

# Projet Evaluations Elevés/Professeurs



JABER Ahmad  
GUILLEMIN Lucas

# SOMMAIRE

- I. Introduction.
- II. Diagramme de classes.
- III. Analyse fonctionnelle.
- IV. Choix de gestion des données.
- V. Diagrammes statistiques.
- VI. Conclusion.

## I. Introduction

Actuellement en 3eme année à l'EFREI, notre matière JAVA 2 nous permet de réaliser un projet d'évaluation élèves/professeurs.

Dans ce projet notre but était de chercher à gérer une liste d'élèves avec leurs notes et les professeurs qui les évaluent. Faut savoir que les élèves et les professeurs possèdent des propriétés communes, ici on parle du Nom et du Prénom.

Les élèves ont des propriétés spécifiques pour eux, tel qu'un numéro d'identifiant unique à chaque élève, un nom et un prénom ainsi que la date de naissance (jj/mm/aa), il possède aussi des évaluations, des notes, des moyennes et des medianes. Pour chaque évaluation, sont indiqués la matière concernée, la valeur de la note, l'élève corrigé et le professeur correcteur

Les élèves sont classés par promotion, les élèves d'une même promotion peuvent être classés selon leur note moyenne et leur note médiane, des classements par matière sont également possibles. Chaque élève ne peut consulter que ses propres notes.

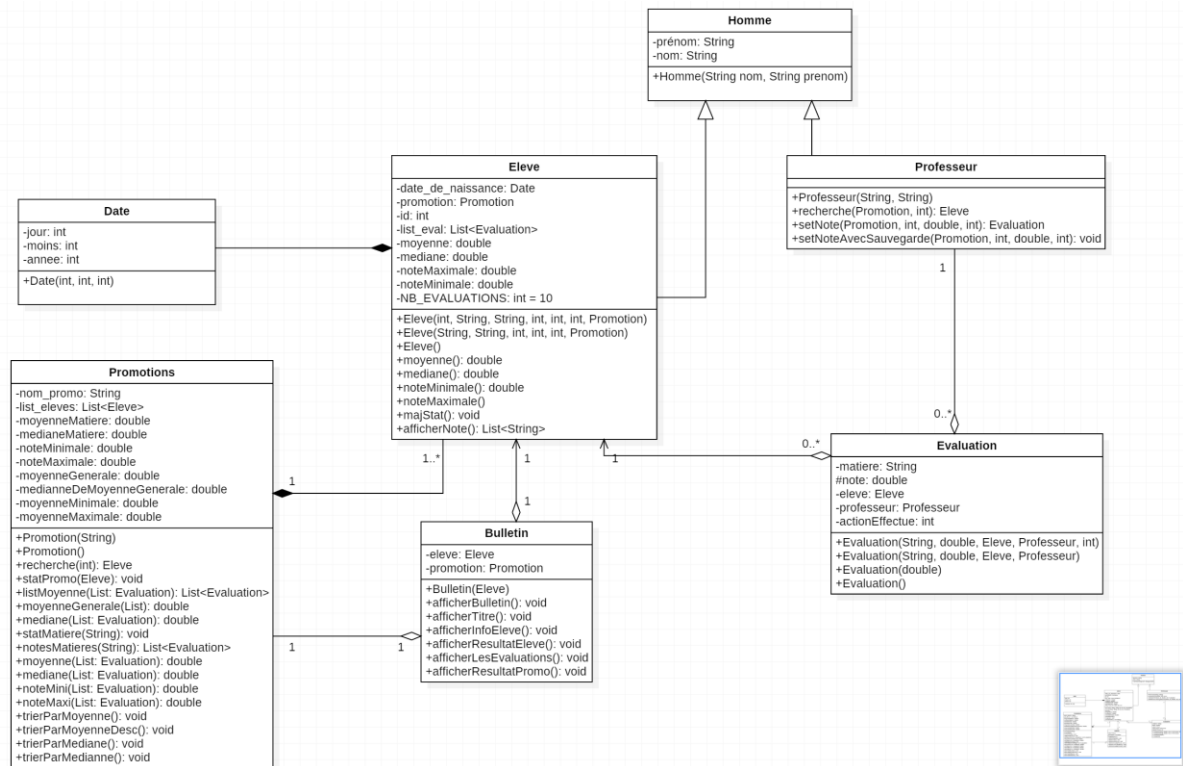
Le professeur peut effectuer les opérations suivantes :

- Rechercher un élève dans la promotion à partir de son numéro d'identifiant.
- Modifier des notes d'un élève recherché.

The screenshot shows a Java Swing window titled "Nabil Andrieux". Inside the window, there is a table with the following columns: "Prénom", "Nom de Famille", "Promotion", and "ID". The table contains one row with the values "Ahmad", "Jaber", "3,1", and "1". To the right of the table, there are two input fields labeled "Eleve ID:" and "Promotion elev:". Below these input fields are two buttons labeled "Ajouter n..." and "Ajouter el...".

Prénom	Nom de Famille	Promotion	ID
Ahmad	Jaber	3,1	1

## II. Diagramme de classe



## III. Analyse fonctionnelle

Pour réaliser ce projet, nous avons utilisé plusieurs classes :

- Une classe élève, qui est liée à la base de données qu'on a faite. Dans cette classe nous avons toutes les caractéristiques d'un élève (nom, prénom, date de naissance et les notes). Cette classe appelle la base de données qui contient toutes les informations préciser précédemment. Chaque élève ce caractérise par un nom, un prénom, une date de naissance et des notes différentes des autres.

```

1 import java.text.DecimalFormat;
2
3 public class Eleve extends Personne {
4     //They have a com.JavaProject.BasicClasses.Date of birth stored
5     private Date dateNaissance;
6
7     //Since we'll use a database, the students need an ID field
8     private String ID;
9
10    //They have a multiple evaluations to which they can access
11    private List<Evaluation> evaluations;
12
13    //They belong to a promotion, and they know to which one
14    private Promotion promotion;
15
16    //Constructor
17    public Eleve(int id, String nomFamille, String prenom, int annee, int mois, int jour, Promotion promotion){
18        super(nomFamille, prenom);
19        this.dateNaissance = new Date(annee, mois, jour);
20        this.promotion = promotion;
21        this.ID = Integer.toString(id);
22        this.evaluations = new ArrayList();
23        promotion.ajouterEleve(this);
24    }
25
26    //Getters
27    public String getID() { return ID; }
28
29    public Date getDateNaissance() { return dateNaissance; }
30
31    @Override
32    public String getNomFamille() {
33        return super.getNomFamille();
34    }
35
36    @Override
37    public String getPrenom() {
38        return super.getPrenom();
39    }
40 }

```

- Une classe date, pour afficher la date de naissance de chaque élève. Dans cette classe nous avons utilisé un constructeur pour l'initialisation et l'affichage du jour, du mois et de l'année de naissance de chaque élève.

```

1 public class Date {
2     private int year;
3     private int month;
4     private int day;
5
6     //Constructor
7     public Date(int year, int month, int day) {
8         this.year = year;
9         this.month = month;
10        this.day = day;
11    }
12
13    //Getters and setters
14    public int getYear() {
15        return year;
16    }
17
18    public void setYear(int year) {
19        this.year = year;
20    }
21
22    public int getMonth() {
23        return month;
24    }
25
26    public void setMonth(int month) {
27        this.month = month;
28    }
29
30    public int getDay() {
31        return day;
32    }
33
34    public void setDay(int day) {
35        this.day = day;
36    }
37
38    //HashCode, equals, toString
39 }

```

```

57 @Override
58 public String toString() {
59     return "(" + this.day + '/' + this.month + '/' + this.year + ")";
60 }
61 }
62
63

```

- Une classe bulletin, qui est liée à la classe élève et la base de données. Dans cette classe on reprend toutes les notes d'examen qu'un élève a passé et puis on les range dans un tableau et puis on calcul la moyenne

minimal et maximal ainsi que la médiane minimal et maximal et la note global.

```
1 import javax.swing.*;
2
3 public class Bulletin {
4     //Attributes
5     private Eleve eleve;
6     private Promotion promotion;
7     //Constructor
8     public Bulletin(Eleve s){
9         this.eleve = s;
10        this.promotion = s.getPromotion();
11    }
12
13    //Methods
14    public void creerDonneeBulletin(){
15        String[] colonnes = {"Sujet", "Min_Moyenne", "Moyenne", "Max_Moyenne", "Min_Mediane", "Mediane", "Max_Mediane"};
16        Object[][] donnees;
17
18        //Getting all the subjects the élève passed exams for
19        //Not taking care if a élève has less marks than another, since they should all have the same number of test
20        ArrayList<String> sujets = new ArrayList<>();
21
22        //Retrieving the averages, medians and the list of subject
23        for(Evaluation e : this.eleve.getEvaluations()){
24            //Getting the list of subject
25            if(!sujets.contains(e.getSubject())){
26                sujets.add(e.getSubject());
27            }
28        }
29
30        //Now we have the list of subjects we can initialize the data array
31        String[] tableau_sujet = new String[sujets.size()+1];
32        tableau_sujet[0] = "Global";
33        int i = 1;
34        for(String s : sujets) {
35            tableau_sujet[i] = s;
36            i++;
37        }
38    }
39}
```

- Une classe évaluation, dans laquelle nous avons les différentes matières, les correcteurs, la note et l'élève qui a effectué l'examen. Cette classe va nous servir à remplir le bulletin des élèves.

```
1 import java.text.DecimalFormat;
2
3 public class Evaluation {
4     //Attributes
5     private String subject;
6     private float mark;
7     private Professeur professeur;
8     private Eleve eleve;
9
10    //Constructor
11    Evaluation(String subject, float mark, Professeur professeur, Eleve eleve){
12        this.subject=subject;
13        this.professeur = professeur;
14        this.mark=mark;
15        this.eleve = eleve;
16    }
17
18    //Getters
19    public String getSubject() {
20        return subject;
21    }
22
23    public float getMark() {
24        return mark;
25    }
26
27    public Professeur getProfesseur() {
28        return professeur;
29    }
30
31    public Eleve getEleve() {
32        return eleve;
33    }
34
35    //Setter
36    public void setMark(float mark) { this.mark = mark; }
37
38    //HashCode, equals, toString overridden methods
39    @Override
40    public int hashCode() {
41        return subject.hashCode() + mark;
42    }
43
44    public boolean equals(Object o) {
45        if(o instanceof Evaluation) {
46            Evaluation e = (Evaluation) o;
47            return subject.equals(e.subject) && mark == e.mark;
48        }
49        return false;
50    }
51
52    public String toString() {
53        return "Evaluation [subject=" + subject + ", mark=" + mark + ", professeur=" + professeur + ", eleve=" + eleve + "]";
54    }
55}
```

- Une classe professeur, pour créer et saisir les caractéristiques des chaque professeurs. (Nom, prénom, ID, la promotion dans laquelle il enseigne)
- Une classe promotion, qui est liée aux élèves, aux professeurs ainsi qu'à la base de données. Dans cette classe ont reparti les élèves selon leur promotion, chaque promotion est à ses propres élèves, des élèves de promotions différentes ne peuvent pas être mélanger. Dans cette classe

on trie les élèves par leur moyenne et la médiane c'est-à-dire que les élèves sont classés dans l'ordre. (Moyenne la plus haute jusqu'à la moyenne la plus basse)

- Une classe automaticSQL, pour vérifier que tout est sauvegarder dans la base de données, ici on parle des professeurs des noms, des prénoms, des ID, des élèves, des évaluations...

```

134 public void toutSauvegarder(Set<Promotion> promotions, Set<Professeur> professeurs)
135     throws SQLException {
136     Statement stat = null;
137     ResultSet rs = null;
138     String requete;
139     try {
140         stat = con.createStatement();
141         //We start with the professeurs, since they're not linked to any other table
142         for(Professeur p : professeurs){
143             requete = "SELECT * from lecturer_table where id = " + p.getId() + "";
144             rs = stat.executeQuery(requete);
145             if(rs.next()){
146                 //The professeur was added during the session, we've to save it in database
147                 requete = "INSERT INTO lecturer_table VALUES (default, " + p.getPrenom() + ", " + p.getNomFamille() + ")";
148                 stat.executeQuery(requete);
149             }
150             else{
151                 //The lecturer was found, we've to save it's information in case they were modified
152                 requete = "UPDATE lecturer_table SET name = " + p.getPrenom() + ", last_name = " + p.getNomFamille() + " " +
153                     "WHERE id = " + p.getId() + " ";
154                 stat.executeQuery(requete);
155             }
156         }
157     }
158     //Now we do the same for promotions
159     for(Promotion p : promotions){
160         //verify the promotion is saved in db
161         requete = "SELECT id from promotion_table where name = " + p.getNomPromotion() + " ";
162         rs = stat.executeQuery(requete);
163         //the promotion wasn't found in database, we've to save it
164         if(rs.next()){
165             //We save the name of the promotion
166             requete = "INSERT INTO promotion_table VALUES (default, " + p.getNomPromotion() + ")";
167             stat.executeQuery(requete);
168         }
169         //the promotion was found, we save it's information in case they were modified (should not happen but whatever)
170         else{
171             requete = "UPDATE promotion_table SET name = " + p.getNomPromotion() + " WHERE id = " + p.getId() + " ";
172         }
173     }
174 }

```

- Une dernière classe Application, pour gérer l'affichage des informations dans des interfaces.

```

44 //We will have a table with all the students , the information of the lecturer and a panel to add a note/see the student
45 JPanel profPanel = new JPanel();
46 JPanel profInfo = new JPanel();
47 JPanel tableauEleve = new JPanel();
48 JPanel panelNote = new JPanel();
49 JPanel Panel = new JPanel();
50
51 profPanel.setPreferredSize(new Dimension(490,490));
52 profInfo.setPreferredSize(new Dimension(490,90));
53 tableauEleve.setPreferredSize(new Dimension(300,400));
54 panelNote.setPreferredSize(new Dimension(150,400));
55 Panel.setPreferredSize(new Dimension(490,400));
56
57 profPanel.setLayout(new BorderLayout());
58 Panel.setLayout(new BorderLayout());
59 String idProf = JOptionPane.showInputDialog(null,"ID prof");
60 professeur = null;
61 for(Professeur p : profs) {
62     if (p.getId() == Integer.parseInt(idProf)){
63         professeur = p;
64         break;
65     }
66 }
67 professeur.setPromotions(promo);
68 //LecturerInfo panel
69 profInfo.setLayout(new FlowLayout(FlowLayout.CENTER, 30,20));
70
71 JLabel prenom = new JLabel(professeur.getPrenom());
72 JLabel nomFamille = new JLabel(professeur.getNomFamille());
73 prenom.setPreferredSize(new Dimension(80,50));
74 nomFamille.setPreferredSize(new Dimension(80,50));
75
76 profInfo.add(prenom);
77 profInfo.add(nomFamille);
78
79 profPanel.add(profInfo,BorderLayout.NORTH);
80
81 //Lecturer table
82 //We need a table with field name, last name, promotion, Button to see student info, Button to add a note to the student

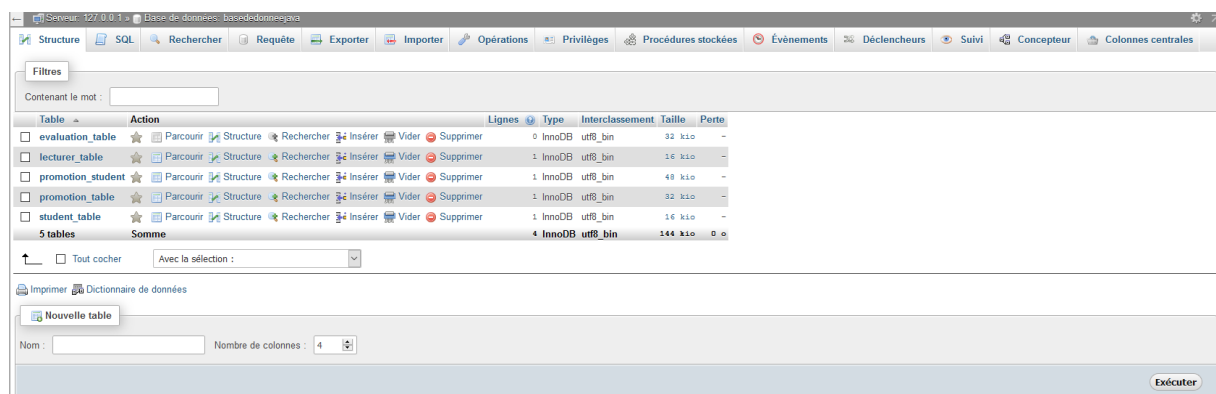
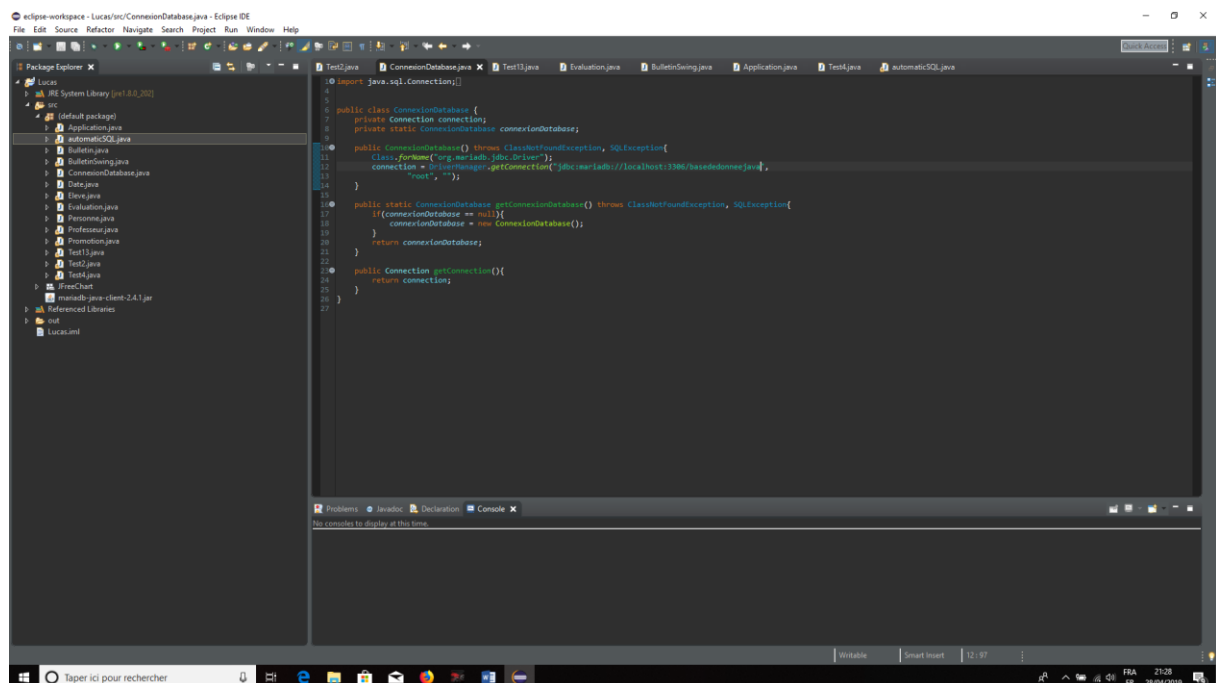
```

## IV. Choix de gestion de données

Pour ce projet nous avons fait le choix de ne pas utiliser de fichiers CSV, mais plutôt une base de données. En effet, les fichiers CSV peuvent être modifiés manuellement, ce qui, en plus de pouvoir engendrer des erreurs, est beaucoup moins sécurisé qu'une base de données. Pour ce faire, nous avons donc créé une base de données sous Maria DB.

Nous avons donc dû ajouter un fichier ConnectionDatabase, faisant la connexion à la Database.

Pour lancer le Projet, il faut donc mettre en route son serveur Xampp, ou Wamp, et lancer les services Apache et MySQL.

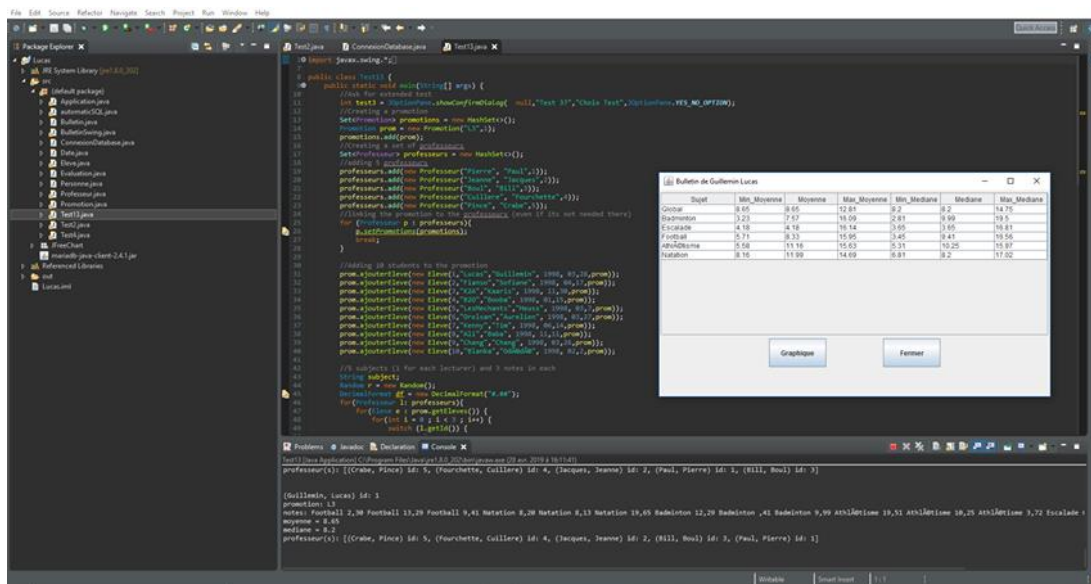




## V. Diagrammes statistiques

Le but de cette partie est de modéliser le bulletin de notes d'un élève. On doit créer un tableau qui comporte es évaluations, les moyennes minimal et maximal ainsi que les médianes minimal et maximal par matière enseignée et les globales sont à renseigner. Ensuite les notes obtenu (moyenne et médiane notamment) devaient être représenter graphiquement.

A partir des moyennes et des medianes on pouvait tracer des graphiques.

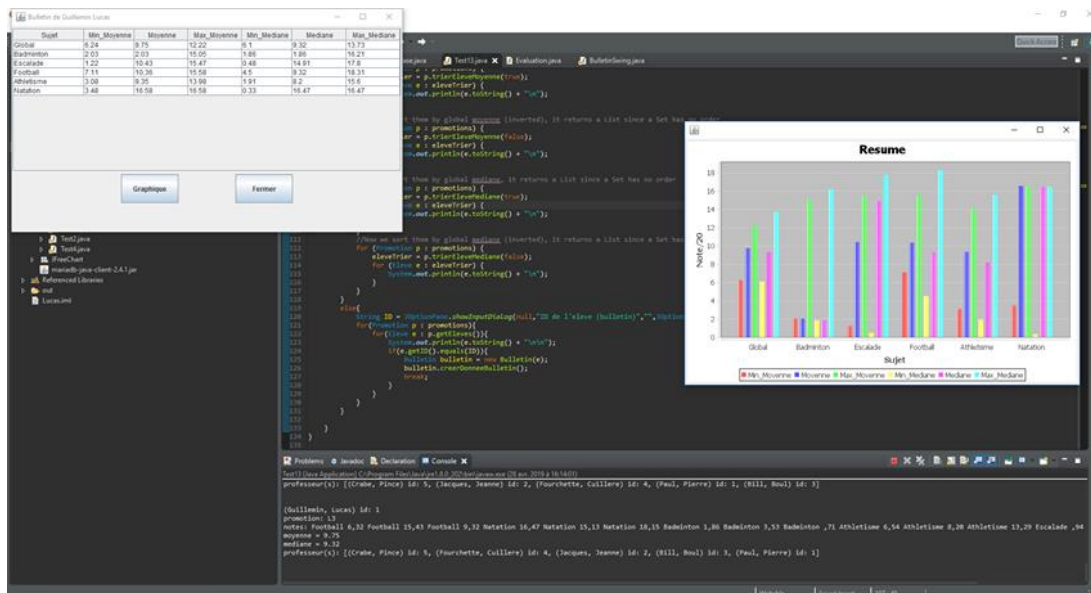


The screenshot shows a Java IDE with a project named 'Lecurand'. The main application window, titled 'Bulletin de Guillaume Lucas', displays a table with the following data:

Sujet	Min_Moyenne	Moyenne	Max_Moyenne	Min_Mediane	Mediane	Max_Mediane
Global	8.65	8.65	12.81	8.2	8.2	14.75
Badminton	3.23	3.57	16.59	2.81	6.89	18.5
Escalade	4.18	4.18	16.14	3.85	3.85	16.81
Football	5.73	8.33	15.95	3.45	8.41	18.56
Athlétisme	5.16	11.18	15.82	5.11	10.25	16.87
Natation	8.15	11.99	14.69	8.87	8.2	17.02

The code in the background shows the logic for generating this data, including creating a student, adding subjects, and calculating statistics.

Si on clique sur le bouton « graphique » on obtient :



The screenshot shows the same Java IDE, but now the application window displays a bar chart titled 'Resume'. The chart shows the 'Moyenne' (Average) for each subject, color-coded by subject: Global (red), Badminton (blue), Escalade (green), Football (yellow), Athlétisme (purple), and Natation (brown). The y-axis represents the 'Note (0)' and ranges from 0 to 15. The x-axis represents the 'Sujet' (Subject).

The code in the background shows the logic for generating this chart, including creating a student, adding subjects, and calculating statistics.

## Conclusion

Ce projet est à notre gout une réussite. Nos principaux objectifs ont été atteints. Nous avons réussi à utiliser de manière efficace les fonctionnalités de plusieurs software (Eclipse, xampp, phpMyAdmin).

Nous avons fait preuve d'une patience et d'une camaraderie qui ont été des atouts dans l'élaboration de notre projet. La répartition du travail et la présence de chacun à toutes les réunions ont permis un développement dans de très bonnes conditions.