Prototype Commenté de l'Application JavaFX

1. Présentation Générale

Cette application JavaFX permet à un utilisateur de saisir un matricule et de valider les informations d'un étudiant à partir d'une base de données. Elle fournit des retours visuels sur le statut de l'étudiant ainsi que des détails supplémentaires.

2. Structure de l'Application

2.1 Imports

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
```

- **Imports nécessaires** : Comprend des classes pour la gestion des bases de données, la création d'interfaces utilisateur et la mise en page.

2.2 Attributs de la classe

```
private Label labelMatricule;
private static TextField inputMatricule;
private Button buttonValider;
```

- **Éléments de l'interface** : Inclut des labels, des boutons et un champ de texte pour la saisie de l'utilisateur.

3. Méthode `start()`

```
@Override
public void start(Stage primaryStage) throws Exception {
    VBox root = new VBox(10);
    // Configuration des éléments UI
    labelMatricule = new Label("Veuillez entrer votre Matricule");
    inputMatricule = new TextField();
    buttonValider = new Button("VALIDER");
    buttonVider = new Button("");
    buttonDetails = new Button("DETAILS");
```

- **Mise en page** : Utilise un `VBox` pour empiler les éléments verticalement et un `HBox` pour organiser les boutons horizontalement.

4. Gestion des Événements

```
buttonVider.setOnAction(e -> {
    inputMatricule.setText("");
```

```
    labelDetails.setText("");
    labelDecision.setText("");
});

buttonValider.setOnAction(e -> {
    verificationResultat();
    labelDetails.setText("");
});

buttonDetails.setOnAction(e -> {
    detailsResultat();
});
```

- **Interactions utilisateur** : Chaque bouton déclenche une action :
    - **Vider** : Réinitialise les champs.
    - **Valider** : Vérifie le matricule dans la base de données.
    - **Détails** : Affiche des informations supplémentaires sur l'étudiant.

5. Vérification des Résultats

```
Cette partie est une methode verifiant le resultat d’un etudiant,et pour cela
nous avons besoin davoir accès a la base de données
```

- **Accès à la base de données** : Vérifie si le matricule existe et détermine si l'étudiant a réussi ou échoué en fonction de sa moyenne.

6. Détails de l'Étudiant

```
public void detailsResultat() {
    // Même logique de connexion et de requête
    if (resultSet.next()) {
        // Affichage des détails de l'étudiant
    } else {
        labelDecision.setText("Veuillez entrer un matricule correct!!!");
    }
}
```

- **Affichage des détails** : Récupère et affiche les informations de l'étudiant si le matricule est valide.

- **Style et apparence** : Charge une feuille de style pour personnaliser l'interface.

Conclusion

Cette application permet à un utilisateur d'interagir avec une base de données d'étudiants de manière intuitive. En fournissant des retours visuels sur les actions, elle améliore l'expérience utilisateur et facilite la gestion des informations étudiantes.

NB : Matricule valide= 0075847 ,0075848