جامعة هواري بومدين
للعلوم والتكنولوجيا
U S T H B

Corrigé d'exercices de TD POO

Série n°4 – Héritage et Polymorphisme

S. BOUKHEDOUMA

USTHB – FEI – département d'Informatique
Laboratoire des Systèmes Informatiques -LSI
sboukhedouma@usthb.dz

Exercice 3 –Série n°4

Exercice 3

Dans cet exercice, on s'intéresse à la manipulation de comptes bancaires de clients. Un compte bancaire est décrit par un ***numéro***, une ***date*** de création, un ***état*** (bloqué ou non), et un ***solde*** (somme d'argent contenue dans le compte), il possède un ***titulaire*** (client, dans la classe compte, on enregistre le numéro du client). Un compte peut être créé, crédité (ajouter argent), débité (retirer argent) ou bloqué.
Un client est décrit par un ***numéro***, un ***nom***, un ***prénom***, une ***date*** de naissance et une ***adresse***. Un client peut changer son adresse. A tout moment, on doit pouvoir afficher les informations des clients et des comptes (redéfinir la méthode toString).

1. Définir en java les classes qui décrivent les deux entités **COMPTE** et **CLIENT**.

Exercice 3 –Série n°4

```
public class Date
{  private int jour, mois, année;

public Date ( int jour, int mois, int année)
  { this.jour =  jour;  this.mois = mois;  this.année = année;}
public Date () {}

public String toString()
            { return  (jour + "-" +mois+ "-" + année); }

public void lire ()
{ java.util.Scanner e = new java.util.Scanner (System.in);
  System.out.println ("Saisir Date");
   jour = e.nextint(); mois = e.nextInt(); année = e.nextint(); }

Public void setDate  (Date d)
{ jour = d.jour; mois= d.mois; année  = d.année; }                    }
```

Exercice 3 –Série n°4

```
public enum  Etat { actif , bloqué}

public class Compte

{  private long numCompte ;
   private Date DateCreation  = new Date();
   private Etat  etaCompte ;
   private float solde ;
   private long titulaire ;
                                   // constructeur
public Compte ( long num,  Date d, float solde, long titulaire)
  { numCompte = num; DateCreation . setDate (d);
    this.solde =  solde;  this.titulaire = titulaire;  etaCompte = Etat.actif;}
```

Exercice 3 –Série n°4

```
public Compte ()
  { etaCompte =  Etat.actif;}

public void saisir ()
{java.util.Scanner e = new java.util.Scanner (System.in);
 System.out.print ("Saisir Compte avec solde initial");
 numCompte = e.nextLong(); DateCréation.lire();
 solde = E.nextFloat(); }
```

Exercice 3 –Série n°4

```
//suite de la classe Compte
public String toString()
        { return  (numCompte + "\t" +DateCréation+ "\t" + etaCompte+ "\t"
                + solde + "\t"+ titulaire ); }

Public void afficher ()
   { System.out.println (this.toString()); }

// accesseurs
public long getNumCompte()  { return numCompte; }
public long getTitualire()  { return  titulaire; }
public long setTitualire(long num)  {titulaire = num; }
public Date getDatCréation()  { return DateCréation; }
public float getSolde()  { return solde; }
public Etat getEtatCompte()  { return etaCompte; }
```

Exercice 3 –Série n°4

//suite de la classe Compte

```
public void créditer(float s)  {
 { System.out.println (“ Opération de crédit d’une somme :” + s);
                    solde += s;}
public void débiter(float s)  {
System.out.println (“ Opération de débit d’une somme :” + s);
                solde -= s;}
public void changerEtat (Etat e)  { etaCompte = e; }

} //fin de la classe Compte
```

Exercice 3 –Série n°4

```
public class Client
{ private long numClient ;
  private String nom, prénom ;
  private Date  dateNaiss  = new Date();    private  String  adresse ;

public Client  (Long num, String nom, String prénom, Date d, String adr )
{  numClient = num; this.nom = nom; this.prénom = prénom;
  dateNaiss.setDate (d);    adresse = adr; }

public Client () {}

public void saisir()
{ System.out.print ("introduire la description du Client"); }
  Scanner e = new Sacnner (System.in);    numClient = e.nextLong(); nom =
  e.nextLine(); prénom = e.nextLine();    dateNaiss.lire();
  adresse = e.nextLine();
```

Exercice 3 –Série n°4

```
//suite de la classe Client
public String toString()
        { return  (numClient + "\t" +nom+ "\t" + prénom+ "\t" +
           dateNaiss.toString()+ "\t" + adresse); }

Public void afficher ()
   { System.out.print (this.toString()); }

                                        // accesseurs
public long getNumClient()  { return numClient; }
public String getNom()  { return nom; }
public String getPrénom()  { return prénom; }
public Date getDatNais()  { return dateNaiss; }
public String getAdresse()  { return adresse; }
public void setAdresse(String adr)  { adresse = adr; }
} //fin de la classe Client
```

Exercice 3 –Série n°4

Exercice 3 (suite)

2. Ecrire un programme qui crée et remplit deux structures (de type Vector ou ArrayList), l'une regroupant les données de clients et l'autre regroupant les données des comptes.
Les informations concernant les comptes et les clients sont entrées au clavier.

3. Ecrire un programme qui affiche les numéros de compte, les soldes, les noms et prénoms de clients ayant des comptes débiteurs (solde négatif).

Exercice 3 –Série n°4

```
import java.util.Scanner;
import java.util.Vector;
import java.util.ArrayList;

public class ProgCompte
 {  public static void main ( String args[])
   {  Scanner e = new Scanner (System.in);

      Vector <Compte>  CP = new Vector <Compte> ();

      ArrayList <Client>  CL = new ArrayList<Client> ();

      Compte  cpt;  Client  C;  // références d'objets
      char rep;   // réponse de l'utilisateur
```

Exercice 3 –Série n°4

```
// création des objets Client et Compte
  do
     { System.out.println ("Saisie d'un client");
        C = new Client();
         C.saisir ();  CL.add(C);

        System.out.println ("le compte associé");
        cpt = new Compte();
        cpt. Saisir();
        cpt.setTitulaire (C.getNumClient());  //associer la num client au compte
        CP.add(cpt);
         System.out.println ("Autre client?  O/N");
         rep = e.next().charAt(0);
      }
while (rep.toUpperCase() = = 'O')
```

Exercice 3 –Série n°4

```
// affichage des comptes débiteurs

System.out.println  ( " Les comptes débiteurs sont:"


for (int i = 0, i< CP.size(); i++)
        { cpt = CP.get(i);     // récupérer l'objet Compte
          if (cpt.getSolde()<0)
              C = CL.get(i); // récupérer l'objet Client correspondant
               System.out.println  (cpt.getNumCompte() + "\t" +C.getNom()+
                        "\t" + C.getPrénom() + "\t" + cpt.getSolde());
         }
}
```

Exercice 3 –Série n°4

Exercice 3 (suite)

Dans cette partie, on considère différents types de comptes :

- Un compte « Chèque » où le solde ne peut être que créditeur (positif).
- Un compte « Courant » dont le solde peut devenir négatif grâce à une autorisation du banquier et ne peut descende en dessous d'un seuil S négatif fixe (on suppose que le seuil est fixé à 5000 da).
- Un compte « Epargne » générateur d'intérêts selon un taux fixe (le taux est égal à 7%). Le cumul des intérêts est calculé par rapport au solde et en fonction du taux d'intérêt.

4. Proposer une nouvelle description des classes en utilisant l'héritage.

Exercice 3 –Série n°4

```
Import java.util.Scanner;
public class CompteCheque extends  Compte {

                                        // constructeurs
public CompteCheque ( long num,  Date d, float solde, long titulaire)
{  super (num, d, solde,  titulaire);     }

public CompteCheque ()
{  super ();     }
                                // redéfinition de la méthode débiter

public  void débiter (float s)
   { if (super.getsolde () >= s)   super.débiter(s);
            else System.out.println ( “solde insuffisant, opération impossible”);}
```

Exercice 3 –Série n°4

```
// suite de la classe CompteCheque

                    // redéfinition de la méthode toString

public  String toString ()
   {  return ("Compte Chèque:     "  + super. toString(); }

} // fin de la classe CompteCheque
```

Exercice 3 –Série n°4

```
Import java.util.Scanner;

public class CompteCourant extends  Compte {
        private final static float Seuil= 5000;     // constante de classe
        private boolean autorisation;

                                        // constructeurs
public CompteCourant ( long num,  Date d, float solde, long titulaire)
{  super (num, d, solde,  titulaire);     }

public CompteCourant ()
{  super ();      }
```

Exercice 3 –Série n°4

```
// suite de la classe CompteCourant

                    // redéfinition de la méthode débiter

public  void débiter (float s)
   { if (super.getSolde() >= s)   super.débiter(s);
                else
     if (super.getSolde() – s <-5000 ||  autorisation == false)
            System.out.println ( “Opération impossible”);
      else
        { super.debiter (s);
          System.out.println ( “votre solde est négatif”);}
    }
```

Exercice 3 –Série n°4

```
// suite de la classe CompteCourant
                                    // redéfinition de la méthode toString

public  String toString ()
   {  return ("Compte Courant:     "  + super. toString() + "    " autorisation); }

                                    // Autres méthodes

public boolean getAuto()  { return autorisation; }
public void setAuto (boulean auto)  { autorisation = auto; }

} // fin de la classe CompteCourant
```

Exercice 3 –Série n°4

```
Import java.util.Scanner;

public class CompteEpargne extends  Compte {
        private final static taux= 0,07;     // constante de classe


                                        // constructeurs
public CompteEpargne( long num,  Date d, loat solde, long titulaire)
{  super (num, d, solde,  titulaire);     }

public CompteEpargne ()
{  super ();      }
```

Exercice 3 –Série n°4

```
// suite de la classe CompteEpargne

                    // redéfinition de la méthode débiter

public  void débiter (float s)
   { if (super.getSolde() >= s)   super.débiter(s);
            else System.out.println ( “solde insuffisant, opération impossible”);}

                    // redéfinition de la méthode toString

public  String toString ()
   {  return (“Compte Epargne:     ”  + super. toString() ; }
```

Exercice 3 –Série n°4

```
// suite de la classe CompteEpargne

                                        // autre méthode

public  float claculInterêts ()
   { return (taux*super.getSolde();}

} // fin de la classe CompteEpargne
```

Exercice 3 –Série n°4

Exercice 3 (suite)

On suppose que les données des comptes et des clients sont déjà stockées dans des structures de type ArrayList (classe prédéfinie dans java.util).

5. Ecrire un programme qui calcule les intérêts cumulés sur tous les comptes Epargne.
   Les résultats devront être affichés en précisant le nom et prénom des clients.

Exercice 3 –Série n°4

```
import java.util.Scanner;
Import java.util.Vector;
Import java.util.ArrayList;

public class ProgCompte
 {  public static void main ( String args[])
   {  Scanner e = new Scanner (System.in);
      ArrayList <Compte>  CP = new ArrayList <Compte> ();

      ArrayList <Client>  CL = new ArrayList<Client> ();

      Compte  cpt;  Client  C;  // références d'objets
```

Exercice 3 –Série n°4

```
// affichage des intérêts des comptes Epargne

System.out.println  ( " Les comptes Epargnes sont:"
for (int i = 0, i< CP.size(); i++)
        { cpt = CP.get(i);
          if (cpt  instanceof  CompteEpargne)
              {C = CL.get(i);
                System.out.println  (cpt.getNumCompte() + "\t" +C.getNom()+
                        "\t" + C.getPrénom() + "\t" + cpt.getSolde() + "\t" +
                         ((CompteEpargne) cpt).calculInterets() ); }
                                        // on applique un cast explicite
}     }
```

Exercice 3 –Série n°4

Question

Si les objets Compte et Client n'étaient pas rangés dans le même ordre dans les collections CP et CL

Comment devrait-on procéder?

Exercice 3 –Série n°4

```
// affichage des intérêts des comptes Epargne

System.out.println  ( " Les comptes Epargnes sont:"
for (int i = 0, i< CP.size(); i++)
        { cpt = CP.get(i);
          if (cpt instanceof  CompteEpargne)
              { for (Client  C: CL)     // parcourir  la collection CL avec la réference C
                  if  (cpt.titulaire = = C.numClient))
                      { System.out.println  (cpt.getNumCompte() + "\t"
                +C.getNom()+   "\t" + C.getPrénom() + "\t" + cpt.getSolde() + "\t" +
                          ((CompteEpargne) cpt).calculInterets() );
                      break;}          }
}    }
```