RAPPORT DE PROJET:
POO-Base de Données

Nour Chachia GL2/3
Med Aziz Derbel GL2/3

Année Scolaire: 2025 / 2026

Sujet: Mini-Projet en POO-Base de données
Système de gestion d'une Pharmacie

Le diagramme de classe (UML)

Description des classes:

Classes de Base (Acteurs et Produits)

Ces classes représentent les entités principales du système.

- `Utilisateur` : Représente les employés utilisant le logiciel. Elle contient les identifiants de connexion (`username`, `mdp`) et définit les permissions via l'attribut `role`. La méthode `isAdmin()` permet de vérifier si l'utilisateur possède des droits d'administrateur.
- `Client` : Stocke les informations relatives aux clients, notamment leur nom complet (`nomPrenom`) et leur adresse électronique.
- `Fournisseur` : Contient les informations des partenaires commerciaux auprès desquels les produits sont commandés, incluant leur nom et leurs coordonnées de contact.
- `Produit` : Gère les articles en stock. Elle suit le `prix`, la `quantite` disponible et un `seuilMinimal`. La méthode `stockCritique()` est essentielle car elle retourne un booléen indiquant si le stock est passé sous le seuil d'alerte.

Classes de Mouvement (Ventes et Commandes)

Ces classes enregistrent les transactions financières et logistiques.

- `Vente` : Enregistre une transaction effectuée par un `Utilisateur` pour un `Client`. Elle conserve la trace de la `dateVente` et du `montantVente` total.
- `CommandeFournisseur` : Représente un réapprovisionnement auprès d'un fournisseur. Elle inclut un `statut` (ex: en attente, reçu, annulé) permettant de suivre l'évolution de la commande.

Classes de Liaison (Détails des transactions)

Ces classes servent de tables de jointure pour lier les produits aux transactions (relation "plusieurs à plusieurs").

- `Correspond` : Fait le lien entre une Vente et les Produits vendus. Elle précise la `quantite` de chaque produit spécifique dans une vente et le `montant` correspondant.
- `Concerne` : Fait le lien entre une CommandeFournisseur et les Produits commandés. Elle indique la `quantite` commandée et le `montant` facturé par le fournisseur pour ce produit.

🚀 Les principales fonctionnalités implémentées

Le projet couvre l'ensemble du cycle de vie des produits et des transactions au sein d'une officine.

1. Gestion des Stocks et Produits (CRUD)

- **Contrôle total :** Ajout, modification, suppression et recherche de produits.
- **Alertes de seuil :** Surveillance automatique des quantités pour éviter les ruptures de stock.

2. Système de Vente Intelligent

- **Panier dynamique :** Ajout multi-produits avec calcul automatique du total en temps réel.
- **Gestion Client :** Identification par email, création rapide de nouveaux clients via une fenêtre modale (`JDialog`) sans interrompre la vente.
- **Sécurité transactionnelle :** Déstockage automatique lors de la validation. Si le stock est insuffisant, la vente est bloquée.

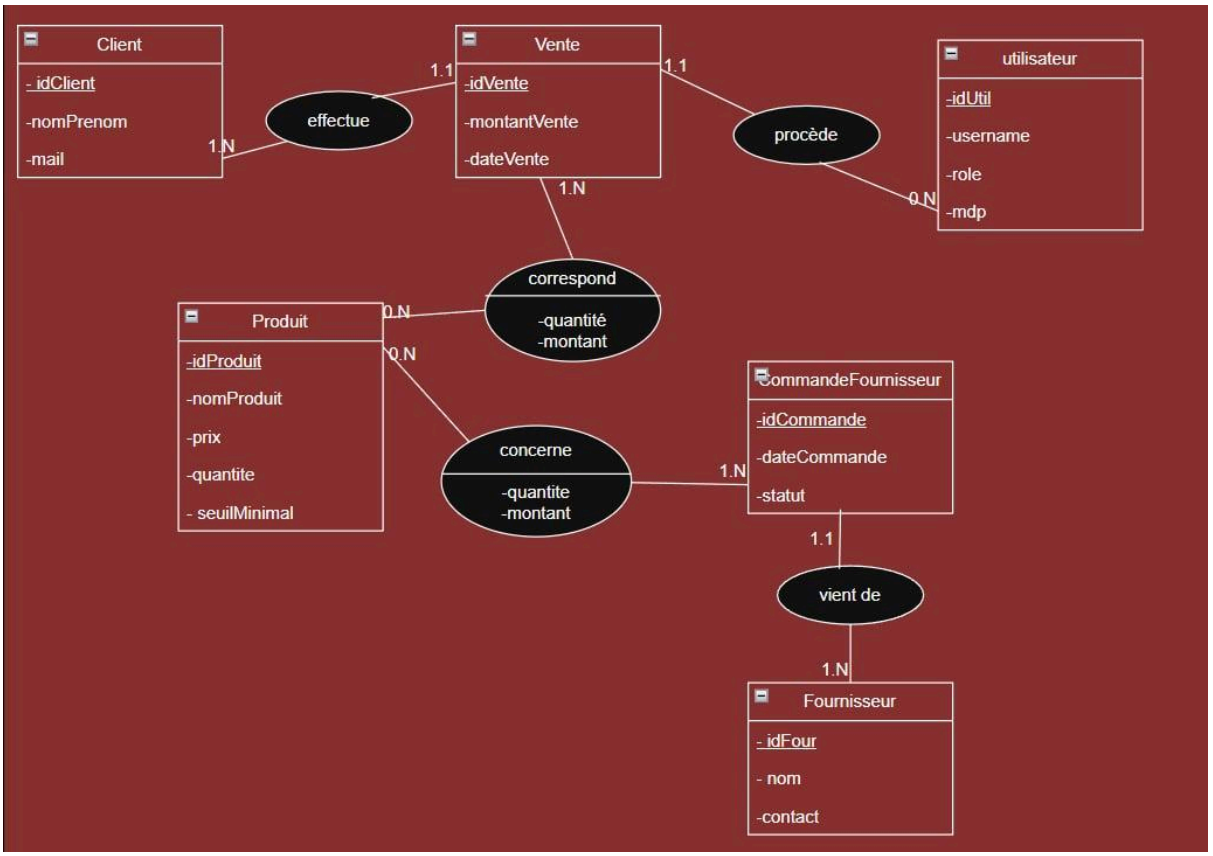
3. Gestion des Commandes Fournisseurs

- **Cycle d'achat :** Création de bons de commande en statut "en attente".
- **Réception automatisée :** Une fois la commande marquée comme "reçue", le système incrémente automatiquement le stock des produits concernés via une transaction SQL sécurisée.

4. Rapports et Analyse (Business Intelligence)

- **Tableau de bord :** Visualisation du chiffre d'affaires quotidien.
- **Suivi Client :** Historique complet des achats par client pour un suivi médical et commercial précis.
- **Performance :** Analyse des dépenses par fournisseur.

Le Modèle Conceptuel des Données



Le Modèle Logique des Données (MLD)

utilisateur (idUtil, username, mdp, role)

client (idClient, nomPrenom, mail)

fournisseur (idFour, nom, contact)

produit (idProduit, nomProduit, prix, quantite ,seuilMinimal)

commandeFournisseur (idCommande, dateCommande, statut, #idFour)

vente (idVente, montantVente, dateVente, #idUtil, #idClient)

correspond (#idVente,#idProduit, quantite, montant)

concerne (#idCommande,#idProduit, quantite, montant)

🛠️ Les difficultés rencontrées

Le développement d'une application de gestion robuste a présenté plusieurs défis techniques :

1. Gestion de la Portée des Variables (Scope)

- **Défi :** Accéder aux données d'un produit (`Produit p`) en dehors d'un bloc `try-catch` de recherche pour effectuer les calculs de prix.
- **Solution :** Initialisation des variables à `null` avant les blocs d'exception et vérification de non-nullité avant traitement.

2. Intégrité des Données et Transactions SQL

- **Défi :** Garantir qu'une vente ne soit pas enregistrée si le stock ne peut pas être mis à jour (risque d'incohérence).
- **Solution :** Utilisation de `conn.setAutoCommit(false)` et `conn.commit()` pour assurer que toutes les opérations (insertion vente + mise à jour stock) réussissent ensemble ou échouent ensemble (`rollback`).

3. Interface Utilisateur (UX) Dynamique

- **Défi :** Rafraîchir les données d'un panneau (ex: Historique) suite à une action dans un autre panneau (ex: Sélection client) sans redémarrer l'application.
- **Solution :** Mise en place de méthodes d'actualisation (`chargerDonnees`) liées à des écouteurs d'événements (`ListSelectionListener`).