes vues à disposition (Flame Graph, etc.)
- Sur l'ensemble de l'exécution, identifier la proportion de temps prise pour l'exécution de la méthode « Main.main »

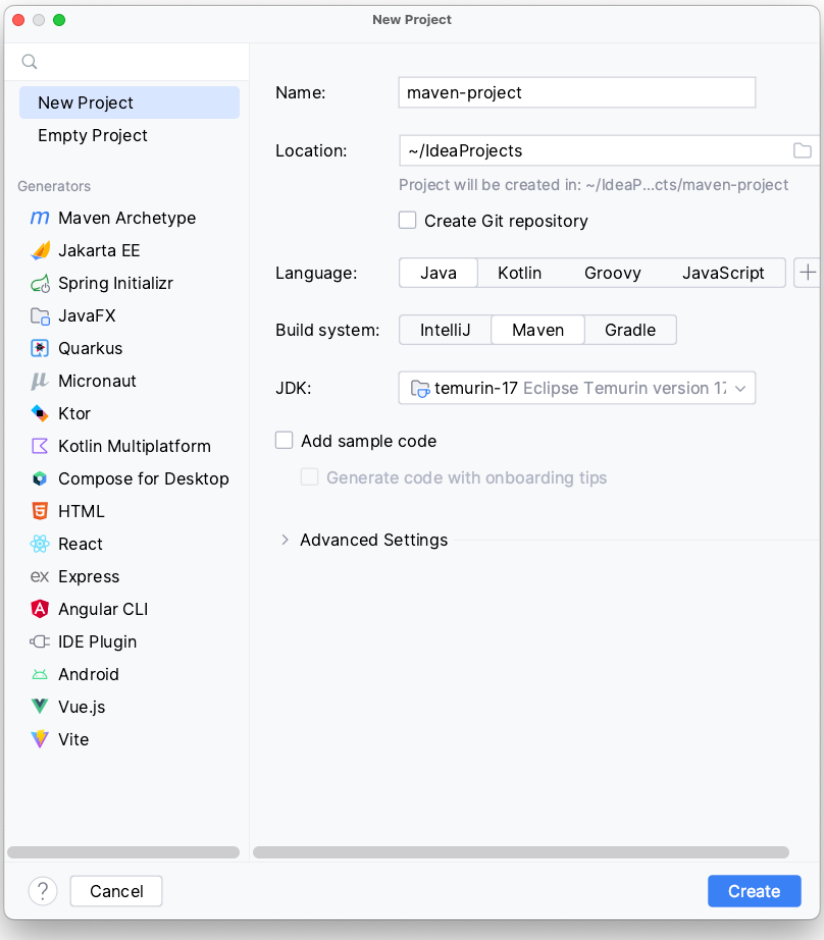


8. Utilisation d'un Build Tool (Maven)

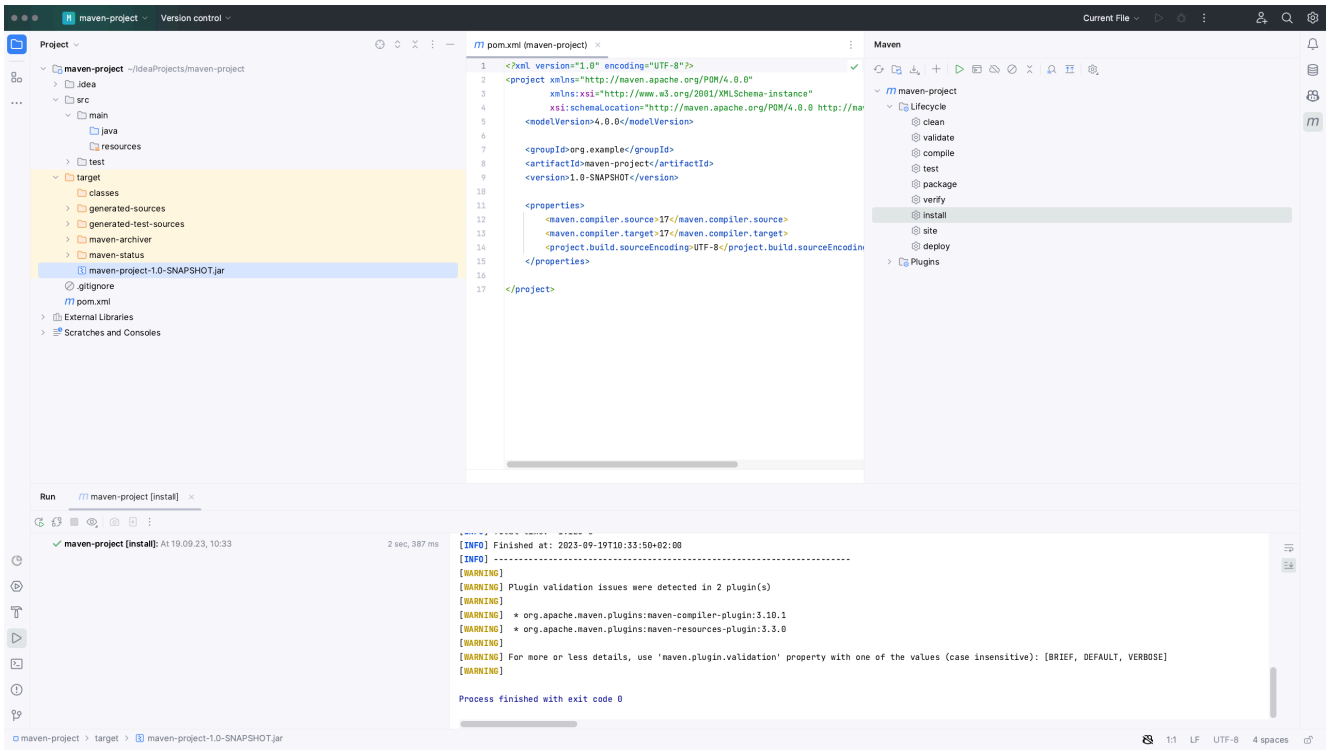
La plupart des langages récents sont distribués avec leur propre build tool (p. ex. Cargo pour Rust). Lorsque Java a été conçu, cette tendance n'existait pas, ce qui a favorisé l'émergence de plusieurs build tools différent dont Maven que nous utiliserons dans le cadre de ce cours.

Les rôles de Maven sont multiples. Maven permet d'automatiser le télécharger des dépendances sur internet, de compiler le projet, d'exécuter des tests unitaires et d'intégration, d'assembler les classes compilées dans un paquet, de publier la documentation du projet, ou encore de publier la distribution d'une version majeure.

Pour créer un projet Java en utilisant Maven comme build tool, il suffit de créer un nouveau projet, de sélectionner Maven comme Build system.



Le nouveau projet contient un fichier pom.xml (abréviation de project objet model). C'est le fichier qui permet de configurer les actions effectuées par le build tool. L'interface d'IntelliJ affiche maintenant un onglet « Maven » qui liste les actions qui peuvent être exécutées par le build tool (clean, validate, compile, etc.). Par exemple, l'action « Install » crée une distribution du projet (fichier jar) dans le dossier « target » du projet.



A tester :

L'Objectif est d'utiliser Maven pour exécuter des tests unitaires.

Modifier le fichier pom.xml comme suit de manière à spécifier la version de Java et ajouter une dépendance à JUnit5 (librairie de tests unitaires), puis cliquer sur le bouton de rafraichissement de la configuration Maven (pastille en haut à droite de l'éditeur ou bouton dans l'onglet « Maven »).

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>maven-project</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>5.10.0</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

Ajouter une classe Calculator dans le dossier src/main/java du projet.

```
public class Calculator {

  public int add(int a, int b) {
    return a + b;
  }

}
```

Ajouter une classe CalculatorTest dans le dossier src/test/java du projet.

```
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

class CalculatorTest {

  @Test
  void add() {
    assertEquals(new Calculator().add(1, 1), 2);
  }

}
```

Exécuter l'action « test » dans l'onglet Maven. Cette dernière se termine avec succès que si l'assertion présente dans la classe CalculatorTest est vraie. A noter qu'IntelliJ permet également

d'exécuter les tests de manière indépendante depuis l'éditeur (clic droit) ou l'onglet « Projet ». Dans ce cas, un onglet de test apparaît au bas de l'écran.