Projet de fin de module

Développement Orienté Objet

IWA 2022-2023
A. EL HIBAOUI

2023

TABLE DES MATIÈRES

1 Chapitre 1
Projet de Fin de Module – Gestion de Filières

1.1 Objectif *1*
1.2 Classes Java correspondantes *2*
1.2.1 Classe Etudiant *2*
1.2.2 Classe Enseignant *2*
1.2.3 Classe Module *3*
1.2.4 Classe Note *4*
1.2.5 Classe Département *4*
1.2.6 Classe Filière *4*
1.2.7 Classe Chef de Département *5*
1.2.8 Classe Responsable de filière *5*
1.3 Base de données MySQL *6*
1.3.1 Création de la base de données *6*
1.3.2 Création de la table Etudiant *6*
1.3.3 Création de la table Module *6*
1.3.4 Création de la table Note *6*
1.3.5 Création de la table Enseignant *7*
1.3.6 Création de la table Departement *7*
1.3.7 Création de la table Filière *7*
1.4 Interface Graphique *7*
1.5 Outils de développement *8*
1.6 Remise du projet *8*
1.6.1 Étapes à suivre *8*
1.6.2 Dates Importantes *8*

CHAPITRE

PROJET DE FIN DE MODULE – GESTION DE FILIÈRES

1

1.1 Objectif

Le but de ce projet est de mettre en pratiques tous les concepts de la programmation orientée objets vus dans les différentes parties de l'enseignement de l'élément de module "Développement Orienté Objet". Pour ce faire, les étudiants de cette formation sont amenés à concevoir et implémenter en Java une application permettant la gestion de filières. Le cahier de charges de cette application est donné comme suit :

— `'Étudiant'` : a un nom, prénom, code apogée, date de naissance. Il est inscrit dans une filière et suit les enseignements des modules dans lesquels il est inscrit pédagogiquement.

— `'Module'` : affecté à une filière et peut avoir des notes associées à des étudiants.

— `'Note'` : obtenue par un étudiant dans un module après avoir passé un examen soit dans la session normale ou session de rattrapage. Autrement dit, la note d'un étudiant est caractérisée par sa valeur, la date où elle a été obtenue ainsi que la session. Un module est validé si sa note est supérieure ou égale à 10.00, autrement, il est considéré comme non validé.

— `'Enseignant'` : a un numéro de somme, un nom, un prénom, une date de naissance, date d'embauche, un grade. Un enseignant enseigne un ou plusieurs modules dans une ou plusieurs Filières, appartient à un seul département. Il est le seul à pouvoir ajouter/supprimer/modifier les notes de ses modules.

— `'Département'` : a un nom, un enseignant élu comme chef de département et il contient une ou plusieurs filières.

— `'Filière'` : est attaché à un département et contient plusieurs modules et a un enseignant responsable qui fait partie des enseignants du département d'attache.

— `'Chef de département'` : est un enseignant qui a le doit de créer des filières et d'affecter un responsable pour chaque filière créée.

— `'Responsable de filière'` : est un enseignant qui manipule les étudiants de sa filière. Il ajoute/supprime/modifie la liste des étudiants.

Il convient de noter que ces classes sont génériques, donc vous devriez ajouter des méthodes et des champs pour répondre à vos besoins spécifiques.

L'ensemble des acteurs humains accèdent par authentification à l'application et chacun d'eux a une vue propre à son rôle.

1.2 Classes Java correspondantes

Ci-dessous le code des classes : Étudiant, Module, Note, Enseignant, Département, Filière, chef de département et responsable de filière.

1.2.1 Classe Etudiant

```
class Etudiant {
    private String codeApogee;
    private String nom;
    private String prenom;
    private Date dateNaissance;
    private Filiere filiere;
    private List<Module> modules;

    // Getters and setters
    // ...

    // Method to add a module
    public void addModule(Module module) {
        // Add the module to the list of modules
        // ...
    }

    // Method to remove a module
    public void removeModule(Module module) {
        // Remove the module from the list of modules
        // ...
    }
}


```

1.2.2 Classe Enseignant

```

class Enseignant {
    private int numeroSomme;
    private String nom;
    private String prenom;
    private Date dateNaissance;
    private Date dateEmbauche;
    private String grade;
    private List<Module> modules;
    private Departement departement;
```

```
    // Getters and setters
    // ...

    // Method to add a module
    public void addModule(Module module) {
        // Add the module to the list of modules
        // ...
    }

    // Method to remove a module
    public void removeModule(Module module) {
        // Remove the module from the list of modules
        // ...
    }

    // Method to add a note
    public void addNote(Note note) {
        // Add the note to the corresponding module
        // ...
    }

    // Method to remove a note
    public void removeNote(Note note) {
        // Remove the note from the corresponding module
        // ...
    }
}


```

1.2.3 Classe Module

```
class Module {
    private String nom;
    private Filiere filiere;
    private List<Note> notes;

    // Getters and setters
    // ...

    // Method to add a note
    public void addNote(Note note) {
        // Add the note to the list of notes
        // ...
    }

    // Method to remove a note
    public void removeNote(Note note) {
        // Remove the note from the list of notes
        // ...
    }

    // Method to check if the module is validated
    public boolean isValidated() {
        // Check if the module is validated
        // ...
    }
}
```

```

```

1.2.4 Classe Note

```
class Note {
    private float note;
    private Date dateObtention;
    private String session;
    private Etudiant etudiant;
    private Module module;

    // Getters and setters
    // ...
}


```

1.2.5 Classe Département

```
class Departement {
    private String nom;
    private Enseignant chefDepartement;
    private List<Filiere> filieres;

    // Getters and setters
    // ...

    // Method to add a filiere
    public void addFiliere(Filiere filiere) {
        // Add the filiere to the list of filieres
        // ...
    }

    // Method to remove a filiere
    public void removeFiliere(Filiere filiere) {
        // Remove the filiere from the list of filieres
        // ...
    }
}


```

1.2.6 Classe Filière

```
class Filiere {
    private String nom;
    private Departement departement;
    private List<Module> modules;
    private Enseignant responsableFiliere;
    // Getters and setters
// ...

// Method to add a module
```

```
public void addModule(Module module) {
    // Add the module to the list of modules
    // ...
}

// Method to remove a module
public void removeModule(Module module) {
    // Remove the module from the list of modules
    // ...
}

// Method to add an etudiant
public void addEtudiant(Etudiant etudiant) {
    // Add the etudiant to the filiere
    // ...
}

// Method to remove an etudiant
public void removeEtudiant(Etudiant etudiant) {
    // Remove the etudiant from the filiere
    // ...
 }
}

```

1.2.7 Classe Chef de Département

```
class ChefDepartement extends Enseignant {
// Method to create a filiere
    public Filiere createFiliere(String nom) {
    // Create a new filiere with the given name
    // ...
    }

// Method to assign a responsableFiliere for a filiere
public void assignResponsableFiliere(Filiere filiere, Enseignant enseignant) {
    // Assign the given enseignant as responsableFiliere for the given filiere
    // ...
}

}


```

1.2.8 Classe Responsable de filière

```
class ResponsableFiliere extends Enseignant {
private Filiere filiere;
// Getters and setters
// ...

}

```

Il est important de noter que ces classes sont très simplifiées, et il est possible que vous ayez besoin d'ajouter des méthodes et des champs supplémentaires pour répondre à vos besoins spéci-

fiques. Il est également possible que vous deviez ajouter des relations entre les classes pour gérer les associations entre elles de manière appropriée.

1.3 Base de données MySQL

Ci-dessous ce à quoi pourraient ressembler les tables de la base de données MySQL '$Gestion\_Filiere\_DB$' associées aux classes '`Etudiant`', '`Module`', '`Not`'e, '`Enseignant`', '`Departement`' et '`Filiere`' :

1.3.1 Création de la base de données

```sql
CREATE DATABASE Gestion_Filiere_DB;
```

Usage :

```sql
USE Gestion_Filiere_DB;

```

1.3.2 Création de la table Etudiant

```sql
CREATE TABLE Etudiant (
    codeApogee VARCHAR(255) NOT NULL PRIMARY KEY,
    nom VARCHAR(255) NOT NULL,
    prenom VARCHAR(255) NOT NULL,
    dateNaissance DATE NOT NULL,
    filiere_id INT NOT NULL,
    FOREIGN KEY (filiere_id) REFERENCES Filiere(id)
);

```

1.3.3 Création de la table Module

```sql
CREATE TABLE Module (
    id INT NOT NULL PRIMARY KEY,
    nom VARCHAR(255) NOT NULL,
    filiere_id INT NOT NULL,
    FOREIGN KEY (filiere_id) REFERENCES Filiere(id)
);

```

1.3.4 Création de la table Note

```sql
CREATE TABLE Note (
    id INT NOT NULL PRIMARY KEY,
    valeur FLOAT NOT NULL,
    dateObtention DATE NOT NULL,
    session VARCHAR(255) NOT NULL,
    etudiant_codeApogee VARCHAR(255) NOT NULL,
    module_id INT NOT NULL,
    FOREIGN KEY (etudiant_codeApogee) REFERENCES Etudiant(codeApogee),
    FOREIGN KEY (module_id) REFERENCES Module(id)
);

```

1.3.5 Création de la table Enseignant

```sql
CREATE TABLE Enseignant (
    numeroSomme INT NOT NULL PRIMARY KEY,
    nom VARCHAR(255) NOT NULL,
    prenom VARCHAR(255) NOT NULL,
    dateNaissance DATE NOT NULL,
    dateEmbauche DATE NOT NULL,
    grade VARCHAR(255) NOT NULL,
    departement_id INT NOT NULL,
    FOREIGN KEY (departement_id) REFERENCES Departement(id)
);


```

1.3.6 Création de la table Departement

```sql
CREATE TABLE Departement (
    id INT NOT NULL PRIMARY KEY,
    nom VARCHAR(255) NOT NULL,
    chefDepartement_numeroSomme INT NOT NULL,
    FOREIGN KEY (chefDepartement_numeroSomme) REFERENCES Enseignant(numeroSomme)
);


```

1.3.7 Création de la table Filière

```sql
CREATE TABLE Filiere (
    id INT NOT NULL PRIMARY KEY,
    nom VARCHAR(255) NOT NULL,
    departement_id INT NOT NULL,
    responsableFiliere_numeroSomme INT NOT NULL,
    FOREIGN KEY (departement_id) REFERENCES Departement(id),
    FOREIGN KEY (responsableFiliere_numeroSomme) REFERENCES Enseignant(numeroSomme
    )
);

```

1.4 Interface Graphique

L'application demandée devra être interfacée graphiquement. Elle ne doit être accessible que par les utilisateurs enregistrés. Il se peut que vous soyez amenés à ajouter une table `'utilisateurs'` qui servira à stocker les identifiants des utilisateurs, leurs mots de passe respectifs ainsi que leurs rôles.

L'application devra contenir des menus et des sous-menus qui facilite la navigation et la gestion de filières. Vous avez le libre choix de développer ce que vous estimez bon pour satisfaire les exigences fonctionnelles de l'application.

1.5 Outils de développement

1. Eclipse : pour le développement Java
2. Wamp : pour la gestion de la Base de données
3. Windows Builder : pour la gestion des interfaces graphiques.
4. Install4J : pour le déploiement de l'application (la rendre exécutable)

1.6 Remise du projet

1.6.1 Étapes à suivre

1. Créer un répertoire sur ton propre drive sous le nom `'NomEtudiantPrenomEtudiant'`.
2. Partager ce répertoire avec toute personne disposant de son lien.
3. Glisser votre application, votre rapport, votre code source (Java), votre base de données (.sql) dans ce répertoire.
4. Compléter le formulaire ci-dessous.
   https://docs.google.com/forms/d/18w8GL-YPkQcFWG0sNlP9C-ejbcpBmG5Kc42sCnoECEI/prefill

1.6.2 Dates Importantes

— Dernier délai pour la soumission du projet le : **05 février 2023.**

— Date de présentation du projet :**à partir du jeudi 09 février 2023.**