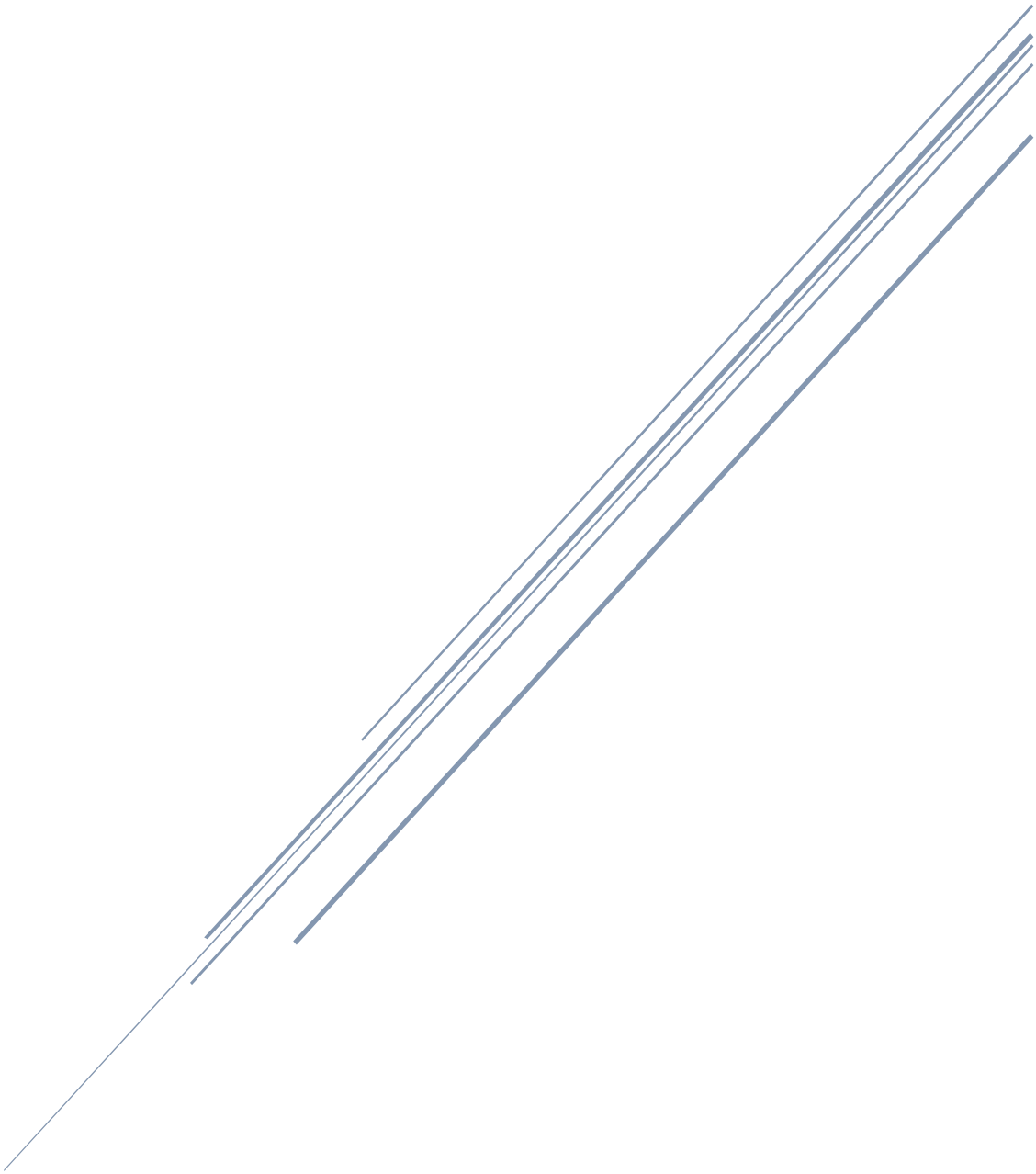
RAPPORT PROJET JAVA

Evaluations Elèves/Professeur



HUAULT JEAN – INGHELBRECHT PAUL-ANTOINE

Sommaire

Introduction :

Dans le cadre de notre cours de programmation, nous avons dû réaliser un projet en Java. Le but du projet est de créer une interface, en console, où les professeurs puissent donner des notes aux élèves et où les élèves puissent voir leurs notes.

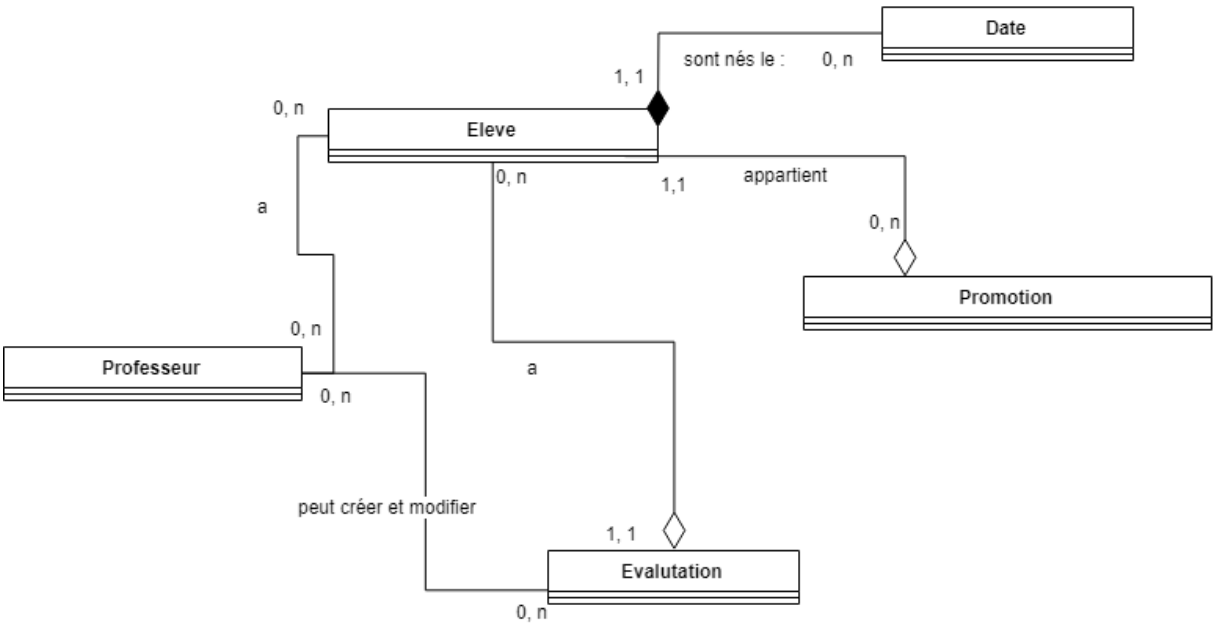
L'objectif est donc de mettre en place un code source java capable de récupérer les notes données par les professeurs pour chaque élève, les stocker et ensuite les afficher lorsque l'étudiant veut les voir. A travers l'élaboration de différentes classes, de méthodes ou encore l'utilisation librairies java, nous avons réalisé ce projet.

La problématique principale est : Comment bien respecter le cahier des charges en fonction de nos compétences techniques ? Toutefois, d'autres problèmes apparaissent au fil de l'avancée du projet. Des problèmes mineurs dû à un manque de connaissance du langage de programmation.

Afin de réaliser ce projet, nous avons beaucoup appris sur le langage Java. Bien connaitre la structure, les classes, etc, permet de gagner beaucoup de temps pour mener à bien ce projet. L'IDE (integrated development environment) est aussi très importante. En ce qui nous concerne, nous avons utilisé Intellij, qui a un très bon débuggeur ainsi qu'une interface professionnelle.

Diagrammes de Classes :

Avant de se lancer dans la réalisation du projet, il est essentiel de bien comprendre le cahier des charges pour en faire ressortir les informations essentielles. Nous avons donc esquissé un premier diagramme de classe à travers les information du cahier des charges.

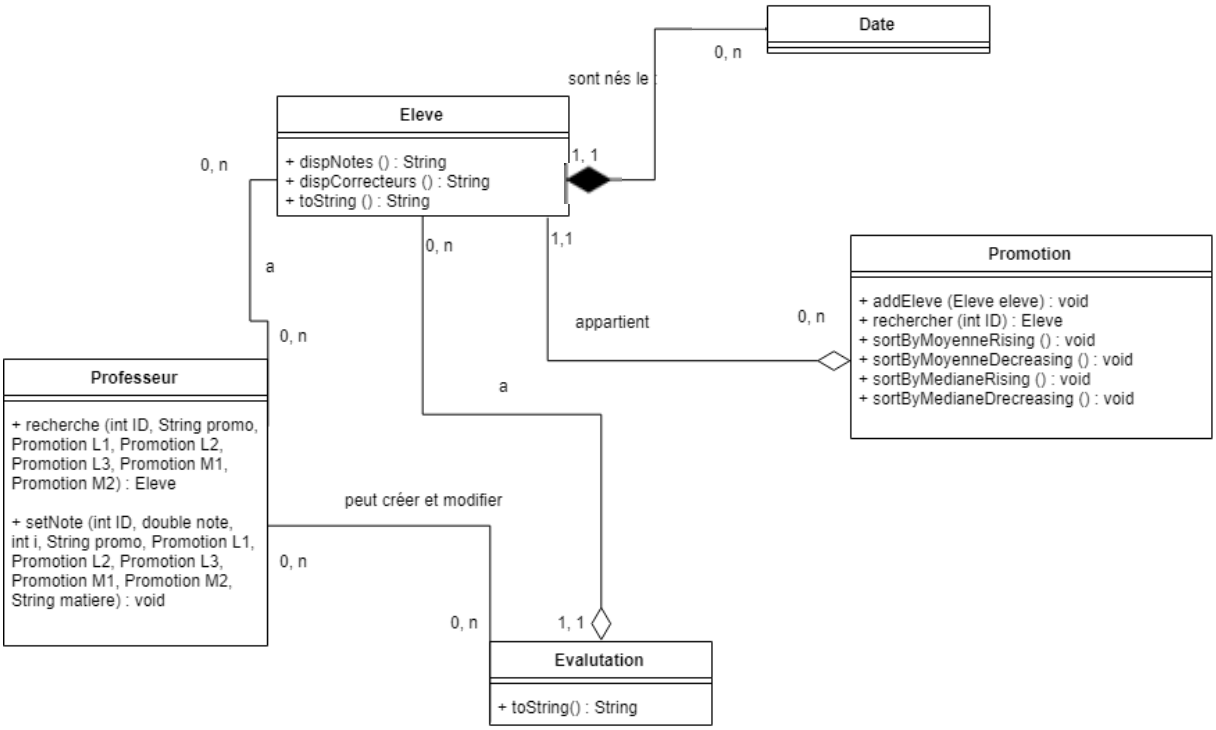


Ce premier diagramme de classes fait apparaitre, les différents classes qui vont être utilisées durant le projet. La principale classe est la classe Eleve qui possède une relation avec toutes les autres classes. Toutefois, les relations sont différentes entre chaque classe.

- Entre la classe Eleve et la classe Professeur, nous avons une relation simple et une cardinalité identique dans les deux sens, 0 à n possibilités.
- Entre la classe Eleve et Promotion, nous avons une autre relation, ici c'est une agrégation. En d'autres termes, un élève est attaché à une promotion et une promotion peut être rattachée à un élève en particulier. Les cardinalités sont différentes, un élève fait partie d'une et d'une seule promotion, alors qu'une promotion peut avoir de 0 à n élèves.
- Enfin la dernière relation possible est une composition entre Eleve et Date. Sans au moins un élève, la classe Date n'excite pas, elle dépend donc de la classe Eleve. Niveau cardinalité, un élève n'a qu'une seule date de naissance, mais au cours de la même date, il peut y avoir plusieurs naissances donc, 0 à n.

De plus chaque relation possède un nom, ce qui nous permet de nous bien comprendre les liens entre les classes.

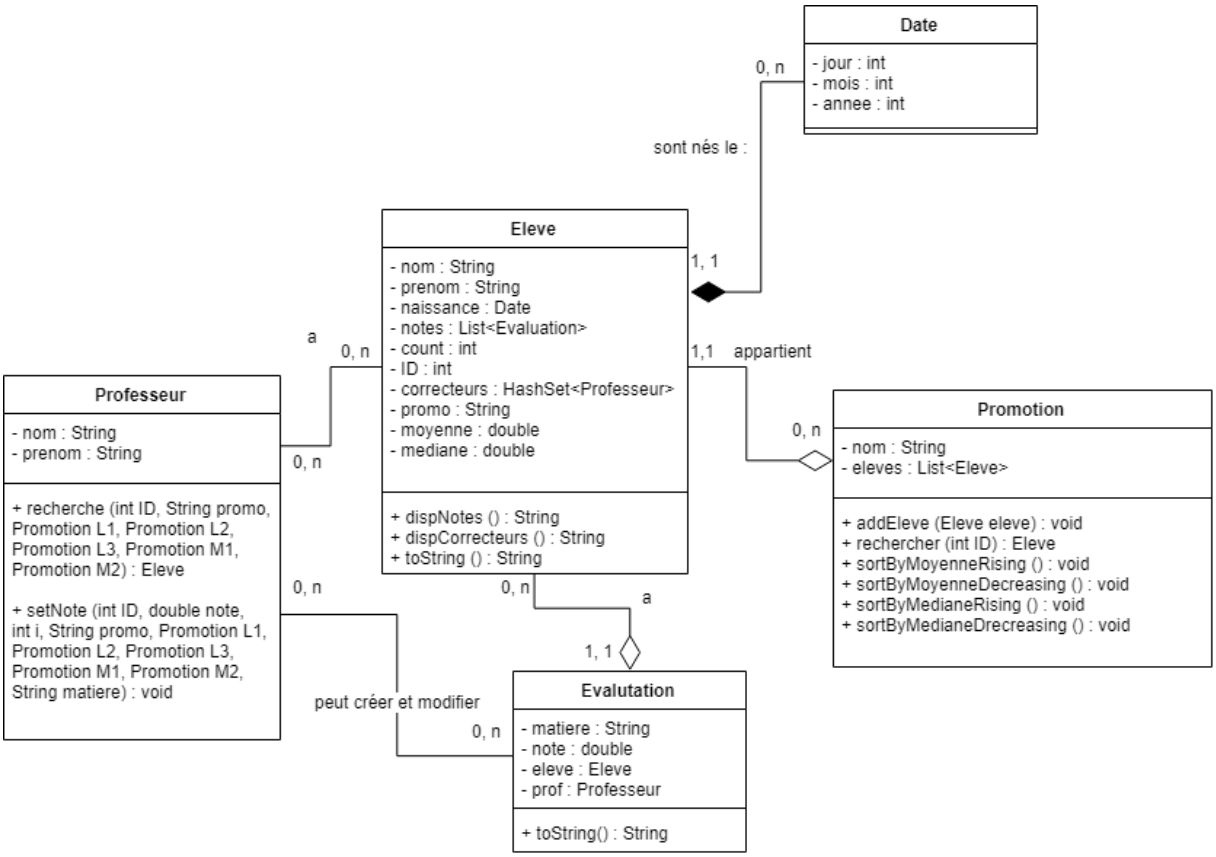
A partir du premier diagramme, nous l'avons complété pour obtenir celui-ci :



Ce second diagramme reprend le précédent et on y ajouté les méthodes publiques de chaque classe. Pour chaque méthode d'une classe, vous retrouvez son nom, ses paramètres et enfin si la méthode retourne quelque chose.

Dans ce diagramme, nous avons choisi de ne pas faire apparaitre les constructeurs de chaque classe, ainsi que les getters et les setters de toutes les variables. Ne pas les faire apparaitre, nous permet d'avoir un diagramme de classes simple et peu chargé.

Enfin, nous avons réalisé le diagramme de classes complet en y rajoutant les attributs de chaque classe :



Voici, le diagramme de classes complet. Il y a donc 5 classes dans le projet. Dans chacune des classes, on y retrouve son nom, ses attributs qui sont public (quand il y a un + devant) ou privé (avec un -) et enfin les méthodes de la classe.

Analyse fonctionnelle générale

Elaboration d'une classe en Java :

Pour réaliser ce projet, nous sommes dans l'obligation de créer des classes. Une classe est un ensemble d'attribut et de méthodes liés entre elles.

Prenons l'exemple de la classe Eleve, de notre projet :

Vous avez tout d'abord, les différents attributs qui constituent la classe, ici vous avez, le nom (string), le prénom (string), etc.

Ensuite, vous avez le constructeur de la classe, il permet de créer un nouvel objet pour sa propre classe.

Enfin, les getters et les setters des attributs. Ils permettent de donner une valeur à la variable ou retourner sa valeur.

Modules principaux :

Pour mener à bien ce projet, nous avons eu besoin de mettre en place cinq classes principales : la classe Eleve, pour gérer toutes les données relatives aux informations personnelles d'un élève ; la classe Professeur pour gérer toutes les données et actions relatives aux professeurs ; la classe Promotion pour rassembler les élèves et assurer les fonctions de triage de ces derniers ; la classe Evaluations pour centraliser les données de chaque évaluation ; et enfin la classe date, permettant de créer une seule entité sur la base de toutes les données composant la date de naissance des élèves.

Interface Comparators :

Une interface contient des méthodes permettant, dans notre cas, de trier des données. Elle permet aussi d'être utilisée sur différentes classes. Les méthodes de tri peuvent être utilisées par exemple pour trier des élèves en fonction de leur moyenne mais aussi les évaluations en fonction des notes.

Nous avons donc créé des méthodes pour trier les notes par ordre croissant, et aussi les moyennes, etc. Par la suite, nous utilisons la méthode Collections.sort() qui permet de classer les éléments d'une liste par ordre croissant. Dans le cas où nous voudrions effectuer un tri dans l'ordre décroissant, nous utilisons ensuite la méthode Collection.reverse().

Manipulation de fichiers :

Dans la deuxième partie du projet, nous avons mis en place une nouvelle classe, la classe FileManipulation, qui permet de gérer les relations entre les fichiers .csv (contenant toutes les données nécessaires à la création des objets utilisés) et le programme en lui-même.

Analyse fonctionnelle détaillée

Classe Eleves :

La classe Eleve est relativement simple, à commencer par ses attributs : elle comprend le nom et prénom de l'élève, sa date de naissance, un identifiant unique, le nom de la promotion de cet élève, la valeur de la moyenne de ses notes, ainsi que leur médiane. Elle contient aussi une liste des évaluations de cet élève sous la forme d'une ArrayList, ainsi qu'une liste de ses correcteurs sous la forme d'un HashSet, ce qui permet d'éviter les doublons.
En terme de méthodes, on retrouve les getters et setters habituels, un constructeur, une méthode permettant d'initialiser une liste de notes initiale lors de la création de l'élève, une méthode pour calculer la moyenne et une pour la médiane, et deux méthodes pour afficher les notes et les correcteurs de cet élève. Enfin, on retrouve une méthode toString() qui affiche toutes les informations de l'élève dans la console.

Classe Professeurs :

La classe Professeur est encore plus simple : pour ce qui est des attributs, elle ne comprend que le nom et le prénom du professeur.
Elle possède trois méthodes principales, au-delà des getters et setters : la méthode recherche(), qui permet de rechercher un élève dans une promotion grâce à son identifiant, la méthode setNote() qui permet d'attribuer une note à un élève spécifié, appartenant à une promotion spécifiée, dans une matière spécifiée, et à un index spécifié dans sa liste d'évaluations. Enfin, une méthode toString() retourne le nom et prénom du professeur de manière formatée.

Classe Promotion :

La classe Promotion n'a que deux attributs : son nom, et une ArrayList des élèves qu'elle contient.
Ses méthodes sont plus intéressantes : une pour ajouter un élève, une pour effectuer une recherche dans la promo (notamment appelée par la classe Professeur), mais surtout quatre méthodes de tri. Nous avons donc la possibilité de trier par médiane ou par moyenne, par ordre croissant ou décroissant. On implémente pour cela les méthodes de tri mises au point dans l'interface Comparators.

Classe Evaluations :

La classe Evaluation est principalement définie par ses attributs : le nom de la matière, la valeur de la note, l'élève concerné et le professeur correcteur.
Elle ne présente aucune méthode intéressante en dehors de ses getters et setters, appelés par les autres classes pour appliquer des opérations à ces notes.

Classe Date :

La classe date n'a pas d'utilité réelle. Elle nous est imposée par le sujet mais n'est jamais réellement utilisée. Elle ne comprend que son constructeur et ses setters.

Interface Comparators :

Cette interface comprend une méthode principale : la méthode comparaison(). Elle compare la valeur de deux éléments, et retourne une valeur numérique entre -1 et 1 en fonction du résultat. Combiné avec la méthode compare(), qui varie selon le type des paramètres qu'on lui passe, elle permet de trier l'intégralité de la liste des élèves, selon la moyenne ou la médiane, au choix.

Classe de tests :

Nous avons évidemment mis en place une classe contenant des tests pour notre programme, comme demandé dans le sujet. Cette classe s'occupe donc de la création de plusieurs élèves, professeurs, promotions, de l'attribution des élèves à leur promotion respective, de la création de notes pour les élèves, de l'affichage complet des informations d'un élève, de l'affichage de la liste des élèves d'une promotion, de la recherche d'un élève grâce à son identifiant, du classement des élèves d'une promotion par moyenne et médiane, de manière croissante et décroissante, et de l'affichage de ces classements par promotion.

Gestion des données CSV :

Dans la partie deux du projet, les données ne sont plus codées en dur, mais lues dans des fichiers .csv contenus dans la racine du projet.
On utilise ensuite des Scanners pour lire les données dans le fichier. On crée par la suite des tableaux de chaines de caractères, en utilisant la fonction split() et en spécifiant le séparateur, ici le caractère « ; ».
On peut ainsi gérer la lecture de données dans les fichiers externes.

On gère aussi l'écriture dans des fichiers externes, en particulier lorsque des nouvelles notes sont créées. Pour ce faire nous utilisons un FileWriter, qui ajoute les nouvelles données à la fin du fichier existant.

Conclusion

Pour conclure, les objectifs de la partie 1 et 2 ont été atteints. Nous avons rencontré des difficultés pour gérer la lecture de données dans un fichier externe, mais nous avons su les surmonter. Cependant, à partir de la partie 3, les principes de programmation à utiliser nous sont totalement étrangers et nous n'avons pas réussi à les maîtriser à temps pour réaliser cette partie. Enfin, nous nous sommes retrouvés à cours de temps pour réaliser une interface graphique complète pour notre projet, nous n'avons donc pas pu réaliser la partie 4.

Annexes

Partie 1 :

```
////////////////////////////
Creation de 5 groupes de promo
////////////////////////////

 ////////////////////////////
Creation de 9 élèves
////////////////////////////

 ////////////////////////////
Test de bonne création d'un élève:
L'eleve 1 s'appelle Jean Huault
////////////////////////////


////////////////////////////
Creation de 4 professeurs
////////////////////////////

 ////////////////////////////
Ajout des eleves a leur promo respective
////////////////////////////

 ////////////////////////////
Creation de 4 notes pour chaque eleve de L3
////////////////////////////

 ////////////////////////////
Affichage complet d'un eleve
(Jean,Huault)
id:0
Promotion: L3
notes: Java: 20.0 Maths: 17.0 Physique: 12.0 Anglais: 16.0  : -1.0  : -1.0  : -1.0  : -1.0  : -1.0  : -1.0
moyenne: 16.25
mediane: 16.5
correcteur(s): [[(Hayao,Miyazaki), (/,/), (John,Doe), (Wes,Anderson), (Michael,Bay)]]
////////////////////////////


////////////////////////////
Affichage de tous les eleves de la promo L3La promo L3 est composée de
Jean Huault
Paul-Antoine Inghelbrecht
Augustin Bouvet
////////////////////////////


////////////////////////////
Recherche dans la promo M1 de l'eleve avec identifiant = 7
L'eleve s'appelle Thomas Faugier
////////////////////////////
```

```
Tri de la promo L3 par moyenne croissante:
Augustin Bouvet: 14.5
Paul-Antoine Inghelbrecht: 16.0
Jean Huault: 16.25


Tri de la promo L3 par moyenne decroissante:
Jean Huault: 16.25
Paul-Antoine Inghelbrecht: 16.0
Augustin Bouvet: 14.5


Tri de la promo L3 par mediane croissante:
Augustin Bouvet: 14.0
Paul-Antoine Inghelbrecht: 16.0
Jean Huault: 16.5


Tri de la promo L3 par mediane decroissante:
Jean Huault: 16.5
Paul-Antoine Inghelbrecht: 16.0
Augustin Bouvet: 14.0
```

Résultats des fonctions de test de la première version du programme

Partie 2 :

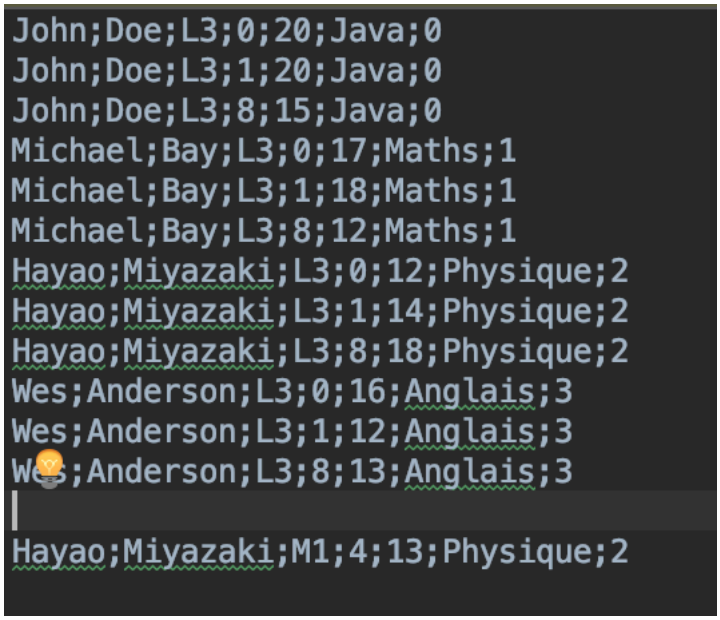
```
////////////////////////////
Import des professeurs
////////////////////////////

 ////////////////////////////
Test de bon import d'un professeur:
////////////////////////////

////////////////////////////
Import des élèves
////////////////////////////

 ////////////////////////////
Test de bon import d'un élève:
L'eleve 1 s'appelle Jean Huault
////////////////////////////


////////////////////////////
Import des promotions
////////////////////////////

 ////////////////////////////
Attribution des élèves à leur promotion respective
////////////////////////////

 ////////////////////////////
Affichage de tous les élèves de la promotion L3
La promo L3 est composée de
Jean Huault
Paul-Antoine Inghelbrecht
Augustin Bouvet
////////////////////////////
```

```
//////////////////////////
Import des evalutations
//////////////////////////
//////////////////////////
Affichage complet d'un eleve
(Jean,Huault)
id:0
Promotion: L3
notes: Java: 20.0 Maths: 17.0 Physique: 12.0 Anglais: 16.0  : -1.0  : -1.0  : -1.0  : -1.0  : -1.0  : -1.0
moyenne: 16.25
mediane: 16.5
correcteur(s): [[(Michael,Bay), (/,/), (Wes,Anderson), (Hayao,Miyazaki), (John,Doe)]]
//////////////////////////


//////////////////////////
Recherche dans la promo M1 de l'eleve avec identifiant = 7
L'eleve s'appelle Thomas Faugier
//////////////////////////


//////////////////////////
Ajout d'une nouvelle note à l'élève 4, sauvegarde dans le fichier evaluations

Process finished with exit code 0
```

Résultats des fonctions de test de la version 2 du programme

```
John;Doe;L3;0;20;Java;0
John;Doe;L3;1;20;Java;0
John;Doe;L3;8;15;Java;0
Michael;Bay;L3;0;17;Maths;1
Michael;Bay;L3;1;18;Maths;1
Michael;Bay;L3;8;12;Maths;1
Hayao;Miyazaki;L3;0;12;Physique;2
Hayao;Miyazaki;L3;1;14;Physique;2
Hayao;Miyazaki;L3;8;18;Physique;2
Wes;Anderson;L3;0;16;Anglais;3
W;Anderson;L3;1;12;Anglais;3
Wes;Anderson;L3;8;13;Anglais;3
```

Fichier .csv des évaluations avant l'exécution de l'exécution du programme



Fichier .csv des évaluations après l'exécution de l'exécution du programme