Mots-clés : *Passage de paramètre, abstraction, héritage, polymorphisme, exceptions, javadoc*

$\Longrightarrow$ ***Les mots ont un sens : prendre le temps de lire attentivement le sujet…*** $\Longleftarrow$

1 Présentation générale, et consignes préliminaires

L'objet de cet exercice est de simuler le fonctionnement d'opérations bancaires inter-comptes de base. Bien sûr, ce fonctionnement est ultra-simplifié.

L'idée directrice de ce sujet est de constituer des *spécifications fonctionnelles*, qui auraient été rédigées par un expert métier. Le fonctionnement et le comportement attendus pour une banque sont donc décrits sous la forme de *règles de gestion*, que le programme final devra implanter.

Ce sujet comporte deux aspects importants :

— un aspect *conception*, avec la modélisation d'une situation (pseudo-)réelle décrite en langage naturel, et

— un aspect *programmation*, avec le développement de ce modèle, en faisant appel à des notions avancées comme les classes abstraites, ou les exceptions.

Pour guider à la fois la conception et le développement, nous allons procéder par *incréments* fonctionnels successifs, *en testant au fur et à mesure le code correspondant*. Chaque incrément correspond à un ensemble de règles de gestions, un ensemble de fonctionnalités, et un jeu de tests. Noter que dans la mesure du possible, à chaque nouvel incrément le code développé précédement est augmenté, mais n'est ni modifié, ni supprimé.

Question 1.1 (Optionnel) Construire la javadoc sur l'ensemble du sujet, partout où c'est possible.

2 Banque, comptes, clients, et attachés de clientèle

2.a Règles de gestion

1. Une banque gère des comptes bancaires, des clients, et des attachés de clientèles.
2. Un compte bancaire est nécessairement soit un compte courant, soit un compte carte de crédit, soit un compte rémunéré.
3. Un compte rémunéré est nécessairement soit un compte épargne, soit un Plan Épargne bloqué.
4. Tout compte possède un numéro identifiant et un libellé.
5. Tout compte appartient à un client, qui en est le titulaire.
6. Tout compte possède un solde.
7. Tout compte rémunéré possède un taux de rémunération, et un solde plafond.
8. Un Plan Épargne bloqué possède une date de mise à disposition des fonds, avant laquelle aucune opération n'est autorisée.
9. Un client est identifié par un numéro client ; il a également un nom.
10. Un client peut être titulaire de plusieurs comptes.
11. Un client est suivi par un attaché de clientèle.
12. Un attaché de clientèle suit plusieurs clients.

2.b Fonctionnalités

La Banque La classe `Banque` contiendra le programme principal permettant de tester, au fur et à mesure, les différentes structures de données et fonctionnalités associées.

Question 2.1 Créer la classe *exécutable* `Banque`.

Les comptes Tout compte offre les fonctionnalités suivantes (qui seront détaillées dans l'incrément suivant) :

— débiter ce compte d'un montant donné,

— créditer ce compte d'un montant donné,

— consultation du solde.

Question 2.2 Proposer un premier modèle de classes pour les règles ci-dessus.

2.c Jeu de tests

Dans le programme principal, tester les cas suivants :

— la création d'attachés de clientèle (au moins 2),

— la création de clients (au moins 2, dont un qui servira de référence pour la suite, sous le nom $\mathcal{C}$),

— la création de comptes (au moins un client ($\mathcal{C}$) doit posséder à la fois un compte courant, et un compte épargne au plafond de 700 €).

3 Opérations sur compte

3.a Règles de gestion

13. Une opération sur compte est nécessairement soit un *débit*, soit un *crédit* ; on parle de la *nature* d'un opération.
14. Chaque opération, ou tentative d'opération sur un compte (débit ou crédit) fait l'objet d'un enregistrement spécifique.
15. L'historique des opérations associées à tout compte est conservé sans limite de durée.
16. Une opération, pour un compte et un client donnés, est identifiée par un numéro, et possède éventuellement un libellé.
17. Une opération a un *statut*, qui peut prend l'une des valeurs suivantes : `OK` (*succès*), `KO` (*échec*, pour opération rejetée), `ATTENTE` (en attente de validation).
18. Une opération a une date de prise d'effet.
19. Un débit ou un crédit ne sont possibles que pour un montant à débiter (respectivement à créditer) strictement positif ; une tentative de débit ou crédit d'un montant négatif ou nul doit être rejeté, et doit lever une alerte (sous la forme d'une exception `OperationBancaireException`) avec un message approprié.
20. Un débit n'est possible que si le solde du compte concerné est suffisant ; une tentative de débit sur un compte inssufisament approvisionné doit être rejeté, et doit lever alerte (sous la forme d'une exception `OperationBancaireException`) avec un message approprié.
21. Dans le cas d'un compte auquel s'applique un solde plafond, toute tentative de crédit pouvant porter le solde au-delà du plafond doit être rejeté, et une alerte levée avec un message approprié.

3.b Fonctionnalités

Débiter Tout compte doit pouvoir être débité selon les modalités suivantes (en plus des règles de gestion ci-dessus) :

— si le débit est possible alors le solde est modifié directement, et une opération de succès est créée ;

— si le débit n'est pas possible, alors le solde reste inchangé, une alerte est levée, et une opération d'échec est créée.

Créditer Tout compte doit pouvoir être crédité selon les modalités suivantes (en plus des règles de gestion ci-dessus) :

— si le crédit est possible alors le solde est modifié directement, et une opération de succès est créée ;

— si le crédit n'est pas possible, alors le solde reste inchangé, et une opération d'échec est créée.

Question 3.1 Compléter le modèle de classes pour inclure la gestion des opérations sur compte. Penser à la gestion de la nature d'une opération d'une part, et du statut d'une opération d'autre part.

Question 3.2 Inclure, dans le modèle de classes, les signatures des méthodes permettant d'implanter les fonctionnalités liées au débit et au crédit d'un compte.

Question 3.3 Implémenter l'ensemble des règles de gestion ci-dessus.

3.c Jeu de tests

Tester les cas suivants :

— Crédit initial OK de 2000 € sur le compte courant de $\mathcal{C}$(vérifier le solde, la création d'une opération, et l'ajout de l'opération dans l'historique concerné).

— Crédit initial OK de 33 € sur le compte épargne de $\mathcal{C}$.

— Débit OK de 500 € sur le compte courant de $\mathcal{C}$.

— Tentative de débit rejetée de 2500 € sur le compte courant de $\mathcal{C}$(penser à vérifier l'enregistrement de l'opération dans l'historique).

4 Ordres de virement

On introduit dans le modèle la possibilité pour un client de passer un *ordre de virement* de compte à compte. Un client pourra ainsi, par exemple, ordonner le virement d'une somme depuis son compte courant vers son compte épargne.

4.a Règles de gestion

22. Un ordre de virement (OV) est passé par un donneur d'ordre (client).
23. Un OV concerne un compte d'origine, un compte destinataire, et un montant.
24. Les comptes d'origine et destinataire d'un OV doivent nécessairement appartenir au donneur d'ordre concerné. Toute tentative d'OV sur des comptes non-autorisés doit lever une alerte spécifique.
25. Tout OV *possible* donne lieu à deux opérations : une opération de débit sur le compte origine, et une opération de crédit sur le compte destination.
26. Un OV est possible si et seulement si les deux opérations qui lui sont associées sont elles-mêmes possible (i.e. qu'elles satisfont toutes les règles de gestion des comptes concernés).
27. Un OV est impossible si le solde du compte origine n'est pas suffisant. Dans ce cas aucune opération n'est créée, et la tentative d'OV échoue "simplement" (i.e. sans mécanisme complexe).
28. Un OV pour lequel le compte destinataire ne peut pas être crédité (à cause d'un possible dépassement de plafond autorisé) fait l'objet d'une opération, qui est mise *en attente* de validation par l'attaché du donneur d'ordre.
29. Toute opération en attente de validation est envoyée à l'attaché concerné pour validation manuelle.
30. La validation manuelle, par un attaché, d'une opération en attente, permet à l'attaché de forcer le crédit (ou le débit) d'une somme qui entraine un solde supérieur au plafond autorisé (respectivement, un solde négatif).

Question 4.1 Compléter le modèle de classes, pour intégrer les ordres de virement. Penser aux exceptions.

4.b Fonctionnalités

Noter que les fonctionnalités suivantes ne se substituent pas aux précédentes, mais viennent les compléter.

Débiter Tout compte doit fournir une méthode qui permet d'effectuer un débit, permet de spécifier en entrée quelle opération (i.e., quel objet `Operation`) est affectée par ce débit, et retourne un booléen indiquant le succès ou l'échec de cette opération.

Plus précisément, si le débit est possible alors le solde est modifié directement, et l'opération placée en paramètre est modifiée en conséquence, avec un statut de succès.

Si le débit n'est pas possible, alors le solde reste inchangé, et l'opération placée en paramètre est modifiée en conséquence, avec un statut d'échec.

Créditer Tout compte doit fournir une méthode qui permet d'effectuer un crédit, permet de spécifier en entrée quelle opération est affectée par ce crédit, et retourne un booléen indiquant le succès ou l'échec de cette opération.

Plus précisément, si le crédit est possible alors le solde est modifié directement, et l'opération placée en paramètre est modifiée en conséquence, avec un statut de succès.

Si le crédit n'est pas possible, alors le solde reste inchangé, et l'opération placée en paramètre est modifiée en conséquence, avec un statut d'échec.

Passer un ordre de virement Tout ordre de virement doit pouvoir être passé.

En premier lieu, le compte origine est testé pour déterminer si le débit du montant à virer est possible (sans créer d'opération). Si le débit est impossible, alors le virement lui-même est impossible (puisqu'on ne peut effectuer de virement que si on dispose effectivement de la somme à virer).

Si le débit est possible, alors une tentative de crédit est effectuée, avec création d'opération ; si la tentative échoue, alors deux opérations en ATTENTE sont créées, qui sont envoyées à l'attaché client pour validation manuelle ; si la tentative réussit, alors l'opération est effective, et on effectue le débit correspondant. Noter qu'à ce stade le débit est forcément possible...

Question 4.2 Implémenter le (ou les) constructeur(s) requis pour un ordre de virement. Penser à lever, si nécessaire, la (ou les) alerte(s) spécifiée(s) dans les règles de gestion.

4.c Jeu de tests

Tester les cas suivants.

— Débit (réussi) de 123 € sur le compte courant de $\mathcal{C}$, en spécifiant une opération en entrée.

— Crédit (réussi) de 1,55 € sur le compte courant de $\mathcal{C}$, en spécifiant une opération en entrée.

— OV (réussi) de 666 € depuis le compte courant de $\mathcal{C}$, vers son compte épargne.

— OV avec échec, sans mise en attente, de 10^4 € depuis le compte courant vers le compte épargne de $\mathcal{C}$.

— OV avec échec et mise en attente, de 10 € depuis le compte courant vers le compte épargne de $\mathcal{C}$.

Question 4.3 Implémenter les nouvelles règles de gestion relatives aux ordres de virement, en supposant dans un premier temps que la validation manuelle existe (elle sera développée plus tard).

Question 4.4 Lancer l'exécution de votre programme principal plusieurs fois : que remarquez-vous, quant aux messages d'alerte (exceptions) ? Proposer une explication.

5 (subsidiaire) Validation manuelle d'une opération en attente

Question 5.1. Implémenter les règles de gestion relatives à la validation manuelle d'une opération en attente, par l'attaché autorisé. Pour cela, commencer par implémenter les méthodes suivantes, dans la (ou les) classe(s) adéquate(s).

— `double forcerDebit(double montant, Operation op, Attache lAttacheAutorise) throws PersonnelNonAutoriseException`
qui retourne le nouveau solde, et lève une exception si l'attaché de clientèle qui tente l'opération n'est pas autorisé.

— `double forcerCredit(double montant, Operation op, Attache lAttacheAutorise) throws PersonnelNonAutoriseException`
qui retourne le nouveau solde, et lève une exception si l'attaché de clientèle qui tente l'opération n'est pas autorisé.