Compte bancaire OO [oo05] - Exercice résolu

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Université Haute-Alsace

Version 20 mai 2018

Table des matières

1 Compte bancaire OO / pgcbancaire **2**
- 1.1 Classe CBancaire . . . 2
- 1.2 Procédure de traitement . . . 4
- 1.3 Programme de test . . . 4

2 Applications **6**
- 2.1 Classe CBancaire (1) / pgcbancaire1a . . . 6
- 2.2 Classe CBancaire (2) / pgcbancaire1b . . . 7
- 2.3 Classe CBancaire (3) / pgcbancaire1c . . . 8
- 2.4 Classe CBancaire (extension) / pgcbancaire2 . . . 9

Java - Compte bancaire (Solution)

Utilise Axiomatique objet, Règles OO et Structeurs ■
Durée estimée 45 min ■

Objectif
Cet exercice applique les principes de la conception objet afin de réaliser une version OO « pure » de la gestion d'un compte bancaire.

1 Compte bancaire OO / pgcbancaire

1.1 Classe CBancaire

Un compte bancaire (simplifié) est défini par le solde disponible sur le compte. Les opérations de manipulation minimales seront :

- Initialiser un compte bancaire
- Accéder au solde d'un compte bancaire
- Créditer un compte bancaire
- Débiter un compte bancaire

On étendra également l'interface avec :

- Un constructeur permettant d'initialiser le solde.
- Une méthode `afficher` qui affiche l'objet.
- Une méthode `virerVers` qui effectue un virement vers un autre compte bancaire.

Rappelez les quatre règles de l'OO.

Solution simple

Règle 1 La **structure interne** de l'objet est **inaccessible** directement : il faut passer par les méthodes de l'interface.

Règle 2 Les attributs **ne doivent pas être accessibles directement** depuis l'extérieur mais uniquement par des méthodes.

Règle 3 Il existe un mécanisme qui permet d'éviter les problèmes dus aux valeurs indéterminées, en rendant l'initialisation automatique à la déclaration d'un objet. Ce mécanisme est fondé sur les **constructeurs**.

Règle 4 De même, il existe une méthode particulière qui est appelée lorsqu'un objet doit être détruit. Cette méthode s'appelle **le destructeur**.

Écrivez une classe `CBancaire` qui inclut un attribut `solde` (réel).
(Veillez à respecter la **Règle 2**.)

Écrivez un constructeur par défaut.
(Application de la **Règle 3**.)

Écrivez un constructeur à un paramètre `x` (réel) initialisant son attribut.

Écrivez un accesseur `getSolde` du solde du compte.
(Application de la **Règle 1**.)

Écrivez une méthode `crediter(x)` qui crédite le compte du montant `x` (réel).
Supposez `x` positif.

Écrivez une méthode `debiter(x)` qui débite le compte du montant `x` (réel).
Supposez `x` positif.

Écrivez une méthode `afficher` qui affiche le solde.

Écrivez une méthode `virerVers(cb,montant)` qui effectue un virement de `montant` (réel) vers un `CBancaire cb`.

Validez votre classe et vos méthodes avec la solution.

Solution Java @[CBancaire.java]

```java
class CBancaire{
private double m_solde;

public CBancaire(){
  m_solde = 0.0;
}

public CBancaire(double x){
  m_solde = x;
}

public double getSolde(){
  return m_solde;
}

public void crediter(double x){
  m_solde += x;
}

public void debiter(double x){
  m_solde -= x;
}

public void afficher(){
  System.out.println(getSolde() + " euros");
}

public void virerVers(CBancaire cb, double mt){
  cb.crediter(mt);
  debiter(mt);
}
}
```

1.2 Procédure de traitement

Écrivez **l'en-tête** d'une procédure `traiterCompte(cb)` qui traite un `CBancaire cb`.

Demandez à l'utilisateur un `montant` (réel). Affichez :

```
Tapez vos montants successifs (0 pour finir)
Votre montant (0==fin)?
```

Complétez votre procédure comme suit :
`TantQue` le `montant` n'est pas nul :

1. Selon qu'il est positif ou négatif, elle crédite ou débite `cb` de `montant`.
2. Ensuite elle affiche l'état du nouveau compte.
3. Puis si le solde de `cb` est négatif, elle avertit l'utilisateur afin qu'il le réapprovisionne.
4. Finalement elle demande à nouveau le `montant` afin de pouvoir quitter la boucle.

Validez votre procédure avec la solution.

Solution Java @[UtilsCB.java]

```java
import java.util.Scanner;
import java.util.Locale;
public class UtilsCB {
static void traiterCompte(CBancaire cb){
  Scanner cin = new Scanner(System.in);
  cin.useLocale(Locale.US);
  System.out.print("Votre montant (0==fin)? ");
  double montant = cin.nextDouble();
  while (montant != 0.0){
    if (montant > 0.0){
      cb.crediter(montant);
    }
    else{
      cb.debiter(-montant);
    }
    System.out.println("Etat du compte: " + cb.getSolde() + " euros");
    if (cb.getSolde() < 0.0){
      System.out.println("OUPS... cb en negatif -- veuillez alimenter");
    }
    System.out.print("Votre montant (0==fin)? ");
    montant = cin.nextDouble();
  }
}
}
```

1.3 Programme de test

Écrivez un programme qui instancie un `CBancaire cb` puis le manipule par la procédure `traiterCompte`. Après avoir manipulé le compte, si le solde est positif :

- Saisissez le montant du virement.
- Effectuez le virement sur un autre compte.
- Affichez l'état des deux comptes.

Testez. Exemple d'exécution :

```
Etat du compte: 200.0 euros
Votre montant (0==fin)? 150
Etat du compte: 350.0 euros
Votre montant (0==fin)? -400
Etat du compte: -50.0 euros
OUPS... cb en negatif -- veuillez alimenter
Votre montant (0==fin)? 170
Etat du compte: 120.0 euros
Votre montant (0==fin)? 0
```

Validez votre programme avec la solution.

Solution Java @[pgcbancaire1.java]

```java
public class PGCBancaire1 {
  public static void main(String[] args) {
    CBancaire cb = new CBancaire(200.0);
    System.out.print("Etat du compte: ");
    cb.afficher();
    UtilsCB.traiterCompte(cb);
  }
}
```

2 Applications

2.1 Classe CBancaire (1) / pgcbancaire1a

Utilise Compte bancaire OO ■
Durée estimée 10 min ■

Créez un programme qui déclare et initialise trois objets de la classe `CBancaire` avec les données suivantes :

- Numéro de compte 1 : solde initial de 200€.
- Numéro de compte 2 : solde initial 40€.
- Numéro de compte 3 : solde initial 20000€.

Affichez l'état de chacun des comptes.

Augmentez chaque compte de 10%.

Le compte 3 donne 5000€ à compte 1 et le reste à compte 2.

2.2 Classe CBancaire (2) / pgcbancaire1b

Utilise Compte bancaire OO ■
Durée estimée 10 min ■

Comment fonctionne la méthode `virerVers` ?
Combien de comptes fait-elle intervenir ?

Écrivez un programme qui crée deux comptes.

Réalisez les opérations suivantes :

- Dépôt de 500 euros sur le premier compte.
- Dépôt de 1000 euros sur le second compte.
- Retrait de 10 euros sur le second compte.
- Virement de 75 euros du premier compte vers le second.
- Affichage des soldes des deux comptes.

2.3 Classe CBancaire (3) / pgcbancaire1c

Utilise Compte bancaire OO ■
Durée estimée 15 min ■

Créez un tableau de dix `CBancaire`.

Faites un dépôt de 200 euros plus une somme égale à 100 fois l'indice du compte dans chaque case du tableau.

Faites un virement de 20 euros de chaque compte vers chacun des comptes qui le suivent dans le tableau. Exemple : du compte d'indice 5, il faut faire des virements vers les comptes d'indice 6, 7, 8 et 9.

Affichez les soldes de tous les comptes.

2.4 Classe CBancaire (extension) / pgcbancaire2

Utilise Compte bancaire OO ■
Durée estimée 10 min ■

Modifiez la méthode `debiter` pour empêcher le retrait quand le compte n'est pas suffisamment approvisionné.

Comment modifier la méthode pour savoir si le débit a été ou non effectué ?

Écrivez un petit programme de test.

Testez.