TP : Système Bancaire en Java

2ème année BTS DSI - POO (JAVA)

Objectif général

Créer un programme Java pour la gestion d'une banque, en utilisant les concepts de programmation orientée objet, comme les classes, les objets, les méthodes, les constructeurs, les getters, setters, et l'héritage.

Exercice 1 : Gestion des Comptes Bancaires

Question 1

Écrire une classe `CompteBancaire` qui modélise un compte bancaire.

Attributs :

- `numeroCompte` : un numéro unique de compte (type `String`).
- `solde` : le solde du compte (type `double`).
- `titulaire` : le nom du titulaire du compte (type `String`).

Constructeurs :

- Un constructeur qui initialise le numéro de compte et le nom du titulaire, avec un solde initial de 0.

Méthodes :

- `deposer(double montant)` : permet de déposer de l'argent sur le compte.
- `retirer(double montant)` : permet de retirer de l'argent, en vérifiant que le solde est suffisant.
- `afficherSolde()` : affiche le solde actuel du compte.

Getters et Setters :

- Créez les getters pour le numéro de compte et le solde.
- Créez le setter pour modifier le nom du titulaire.

Question 2

Écrire une classe `Banque` qui permet de gérer plusieurs comptes bancaires.

Attributs :

- `comptes` : un tableau d'objets `CompteBancaire`.
- `nombreComptes` : Cet attribut doit être partagé entre toutes les instances de la classe.

Méthodes :

- `ajouterCompte(CompteBancaire compte)` : permet d'ajouter un nouveau compte à la banque.
- `chercherCompte(String numeroCompte)` : permet de rechercher un compte par son numéro.
- `transfert(String numeroSource, String numeroDest, double montant)` : permet de transférer de l'argent d'un compte à un autre.

Méthode statique :

- `afficherNombreComptes()` : affiche le nombre total de comptes bancaires (utilisez un attribut que vous avez défini pour le nombre de comptes).

Question 3

Dans une classe de test `BanqueTest` :

- Créez une banque.
- Ajoutez trois comptes bancaires avec des numéros et des titulaires différents.
- Effectuez des dépôts, des retraits, et des transferts entre les comptes.
- Affichez le solde des comptes après les transactions.
- Utilisez la méthode `afficherNombreComptes()` pour afficher le nombre total de comptes créés.

Exercice 2 : Comptes Courants et Comptes Épargne (Héritage)

Question 4

Créez une classe `CompteCourant` qui hérite de la classe `CompteBancaire`.

Attribut supplémentaire :

- `decouvertAutorise` : le montant maximum du découvert autorisé.

Méthode :

- Surchargez la méthode `retirer(double montant)` pour permettre un découvert autorisé.

Question 5

Créez une classe `CompteEpargne` qui hérite de la classe `CompteBancaire`.

Attribut supplémentaire :

- `tauxInteret` : le taux d'intérêt pour ce type de compte.

Méthode :

- `calculerInteret()` : permet de calculer les intérêts et de les ajouter au solde du compte.

Question 6

Dans `BanqueTest` :

- Créez des comptes courants et des comptes épargne.
- Effectuez des opérations (dépôts, retraits).
- Calculez et appliquez les intérêts pour les comptes épargne.
- Affichez les informations des comptes après chaque opération.

Exercice 3 : Gestion des Transactions

Question 7

Créez une classe `Transaction` pour enregistrer l'historique des transactions de chaque compte bancaire.

Attributs :

- `type` : crédit ou débit (type `String`).
- `montant` : montant de la transaction (type `double`).
- `date` : date de la transaction (type `Date`).

Question 8

Modifiez la classe `CompteBancaire` pour enregistrer chaque dépôt, retrait, ou transfert dans une liste de transactions.

Question 9

Dans `BanqueTest` :

- Effectuez plusieurs transactions sur différents comptes.
- Affichez l'historique des transactions pour chaque compte.