Aix*Marseille université
Initiative d'excellence
Année 2018-19

Domaine Sciences et Technologies
Licence informatique

Programmation 2 : TP 1
Code UE : SIN3U02

Introduction à la programmation objet

1 Compilation et exécution en Java

1.1 Explications sur l'exécution de code Java

La portabilité sur différentes plates-formes était un objectif important lors de la conception du langage Java. Afin d'atteindre cet objectif, le code Java n'est pas directement compilé en un exécutable spécifique pour chaque architecture ou système comme cela peut être le cas pour le langage C. À la place, le code java est compilé dans un fichier exécutable en **bytecode java** par une machine spécifique à Java. Cette machine virtuelle est appelé **Java Virtual Machine** (JVM) et des versions existent pour quasiment toutes les plates-formes existantes. L'avantage est que contrairement à des langages directement compilés en exécutable, il suffit de compiler une seule fois le code et pas une fois par cible souhaitée.

Pour exécuter du code java en ligne de commande, on doit donc le compiler en bytecode Java à l'aide de la commande `javac` puis l'exécuter à l'aide de la commande `java`.

1.2 But de l'exercice

Comprendre à très haut niveau le processus d'exécution et de compilation de code Java.

1.3 Déroulé de l'exercice

Vous devez réaliser les tâches suivantes :

1. Créez un dossier **Programmation2**, un dossier **TP_1** contenu dans **Programmation2** et un dossier **Exercice1** contenu dans **TP_1**. Vous pouvez utiliser la commande **mkdir** pour créer des dossiers ou bien le gestionnaire de fichiers de votre système.
2. Téléchargez le fichier HelloWorld.java et le mettez-le dans le dossier **Exercice1**.
3. Ouvrer un terminal et placez-vous dans le dossier **Exercice1** (rappel : on se déplace dans les dossiers grâce à la commande `cd`). Vérifiez que vous êtes bien dans le bon dossier en exécutant la commande `pwd`.
4. Vérifiez que le dossier contient le fichier **HelloWorld.java** en listant les fichiers à l'aide de la commande `ls`
5. Compiler ce fichier à l'aide de la commande `javac` en exécutant `javac HelloWorld.java`.
6. Lancez à nouveau la commande `ls`. Que remarquez-vous ?
7. Exécuter le bytecode Java en lançant la commande `java HelloWorld`. Que se passe-t-il ?

2 Manipulation de listes

2.1 Interpréteur Java JShell

Dans cet exercice, on va manipuler des listes et afin d'éviter de repasser pour chaque test par le processus de compilation, on va utiliser l'interpréteur jshell inclus avec Java de puis sa version 9. Jshell se lance dans le terminal en tapant `jshell`. En salle de TP à Aix, `jshell` se lance avec `/usr/java/jdk-10.0.2/bin/jshell`. Un fois lancé, il est possible d'exécuter des instructions ou de définir des variables et des méthodes tout simplement en écrivant le code correspondant à l'intérieur de la console. Il est aussi possible de vérifier le contenu des variables en tapant leur nom. Pour sortir de `jshell`, il suffit de taper `/exit`.

2.2 Les listes en Java

En Java, il est possible de créer une liste d'entiers grâce à l'instruction suivante [1] :

```
List<Integer> l = new ArrayList<Integer>();
```

Les listes en Java sont des collections d'objets ordonnés. Nous allons seulement utiliser quelques méthodes de cette classe. Pour plus de détails sur les listes, nous vous conseillons (comme dans la plupart des cas pour Java) de vous référer à la documentation officielle de `List`.

Pour cet exercice, nous allons seulement considérer des listes d'`Integer` [2]. Les `Integer` est une classe encapsulant le type primitif `int`. Cela permet de manipuler des entiers en tant qu'objet car les `int` ne sont pas des objets en Java.

Dans cet exercice, on va utiliser les méthodes suivantes de `List` :

— `add(Integer e)` : ajoute l'entier `e` à la fin de la liste

— `equals(Object o)` : compare la liste avec l'objet `o` et renvoie `true` si `o` est une liste contenant les éléments de la liste dans le même ordre.

2.3 But de l'exercice

— Comprendre l'utilisation des méthodes de `List` et d'`Object`.

— comprendre le concept de référence

2.4 Déroulé de l'exercice

Vous devez réaliser les tâches suivantes :

1. lancez la commande `jshell` dans un terminal.
2. Définissez et créez une liste `l1` en tapant l'instruction `List<Integer> l1 = new ArrayList<Integer>();`.
3. Vérifiez que la liste est bien vide en tapant `l1`.
4. Ajoutez 1 à la fin de la liste `l1` à l'aide de la méthode `add`.
5. Vérifiez que la liste `l1` contient bien 1 en tapant `l1`.
6. Ajoutez 2 à la fin de la liste `l1` à l'aide de la méthode `add`.
7. Vérifiez que la liste `l1` contient bien 1 et 2 en tapant `l1`.
8. Définissez une liste `l2` en tapant l'instruction `List<Integer> l2 = l1`.
9. Quels sont les entiers contenus dans `l2` ? Pourquoi `l2` contient ces entiers ?
10. Ajoutez 3 à la fin de la liste `l1` à l'aide de la méthode `add`.
11. Vérifiez le contenu de la liste `l2`. Est-ce qu'il a changé ?
12. Définissez et créez une liste `l3` en tapant l'instruction `List<Integer> l3 = new ArrayList<Integer>();`.
13. Ajoutez 1 puis 2 puis 3 à la fin de la liste `l3` à l'aide de la méthode `add`.
14. Vérifiez que la liste `l3` contient bien 1, 2 et 3 en tapant `l3`.
15. Comparer les listes `l1` et `l2` à l'aide de l'opérateur `==` en tapant `l1 == l2`.
16. Faites de même pour comparer `l1` et `l3` ainsi que `l2` et `l3`.
17. Comparer les listes `l1` et `l2` à l'aide de la méthode `equals` en tapant `l1.equals(l2)`.

1. Depuis Java 10, il est possible d'utiliser la ligne suivante `var l = new ArrayList<Integer>();` à la place et ce uniquement pour les variables locales.
2. Nous verrons plus en détails les listes dans un chapitre ultérieur du cours.

18. Faites de même pour comparer `l1` et `l3` ainsi que `l2` et `l3`.
19. Comment expliquez-vous le résultat des comparaisons ?
20. Arréter `jshell` en tapant `/exit`.

3 Student

3.1 But de l'exercice

Connaître la syntaxe permettant de définir une nouvelle classe en Java.

3.2 Environnement de développement (IDE)

Afin de programmer dans ce cours, nous allons utiliser un environnement de développement (Integrated Development Environment : IDE). Il existe de nombreux IDE pour Java. Dan ce cours, nous vous conseillons d'utiliser IntelliJ IDEA de chez **JetBrains** mais vous pouvez aussi utiliser Eclipse ou un autre IDE.

3.2.1 Lancement de l'IDE

Allez dans le Menu 'Application' (en haut à gauche de l'écran) et ouvrir 'IntelliJ IDEA' dans la section programmation.

Si vous avez installé IntelliJ sur votre machine :

Après le chargement, vous pouvez tomber sur une première fenêtre vous proposant d'importer vos paramètres. Il faudra dans l'ordre :

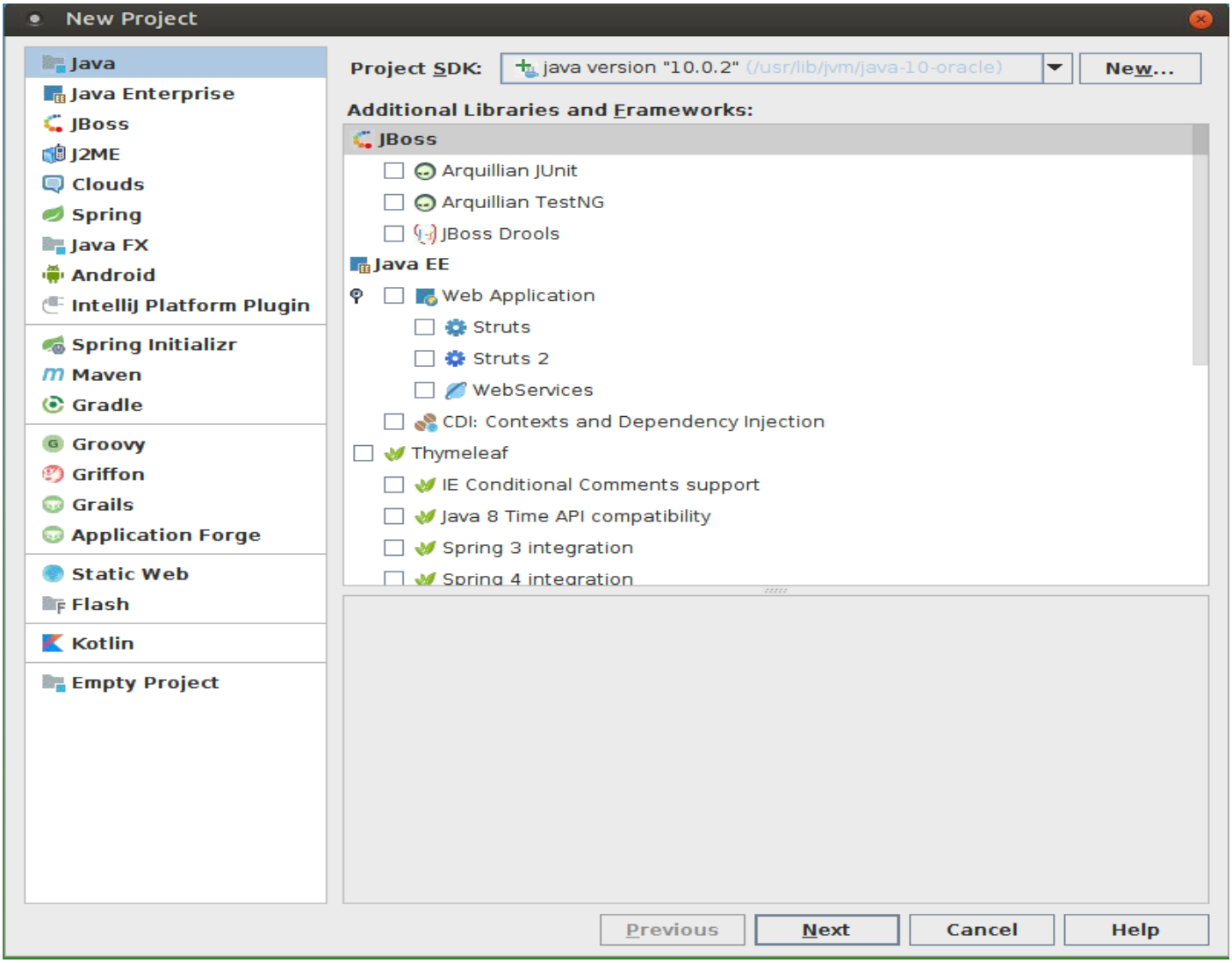
1. Laissez sur 'Do not import settings' et cliquez sur 'OK'. À partir de là vous allez personnaliser votre installation pour qu'elle corresponde à vos besoins.
2. Sur la première fenêtre de personnalisation, vous devez choisir l'apparence de votre IDE. Ici rien de fondamental, vous pouvez choisir l'option que vous voulez entre `Default` (texte foncé sur fond blanc) ou `darcula` (texte clair sur fond noir).
3. Les deux écrans suivants sont plutôt destinés à une installation sur une machine personnelle, vous pouvez laisser les options par défaut et passer à la suite.
4. Les deux écrans suivants vous permettent de choisir les plugins que vous allez activer. De manière générale, il vaut mieux en activer le moins possible pour éviter les mauvaises surprises.

3.2.2 Création de projet

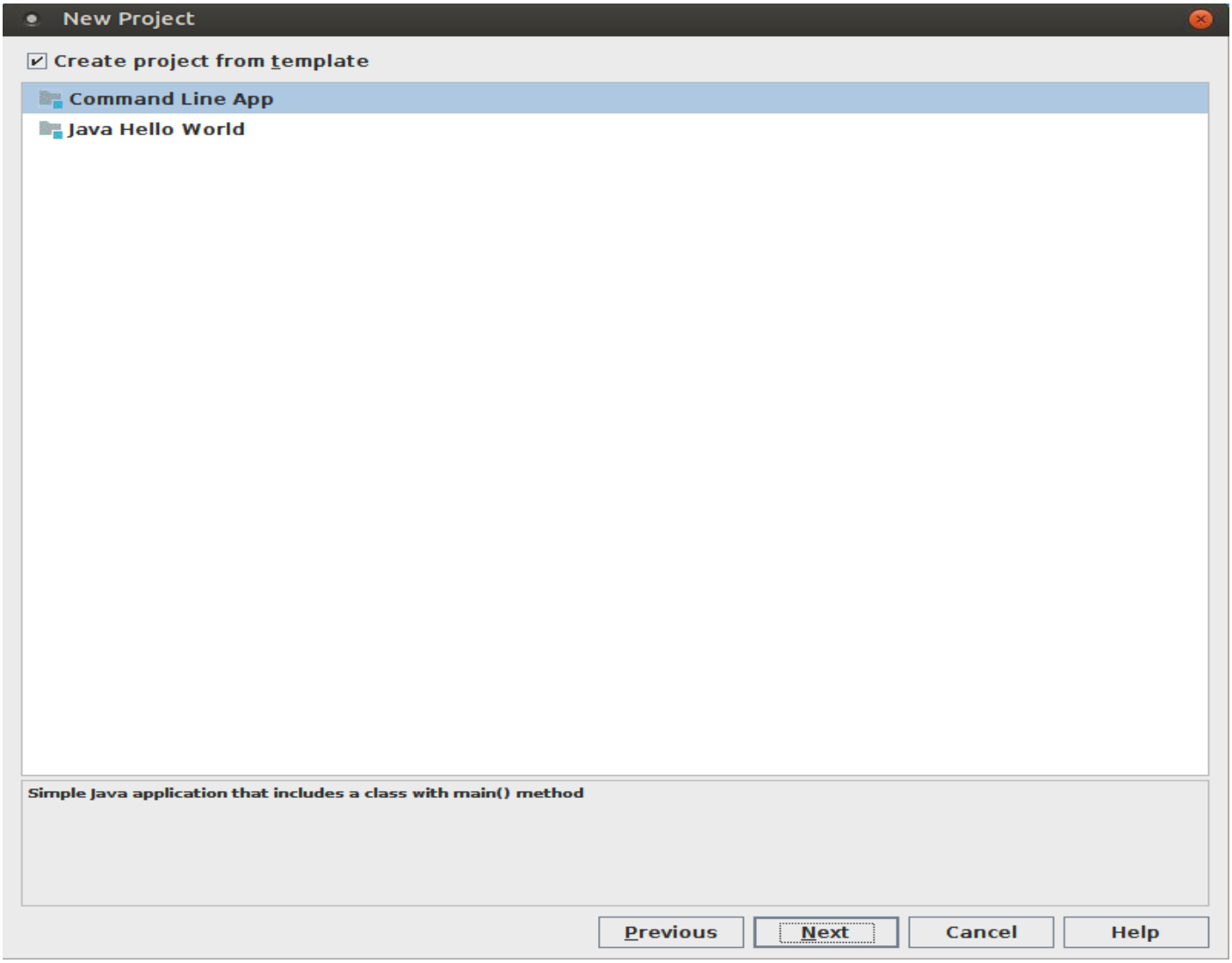
Ensuite vous allez arriver sur la fenêtre d'accueil d'IntelliJ IDEA (vois l'image ci-dessous).

Pour le premier TP, nous allons créer un projet pour la gestion d'inscription d'étudiant. Vous allez donc créer un projet nommé `student_management`. Pour cela, il vous suffit de cliquer sur `Create New Project` sur la fenêtre suivante :

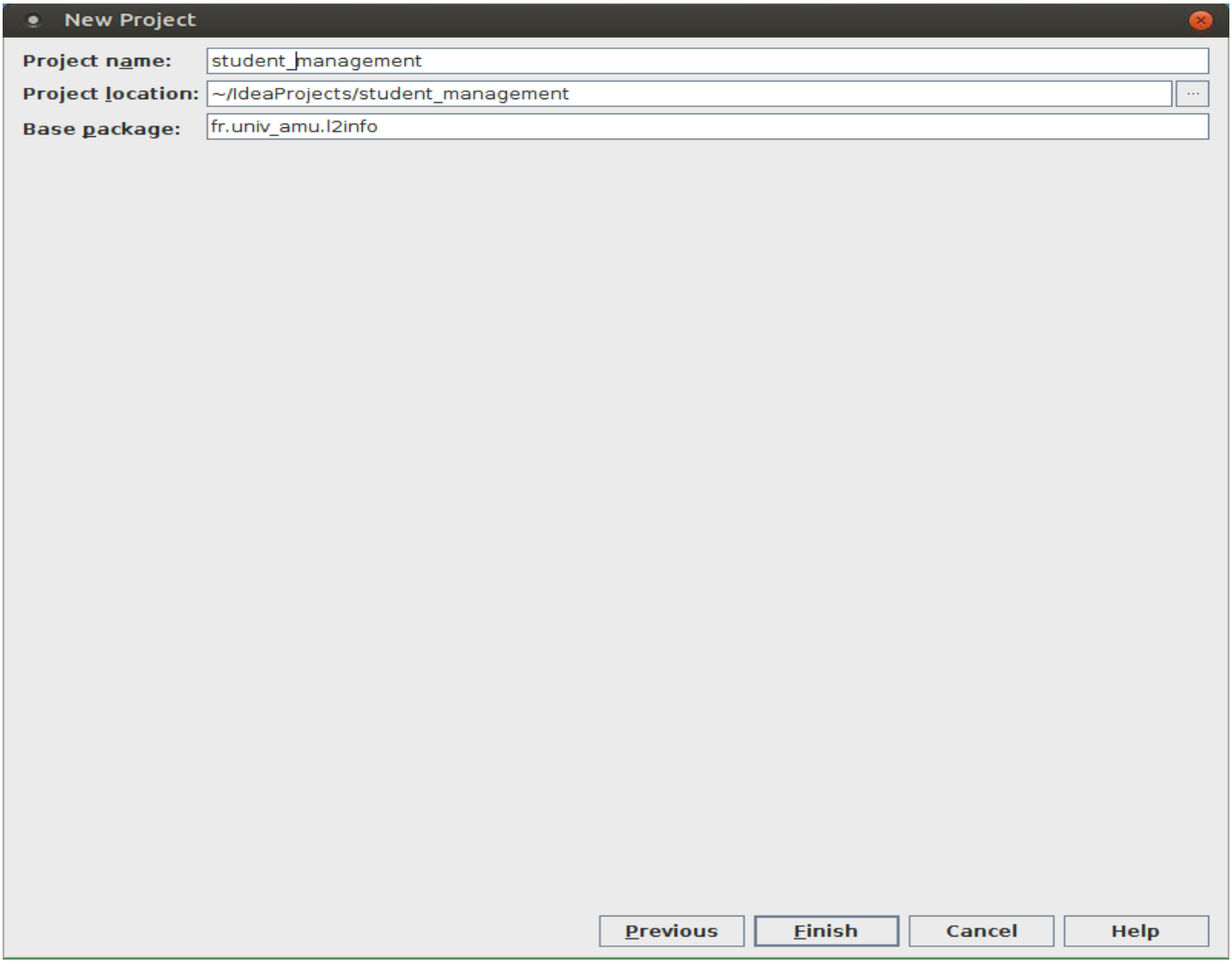
Vous allez devoir choisir la version de java ainsi que le moteur de production. Pour le moment, on va laisser les paramètre de base (Java 10.0.2 et Java). Vous pouvez donc cliquer sur `next` à la fenêtre suivante :



Vous devez maintenant choisir si vous souhaitez créer le projet à partir d'un modèle (`template`). Pour ce premier TP, nous allons utiliser le modèle pour les applications en ligne de commande. Vous devez donc cochez la case `Create project from template` et choisir `Command Line App`.

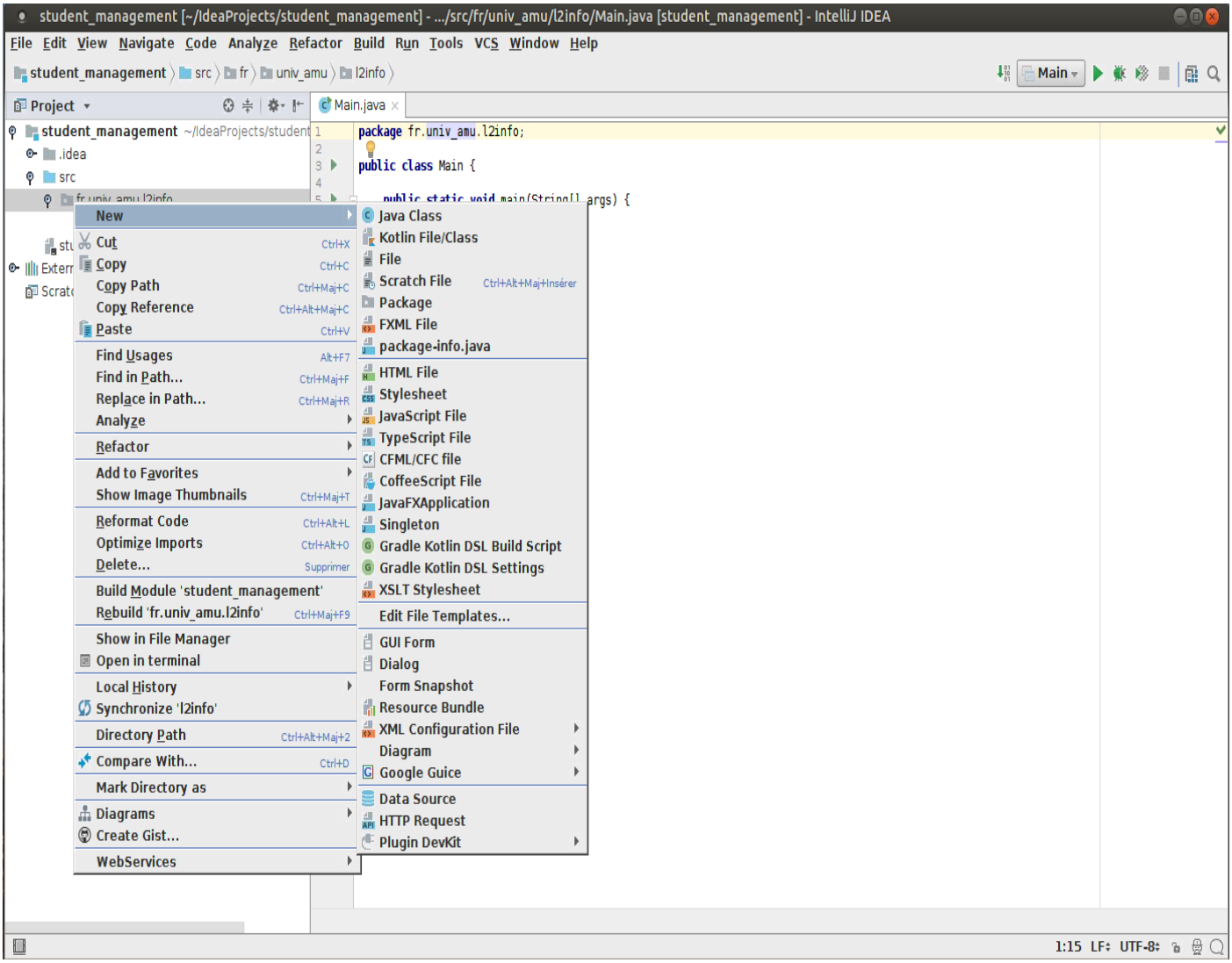


Vous devez maintenant choisir le nom du projet ainsi que le nom du package. Le projet devra s'appeler `student_management` et le package `fr.univ_amu.l2info`.



3.3 La classe Student

Maintenant que le projet est créé, vous allez créer une premiere classe Java. Pour cela, placez-vous sur le répertoire `student_management` → `src` → `fr.univ_amu.l2info` dans la partie gauche de l'IDE et cliquez sur le bouton droit de votre souris. Un menu s'ouvrira. Choisissez `New` puis `Java class` pour créer une classe nommée `Student`.



3.3.1 Attributs

Vous devez créer une classe `Student` ayant les attributs suivants :

— un numéro d'identification appelé `id` de type `int`,

— le prénom `firstName` de type `String`,

— le nom `lastName` de type `String`.

Pour chacun de ces attributs, choisissez s'il doit être public ou privé, mutable ou immutable.

— **privée (mot-clé private) :** la valeur de l'attribut ne peut être accédée ou modifiée qu'à l'intérieur de la classe

— **publique (mot-clé public) :** la valeur de l'attribut peut être accédée ou modifiée à l'extérieur de la classe

— **Mutable :** la valeur de la propriété peut changer au cours du temps

— **Immutable :** la valeur de la propriété ne peut pas être modifiée après la construction de l'objet

Rajoutez le code correspondant dans le fichier `Student.java` que vous venez de créer.

3.3.2 Constructeur

Vous devez maintenant ajouter un constructeur à la classe `Student`. Quels arguments devraient avoir ce constructeur selon vous ?

3.3.3 Méthode `toString`

Vous devez maintenant ajouter une méthode `String toString()` à la classe `Student`. Cette méthode devra renvoyer une chaîne de caractères au format suivant : "Paul Calcul id : 12".

3.3.4 Vérifier que la classe est correcte

Afin de vérifier que vous avez bien codé la classe, vous devez remplacer le code de la classe main par le code suivant :

```
public class Main {

    public static void main(String[] args) {
        final int NB_STUDENTS = 3;
        final String[] firstNames = {"Pierre", "Marie", "Paul"};
        final String[] lastNames = {"Martin", "Dubois", "Calcul"};
        for(int index = 0; index < NB_STUDENTS; index ++){
            System.out.println(new Student(index, firstNames[index], lastNames[index]));
        }
    }
}
```

Vous devriez obtenir le résultat suivant à l'exécution (triangle en haut à droite de l'écran) :

```
Pierre Martin id: 0
Marie Dubois id: 1
Paul Calcul id: 2
```

3.4 Déroulé de l'exercice

Vous devez réaliser les tâches suivantes :

1. Lancer un IDE, IntelliJ IDEA de préférence.
2. Créer un projet `student_management` dans un package `fr.univ_amu.l2info`.
3. Créer un fichier `Student.java` dans ce projet.
4. Ajouter les attributs : `id`, `firstName` et `lastName` à la classe `Student`.
5. Ajouter un constructeur à la classe `Student`.
6. Ajouter une méthode `toString` à la classe `Student`.
7. Vérifier que votre code est correct.

4 Lectures simples au clavier : la classe Scanner

Le but de cet exercice est de vous initier à l'utilisation de la classe `java.util.Scanner`, qui permet d'effectuer simplement des lectures à la console. La documentation de cette classe est disponible au lien suivant : https ://docs.oracle.com/javase/10/docs/api/java/util/Scanner.html.

Cette classe s'emploie de la manière suivante : au début du programme, on crée une instance (nommée ici `input`) connectée à un flot en entrée, qui très souvent est l'entrée standard `System.in` :

```
Scanner input = new Scanner(System.in);
```

Ensuite, pour lire un entier `integerValue`, il suffit d'écrire :

```
int integerValue = input.nextInt();
```

Pour lire un double `doubleValue`, il suffit d'écrire :

```
double doubleValue = input.nextDouble();
```

Pour lire une chaîne de caractères `string`, il suffit d'écrire :

```
String string = input.nextLine();
```

La chaîne renvoyée comme résultat par `nextLine` sera formée de tous les caractères qu'on va trouver sur `System.in` à partir de maintenant et jusqu'au prochain caractère de fin de ligne (on produit un caractère fin de ligne au clavier en pressant la touche « Entrée »). Le caractère fin de ligne lui-même sera lu, mais n'apparaîtra pas dans la chaîne résultat.

Pour essayer la classe Scanner, écrivez en Java le programme qui calcule et affiche la somme d'une suite de nombres décimaux strictement positifs, lus au clavier, dont la fin est indiquée par un nombre égal à 0.

5 Exercices supplémentaires

— Coder les classes `Article`, `Point`, `LineSegment`, `BankAccount` et `Bottle` des feuilles de TD.
— Analyser le code de l'algorithme de parcoursup. Relever tous les problèmes dans le code dans la classe GroupeClassement.