Rapport du projet : Gestion des Étudiants

7issi

**Introduction**

Ce projet est une application de bureau écrite en **Java** avec l’utilisation de la bibliothèque **Swing** pour l’interface graphique. L’objectif principal est de fournir une interface intuitive permettant de gérer des étudiants : ajout, suppression et affichage des informations. Le projet suit une structure modulaire, facilitant son extensibilité et sa maintenance.

### **Objectifs du projet**

1. Offrir une interface utilisateur conviviale pour la gestion des étudiants.
2. Permettre d’ajouter, de consulter et de supprimer les informations des étudiants.
3. Adopter une structure de code claire et bien organisée.
4. Faciliter la compréhension pour les débutants en Java.

### **Structure globale du projet**

#### **1. Organisation des packages**

Le projet est organisé en plusieurs packages :

* **main** : Contient la classe principale qui gère la fenêtre principale et la navigation entre les panneaux.
* **module** : Contient la classe Student, qui représente le modèle de données pour chaque étudiant.
* **uu** : Contient les panneaux Swing (HomePanel, AddStudentPanel, StudentListPanel) pour chaque fonctionnalité de l'application.

#### **2. Technologies utilisées**

* **Java** : Langage de programmation principal.
* **Swing** : Pour l’interface utilisateur graphique.
* **CardLayout** : Gestion de la navigation entre différents écrans de l’application.

### **Description détaillée des classes**

#### **1. MainFrame**

* **Package :** main
* **Rôle :** Classe principale de l’application.
* **Fonctionnalités :**
  + Gère la structure globale avec une disposition en deux parties :
    - **Menu latéral** : Permet la navigation entre les différentes fonctionnalités.
    - **Panneau principal** : Affiche dynamiquement le contenu correspondant à la fonctionnalité sélectionnée.
  + Implémente un CardLayout pour basculer entre les panneaux (Home, Ajouter étudiant, Liste des étudiants).
* **Menu de navigation :**
  + **Accueil** : Redirige vers la page d’accueil.
  + **Ajouter Étudiant** : Redirige vers le formulaire d’ajout d’étudiants.
  + **Liste des Étudiants** : Affiche la liste des étudiants et permet de les gérer.
  + **Quitter** : Ferme l’application.

#### **2. HomePanel**

* **Package :** uu
* **Rôle :** Affiche un écran d'accueil avec un message de bienvenue et un logo.
* **Particularités :**
  + Design simple avec un **message centré**.
  + Possibilité d'ajouter un logo pour améliorer l'attrait visuel.

#### **3. AddStudentPanel**

* **Package :** uu
* **Rôle :** Permet à l’utilisateur d’ajouter un étudiant.
* **Détails techniques :**
  + Champs de saisie pour :
    - Prénom
    - Nom
    - Âge
    - Email
  + Bouton **Ajouter** pour valider les informations.
  + Les informations saisies sont ajoutées à la liste globale des étudiants.
* **Gestion des erreurs :**
  + Actuellement, aucune validation n’est implémentée. Les champs sont acceptés tels quels.

#### **4. StudentListPanel**

* **Package :** uu
* **Rôle :** Affiche tous les étudiants enregistrés sous forme de tableau.
* **Fonctionnalités :**
  + **Affichage dynamique :** Chaque étudiant est affiché dans une ligne du tableau.
  + **Suppression :** Permet de supprimer un étudiant sélectionné.
  + Bouton **Supprimer** :
    - Supprime l’étudiant de la liste.
    - Met à jour automatiquement le tableau après suppression.
* **Méthodes importantes :**
  + refreshTable() : Actualise les données affichées dans le tableau.

#### **5. Student**

* **Package :** module
* **Rôle :** Modèle de données pour représenter un étudiant.
* **Attributs :**
  + nom : Nom de l’étudiant.
  + prenom : Prénom de l’étudiant.
  + age : Âge de l’étudiant.
  + niveauEtude : Niveau d’étude de l’étudiant.
* **Méthodes :**
  + Getters et setters pour accéder et modifier les attributs.

### **Fonctionnalités principales**

1. **Ajout d’un étudiant :**
   1. L’utilisateur entre les informations dans le formulaire.
   2. Les données sont ajoutées dynamiquement dans la liste des étudiants.
2. **Affichage de la liste des étudiants :**
   1. Les étudiants ajoutés sont affichés dans un tableau.
   2. Les informations affichées incluent le prénom, le nom, l’âge et le niveau d’étude.
3. **Suppression d’un étudiant :**
   1. L’utilisateur sélectionne un étudiant dans le tableau.
   2. L’étudiant est supprimé de la liste après confirmation.
4. **Navigation intuitive :**
   1. Utilisation d’un menu latéral pour passer d’une fonctionnalité à une autre.

### **Exemple de flux de l’application**

1. **Démarrage :**
   1. L’application affiche l’écran d’accueil avec un message de bienvenue.
2. **Ajout d’un étudiant :**
   1. L’utilisateur clique sur le bouton "Ajouter Étudiant".
   2. Il remplit les informations nécessaires (nom, prénom, âge, niveau d’étude).
   3. Les données sont validées et ajoutées à la liste.
3. **Affichage des étudiants :**
   1. L’utilisateur clique sur "Liste des Étudiants".
   2. La liste complète des étudiants s’affiche dans un tableau.
   3. L’utilisateur peut supprimer un étudiant en le sélectionnant et en cliquant sur "Supprimer".
4. **Quitter :**
   1. L’utilisateur peut quitter l’application en cliquant sur "Quitter".

### **Design et interface utilisateur**

* **Menu latéral :**
  + Couleur de fond : Bleu foncé.
  + Boutons blancs avec des icônes pour une meilleure reconnaissance visuelle.
* **Panneau principal :**
  + Utilise des polices modernes et des couleurs claires.
  + Design centré et espacé pour faciliter l’interaction.

### **Points forts du projet**

* **Structure modulaire :** Les fonctionnalités sont séparées en différentes classes pour une meilleure organisation.
* **Interface intuitive :** Navigation simple et boutons explicites.
* **Extensibilité :** Le projet peut facilement être étendu avec de nouvelles fonctionnalités (recherche, tri, etc.).
* **Apprentissage pratique :** Convient aux débutants souhaitant comprendre les concepts de base de Java et Swing.

### **Limitations et améliorations possibles**

1. **Validation des champs :**
   1. Ajouter une validation pour s’assurer que les champs obligatoires ne sont pas vides et que l’âge est un nombre valide.
2. **Recherche et tri :**
   1. Ajouter une barre de recherche pour trouver des étudiants facilement.
   2. Permettre le tri par nom, âge ou niveau d’étude.
3. **Persistance des données :**
   1. Ajouter une base de données ou permettre l’exportation/importation des données via un fichier CSV.
4. **Interface graphique améliorée :**
   1. Moderniser l’apparence avec une bibliothèque graphique comme JavaFX.

### **Conclusion**

Ce projet constitue une base solide pour une application de gestion des étudiants. Avec une structure claire et une interface conviviale, il est adapté à une utilisation pédagogique. Les futures améliorations pourraient inclure une meilleure validation, une persistance des données, et des fonctionnalités supplémentaires comme la recherche et le tri.

Fait par : Khadija Nachid Idrissi