UNIVERSITÉ NATIONALE DU VIETNAM À HANOÏ
INSTITUT FRANCOPHONE INTERNATIONAL.

Rapport du TP1 de Génie Logiciel

Conception d'un logiciel de gestion bancaire.

Par

OUBDA Raphael Nicolas Wendyam

SAÏDI Ally Azaria

Sous la supervision de : Dr. Ho Tuong Vinh

Année académique 2017-2018

Table des matières

Introduction

La banque est un moyen efficace de garder, de gérer et de faire circuler de l'argent. Elle a pour but de générer des intérêts ou des profits pour ses clients et pour elle-même.

Pour cela la banque doit structurer les données, les comptes clients et les opérations qui y sont faites en vue de garder les traces.

Ainsi c'est dans le cadre du premier travail pratique (TP1) de génie logiciel, il nous a été demandé de concevoir et de réaliser une application de gestion bancaire afin de nous rappeler le concept de modélisation avec UML, la programmation orientée objet avec java et de nous familiariser avec un environnement de développement (IDE) libre. Le présent rapport expose les étapes de conception et de réalisation de l'application.

I. Les Spécifications du logiciel.

Les spécifications de l'application de gestion de banque sont :

1. La banque gère un ensemble de compte ;
2. Un client peut posséder plusieurs comptes avec des taux d'intérêt différents ;

3. Chaque compte contient les informations suivantes :
   - le numéro de compte, un entier positif, qui sera indiqué par la banque ;
   - le numéro d'identification du titulaire du compte, un entier positif ;
   - le nom du titulaire ;
   - le solde du compte, un nombre flottant ;
   - le taux d'intérêt, un nombre flottant entre 0 et 100.

4. La banque peut exécuter les opérations suivantes :
   - créer un nouveau compte ;
   - déposer un certain montant d'argent dans un compte ;
   - retirer un certain montant d'argent d'un compte ;
   - consulter le solde d'un compte ;
   - calculer l'intérêt pour tous les comptes et mettre à jours leur solde ;
   - produire un rapport qui inclut une liste de comptes avec les informations suivantes : le numéro de compte, le solde, et les transactions effectuées (déposer et retirer l'argent) depuis que le compte est créé ;
   - faire une recherche qui permet, à partir du numéro d'identification d'une personne, de savoir si elle possède un ou des comptes dans la banque.

5. En ce qui concerne les transactions d'un compte spécifique, les informations suivantes sont intéressantes :
   - la date de transaction ;
   - le type de transaction : déposer ou retirer d'argent ;
   - le montant d'argent.

II. Conception de l'application.

Dans la phase de conception de notre application nous avons élaboré nos différents diagrammes avec le logiciel StarUml et nous avons conçu une base de données banque avec MYSQL qui servira a stocké les différentes informations de notre application.

II.1 Diagramme de cas d'utilisation.

Le diagramme de cas d'utilisation modélise les besoins permettant de faire l'inventaire des grandes fonctions attendues dans notre application de gestion de compte bancaire. Dans notre application nous avons un acteur:

- Les caissiers : ils représentent la banque et ils interagissent directement avec le système.
- Les cas d'utilisations sont les suivantes :
- créer un nouveau compte ;
- créditer un compte ;
- solder un compte ;
- consulter le solde d'un compte ;
- calculer l'intérêt pour tous les comptes et mettre à jours leur solde ;
- produire un rapport qui inclut une liste de comptes avec les informations suivantes : le numéro de compte, le solde, et les transactions effectuées (déposer et retirer l'argent) depuis que le compte est créé ;
- faire une recherche des comptes d'un client.

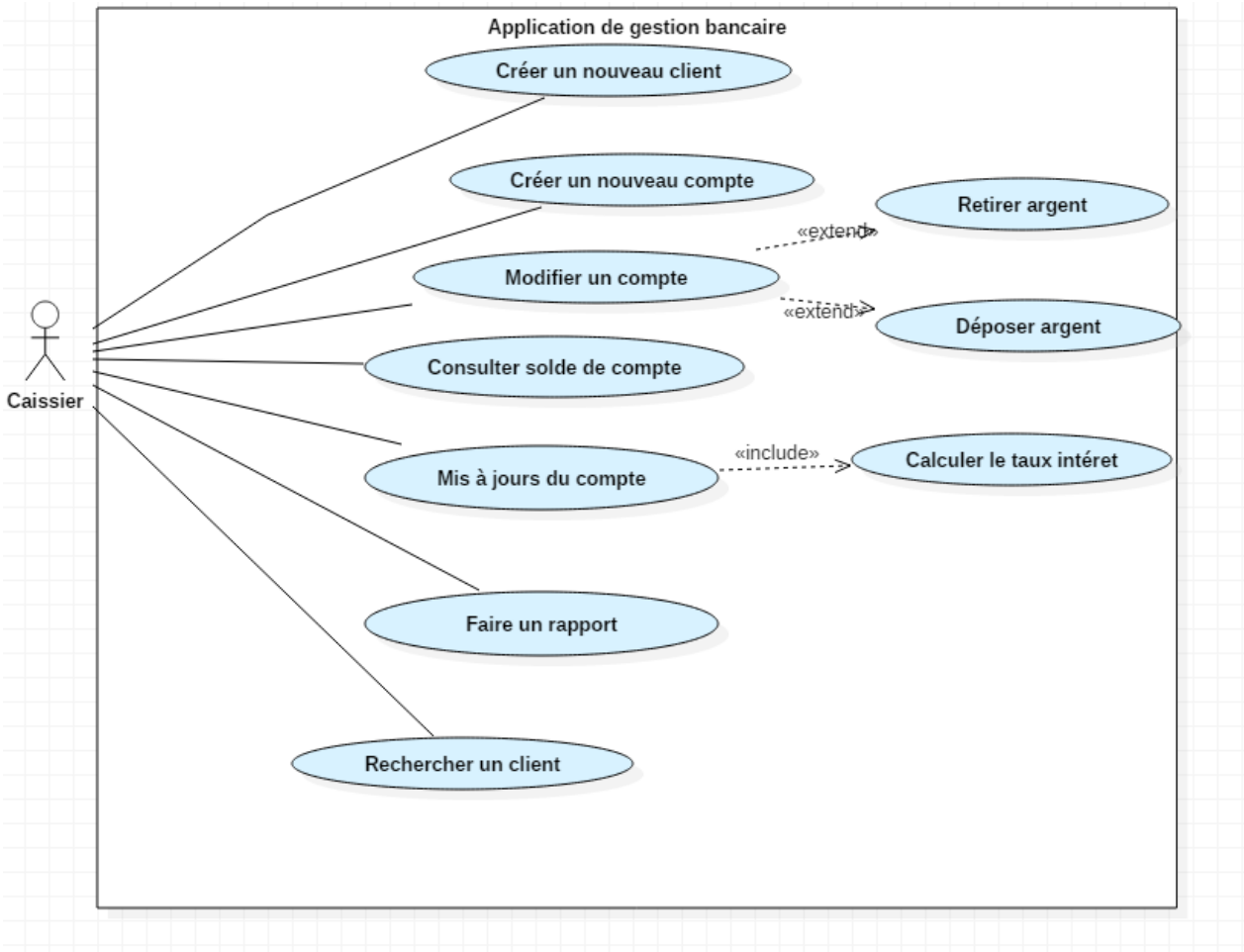

Figure 1:Diagramme de cas d'utilisation.

II.2 Diagramme de séquence.

Le Diagramme de séquence montre le scénario de déroulement d'un cas d'utilisation. Déroulement normal de création d'un compte bancaire :

- Le caissier envoie une demande de création de compte bancaire ;
- Le système renvoie le formulaire de création d'un compte ;
- Le caissier remplit les informations relatives et renvoie le formulaire pré remplit ;
- Le système enregistre le compte et renvoie un message de réussite de l'opération.

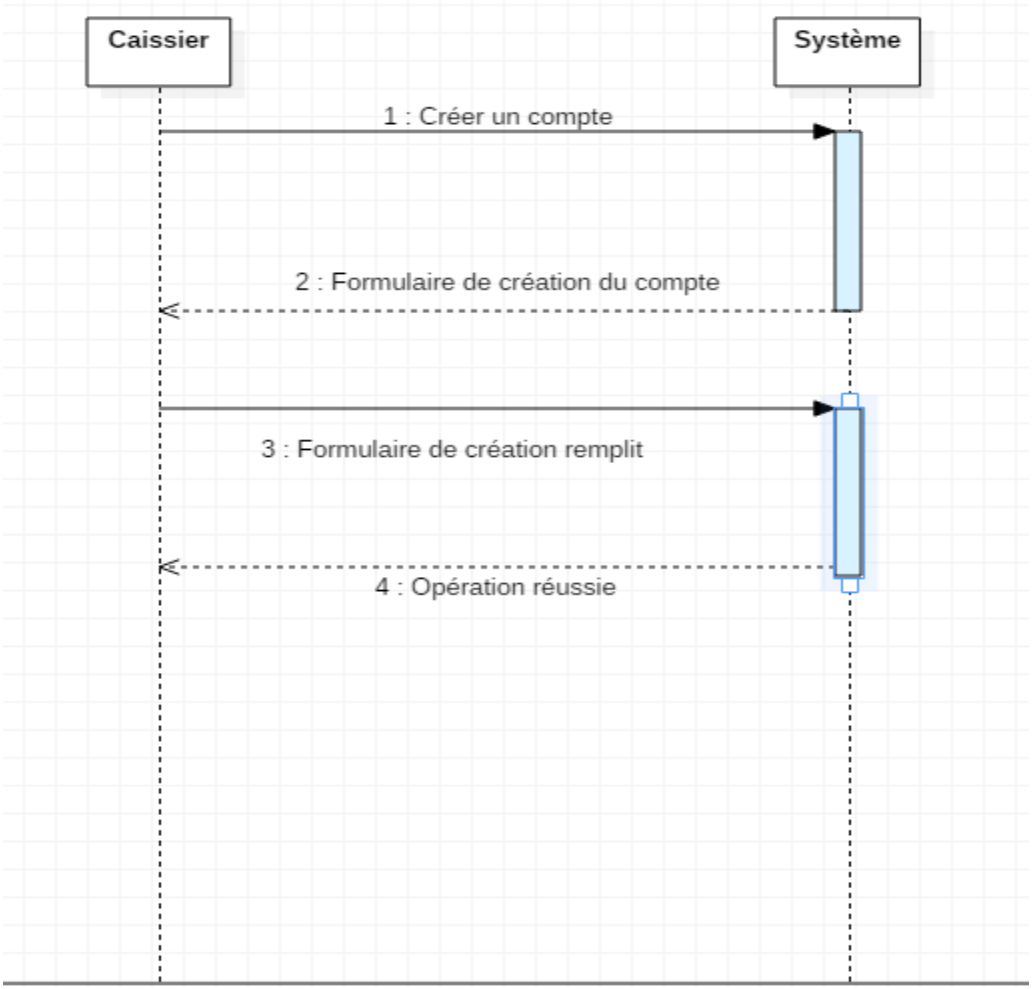

Figure 2: Diagramme de séquence de création d'un compte bancaire.

II.3 Diagramme de classe.

Le diagramme de classe permet de représenter les différentes classes et interfaces de notre application ainsi que les différentes relations entre elles. Notre diagramme de classes comporte six(6) classes.

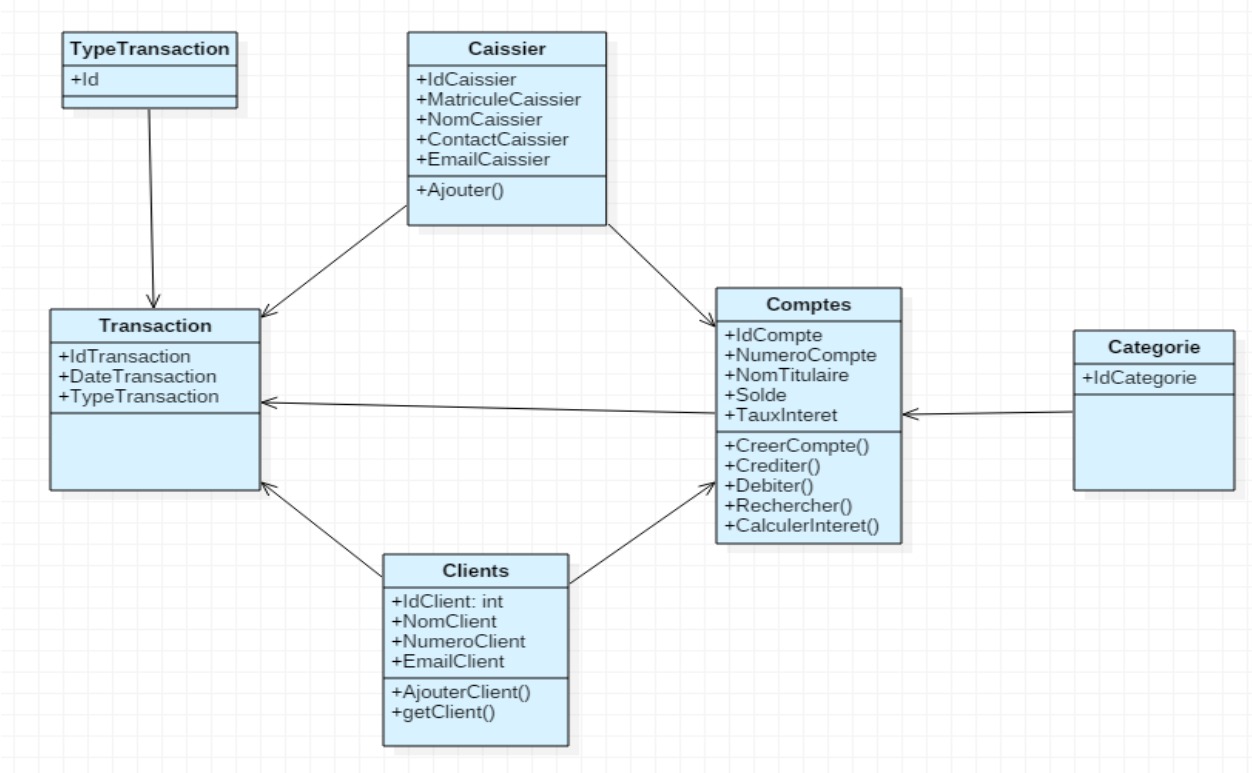

Figure 3:Diagramme de classe

II.4 Conception de la base de données.

Pour la base de données nous avons utilisé le système de gestion de base de données MySQL. La base de données se nomme 'banque' et comporte six tables :

- La table 'clients' stocke les différentes informations concernant les clients. Il contient tous les clients de la banque ;
- La table 'caissiers' stocke les informations concernant les caissiers de la banque;
- La table 'categoriecompte' stocke les différents type de comptes crées par la banque, c'est-à-dire contient les différents taux d'intérêts que peut générer un compte.
- La table 'comptes' stocke tous les comptes de la banque ;
- La table 'typetransaction' contient les différents types de transactions possibles c'est-à-dire les transactions créé, consulté, débité, crédité.
- La table 'transaction' qui stocke les différentes transactions (débiter, créditer, consulter..) d'un compte.

	Table	Action	Lignes [1]	Type	Interclassement	Taille	Perte
☐	banktransaction		9	MyISAM	latin1_swedish_ci	6,3 Kio	-
☐	caissiers		3	MyISAM	latin1_swedish_ci	2,1 Kio	-
☐	categoriecompte		2	MyISAM	latin1_swedish_ci	2,1 Kio	-
☐	clients		3	MyISAM	latin1_swedish_ci	2,1 Kio	-
☐	comptes		7	MyISAM	latin1_swedish_ci	5,2 Kio	-
☐	typetransac		4	MyISAM	latin1_swedish_ci	2,1 Kio	-
	6 table(s)	Somme	28	InnoDB	latin1_swedish_ci	19,9 Kio	0 o

Figure 4:Les tables de la base de données.

III. Implémentation et tests de l'application.

III.1 Implémentation de l'application.

Pour l'implémentation de l'application nous avons utilisé l'IDE NetBeans qui est un logiciel libre qui permet de coder en java. L'application a été implémentée sous Linux 16.0.4.
Pour l'implémentation nous avons fait une approche objet en créant des packages et en utilisant un patron de conception MVC:

- Le package 'META-INF' qui contient notre fichier 'persistence.xml'. Ce fichier permet la connexion à notre base de données 'banque' ;

- Le package 'Manager.models.com' : qui contient les différents managers de notre application. Ces managers assurent la gestion de nos données et garantit leur intégrité dans notre base de données 'banque'. Ils contiennent les différentes méthodes d'insertion et de suppression. Nous avons six managers (ManagerCaissiers, ManagerClient, ManagerCategories, ManagerCompte, ManagerTransac, et ManagerTypeTransac) ;

- Le package 'controller.banque.com' : il contient un seul controlleur qui est 'BanqueController.java' qui permet de controller les entrées (saisies) de clavier de l'utilisateur (caissier) de l'application en vue de déterminer l'opération à effectuer.

- 'Banque' : ce package contient la classe 'Banque ' dans lequel nous avons défini la méthode d'entée de notre application c'est à dire le « main. ».

III.2 Test de l'application.

Les tests réalisés dans notre application sont les suivantes :

- Menu de l'application ;

```
**********************************
*GESTION DE COMPTE D'UN CLIENT*
**********************************
Menu:
1.CREER UN COMPTE POUR LE CLIENT
2.CONSULTER UN COMPTE CLIENT
3.DEBITER UN COMPTE CLIENT
4.CREDITER UN COMPTE CLIENT
5.RECHERCHER UN COMPTE CLIENT
6.CALCULE DE L'INTERET POUR TOUS LES COMPTES
7.RAPPORT SUR UN OU PLUSIEURS COMPTES
8.RETOUR AU MENU
ENTRER VOTRE CHOIX ...
```

Figure 5: Menu de l'application.

- Création d'un compte bancaire ;

```
**********************************
*GESTION DE COMPTE D'UN CLIENT*
**********************************
Menu:
1.CREER UN COMPTE POUR LE CLIENT
2.CONSULTER UN COMPTE CLIENT
3.DEBITER UN COMPTE CLIENT
4.CREDITER UN COMPTE CLIENT
5.RECHERCHER UN COMPTE CLIENT
6.CALCULE DE L'INTERET POUR TOUS LES COMPTES
7.RAPPORT SUR UN OU PLUSIEURS COMPTES
8.RETOUR AU MENU
ENTRER VOTRE CHOIX ...

1
vous avez choisi le 1 pour la création de compte:
Veuillez saisir votre numero de téléphone:
```

Figure 6: Création d'un compte bancaire.

- Créditer un compte ;

```
1.CREER UN COMPTE POUR LE CLIENT
2.CONSULTER UN COMPTE CLIENT
3.DEBITER UN COMPTE CLIENT
4.CREDITER UN COMPTE CLIENT
5.RECHERCHER UN COMPTE CLIENT
6.CALCULE DE L'INTERET POUR TOUS LES COMPTES
7.RAPPORT SUR UN OU PLUSIEURS COMPTES
8.RETOUR AU MENU
ENTRER VOTRE CHOIX ...
4
vous avez choisi le 4
Entrer le numero de compte à Créditer
106
Entrez le montant à créditer:
1299009099
Le compte 106 avec un montantde 1.0000900988001982E15 a été crédité de 1.299009099E9
 d'ou le nouveau solde est:1.0000913978092972E15
```

Figure 7:Créditer un compte.

- Calcul du taux d'intérêt ;

```
vous avez choisi le 6
************************************************************
*****CALCUL DE L'INTERET POUR CHAQUE COMPTE DE LA BANQUE*****
************************************************************
Le compte 100 appartenant à SAIDI AZARIA dont le solde est de 100000.23 de Type PREMIUM de Taux d'interet 0.5
       -àprès calcul de l'interet de 50000.115, Le solde Total est de:150000.345
Le compte 101 appartenant à SAIDI AZARIA dont le solde est de 1232340.34 de Type PREMIUM de Taux d'interet 0.5
       -àprès calcul de l'interet de 616170.17, Le solde Total est de:1848510.5100000002
Le compte 102 appartenant à SAIDI AZARIA dont le solde est de 10000.3454 de Type PREMIUM de Taux d'interet 0.5
       -àprès calcul de l'interet de 5000.1727, Le solde Total est de:15000.518100000001
Le compte 103 appartenant à KANDA HUGUES dont le solde est de 2.0002233334E8 de Type PREMIUM de Taux d'interet 0.5
       -àprès calcul de l'interet de 1.0001116667E8, Le solde Total est de:3.0003350001E8
Le compte 104 appartenant à KANDA HUGUES dont le solde est de 1.0000000345667677E7 de Type PREMIUM de Taux d'interet 0.5
       -àprès calcul de l'interet de 5000000.1728338385, Le solde Total est de:1.500000051850151 6E7
Le compte 105 appartenant à TSHIBANDA FRANCK dont le solde est de 100.0 de Type STANDARD de Taux d'interet 0.3
       -àprès calcul de l'interet de 30.0, Le solde Total est de:130.0
Le compte 106 appartenant à SAIDI AZARIA dont le solde est de 1.0000900988002102E15 de Type PREMIUM de Taux d'interet 0.5
       -àprès calcul de l'interet de 5.00045049400105 1E14, Le solde Total est de:1.500135148200315 5E15
-----------------fin du calcul de l'interet------------------
******************************
```

Figure 8: Calcul du taux d'intérêt.

- Rechercher un client ;

```
4.CREDITER UN COMPTE CLIENT
5.RECHERCHER UN COMPTE CLIENT
6.CALCULE DE L'INTERET POUR TOUS LES COMPTES
7.RAPPORT SUR UN OU PLUSIEURS COMPTES
8.RETOUR AU MENU
ENTRER VOTRE CHOIX ...
5
vous avez choisi le 5
Entrer le numero d'identification du client à rechercher
652
voici le nombre de compte associé au client 652 : 4
Mm/Mr SAIDI ALLY AZARIA N°652 possede le(s) compte(s) suivant:
le 100 a un solde de 100000.23 qui un compte PREMIUM avec un Taux d'interet de 0.5--- 652
le 101 a un solde de 1232340.34 qui un compte PREMIUM avec un Taux d'interet de 0.5--- 652
le 102 a un solde de 10000.3454 qui un compte PREMIUM avec un Taux d'interet de 0.5--- 652
le 106 a un solde de 1.0000900988002102E15 qui un compte PREMIUM avec un Taux d'interet de 0.5--- 652
********************************
*GESTION DE COMPTE D'UN CLIENT*
********************************
Menu:
1.CREER UN COMPTE POUR LE CLIENT
```

Figure 9: Recherche d'un client.

- Consulter un solde ;

```
4.CREDITER UN COMPTE CLIENT
5.RECHERCHER UN COMPTE CLIENT
6.CALCULE DE L'INTERET POUR TOUS LES COMPTES
7.RAPPORT SUR UN OU PLUSIEURS COMPTES
8.RETOUR AU MENU
ENTRER VOTRE CHOIX ...
2
vous avez choisi le 2 qui est la consultation d'un  compte
Entrer le numero de compte à consulter
106
Numero de Compte:106
 Type de compte:PREMIUM
 Taux d'interet:0.5
 Solde:1.0000913978092972E15
Date de creation:Thu Jan 25 00:00:00 CET 2018
Titulaire de Compte:SAIDI AZARIA ALLY
Numero de Telephone:0820355951
-----------------------------------------Fin consultation-----------------------------------------
```

- Produire le rapport d'un compte.

```
3.DEBITER UN COMPTE CLIENT
4.CREDITER UN COMPTE CLIENT
5.RECHERCHER UN COMPTE CLIENT
6.CALCULE DE L'INTERET POUR TOUS LES COMPTES
7.RAPPORT SUR UN OU PLUSIEURS COMPTES
8.RETOUR AU MENU
ENTRER VOTRE CHOIX ...
7
vous avez choisi le 7
Entrer une liste des comptes separé par des traits d'unions,ex:100-109-111 ou juste un compte
106-102-100-101
---------------------------------------106€----------------------------------------
Compte 106 solde 1.0000900988002102E15 date de création Thu Jan 25 00:00:00 CET 2018
        DEBITER-- 2000.67--- Thu Jan 25 00:00:00 CET 2018
---------------------------------------106€----------------------------------------
Compte 106 solde 1.0000900988002102E15 date de création Thu Jan 25 00:00:00 CET 2018
        DEBITER-- 2000.345--- Thu Jan 25 00:00:00 CET 2018
---------------------------------------106€----------------------------------------
Compte 106 solde 1.0000900988002102E15 date de création Thu Jan 25 00:00:00 CET 2018
        DEBITER-- 1290.123--- Thu Jan 25 00:00:00 CET 2018
---------------------------------------106€----------------------------------------
Compte 106 solde 1.0000900988002102E15 date de création Thu Jan 25 00:00:00 CET 2018
        DEBITER-- 19009.12--- Thu Jan 25 00:00:00 CET 2018
---------------------------------------106€----------------------------------------
Compte 106 solde 1.0000900988002102E15 date de création Thu Jan 25 00:00:00 CET 2018
        DEBITER-- 12389.23--- Thu Jan 25 00:00:00 CET 2018
---------------------------------------106€----------------------------------------
Compte 106 solde 1.0000900988002102E15 date de création Thu Jan 25 00:00:00 CET 2018
        DEBITER-- 1897.23--- Thu Jan 25 00:00:00 CET 2018
################################################################################################################################
################################################################################################################################
################################################################################################################################
--------------------------------------------------------------------------------------------------------------------------------
```

Figure 10: Rapport d'un compte.

Conclusion.

Au terme de notre TP1 nous avons atteint les objectifs fixés et respectés les spécifications de notre application. Ainsi ce travail nous a permis de nous rappeler des concepts sur la programmation orientée objet ainsi que de l'approche MVC. Toutefois des efforts restent à fournir dans le sens de la réflexion orientée objet et de la maîtrise du langage java mais aussi de la mise en application des bonnes pratiques de programmation. Nous retenons en conclusion que ce TP a été un excellent exercice pour l'assimilation des concepts de base du cours de génie logiciel.

Lien gitHub

https://github.com/azy64/gestionBancaire/tree/master/Banque/src

ANNEXES

Annexe1 : Code du contrôleur

```
package controller.banque.com;

import Manager.models.com.ManagerCaissiers;
import Manager.models.com.ManagerCategories;
import Manager.models.com.ManagerClients;
import Manager.models.com.ManagerComptes;
import Manager.models.com.ManagerTransac;
import Manager.models.com.ManagerTypeTransac;
import bnk.models.com.BankTransaction;
import bnk.models.com.Caissiers;
import bnk.models.com.CategorieCompte;
import bnk.models.com.Clients;
import bnk.models.com.Comptes;
import bnk.models.com.TypeTransac;
import java.util.Date;
import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;
import javax.persistence.EntityTransaction;
public class BanqueController {
    private final ManagerTransac mnT=new ManagerTransac();
    private final ManagerTypeTransac managerType=new
ManagerTypeTransac();
    private final ManagerComptes managerCompte=new ManagerComptes();

   public void traitement(int choix,Scanner sc){
      switch(choix){
         case 1:/*****ici choix==1 pour dire qu'on crée le compte client**/
            System.out.println("vous avez choisi le 1 pour la création de compte:");
            String num;
            /***on recupere le numero du client pour verifier si il existe dans notre
systeme**/
            num=(String)Input("string", "Veuillez saisir votre numero de
téléphone:");
            ManagerClients mn=new ManagerClients();
            List cls=mn.fetchBy("numberPhone", num.trim());
            int response;
            double mnt;
            Comptes cmp=new Comptes();
            String req="";
            EntityTransaction tx=mn.getEm().getTransaction();
            /**si la liste est vide , alors le client n'est pas enregistré dans le
systeme**/
            /*
```

```
            *on va créer le client et son compte dans les lignes qui suivent
            */
            if(cls.isEmpty()){
               System.out.println("creation d'un nouveau Client...");
               String ch;
               ch=(String)Input("string", "Entrez vos informations comme suit: nom
pos-tnom prenom sexe(M ou F) date de naissance");
               Clients cl=new Clients();
               cl.setNumberPhone(num);
               String[]chs;
               if(!ch.isEmpty()){
                  try{
                     chs=ch.split(" ");
                  cl.setNom(chs[0]);
                  cl.setPostnom(chs[1]);
                  cl.setPrenom(chs[2]);
                  cl.setSexe(chs[3]);
                  cl.setBirthDay(new Date(chs[4]));
                  mn.creer(cl);
                  cmp.setClient(cl);
                  cmp.setDateCreation(new Date());
                  }
                  catch(Exception in){
                     System.out.println("----------------------------restart  Operation-------
----------------------------");
                     menu(sc);
                  }

               }


               mnt=(double)Input("double","Veuillez entrer un montant initial pour
le compte, ex:0.00,100.00 ...");

               ManagerComptes mnC=new ManagerComptes();
               cmp.setSold(mnt);
               cmp.setNumCompte(mnC.generateNumCompte());
               response=(int)Input("int","Veuillez choisir le type de compte:\n
1.PREMIUM avec 0.5 TAUX d'interet \n 2.STANDARD avec 0.3 TAUX
d'interet");

               ManagerCategories mnCat=new ManagerCategories();
               CategorieCompte cat=mnCat.fetchById(response);
               cmp.setCategorie(cat);
               req=(String)Input("string","Veuillez entrer votre matricule Agent:");
               ManagerCaissiers mnCaissier=new ManagerCaissiers();
               List<Caissiers> cc=mnCaissier.fetchBy("matricule", req.trim());
               if(!cc.isEmpty())
```

```
                    cmp.setGestionnaire(cc.get(0));
                else{
                    System.out.println("Matricule de l'agent n'existe pas, vous devez recommencer la procedure!");
                    menu(sc);
                }

                   // mnC.getEm().persist(cl);
                    mnC.creer(cmp);
                    System.out.println("le compte :"+cmp.getNumCompte()+" avec un solde de "+cmp.getSold()+"\n dont le taux d'interet:"+cmp.getCategorie().getTaux()+" de\n Mr "+cmp.getClient().getNom()+" a été crée avec succès...");

                    /*tx.rollback();
                    System.out.println("le compte Client n°"+cmp.getNumCompte()+" pour le Mr "+cmp.getClient().getNom()+" n'a pas été crée! ");*/

                System.out.println("----------fin de la creation----------");
                menu(sc);

            }
            /** si le client existe alors le traitement suivant***/
            /*
            *qui va consister seulement à créer son compte
            */
            else{
                Clients client=(Clients)cls.get(0);
                cmp.setClient(client);
                cmp.setDateCreation(new Date());
                System.out.println("----------le client existe----------");
                mnt=(double)Input("double","Veuillez entrer un montant initial pour le compte, ex:0.00,100.00 ...");

                ManagerComptes mnC=new ManagerComptes();
                cmp.setSold(mnt);
                cmp.setNumCompte(mnC.generateNumCompte());
                response=(int)Input("int","Veuillez choisir le type de compte:\n 1.PREMIUM avec 0.5 TAUX d'interet \n 2.STANDARD avec 0.3 TAUX d'interet");
                ManagerCategories mnCat=new ManagerCategories();
                CategorieCompte cat=mnCat.fetchById(response);
                cmp.setCategorie(cat);
                req=(String)Input("string","Veuillez entrer votre matricule Agent:");
                ManagerCaissiers mnCaissier=new ManagerCaissiers();
                List<Caissiers> cc=mnCaissier.fetchBy("matricule", req);
```

```
                if(!cc.isEmpty())
                    cmp.setGestionnaire(cc.get(0));
                else{
                    System.out.println("Matricule de l'agent n'existe pas, vous devez recommencer la procedure!");
                    menu(sc);
                }

                mnC.creer(cmp);
                System.out.println("le compte :"+cmp.getNumCompte()+" avec un solde de "+cmp.getSold()+"\n dont le taux d'interet:"+cmp.getCategorie().getTaux()+" de\n Mr "+cmp.getClient().getNom()+" a été crée avec succès...");

                System.out.println("----------fin de la creation----------");
                menu(sc);
            }

            System.out.println("pourquoi?");
            break;
        case 2:
            System.out.println("vous avez choisi le 2 qui est la consultation d'un compte");
            this.consulter();
            this.menu(sc);
            //System.out.println("pourquoi?");
            break;
        case 3:
            System.out.println("vous avez choisi le 3");
            this.debiter();
            this.menu(sc);
            break;
        case 4:
            System.out.println("vous avez choisi le 4");
            this.crediter();
            this.menu(sc);
            break;
        case 5:
            System.out.println("vous avez choisi le 5");
            this.Rechercher();
            this.menu(sc);
            break;
        case 6:
            System.out.println("vous avez choisi le 6");
            this.calculInteret();
            menu(sc);
            break;
```

```
        case 7:
            System.out.println("vous avez choisi le 7");
            this.rapport();
            menu(sc);
            break;

        default:
            System.out.println("vous etes en dessous de la moyenne mentale!");

    }

}
public Object Input(String type,String text){
    Scanner sc=new Scanner(System.in);
    int result;
    Object ob=null;
    if(type.equals("int")){
        System.out.println(text);
            //int result;
        try{
            result=sc.nextInt();
        }
        catch(Exception ex){
            System.out.println(text);
            result=sc.nextInt();
        }
        ob= result;
    }
    else if(type.equals("double")){
        System.out.println(text);
        double result1;
        try{
            result1=sc.nextDouble();
        }
        catch(Exception ex){
            System.out.println(text);
            result1=sc.nextDouble();
        }
        ob= result1;
    }
    else if(type.equals("string")){
        System.out.println(text);
        String result2;
        try{
            result2=sc.nextLine();
        }
        catch(Exception ex){
            System.out.println(text);
```

```
            result2=sc.nextLine();
        }
        ob= result2;
    }
    else if(type.equals("long")){
        System.out.println(text);
        long result2;
        try{
            result2=sc.nextLong();
        }
        catch(Exception ex){
            System.out.println(text);
            result2=sc.nextLong();
        }
        ob= result2;
    }
    else{

    }

   return ob;
  }
  public void consulter(/*Scanner sc*/){
    long numCmp=(long)Input("long", "Entrer le numero de compte à consulter");
    ManagerComptes managerCompte=new ManagerComptes();
    List<Comptes> comptes=managerCompte.fetchBy("numCompte", numCmp);
    /*
    *recupere le compte correspondant au numero de compte entré
    */
    if(!comptes.isEmpty()){
        Comptes cmp=comptes.get(0);
        double t=cmp.getCategorie().getTaux();
        System.out.println("Numero de Compte:"+cmp.getNumCompte()+"\n Type
de compte:"+cmp.getCategorie().getLibele()+"\n Taux d'interet:"+t+"\n
Solde:"+cmp.getSold());
        System.out.println("Date de creation:"+cmp.getDateCreation()+"");
        System.out.println("Titulaire de Compte:"+cmp.getClient().getNom()+"
"+cmp.getClient().getPrenom()+" "+cmp.getClient().getPostnom());
        System.out.println("Numero de
Telephone:"+cmp.getClient().getNumberPhone());
        BankTransaction tr=new BankTransaction();
        tr.setCleint(cmp.getClient());
        tr.setCompte(cmp);
        tr.setDateTransac(new Date());
        tr.setGestionnaire(cmp.getGestionnaire());
        tr.setMontant(0.00);
        TypeTransac tp= managerType.fetchById(2);
        tr.setTypeTransaction(tp);
```

```
            mnT.creer(tr);
            System.out.println("----------------------------------------Fin consultation----------
-----------------------------------");
        }
        else{
            System.out.println("Le Numero de entré n'exite pas!!!");
            System.out.println("----------------------------------------Fin consultation----------
-----------------------------------");
        }
    }
    public void debiter(){
        double numCmp=(double)Input("double", "Entrer le numero de compte à
Debiter");
        ManagerComptes managerCompte=new ManagerComptes();
        List<Comptes> comptes=managerCompte.fetchBy("numCompte", numCmp);
        if(!comptes.isEmpty()){
            Comptes cmp=comptes.get(0);
            /*double t=cmp.getCategorie().getTaux();
            System.out.println("Numero de Compte:"+cmp.getNumCompte()+"\n Type
de compte:"+cmp.getCategorie().getLibele()+"\n Taux d'interet:"+t+"\n
Solde:"+cmp.getSold());
            System.out.println("Date de creation:"+cmp.getDateCreation()+"");
            System.out.println("Titulaire de Compte:"+cmp.getClient().getNom()+"
"+cmp.getClient().getPrenom()+" "+cmp.getClient().getPostnom());
            System.out.println("Numero de
Telephone:"+cmp.getClient().getNumberPhone());*/
            double montant=(double)Input("double","Entrez le montant à debiter:");
            if(cmp.getSold()>montant)
                cmp.setSold(cmp.getSold()-montant);
            managerCompte.update(cmp);
            System.out.println("Le compte "+cmp.getNumCompte()+" a été debité de
"+montant+"\n d'ou le nouveau solde est:"+cmp.getSold());
            BankTransaction tr=new BankTransaction();
            tr.setCleint(cmp.getClient());
            tr.setCompte(cmp);
            tr.setDateTransac(new Date());
            tr.setGestionnaire(cmp.getGestionnaire());
            tr.setMontant(montant);
            TypeTransac tp= managerType.fetchById(3);
            tr.setTypeTransaction(tp);
            mnT.creer(tr);
            System.out.println("----------------------------------------Fin consultation----------
-----------------------------------");
        }
        else{
            System.out.println("Le Numero de compte entré n'exite pas!!!");
            System.out.println("----------------------------------------Fin consultation----------
-----------------------------------");
```

```
    }
  }
  public void crediter(){
    double numCmp=(double)Input("double", "Entrer le numero de compte à
Créditer");
    ManagerComptes managerCompte=new ManagerComptes();
    List<Comptes> comptes=managerCompte.fetchBy("numCompte", numCmp);
    if(!comptes.isEmpty()){
      Comptes cmp=comptes.get(0);
      /*double t=cmp.getCategorie().getTaux();
      System.out.println("Numero de Compte:"+cmp.getNumCompte()+"\n Type
de compte:"+cmp.getCategorie().getLibele()+"\n Taux d'interet:"+t+"\n
Solde:"+cmp.getSold());
      System.out.println("Date de creation:"+cmp.getDateCreation()+"");
      System.out.println("Titulaire de Compte:"+cmp.getClient().getNom()+"
"+cmp.getClient().getPrenom()+" "+cmp.getClient().getPostnom());
      System.out.println("Numero de
Telephone:"+cmp.getClient().getNumberPhone());*/
      double montant=(double)Input("double","Entrez le montant à créditer:");
      double avant=cmp.getSold();
      if(montant>0)
        cmp.setSold(cmp.getSold()+montant);
      managerCompte.update(cmp);
      System.out.println("Le compte "+cmp.getNumCompte()+" avec un
montantde "+avant+" a été crédité de "+montant+"\n d'ou le nouveau solde
est:"+cmp.getSold());
      BankTransaction tr=new BankTransaction();
      tr.setCleint(cmp.getClient());
      tr.setCompte(cmp);
      tr.setDateTransac(new Date());
      tr.setGestionnaire(cmp.getGestionnaire());
      tr.setMontant(montant);
      TypeTransac tp= managerType.fetchById(4);
      tr.setTypeTransaction(tp);
      mnT.creer(tr);
      System.out.println("----------Fin consultation----------");
    }
    else{
      System.out.println("Le Numero de compte entré n'exite pas!!!");
      System.out.println("----------Fin consultation----------");
    }
  }
  public void Rechercher(){
    int idcl=(int)Input("int","Entrer le numero d'identification du client à
rechercher");
    List <Comptes>cmps=managerCompte.fetchAll();
```

```
        List<Comptes> cmpsl=new LinkedList();
        if(!cmps.isEmpty()){
            for(int i=0;i<cmps.size();i++){
                if(cmps.get(i).getClient().getId().equals(idcl))
                    //cmps.remove(i);
                    cmpsl.add(cmps.get(i));
            }
            System.out.println("voici le nombre de compte associé au client "+idcl+" : "+cmpsl.size());
            System.out.println("Mm/Mr "+cmps.get(0).getClient().getNom()+" "+cmps.get(0).getClient().getPostnom()+"  "+cmps.get(0).getClient().getPrenom()+" N°"+cmps.get(0).getClient().getId()+" possede le(s) compte(s) suivant:");
            cmpsl.forEach((cmp) -> {
                System.out.println("le "+cmp.getNumCompte()+" a un solde de "+cmp.getSold()+" qui un compte "+cmp.getCategorie().getLibele()+" avec un Taux d'interet de "+cmp.getCategorie().getTaux()+"--- "+cmp.getClient().getId());
            });
        }
        else{
            System.out.println("Cet identifiant n'existe pas chez nous");
        }
    }
    public void calculInteret(){

System.out.println("**********************************************************");
        System.out.println("*****CALCUL DE L'INTERET POUR CHAQUE COMPTE DE LA BANQUE*****");

System.out.println("**********************************************************");
        List <Comptes>cmps=managerCompte.fetchAll();
        cmps.forEach((cmp)->{
            double taux=(cmp.getCategorie().getTaux()*cmp.getSold());
            double sommeTotal=cmp.getSold()+taux;
            System.out.println("Le compte "+cmp.getNumCompte()+" appartenant à "+cmp.getClient().getNom()+" "+cmp.getClient().getPrenom()
            +" dont le solde est de "+ cmp.getSold()+" de Type "+cmp.getCategorie().getLibele()
            +" de Taux d'interet "+cmp.getCategorie().getTaux());
            System.out.println("\t-àprès calcul de l'interet de "+taux+", Le solde Total est de:"+sommeTotal);

        });
        System.out.println("-----------------fin du calcul de l'interet-----------------");
    }
    public void rapport(){
        String str=(String)Input("string","Entrer une liste des comptes separé par
```

```
des traits d'unions,ex:100-109-111 ou juste un compte");
        String[]tr;
        String trait="-";
        if(str.contains(trait)){
            tr=str.split(trait);
        }
        else{
            tr=new String[1];
            tr[0]=str;
        }
        List<BankTransaction> transacs=mnT.fetchAll();
         for (String tt : tr) {
             String str1=tt;
             transacs.forEach((transac)->{
             long cc=Long.parseLong(str1);
             if(transac.getCompte().getNumCompte()==cc){
                 Comptes cmp=transac.getCompte();
                 System.out.println("Compte          "+cmp.getNumCompte()+"          solde
"+cmp.getSold()+" date de création "+cmp.getDateCreation() );
                 System.out.println("\t"+transac.getTypeTransaction().getLibele()+"--
"+transac.getMontant()+"--- "+transac.getDateTransac());
             }

             });
         }
        System.out.println("----------------------------fin Rapport-------------------------------
------------");
    }
    public void menu(Scanner sc){
        this.init();
        int choix=sc.nextInt();
        this.traitement(choix,sc);
    }
    public void init(){
        System.out.println("********************************");
        System.out.println("*GESTION DE COMPTE D'UN CLIENT*");
        System.out.println("********************************");
        System.out.println("Menu:");
        System.out.println("1.CREER UN COMPTE POUR LE CLIENT");
        System.out.println("2.CONSULTER UN COMPTE CLIENT");
        System.out.println("3.DEBITER UN COMPTE CLIENT");
        System.out.println("4.CREDITER UN COMPTE CLIENT");
        System.out.println("5.RECHERCHER UN COMPTE CLIENT");
        System.out.println("6.CALCULE DE L'INTERET POUR TOUS LES
COMPTES");
        System.out.println("7.RAPPORT SUR UN OU PLUSIEURS COMPTES");
        System.out.println("8.RETOUR AU MENU");
        System.out.println("ENTRER VOTRE CHOIX ...");
```

```
    }
}
```

Annexe 2 : Codes des modèles

Modele client.

```
package bnk.models.com;

import java.io.Serializable;
import java.util.Date;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity
public class Clients implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;
    @Column( name = "nom" )
    private String nom;
    @Column( name = "postnom" )
    private String postnom;
    @Column( name = "prenom" )
    private String prenom;
    @Column( name = "sexe" )
    private String sexe;
    @Column( name = "numberPhone" )
    private String numberPhone;
    @Column( name = "birthDay")
    @Temporal(TemporalType.DATE)
    private Date birthDay;



    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
```

```
        this.id = id;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not
set
        if (!(object instanceof Clients)) {
            return false;
        }
        Clients other = (Clients) object;
        if ((this.id == null && other.id != null) || (this.id != null
&& !this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "bnk.models.com.Clients[ id=" + id + " ]";
    }

    public String getNom() {
        return nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    public String getPostnom() {
        return postnom;
    }

    public void setPostnom(String postnom) {
        this.postnom = postnom;
    }

    public String getPrenom() {
        return prenom;
```

```
    }

    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }

    public String getSexe() {
        return sexe;
    }

    public void setSexe(String sexe) {
        this.sexe = sexe;
    }

    public String getNumberPhone() {
        return numberPhone;
    }

    public void setNumberPhone(String numberPhone) {
        this.numberPhone = numberPhone;
    }

    public Date getBirthDay() {
        return this.birthDay;
    }

    public void setBirthDay(Date birthDay) {
        this.birthDay = birthDay;
    }


}
```

Modele caissiers

```
package bnk.models.com;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class CategorieCompte implements Serializable {
```

```
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;
    @Column(name="libele")
    private String libele;
    @Column(name="taux")
    private double taux;
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not
set
        if (!(object instanceof CategorieCompte)) {
            return false;
        }
        CategorieCompte other = (CategorieCompte) object;
        if ((this.id == null && other.id != null) || (this.id != null
&& !this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    public String getLibele() {
        return libele;
    }

    public void setLibele(String libele) {
        this.libele = libele;
    }

    public double getTaux() {
        return taux;
```

```
    }

    public void setTaux(double taux) {
        this.taux = taux;
    }


    @Override
    public String toString() {
        return "bnk.models.com.CategorieCompte[ id=" + id + " ]";
    }

}
```

Modele Compte.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package bnk.models.com;

import java.io.Serializable;
import java.util.Date;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity
public class Comptes implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;
    @Column(name="sold")
    private double sold;
    @Column(name="dateCreation")
    @Temporal( TemporalType.DATE)
    private Date dateCreation;
    @Column(name="numCompte")
    private long numCompte;
```

```
    @ManyToOne(cascade = {CascadeType.MERGE})
    private Clients client;
    @ManyToOne/*(cascade = {CascadeType.PERSIST,CascadeType.MERGE})*/
    private CategorieCompte categorie;
    @ManyToOne(cascade = {CascadeType.MERGE})
    private Caissiers gestionnaire;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not
set
        if (!(object instanceof Comptes)) {
            return false;
        }
        Comptes other = (Comptes) object;
        if ((this.id == null && other.id != null) || (this.id != null
&& !this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    public double getSold() {
        return sold;
    }

    public void setSold(double sold) {
        this.sold = sold;
    }

    public Date getDateCreation() {
        return dateCreation;
    }
```

```
    public void setDateCreation(Date date_cr) {
        this.dateCreation = date_cr;
    }

    public long getNumCompte() {
        return numCompte;
    }

    public void setNumCompte(long numCompte) {
        this.numCompte = numCompte;
    }

    public Clients getClient() {
        return client;
    }

    public void setClient(Clients client) {
        this.client = client;
    }

    public CategorieCompte getCategorie() {
        return categorie;
    }

    public void setCategorie(CategorieCompte categorie) {
        this.categorie = categorie;
    }

    public Caissiers getGestionnaire() {
        return gestionnaire;
    }

    public void setGestionnaire(Caissiers gestionnaire) {
        this.gestionnaire = gestionnaire;
    }

    @Override
    public String toString() {
        return "bnk.models.com.Comptes[ id=" + id + " ]";
    }
    }
```

Modele BankTransaction.

```
package bnk.models.com;
```

```
import java.io.Serializable;
import java.util.Date;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity
public class BankTransaction implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;
    @Column(name="dateTransac")
    @Temporal(TemporalType.DATE)
    private Date dateTransac;
    @Column(name="montant")
    private double montant;
    @ManyToOne(cascade = CascadeType.MERGE)
    private Clients cleint;
    @ManyToOne(cascade = CascadeType.MERGE)
    private Comptes compte;
    @ManyToOne(cascade = CascadeType.MERGE)
    private Caissiers gestionnaire;
    @ManyToOne(cascade = CascadeType.MERGE)
    private TypeTransac typeTransaction;
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }
```

```
    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof BankTransaction)) {
            return false;
        }
        BankTransaction other = (BankTransaction) object;
        if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    public Date getDateTransac() {
        return dateTransac;
    }

    public void setDateTransac(Date dateTransac) {
        this.dateTransac = dateTransac;
    }

    public double getMontant() {
        return montant;
    }

    public void setMontant(double montant) {
        this.montant = montant;
    }

    public Clients getCleint() {
        return cleint;
    }

    public void setCleint(Clients cleint) {
        this.cleint = cleint;
    }

    public Comptes getCompte() {
        return compte;
    }

    public void setCompte(Comptes compte) {
        this.compte = compte;
    }

    public Caissiers getGestionnaire() {
```

```
        return gestionnaire;
    }

    public void setGestionnaire(Caissiers gestionnaire) {
        this.gestionnaire = gestionnaire;
    }

    public TypeTransac getTypeTransaction() {
        return typeTransaction;
    }

    public void setTypeTransaction(TypeTransac typeTransaction) {
        this.typeTransaction = typeTransaction;
    }


    @Override
    public String toString() {
        return "bnk.models.com.BankTransaction[ id=" + id + " ]";
    }

}
```

Modele TypeTransac

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package bnk.models.com;
import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class TypeTransac implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;
    @Column(name="libele")
    private String libele;
```

```
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof TypeTransac)) {
            return false;
        }
        TypeTransac other = (TypeTransac) object;
        if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    public String getLibele() {
        return libele;
    }

    public void setLibele(String libele) {
        this.libele = libele;
    }

    @Override
    public String toString() {
        return "bnk.models.com.TypeTransac[ id=" + id + " ]";
    }
    }
```

Annexe 3: Code des managers.

ManagerClient

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Manager.models.com;

import bnk.models.com.Caissiers;
import java.util.List;
import javax.persistence.EntityTransaction;
import javax.persistence.Query;


public class ManagerCaissiers extends ManagerModel<Caissiers>{

    private final String select_all="select u from Caissiers as u";
    private Query query;
    @Override
    public void creer(Caissiers ob) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        EntityTransaction tx=this.getEm().getTransaction();
        try{
            tx.begin();
            this.getEm().persist(ob);
            tx.commit();
            this.getEm().close();
            this.getFac().close();
        }
        catch(Exception e){
            tx.rollback();
            e.printStackTrace();
        }
    }

    @Override
    public Caissiers fetchById(int id) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
         return this.getEm().find(Caissiers.class, id);
    }

    @Override
    public List<Caissiers> fetchAll() {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        this.query=this.getEm().createQuery(select_all);
```

```
        return this.query.getResultList();
    }

    @Override
    public List<Caissiers> fetchBy(String attribut, Object ob) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        String req="select u from Caissiers as u where u."+attribut+"=:"+attribut;
        this.query=this.getEm().createQuery(req);
        this.query.setParameter(attribut, ob);
        return this.query.getResultList();
    }

    @Override
    public boolean update(Caissiers ob) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        EntityTransaction tx=this.getEm().getTransaction();
        tx.begin();
        try{
            this.getEm().persist(ob);
            tx.commit();
            return true;
        }
        catch(Exception ex){
            tx.rollback();
            return false;
        }
    }
    @Override
    public void delete(Caissiers ob) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        this.getEm().remove(ob);
    }
    }
```

ManagerComptes

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Manager.models.com;

import bnk.models.com.Comptes;
import java.util.List;
import javax.persistence.EntityTransaction;
```

```
import javax.persistence.Query;

public class ManagerComptes extends ManagerModel<Comptes>{
    private final String select_all="select u from Comptes as u";
   private Query query;

   @Override
   public void creer(Comptes ob) {
      //throw new UnsupportedOperationException("Not supported yet."); //To
change body of generated methods, choose Tools | Templates.
      EntityTransaction tx=this.getEm().getTransaction();
      try{
         tx.begin();
         this.getEm().persist(ob);
         tx.commit();
         this.getEm().close();
         this.getFac().close();
      }
      catch(Exception e){
         tx.rollback();
         e.printStackTrace();
      }
   }

   @Override
   public Comptes fetchById(int id) {
      //throw new UnsupportedOperationException("Not supported yet."); //To
change body of generated methods, choose Tools | Templates.
      return this.getEm().find(Comptes.class, id);
   }

   @Override
   public List<Comptes> fetchAll() {
      //throw new UnsupportedOperationException("Not supported yet."); //To
change body of generated methods, choose Tools | Templates.
      this.query=this.getEm().createQuery(select_all);
      return this.query.getResultList();
   }

   @Override
   public List<Comptes> fetchBy(String attribut, Object ob) {
      //throw new UnsupportedOperationException("Not supported yet."); //To
change body of generated methods, choose Tools | Templates.
      String req="select u from Comptes as u where u."+attribut+"=:"+attribut;
      this.query=this.getEm().createQuery(req);
      this.query.setParameter(attribut, ob);
      return this.query.getResultList();
   }
```

```
    @Override
    public boolean update(Comptes ob) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
         EntityTransaction tx=this.getEm().getTransaction();
        tx.begin();
        try{
            this.getEm().persist(ob);
            tx.commit();
            return true;
        }
        catch(Exception ex){
            tx.rollback();
            return false;
        }
    }

    @Override
    public void delete(Comptes ob) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        this.getEm().remove(ob);
    }

    public long generateNumCompte(){
        List<Comptes> l=this.fetchAll();
        if(!l.isEmpty()){
            long num=l.get(l.size()-1).getNumCompte();
            return num+1;
        }
        else
            return 100;
    }
}
```

Modele Categorie.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Manager.models.com;

import bnk.models.com.CategorieCompte;
import java.util.List;
```

```
import javax.persistence.EntityTransaction;
import javax.persistence.Query;

/**
 *
 * @author AZARIA
 */
public class ManagerCategories extends ManagerModel<CategorieCompte>{

    private final String select_all="select u from CategorieCompte as u";
    private Query query;
    @Override
    public void creer(CategorieCompte ob) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
         EntityTransaction tx=this.getEm().getTransaction();
        try{
            tx.begin();
            this.getEm().persist(ob);
            tx.commit();
            this.getEm().close();
            this.getFac().close();
        }
        catch(Exception e){
            tx.rollback();
            e.printStackTrace();
        }
    }

    @Override
    public CategorieCompte fetchById(int id) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
         return this.getEm().find(CategorieCompte.class, id);
    }

    @Override
    public List<CategorieCompte> fetchAll() {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        this.query=this.getEm().createQuery(select_all);
        return this.query.getResultList();
    }

    @Override
    public List<CategorieCompte> fetchBy(String attribut, Object ob) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
```

```
        String req="select u from CategorieCompte as u where u."+attribut+"=:"+attribut;
        this.query=this.getEm().createQuery(req);
        this.query.setParameter(attribut, ob);
        return this.query.getResultList();
    }

    @Override
    public boolean update(CategorieCompte ob) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        EntityTransaction tx=this.getEm().getTransaction();
        tx.begin();
        try{
            this.getEm().persist(ob);
            tx.commit();
            return true;
        }
        catch(Exception ex){
            tx.rollback();
            return false;
        }
    }

    @Override
    public void delete(CategorieCompte ob) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        this.getEm().remove(ob);
    }

}
```

Manager Model

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Manager.models.com;

import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 *
```

```
 * @author AZARIA
 */
public abstract class ManagerModel<T> {
    private final EntityManagerFactory fac=Persistence.createEntityManagerFactory("BanquePU");
    private final EntityManager em=fac.createEntityManager();

    public EntityManagerFactory getFac() {
        return fac;
    }

    public EntityManager getEm() {
        return em;
    }

    public abstract void creer(T ob);
    public abstract T fetchById(int id);
    public abstract List<T> fetchAll();
    public abstract List<T> fetchBy(String attribut,Object ob);
    public abstract boolean update(T ob);
    public abstract void delete(T ob);

}
```

Modele transaction

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Manager.models.com;

import bnk.models.com.BankTransaction;
import java.util.List;
import javax.persistence.EntityTransaction;
import javax.persistence.Query;

/**
 *

 * @author AZARIA

 */

public class ManagerTransac extends ManagerModel<BankTransaction>{

   private final String select_all="select u from BankTransaction as u";
```

```
private Query query;
@Override
public void creer(BankTransaction ob) {
    EntityTransaction tx=this.getEm().getTransaction();
    try{
        tx.begin();
        this.getEm().persist(ob);
        tx.commit();
    }
    catch(Exception e){
        //tx.rollback();
        e.printStackTrace();
    }
    //this.getEm().close();
    //this.getFac().close();
}
@Override
public BankTransaction fetchById(int id) {
    return this.getEm().find(BankTransaction.class, id);
}

@Override
public List<BankTransaction> fetchAll() {
    this.query=this.getEm().createQuery(select_all);
    return this.query.getResultList();
}
```

```
    @Override
    public List<BankTransaction> fetchBy(String attribut, Object ob) {
        String req="select u from BankTransaction as u where
u."+attribut+"=:"+attribut;
        this.query=this.getEm().createQuery(req);
        this.query.setParameter(attribut, ob);
        return this.query.getResultList();
    }
    @Override
    public boolean update(BankTransaction ob) {
        EntityTransaction tx=this.getEm().getTransaction();
        tx.begin();
        try{
            this.getEm().persist(ob);
            tx.commit();
            return true;
        }
        catch(Exception ex){
            tx.rollback();
            return false;
        }
    }
    @Override
    public void delete(BankTransaction ob) {
        this.getEm().remove(ob);
    }
}
```