TD et TP : Gestion des Exceptions en Java

Alla LO

Travaux Dirigés (TD)

Exercice 1 : Calculatrice avec Gestion Avancée des Erreurs

Créez une calculatrice simple en Java qui permet les opérations de base : addition, soustraction, multiplication et division.
Gérez les exceptions suivantes :

- Division par zéro (`ArithmeticException`).
- Entrée invalide de l'utilisateur pour les nombres (`NumberFormatException`).
- Opération inconnue (lève une exception personnalisée `OperationInvalideException` si l'utilisateur saisit une opération non reconnue).

Pour chaque erreur, affichez un message expliquant l'erreur et demandez à l'utilisateur de saisir à nouveau les valeurs.

Exercice 2 : Système de Gestion de Réservations avec Exceptions Personnalisées

Créez une classe `Reservation` avec des attributs pour le nom du client, la date et le nombre de places demandées.
Créez une exception `NombreDePlacesInsuffisantException` qui est levée lorsque le nombre de places demandées dépasse le nombre de places disponibles.
Dans la classe principale, implémentez un système qui permet aux utilisateurs de réserver des places pour un événement, en vérifiant à chaque réservation que les places restantes suffisent.

Exercice 3 : Validateur de Mot de Passe avec Critères

Créez une classe `PasswordValidator` qui vérifie la sécurité d'un mot de passe entré par l'utilisateur.
Le mot de passe doit respecter les critères suivants :

- Longueur minimale de 8 caractères.
- Contenir au moins un chiffre.
- Contenir au moins une lettre majuscule et une lettre minuscule.

Créez des exceptions personnalisées (`MotDePasseTropCourtException`, `MotDePasseSansChiffreException`, etc.) pour chaque critère non respecté, et affichez un message explicatif pour chaque erreur.

Exercice 4 : Système de Panier d'Achat avec Gestion des Stocks

Créez une classe `Produit` avec des attributs pour le nom du produit, le prix et la quantité en stock.
Créez une classe `Panier` qui permet d'ajouter des produits et de calculer le total. Si un utilisateur tente d'ajouter une quantité supérieure à la quantité en stock, le programme lève une `QuantiteIndisponibleException`.

Exercice 5 : Analyseur de Fichier CSV

Créez une classe `CSVReader` qui lit un fichier CSV ligne par ligne.
Si le fichier n'existe pas, le programme lève une `FileNotFoundException`.
Si une ligne du fichier comporte un nombre incorrect de colonnes, le programme lève une exception personnalisée `CSVFormatException` avec le numéro de la ligne en question.

Exercice 6 : Gestionnaire d'Opérations Bancaires

Créez une classe `CompteBancaire` avec des méthodes pour `deposer`, `retirer`, et `consulterSolde`.
Le retrait doit lever une `FondsInsuffisantsException` si le solde est insuffisant.
Ajoutez une méthode `virement` qui transfère un montant entre deux comptes bancaires.

Exercice 7 : Système de Connexion avec Blocage après Échecs Successifs

Créez une classe `Utilisateur` avec un nom d'utilisateur et un mot de passe.
Implémentez une méthode `connexion` qui vérifie les informations saisies et lève une `AuthentificationEchoueeException` en cas d'échec.
Si l'utilisateur échoue trois fois consécutivement, la méthode lève une exception `CompteBloqueException`.

Exercice 8 : Simulateur de Stock d'Actions avec Risque d'Exception

Créez une classe `Portefeuille` qui contient une liste d'actions.
Chaque action possède un prix d'achat, un prix de vente et une quantité.
Ajoutez une méthode `vendreAction` qui vend des actions au prix actuel, lève une `QuantiteInsuffisanteException` si la quantité est insuffisante, et lève une `PrixInvalideException` si le prix de vente est inférieur au prix d'achat.

Travaux Pratiques (TP)

Exercice 1 : Simulation de Transactions de Cryptomonnaies

Créez une classe `Wallet` qui simule un portefeuille de cryptomonnaies, avec des méthodes pour `acheter` et `vendre` des cryptomonnaies.
Chaque transaction est soumise à des frais et des règles spécifiques :

- `FraisInsuffisantsException` : levée si les frais sont inférieurs à un seuil requis.
- `CryptoIndisponibleException` : levée si la cryptomonnaie demandée n'est pas disponible dans le portefeuille.

Implémentez un historique des transactions réussies et échouées.

Exercice 2 : Gestion de la Chaîne Logistique d'une Entreprise

Créez des classes pour représenter les étapes de traitement d'une commande (Réception, Préparation, Livraison).
Chaque étape peut générer des exceptions :

- `ProduitIndisponibleException` si un produit manque.
- `LivraisonImpossibleException` si l'adresse de livraison est incorrecte.

Créez une méthode `traiterCommande` qui parcourt les étapes. Si une exception survient, la commande est mise en attente de résolution.

Exercice 3 : Analyseur de Logs avec Regroupement d'Exceptions

Créez un programme `LogAnalyzer` qui lit un fichier de logs et extrait les informations d'erreurs pour chaque ligne.
Si une ligne est mal formatée, le programme lève une `LogFormatException`.
Les erreurs sont regroupées par type dans des listes distinctes pour générer un rapport global :

- `ErreurCritiqueException` : erreurs critiques à adresser immédiatement.
- `AvertissementException` : avertissements à surveiller.

Affichez un récapitulatif des erreurs critiques et des avertissements rencontrés, avec leur fréquence.