Rapport de Projet JAVA
Gestion de comptes Bancaires

Résumé
Rapport de Projet JAVA sur la gestion de comptes bancaires dans le cadre de la licence Professionnelle ACSID du CNAM

SOMMAIRE

1 Introduction

Pour permettre à ses élèves de Licence Professionnel Analyste/Conception de Systèmes d'Informations et de Décisions de mettre le savoir en pratique, Mme Aubonnet nous a confiés la réalisation entière d'un projet de gestion de compte bancaire à réalisée en groupe de deux ou trois.

Ce projet à pour vocation d'appliquer les différentes connaissances acquises au cours de l'année pour réaliser l'analyse (MSI, BDC, ABD) et de concevoir un programme en JAVA.

2 Présentation du projet

Le projet Gestion des comptes bancaires doit permettre à un utilisateur client de la banque fictive BVN de posséder plusieurs types de comptes.

Les différentes possibilités de comptes sont :

- COMPTE COURANT
- COMPTE REMUNERE
  Possède un taux d'intérêt et une méthode de calcul des intérêts spécifique au titulaire
- COMPTE SECURISE
  Retrait seulement si le solde du compte est supérieur ou égal à 5% de la somme demandée
- COMPTE REMUNERE ET SECURISE
  Possède un taux d'intérêt et une méthode de calcul des intérêts spécifique au titulaire
  Retrait seulement si le solde du compte est supérieur ou égal à 5% de la somme demandée

Chacun de ses comptes possèdent son propre identifiant sous la forme d'un numéro et un solde. D'autre part, tout compte appartient à un titulaire pouvant le consulter, le crédité ou bien le débité.

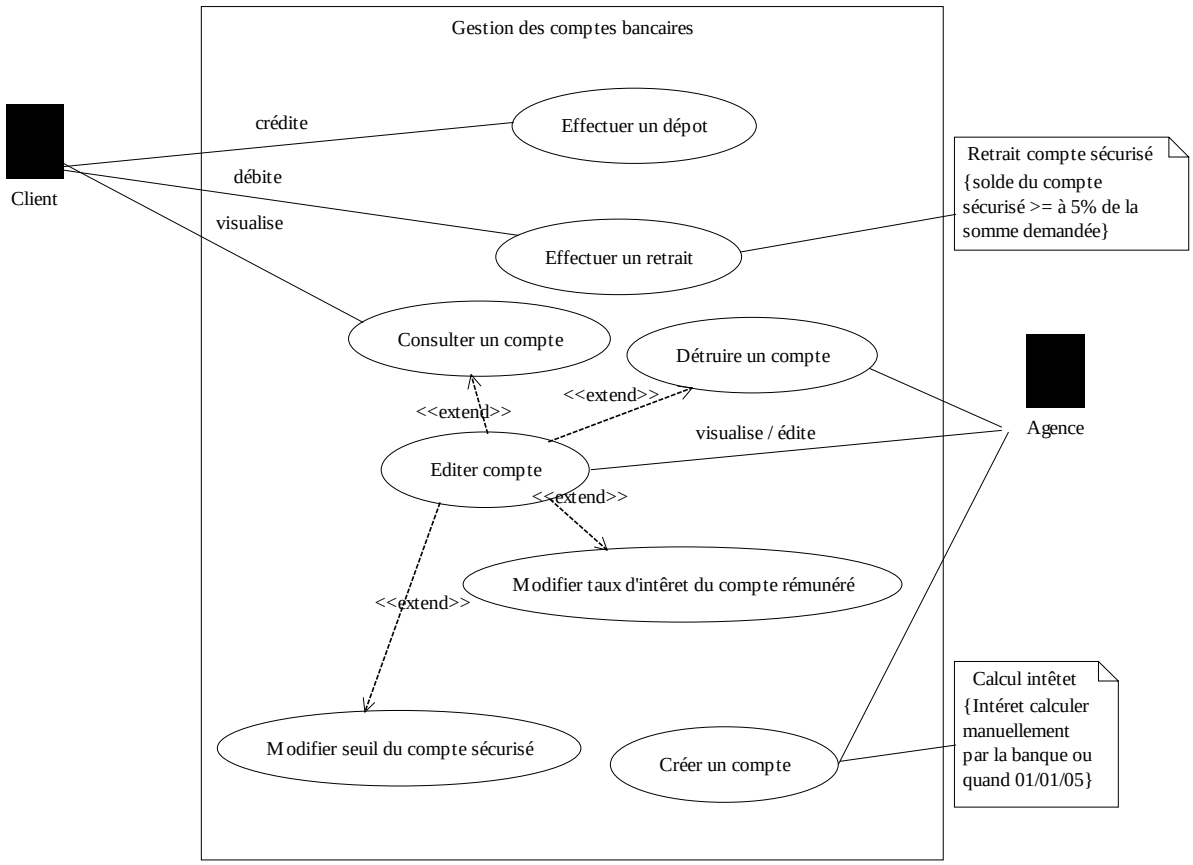
Le client effectue les opérations sur ses comptes à travers un distributeur automatique de billet DAB de son agence.

Une agence de la banque détient un certain nombre de comptes appartenant à ses clients. Elle a la possibilité de créer, de détruire ou de consulter un compte.
Da plus, l'agence doit être capable de calculer les intérêts des comptes rémunérés (*rémunéré et sécurisé compris).*

3 Analyse

3.1 Les Diagrammes

3.1.1 Diagramme des cas d'utilisations



3.1.2 Description des cas d'utilisations

Le diagramme d'utilisation général rédigé, à nous ensuite d'établir différents cas d'utilisation dont voici un exemple : le cas « Consulter un compte ».

PROJET : Gestion de Compte
Nom du cas d'utilisation : Consulter un compte
Référence : GDC01
Date : 23/01/2005
Résumé : Ce cas d'utilisation permet à n'importe quel client de l'application de consulter un compte dont il connaît le numéro à condition que celui-ci n'ait pas été clôturé.
Acteur déclencheur : Client
Acteur participant : Néant
Pré conditions : L'application est exécutée et l'écran d'accueil est affichée.

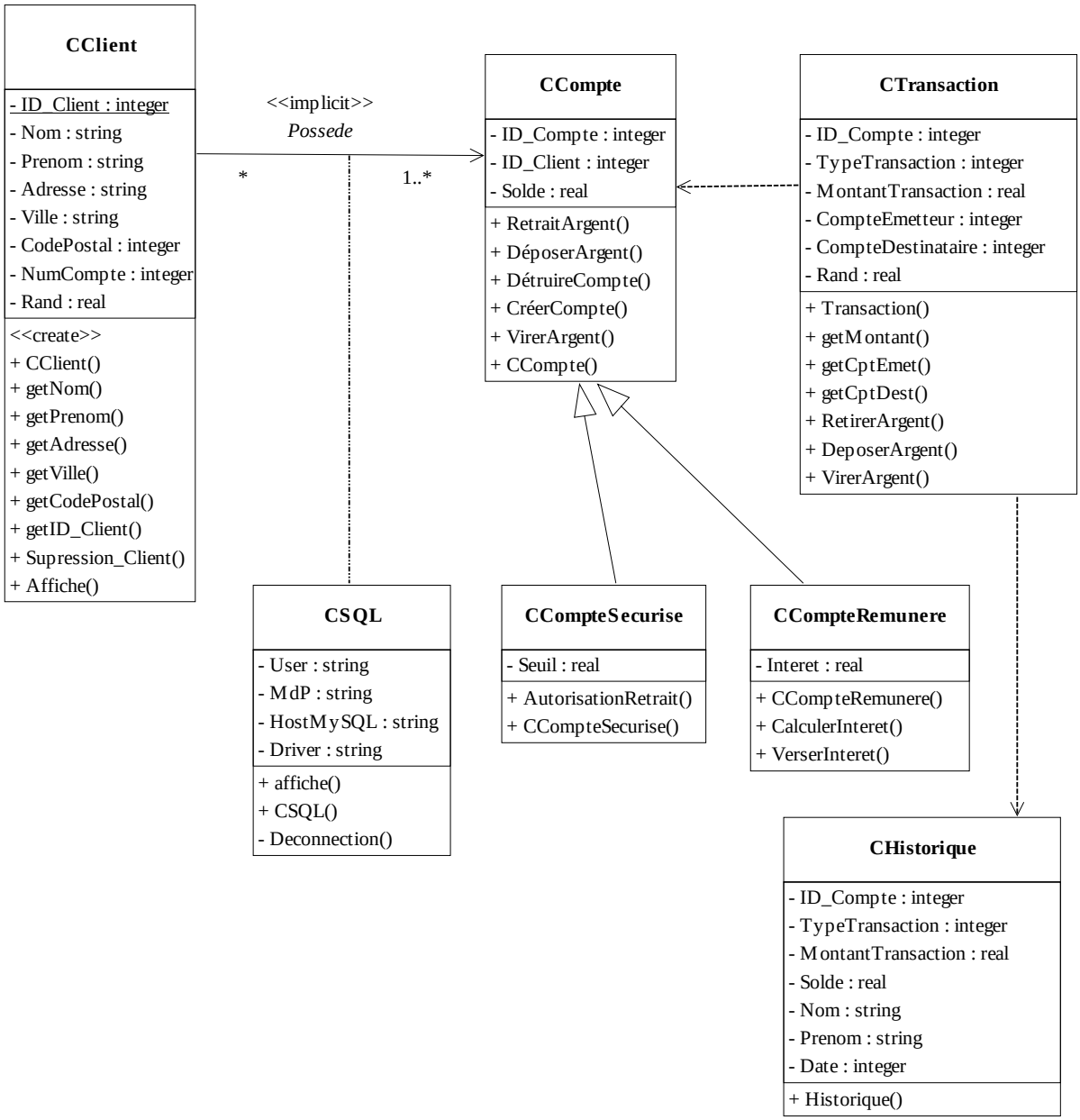
Description des enchaînements :
Scénario nominal GDC01-N
1. Choisir l'action « consulter un compte »
2. Saisir le numéro de compte
3. Vérification du numéro de compte
4. Affichage du solde et du libellé
5. L'écran d'invite est affiché

Enchaînements alternatifs : GDC01-A-01
Numéro de compte invalide
L'enchaînement GDC01-A-01 démarre au point 3 du scénario nominal.
4. L'utilisateur est informé que le numéro de compte est inexistant
5. L'écran d'invite est affiché

Enchaînements alternatifs : GDC01-A-02
Numéro de compte valide mais le compte est bloqué
L'enchaînement GDC01-A-02 démarre au point 3 du scénario nominal.
4. L'utilisateur est informé le compte a été clôturé et qu'il est désormais impossible de le consulter
5. L'écran d'invite est affiché

Enchaînements d'exceptions : GDC01-E-01
Aucun numéro de compte n'est saisi
L'enchaînement GDC01-E-01 démarre au point 2 du scénario nominal.
3. L'application invite l'utilisateur à saisir de nouveau un numéro de compte
4. L'écran d'invite est affiché

3.1.3 Diagramme de classe documentée



4 Conception

4.1 La Base de données SQL

4.1.1 Structure de la base de données

Base de données: `compte_bancaire`

```
CREATE DATABASE `compte_bancaire`;
USE compte_bancaire;
```

Structure de la table `client`

```
CREATE TABLE `client` (
  `ID_Client` int(10) NOT NULL auto_increment,
  `Nom` varchar(20) NOT NULL default '',
  `Prenom` varchar(20) NOT NULL default '',
  `Adresse` varchar(20) NOT NULL default '',
  `Ville` varchar(20) NOT NULL default '',
  `Numero_Compte` varchar(20) NOT NULL default '',
  PRIMARY KEY  (`ID_Client`)
) TYPE=MyISAM AUTO_INCREMENT=5460 ;
```

Structure de la table `compte`

```
CREATE TABLE `compte` (
  `ID_Compte` int(10) NOT NULL auto_increment,
  `Numero_Compte` int(10) NOT NULL default '0',
  `ID_Client` int(10) NOT NULL default '0',
  `Solde` float NOT NULL default '0',
  `Type` varchar(10) NOT NULL default '',
  `Date_Creation` date NOT NULL default '0000-00-00',
  PRIMARY KEY  (`ID_Compte`)
) TYPE=MyISAM AUTO_INCREMENT=99 ;
```

Structure de la table `transaction`

```
CREATE TABLE `transaction` (
  `ID_Transac` int(11) NOT NULL auto_increment,
  `Type_Transac` varchar(20) NOT NULL default '',
  `Montant_Transac` int(11) NOT NULL default '0',
  `CompteE` int(11) NOT NULL default '0',
  `CompteD` int(11) NOT NULL default '0',
  `Date_Transac` date NOT NULL default '0000-00-00',
  `Heure_Transac` time NOT NULL default '00:00:00',
  PRIMARY KEY  (`ID_Transac`)
) TYPE=MyISAM AUTO_INCREMENT=30022 ;
```

4.2 Exemples de traitements SQL

- Retrait:

```
insert into Transaction VALUES (77,'R',50,6021128,0,'2005-02-
15','19:32:58');
```

- Dépôt:

```
insert into Transaction VALUES (69,'D',30,0,6021128,'2005-02-
15','19:34:16') ;
```

4.3 Nos Choix

4.3.1 Nos logiciels utilisés

4.3.1.1 Eclipse

Eclipse (v2.1) est un AGL très répandue et réputée pour développer des applications JAVA. C'est un logiciel gratuit et ayant une communauté d'utilisateur très active sur les forums. Ainsi notre choix de développer sous Eclipse s'est imposé naturellement.

4.3.1.2 Jdk

Le JDK (Java Development Kit) **(v1.5)** indispensable à l'interprétation des applets ou des applications JAVA (*machine virtuel JAVA*).

4.3.1.3 EasyPhp 1.7

EasyPhp est un pack complet comportant un serveur MySQL et Apache (*compilée avec l'extension PHP 4.2*)

Sa facilité d'utilisation permet entre autres d'administrer sa base de données facilement, de stocker des pages Web pour y insérer des applets JAVA.

4.3.2 Nos choix techniques

4.3.2.1 Les Applets

Il s'agit de petites applications destinées à fonctionner sur un navigateur grâce à ce qu'on appel la machine virtuel Java qui s'installe sur la majorité des systèmes d'exploitation actuel.

Les applets peuvent servir à créer des animations, des figures, des jeux, des formulaires capables de réagir à des informations entrées par l'utilisateur, ou tout autre type d'effets interactifs sur une même page Web, au milieu d'éléments de texte et de graphismes.

4.3.2.1.1 La compilation d'un code source

Pour compiler un fichier source il suffit d'invoquer la commande *javac* avec le nom du fichier source avec son extension *.java*

```
javac NomFichier.java
```

Le nom du fichier doit correspondre au nom de la classe principale en respectant la casse même si le système d'exploitation n'y est pas sensible.
Suite à la compilation, le pseudo code Java est enregistré sous le nom *NomFichier.class*.

4.3.2.1.2 L'exécution d'une applet

Il suffit de créer une page HTML pouvant être très simple :

```
<HTML>
<TITLE> test applet Java </TITLE>
<BODY>
<APPLET code=« NomFichier.class » width=270 height=200>
</APPLET>
</BODY>
</HTML>
```

Exemple d'intégration d'un applet dans du code HTML

Il faut ensuite visualiser la page créée dans l'appletviewer ou dans un navigateur 32 bits compatible avec la version de Java dans laquelle l'applet est écrite.

4.3.3 Les Méthodes Utilisés

4.3.3.1 L'Héritage

L'héritage est un mécanisme qui facilite la réutilisation du code et la gestion de son évolution. Elle définit une relation entre deux classes :

- une classe mère ou super classe
- une classe fille ou sous classe qui hérite de sa classe mère

4.3.3.1.1 Le principe de l'héritage

Grace à l'heritage, les objets d'une classe fille ont accès aux données et aux méthodes de la classe parent et peuvent les étendre. Les sous classes peuvent redéfinir les variables et les méthodes héritées. Pour les variables, il suffit de les redéclarer sous le même nom avec un type différent. Les méthodes sont redéfinies avec le même nom, les mêmes types et le même nombre d'arguments, sinon il s'agit d'une surcharge.
L'héritage successif de classes permet de définir une hiérarchie de classe qui se compose de super classes et de sous classes. Une classe qui hérite d'une autre est une sous classe et celle dont elle hérite est une super classe. Une classe peut avoir plusieurs sous classes. Une classe ne peut avoir qu'une seule classe mère : il n'y a pas d'héritage multiple en Java.
Object est la classe parente de toutes les classes en Java. Toutes les variables et méthodes contenues dans Object sont accessibles à partir de n'importe quelle classe car par héritage successif toutes les classes héritent d'Object.

4.3.3.1.2 La mise en oeuvre de l'héritage

On utilise le mot clé extends pour indiquer qu'une classe hérite d'une autre. En l'absence de ce mot réservé associé à une classe, le compilateur considère la classe Object comme classe parent.

```
public class Applet_Agence_Creation_Client extends Applet { … }
```

Exemple

Pour invoquer une méthode d'une classe parent, il suffit d'indiquer la méthode préfixée par super. Pour appeler le constructeur de la classe parent il suffit d'écrire super(paramètres) avec les paramètres adéquats.

Le lien entre une classe fille et une classe parent est géré par le langage : une évolution des règles de gestion de la classe parent conduit à modifier automatiquement la classe fille dès que cette dernière est recompilée.

En Java, il est obligatoire dans un constructeur d'une classe fille de faire appel explicitement ou implicitement au constructeur de la classe mère.

4.3.3.1.3 L'accès aux propriétés héritées

Les variables et méthodes définies avec le modificateur d'accès public restent publiques à travers l'héritage et toutes les autres classes.

Une variable d'instance définie avec le modificateur private est bien héritée mais elle n'est pas accessible directement mais via les méthodes héritées.

Si l'on veut conserver pour une variable d'instance une protection semblable à celle assurée par le modificateur private, il faut utiliser le modificateur protected. La variable ainsi définie sera hérité dans toutes les classes descendantes qui pourront y accéder librement mais ne sera pas accessible hors de ces classes directement.

4.3.3.1.4 La redéfinition d'une méthode héritée

La redéfinition d'une méthode héritée doit impérativement conserver la déclaration de la méthode parent (type et nombre de paramètres, la valeur de retour et les exceptions propagées doivent être identique).

Si la signature de la méthode change, ce n'est plus une redéfinition mais une surcharge. Cette nouvelle méthode n'est pas héritée : la classe mere ne possède pas de méthode possédant cette signature.

4.3.3.2 Le polymorphisme

Le polymorphisme est la capacité, pour un même message de correspondre à plusieurs formes de traitements selon l'objet auquel ce message est adressé. La gestion du polymorphisme est assurée par la machine virtuelle dynamiquement à l'exécution.

4.3.3.2.1 Le transtypage induit par l'héritage facilitent le polymorphisme

L'héritage définit un cast implicite de la classe fille vers la classe mere : on peut affecter à une référence d'une classe n'importe quel objet d'une de ses sous classes.

```
Personne p = new Personne («Dupond», «Jean»);
  Employe e = new Employe(«Durand», «Julien», 10000);
  p = e ; // ok : Employe est une sous classe de Personne
  Objet obj;
```

```
      obj = e ; // ok : Employe herite de Personne qui elle même
                        hérite de Object
```

Exemple : la classe Employe hérite de la classe Personne

Il est possible d'écrire le code suivant si Employe hérite de Personne

```
  Personne[] tab = new Personne[10];
      tab[0]:= new Personne( «Dupond»,«Jean»);
      tab[1]:= new Employe(«Durand», «Julien», 10000);
```

Exemple

Il est possible de surcharger une méthode héritée : la forme de la méthode à exécuter est choisie en fonction des paramètres associés à l'appel.

Compte tenu du principe de l'héritage, le temps d'exécution du programme et la taille du code source et de l'exécutable augmentent.

4.3.3.3 Exception

Pour traiter les erreurs, Java propose un mécanisme qualifié d'exception, consistant à effectuer les instructions dans un bloc d'essai (le bloc *try*) qui surveille les instructions. Lors de l'apparition d'une erreur, celle-ci est lancée dans un bloc de traitement d'erreur (le bloc *catch*, appelé *handler d'exception*) sous forme d'un objet appelé **Exception**.

La gestion des exceptions avec Java consiste à définir au sein d'une méthode une clause "*try{ }*" contenant les instructions qui risquent de provoquer une exception et de la faire suivre immédiatement par une clause "*catch( ){ }*" contenant comme paramètre l'exception attendue précédée de son type (pour une erreur mathématique ce sera *ArithmeticException*) et dont le contenu sera une liste d'instruction à exécuter lorsque l'exception se produira.

```
class Nom_de_la_classe {
   public static void main(String[] args) {
   // Instructions inoffensives (affectations, ...);
   try {
   // Instructions susceptibles de provoquer des erreurs;
   }
   catch (TypeException e) {
   // Instructions de traitement de l'erreur;
   }
   // Instructions si aucune erreur est apparue;
   }
}
```

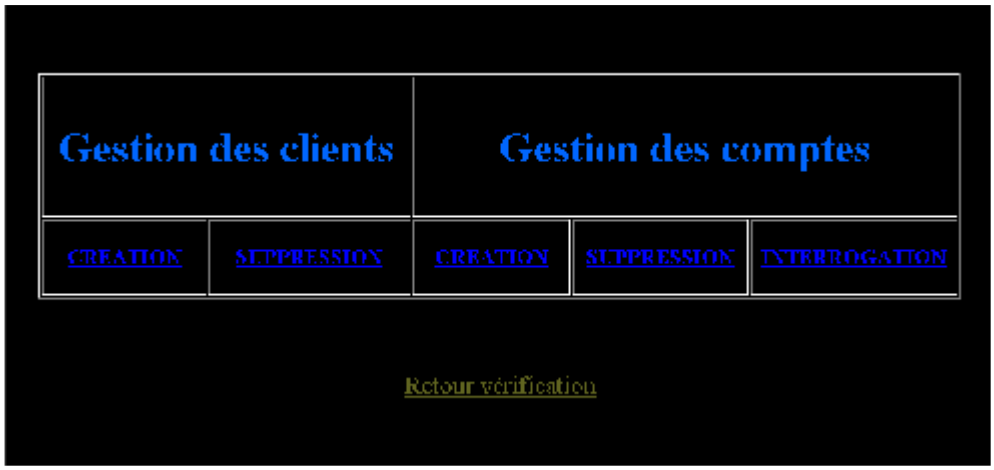
Syntaxe type d'une classe gérant des exceptions

4.4 Les modes de la banque virtuelle

4.4.1 L'Accueil

- Index.html → Accueil Banque BVN contient des liens (bouton swf) vers:
- Accueil_Client.htm → Accueil Client
- Verification_employe.htm → Identification employé agence
- Accueil_Agence.php → Accueil Agence

4.4.2 Le mode Agence



Page PHP contenant des liens simples vers:

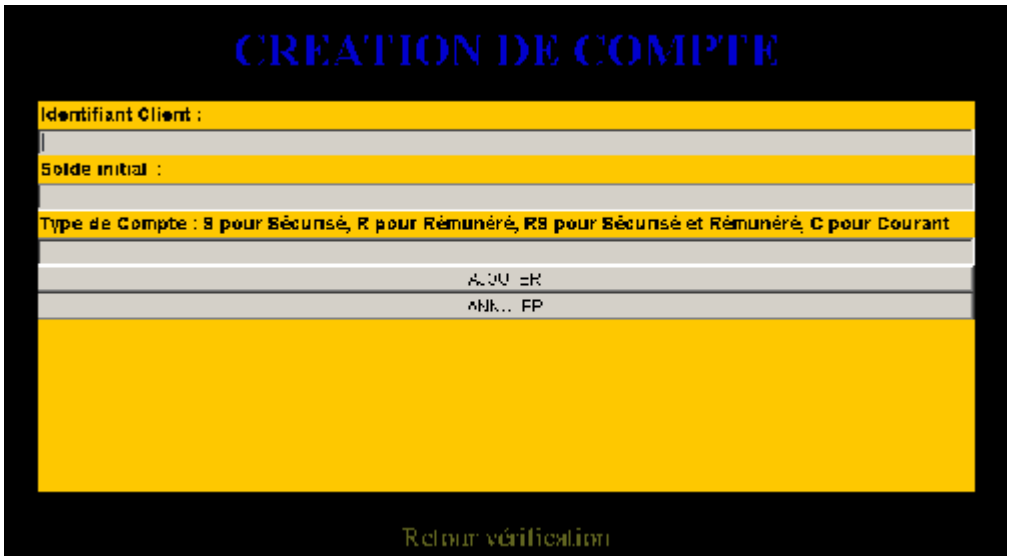
- Creation_Client.htm → Création de client → Applet_Agence_Creation_Client.java
- Suppression_Client.htm → Suppression de client → Applet_Agence_Suppression_Client.java
- Creation_Compte.htm → Création de compte → Applet_Agence_Creation_Compte.java
- Interroger_Compte.htm → Interrogation de compte → Applet_Agence_Interroger_Compte.java
- Suppression_Compte.htm→ Suppression de compte→Applet_Agence_Suppression_Compte.java

4.4.3 Le mode Client

Page Html contenant :

- Des boutons Swf (flash) ayant pour lien :
  - Retrait_Argent.htm → Retrait d'argent → Applet_Client_Retrait_Argent.java
  - Depot_Argent.htm → Dépôt d'argent → Applet_Client_Depot_Argent.java
  - Virement_Argent.htm → Virement d'argent → Applet_Client_Virement_Argent.java
  - Consul_Compte.htm → Consultation du compte → Applet_Client_Consultation_Compte.java

4.5 Les Comptes



4.5.1 Compte Courant

Le compte courant ne dispose pas chez nous (notre banque virtuelle) de rémunération, ni de sécurisation possible.
Tout retrait est possible.

4.5.2 Compte Rémunéré

Le compte rémunéré fait bénéficier au client d'un intérêt sur son solde calculé en fonction de la date de création et du solde qu'il dispose.
Le solde est dans notre banque augmenté de 1.5% chaque année pour les soldes compris entre 10€ (minimum) et 1500€, au dessus de il est augmenté de 1% chaque année au mois de décembre.

4.5.3 Compte Rémunéré et sécurisé

Lorsqu'un compte est rémunéré et sécurisé, le client voulant faire un retrait ne peut retiré une somme que si et seulement si son solde est supérieur ou égal à la somme demandée.

De plus il est rémunéré de la même manière que le compte rémunéré simple.

5 Intégration

5.1 Contenu du fichier Projet_Java.zip

- Dossier ‘www’ : à mettre ds Easyphp. Il contient les pages html ainsi que les classes Java.
- Dossier ‘Sources_Classes’ : contient les sources des classes Java.
- Dossier ‘A installer’ : contient ce qu’il faut installer en plus d’Eclipse et du JDK.

5.2 Procédure de déploiement

- Installer le JDK SunSystem.
- Installer EasyPHP (de préférence C:\ProgramFiles\EasyPHP\)
- Décompresser le ficher zip et copier le fichier mysql-connector-java-3.1.6-bin.jar
- dans les fichiers qui vont bien.
- Copier le répertoire ‘www’ dans le répertoire de EasyPHP.
- Lancer EasyPHP.
- Lancer Projet_Java.html.

5.3 A installer

5.3.1 EasyPHP

Lancez l’exécutable et suivre les procédures.
Après installation, lancez ‘mysqlmyadmin’ administrateur de la base de données MySQL et importer le fichier texte ‘Script_SQL.txt’, il créera la base de données et les tables adéquates.

5.3.2 mysql-connector-java-3.1.6.zip

Ce fichier va servir à la connexion à la base de données MySQL à partir de JAVA en utilisant le JDBC.
Il faut le décompresser et récupérer le fichier ***mysql-connector-java-3.1.6-bin.jar*** et le mettre dans plusieurs répertoires. Ceux des libraires de Java, d’Eclipse. (Eclipse 2.1 avec EasyJSP)

Java : C:\Program Files\Java\jdk1.5.0_01\lib\ C:\Program Files\Java\jre1.5.0_01\lib\ext Eclipse: C:\EASYJSP\Eclipse2.1\jre\lib\ext C:\EASYJSP\Eclipse2.1\lib

6 Conclusion

Le projet de Gestion de comptes bancaires nous a permit de mettre en pratique nos connaissances de JAVA. Nous avons ainsi acquis une meilleure appréciation de ce langage Objet. En effet, on a pu manier les différents concepts de la programmation POO en utilisant le polymorphisme, l'héritage, la surcharge et la généricité.

De plus, ce projet nous a permis d'exploiter l'une des spécificités de JAVA, les applets, et a permis de mettre en avant le fort potentiel de ce langage.

Il était aussi très intéressant d'exploiter les connaissances acquises dans nos autre cours pour effectuer l'analyse et la mise en place d'une base de données au travers du langage SQL par JDBC.

Le travail en équipe a été mis en avant pour se répartir les taches et effectuer des choix. De cette façon, on a pu organiser des réunions, et apprendre à gérer le planning au sein du projet.