

Gestion des notes des étudiants

Introduction

Le but de ce TP est de créer des classes permettant de représenter des étudiants (classe **Student**), des notes (classe **Grade**), des résultats à une unité d'enseignement (classe **TeachingUnitResult**) et des promotions d'étudiants (classe **Cohort**).

L'objectif pour vous est de vous initier :

- à l'utilisation d'un IDE : IntelliJ IDEA de JetBrains est mis en avant mais n'importe quel IDE équivalent pourra être utilisé,
- à la gestion de version grâce à git,
- à l'utilisation des tests (ici déjà programmé) pour valider votre programme.

Vous allez sans doute découvrir un certain nombre de nouveaux outils durant ce TP. Le temps d'apprentissage de ces outils peut être au début frustrant mais ils vont vous faire gagner du temps par la suite.

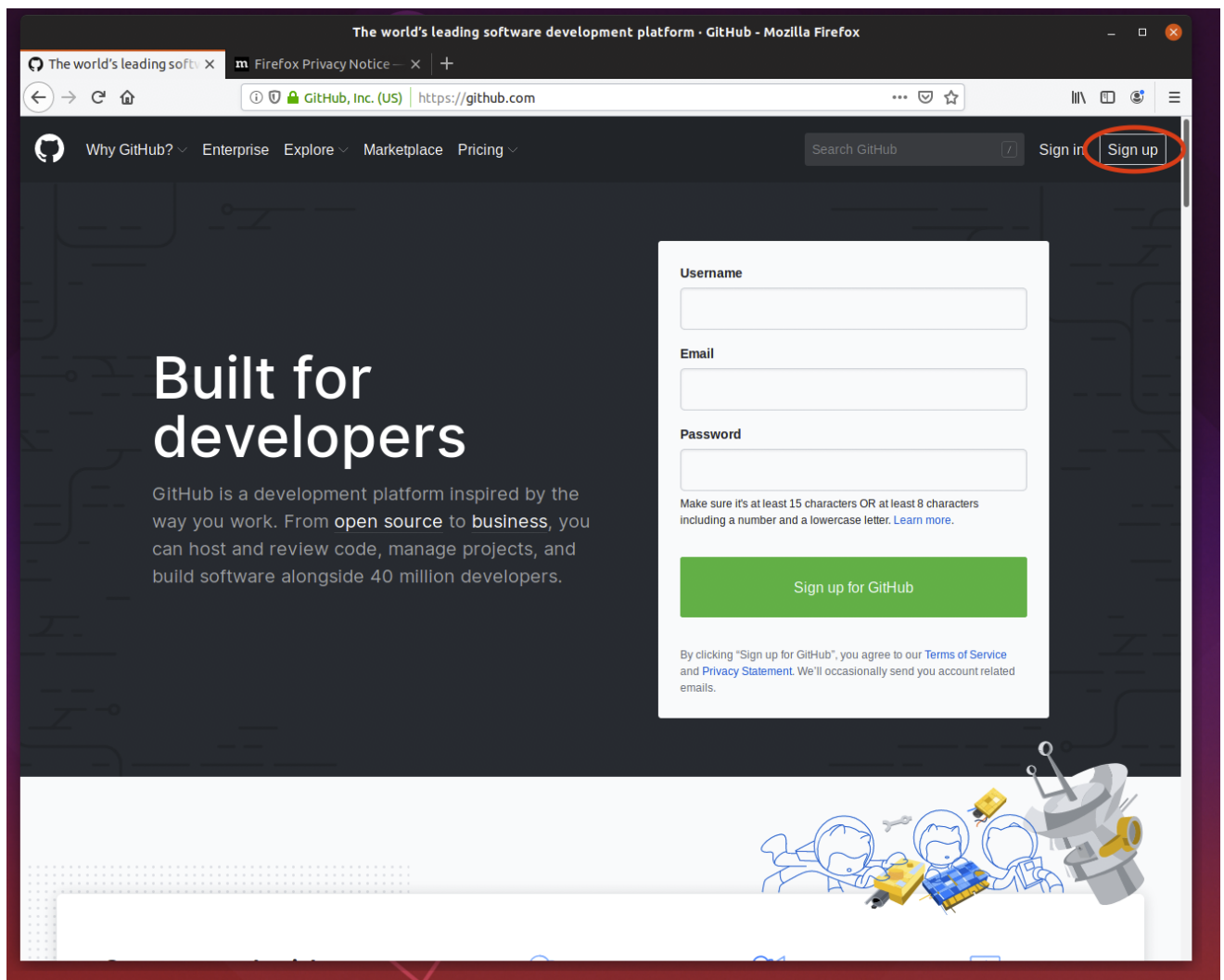
Mise en place du TP

Créer le dépôt git pour le TP

Ce TP est l'occasion d'apprendre à utiliser des outils de développement professionnel. En plus de l'IDE IntelliJ, vous aurez à utiliser un outil de gestion de versions, en l'occurrence **git**. Un tel outil permet d'enregistrer à distance votre projet et de permettre à plusieurs personnes de travailler simultanément, sans devoir échanger les fichiers par courriel, et sans se marcher sur les pieds (par exemple modification sans le savoir du même fichier).

La première étape pour utiliser la gestion de version est de vous inscrire à github et de créer un dépôt à l'aide de github classroom.

1. Si vous ne possédez pas de compte Github, aller sur github et créez-vous un compte gratuit. Pour cela, allez sur <https://github.com> et cliquez sur le bouton **signup**.



Remplissez le formulaire avec votre login, adresse email et mot de passe. Compléter le test pour vérifier que vous n'êtes pas une machine et finissez en appuyant sur le bouton create account.

Join GitHub · GitHub · Mozilla Firefox

Join GitHub · GitHub × Firefox Privacy Notice ×

GitHub, Inc. (US) | https://github.com/join?source=header-home

Why GitHub? Enterprise Explore Marketplace Pricing

Search GitHub

Sign in Sign up

Join GitHub

The best way to design, build, and ship software.

 **Step 1:**
Set up your account

 **Step 2:**
Choose your subscription

 **Step 3:**
Tailor your experience

Create your personal account

Username *

This will be your username. You can add the name of your organization later.

Email address *

We'll occasionally send updates about your account to this inbox. We'll never share your email address with anyone.

Password *

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

You'll love GitHub

Unlimited public repositories

Unlimited private repositories

✓ Limitless collaboration

✓ Frictionless development

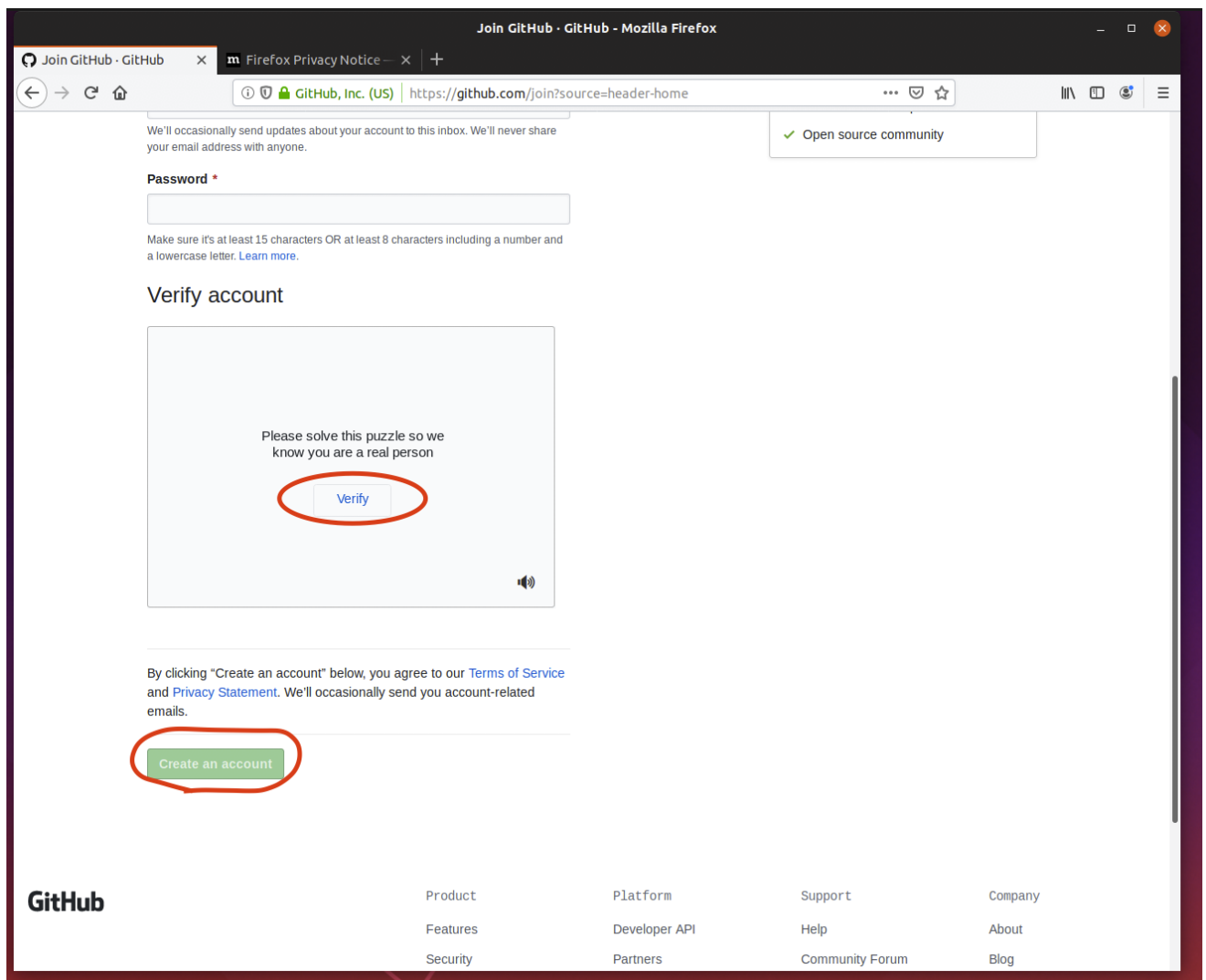
✓ Open source community

Verify account

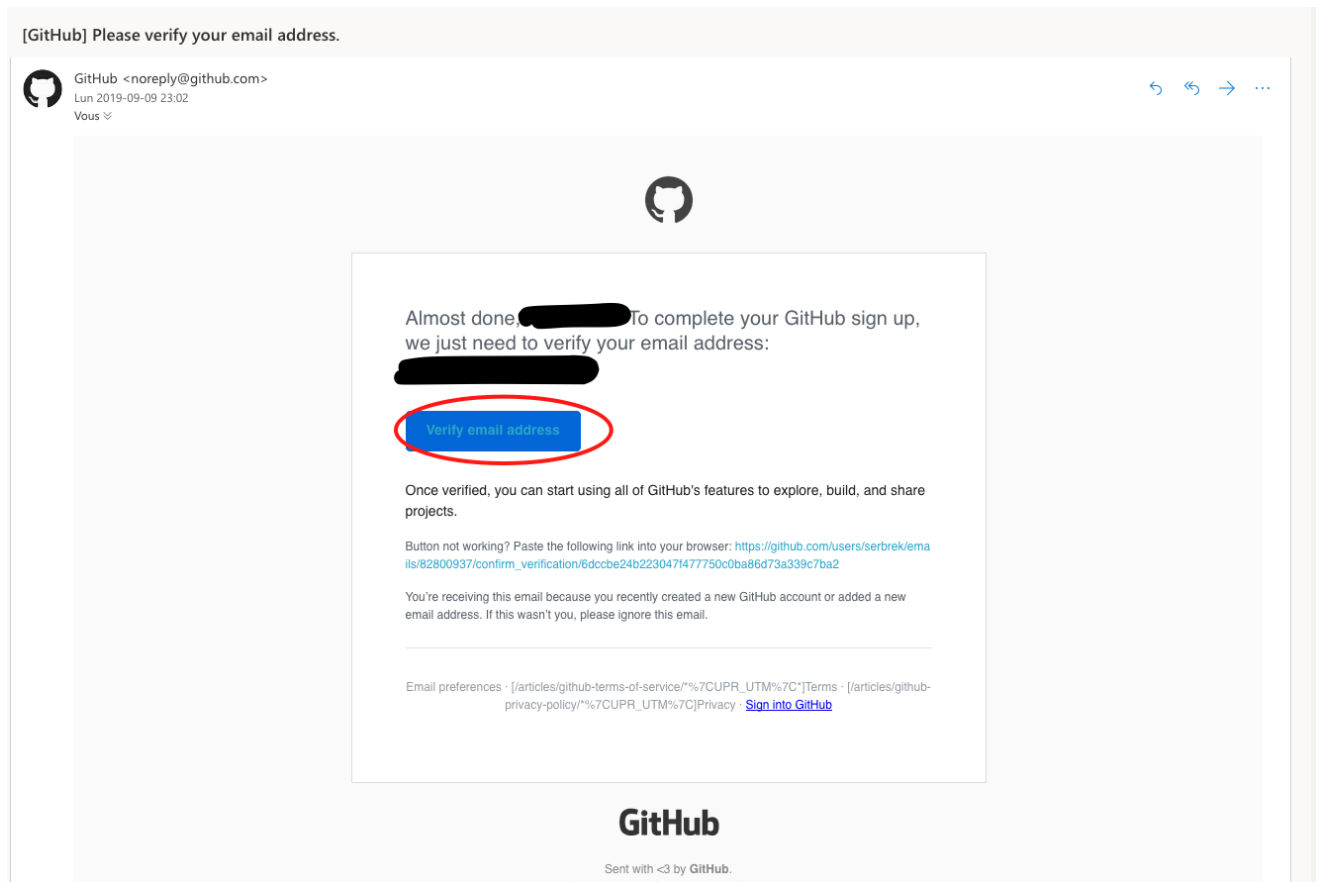
Please solve this puzzle so we know you are a real person

Verify

3



Vous allez recevoir un mail sur l'adresse que vous avez indiquée. Cliquez sur le bouton `verify email address` pour valider votre adresse.



2. Vous allez maintenant créer un dépôt grâce à github classroom. Pour cela, il vous suffit d'aller à l'adresse suivante : <https://classroom.github.com/g/vjEBS5Ee> en étant connecté sur github à votre compte et d'accepter le devoir en cliquant sur le bouton **accept this assignment**.

GitHub Classroom

GitHub Education

Programmation 2 (2019-2020)
L2InfoAMU

 **Accept the students assignment**

Accepting this assignment will give your team (labourel) access to the **students-labourel** repository in the [@L2InfoAMU](#) organization on GitHub.

Accept this assignment

On vous demande maintenant de cliquer sur votre adresse étudiante (adresse @etu.univ-amu.fr) afin que vous soyez clairement identifié sur **github**. C'est très important car certains des TP que vous ferez par la suite sur **github** seront notés.

L2InfoAMU



Identifiers

arnaud.labourel@univ-amu.fr

arnaud.labourel@univ-amu.fr

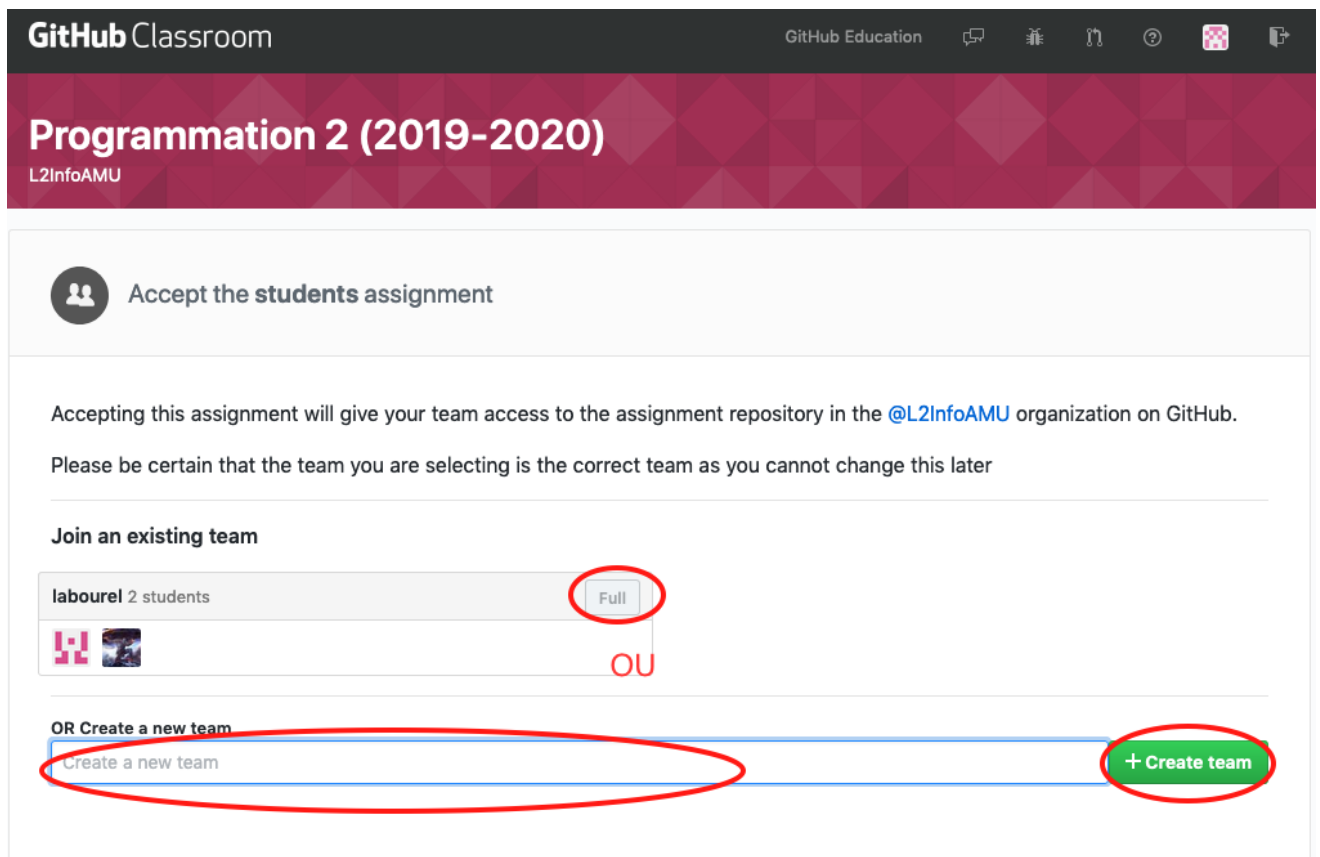
[REDACTED]

[REDACTED]

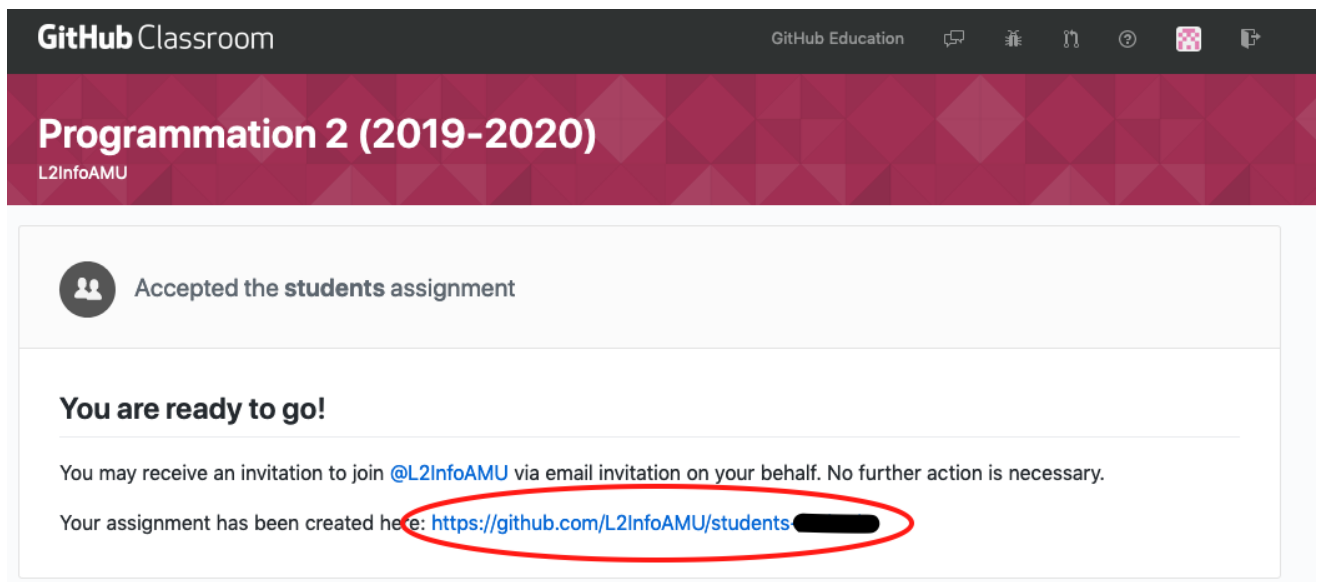
[REDACTED]

Skip

7



Après cette étape, github va créer un dépôt à partir des sources. Vous devriez avoir accès à l'écran suivant et il vous suffit de cliquer sur le lien.



- Maintenant vous devriez être sur la page de votre dépôt. Cliquez sur le bouton clone or download.

L2InfoAMU / students
generated from laboure/students_template

Private

Watch 4Star 0Fork 0

<> CodeIssues 0Pull requests 0Projects 0WikiSecurityInsightsSettings

students-serbrek created by GitHub Classroom

Manage topics

1 commit1 branch0 releases1 contributor

Branch: masterNew pull requestCreate new fileUpload filesFind FileClone or download

bestellon Initial commitLatest commit 9e0ef53 12 minutes ago

.idea	Initial commit	12 minutes ago
gradle/wrapper	Initial commit	12 minutes ago
src	Initial commit	12 minutes ago
.gitignore	Initial commit	12 minutes ago
build.gradle	Initial commit	12 minutes ago
gradlew	Initial commit	12 minutes ago
gradlew.bat	Initial commit	12 minutes ago
settings.gradle	Initial commit	12 minutes ago

Help people interested in this repository understand your project by adding a README.

Add a README

© 2019 GitHub, Inc. Terms Privacy Security Status Help

Contact GitHub Pricing API Training Blog About

Cela va vous donner l'accès au lien vous permettant de récupérer votre dépôt. Notez bien ce lien. Vous pouvez le copier (le sauver dans le presse-papier) en cliquant sur le bouton à côté.

[Read the guide](#)

L2InfoAMU / **students-serbrek** Private
Watch 4
Star 0
Fork 0

[Code](#)
[Issues 0](#)
[Pull requests 0](#)
[Projects 0](#)
[Wiki](#)
[Security](#)
[Insights](#)
[Settings](#)

students-serbrek created by GitHub Classroom [Edit](#)

[Manage topics](#)

1 commit
1 branch
0 releases
1 contributor

Branch: master
New pull request
Create new file
Upload files
Find File
Clone or download

bestellon Initial commit		
<code>.idea</code>	Initial commit	
<code>gradle/wrapper</code>	Initial commit	
<code>src</code>	Initial commit	
<code>.gitignore</code>	Initial commit	
<code>build.gradle</code>	Initial commit	13 minutes ago
<code>gradlew</code>	Initial commit	13 minutes ago
<code>gradlew.bat</code>	Initial commit	13 minutes ago
<code>settings.gradle</code>	Initial commit	13 minutes ago

Help people interested in this repository understand your project by adding a README.
Add a README

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

<https://github.com/L2InfoAMU/students-serbrek>

[Open in Desktop](#)
[Download ZIP](#)

Use SSH

IDE IntelliJ

Environnement de développement (IDE)

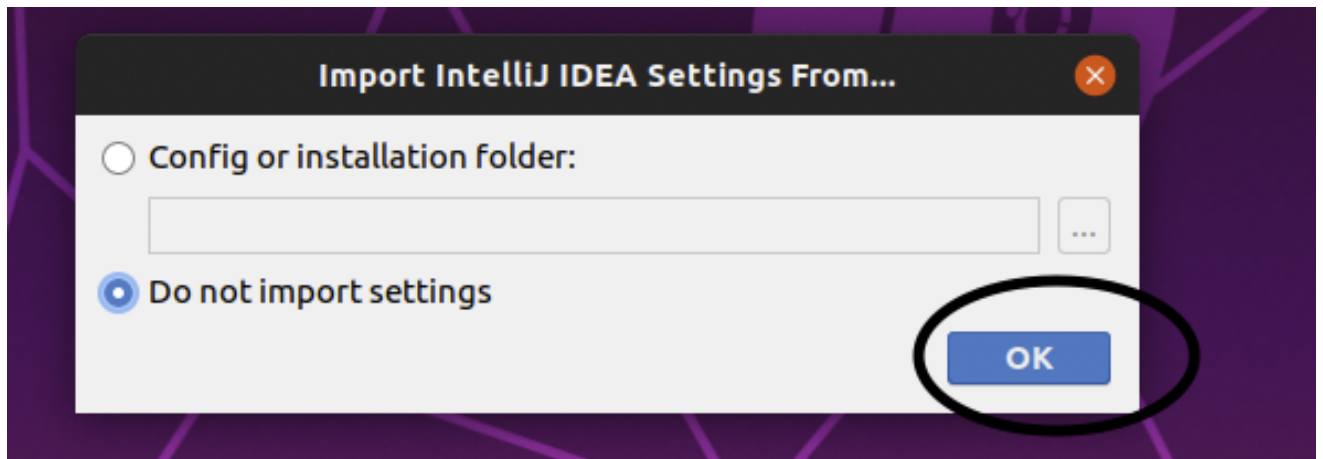
Afin de programmer dans ce cours, nous allons utiliser un environnement de développement (Integrated Development Environment : IDE). Il existe de nombreux IDE pour Java. Dans ce cours, nous vous conseillons d'utiliser IntelliJ IDEA de JetBrains mais vous pouvez aussi utiliser Eclipse netbeans ou un autre IDE si vous êtes familier avec ceux-ci.

Lancement de l'IDE

Pour lancer le logiciel sur les machines de l'université, il vous faut aller dans le Menu (en haut à gauche de l'écran) et cliquer sur 'IntelliJ IDEA 2019.2' dans la partie programmation du menu.

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

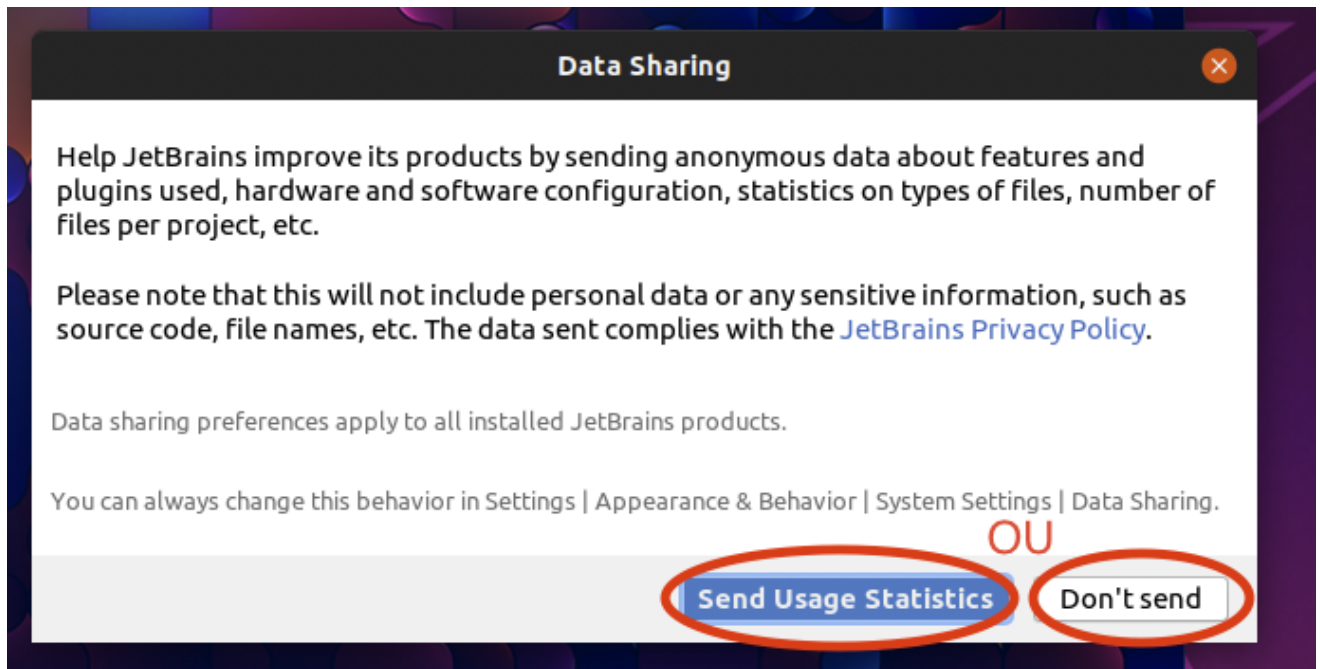
1. Laissez sur 'Do not import settings' et cliquez sur 'OK'.



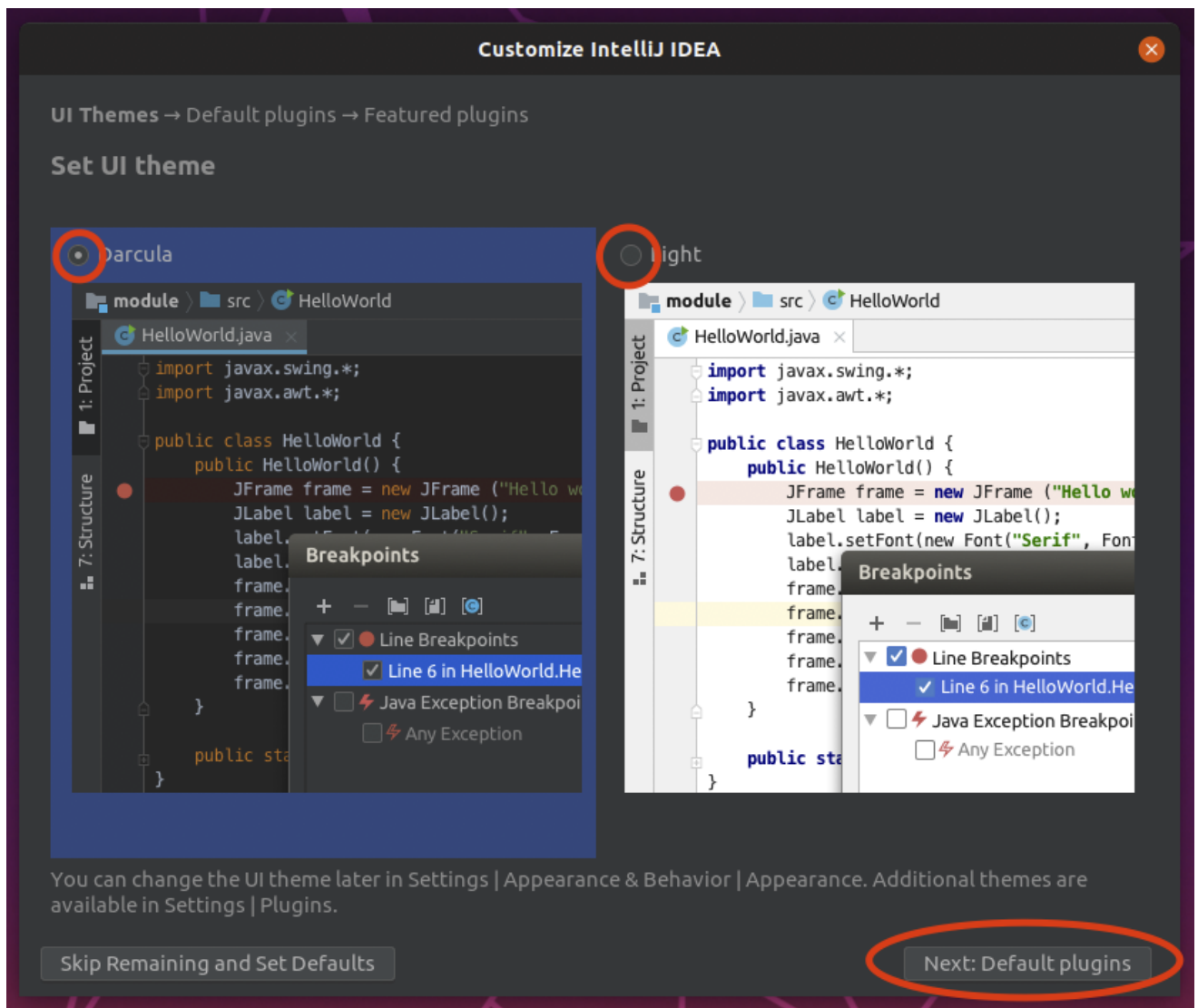
2. Cochez la case comme quoi vous acceptez les termes d'utilisation d'IntelliJ IDEA puis sur continue.



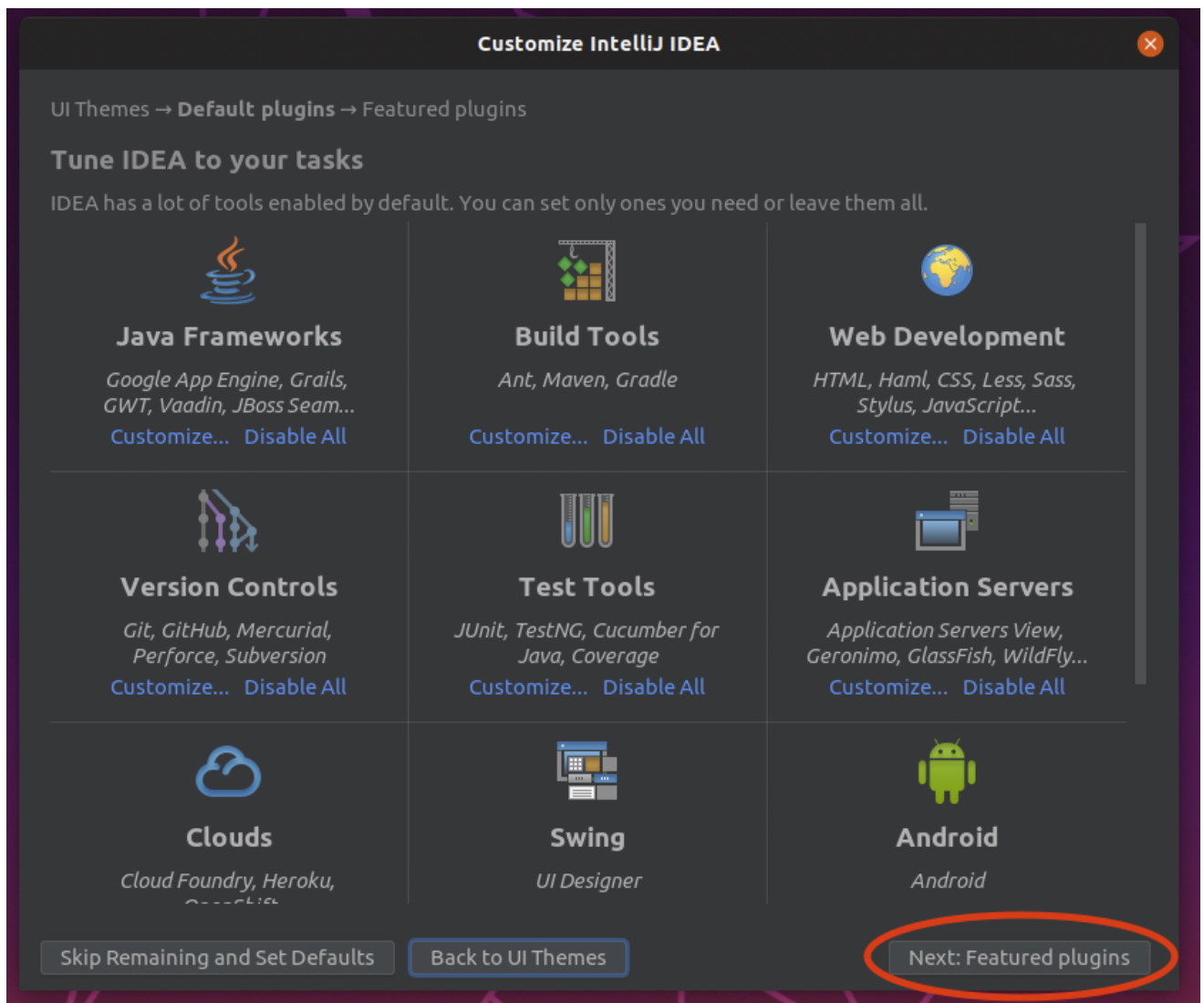
3. Choisissez si vous acceptez ou non d'envoyer les données d'utilisation du logiciel à JetBrains. À partir de là vous allez personnaliser votre installation pour qu'elle corresponde à vos besoins.

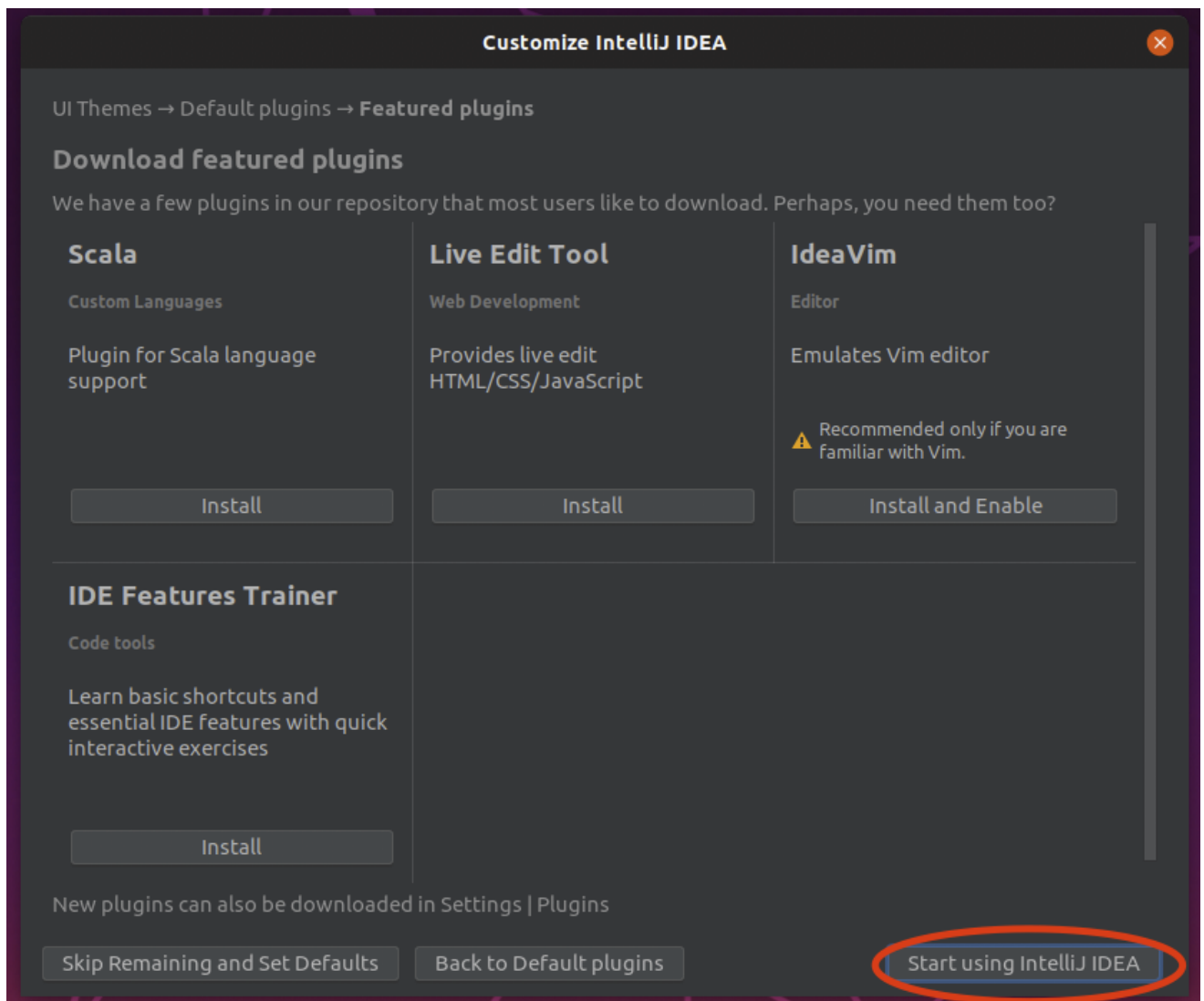


4. Sur la première fenêtre de personnalisation, vous devez choisir l'apparence de votre IDE. Ici rien de fondamental, vous pouvez choisir l'option que vous voulez entre **darcula** (texte blanc sur fond noir) ou **light** (texte noir sur fond clair). Cliquez sur next quand vous avez choisi.



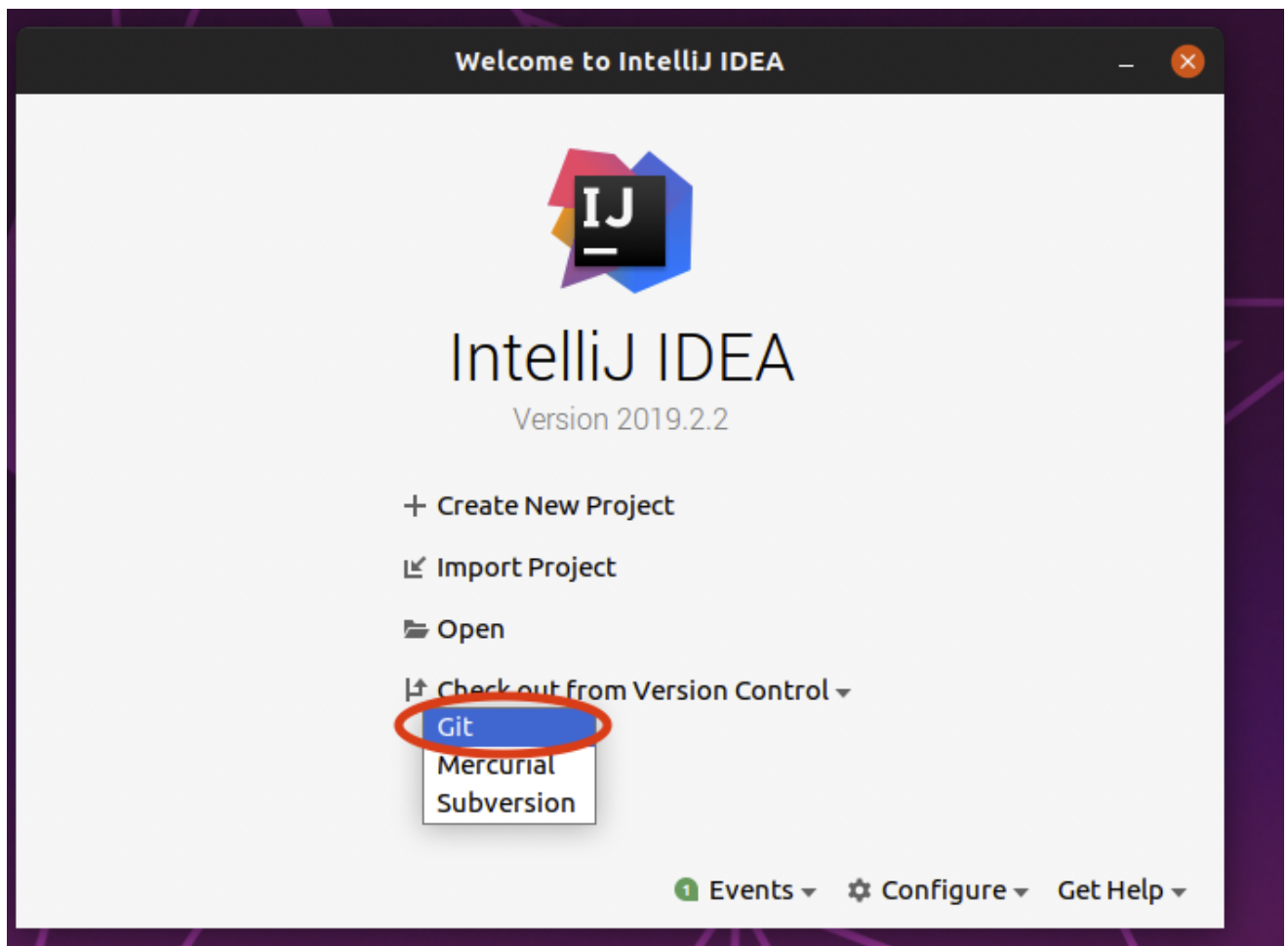
5. Les deux écrans suivants vous permettent de choisir les plugins que vous allez activer. De manière générale, il vaut mieux en activer le moins possible de plugins pour éviter les mauvaises surprises. Vous pouvez donc cliquer sur **next** puis **start**



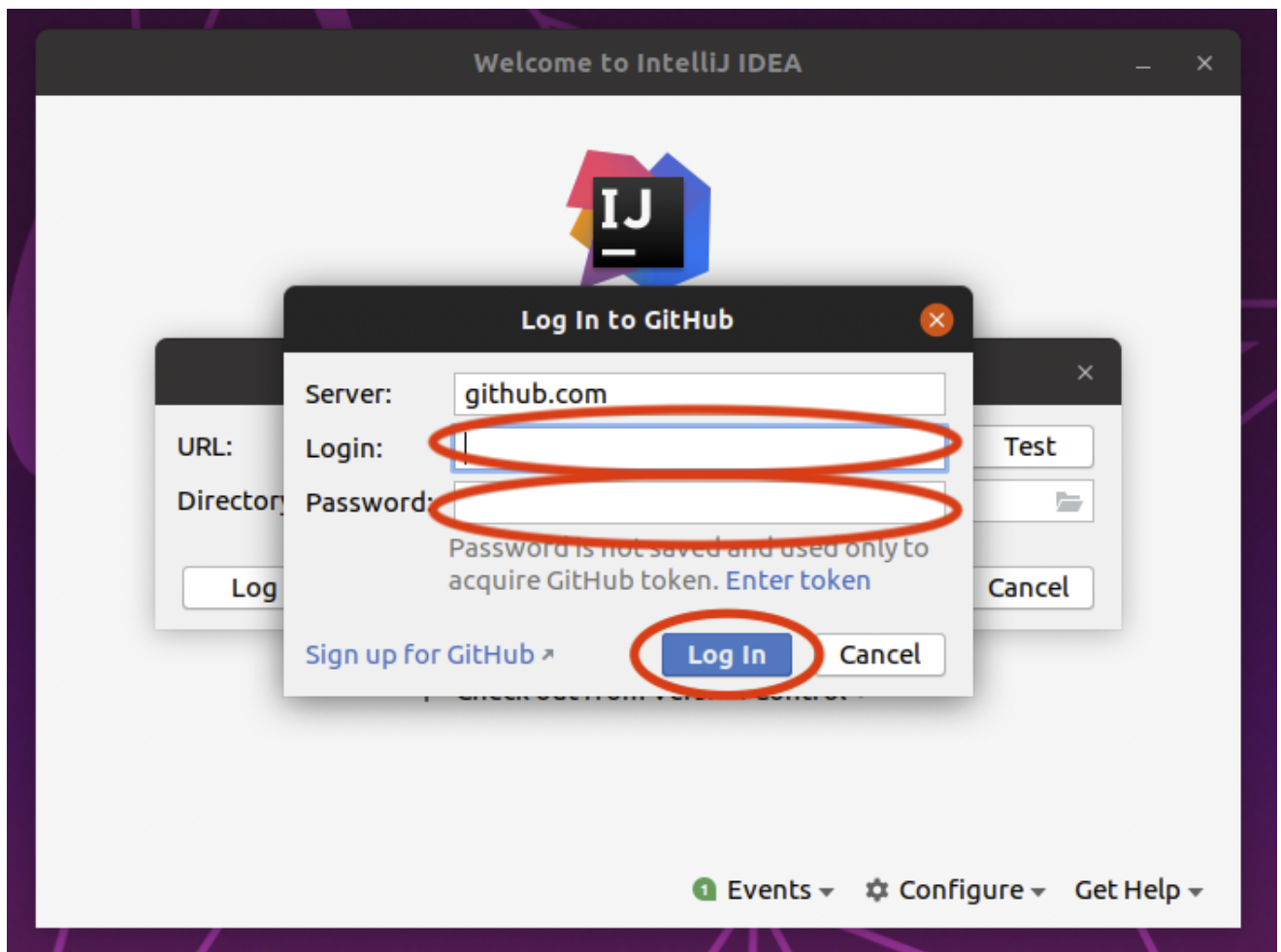


6. Vous devriez accéder à l'écran d'accueil d'IntelliJ IDEA. Vous allez récupérer le contenu de votre dépôt. Pour cela, cliquez sur `Check out from version control` puis sur `git` dans le menu qui apparaît.

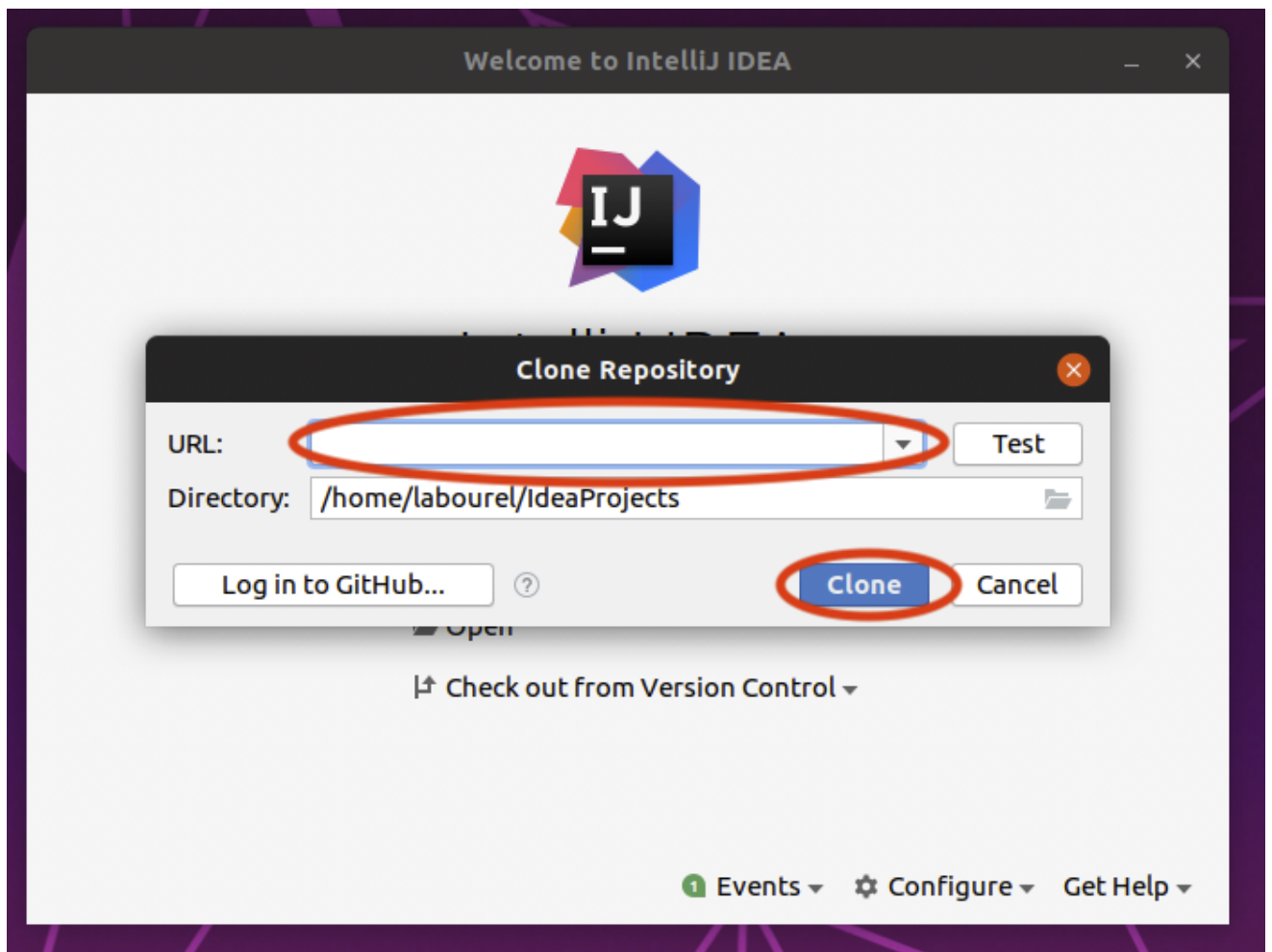




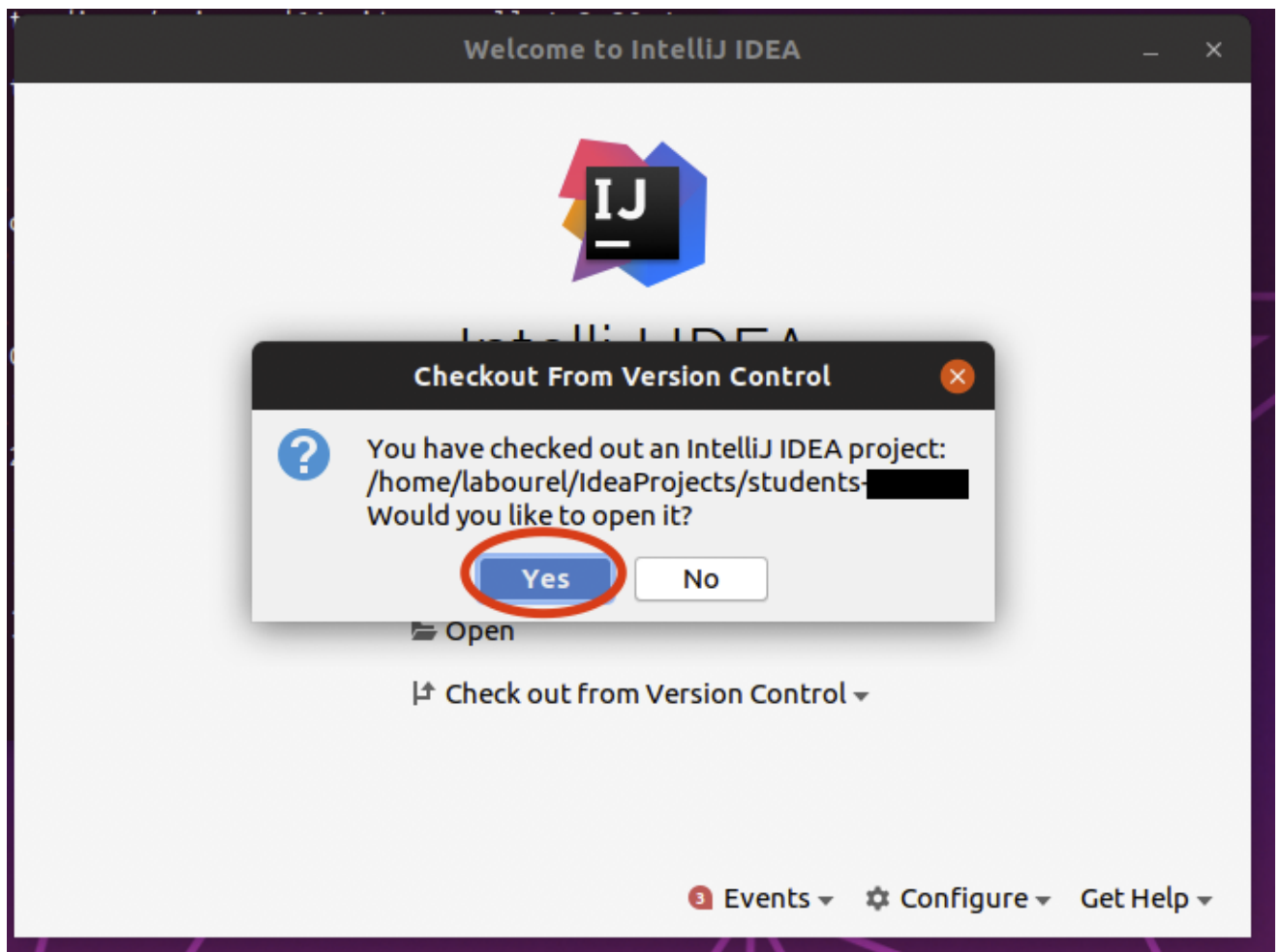
7. Vous allez maintenant récupérer votre dépôt (en fait le cloner). Pour cela, connectez-vous à votre compte github en cliquant sur le bouton `log in to github`. Vous devriez obtenir la fenêtre suivante dans laquelle vous devez remplir votre identifiant et mot de passe de votre compte github puis cliquer sur `Log in`.



Entrez dans le champs URL, l'adresse de votre dépôt (celle que vous récupérer sur la page web de votre dépôt en cliquant sur le bouton `clone` or `download`) puis cliquez sur le bouton `clone`.



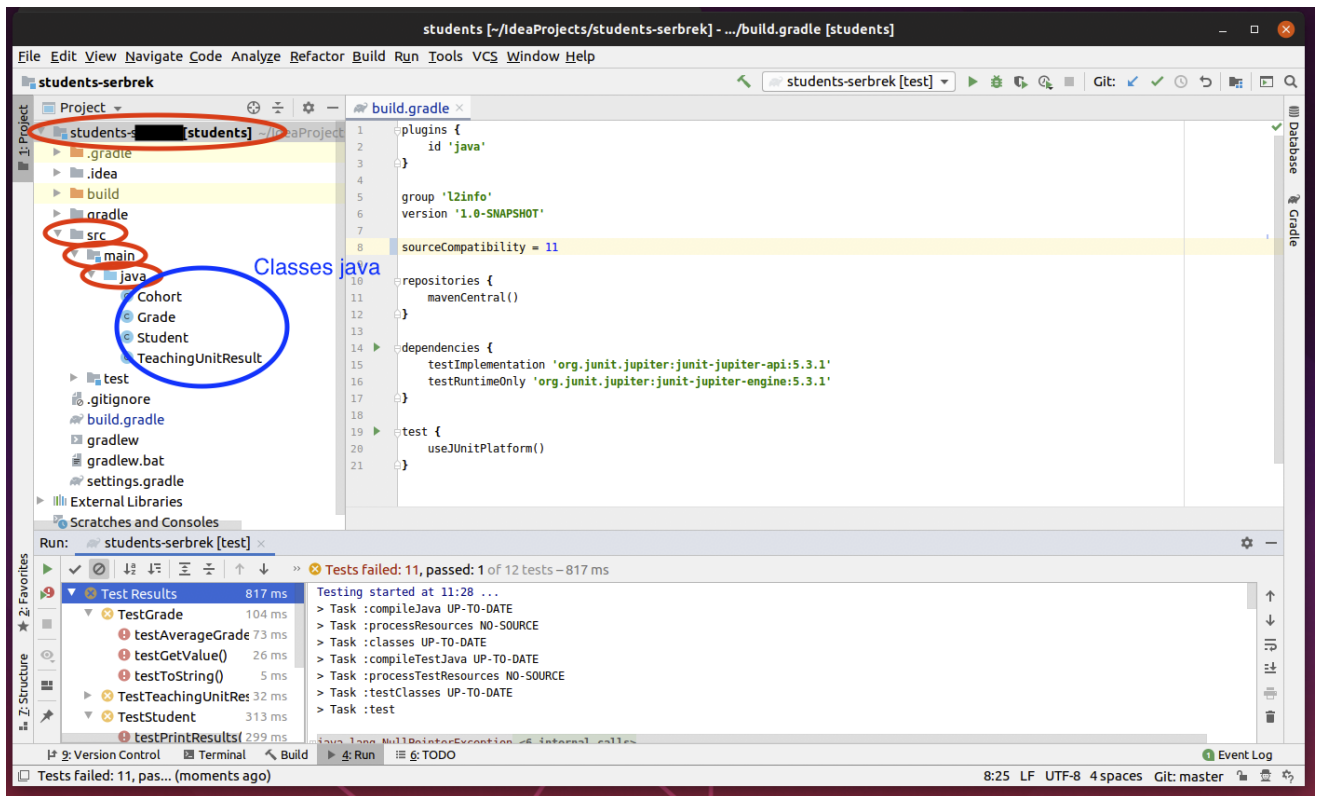
8. Après quelques instants de chargement, on vous demandera si vous souhaitez ouvrir votre projet. Il faut bien évidemment cliquer sur le bouton yes.



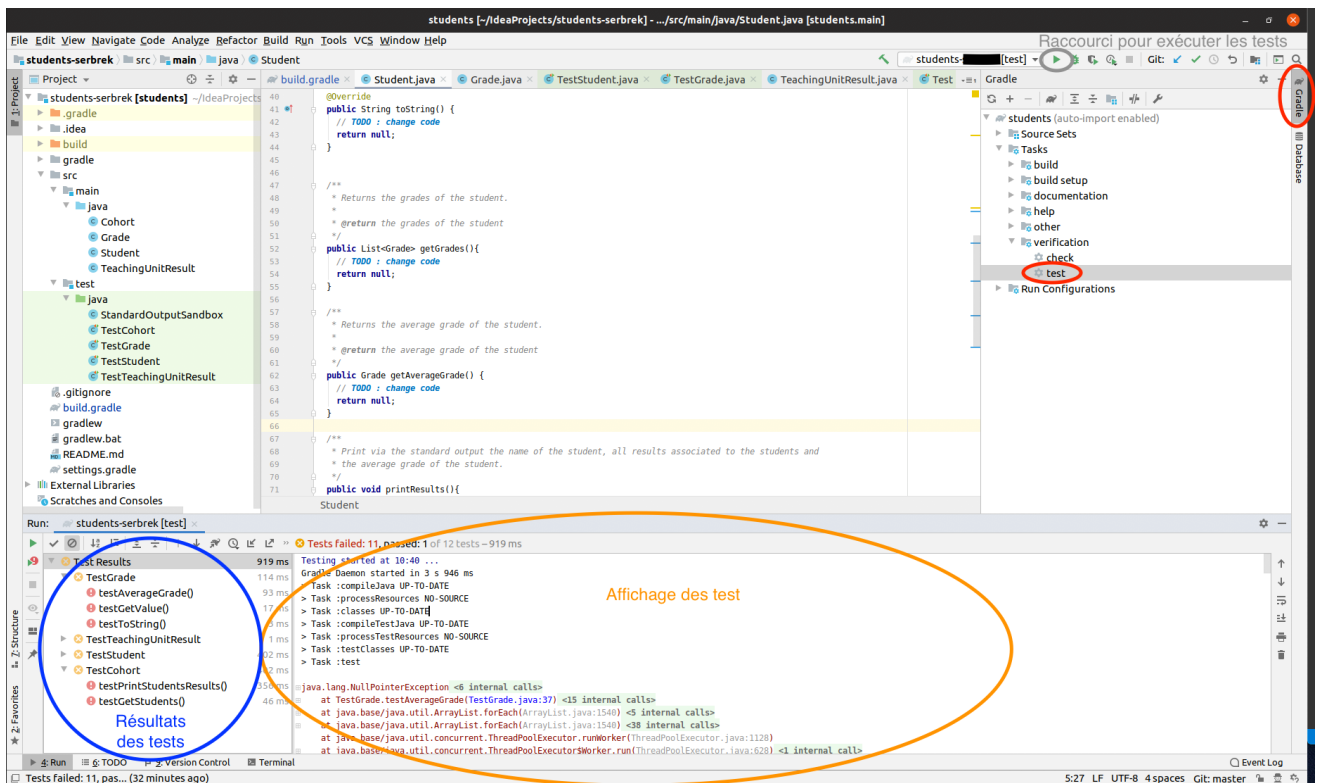
9. Une fois le projet ouvert, vous pouvez fermer la fenêtre `tip of the day`. Il vous faut autoriser l'import gradle en cliquant sur le pop up en bas à droite de votre fenêtre.

Prise en main d'IntelliJ IDEA

1. Pour accéder aux fichiers `java` de votre dépôt il vous faut naviguer dans l'arborescence du projet. Les fichiers que vous devez modifier sont dans `students-*** -> src -> main -> java` (il faut cliquer sur les triangles pour déplier l'arborescence).



2. Pour compiler et exécuter le programme, il faut passer par l'onglet gradle à droite et cliquer deux fois sur students -> Tasks -> verification -> test. Cela va compiler et exécuter les tests. Pour le moment, les tests ne passeront pas car certaines classes sont incomplètes.



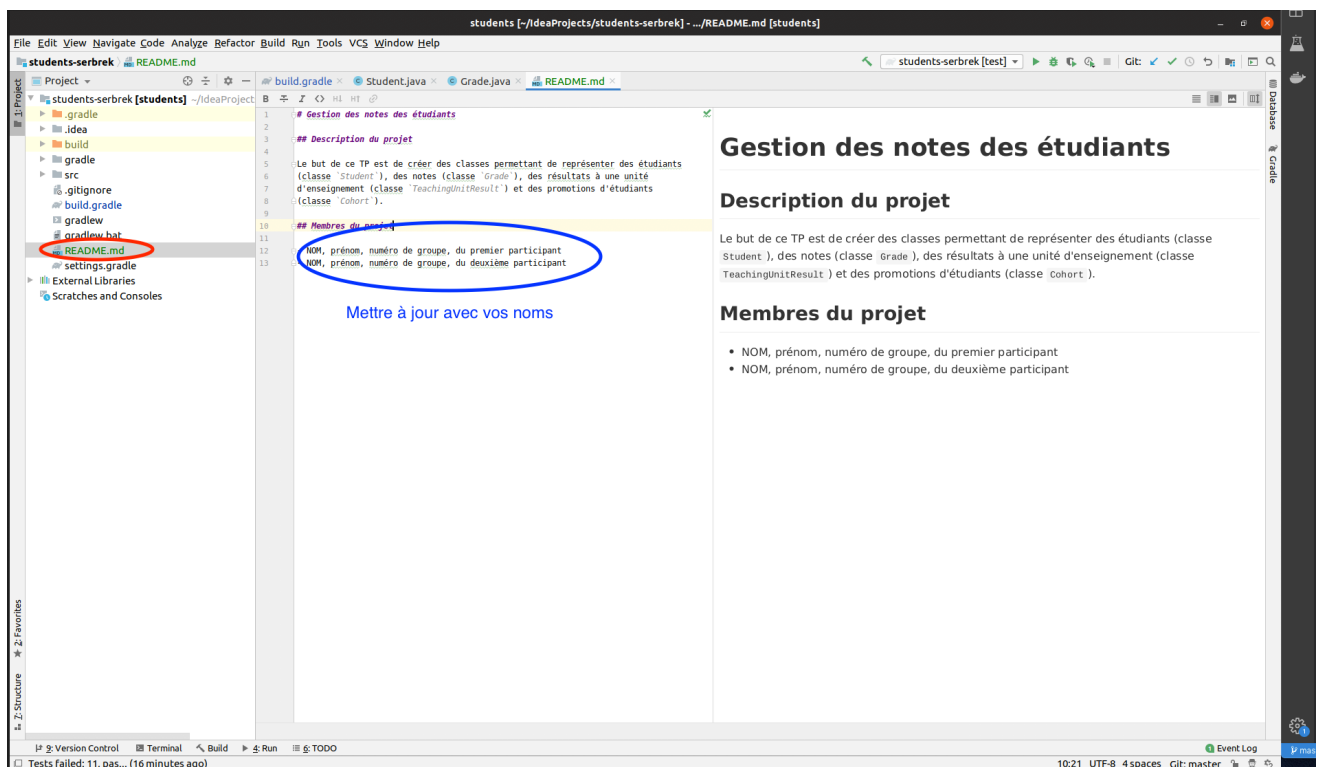
Git

Principe de git

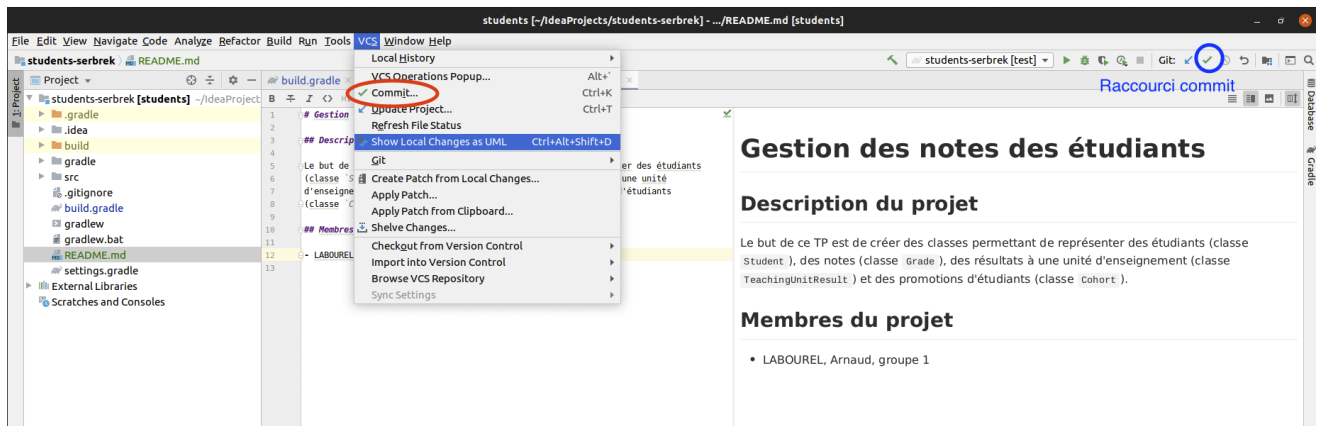
Le principe de `git` est d'avoir un *dépôt* distant : une version de votre projet stockée sur un serveur accessible par Internet (en l'occurrence hébergé par `github`). Vous disposez en plus de dépôts locaux sur les ordinateurs sur lesquels vous travaillez. Vous faites vos modifications sur votre ordinateur, et lorsque vous avez accompli une amélioration qui fonctionne bien, vous pouvez la faire enregistrer (`commit`) par `git`. Ces enregistrements sont locaux à votre ordinateur, et vous pouvez en faire autant que vous le souhaitez. Si vous voulez partager votre travail avec votre équipe, il vous faut l'envoyer vers le dépôt distant (`push`). À l'inverse, si vous souhaitez récupérer le travail fait par vos coéquipiers, il faut ramener ces modifications depuis le dépôt distant (`pull`). IntelliJ est capable de gérer `git`; vous trouverez dans le menu `VCS` l'option `Commit`, et l'option `Git` qui contient `push` et `pull`.

Première modification de votre dépôt

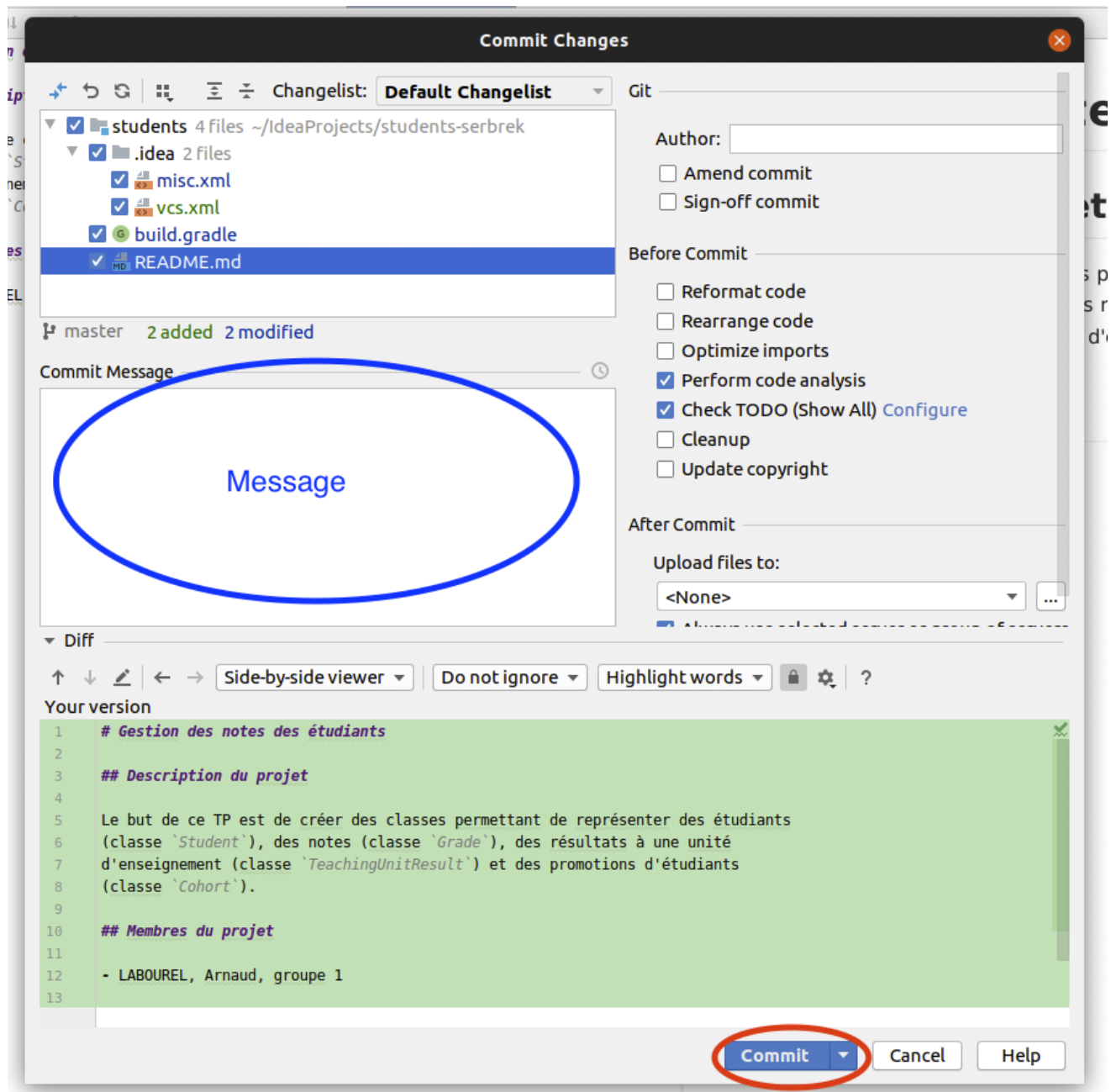
1. Modifiez le fichier `README.md`. Mettez votre nom ainsi que le nom de votre éventuel coéquipier (si vous êtes seul enlever la deuxième ligne de participants).



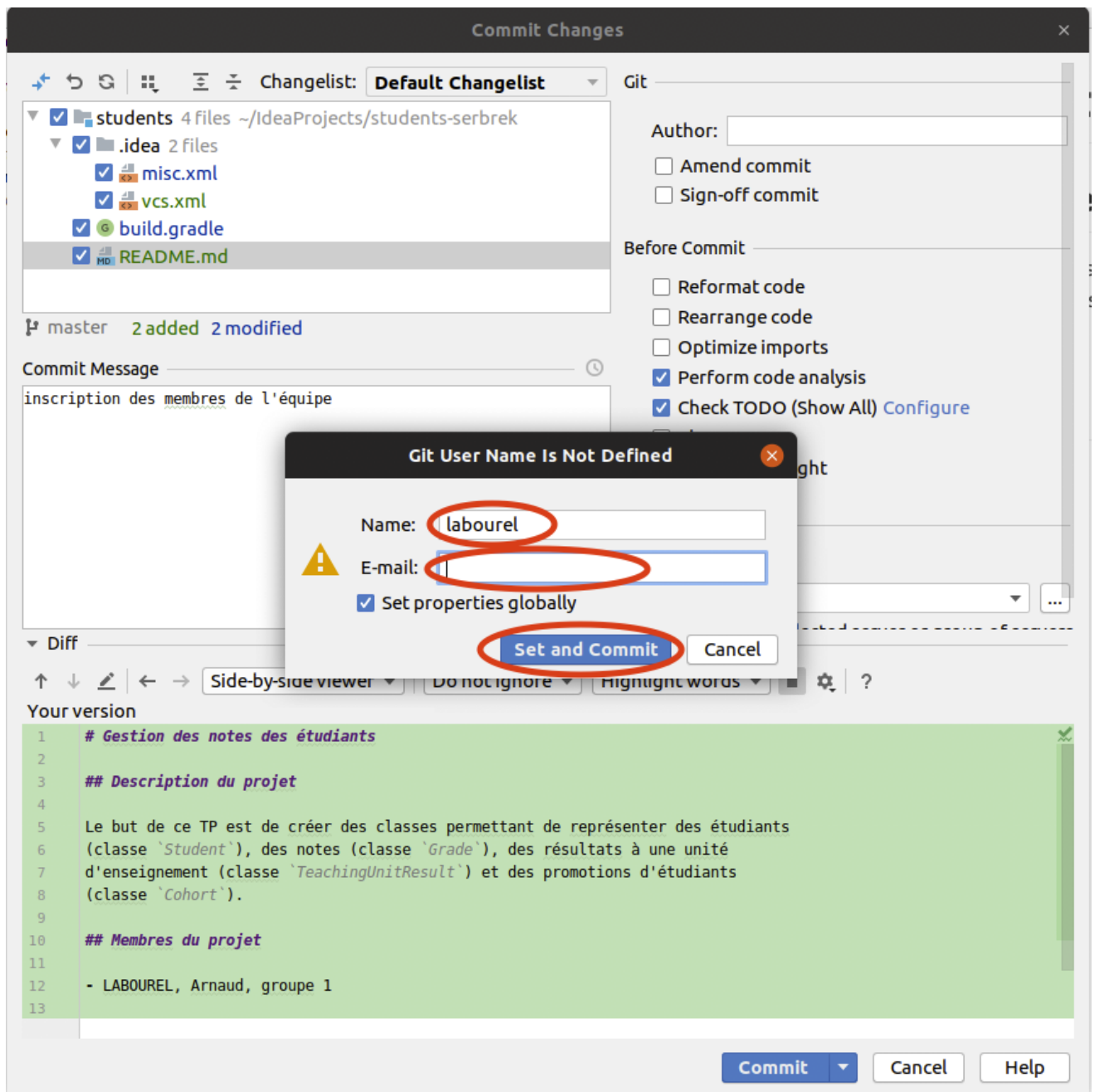
2. Vous allez maintenant mettre à jour votre dépôt local (celui sur votre machine) en effectuant un `commit`. Pour cela vous pouvez aller dans le menu `VCS` -> `commit` ou bien cliquer sur le raccourci en haut à droite de votre fenêtre.



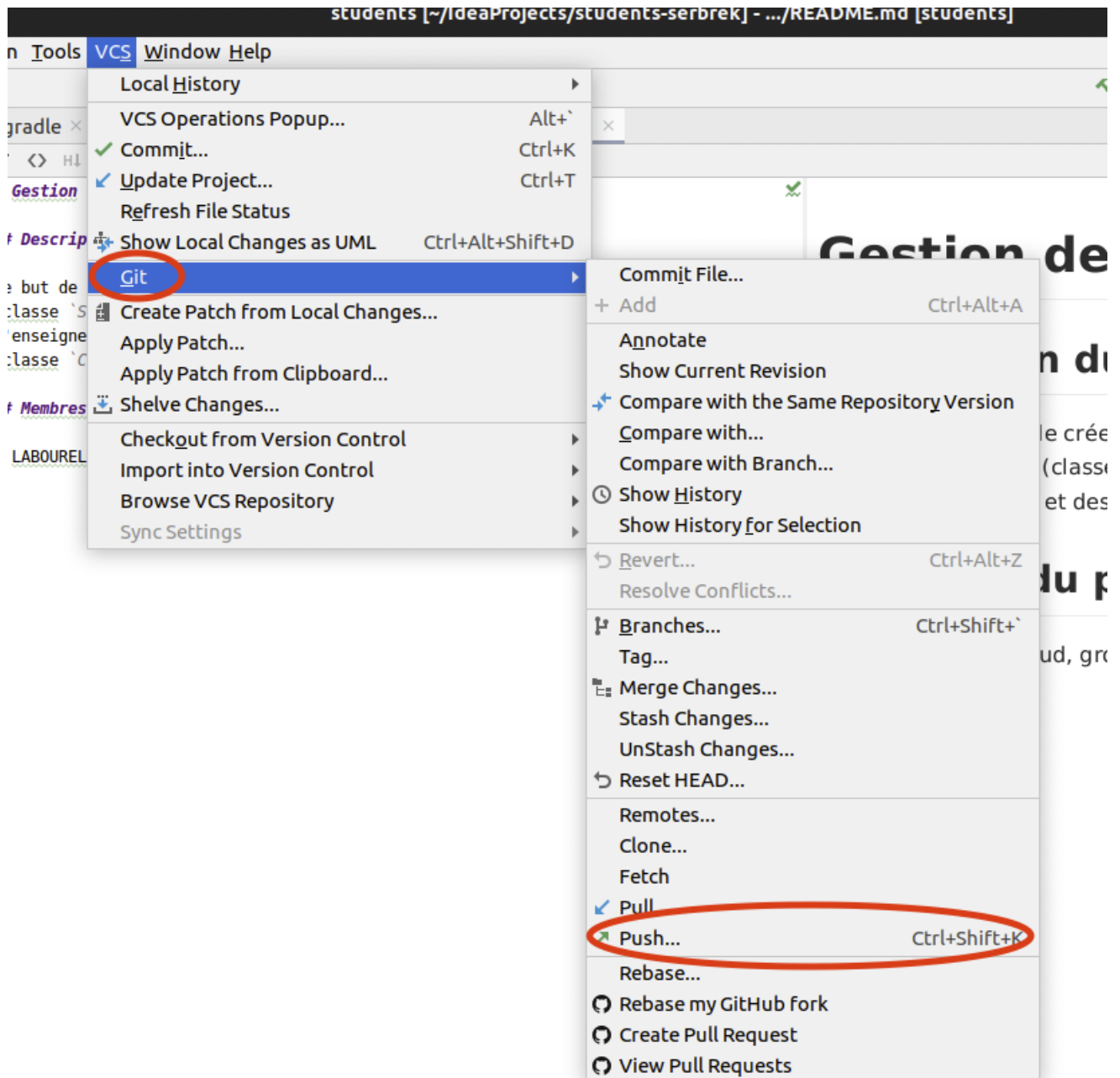
Faites un commit avec pour message “inscription des membres de l’équipe” en cliquant sur commit après avoir rempli le champs message.



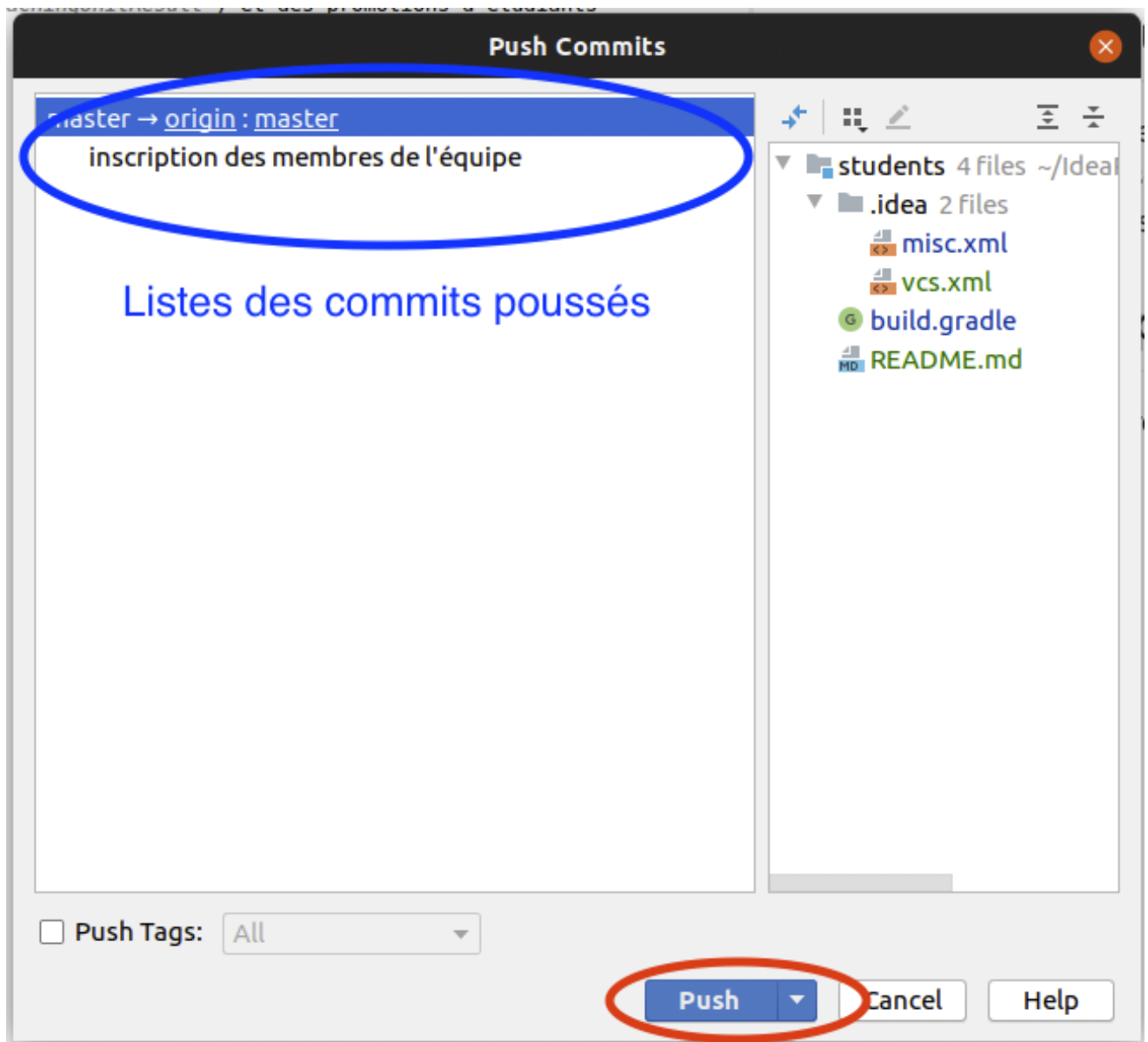
Préciser votre nom et email et cliquez sur le bouton Set and Commit.



3. Vous avez mis à jour votre dépôt local (sur votre machine) mais pas le dépôt distant (celui sur les serveurs de github). Pour cela, il faut faire un push. Allez dans le menu VCS -> git -> push.

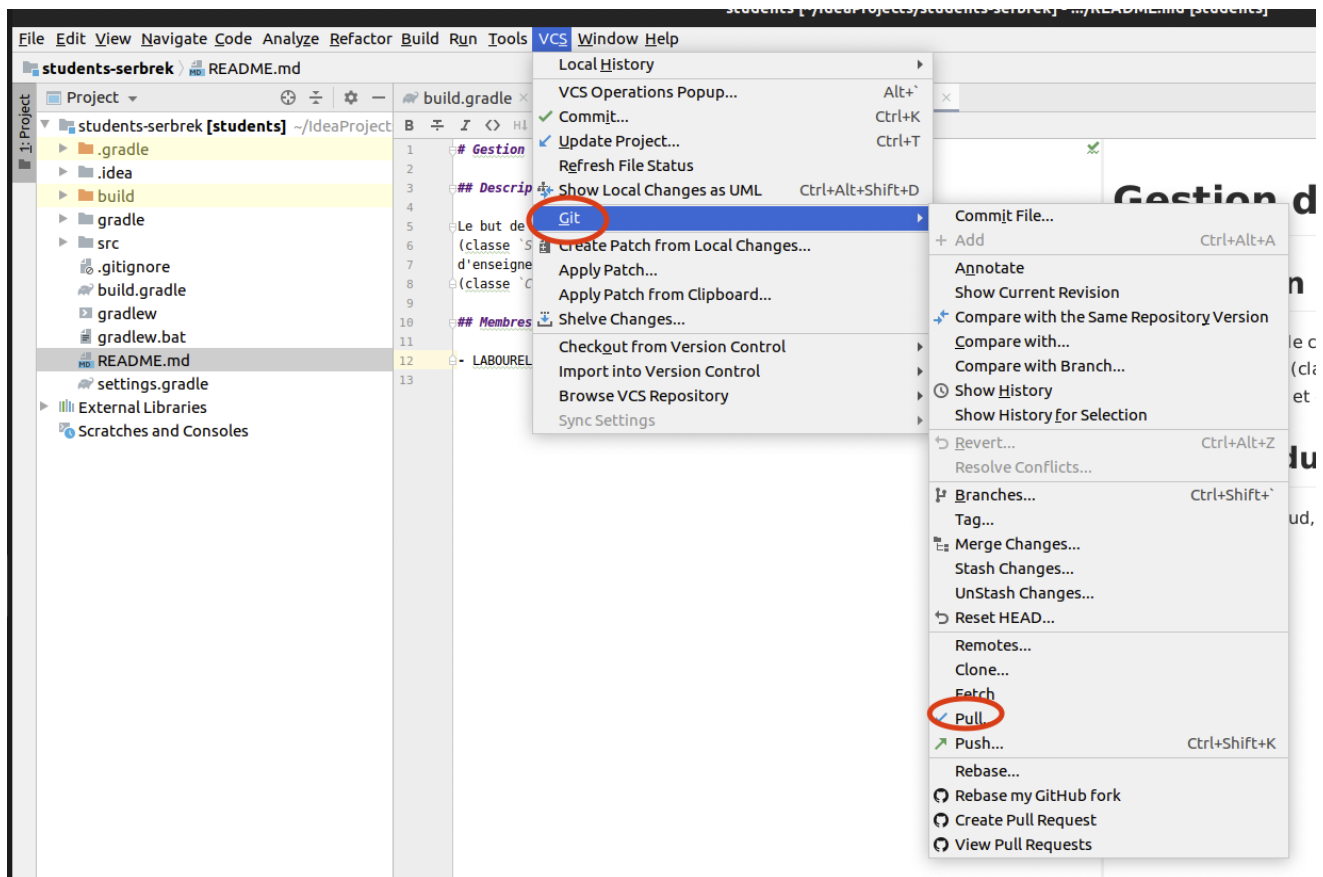


Cliquez sur le bouton push de la fenêtre qui vient d'apparaître. Vous pouvez voir la liste des commits que vous vous apprêtez à pousser.



Si tout se passe bien un popup **push successful** devrait apparaître en bas à droite de votre fenêtre.

4. Si jamais vous avez besoin de récupérer le projet sur le serveur (par exemple après un push de votre camarade), il vous suffit de faire un pull. Allez dans le menu VCS -> git -> pull.



Correction du programme

Méthodologie pour le TP

Vous allez maintenant pouvoir attaquer la correction du programme java du dépôt. Pour cela vous allez devoir respecter les consignes suivantes :

- À chaque modification de programme, faites un **commit** en sélectionnant les fichiers modifiés. Chaque **commit** doit contenir un message précisant la nature des modifications effectuées.
- À chaque tâche terminée, faites un **push** de votre travail.
- Ceci est le minimum. Vous pouvez faire plus de **commit** et plus de **push**, ainsi que des **pull** pour récupérer le travail de votre co-équipier.
- Si vous avez un problème et souhaitez l'aide de votre instructeur en dehors des séances, un **push** lui permet de voir votre programme.

Tâche 1 : classe Grade

documentation de la classe

Cette classe va permettre de représenter une note obtenue par un étudiant. Une note est une valeur flottante comprise entre 0 et 20.

Cette classe contient les éléments suivants qui sont corrects :

- `private static final int MAXIMUM_GRADE` : un attribut statique représentant la valeur de la note maximale qui est égal à 20.
- `private final double value` : la valeur de la note comprise entre 0 et `MAXIMUM_GRADE`.
- `public Grade(double value)` : constructeur évident.
- `public boolean equals(Object o)` : méthode permettant de tester l'égalité de deux notes.

Votre but est de compléter les instructions des méthodes suivantes :

- `public double getValue()` : retourne la valeur (`value`) de la note.
- `String toString()` : retourne une représentation de la note sous forme de chaîne de caractères. Pour une note ayant une valeur 12, cette méthode devra retourner la chaîne de caractères : "12.0/20".
- `public static Grade averageGrade(List<Grade> grades)` : calcule et renvoie la moyenne d'une liste de notes.

Assurez-vous que votre classe est correcte en exécutant à nouveau les tests et en vérifiant que votre classe passe les tests `testToString`, `testGetValue` et `testAverageGrade` de `TestGrade` avec succès.

Tâche 2 : classe `TeachingUnitResult`

documentation de la classe

Cette classe va nous permettre de représenter un résultat obtenu par un étudiant, c'est-à-dire une note associée à une Unité d'Enseignement (UE).

Cette classe contient les éléments suivants qui sont corrects :

- `private final String teachingUnitName` : le nom de l'unité d'enseignement du résultat.
- `private final Grade grade` : la note du résultat.
- `public TeachingUnitResult(String teachingUnitName, Grade grade)` : constructeur évident.
- `public boolean equals(Object o)` : méthode permettant de tester l'égalité de deux résultats.

Votre but est de compléter les instructions des méthodes suivantes :

- `public Grade getGrade()` : retourne la note associée au résultat.
- `public String toString()` : renvoie le nom de l'unité d'enseignement suivi de " : " suivi de la représentation en chaîne de caractère de la note. Par exemple, un résultat d'une UE de Programmation 2 avec une note de 20 devra renvoyer la chaîne de caractères suivante : "Programmation 2 : 20.0/20".

Assurez-vous que votre classe est correcte en exécutant à nouveau les tests et en vérifiant que votre classe passe les tests `testToString` et `testGetGrade` de `TestTeachingUnitResult` avec succès.

Tâche 3 : classe `Student`

documentation de la classe

Cette classe va nous permettre de représenter un étudiant.

Cette classe contient les éléments suivants qui sont corrects :

- `private final String firstName` : le prénom de l'étudiant.
- `private final String lastName` : le nom de famille de l'étudiant.
- `private final List<TeachingUnitResult> results` : les résultats de l'étudiant.
- `public Student(String firstName, String lastName)` : constructeur initialisant le nom et prénom de l'étudiant avec les valeurs données et créant une liste vide pour les résultats.
- `public boolean equals(Object o)` : méthode permettant de tester l'égalité de deux étudiants.

Votre but est de compléter les instructions des méthodes suivantes :

- `public void addResult(String teachingUnitName, Grade grade)` : ajoute un nouveau résultat à partir du nom de l'UE et d'une note.
- `public List<Grade> getGrades()` : renvoie la liste des notes associées aux résultats de l'étudiant.
- `public String toString()` : renvoie le nom de l'étudiant, c'est-à-dire son prénom, suivi d'un espace, suivi de son nom.
- `public Grade averageGrade()` : renvoie la moyenne des notes associés aux résultats de l'étudiant.
- `public void printResults()` : affiche les résultats de l'étudiant en sortie standard. Un étudiant nommé Arnaud Labourel et ayant 20 en Programmation 2 et en structures discrètes devra produire l'affichage suivant (avec un saut de ligne à la fin) :

```
Arnaud Labourel
Programmation 2 : 20.0/20
Structures discrètes : 20.0/20
Note moyenne : 20.0/20
```

Assurez-vous que votre classe est correcte en exécutant à nouveau les tests et en vérifiant que votre classe passe les tests `testToString`, `testGetGrades`, `testGetAverageGrade` et `testPrintResults` de `TestTeachingUnitResult` avec succès.

Tâche 4 : classe Cohort

documentation de la classe

Cette classe va nous permettre de représenter une promotion d'étudiants. La classe `Cohort` contiendra les attributs, méthodes et constructeurs suivants :

Cette classe contient les éléments suivants qui sont corrects :

- `private final String name` : le nom de la promotion
- `private final List<Student> students` : les étudiants de la promotion
- `public Cohort(String name)` : constructeur à partir du nom de la promotion et initialisant à vide la liste des étudiants

Votre but est de compléter les instructions des méthodes suivantes :

- `public void addStudent(Student student)` : ajoute un étudiant à la promotion.
- `public List<Student> getStudents()` : renvoie la liste des étudiants de la promotion.
- `String toString()` : retourne une représentation de la promotion correspondant à son nom.

- `public void printStudentsResults()` : affiche les résultats de l'étudiant en sortie standard. Une promotion ayant pour nom L2 informatique et deux étudiants devra produire l'affichage suivant (avec un saut de ligne à la fin)

```
L2 informatique

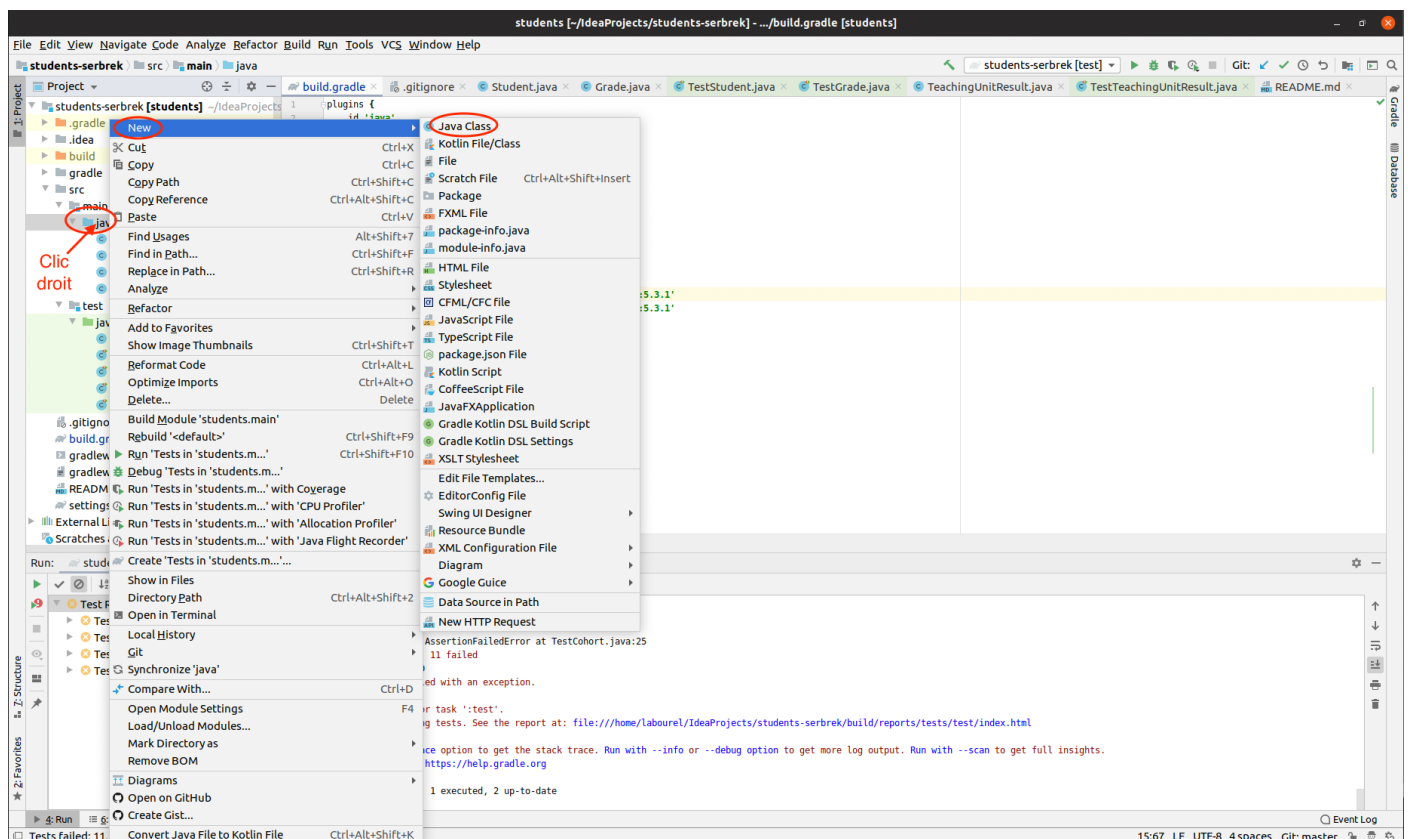
Paul Calcul
Programmation 2 : 10.0/20
Structures discrètes : 20.0/20
Note moyenne : 15.0/20

Pierre Kiroul
Programmation 2 : 10.0/20
Structures discrètes : 0.0/20
Note moyenne : 5.0/20
```

Créer la classe *Cohort* avec les méthodes demandées.

Tâche 5 : classe Main

Vous allez maintenant ajouter une classe *Main* au projet. Pour cela, il vous faut faire un clic droit sur le répertoire `src -> main -> java` et cliquer dans le menu `new -> Java Class`.



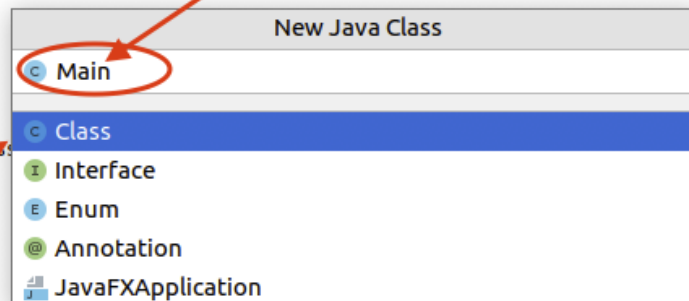
Tapez le nom de la classe, choisissez bien que vous souhaitez créer une classe et validez en appuyant sur entrée.

```
is {  
    implementation 'org.junit.jupiter:junit-jupiter-api:5.3.1'  
    runtimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.3.1'
```

```
.tPlatform()
```

```
pe: JavaExec) {  
    with sourceSets.main.runtimeClasspath {  
        "Main"
```

Choisir Class



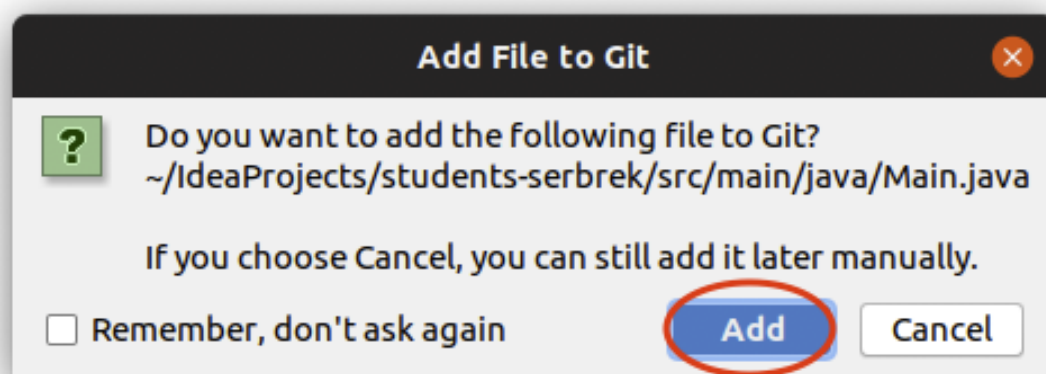
```
cies{}
```

d: 11, passed: 1 of 12 tests – 747 ms

.base/java.lang.Thread.run(Thread.java:834)

```
> testGetStudents() FAILED  
ntest4j.AssertionFailedError at TestCohort.java:25  
mpleted 11 failed
```

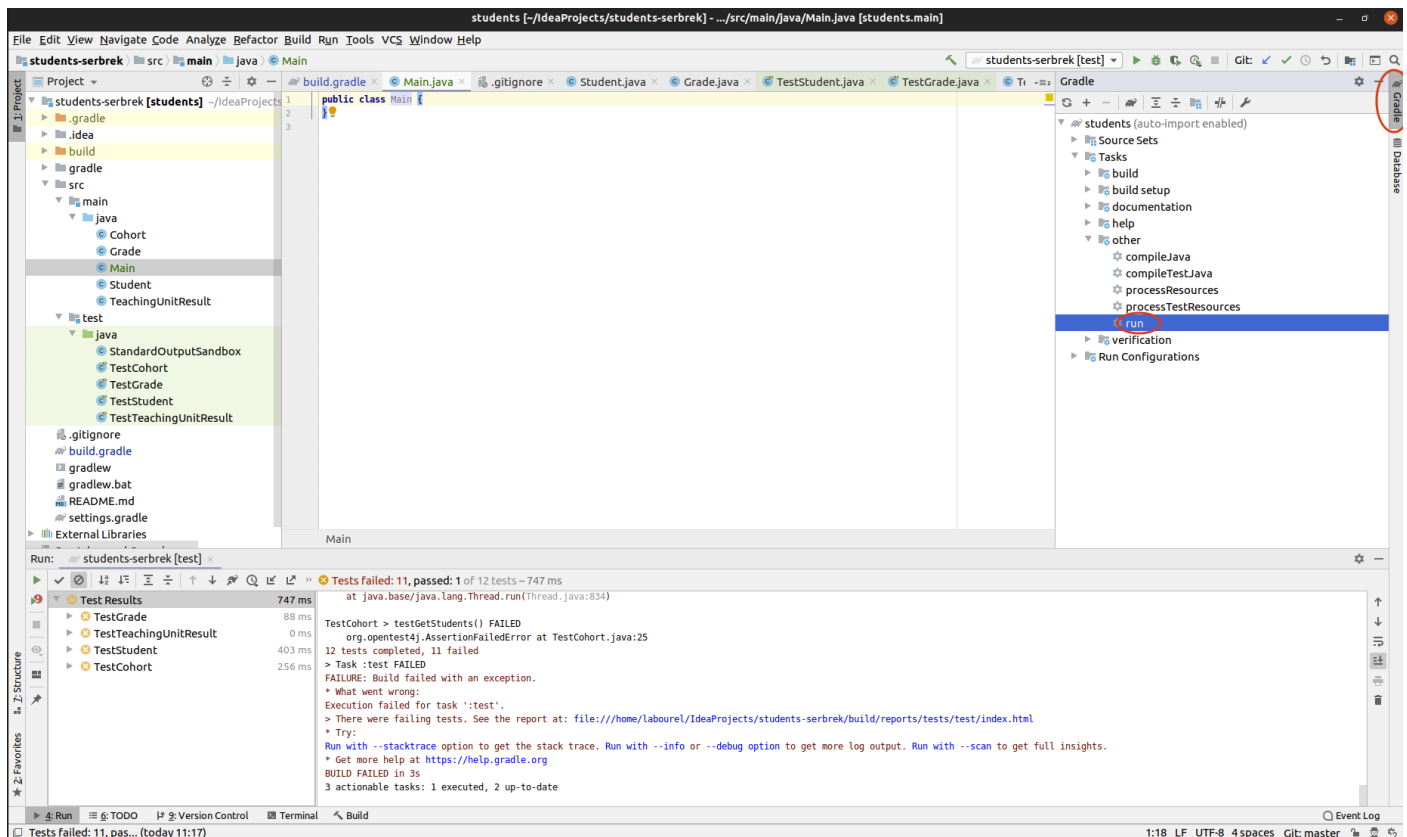
Normalement, on va vous demander si vous souhaitez ajouter ce nouveau fichier au dépôt git. Cliquez sur **add** pour l'ajouter afin qu'il soit mis à jour lors de votre prochain *commit*.



Ajoutez dans votre classe `Main` le code d'une méthode `public static void main(String[] args)` qui :

1. crée des instances de **Student** ayant les noms et prénoms des membres du projets,
2. ajoute à ces étudiants les notes en “Programmation 2” et “Structures discrètes” que vous aimeriez avoir,
3. crée une promotion (instance de **Cohort**) ayant nommée “L2 informatique”,
4. ajoute les étudiants créés à la promotion et
5. affiche les résultats de la promotion.

Pour compiler et exécuter le **main**, il faut passer par l’onglet gradle à droite et cliquer deux fois sur **students** -> **Tasks** -> **other** -> **run**. Cela va compiler et exécuter votre méthode **main**.



Tâches optionnelles

Si vous avez fini les tâches précédentes, vous pouvez améliorer votre projet en rajoutant les fonctionnalités suivantes :

- Ajout d’une méthode comptant le nombre d’étudiants ayant validé leur année (moyenne supérieure ou égale à 10) dans une promotion.
- Changement de la classe **Grade** pour qu’elle permette de stocker des notes correspondant à une absence du résultat (affiché **ABS**).
- Calcul du nombre d’absents d’une promotion.
- Calcul de la note maximum et minimum (hors absence) d’une promotion.
- Création d’une classe **TeachingUnit** qui permet d’associer des crédits à une UE.
- Calcul pondéré de la moyenne de résultats en fonction du nombre de crédits des UE.

— Calcul de la moyenne (hors absence) d'une promotion.