TP6 - Héritage

Le but de ce TP est de comprendre à travers d'un exemple simplifié de comptes bancaires, les concepts d'encapsulation, de propriétés d'instances et de classes, d'héritage, de surcharge et de redéfinition de méthodes, de polymorphisme et de liaison dynamique.

Dans un premier temps, nous nous intéressons à un compte simple, caractérisé par un solde exprimé en euros, positifs ou négatifs, et son titulaire. Il est possible de créditer ce compte ou de le débiter d'un certain montant.

1. Spécifier la classe SimpleAccount. Sur le diagramme de classe, on fera apparaître aussi la classe Person. Une personne est simplement définie par son nom, son prénom et son sexe qui sont les informations nécessaires pour pouvoir la construire.
2. Proposer un programme de test de la classe SimpleAccount.
3. Écrire la classe SimpleAccount.

En fait, une banque conserve, pour chaque compte, l'historique des opérations de crédit et de débit qui le concernent. On souhaite modéliser un tel compte qu'on appelle compte courant. En plus des méthodes d'un compte simple, un compte courant offre des méthodes pour afficher l'ensemble des opérations effectuées (getAccountStatement) ou seulement les opérations crédit (getCreditStatement) ou de débit (getDebitStatement).

Pour représenter l'historique, on utilisera la classe History. Cette classe gère un historique chronologique, indexé par des entiers, des montants d'opérations enregistrés. Concrètement, on doit pouvoir enregistrer un montant d'opération (si le montant est positif, cela correspond à un crédit, et s'il est négatif, il correspond à un débit), récupérer la valeur d'une opération désignée par son index dans l'historique (1 est la plus ancienne), et connaitre le nombre d'entrées dans l'historique.

1. Améliorer le diagramme de classe en faisant apparaître les deux classes.
2. Écrire la classe History.
3. Écrire la classe CheckingAccount.
4. Proposer un programme de test de la classe CheckingAccount.

Une banque est bien entendu un organisme qui gère un grand nombre de comptes, qu'ils soient simples ou courants.

1. En supposant que tous les comptes sont rangés dans un unique tableau, indiquer quels sont les attributs de la classe Bank. Donner ses constructeurs.
2. Améliorer le diagramme de classe en y faisant apparaître la banque.

3. Définir sur la classe Bank une méthode pour ouvrir un compte simple et une méthode pour ouvrir un compte courant.
4. Définir plusieurs méthodes sur la banque. Une qui donne le cumul des soldes disponibles sur chacun des comptes, une qui donne le nombre de comptes débiteurs, une qui donne la somme des débits.
5. On considère que la banque prélève périodiquement des frais de tenue de compte. Écrire une méthode qui débite sur tous les comptes la somme de 2 euros.
6. Écrire une opération qui édite les relevés de tous les comptes courants.

Pour les différencier, chaque compte possède un numéro unique. On suppose que les numéros sont des entiers et qu'ils sont attribués par ordre croissant en commençant à 10001. Dans la suite, on envisage deux solutions pour attribuer les numéros de compte.

1. On souhaite que l'attribution du numéro de compte soit de la responsabilité des classe SimpleAccount et CheckingAccount. Indiquer les modifications à apporter à ces classes.
2. On suppose maintenant que c'est la banque qui gère et attribue les numéros de compte. Indiquer les modifications à apporter aux classes SimpleAccount et CheckingAccount.

Un livret A est un produit bancaire dont le titulaire peut faire des opérations de dépôt dans la limite d'un plafond et des retraints dont le montant ne peut pas dépasser le solde du livret. Les sommes versées sur un Livret A donnent lieu à rémunération. Le taux d'intérêt ainsi que le plafond du Livret A sont identiques pour tous les Livrets A. On supposera que le taux d'intérêt est fixé à 1% et le plafond à 22.950 €.

Tout comme pour les comptes courants, un historique des opérations est géré par la banque. Il est ainsi possible d'accéder aux dernières opérations de crédit ou de débit et d'éditer un relevé.

1. Compléter le diagramme de classe UML du système pour faire apparaître les livrets A.
2. Écrire la classe SavingsAccount.
3. Proposer un programme de test de la classe SavingsAccount.