amU Faculté des sciences Aix Marseille Université

Année 2024-25

Domaine Sciences et Technologies

Licence Informatique : math-info

Initiation au Génie logiciel : TP 1

Codes UE : SIN5U34

M3C

1 Introduction

Le but de ce TD est de concevoir une architecture logicielle permettant de gérer les règles d'attribution du diplôme de licence à l'université d'Aix-Marseille définie par les Modalités de Contrôle des Connaissances et des Compétences (M3C) en licence. Le but du logiciel sera de permettre le stockage des notes ainsi que la décision de validation ou non des différents éléments de la licence par les étudiants.

L'objectif pédagogique du projet est de vous faire travailler sur les outils et les bonnes pratiques pour le développement logiciel :

— la gestion de version ;
— l'utilisation de moteur de production ;
— l'utilisation de tests et séparation code de production et test.

2 Logiciels

Si vous effectuez ce TP depuis votre machine personnelle, il vous faut installer les outils suivants :

— **Terminal shell** si vous êtes sous *windows*, nous vous conseillons d'installer ubuntu sur windows pour avoir accès à un terminal shell. Sous *MacOS* et *linux*, un terminal shell est disponible de base. On vous conseille d'installer Oh-my-zsh pour votre terminal qui rend l'utilisation de *git* dans le terminal plus convivial.
— **JDK version 21 ou plus** la méthode d'installation dépend de votre système d'exploitation :
  — **Windows**: Télécharger et exécuter l'Installeur windows
  — **MacOS**: Télécharger et exécuter l'Installeur MacOS (attention à prendre le bon installeur suivant le processeur de votre ordinateur aarch64 pour les processeurs M1, M2, M3 ou M4 et x64 pour les processeurs *intel*)
  — **Linux**: exécuter la commande suivante dans le terminal `sudo apt install openjdk-21-jdk`. Un mot de passe administrateur vous sera demandé.
— **Gradle**: Suivez les instructions au lien suivant : install gradle
— **git**: Télécharger et installer git au lien suivant : download git
— **IntelliJ IDEA** : vous pouvez télécharger IntelliJ IDEA ou bien JetBrains Toolbox qui est un outil pour gérer les IDE proposé par l'entreprise JetBrains. En tant qu'étudiant, vous avez d'ailleurs accès à des licences gratuites pour leurs produits et notamment la version *ultimate* d'*IntelliJ IDEA* en créant un compte avec votre adresse AMU. Vous pouvez aussi utiliser Eclipse ou Visual studio code qui sont des outils similaires.

3 Création de projet

Le but de la première partie de ce TP est d'apprendre à configurer un projet en utilisant *IntelliJ IDEA* ou bien entièrement en ligne de commande. Le projet permettra d'avoir de :

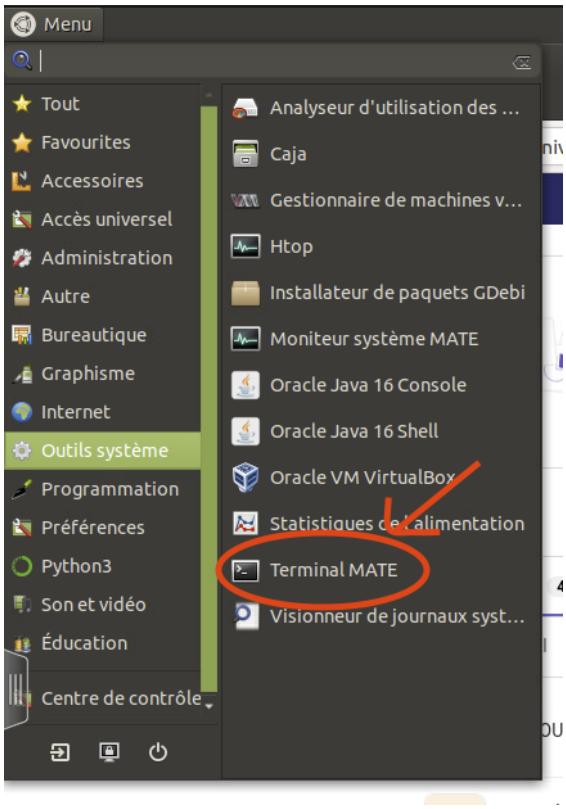
— la gestion de version via git avec la création de branches ;
— les tests unitaires avec Junit et AssertJ ;
— un moteur de production gradle afin de gérer des dépendances et d'exécuter les tests unitaires ou des outils de mesure de la qualité du code.

3.1 Configuration clé ssh pour le gitlab AMU

La première étape pour utiliser la gestion de version est de vous connecter au gitlab de l'université qui est accessible à l'adresse suivante : https://etulab.univ-amu.fr/. L'identifiant et le mot de passe sont ceux de votre compte étudiant AMU.

Afin de pouvoir accéder à distance à vos dépôts git, vous allez configurer une clé SSH dans votre profil. Pour cela, il vous faut :

1. Ouvrir un terminal (par exemple terminal MATE de la vdi Lunix qui est accessible dans le menu au sous-menu Outils systèmes). Si vous êtes sous *windows* et que vous avez installé Git pour windows, il est conseillé d'utiliser le terminal bash de git.



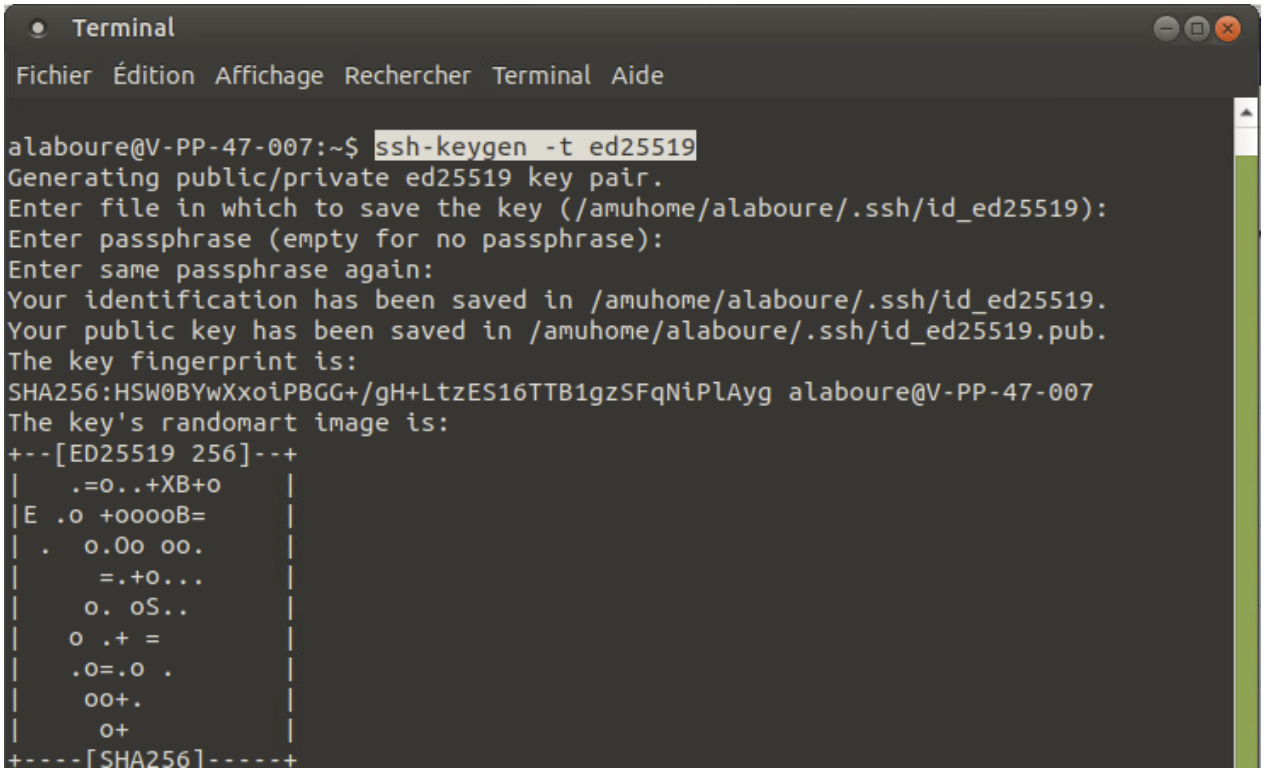
2. Générer une paire de clés privés/publiques pour votre compte. Pour cela, il vous faut entrer la commande suivante dans le terminal :

```
~$ ssh-keygen -t ed25519
```

Appuyer une fois sur Entrée pour confirmer que vous voulez sauvegarder vos clés dans le répertoire par défaut.

Ensuite, il vous est demandé de rentrer deux fois une “passphrase”. Vous pouvez entrer une phrase (plusieurs mots donc) qui servira de mot de passe pour l’accès. Puisque la sécurité de vos dépôts n’est pas critique, vous pouvez vous contenter de ne rien rentrer (pas de *passphrase*) et donc d’appuyer de nouveau sur Entrée deux fois.

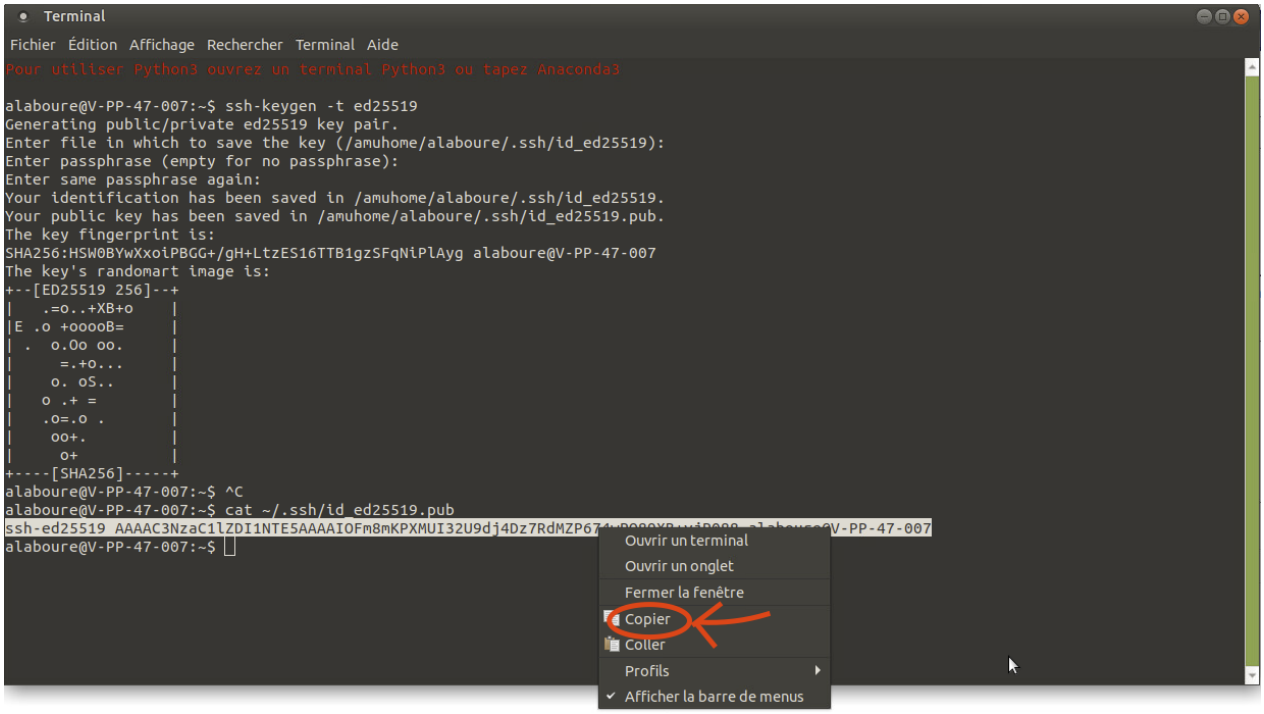
Si tout s’est bien passé, votre couple de clés a été généré et vous devriez voir un affichage similaire à celui ci-dessous :



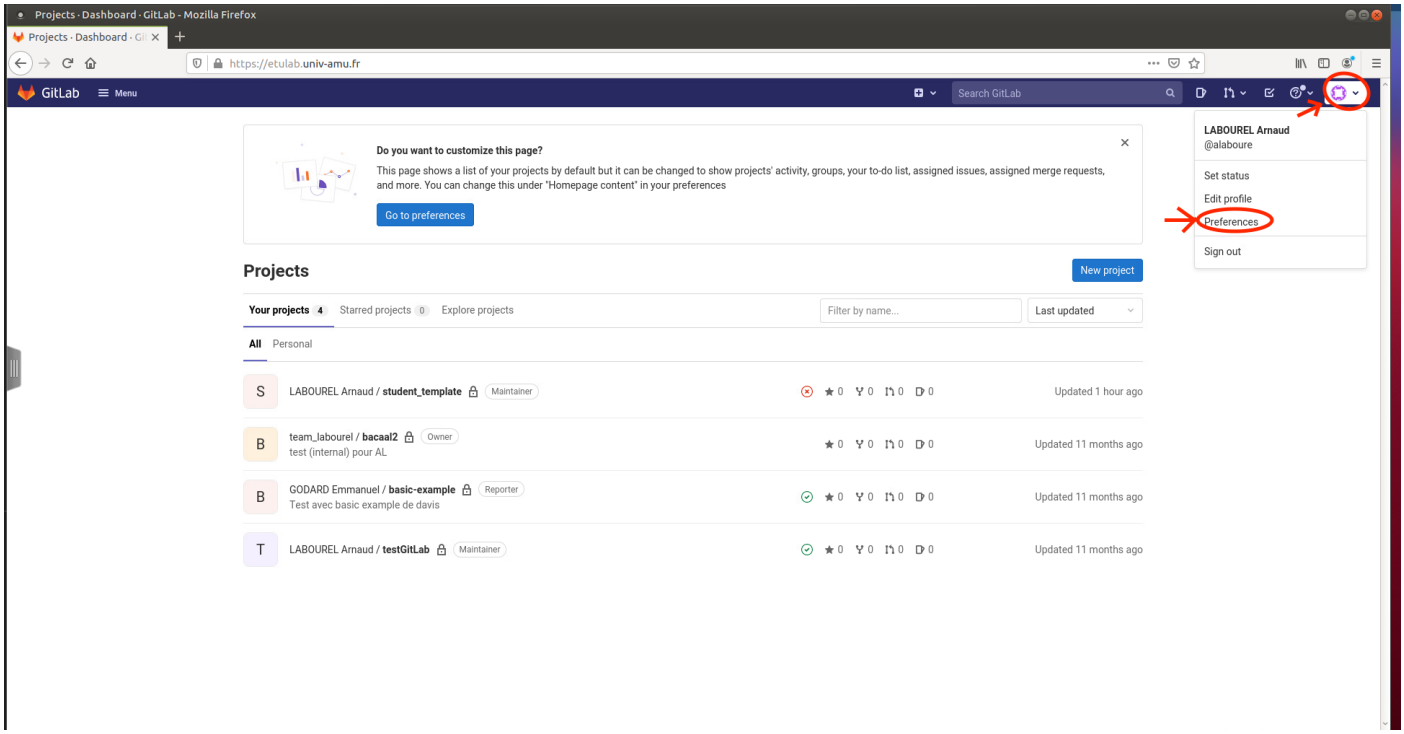
Afficher votre clé dans le terminal en entrant la commande suivante (sous *powershell*, il faut remplacer la commande `cat` par la commande `type`) :

```
~$ cat ~/.ssh/id_ed25519.pub
```

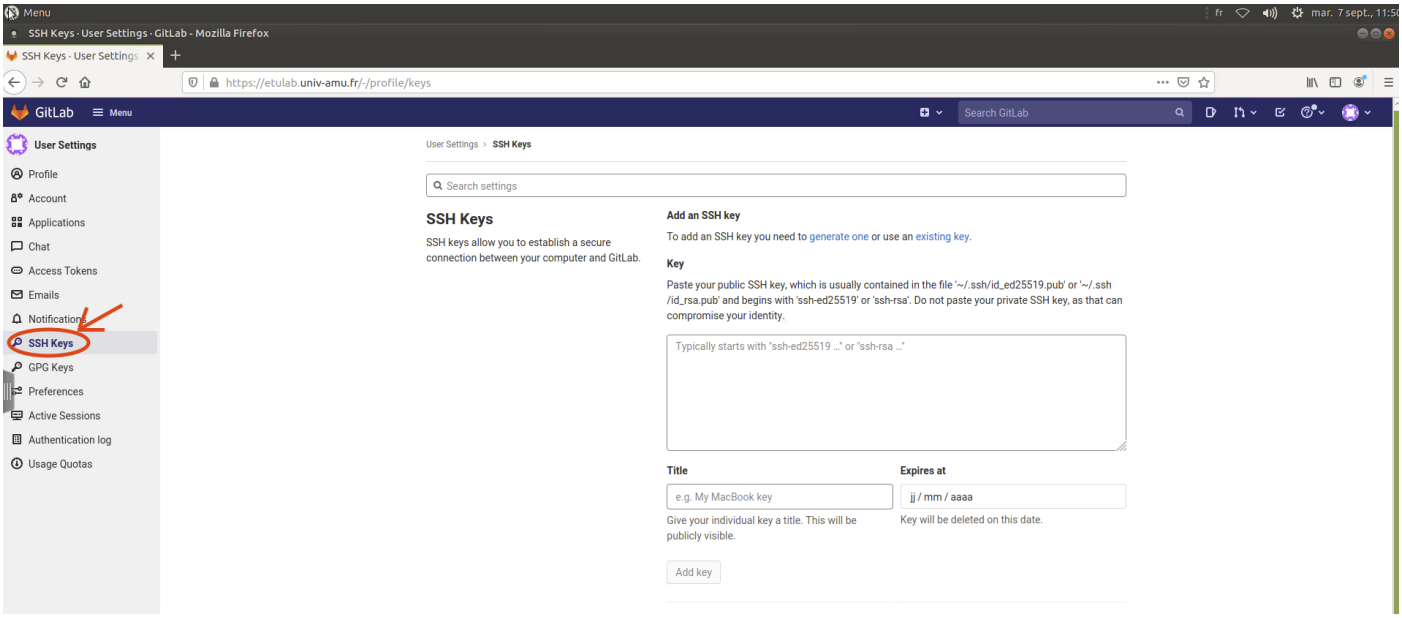
Sélectionner la ligne affichée par le terminal et copier là dans le presse-papier (sélection puis clic droit et copier)



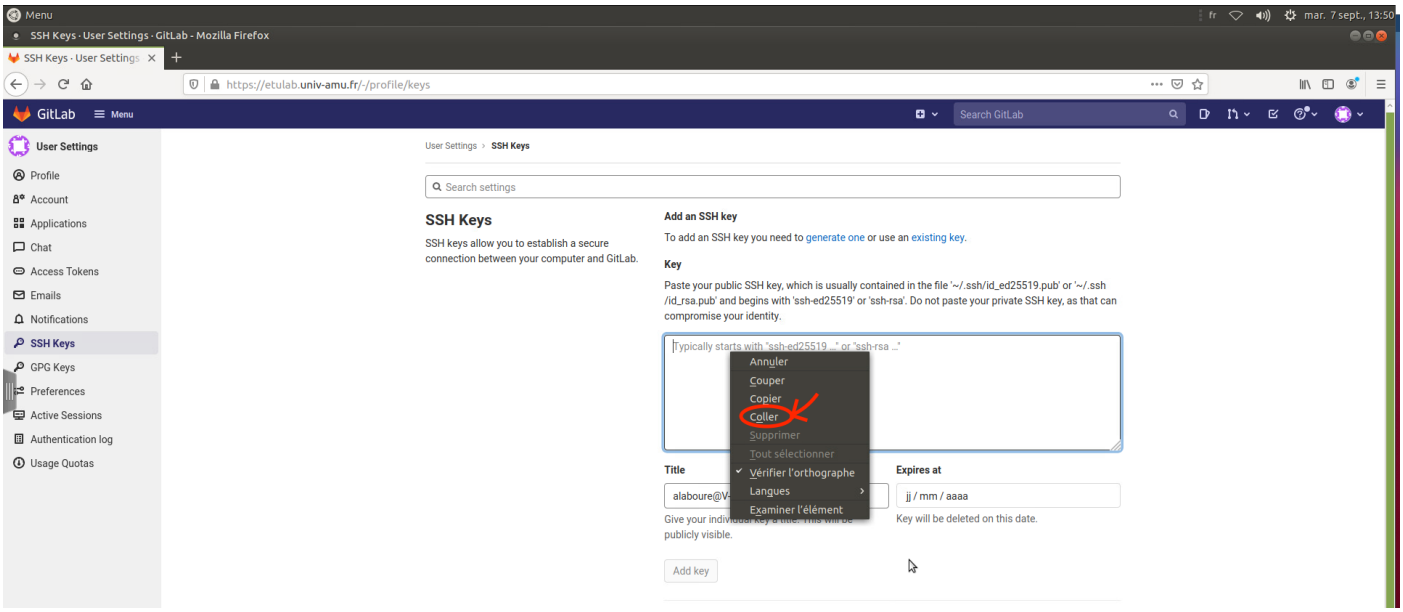
3. Configurer la clé dans votre compte gitlab. Pour cela, vous allez accéder aux préférences de votre compte gitlab. Il vous faut vous connecter à https://etulab.univ-amu.fr/ puis cliquez sur votre avatar en haut à gauche de l'écran puis sur *preferences* dans le menu qui apparaît.



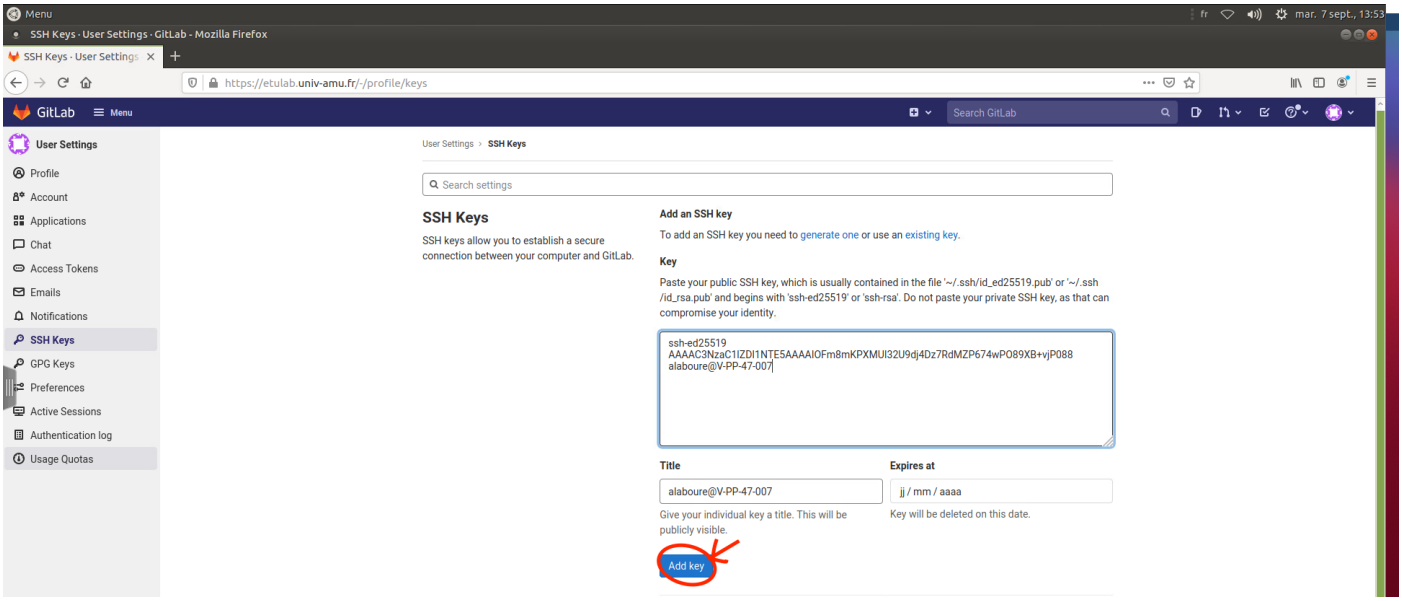
Allez au menu des clés SSH en cliquant sur le lien correspondant dans le menu de gauche.



4. Revenez sur la page de configuration des clés ssh sur votre navigateur. Coller votre clé (clic droit puis coller) dans l'espace prévu pour cela.



Ajouter la clé en cliquant sur le bouton *add*.



Vous devriez recevoir un mail sur votre mail étudiant confirmant que vous avez rajouté une clé. Vous pouvez rajouter autant de clés que vous voulez. Vous pouvez donc faire de même pour rajouter par exemple des clés supplémentaires si vous souhaitez accéder à votre dépôt depuis chez vous.

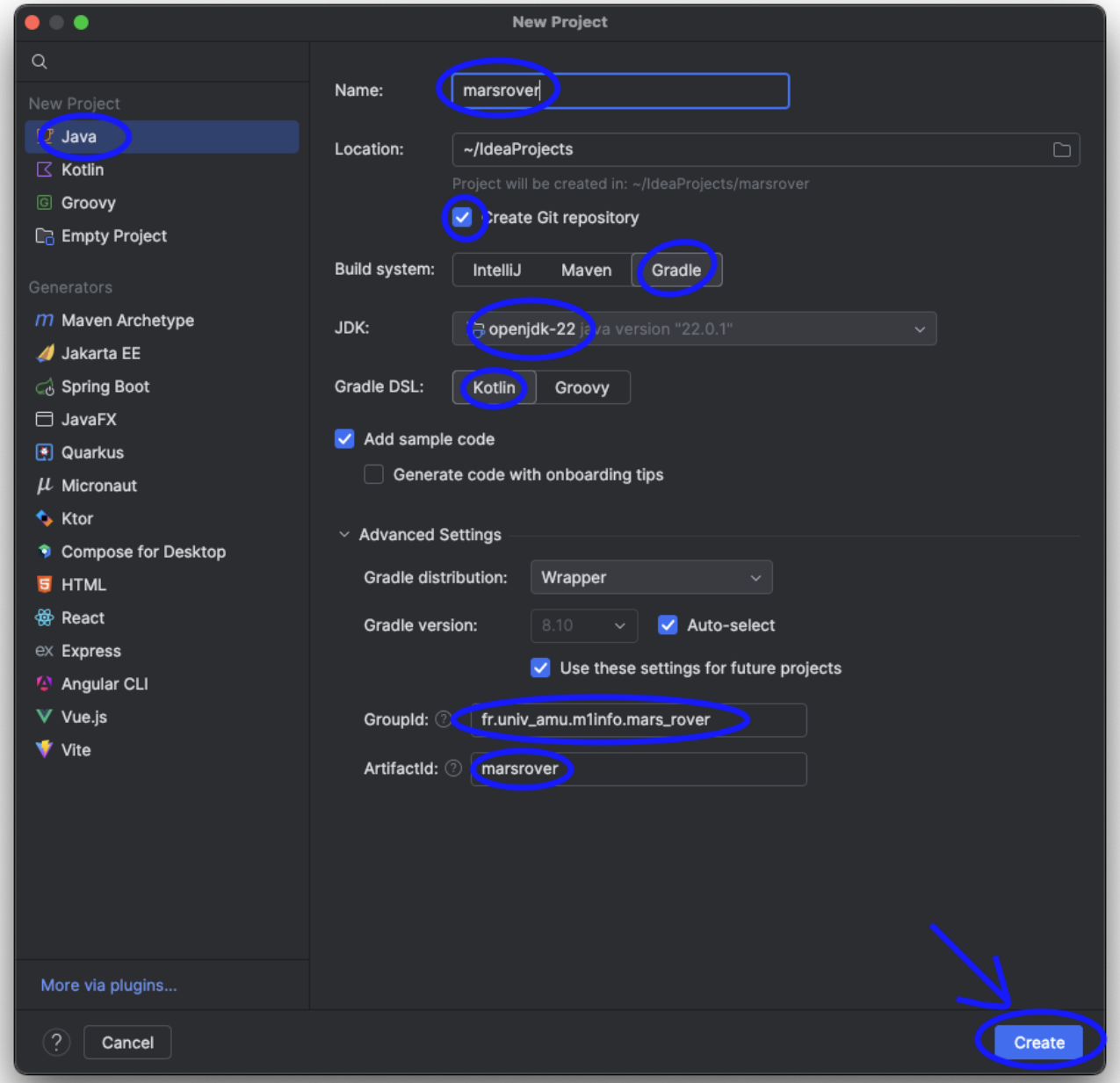
3.2 Création de projet versionné à l'aide d'IntelliJ

3.2.1 Création du dépôt local

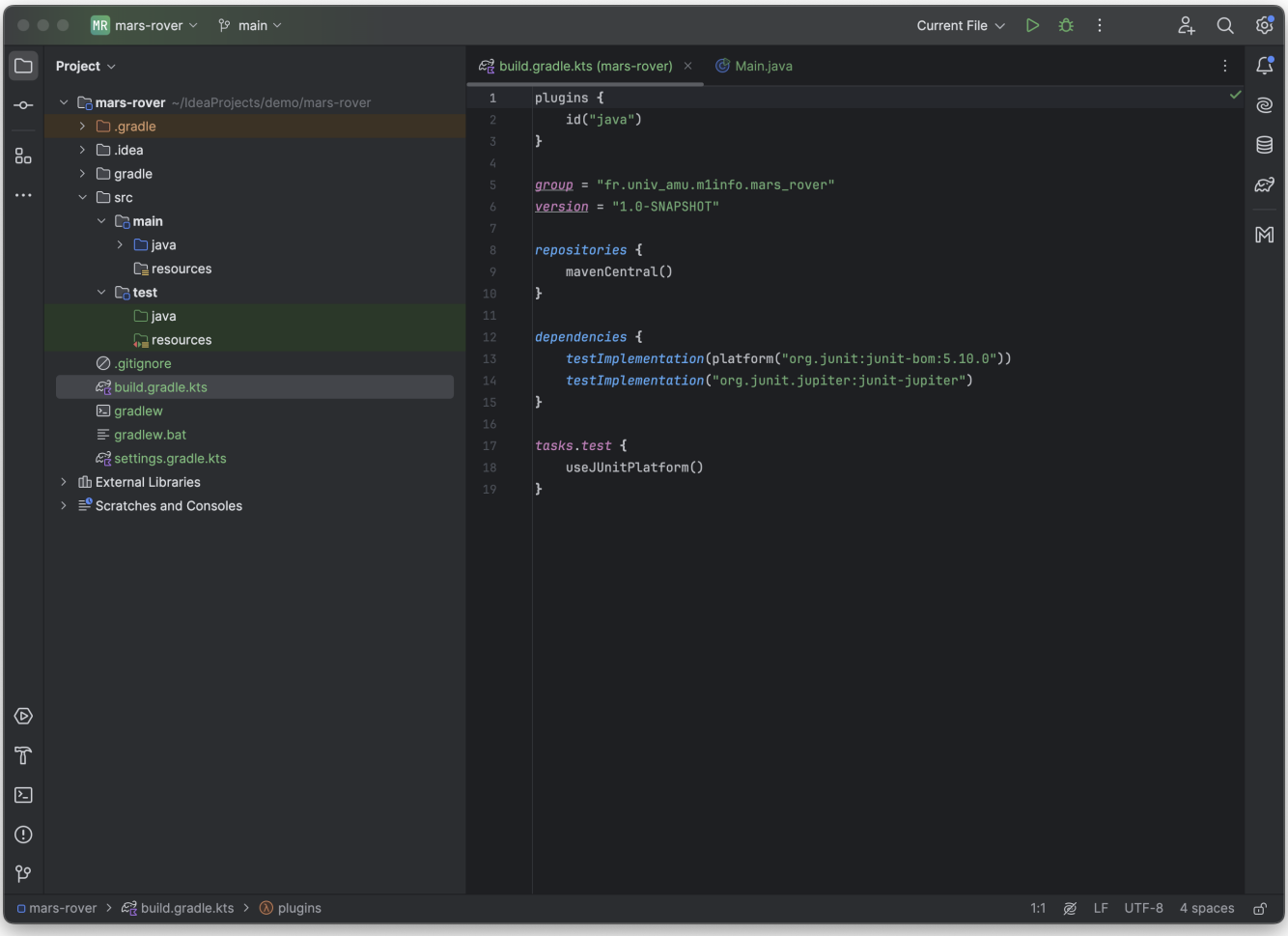
- Lancez IntelliJ IDEA
- Commencez la création d'un nouveau projet via le menu `file` -> `New` -> `Project`.
- Créer un projet pour le TP. Pour ce faire, il vous faut (dans l'ordre du haut vers le bas) :
  - Sélectionner dans la colonne de gauche *Java* comme type de projet
  - choisir un nom (*name*) pour votre projet, pour ce projet, le nom sera `m3c` ;
  - choisir le répertoire de stockage (*Location*) du projet (vous pouvez laisser le répertoire de base qui est `~/IdeaProjects/`)
  - cocher la case *Create Git repository* pour créer un dépôt git local ;
  - choisir un JDK de version 21 ou plus (normalement JDK 22) et
  - choisir gradle comme moteur de production (*Build system*) ;
  - choisir Kotlin comme Gradle DSL ;
  - ouvrir les paramètres avancés (*Advanced Settings*) ;
    - choisir un identifiant de groupe [1] (*GroupId*) pour votre projet, pour ce projet, vous devez mettre `fr.univ_amu.l3mi.m3c` ;
    - choisir un identifiant de composant [2] (*ArtifactId*) pour votre projet, vous pouvez laisser le nom du projet (`m3c`).

1. L'identifiant de groupe (*GroupId*) permet de connaitre l'organisation, l'entreprise, l'entité ou la communauté qui gère le projet. Par convention, on utilise le nom de domaine Internet inversé, selon la même logique que celle généralement recommandée pour les noms de packages Java.
2. L'identifiant de composant (*ArtifactId*) est le nom unique du projet au sein du groupe qui le développe. Généralement, l'identifiant du composant est le nom du projet.

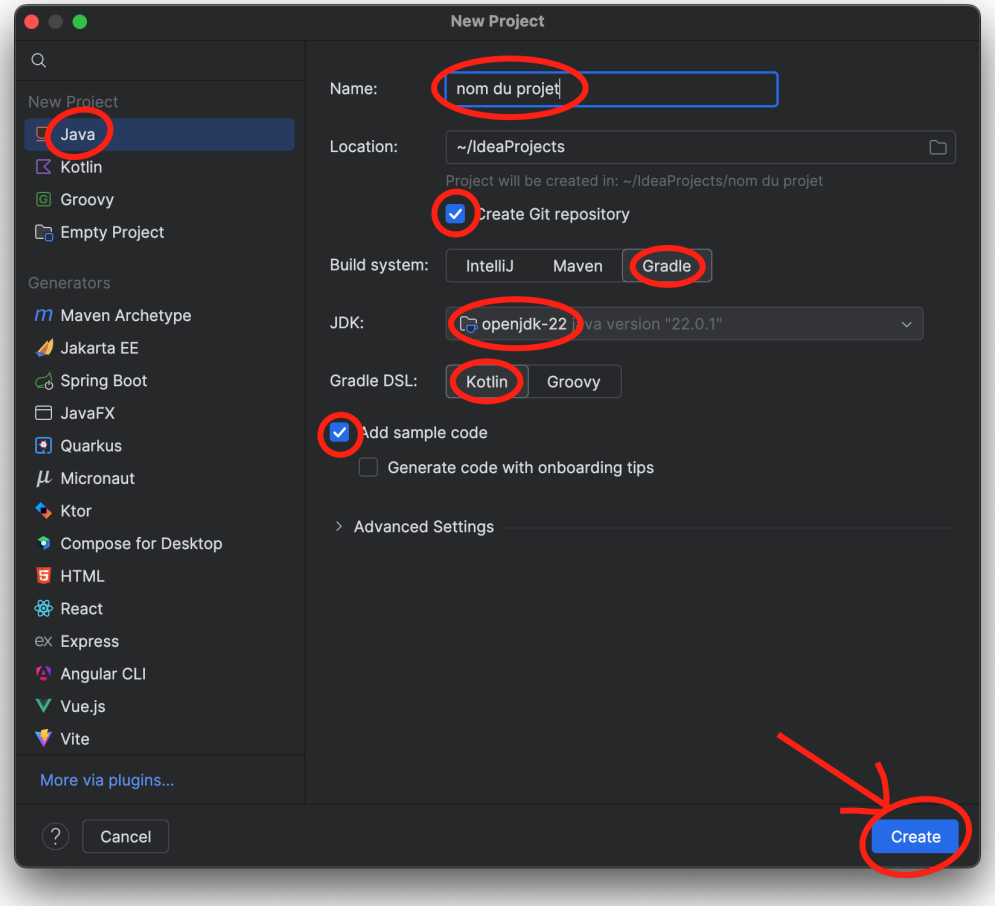
— cliquez sur le bouton *Create*.



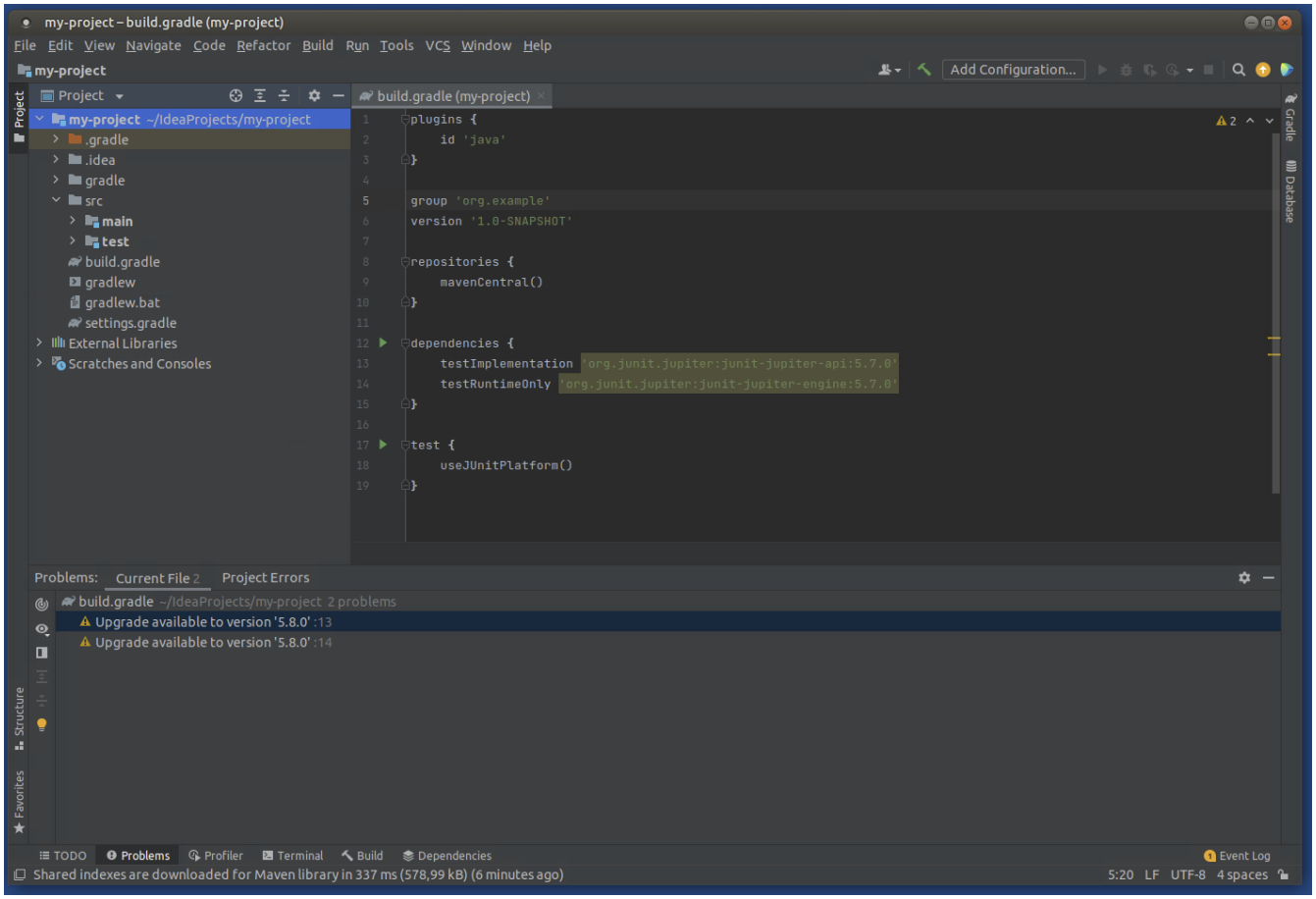
Votre projet devrait se configurer automatiquement et vous devriez obtenir l'affichage suivant.



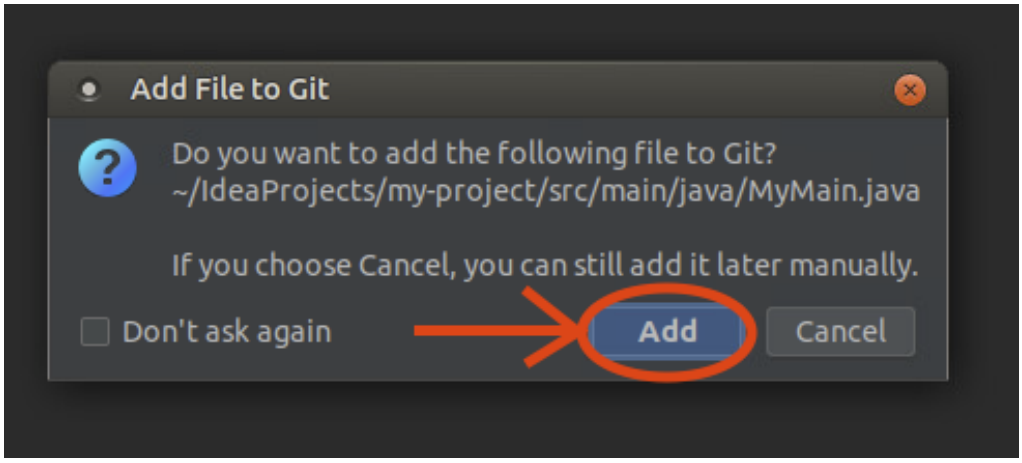
Gradle est ce que l'on appelle un moteur de production. C'est un outil pour compiler et exécuter les projets. Il est particulièrement utile pour gérer les dépendances de bibliothèques (code issu de base de code préexistante). En fait, cet outil télécharge automatiquement les bibliothèques configurées dans le projet dans le fichier `build.gradle.kts`.



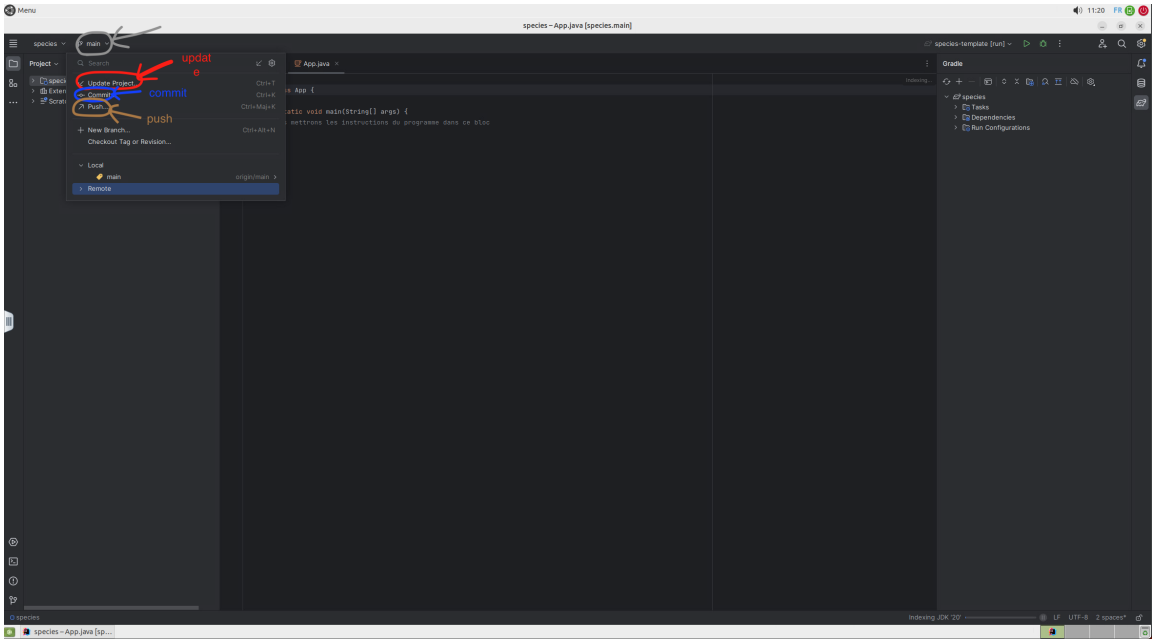
— Votre projet devrait se configurer automatiquement et vous devriez obtenir l'affichage suivant.



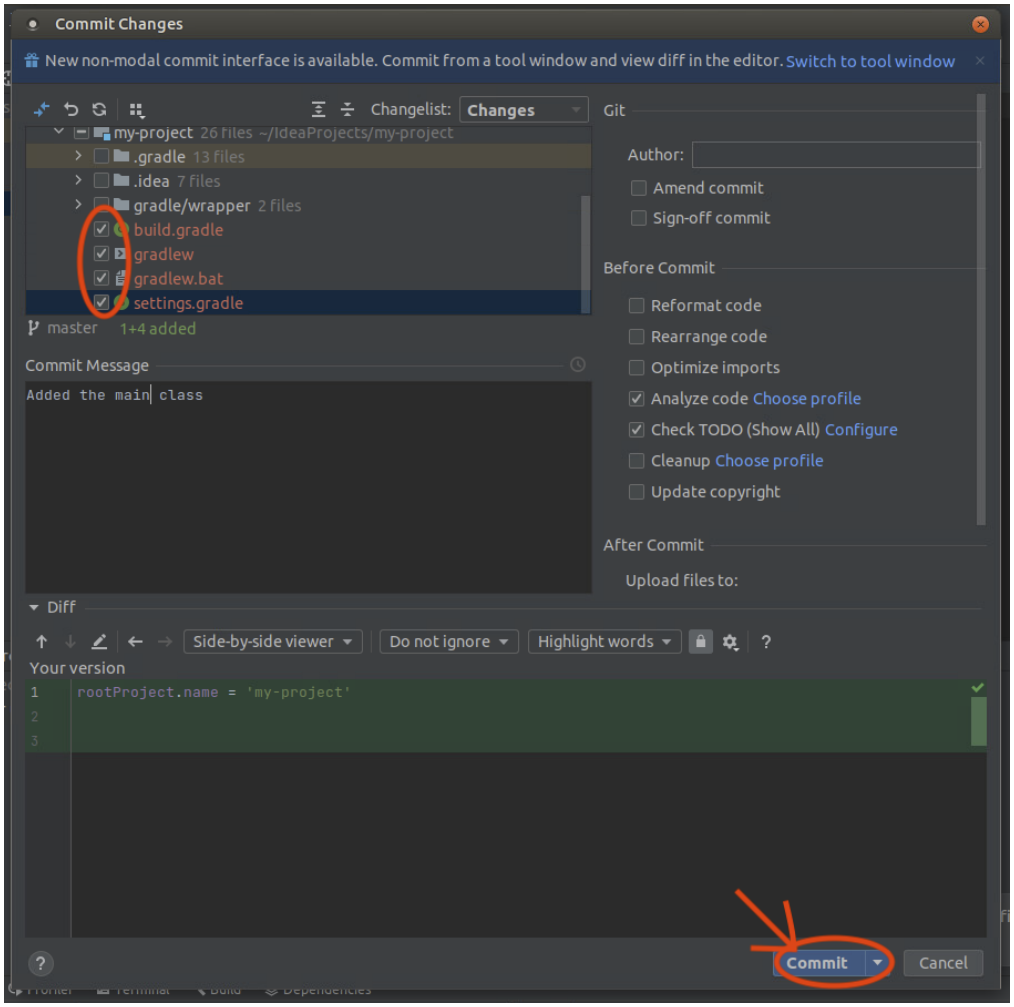
— Créez une classe dans le répertoire `src` -> `main` -> `java`. Une fenêtre va s'ouvrir pour demander si vous voulez rajouter le fichier contenant votre classe dans le dépôt `git`. Cliquez sur le bouton `add` pour valider cet ajout.



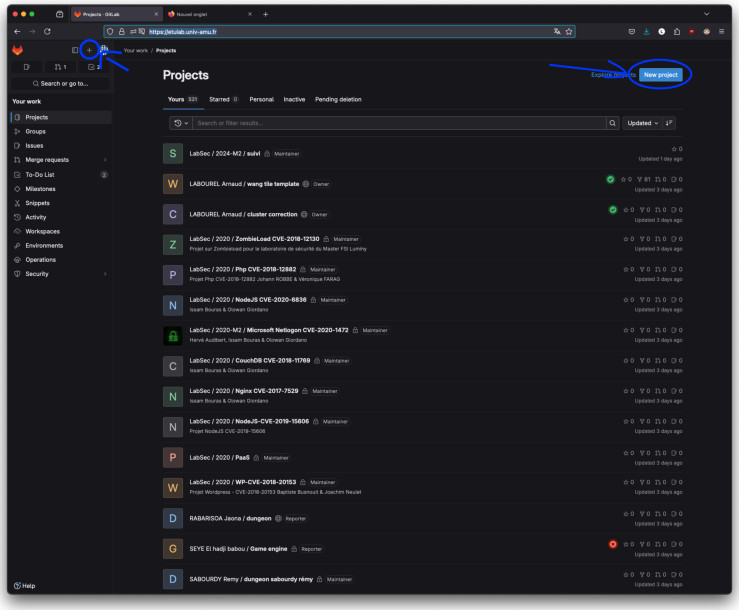
— Vous pouvez effectuer les opérations de base de `git` (`update` du projet, `commit` et `push`) grâce aux raccourcis d'IntelliJ IDEA. Ils sont en haut à gauche de la fenêtre d'IntelliJ IDEA :



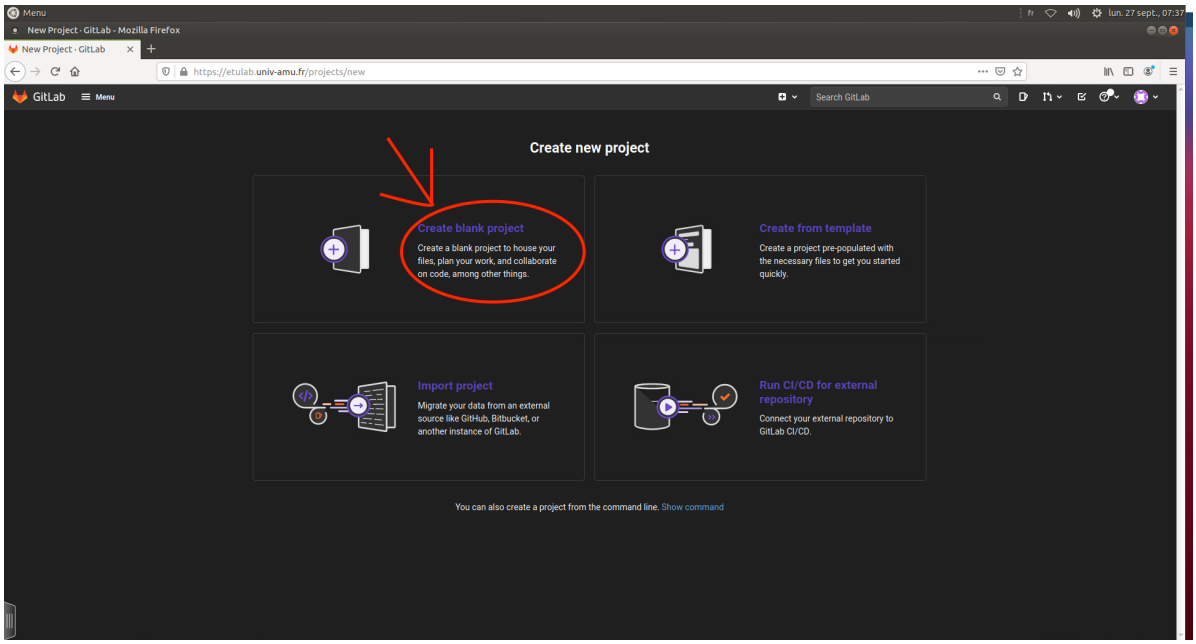
— Faites le premier commit de votre projet en ajoutant à vos fichiers `.java` les fichiers de configuration de *gradle* : `build.gradle.kts`, `gradlew`, `gradlew.bat` et `settings.gradle`.



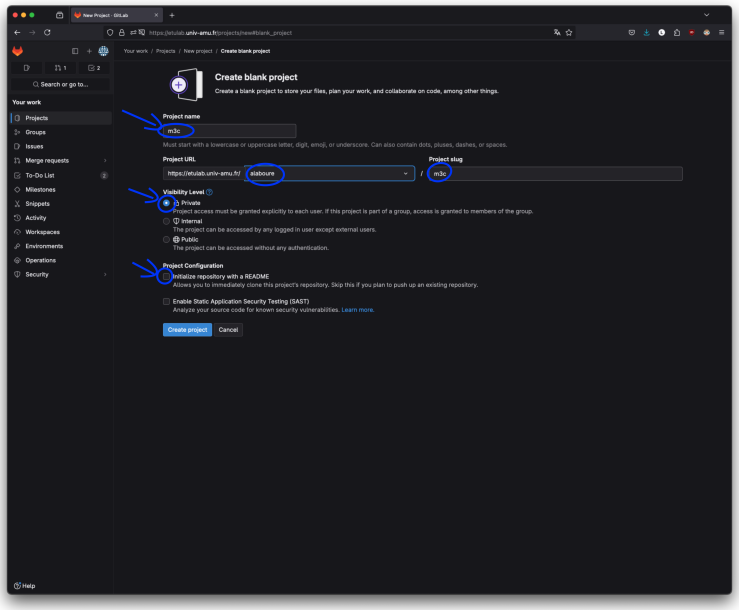
— Connectez-vous via à un navigateur à votre compte etulab et créez un nouveau projet en cliquant sur `+` en haut à gauche puis sur `Create new project/repository` ou bien avec le bouton `New project` en haut à droite.



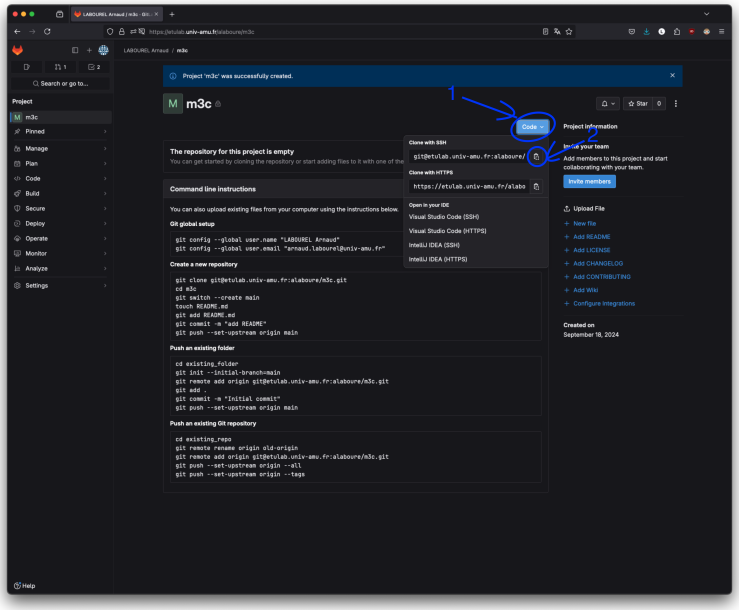
— Choisissez `Create blank project`.



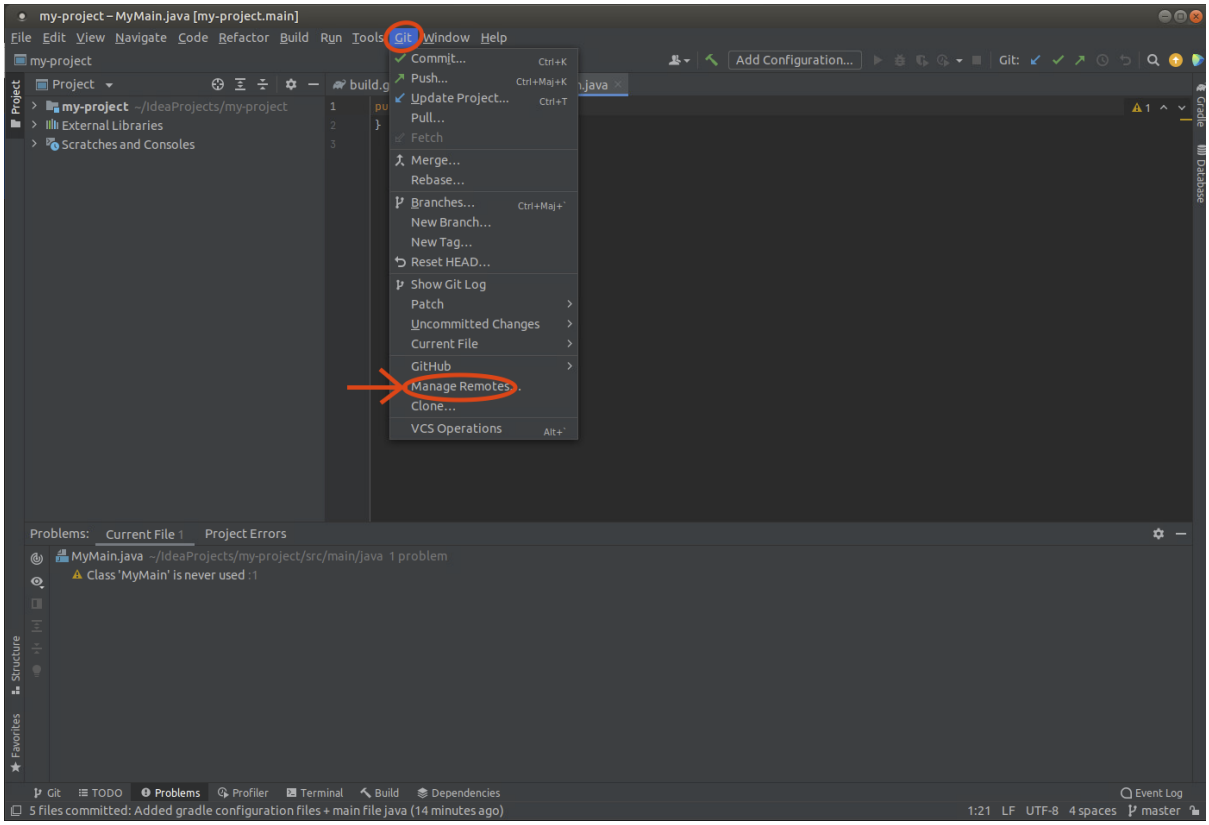
— Choisissez un nom pour votre projet, décochez la création du `README.md` et validez la création en cliquant sur le bouton `Create project`.



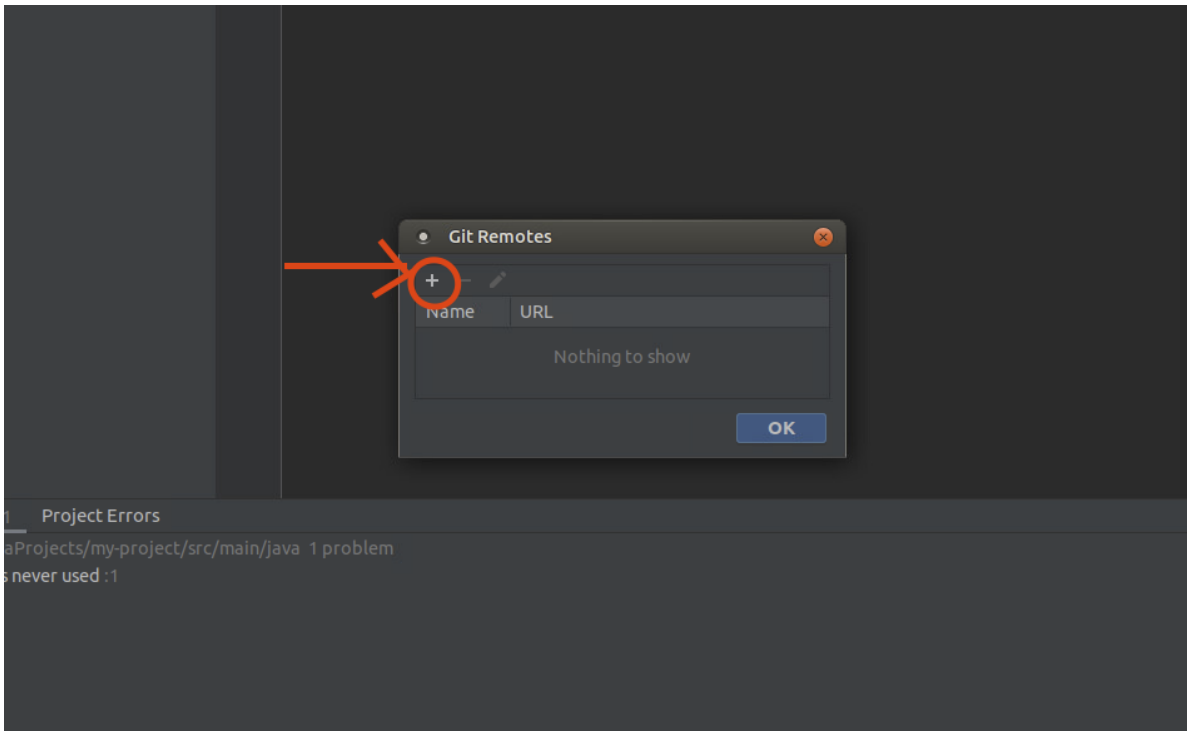
— Copiez l'adresse pour cloner votre projet en cliquant sur le bouton `code` puis sur l'icône de copie de texte de `Clone with ssh`.



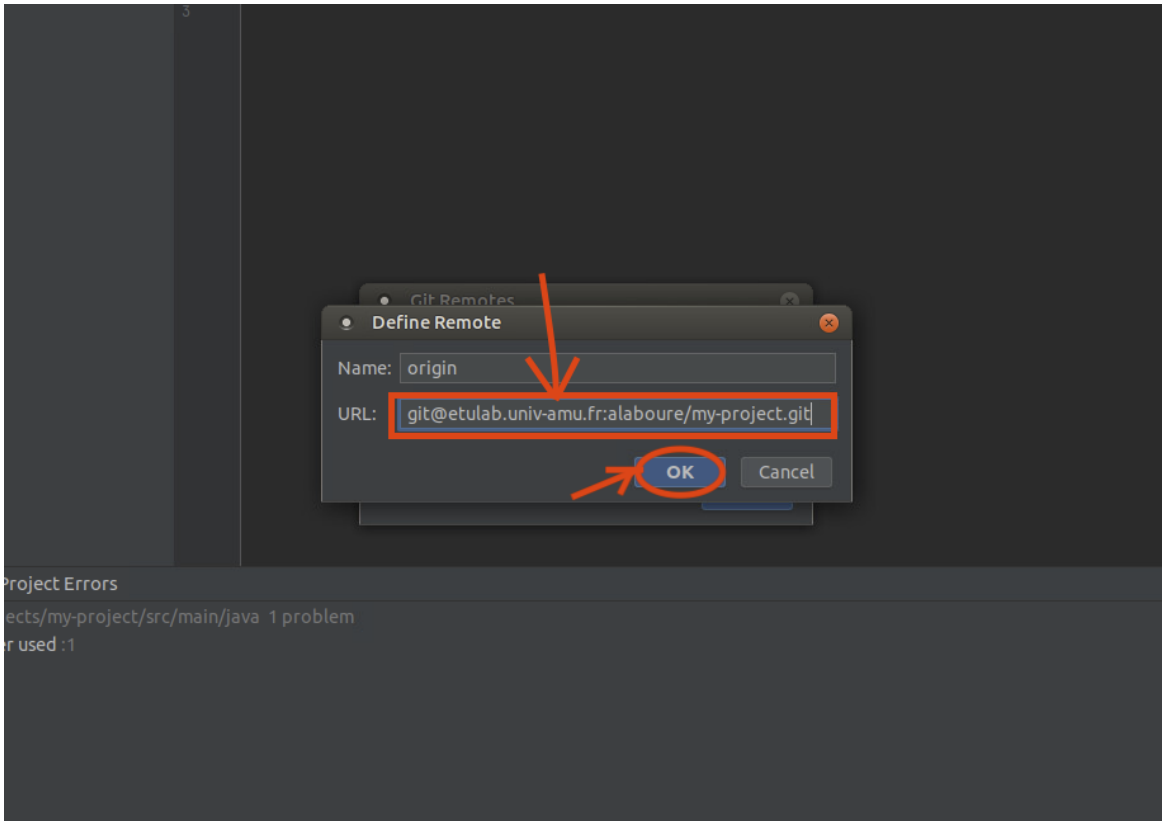
— Retournez sur votre projet sur `IntelliJ` et allez dans le menu : `git` puis `Manage Remotes`.



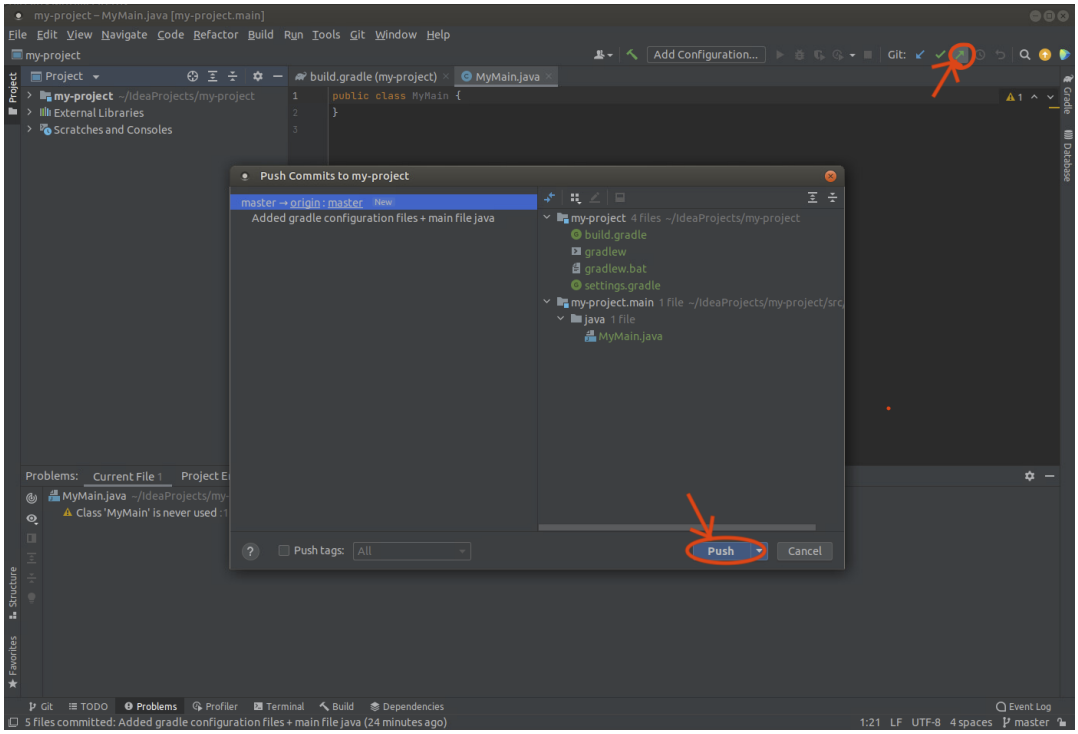
— Cliquez sur le bouton `+` pour ajouter un dépôt distant à votre dépôt local.



— Copiez l'adresse de votre dépôt etulab dans le champ `URL` et validez l'ajout en cliquant sur le bouton `OK` deux fois (une fois pour la fenêtre `Define remote` et une autre fois pour la fenêtre `Git remotes`).



— Vous pouvez maintenant faire le premier `push` de votre projet en cliquant par exemple dans le bouton dédié en haut à droite puis en validant en cliquant sur le bouton `push`.



3.3 Création de projet versionné en ligne de commande

Si vous avez créé votre projet avec *IntelliJ IDEA*, vous pouvez ignorer cette partie qui décrit la création de projet utilisant uniquement le terminal shell et donc sans *IntelliJ IDEA*.

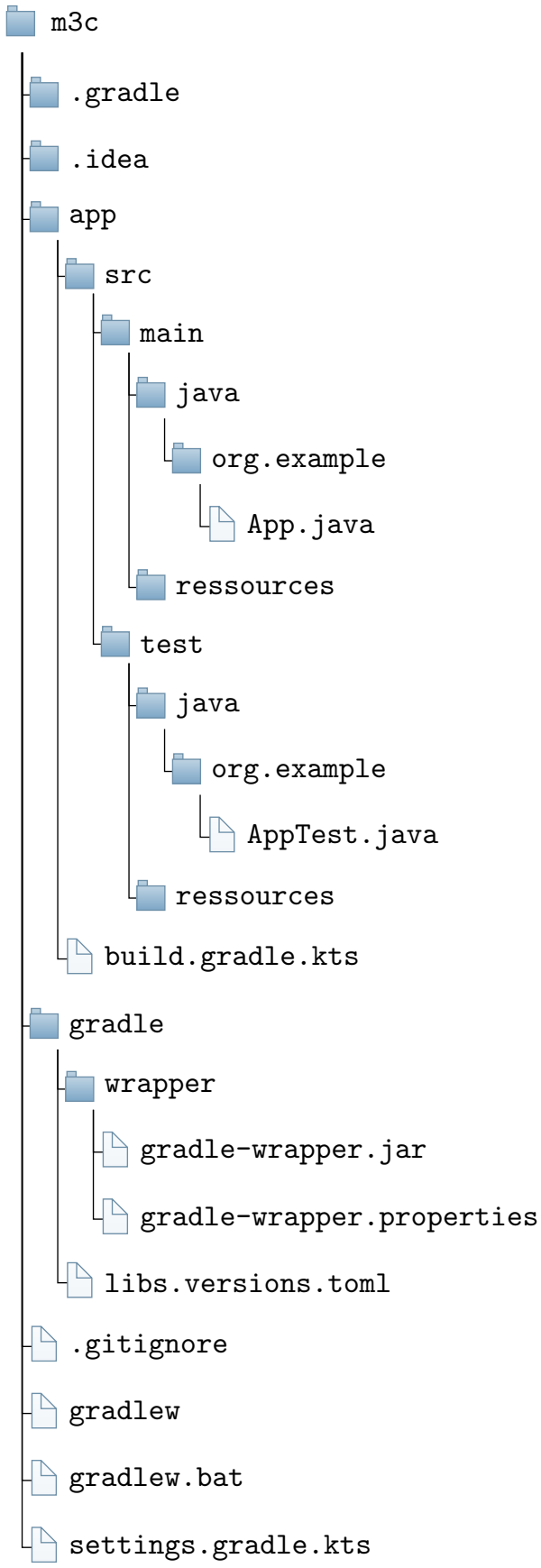
3.3.1 Création de projet gradle en ligne de commande

La première étape de la création du projet est de créer un projet *gradle*.

- Créez le répertoire qui va contenir votre projet qui devra être nommé `m3c`.
- Pour commencer la création de votre projet lancez la commande `gradle init` alors que vous êtes dans le répertoire `m3c`.
  - On vous demande tout d'abord quel type de projet vous souhaitez créer. Tapez `1` puis `entrée` pour choisir application.
  - On vous demande le langage de projet. Tapez `1` puis `entrée` pour choisir Java.
  - On vous demande ensuite la version java du projet, vous pouvez laisser 21 qui est la version par défaut.
  - On vous demande le nom de votre projet. Vous pouvez laisser le nom par défaut ou bien définir votre propre nom de projet.
  - On vous demande si votre projet est une application simple ou application et *library*. Tapez `1` et validez avec la touche entrée pour choisir `Single application project`.
  - On vous demande si vous préférez utiliser `Groovy` ou `Kotlin` pour le langage de script de *build*. Tapez `1` puis `entrée` pour choisir `Kotlin`.
  - On vous demande quel framework de tests vous souhaitez utiliser. Tapez `4` puis `entrée` pour choisir `Junit Jupiter`.

3.3.2 Explication fichiers et répertoires du projet

Votre projet devrait avoir l'arborescence de fichiers suivante :


m3c
.gradle
.idea
app
src
main
java
org.example
App.java
ressources
test
java
org.example
AppTest.java
ressources
build.gradle.kts
gradle
wrapper
gradle-wrapper.jar
gradle-wrapper.properties
libs.versions.toml
.gitignore
gradlew
gradlew.bat
settings.gradle.kts

— Le répertoire `.gradle` contient les fichiers de configuration de *gradle*.
— Le répertoire `.idea` contient les fichiers de configuration d'*IntelliJ Idea*.
— Le répertoire `gradle` contient le gradle wrapper qui permet de fixer la version de *gradle* (la téléchargeant au besoin).
— Le sous-répertoire `src` du répertoire `app` contient les fichiers sources du projet.
  — Le sous-répertoire `main` contient les fichiers pour le code principal de l'application, c'est-à-dire les fichiers nécessaires pour le lancement de l'application. Le sous-répertoire `java` contient les fichiers *Java* alors que le sous-répertoire `ressources` contient les autres types de fichiers comme des fichiers de configuration ou des images pour l'interface graphique. Le fichier `App.java` contient la méthode `main` qui est exécuté lorsque vous taper la commande `gradle run`.
  — Le sous-répertoire `test` contient les fichiers pour le code de test. Le sous-répertoire `java` contient les fichiers *Java* alors que le sous-répertoire `ressources` contient les autres types de fichiers comme des fichiers d'entrée pour des tests de lecture. Le fichier `AppTest.java` contient des tests qui sont exécutés lorsque vous taper la commande `gradle test`. Si les tests échouent la commande vous donne un lien à ouvrir avec un navigateur.
  — Le fichier `build.gradle.kts` est un fichier de configuration *gradle* définissant en autre les dépendances du projet.
— Les fichiers `gradlew` et `gradle.bat` sont les fichiers de script pour le *wrapper gradle* respectivement pour les systèmes *Unix* et *Windows*.
— Le fichier `settings.gradle.kts` est un fichier de configuration de *gradle* permettant en autre de définir le nom du projet.

3.3.3 Création de dépôt `git`

— Dans le répertoire contenant votre projet `gradle`, exécutez la commande `git init --initial-branch=main` pour créer un dépôt `git` local.
— Dans le répertoire contenant votre projet `gradle`, exécutez la commande `git add non_de_fichier` pour ajouter les fichiers de votre projet à votre prochain `commit`.
— Il vous faudra ajouter tous les fichiers `Java` ainsi que les fichiers de configuration de gradle : `build.gradle.kts`, `gradlew`, `gradlew.bat`, `settings.gradle.kts`, `gradle-wrapper.properties` et `libs.versions.toml`.

— Exécutez la commande `git commit -m"premier message"` pour faire un premier `commit` de votre projet avec votre propre message.

3.3.4 Création du dépôt distant et du lien entre les deux dépôts

— Connectez-vous via à un navigateur à votre compte etulab et créez un nouveau projet en cliquant sur `menu` puis `Create new project`.
— Choisissez `Create blank project`.
— Choisissez un nom pour votre projet, décochez la création du `README.md` et validez la création en cliquant sur le bouton `Create project`.

— Copiez l'adresse pour cloner votre projet en cliquant sur le bouton `clone` puis sur l'icône de copie `ssh`.
— Utiliser la commande suivante pour rajouter l'adresse du dépôt distant à votre dépôt local. Il vous faudra bien évidemment remplacer le dernier argument de la commande par l'adresse de votre projet que vous avez préalablement copié : `git remote add origin adresse_projet`.
— Faites le premier `push` à l'aide de la commande `git push -u origin`. Pour les `push` suivants, vous pourrez simplement utiliser la commande `git push`, car vous avez configuré les branches.

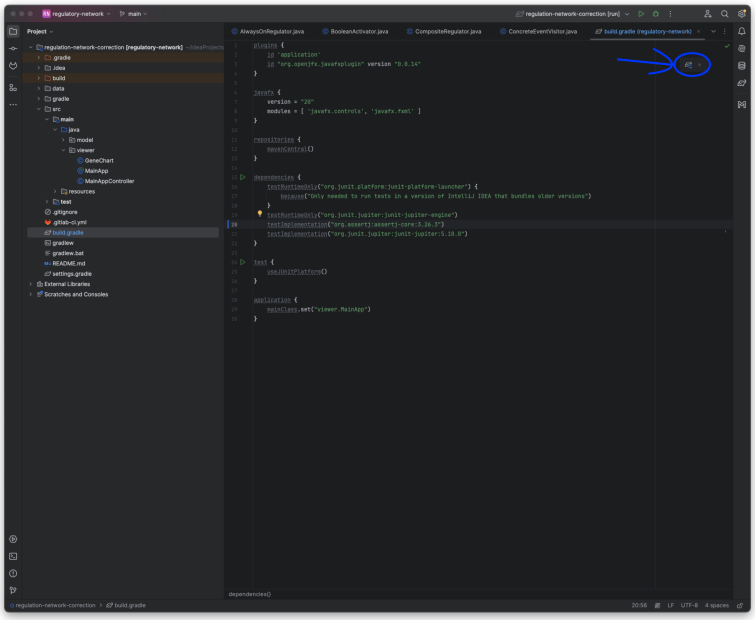
4 Méthodologie de travail

4.1 Tests unitaires avec Junit 5

Pour ce TP, vous devez tester à l'aide de tests unitaires toutes les classes que vous écrirez. Normalement votre projet est déjà configuré pour le framework de test JUnit 5. On vous conseille d'utiliser AssertJ qui ajoute entre autre des assertions facilitant l'écriture des tests. Pour cela, il vous faut rajouter *AssertJ* dans les dépendances de *gradle* en rajoutant les lignes suivantes dans le fichier `build.gradle.kts` :

```
dependencies {
    testImplementation("org.assertj:assertj-core:3.26.3")
}
```

Une fois la ligne ajoutée au fichier, cliquez sur l'icône de mise à jour.



Pour pouvoir utiliser facilement les assertions (méthodes `assertThat`) dans vos classes de test, il vous faut ajouter l'*import* suivant dans chacune vos classes de tests :

```
import static org.assertj.core.api.Assertions.*;
```

Vous trouverez davantage de détails sur les tests unitaires dans le document dédié aux tests

4.1.1 Tâches `run` et `build`

Pour configurer la tâche `run`, on va commencer par rajouter (si besoin) le plugin application dans le fichier `build.gradle.kts`. Pour cela, on rajoute la ligne suivante :

```
plugins {
  id("application")
}
```

Comme indiqué précédemment, la tâche `run` vous permet d'exécuter la méthode `main` de la classe spécifiée dans le fichier `gradle.build.kts`. Cette classe correspond à la propriété `mainClass` du plugin application à l'aide des lignes suivantes :

```
application {
  mainClass = "fr.univ_amu.l3mi.m3c.Main"
}
```

La tâche `build` définie par le plugin `java` permet de construire un `jar` dans le répertoire `build/libs` du projet. Afin que le `jar` fonctionne, il faut préciser la classe contenant le `main` à exécuter. Cela se fait par le biais du fichier manifest que l'on peut configurer en ajoutant les lignes suivantes dans le fichier `gradle.build.kts`.

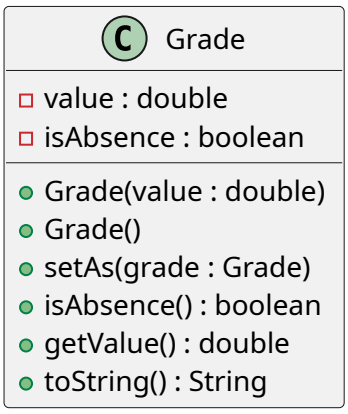
```
tasks.withType<Jar> {
    manifest {
        attributes["Main-Class"] = "fr.univ_amu.l3mi.m3c.Main"
    }
}
```

5 Tâches à réaliser

5.1 La classe `Grade`

5.1.1 Spécification des classes `Grade`

La classe `Grade` permet de représenter des notes qui peuvent être un nombre entre 0 et 20 ou bien correspondre à une absence.



5.1.2 Tâches

Tâche 1 : Créez la classe `Grade` dans le répertoire `src/main/java` de votre projet.

Tâche 2 : Créez une classe de test nommée `GradeTest` dans le répertoire `src/test/java` de votre projet qui devra vérifier via des tests unitaires le comportement de la classe `Grade`. Vous devez tester les comportements suivants :

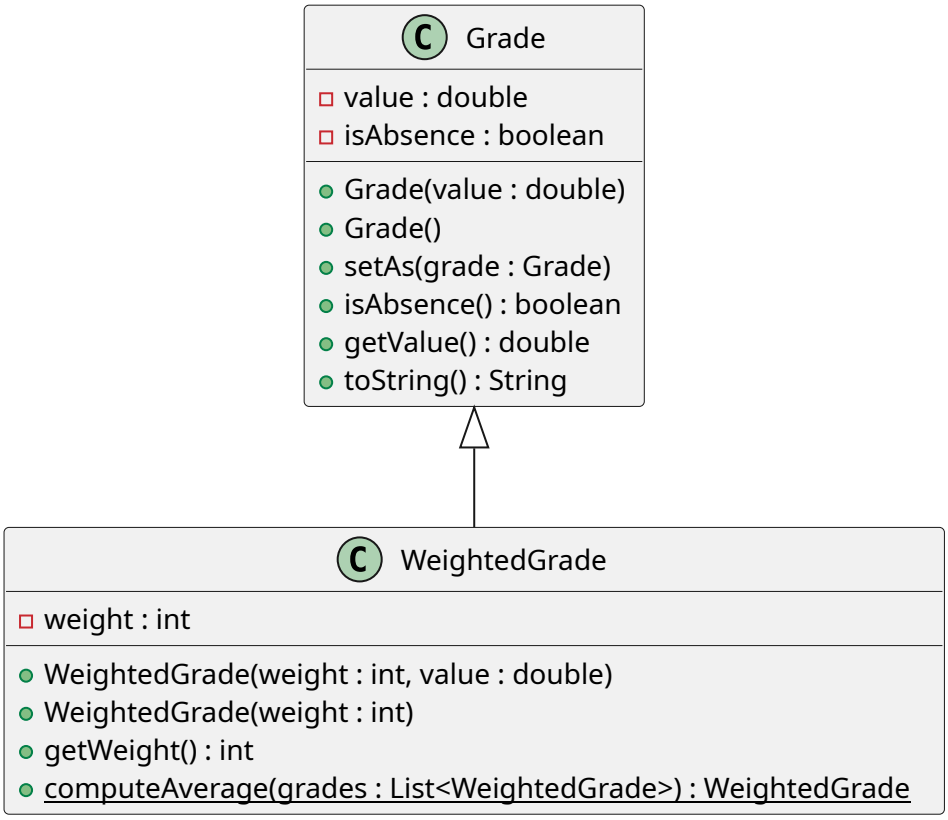
- `new Grade()` construit une note correspondant à une absence ;
- `new Grade(20.0)` construit une note correspondant à une valeur de 20 ;
- `new Grade().toString()` retourne la chaîne de caractères `ABS` ;
- `new Grade(20.0).toString()` retourne la chaîne de caractères `20,000/20` (vous pouvez pour cela utiliser la méthode `format` de `String`) ;
- `setAs(new Grade(10.0))` change la valeur de la note à 10 et la note ne correspond plus à une absence ;
- `setAs(new Grade())` change la note pour qu'elle corresponde à une absence.

5.2 La classe `WeightedGrade`

5.2.1 Spécification des classes `WeightedGrade`

La classe `WeightedGrade` permet de représenter des notes pondérées, c'est-à-dire une note (qui peut être un nombre entre 0 et 20 ou bien une absence) associé à un poids entier. Cette classe sera une extension de la classe `Grade`.

Voici le diagramme de cette classe :



5.2.2 Tâches

Tâche 3 : Créez la classe `WeightedGrade` dans le répertoire `src/main/java` de votre projet.

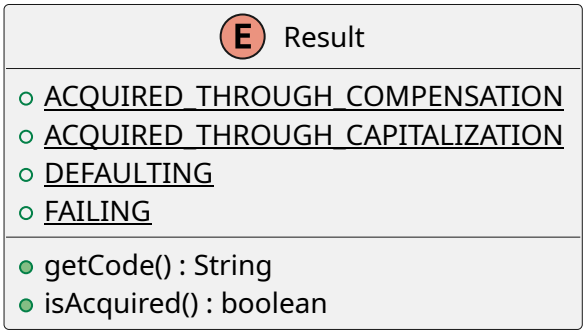
Tâche 4 : Créez une classe de test nommée `WeightedGradeTest` dans le répertoire `src/test/java` de votre projet qui devra vérifier via des tests unitaires le comportement de la classe `WeightedGrade`. Vous devez tester les comportements suivants :

- `new WeightedGrade(10)` construit une note de poids 10 correspondant à une absence ;
- `new WeightedGrade(10, 20.0)` construit une note de poids 10 correspondant à une valeur de 20 ;
- `new WeightedGrade(10).toString()` retourne la chaîne de caractères `ABS` ;
- `new WeightedGrade(10, 20.0).toString()` retourne la chaîne de caractères `20,000/20` ;
- `setAs(new Grade(10.0))` change la valeur de la note à 10 et la note ne correspond plus à une absence ;
- `setAs(new Grade())` change la note pour qu'elle corresponde à une absence ;
- `computeAverage(List.of(new WeightedGrade(1, 6.0), new WeightedGrade(2, 12.0)))` retourne une note de poids 3 ayant une valeur de 10 et ne correspondant pas à une absence ;
- `computeAverage(List.of(new WeightedGrade(1, 6.0), new WeightedGrade(2)))` retourne une note de poids 3 correspondant à une absence.

5.3 L'enum `Result`

Définir un `enum Result` permettant de représenter les 4 résultats possibles pour un élément de la formation (UE, BCC ou année) :

- `ACQUIRED_THROUGH_COMPENSATION` : acquis par compensation
- `ACQUIRED_THROUGH_CAPITALIZATION` : acquis par capitalisation
- `DEFAULTING` : défaillant (au moins un élément avec une note correspondant à une absence)
- `FAILING` : ajourné (note moyenne inférieure à 10/20 et élément pas compensé)



5.3.1 Tâches

Tâche 5 : Créez l'`enum Result` dans le répertoire `src/main/java` de votre projet.

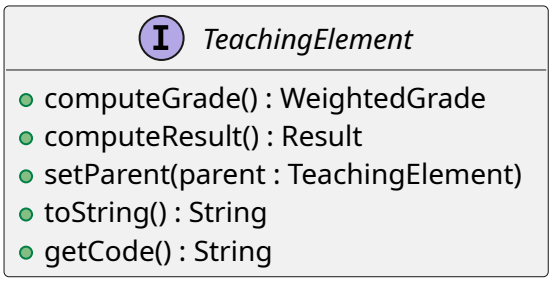
Tâche 6 : Créez une classe de test nommée `ResultTest` dans le répertoire `src/test/java` de votre projet qui devra vérifier via des tests unitaires le comportement de l'`enum Result`. Vous devez tester les comportements suivants :

— `ACQUIRED_THROUGH_COMPENSATION.getCode()` retourne la chaîne de caractères `ACMP` ;
— `ACQUIRED_THROUGH_CAPITALIZATION.getCode()` retourne la chaîne de caractères `ACAP` ;
— `DEFAULTING.getCode()` retourne la chaîne de caractères `DEF` ;
— `FAILING.getCode()` retourne la chaîne de caractères `AJ` ;
— `ACQUIRED_THROUGH_COMPENSATION.isAcquired()` retourne `true` ;
— `ACQUIRED_THROUGH_CAPITALIZATION.isAcquired()` retourne `true` ;
— `DEFAULTING.isAcquired()` retourne `false` ;
— `FAILING.isAcquired()` retourne `false` ;

5.4 L'interface `TeachingElement`

L'interface `TeachingElement` est l'interface que devra implémenter tout élément de la formation (UE, BCC et Année)

Voici le diagramme de cette interface :



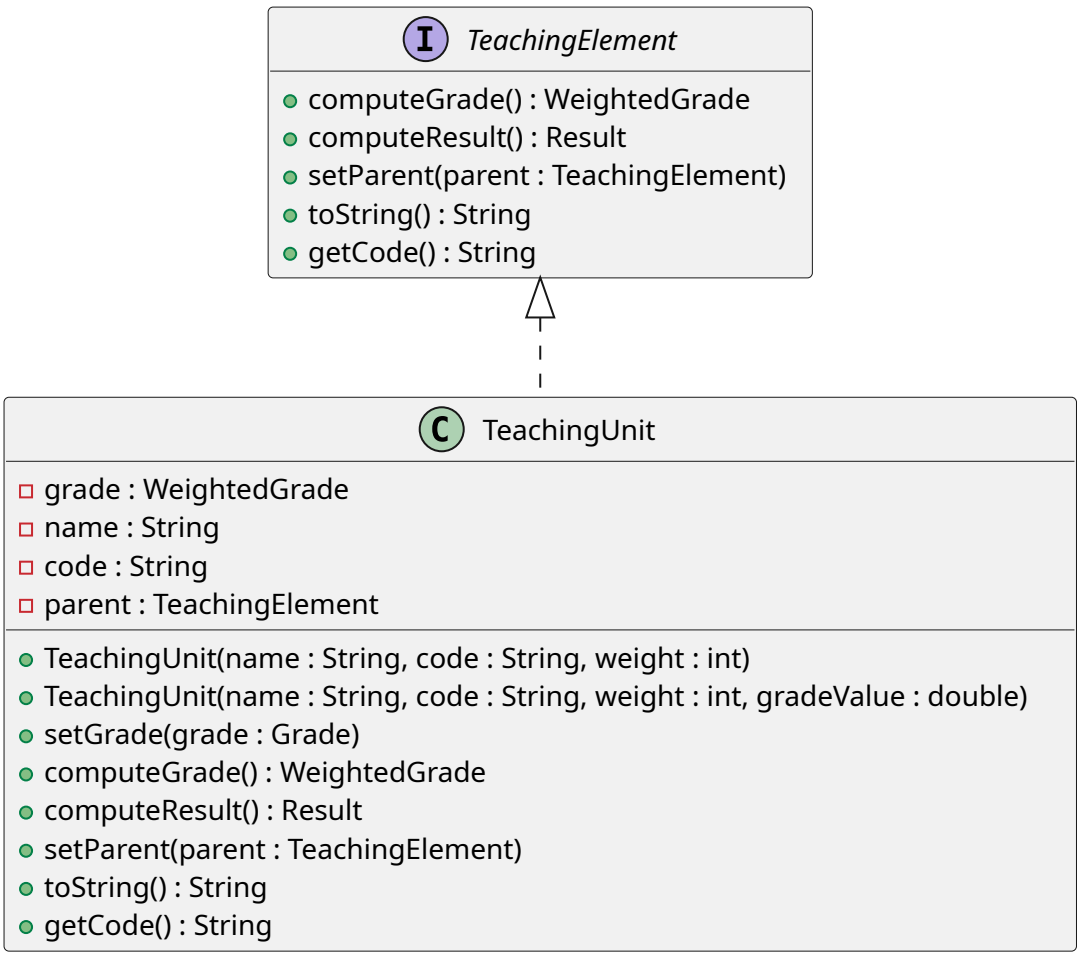
5.4.1 Tâches

Tâche 7 : Créez l'interface `TeachingElement` dans le répertoire `src/main/java` de votre projet.

5.5 La classe `TeachingUnit`

5.5.1 Spécification de la classe `TeachingUnit`

La classe `TeachingUnit` permet de représenter une Unité d'Enseignement (UE). Voici le diagramme de cette classe :



5.5.2 Tâches

Tâche 8 : Créez la classe `TeachingUnit` dans le répertoire `src/main/java` de votre projet.

Tâche 9 : Créez une classe de test nommée `TeachingUnitTest` dans le répertoire `src/test/java` de votre projet qui devra vérifier via des tests unitaires le comportement de la classe `TeachingUnit`. Vous devez tester les comportements suivants :

- `new TeachingUnit("Initiation Génie Logiciel", "SIN5U34", 3)` une unité d'enseignement de nom `Initiation Génie Logiciel` de trois crédits ayant le code `SIN5U34` associé à une note correspondant à une absence et sans parent (valeur `null`) ;
- `setGrade(new Grade(10.0))` fixe la note de l'unité d'enseignement à `10` ;
- `setGrade(new Grade())` fixe la note de l'unité d'enseignement à une absence ;
- un appel à `computeResult()` sur une unité d'enseignement avec une note à 10/20 et sans parent retourne `ACQUIRED_THROUGH_CAPITALIZATION` ;
- un appel à `computeResult()` sur une unité d'enseignement avec une note correspondant à une absence retourne `DEFAULTING` ;
- un appel à `computeResult()` sur une unité d'enseignement avec une note à 5/20 et sans parent retourne `FAILING` ;
- un appel à `computeResult()` sur une unité d'enseignement avec une note à 5/20 et avec parent

correspondant à une UE ayant une note de 20/20 retourne `ACQUIRED_THROUGH_COMPENSATION`.

— `new TeachingUnit("Initiation Génie Logiciel", "SIN5U34", 3, 10.0).toString()` devra renvoyer la chaîne de caractères `Initiation Génie Logiciel, 3 crédits (SIN5U34) : 10,000/20 (ACAP)` ;

— `new TeachingUnit("Initiation Génie Logiciel", "SIN5U34", 3, 0.0).toString()` devra renvoyer la chaîne de caractères `Initiation Génie Logiciel, 3 crédits (SIN5U34) : 0,000/20 (AJ)` ;

— `new TeachingUnit("Initiation Génie Logiciel", "SIN5U34", 3).toString()` devra renvoyer la chaîne de caractères `Initiation Génie Logiciel, 3 crédits (SIN5U34) : ABS (DEF)`.

5.6 L'interface `ResultCalculator` et les classes `AboveTenResultCalculator` et `YearResultCalculator`

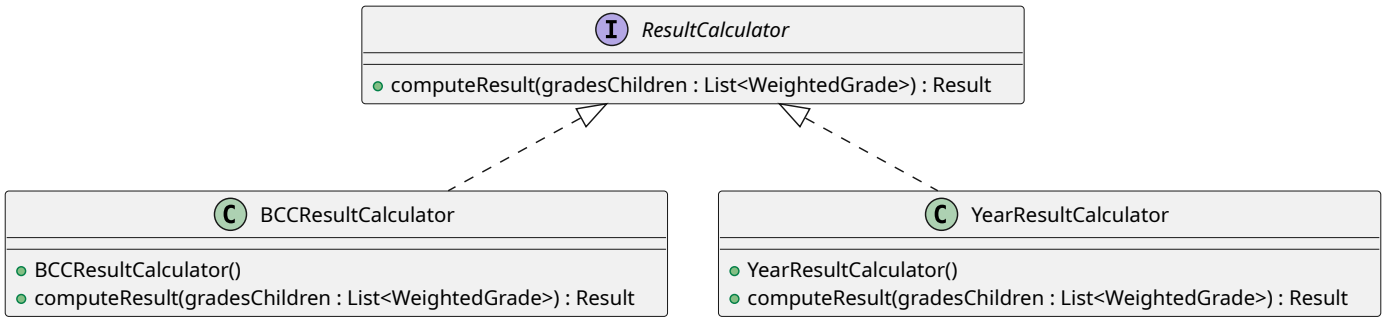
5.6.1 Spécification des classes `BCCResultCalculator` et `YearResultCalculator`

Le but de cette partie est d'implémenter le calcul de règles de validation par des classes dédiées. Un calculateur de résultat aura une seule méthode qui prendra en paramètre les notes des éléments enfants (les UE pour les BCC et les BCC pour l'année). On considérera deux règles de calcul :

— `BCCResultCalculator` : la méthode `computeResult` retourne
  — `DEFAULTING` si au moins une note des enfants correspond à une absence ;
  — `ACQUIRED_THROUGH_CAPITALIZATION` si la moyenne des notes des enfants est supérieure à 10 ;
  — `FAILING` dans tous les autres cas.

— `YearResultCalculator` : la méthode `computeResult` retourne
  — `DEFAULTING` si au moins une note des enfants correspond à une absence ;
  — `ACQUIRED_THROUGH_CAPITALIZATION` si :
    — toutes les notes des enfants sont supérieures ou égales à 9/20 et
    — au plus une seule note d'un des enfants est strictement inférieure à 10.
  — `FAILING` dans tous les autres cas.

On va utiliser le diagramme de classes suivant :



5.6.2 Tâches

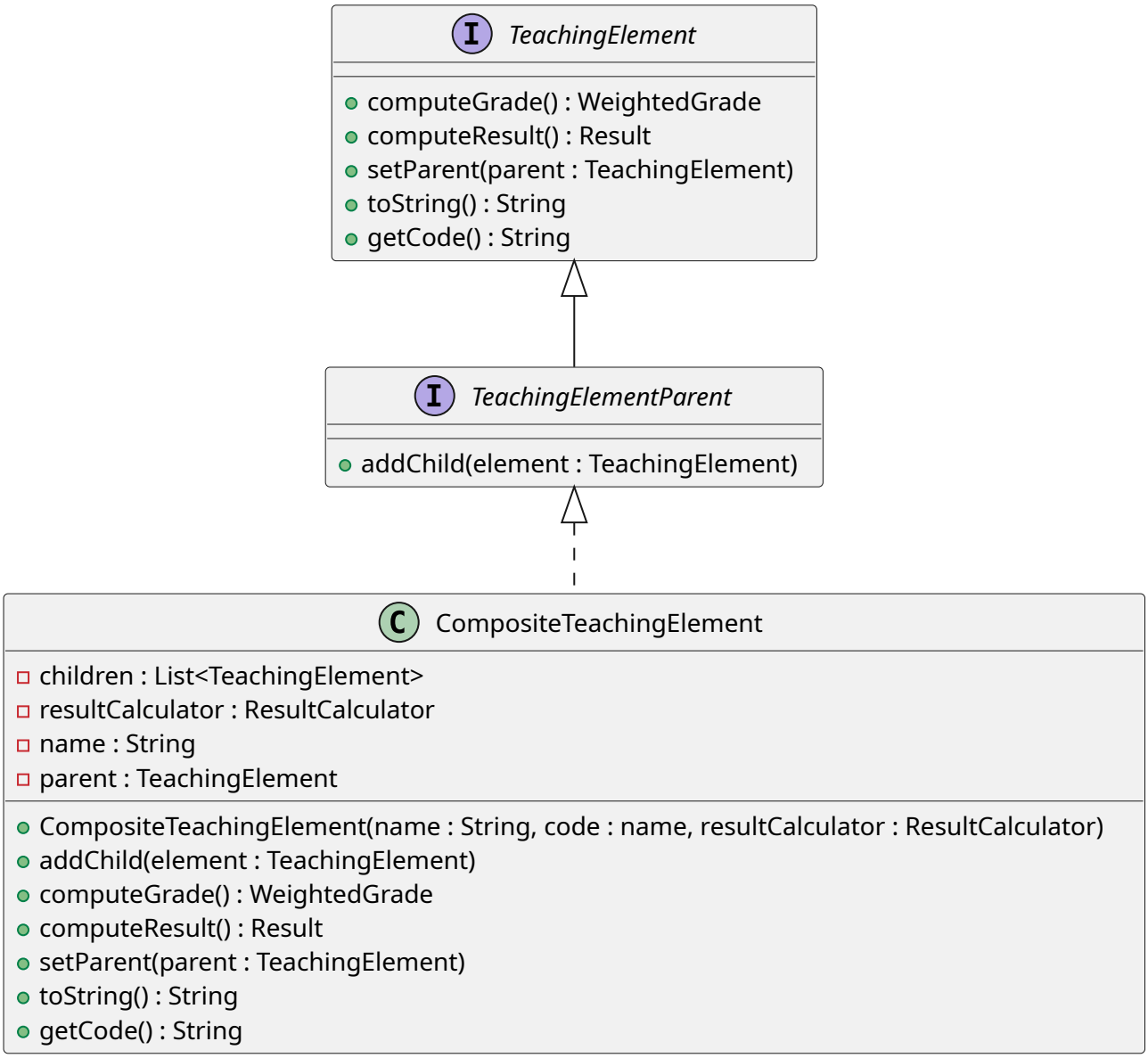
Tâche 10 : Créez l'interface `ResultCalculator` et les classes `BCCResultCalculator` et `YearResultCalculator` dans le répertoire `src/main/java` de votre projet.

Tâche 11 : Créez les classes de test nommées `BCCResultCalculatorTest` et `YearResultCalculatorTest` dans le répertoire `src/test/java` de votre projet qui devra vérifier via des tests unitaires les comportements des classes `BCCResultCalculator` et `YearResultCalculator`.

5.7 Classe `CompositeTeachingElement`

L'idée de la classe `CompositeTeachingElement` est de permettre de factoriser le code commun entre un BBC qui contiendra des UE et une année qui contiendra des BCC. Fondamentalement, la seule véritable différence entre les deux classes sera la règle de calcul de résultat qui sera délégué à un objet implémentant l'interface `ResultCalculator`.

Le diagramme de classe est le suivant :



5.7.1 Tâches

Tâche 12 : Créez l'interface `TeachingElementParent` et la classe `CompositeTeachingElement` dans le répertoire `src/main/java` de votre projet.

Tâche 13 : Créez une classe de test nommée `CompositeTeachingElementTest` dans le répertoire `src/test/java` de votre projet qui devra vérifier via des tests unitaires le comportement de la classe `CompositeTeachingElement`. Vous devez tester les comportements suivants :

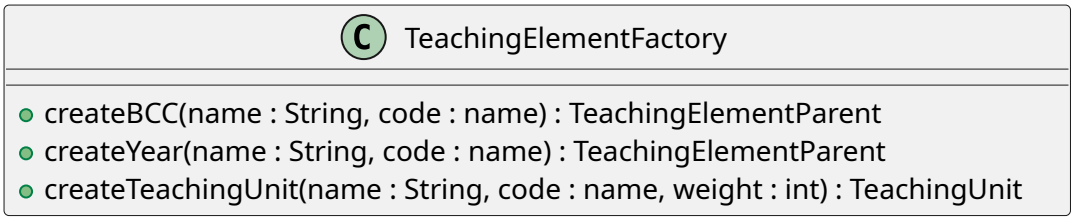
- `new CompositeTeachingElement("L3 MI", "SIN3I1", new YearResultCalculator())` crée un élément de nom `L3`, avec un code `SIN3I1` avec une règle de calcul d'année ;
- `new CompositeTeachingElement("BCC 1 (math)", "SINK10", new BCCResultCalculator())` crée un élément de nom `BCC 1 (math)`, avec un code `SINK10` avec une règle de calcul de BCC ;
- `p.addChild(c)` ajoute un élément `c` comme enfant de l'élément composite `p` et fixe le parent de `c` à `p` ;
- `p.computeGrade()` calcule la moyenne des éléments enfants de `p` ;
- `p.computeResult()` applique la règle de validation calculée par un appel de `resultCalculator` sur les notes des enfants de `p` en ajoutant comme règle que si le parent de `p` est validé (résultat `ACQUIRED_THROUGH_COMPENSATION` ou `ACQUIRED_THROUGH_CAPITALIZATION`) et que le `resultCalculator` de `p` donne un résultat ajourné alors le résultat est `ACQUIRED_THROUGH_COMPENSATION`.

5.8 Classe `TeachingElementFactory`

L'idée de la classe `TeachingElementFactory` est de regrouper dans une seule classe les méthodes de création pour les éléments d'une formation :

- `createBCC` permet de créer un BCC, c'est-à-dire un `CompositeTeachingElement` avec un `BCCResultCalculatorTest` ;
- `createYear` permet de créer une année, c'est-à-dire un `CompositeTeachingElement` avec un `YearResultCalculatorTest` ;
- `createTeachingUnit` crée une unité d'enseignement.

Le diagramme de classe est le suivant :

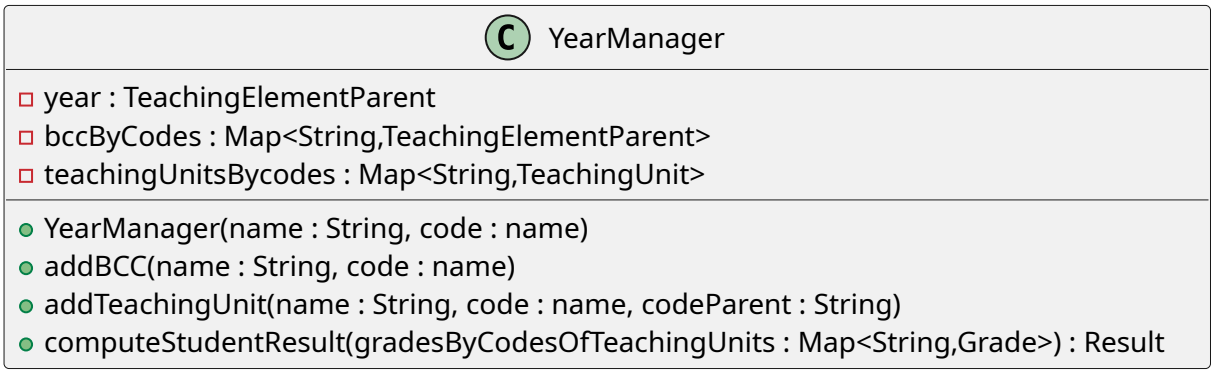


Tâche 14 : Créez la classe `TeachingElementFactory` dans le répertoire `src/main/java` de votre projet.

Tâche 15 : Créez une classe de test nommée `TeachingElementFactoryTest` dans le répertoire `src/test/java` de votre projet qui devra vérifier via des tests unitaires le comportement de la classe `TeachingElementFactory`.

5.9 Classe `YearManager`

L'idée de la classe `YearManager` est de permettre une gestion facile d'une année d'un diplôme. L'idée est de stocker les éléments (BCC et UE (`TeachingUnit`)) de la formation grâce à des `Map` avec comme clé les code des éléments. Le but de la classe est de stocker l'architecture de la formation et de permettre de calculer le résultat d'un étudiant à partir d'un `Map` qui associe chaque code d'UE à une note.



Tâche 16 : Créez la classe `YearManager` dans le répertoire `src/main/java` de votre projet.

5.10 Classe `Main`

Tâche 17 : Utiliser la classe `YearManager` pour calculer dans la méthode `main` de la classe `Main` de votre projet le résultat d'un étudiant suivant une année de L3 double licence mathématiques-informatique qui a la structure suivante :

- BCC 1 (math) :
  - Algèbre linéaire (6 crédits)
  - Suites et séries de fonctions (9 crédits)
  - Théorie des groupes (3 crédits)
  - Topologie (6 crédits)
  - Structures algébriques (6 crédits)
  - Sémantique (3 crédits)
- BCC 2 (info) :
  - Initiation génie logiciel (3 crédits)
  - Algorithmique 2 (6 crédits)
  - Logique (6 crédits)
  - Calculabilité (3 crédits)
  - Compilation (3 crédits)
  - Développement durable (3 crédits)
  - Introduction à l'apprentissage automatique (3 crédits)
- BCC 3 (transverse)
  - PPPE (3 crédits)
  - Anglais S5 (3 crédits)
  - Épreuve intégrative S5 (3 crédits)

— Stage (6 crédits)
— Anglais S6 (3 crédits)
— Épreuve intégrative S6 (3 crédits)