

|  |
| --- |
| Rapport projet JAVA |
| Hugues Bégeot  Camille Briand  Célia Bunouf  L3 Int – Groupe 4 |



Table des matières

[Introduction 3](#_Toc8759809)

[Les diagrammes conceptuels 3](#_Toc8759810)

[Justification des choix d’implémentation 6](#_Toc8759811)

[ Analyse Fonctionnelle Générale : 6](#_Toc8759812)

[ Analyse Fonctionnelle Détaillée : 7](#_Toc8759813)

[Explications de la gestion du format CSV 8](#_Toc8759814)

[Diagrammes Statistiques [JFreeChart] 8](#_Toc8759815)

[Conclusion 9](#_Toc8759816)

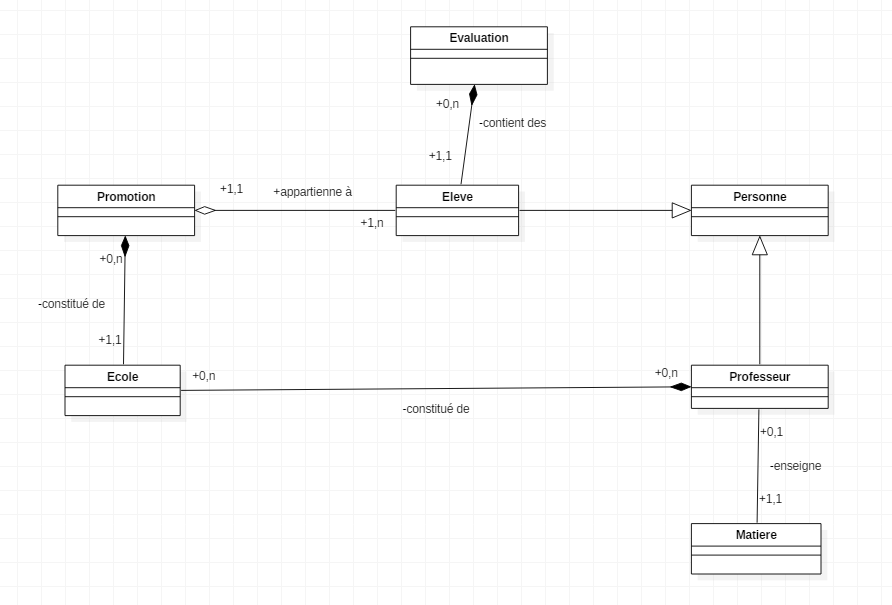
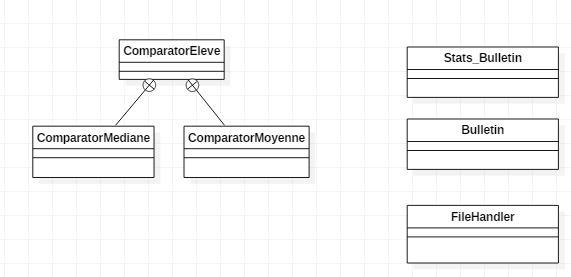
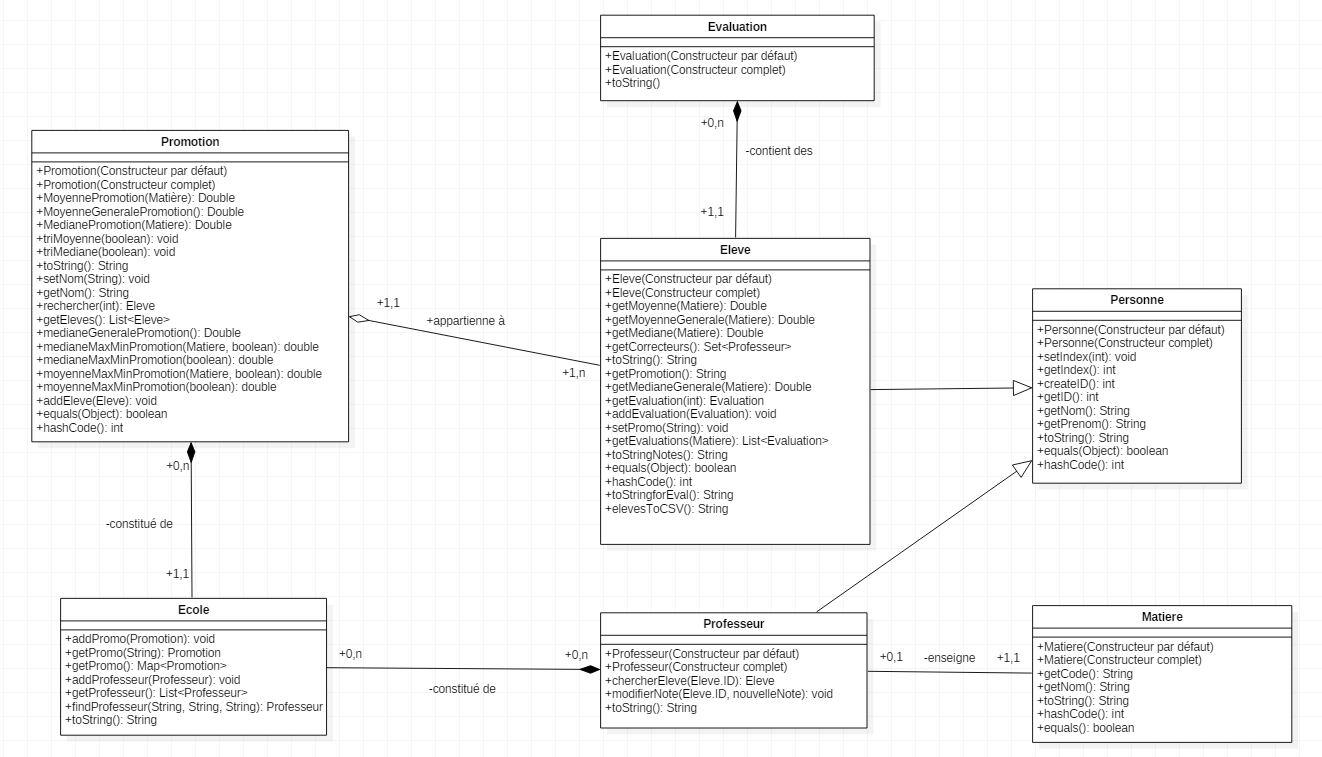
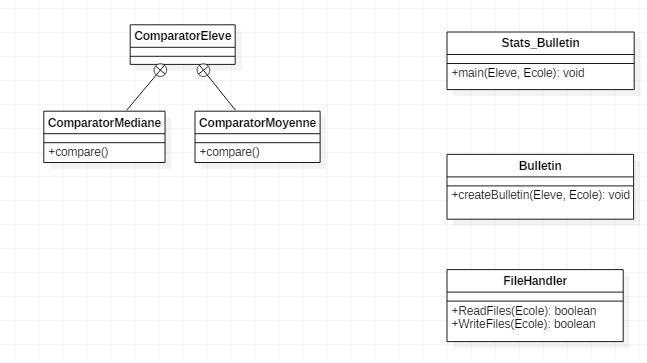
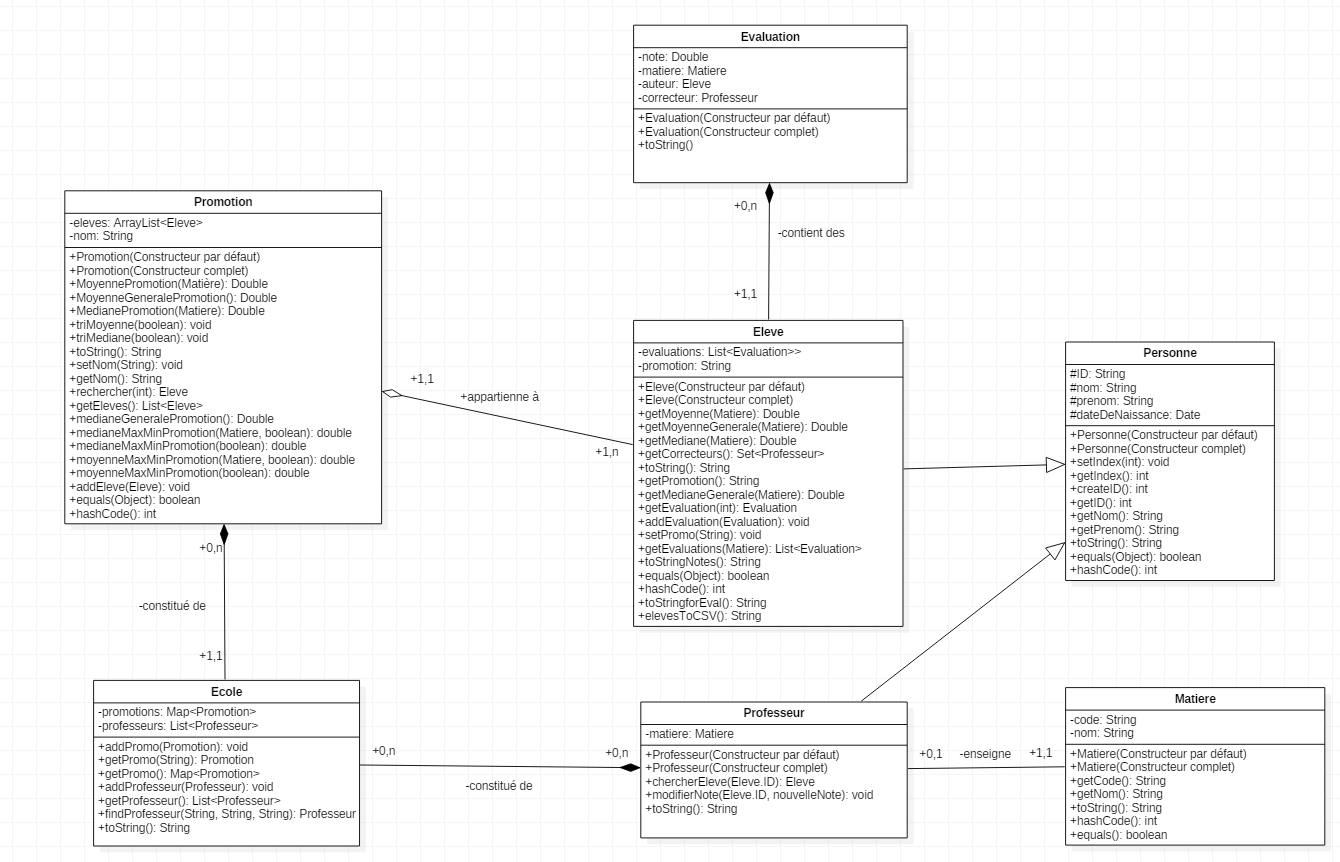
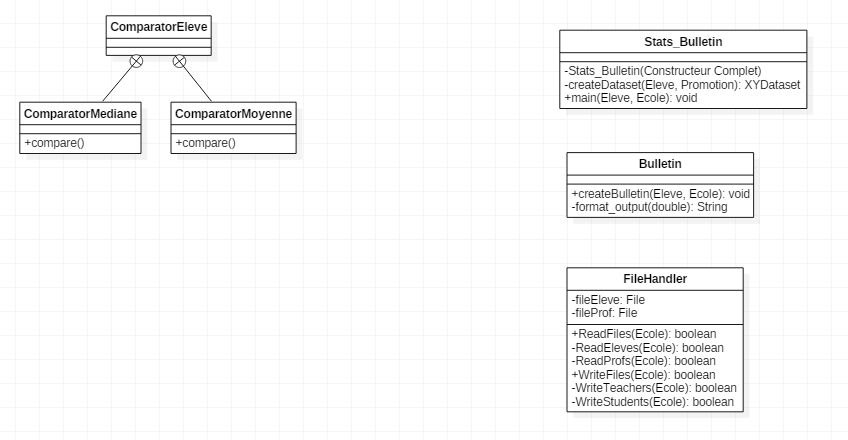
[Annexes 10](#_Toc8759817)

# Introduction

Dans le cours de JAVA, nous avions pour projet de gérer une école avec des professeurs et plusieurs promotions possédant des étudiants. L’objectif final du projet est de créer une interface graphique permettant aux professeurs d’évaluer les étudiants et aux étudiants de consulter leurs notes. Le bulletin d’un étudiant peut également être créer.

Le projet doit être réalisé en JAVA, avec une utilisation d’outils graphiques comme JFreeChart pour diverses statistiques ainsi que de fichiers CSV pour sauvegarder les données de l’école. L’interface à été réaliser à l’aide de JavaSwing.

# Les diagrammes conceptuels

* 1er diagramme : relation et cardinalité
* Classes fondamentales
* Classes fonctionnelles
* 2ème diagramme :
* Classe fondamentale
* Classe fonctionnelle
* 3ème diagramme :
* Classe fondamentale :
* Classe fonctionnelle :

# Justification des choix d’implémentation

# Analyse Fonctionnelle Générale :

Le projet doit pouvoir gérer des étudiants, des notes obtenues aux évaluations et des professeurs. C’est pour quoi nous avons basé le projet sur sept classes principales : une évaluation, une matière, une promotion, un élève et un professeur, tous deux héritant de la classe personne. Nous avons également fait le choix de créer une classe école. Cette classe école a pour but de regrouper tout ce qui constitue une école c.-à-d. un ensemble de professeur et un ensemble de promotion. Chaque promotion est constituée d’un ensemble d’élève.

A partir de l’école, on peut chercher un professeur, obtenir une liste des promotions, une profession particulière et également la totalité des matières enseignées dans cette école.

Une promotion implémente différente fonction pour calculer les moyennes et médianes totales de la promotion, ce calcul permet des tris des élèves par moyenne ou médiane croissante ou décroissante grâce à la classe de ComparatorEleve. Nous pouvons également récupérer une liste des étudiants appartenant à cette promotion, ainsi que chercher un étudiant.

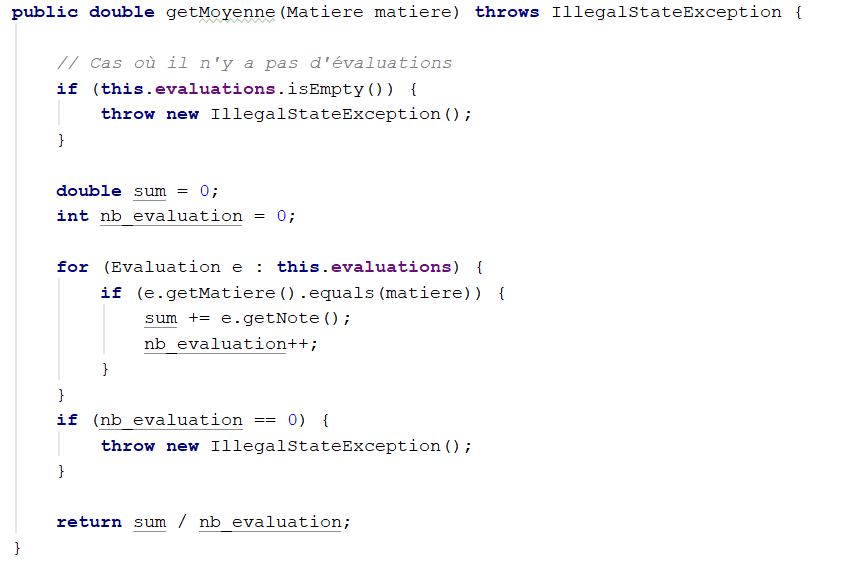
Nous avons fait en sorte qu’un professeur n’enseigne qu’une seule matière ainsi quand ce professeur ajoute une note, l’évaluation est ajoutée dans sa matière. Un professeur peut également chercher un étudiant, donner une note à un étudiant dans sa matière mais également modifier une note existante.

Un étudiant peut voir ses notes, son bulletin, accéder à la liste de ses correcteurs et également consulter les listes des étudiants des promotions. Un étudiant nécessite donc de pouvoir calculer ses moyennes, ses médianes dans les différentes matières ainsi que générale. On peut également trouver une liste des correcteurs de cet étudiant ainsi que l’ensemble de ses évaluations. On peut de ce fait rajouter des notes à cet étudiant. Différentes méthodes d’affichage d’un étudiant ont également été implémentées pour répondre au cahier des charges.

Notre programme donne lieu à un login afin de déterminer quelle interface lancer en fonction de l’utilisateur. Le login est constitué du matricule de la personne ainsi qu’un mot de passe. 3 possibilités : l’utilisateur est un administrateur [ID : 0 | MDP : admin], l’utilisateur est un professeur, l’utilisateur est un étudiant. L’identité de la personne est trouvée en regardant les différentes listes.

Nous avons une classe fonctionnelle FileHandler qui permet de gérer entièrement la base de données CSV. Cette classe contient toutes les fonctions d’écritures et de lectures des fichiers.

# Analyse Fonctionnelle Détaillée :

L’objectif principale du projet est donc la gestion des notes des étudiants. Du coup, il faut pouvoir calculer les moyennes des étudiants pour permettre ceci une fonction très importante est le calcul de moyenne d’un étudiant et notamment le calcul de moyenne par matière. Pour cela nous vérifions si l’étudiant possède des évaluations, s’il n’en a pas on lance l’IllegalStateException. Si oui, nous regardons si l’évaluation correspond à la matière que nous chercherons et si oui on l’ajoute à la somme totale et nous gardons en mémoire le nombre d’évaluations concerné par la matière. Le résultat final correspond à la somme totale diviser par le nombre d’évaluations. Sur cette base les moyennes générales et les moyennes minimum, maximum des promotions peuvent être calculé.

Le même principe est utilisé pour le calcul de médiane en vérifiant le nombre d’évaluations, mettant à part le cas d’une seule évaluation, puis en triant les évaluations de la matière par ordre croissant et finalement retourner la note médiane ou la moyenne des deux notes médianes.

Pour qu’un professeur puisse modifier ou ajouter une note, on doit d’abord chercher un élève, si l’élève n’est pas trouvé une IllegalStateException est donné<e. Puis nous cherchons une évaluation grâce à son index, si l’évaluation est trouvée nous modifions la note, sinon nous ajoutons la note dans la matière enseigné professeur.

# Explications de la gestion du format CSV

Les deux principales classes de notre implémentation sont les élèves ainsi que les professeurs. On a ainsi choisi de représenter les données sous deux fichiers CSV : un contenant les étudiants ainsi que leurs évaluations et un autre contenant les professeurs.

Le fichier CSV des professeurs contient ainsi le nom, prénom, date de naissance, ID, mot de passe ainsi que la matière enseignée par le professeur. Lors de la lecture du fichier les éléments Professeur sont créés et ajoutés à l’école.

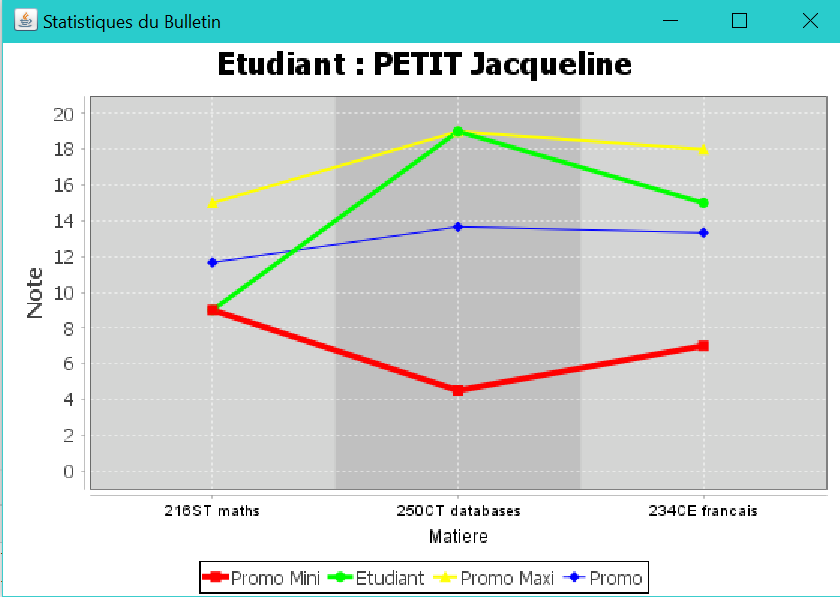
Le fichier CSV des élèves contient le nom, prénom, date de naissance, ID, mot de passe, promotion, ainsi qu’autant de colonne nécessaire en fonction du nombre d’évaluations présentent pour l’étudiant. Dans cette base de données, une évaluation est représentée par la note et l’ID du professeur correcteur. Il a été choisi de l’ID du professeur correcteur car grâce à celui-ci nous pouvons retrouver la matière de l’évaluation étant donné que le professeur n’enseigne qu’une seule matière.

Les mots de passe sont stockés pour la fonction de login, et le login pour un administrateur est unique [ID : 0 | MDP : admin]

L’avantage des fichiers CSV ici par rapport aux bases de données telles que MySQL, est que le nombre de colonne est variable en fonction des élèves. Ce n’est pas une structure fixe.

# Diagrammes Statistiques [JFreeChart]

Sur les statistiques présentes sur le graph, nous avons décidé de représenter la moyenne de l’étudiant à côté de la moyenne générale de la promotion et des moyennes minimales et maximales de la promotion. Pour ceci nous avons utilisé toutes les fonctions déjà créées pour récupérer les noms des matières ainsi que les différentes moyennes nécessaires.

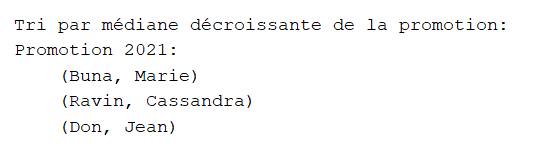
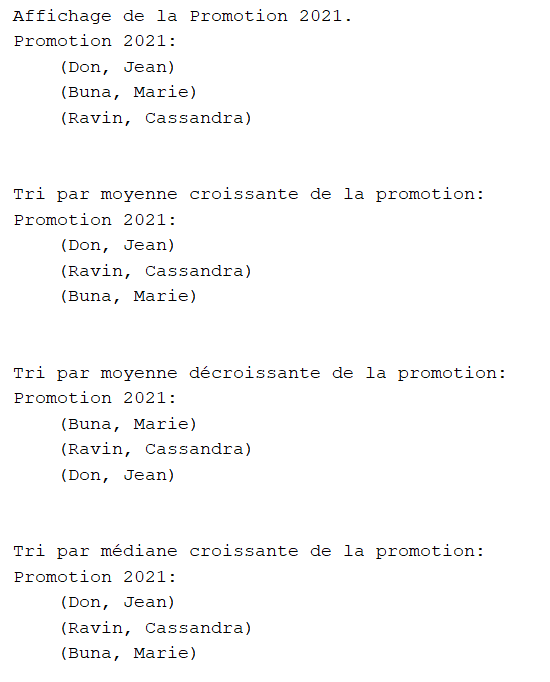
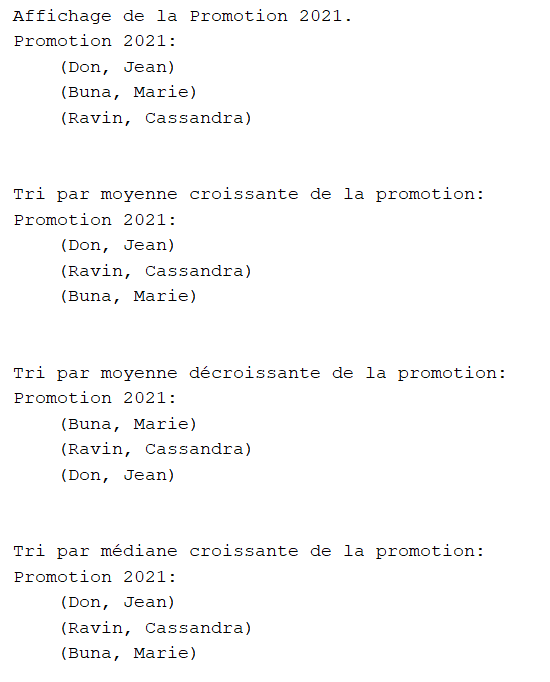
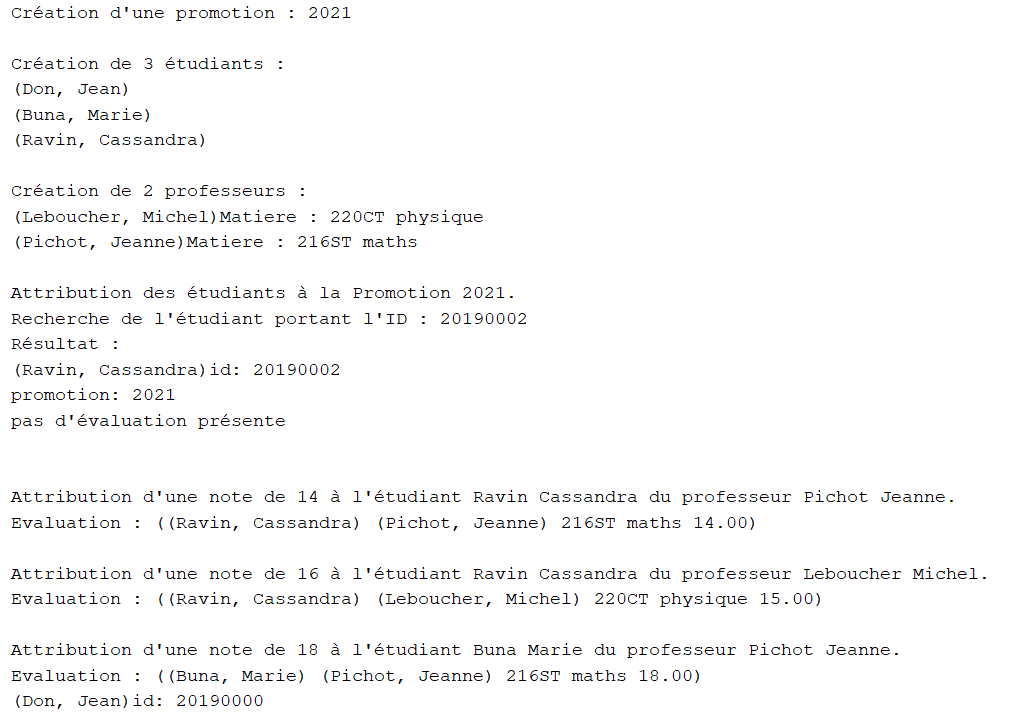
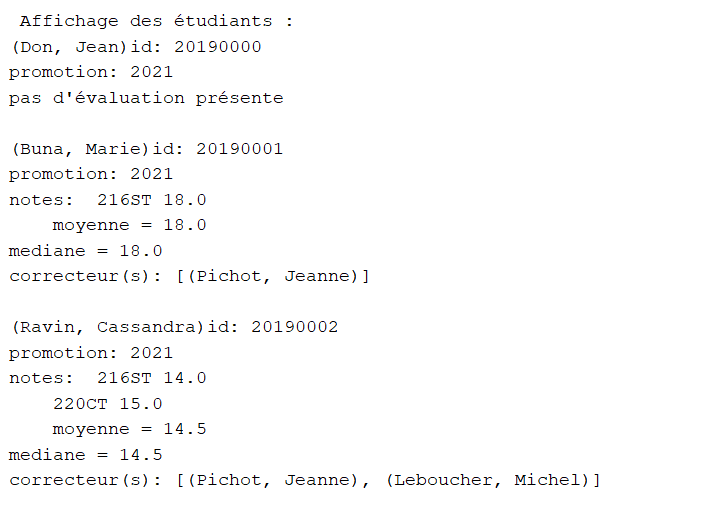


# Conclusion

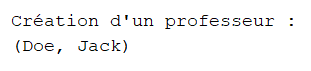
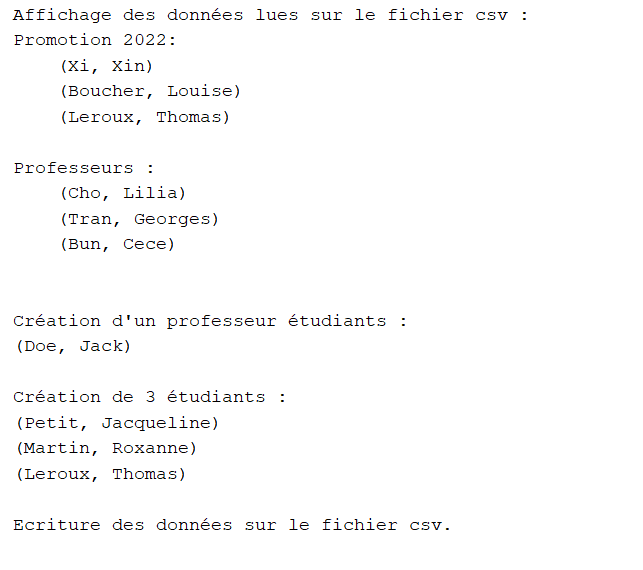
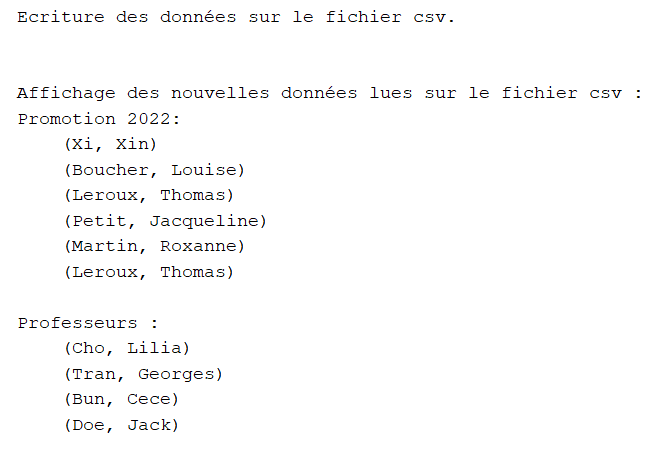
En conclusion, nous avons réussi à créer un programme capable de gérer le fonctionnement des notes d’une école ainsi que leur étudiants et professeurs. Ce projet nous a permis d’utiliser un nouvel outil, JFreeChart, pour la création de différents graphs et également d’approfondir nos connaissances dans l’utilisation du JavaSwing.

# Annexes

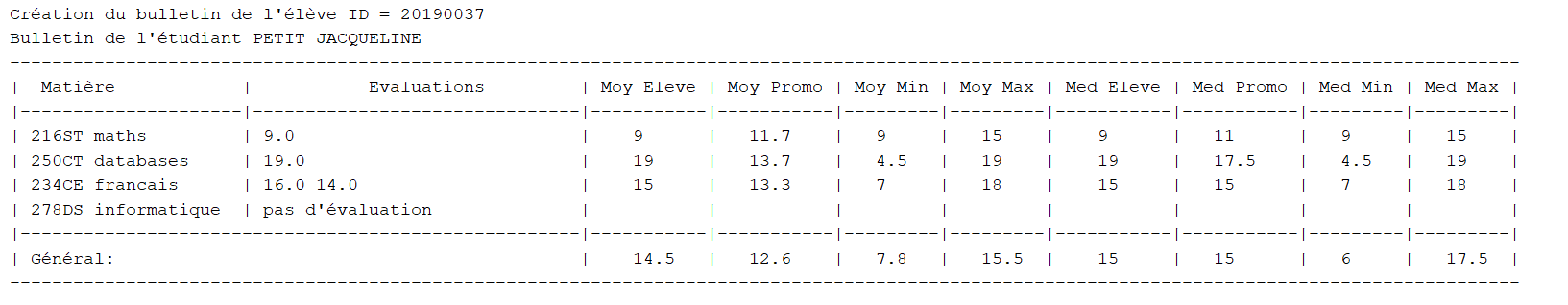
Capture d’écran Version 1 :

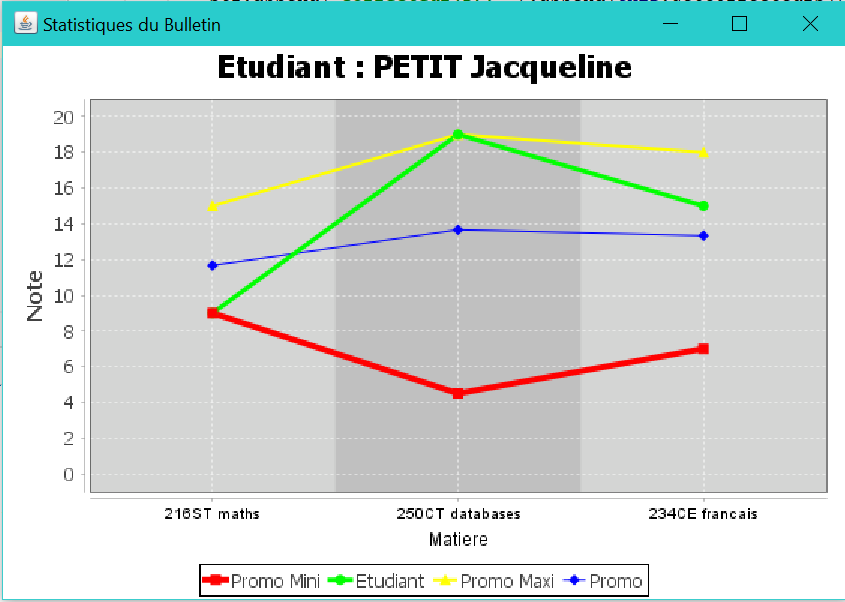


Capture d’écran Version 2 :

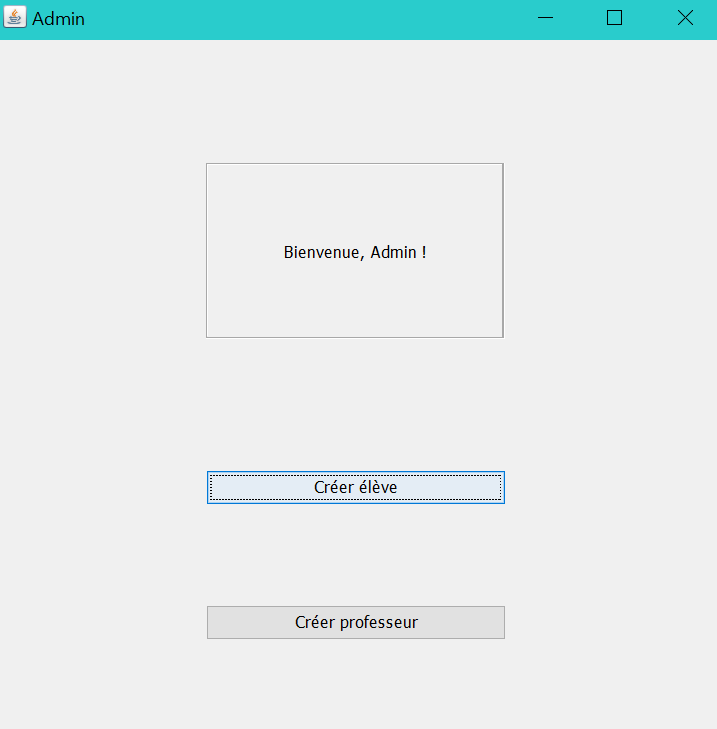


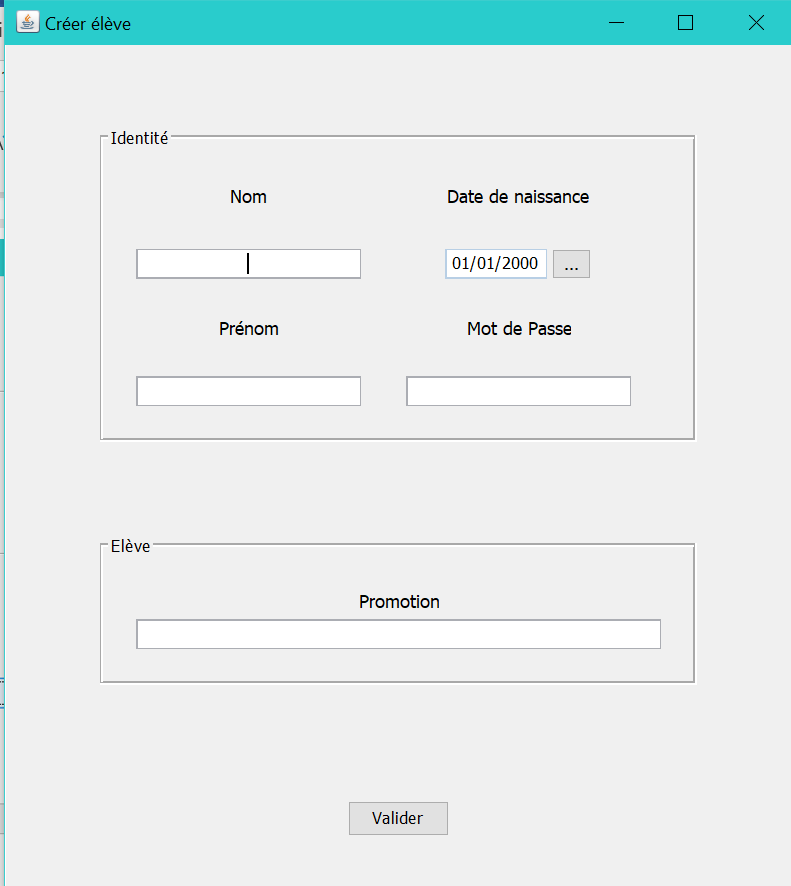
Capture d’écran Version 3 :



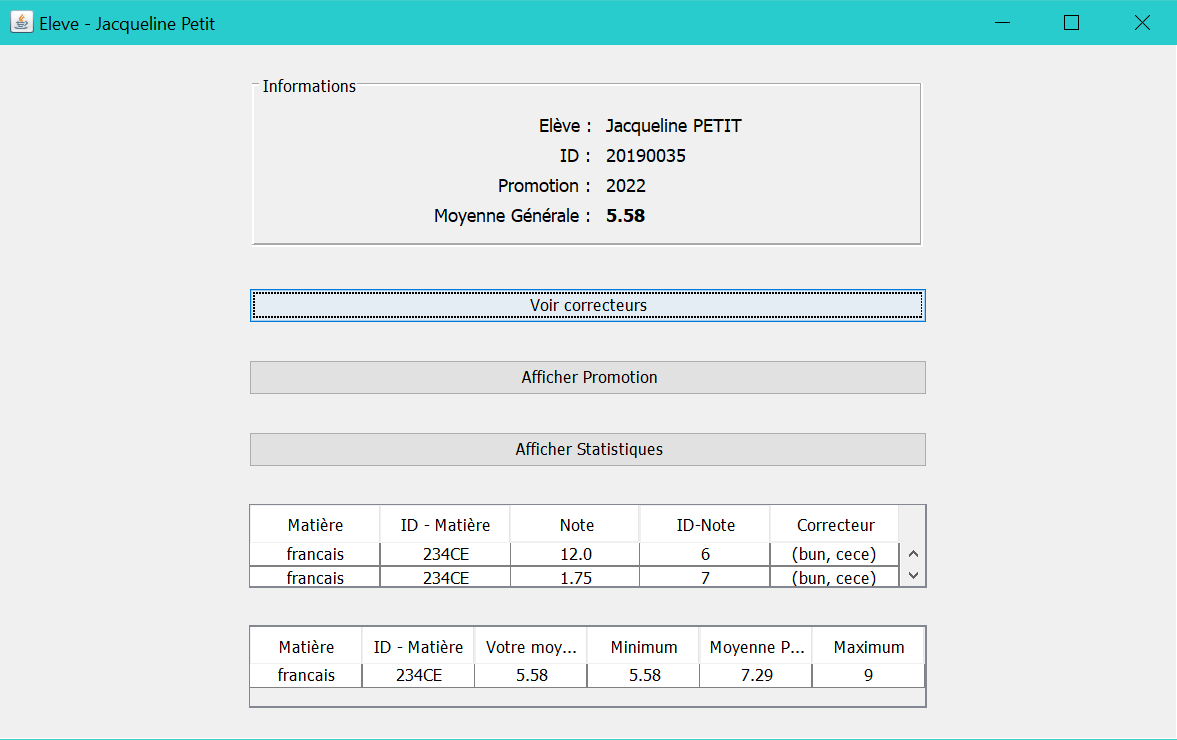
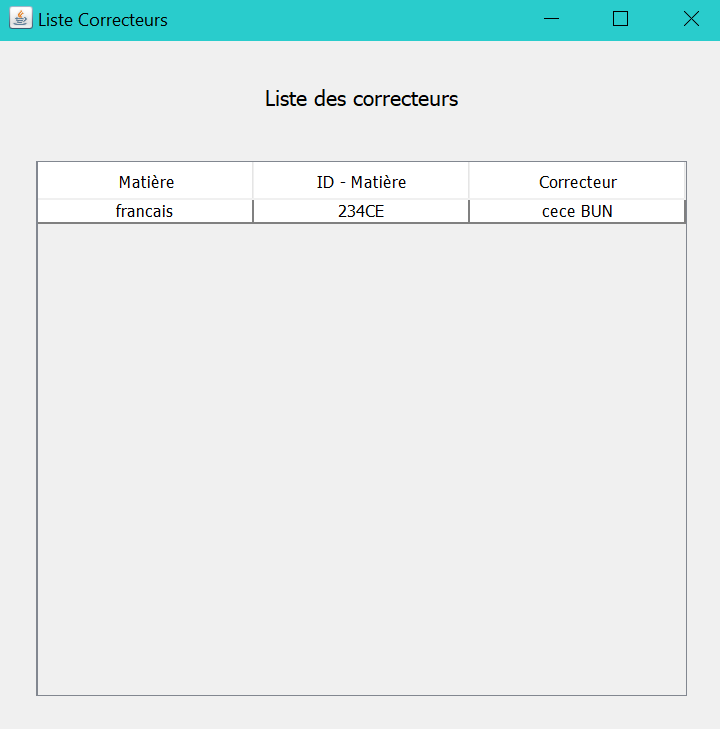


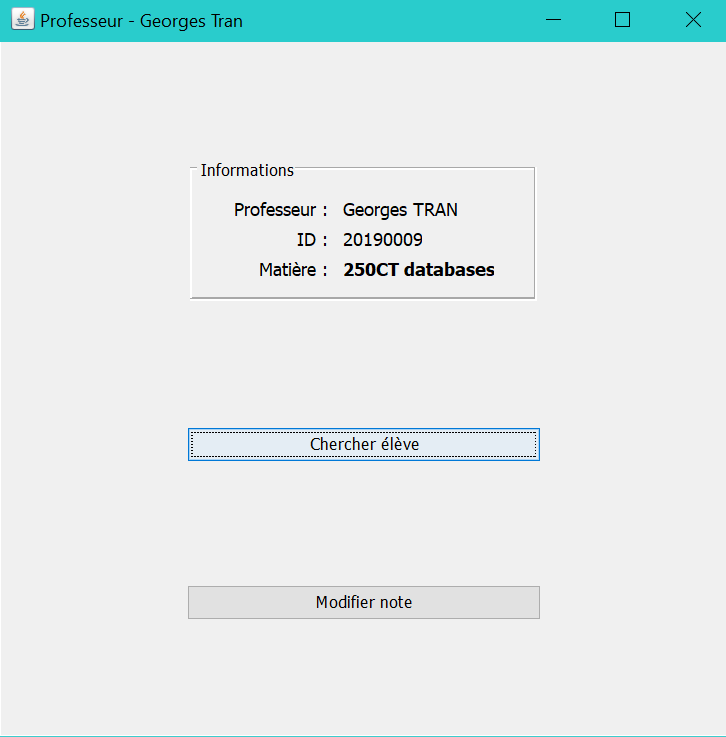
Capture d’écran Version 4 :

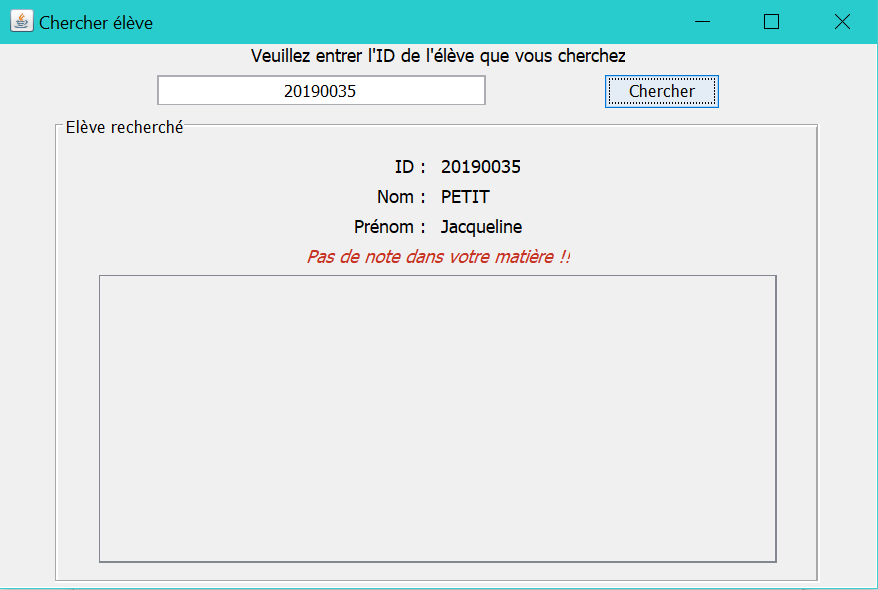
→ l’utilisateur est administrateur

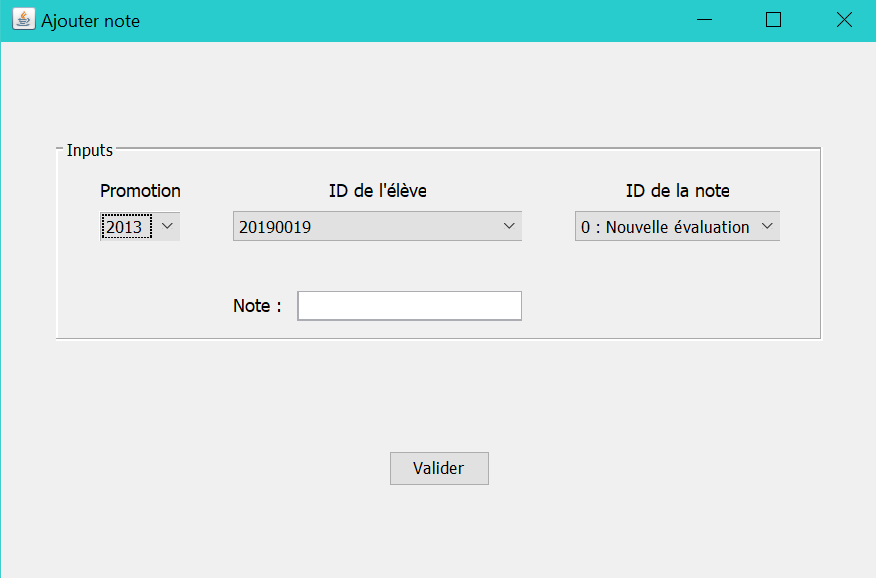


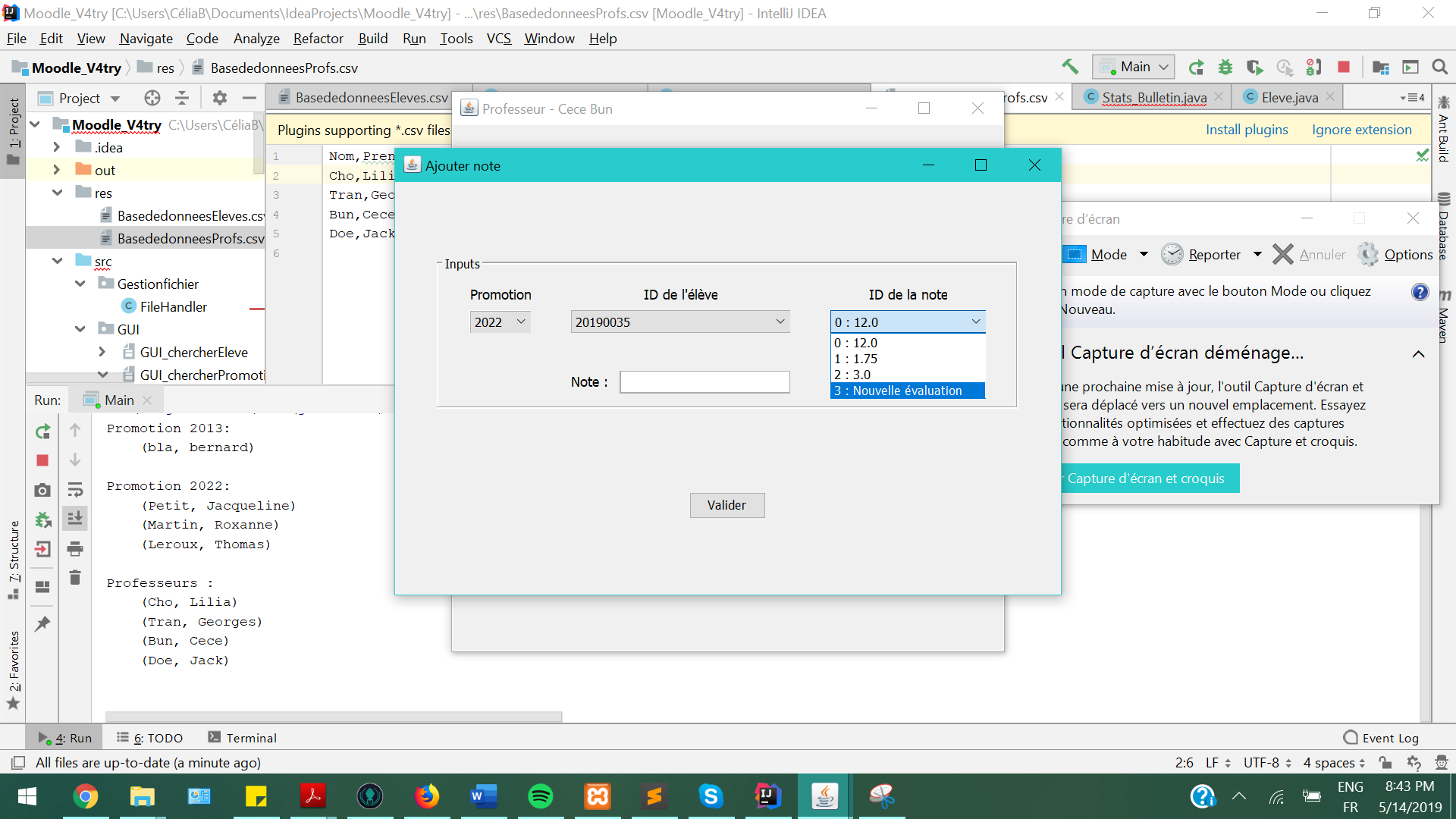
Une image contenant capture d’écran

Description générée automatiquement→ l’utilisateur est un élève

→ l’utilisateur est un professeur



Le professeur n’a pas encore entrée de notes. Ainsi il ne peut pas modifier mais seulement ajouter de nouvelles notes.



Néanmoins en prenant un autre professeur nous pouvons voir sur la capture ci-contre que le professeur peut également modifier les notes déjà existantes.