Made for Web

Gestion des Etudiant Ecole

Java Project

MOUASSEIF MOUAD
ENS RABAT

Table des matières

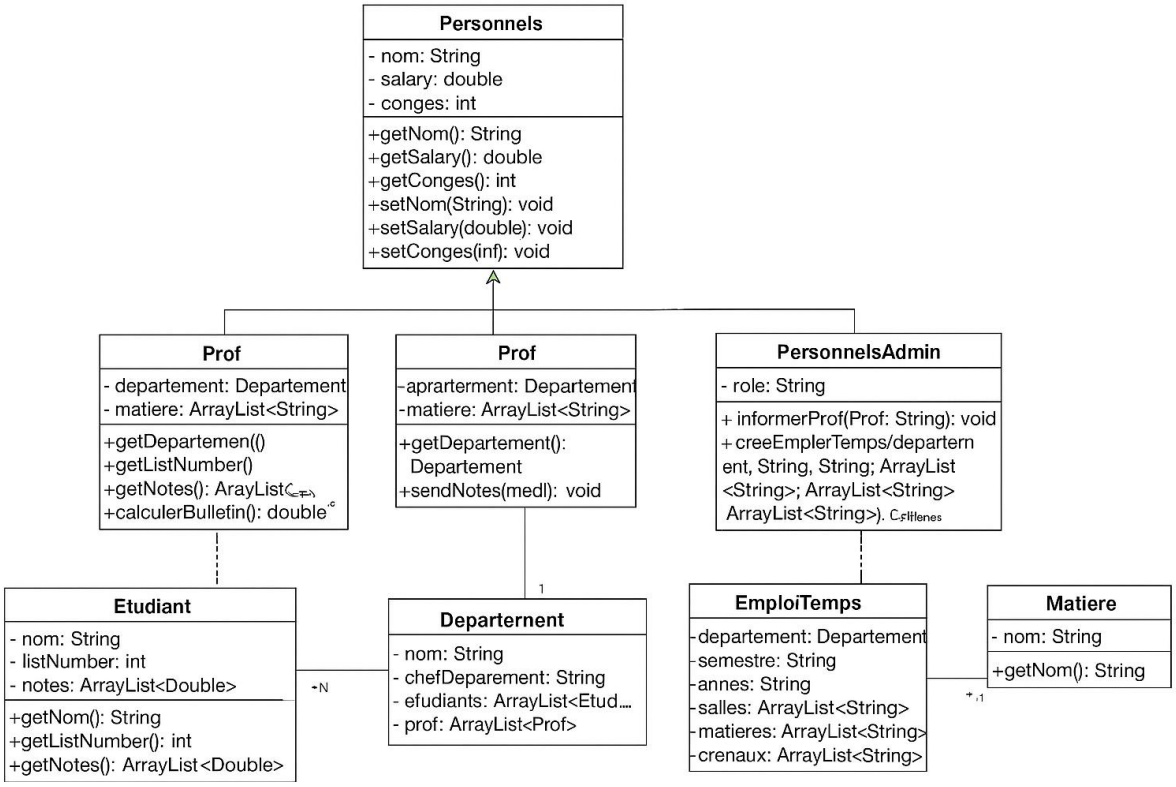
1. Présentation du projet

Le projet est une **application Java de gestion d'une université**.
Il gère :

- Les **étudiants** et leurs notes.
- Les **professeurs** et les matières qu'ils enseignent.
- Les **départements** avec leur chef, étudiants et professeurs.
- Les **personnels administratifs** (scolarité) qui créent des emplois du temps et reçoivent les notes.
- Les **emplois du temps** pour chaque département, semestre et année.

Le projet est conçu initialement en mode **console**, mais a été adapté pour une interface **graphique Swing**.

2. Diagramme UML (texte)



Relations UML

- Prof **hérite** de Personnels.
- PersonnelsAdmin **hérite** de Personnels.
- Departement **contient** Prof et Etudiant (composition / agrégation).

- EmploiTemps **associe** un Departement.
- Prof **enseigne** plusieurs Etudiant (association).
- PersonnelsAdmin **informe et reçoit notes** de Prof (association).

3. Description des classes et responsabilités

Classe	Responsabilité principale
Etudiant	Gérer les informations d'un étudiant, ses notes, calculer le bulletin
Prof	Enseigner des matières, gérer le département et envoyer les notes
PersonnelsAdmin	Créer emploi du temps, gérer les informations administratives
Departement	Gérer les étudiants et professeurs d'un département, collaborer
EmploiTemps	Représenter le planning d'un département pour un semestre
Personnels	Classe parent pour Prof et PersonnelsAdmin

4. Proposition de base de données relationnelle

On peut représenter le projet avec les tables suivantes :

Tables

1. **Departement**

```
CREATE TABLE Departement (
   id INT AUTO_INCREMENT PRIMARY KEY,
   nom VARCHAR(100) NOT NULL,
   chefDepartement VARCHAR(100)
);
```

2. **Etudiant**

```
CREATE TABLE Etudiant (
   id INT AUTO_INCREMENT PRIMARY KEY,
   nom VARCHAR(100) NOT NULL,
   listNumber INT,
   departement_id INT,
```

```
    FOREIGN KEY (departement_id) REFERENCES Departement(id)
);
```

3. **Prof**

```
CREATE TABLE Prof (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(100) NOT NULL,
    salary DOUBLE,
    departement_id INT,
    FOREIGN KEY (departement_id) REFERENCES Departement(id)
);
```

4. **Matiere**

```
CREATE TABLE Matiere (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(100) NOT NULL
);
```

5. **ProfMatiere** (relation N-N entre Prof et Matiere)

```
CREATE TABLE ProfMatiere (
    prof_id INT,
    matiere_id INT,
    PRIMARY KEY(prof_id, matiere_id),
    FOREIGN KEY (prof_id) REFERENCES Prof(id),
    FOREIGN KEY (matiere_id) REFERENCES Matiere(id)
);
```

6. **Notes** (relation N-N entre Etudiant et Matiere)

```
CREATE TABLE Notes (
    etudiant_id INT,
    matiere_id INT,
    note DOUBLE,
    PRIMARY KEY(etudiant_id, matiere_id),
    FOREIGN KEY (etudiant_id) REFERENCES Etudiant(id),
    FOREIGN KEY (matiere_id) REFERENCES Matiere(id)
```

```
);
```

7. **PersonnelsAdmin**

```
CREATE TABLE PersonnelsAdmin (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(100),
    role VARCHAR(50),
    salary DOUBLE
);
```

8. **EmploiTemps**

```
CREATE TABLE EmploiTemps (
    id INT AUTO_INCREMENT PRIMARY KEY,
    departement_id INT,
    semestre VARCHAR(10),
    annee VARCHAR(10),
    FOREIGN KEY (departement_id) REFERENCES Departement(id)
);
```

9. **EmploiTempsDetail** (pour les salles, matières et créneaux)

```
CREATE TABLE EmploiTempsDetail (
    id INT AUTO_INCREMENT PRIMARY KEY,
    emploiTemps_id INT,
    salle VARCHAR(50),
    matiere VARCHAR(100),
    crenau VARCHAR(50),
    FOREIGN KEY (emploiTemps_id) REFERENCES EmploiTemps(id)
);
```

5. Exemple de relations

- **1 Département → N Etudiants**

- **1 Département → N Professeurs**
- **1 Professeur → N Matières**
- **1 Etudiant → N Notes / Matières**
- **1 PersonnelsAdmin** → peut créer N emplois du temps
- **1 EmploiTemps** → N détails (salles/matières/créneaux)

6. Avantages et améliorations

- La structure permet d'ajouter facilement :
    - Une interface graphique Java Swing ou JavaFX.
    - Une persistance avec MySQL ou SQLite.
- L'UML texte peut être transformé en **diagramme visuel UML** avec un outil comme **draw.io**, **StarUML**, ou **Lucidchart**.
- On peut créer une interface pour afficher les tableaux d'étudiants, profs, emplois du temps, etc.