



RAPPORT DU PROJET DE JAVA 2

## **EvaluationsEleves/Professeurs**



**Adin HRELJA**

**Abdou KANE**

**Promotion 2021**

**Groupe 6**

## SOMMAIRE

1. Introduction .....	3
2. Les diagrammes.....	4
3. Analyse fonctionnelle générale.....	5
4. Analyse fonctionnelle détaillée (Choix du csv, de JfreeChart,etc).....	6
5. Conclusion.....	11

### Annexes

**Quelques définitions.**

**Sites web et ressources utilisées.**

**Capture d'écran des 4 versions demandées.**

## 1. Introduction

Durant ce semestre 6 à l'EFREI, nous devons réaliser un projet en JAVA qui a pour objectif principal de mettre en pratique nos connaissances sur ce langage de programmation qui est aujourd'hui très utilisé dans le monde informatique.

Pour ce faire, nous devons concevoir et coder une application qui gère une liste d'élèves avec leurs notes et les professeurs qui les évaluent. Ce genre d'application est assez complexe car il contient des informations (donc une base de données) sur les étudiants et les professeurs qu'il va falloir liées. Notre objectif est donc de faire en sorte que chaque étudiant puisse consulter l'ensemble de ses notes et sa moyenne générale. Mais aussi, faire en sorte que chaque professeur puisse attribuer ou modifier une note à un élève.

Au-delà de ce système de notation, nous devons aussi classer les élèves d'une promotion par ordre croissant ou décroissant en fonction de la moyenne et de la médiane. Enfin, nous devons modéliser le bulletin de note d'un élève via des graphiques et un histogramme.

C'est la raison pour laquelle nous avons eu recours à quelques fonctionnalités de JAVA comme la lecture de données via des fichiers csv ou via une base de données créée dans notre IDE. Nous avons pu aussi utiliser la fonctionnalité jFreeChart. Ce dernier nous a permis en grande partie de modéliser les bulletins de notes et de créer les histogrammes pour les moyennes de chaque élève dans chaque matière.

Ainsi, nous avons décliné notre programme en 4 versions :

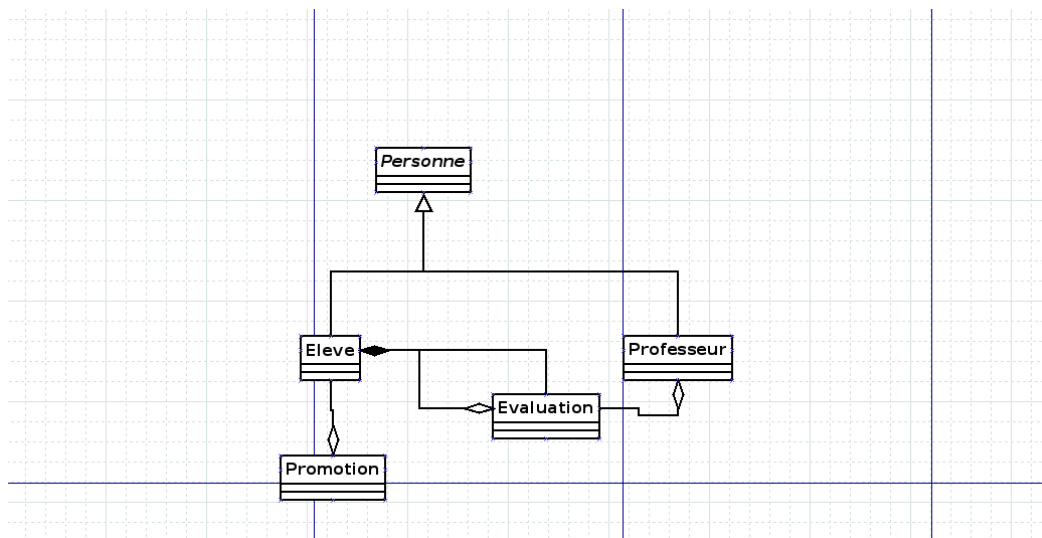
- La première version affiche un mode console avec un jeu de tests fournis.
- La deuxième version effectue une gestion de données en mode console dans des fichiers au format csv (accompagné d'un mode console interactif)
- La troisième version affiche les statistiques diverses (bulletins, histogrammes des moyennes et médianes, etc.)
- La quatrième version reprend les fonctionnalités de la troisième version mais elle permet de sauvegarder les données saisies dans un fichier csv avec une interface graphique encore mieux soignée.

Nous allons ainsi dans ce rapport vous présenter les différentes manipulations effectuées et fonctionnalités utilisées pour mener à bien ce projet.

## 2. Les diagrammes.

### Diagramme des classes :

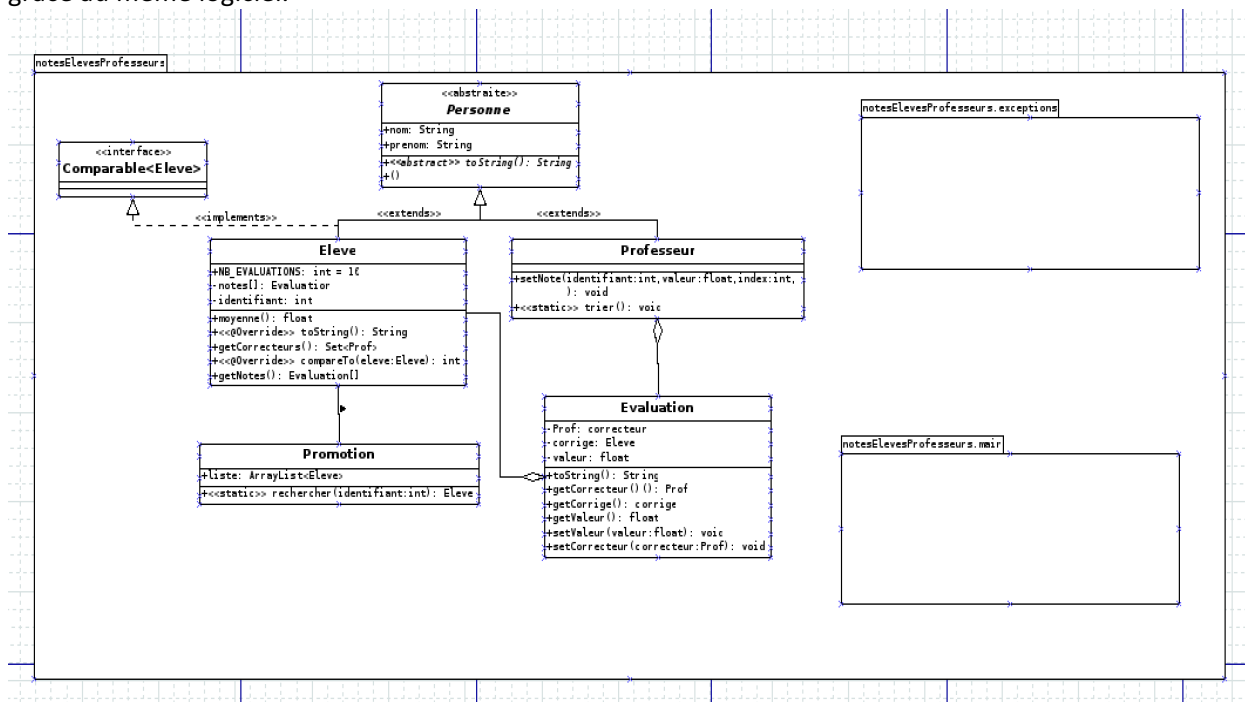
Ci-dessous se trouve le diagramme des classes qu'on a dessiné grâce au logiciel Dia Diagram Editor :



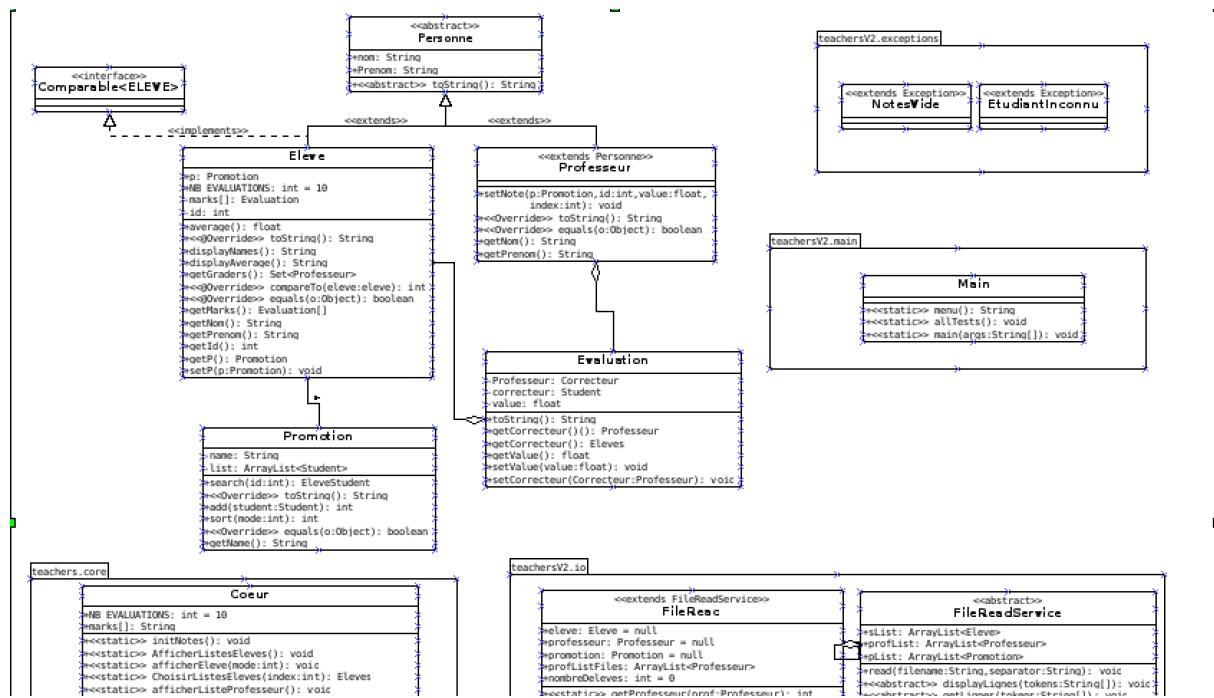
Nous pouvons ainsi y voir les relations de dépendances entre les classes.

### Relation entre classes et attributs :

Ci-dessous se trouve le diagramme qui montre les relations entre les classes et les attributs dessiné grâce au même logiciel.



## Relation entre classes, attributs et méthodes :



### 3. L'analyse fonctionnelle générale.

Les principales données traitées dans le programme sont les élèves, les professeurs, les notes et les promotions. De ce fait, nous avons créés des classes qui portent le nom de ces données. Ces classes sont bien organisées dans le programme (cf. diagramme des classes et attributs). Le programme permettra ainsi à un professeur de rechercher un élève d'une promotion et de lui attribuer ou modifier sa note mais aussi à un élève de consulter ses notes et son bulletin où il peut voir sa moyenne et sa médiane. Il permet aussi d'ajouter ou de supprimer un élève. Cependant, cela ne peut être fait que par un professeur. Un élève n'a pas accès à cette fonctionnalité. De plus, nous pouvons trier ses données par ordre croissant ou décroissant.

Enfin, Ces données peuvent être lues, modifiées ou supprimées à l'aide d'un fichier csv. Ils sont ensuite modélisés sous forme d'histogramme. Pour leur modélisation dans notre programme, nous avons utilisés jFreeChart.

Chaque professeur est identifié par son nom et son prénom et chaque élève est identifié grâce à son ID. Pour affecter un élève à une promotion, il suffit juste de spécifier son id dans la promotion. L'id est donc la clé primaire de l'élève comme on l'a appris en cours de base de données 2. Pour donner une nouvelle évaluation, il faut connaître le professeur, l'élève, la matière et la note.

Pour finir, nous avons utilisé la classe Scanner dans la version 1 et 2 du projet, JfreeChart dans la version 3 et jTable dans la version 4. L'utilisation de ces derniers ont été indispensables pour avoir le squelette complet de notre projet.

## 4. L'analyse fonctionnelle détaillée.

Nous allons maintenant faire l'analyse fonctionnelle détaillée de notre projet.

### - Pour la première et deuxième version :

Comme chaque professeur et chaque élève ont comme propriétés communes le prénom et le nom, on choisit de créer une classe **Personne** avec nom et prénom en paramètre.

Afin de récupérer les paramètres nous devons définir les accesseurs en lecture pour le nom et prénom, c'est-à-dire définir les méthodes suivantes : String **getLastName()**{} et String **getFirstName()**{}.

Afin de respecter l'affichage demandé, nous devons redéfinir la méthode **toString** pour l'affichage en console des élèves et des professeurs. C'est-à-dire qu'il retourne sous cette forme : (Soleil, Tournesol) pour le professeur de prénom Soleil et de nom Tournesol et (Jean, Duval) pour l'élève de prénom Jean et de nom Duval.

Nous devons maintenant créer la classe Evaluation, dans cette classe nous avons 4 attributs :

1. Topic (matière)
2. Grade (la note)
3. studentID (le numéro unique d'identification)
4. professeur (issus de l'objet professeur qui corrige l'élève en question)

A l'aide des 4 attributs nous pouvons créer l'objet Evaluation en utilisant le mot clef « this ».

Permettant de désigner l'instance courante de la classe elle-même.

Nous avons par la suite dû refaire la méthode **toString** pour afficher l'évaluation de cette façon :

((Jean, Duval) (Soleil, Tournesol) mathématiques 12.0)

Chaque **élève** a comme attributs suivants : (numéro d'identifiant (unique), son nom, son prénom, sa date de naissance, ses évaluations avec leur moyenne et leur médiane).

Notre classe **Eleve** est constitué de :

- Une constante de classe NB\_EVALUATIONS de type entier et valant 10.
- Un constructeur public à 5 paramètres permettant de donner un nom, un prénom et une date de naissance (jour, mois, année) à un élève
- Un accesseur en lecture pour l'identifiant de l'élève
- Une méthode **moyenne** et **médiane** calculant et retournant respectivement la moyenne et la médiane en prenant compte de l'**exception IllegalStateException** lancée si l'élève à 0 note.
- La méthode **getCorrecteurs** permettant de ranger dans une instance de la classe **HashSet**, l'ensemble des correcteurs ayant évalué un élève. Elle retourne cette collection.

Ensuite nous devons redéfinir de la class **HashSet** avec une méthode **toString** afin d'afficher une collection de professeurs de la manière suivante :

[(Max, La Menace), (Soleil, Tournesol)]

Nous avons dû ensuite apporter des modifications à la classe **Eleve** pour la méthode **toString** telle qu'elle retourne par exemple :

Résultat en console

```

run:
(Abdou, Kane)
Promotion: 2021
Id: 1
Notes: (13.0) (15.0)
Correcteurs(s): [Olga Mekhelova, Patrick Teller]
Moyennes: 14.0

(Adin, Hrelja)
Promotion: 2021
Id: 2
Notes: (18.0) (16.0) (15.0)
Correcteurs(s): [Patrick Teller]
Moyennes: 16.333334

(Wesh, Morray) n'a pas de notes !

Pas d'eleve avec un id5 dans la promotion 2021 !

(Abdou, Kane), Average: 14.0
(Adin, Hrelja), Average: 16.333334
(Wesh, Morray) doesn't have marks !

```

Maintenant nous allons coder la classe **Promotion** qui contient :

- Un attribut **nom** de type String
- Un attribut **eleves** d'un container générique (afin d'obtenir une collection d'élèves)
- Un constructeur à un paramètre (ici sera le **nom** de la promotion)
- Des accesseurs en lecture et écriture pour le **nom** de la promotion
- Un accesseur en lecture **getEleves** avec en signature : `Set<Eleve> getEleves()` permettant de ranger dans une instance de la classe l'ensemble des élèves d'une promotion. Celle-ci retourne une collection.
- Une méthode **rechercher** afin de rechercher un élève dans la collection des élèves selon son **identifiant**.

Nous devons ensuite créer une méthode **setNote** afin de modifier la i ème note d'un élève :

En paramètre nous avons :

1. Promotion
2. Identifiant de l'élève
3. Valeur de la note
4. Indice i

A partir du numéro d'identifiant de l'élève, cette méthode aura pour but de rechercher l'élève en question pour pouvoir ensuite modifier sa note :

- Si l'élève recherché n'existe pas, l'**exception IllegalStateException** est lancée.
- Si l'élève existe et si la note d'indice i existe aussi, alors la note est modifiée.
- Si l'élève existe et si la note d'indice i n'existe pas alors elle est créée et l'évaluation est ajoutée dans la liste des autres évaluations existantes.

Par la suite du projet nous avons dû créer des méthodes dans la classe **Promotion** afin de classer par ordre croissant puis décroissant selon la moyenne et la médiane de leurs notes.

Pour finir nous avons réalisé une classe de test où nous retrouvons :

- Plusieurs élèves et plusieurs professeurs créés
- Ranger les élèves dans leur promotion
- Mettre des notes aux élèves tout en indiquant les correcteurs
- Afficher un élève avec son nom sa promotion, ses correcteurs, ses notes, leur moyenne et leur médiane
- Afficher tous les élèves d'une promotion
- Rechercher un élève par son identifiant puis l'afficher
- Classer les élèves par ordre croissant puis décroissant de leur moyenne et de leur médiane
- Afficher tous les élèves d'une promotion selon les classements effectués

Vous pouvez vous référer à la capture d'écran de la console ci-dessous pour cela :

```
run:
===== Menu =====
0- Faire les tests! (File .csv empty)
1- Afficher tous les eleves
2- Afficher tous les professeurs
3- Afficher les promotions
4- Afficher les eleves (nom, promotion, notes, correcteurs)
5- Afficher un eleve avec son nom et sa moyenne
6- Trier les eleves d'une promotion
7- Gerer les notes
8- Gerer les eleves
9- Gerer les professeurs
10- Gerer les promotions
11- Sauvegarder le fichier
12- Quitter le Programme
```

#### **Affichage par ordre croissant :**

```
Ordre croissant :
=====
PROMOTION
=====
(Abdou, Kane)
Promotion: 2021
Id: 1
Notes: (13.0) (15.0)
Correcteurs(s): [Olga Mekhelova, Patrick Teller]
Moyennes: 14.0
(Adin, Hrelja)
Promotion: 2021
Id: 2
Notes: (18.0) (16.0) (15.0)
Correcteurs(s): [Patrick Teller]
Moyennes: 16.333334
```

#### **Affichage par ordre décroissant :**

```
Ordre descendant :
=====
PROMOTION
=====
(Adin, Hrelja)
Promotion: 2021
Id: 2
Notes: (18.0) (16.0) (15.0)
Correcteurs(s): [Patrick Teller]
Moyennes: 16.333334
(Abdou, Kane)
Promotion: 2021
Id: 1
Notes: (13.0) (15.0)
Correcteurs(s): [Olga Mekhelova, Patrick Teller]
Moyennes: 14.0
```

On remarque bien ici que c'est l'ID qui fait que l'affichage soit croissant ou décroissant.

- **Pour la troisième et quatrième version**



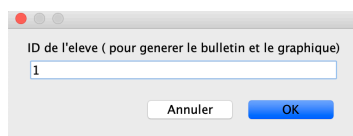
Au lieu d'utiliser une base de données, nous avons utilisé les fichiers csv pour sauvegarder, et lire les données. Pour ce faire, nous avons écrit une classe pour la lecture et l'écriture globale sur les fichiers csv : la classe `FileRead` et `FileReadService` dans le fichier `io` de la version 3. Dans ces classes, nous utilisons la fonction `ArrayList` `<ArrayList<String>>`.

Cela nous permet en effet de véhiculer les informations en écriture ou lecture. De plus, avec cette fonction, il est facile de trier mais aussi de modifier les informations. Le stockage dynamique dans le programme utilise également la fonction `ArrayList`.

Nous avons aussi fait en sorte que les données soient automatiquement sauvegardées dans le fichier csv lorsqu'on ferme une fenêtre. Enfin, la fonction `ArrayList` est plus manipulable que la fonction `Array` d'où la décision d'utiliser cette dernière.

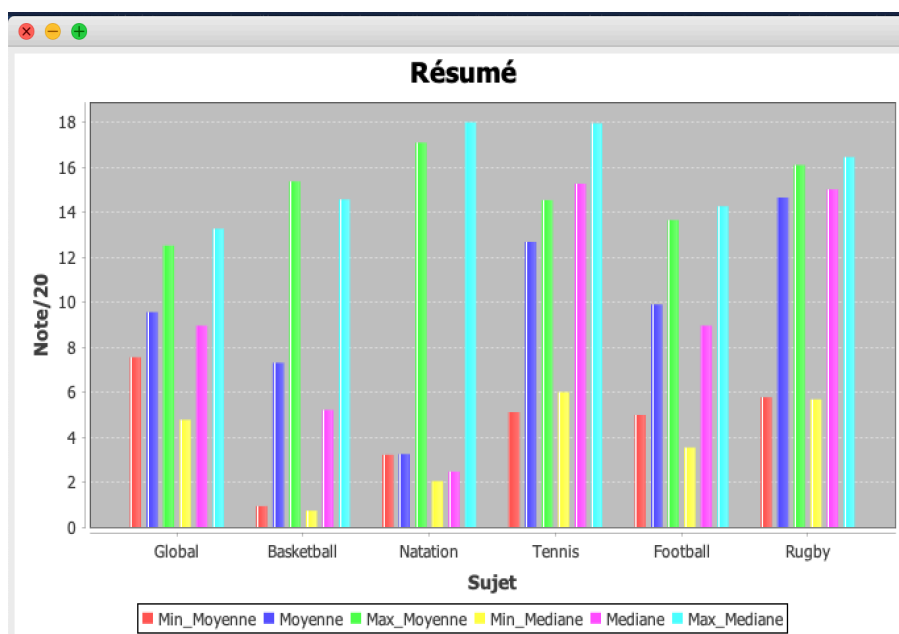
Pour la comparaison, nous avons réécrit la méthode `equals` et créé un nouveau comparateur. Ce qui nous sera d'ailleurs utile pour faire l'ordre de croissance ou de décroissance.

Vu que chaque élève est lié à son ID, le programme de test (v3 et v4) nous demande l'id de l'élève qu'on cherche dès son exécution. (cf. image ci-dessous)



Nous avons utilisé ensuite `jFreeChart` pour modéliser les données. Il suffit de télécharger quelques fichiers `.jar` dans leur site officiel et de les implémenter dans notre code pour avoir le rendu ci-dessous.

Ainsi, nous avons la moyenne et la médiane pour chaque élève et matière enseignée ainsi que leurs valeurs maximums et minimums.



Pour la dernière partie du projet, on reprend la version 3 de ce dernier mais on pourra modifier les notes directement et les sauvegarder dans le fichier csv avec de les modéliser. Nous avons donc utilisé en même temps ArrayList dans les classes FileRead et Jfreechart.

### Modification d'une note dans le programme

Bulletin de Kane Abdou						
Sujet	Min_Moyenne	Moyenne	Max_Moyenne	Min_Mediane	Mediane	Max_Mediane
Global	7.55	9.56	12.52	4.77	8.96	13.27
Basketball	0.93	7.3	15.37	0.73	5.21	14.57
Natation	3.22	3.25	17.09	2.05	2.47	18.0
Tennis	5.11	12.68	14.54	6.0	15.27	17.95
Football	4.99	9.9	13.65	3.54	8.96	14.27
Rugby	5.77	14.65	16.1	5.68	15.01	16.45

Graphique
Fermer

### Bulletin complet

Bulletin de Kane Abdou						
Sujet	Min_Moyenne	Moyenne	Max_Moyenne	Min_Mediane	Mediane	Max_Mediane
Global	7.55	9.56	12.52	4.77	8.96	13.27
Basketball	0.93	7.3	15.37	0.73	5.21	14.57
Natation	3.22	3.25	17.09	2.05	2.47	18.0
Tennis	5.11	12.68	14.54	6.0	15.27	17.95
Football	4.99	9.9	13.65	3.54	8.96	14.27
Rugby	5.77	14.65	16.1	5.68	15.01	16.45

Graphique
Fermer

## Conclusion

Pour conclure, ce projet de Java nous a permis de mettre enfin en pratique nos connaissances étudiées en cours. De plus, il nous a permis de manipuler l'écriture et la lecture des fichiers csv dans un programme en Java. Il faut également noter l'utilisation de JFreeChart qui est aussi une première pour nous dans un projet de cet envergure.

La grande partie des objectifs sont atteints et nous en sommes très satisfaits. En effet, on a rencontré des problèmes par ci par là mais grâce à Internet et à quelques camarades de classes, nous avons su réglés tous ces problèmes.

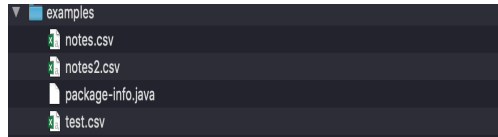
Par exemple, nous avons perdu énormément de temps dans la partie modélisation juste parce que nous ne savions pas comment implémenter JFreeChart dans notre code correctement. Il nous arrivait également que des codes sans erreur ne compile pas correctement. En bref, ce sont des heures de travail mais pas en vain car ces heures nous ont permis de consolider grandement nos connaissances en Java.

Ce projet nous a permis aussi de nous améliorer dans l'organisation du travail. En effet, chacun d'entre nous avait une mission précise et des deadlines précises qu'il devait respecter obligatoirement. Cela nous a permis d'avancer rapidement sans nous perdre dans notre travail

## ANNEXES :

**Jfreechart :** JFreeChart est une API Java permettant de créer des graphiques et des diagrammes de très bonne qualité. Cette API est open source et sous licence LGPL. En revanche, la documentation est payante.

**Fichier csv :** Un fichier csv c'est un fichier de type excel. Dans notre programme, les fichiers csv se trouvent dans le document exemples comme indique ci-dessous.



### Sites web et ressources utilisés :

<http://www.jfree.org/jfreechart/>

<https://docs.oracle.com/javase/tutorial/>

<https://www.geeksforgeeks.org/java/>

<https://thierry-leriche-dessirier.developpez.com/tutoriels/java/afficher-graphe-jfreechart-5-min/>

<https://thierry-leriche-dessirier.developpez.com/tutoriels/java/csv-avec-java/>

### Capture d'écran récapitulatif des 4 versions demandées.

#### Version 1

```
run:
(Abdou, Kane)
  Promotion: 2021
  Id: 1
  Notes: (13.0) (15.0)
  Correcteurs(s): [Olga Mekhelova, Patrick Teller]
  Moyennes: 14.0

(Adin, Hrelja)
  Promotion: 2021
  Id: 2
  Notes: (18.0) (16.0) (15.0)
  Correcteurs(s): [Patrick Teller]
  Moyennes: 16.333334

(Wesh, Morray) n'a pas de notes !

Pas d'eleve avec un id5 dans la promotion 2021 !

(Abdou, Kane), Average: 14.0
(Adin, Hrelja), Average: 16.333334
(Wesh, Morray) doesn't have marks !
```

---

Ordre croissant :

```
=====
      PROMOTION
=====
```

```
(Abdou, Kane)
  Promotion: 2021
  Id: 1
  Notes: (13.0) (15.0)
  Correcteurs(s): [Olga Mekhelova, Patrick Teller]
  Moyennes: 14.0
(Adin, Hrelja)
  Promotion: 2021
  Id: 2
  Notes: (18.0) (16.0) (15.0)
  Correcteurs(s): [Patrick Teller]
  Moyennes: 16.333334
```

---

Ordre descendant :

=====

PROMOTION

=====

(Adin, Hrelja)

Promotion: 2021

Id: 2

Notes: (18.0) (16.0) (15.0)

Correcteurs(s): [Patrick Teller]

Moyennes: 16.333334

(Abdou, Kane)

Promotion: 2021

Id: 1

Notes: (13.0) (15.0)

Correcteurs(s): [Olga Mekhelova, Patrick Teller]

Moyennes: 14.0

## Version 2

run:

===== Menu =====

0- Faire les tests! (File .csv empty)

1- Afficher tous les eleves

2- Afficher tous les professeurs

3- Afficher les promotions

4- Afficher les eleves (nom, promotion, notes, correcteurs)

5- Afficher un eleve avec son nom et sa moyenne

6- Trier les eleves d'une promotion

7- Gerer les notes

8- Gerer les eleves

9- Gerer les professeurs

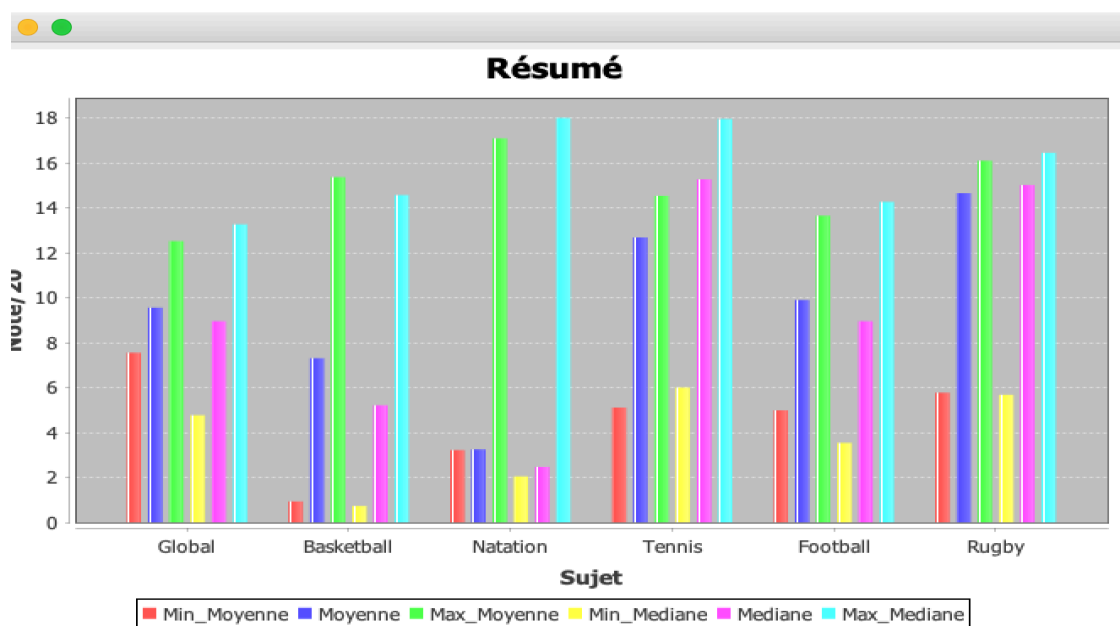
10- Gerer les promotions

11- Sauvegarder le fichier

12- Quitter le Programme

notes							
Chercher dans la feuille							
Accueil Insertion Mise en page Formules Données >>							
Presse-papiers Police Alignement Numérique Mise en forme conditionnelle Mettre sous forme de tableau Styles de cellule							
C7 Hrelja							
A	B	C	D	E	F	G	H
1	Abdou	Kane		1			
2		2021					
3	paumier	sebastien	zipstein	marc			
4		12			14		
5		0			1		
6		13.333333					
7	Adin	Hrelja		2			
8		2 biniou					
9	paumier	sebastien	zipstein	marc			
10		15			16		
11		1			0		
12		6 13.5					
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							

## Version 3



## Version 4

Bulletin de Kane Abdou						
Sujet	Min_Moyenne	Moyenne	Max_Moyenne	Min_Mediane	Mediane	Max_Mediane
Global	7.55	9.56	12.52	4.77	8.96	13.27
Basketball	0.93	7.3	15.37	0.73	5.21	14.57
Natation	3.22	3.25	17.09	2.05	2.47	18.0
Tennis	5.11	12.68	14.54	6.0	15.27	17.95
Football	4.99	9.9	13.65	3.54	8.96	14.27
Rugby	5.77	14.65	16.1	5.68	15.01	16.45

GraphiqueFermer

Bulletin de Kane Abdou						
Sujet	Min_Moyenne	Moyenne	Max_Moyenne	Min_Mediane	Mediane	Max_Mediane
Global	7.55	9.56	12.52	4.77	8.96	13.27
Basketball	0.93	7.3	15.37	0.73	5.21	14.57
Natation	3.22	3.25	17.09	2.05	2.47	18.0
Tennis	5.11	12.68	14.54	6.0	15.27	17.95
Football	4.99	9.9	13.65	3.54	8.96	14.27
Rugby	5.77	14.65	16.1	5.68	15.01	16.45

GraphiqueFermer