

Travaux Dirigés

Java Avancé

—M1—







Apache Maven, Eclipse

Attention : Les réponses aux questions qui ne donnent pas lieu à l'écriture de code doivent se trouver dans un fichier `REPONSES.txt`, à la racine du répertoire `td00` (voir ci-après).

► Exercice 1. Introduction à Maven

Maven est un outil pour la gestion et la diffusion de code source en Java. Il permet de compiler les sources d'un programme java sans dépendre du système d'exploitation ou de l'IDE qui ont été utilisés pour développer le programme. Il s'agit d'un outil incontournable dans l'écosystème Java, et c'est celui nous utiliserons au cours de cette année. L'objectif premier de ce TD est de vous familiariser avec Maven.

Maven ainsi que d'autres outils dont nous nous servirons pendant ce cours s'utilisent en ligne de commande. C'est pourquoi nous profiterons également de ce TD nous familiariser avec le terminal et les outils en ligne de commande, ils vous seront très utiles pour la suite de ce cours.

1. Ouvrez un terminal et tapez `cd`  pour vous rendre dans votre répertoire `home` si vous n'y êtes pas déjà.
2. Créez un nouveau répertoire `javaavance` avec la commande `mkdir javaavance` . Tapez la commande `ls`  pour lister le contenu du répertoire dans lequel vous vous trouvez et vérifiez que votre répertoire y a bien été créé.
3. Tapez la commande `cd javaavance`  pour vous déplacer dans le répertoire que vous venez de créer. Utilisez la commande `nano REPONSES.txt`  pour lancer un éditeur et éditez le (nouveau) fichier `REPONSES.txt`. Les fichiers `RÉPONSES` devront contenir les réponses aux questions du TD. Pour l'instant, inscrivez simplement votre nom, sauvegardez le fichier avec `ctrl` + `O` et quittez l'éditeur avec `ctrl` + `X`.
4. On utilise Maven en invoquant la commande `mvn` avec différents arguments. Par exemple, vous pouvez obtenir de l'aide sur `mvn` en tapant la commande `mvn --help` . En utilisant `mvn --help`, trouvez la commande qui vous permettra de connaître la version de Maven. Quelle version de Maven est installée sur la machine que vous utilisez ? Assurez vous qu'il s'agit d'une version suffisamment récente (i.e. ≥ 3.0).

5. Maven permet de créer des nouveaux projets Java en se basant sur le concept d'*archétypes* (*archetype* en anglais). Les archétypes sont des modèles standards pour différentes classes de projets (application en ligne de commande, application web etc.).

Nous allons créer un premier projet pour ce TD. Pour créer un nouveau projet, il faut demander à Maven de générer un nouveau projet en utilisant la commande `mvn archetype:generate` et en spécifiant l'archétype `maven-archetype-quickstart` qui permet de créer des applications simples. Vous pouvez en savoir plus sur les archétypes dans Maven en lisant la documentation disponible [ici](#).

Entrez la commande suivante,

```
mvn archetype:generate -D archetypeArtifactId=maven-archetype-quickstart
```

Et renseignez les différents champs :

- groupId : `fr.dauphine.ja.nomprenom.td00`
- artifactId : `td00`
- version : `1.0-SNAPSHOT`
- package : `fr.dauphine.ja.nomprenom.td00`

Attention : à bien remplacer *nomprenom* par votre nom et votre prénom !

Aidez vous de la documentation de Maven pour trouver le sens des différents champs. À quoi servent les paramètres `groupId` et `artifactId` ?


Remarque :

L'option `-D` permet de spécifier en champ directement dans la ligne de commande, comme ceci :

```
mvn archetype:generate \  
-D archetypeArtifactId=maven-archetype-quickstart \  
-D groupId=fr.dauphine.ja.nomprenom.td00 \  
-D artifactId=td00 \  
-D interactiveMode=false
```

À l'avenir, servez vous de cette commande pour créer les prochains tds.

Attention : Les caractères d'échappement `'\'` qui précèdent le retour à la ligne servent à faire en sorte que le copié-coller fonctionne correctement sur plusieurs ligne. Vous ne devez pas les taper si vous entrez la commande sur une seule ligne.

6. Déplacez vous dans le nouveau répertoire `td00`, et compilez le projet tel quel, en tapant `mvn compile` . Si l'opération se déroule correctement, vous trouverez un nouveau répertoire `target` au même niveau que le répertoire `src`, que contient ce répertoire ?

7. Exécutez la classe `App` de votre projet avec la commande :
`java -cp target/classes fr.dauphine.ja.nomprenom.td00.App`. Le programme par défaut affiche « Hello World ! », trouvez le fichier source responsable de cet affichage et modifiez le pour que le programme affiche « Bonjour ! » plutôt que « Hello World ! »
8. Dans la commande précédente, l'argument `-cp` permet de spécifier un *class path*. Qu'est ce que le class path, à quoi sert-il ?
9. Déplacez vous dans le nouveau répertoire `td00` et ouvrez le fichier `pom.xml`. Ce fichier constitue la description de votre projet, sous la forme d'un arbre. Repérez la racine de l'arbre et la structure de l'arbre, et familiarisez vous avec la syntaxe XML.
10. Lors de la compilation, vous avez peut être remarqué un message `WARNING`. Ce message signale que le schéma d'encodage Unicode des fichiers sources n'est pas spécifié explicitement. Nous allons ajouter une première propriété pour résoudre ce problème.
 Pour ajouter une première propriété, ajoutez la balise ouvrante `<properties>` et la balise fermante `</properties>` en dessous du nœud `project` (la position exacte dans le fichier n'a pas d'importance).
11. Ajoutez la ligne suivante entre les deux balises `properties`
`<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>`
 Recompilez votre projet et vérifiez que le `WARNING` a disparu.
12. Ajoutez une nouvelle propriété `maven.compiler.source` entre les deux balises `properties`, et donnez lui la valeur `1.7`. Cette nouvelle propriété permet de spécifier la version du compilateur Java utilisée pour compiler les sources.
13. Ajoutez également la propriété `maven.compiler.target`. À votre avis, à quoi sert cette propriété, quelle valeur doit-on lui donner ?

Recompilez votre projet et vérifiez que tout fonctionne comme prévu.

► Exercice 2. Archives JAR et Maven JAR Plugin

Les fichiers JAR (*Java ARchive*) sont des archives zip utilisées pour diffuser des programmes ou des sources java. Si elles sont configurées correctement, les archives JAR peuvent être directement exécutées par la machine virtuelle java au moyen de l'option `-jar`. Dans cet exercice, nous allons utiliser le plugin *Maven JAR Plugin* pour automatiser la construction d'archives JAR exécutables.

1. Retournez à la racine de votre projet, et utilisez la commande `mvn package` pour compiler vos sources et créer une première archive JAR. Vérifiez qu'une archive JAR se trouve dans le répertoire `target` et exécutez votre code avec la commande :

```
java -cp target/nom-du-jar \
    fr.dauphine.ja.nomprenom.td00.App
```

Tentez également d'exécuter votre fichier JAR avec la commande :

```
java -jar target/nom-du-jar
```

que se passe t'il ?

2. Pour que le fichier JAR soit exécutable, il faut que l'archive contienne un fichier MANIFEST avec la ligne

```
Main-Class: fr.dauphine.ja.nomprenom.td00.App
```

Ça n'est pas le cas pour le moment. Vous pouvez le faire à la main, mais pour nous faciliter la tâche, nous allons configurer utiliser Maven JAR Plugin pour que cela soit fait automatiquement lors de la création de l'archive.

3. Éditez le fichier `pom.xml`, et ajoutez à la fin du fichier — avant la balise fermante `</project>` — les lignes suivantes.

```
<build>
  <plugins>
    <plugin>
      <artifactId>maven-jar-plugin</artifactId>
      <version>3.0.2</version>
      <configuration>
        <archive>
          <manifest>
            <mainClass>fr.dauphine.ja.nomprenom.td00.App</mainClass>
            <addClasspath>true</addClasspath>
            <classpathPrefix>lib</classpathPrefix>
          </manifest>
        </archive>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Remarquez la balise `mainClass` du sous arbre `archive/manifest`, qui permet de spécifier le nom de la classe à exécuter, et remplacez *nomprenom*.

4. Relancez la construction de votre JAR et exécutez le avec :

```
java -jar target/nom-du-jar
```

► Exercice 3. Maven et Eclipse

Eclipse est une IDE qui dispose de nombreuses fonctionnalités pour faciliter le développement en Java, c'est l'IDE que nous allons utiliser pendant ce cours.

Attention : Maven et Eclipse utilisent des fichiers de configurations différents (le fichier `pom.xml` pour Maven et le fichier `.project` pour Eclipse). Il se peut donc que votre projet fonctionne dans votre version d'Eclipse, mais ne puisse pas être compilé par ceux qui souhaitent utiliser Maven. À tout moment, vous devez faire en sorte que le fichier `pom.xml` soit à jour et qu'il soit possible de créer un fichier JAR fonctionnel avec la commande `mvn package`, sans passer par Eclipse.

1. Démarrez Éclipse.

Attention : les salles du Crio disposent de deux versions d'Eclipse, assurez vous qu'il s'agit de la version la plus récente ($\geq 4.5.0$) disposant du plugin M2E (Maven to Eclipse) qui permet d'importer des projets Maven dans Eclipse.

2. Importez votre projet en cliquant sur **File/Import** puis **Maven** et **Existing Maven Projects**. Sélectionnez le répertoire qui contient votre projet `td00` et finalisez l'import.
3. Cliquez sur le package `fr.dauphine.ja.nomprenom.td00` dans le menu de gauche et ouvrez le fichier `App.java` pour éditer la méthode `main`. Que se passe-t-il si l'on tape `sysout` et que l'on appuie sur `Ctrl` + `Espace` dans un `main` ?
4. Meme question en tapant `toStr` puis `Ctrl` + `Espace` dans une classe ?
5. Même question en tapant `main` puis `Ctrl` + `Espace` dans une classe ?
6. Créer un champ `toto` de type `int` dans la classe. Que se passe-t-il si l'on tape `get` puis `Ctrl` + `Espace` dans la classe ? Et `set` puis `Ctrl` + `Espace` ?
7. Sélectionner le nom de la classe. Que se passe-t-il si l'on tape `Alt` + `Shift` + `R` ? Meme question avec le champ `toto`.
8. (à la maison) Il peut être utile de voir le code source utilisé par la JDK. Pour cela, télécharger le fichier `src.zip` sur le site d'oracle (sur les machines du CRIO c'est déjà dans `/usr/local/jdk***/src.zip`) et attacher-le dans **Window/Preferences/Java/Installed JREs** puis **Edit**, cliquez sur `rt.jar` et **Source Attachment**. Déclarer une variable de type `String` et cliquer sur `String` en maintenant la touche `Ctrl`. Que se passe-t-il ?

Vous savez maintenant éditer vos projet Maven dans Eclipse ! N'oubliez pas de vérifier que vos projets continuent de fonctionner en dehors d'Eclipse ...

► Exercice 4. Recherche de nombres premiers

Pour finir, nous allons modifier le projet, et en faire une application qui permet de découvrir des nombres premiers dans une collection d'entiers générés aléatoirement. Cet exercice constitue un prérequis pour le cours de Java Avancé, si vous ne parvenez pas à le faire parlez en à votre chargé de TD.

Rappel : pour tester si un nombre p est premier, une méthode simple consiste à tester tous ses diviseurs potentiels entre 2 et \sqrt{p} . Le nombre p est premier si et seulement s'il n'existe aucun $x \in \{2 \dots \sqrt{p}\}$ tel que le *reste* de la division $\frac{p}{x}$ est nul.

1. Renommez (*refactorez*) la classe `App` en une classe `PrimeCollection` et créez une variable membre `numbers` du type `java.util.ArrayList<Integer>`. Ajoutez un constructeur par défaut pour instancier `numbers` correctement.
2. Créez une nouvelle méthode `initRandom(n, m)` pour insérer dans la collection `numbers`, `n` entiers tirés aléatoirement entre 0 et `m`. (Pour générer des nombres aléatoires, utiliser `java.util.Random`).

3. Ajoutez une méthode privée `isPrime(p)` qui retourne `true` si l'entier `p` passé en paramètre est premier ou `false` sinon.
4. Ajoutez une méthode `printPrimes()` qui affiche tous les entiers premiers dans la collection `numbers`.
5. Utilisez les méthodes précédentes pour faire en sorte que la méthode `main` génère 100 entiers tirés aléatoirement entre 1 et 1000, et affiche ceux d'entre eux qui sont premiers.
6. Mettez à jour le fichier de configuration Maven pour que la classe `PrimeCollection` soit appelée par défaut. La commande `mvn package` doit générer un JAR exécutable exécutant la fonction `main` de la classe `PrimeCollection`.

► N'oubliez pas d'écrire les réponses aux questions de ce TD dans le fichier `RE-PONSES.txt`