Mini-Projet Java : Gestion d'une Plateforme de Formation

HAJER DAMMAK

hajer.dammak@fst.utm.tn

Énoncé du Mini-Projet Java

Objectif :
L'objectif de ce projet est de mettre en pratique des concepts fondamentaux de la programmation orientée objet en Java, notamment l'encapsulation, les constructeurs, la gestion des exceptions, l'héritage, et l'accès à une base de données.

Contexte :
L'application permettra aux utilisateurs de s'inscrire à des formations proposées sur la plateforme. Il y aura des utilisateurs de types différents : **formateurs** et **étudiants**. Chaque formation a un titre, une description, un formateur, et un prix. Les étudiants peuvent s'inscrire à des formations et consulter les formations auxquelles ils sont inscrits.

---

1. Modèle de données

Vous devez créer les classes suivantes pour représenter les entités du système :

- **Utilisateur** : Classe de base représentant un utilisateur de la plateforme.
  - Attributs :
    - `nom` (String) : Le nom de l'utilisateur.
    - `email` (String) : L'email de l'utilisateur, unique.
    - `motDePasse` (String) : Le mot de passe de l'utilisateur.
  - Méthodes :
    - Getters et setters pour chaque attribut (encapsulation).
    - Un constructeur permettant d'initialiser un utilisateur avec ses informations.
- **Formateur** : Classe dérivée d'`Utilisateur` représentant un formateur. Un formateur a un ou plusieurs cours à proposer.
  - Attributs :
    - `formations` (Liste de formations) : Liste des formations que le formateur propose.
  - Méthodes :
    - `ajouterFormation(Formation formation)` : Ajoute une formation à la liste des formations du formateur.
- **Étudiant** : Classe dérivée d'`Utilisateur` représentant un étudiant. Un étudiant peut s'inscrire à des formations.

- Attributs :
  - `inscriptions` (Liste de formations) : Liste des formations auxquelles l'étudiant est inscrit.

- Méthodes :
  - `sinscrireFormation(Formation formation)` : Inscrit l'étudiant à une formation.

- **Formation** : Représente une formation disponible sur la plateforme.
  - Attributs :
    - `titre` (String) : Le titre de la formation.
    - `description` (String) : Une description détaillée de la formation.
    - `formateur` (Formateur) : Le formateur qui dispense la formation.
    - `prix` (double) : Le prix de la formation.

  - Méthodes :
    - Getters et setters pour chaque attribut (encapsulation).

2. Gestion des exceptions

Implémentez la gestion des exceptions dans le projet :

- Si un étudiant essaie de s'inscrire à une formation qu'il a déjà suivie, une exception personnalisée (`FormationDejaInscriteException`) doit être lancée.
- Si un utilisateur essaie de se connecter avec des informations incorrectes (email/mot de passe), une exception personnalisée (`UtilisateurNonTrouveException`) doit être lancée.

Les exceptions doivent être correctement gérées avec des blocs `try-catch` pour éviter les erreurs imprévues et garantir une bonne expérience utilisateur.

3. Héritage

L'héritage sera utilisé pour les classes `Formateur` et `Étudiant`, qui héritent de la classe `Utilisateur`. Ce modèle permet de partager des fonctionnalités communes (comme la gestion des informations de base des utilisateurs) tout en ajoutant des comportements spécifiques à chaque type d'utilisateur.

- La classe `Formateur` pourra gérer les formations qu'il propose.
- La classe `Étudiant` pourra gérer ses inscriptions aux formations.

4. Accès à la base de données

L'application devra permettre d'enregistrer et de récupérer les informations des utilisateurs, des formations et des inscriptions dans une base de données. Vous devez utiliser JDBC pour :

- **Connexion à la base de données** : Créez une connexion à une base de données SQLite ou MySQL.
- **Gestion des utilisateurs** : Créez une table pour les utilisateurs où seront stockées les informations personnelles (`nom`, `email`, `mot de passe`). Un utilisateur peut être soit un formateur, soit un étudiant.
- **Gestion des formations** : Créez une table pour les formations et une table de liaison pour gérer les inscriptions des étudiants. Cette table de liaison contiendra l'ID de l'étudiant et l'ID de la formation.
- **Inscriptions** : Les étudiants pourront s'inscrire à des formations, et cette action devra être enregistrée dans la base de données.

5. Exigences supplémentaires :

- Utilisez des constructeurs pour initialiser les objets.
- Implémentez une gestion complète des exceptions pour les actions courantes (inscriptions, connexion, etc.).
- Implémentez l'héritage pour distinguer les différents types d'utilisateurs.
- Utilisez JDBC pour l'accès à la base de données, avec des requêtes pour insérer, récupérer et supprimer des données.

Livrables :

- Le code source de toutes les classes Java.
- Un fichier SQL pour créer la base de données et les tables nécessaires.

Critères d'évaluation :

- Bonne utilisation de l'encapsulation, des constructeurs, et des exceptions.
- Implémentation correcte de l'héritage pour les utilisateurs.
- Accessibilité de la base de données via JDBC et manipulation correcte des données.
- Code propre, bien commenté et bien structuré.