

Rapport

I-Introduction :

Il s'agit de réaliser une application de gestion des comptes bancaires qui permet d'évoluer le système de gestion de la relation client dans le secteur bancaire. En effet, elle facilite la communication des clients avec le système bancaire. De plus, elle contribue à l'optimisation du service bancaire qui permet d'acquérir de nouveaux clients.

- **Problématique :**

Dans le système bancaire, pour que un client peut accéder à son compte bancaire, il doit aller à la banque et attend son tour pour qu'il puisse bénéficier du service bancaire. De plus, le service bancaire a des difficultés à servir plusieurs clients chaque jour. C'est pour cela, on a pensé de créer des interfaces bancaires qui cherche à satisfaire les clients en optimisant ce processus et aussi la banque en acquérant de nouveaux clients.

- **Objectif :**

Le but de cette application est :

- Optimiser le service client des banques.
- Facilité l'accès du client à son compte bancaire et effectuer les opérations disponibles.
- Améliorer les systèmes bancaires, en servant plusieurs clients au même temps.

II-Analyse et conception :

Pour gérer les comptes bancaires, on a codé plusieurs interfaces en utilisant les classes.

La classe Compte :

Pour gérer des comptes bancaires, j'ai commencé par créer la classe Compte. Chaque compte est associé à une personne (**nom**) titulaire du compte et dispose d'un **solde**. Il possède, en outre, un **numéro** unique qui lui est attribué automatiquement dès l'ouverture du compte et ne peut être modifié par la suite. Pour garantir l'intégrité des données des comptes nous mettons en œuvre le concept d'encapsulation en masquant les données.

-Les attributs privées :

- numéro: int
- nom : String
- solde: double
- operations: liste des opérations

-nbComptes : int statique

-Les constructeurs :

Compte(nom, solde) : le nom et solde est saisi au clavier.

-Les méthodes d'instance :

void deposter(montant) : déposer un montant donné dans le compte

void retirer(montant) : retirer un montant donné du compte

void virer(montant, compte) : faire un virement vers un autre compte

-Les méthodes usuelles :

String toString() : Renvoie la représentation du compte sous forme d'une chaîne

boolean equals(Object) : Deux comptes égaux, s'ils ont le même propriétaire et le même numéro

getters/setters : setSolde(solde) lève une exception si solde < 0

La classe SoldeInsuffisantException :

Le retrait d'argent ou le virement n'est pas toujours possible. Il faut gérer également les exceptions. C'est pour cela, j'ai créé la classe SoldeInsuffisantException.

La classe Operation :

Pour la classe Operation. Chaque opération bancaire (dépôt, retrait ou virement) effectuée sur un compte est une information structurée qui peut être modélisée à l'aide d'un objet Java. Elle est effectuée à une date donnée et correspond à un type particulier (**DEPOT, RETRAIT, VIREMENT**) et concerne un certain montant. Chaque compte maintient l'ensemble de ses opérations dans une liste

-Les attributs privées :

- type : int

- montant : double

- date: java.util.Date

-DEPOT=0

-RETRAIT=1

-VIREMENT=2

-Les constructeurs :

Operation()

Operation(type, montant) : le type et montant est saisi au clavier.

-Les méthodes usuelles :

String toString() : Renvoie la représentation d'opération sous forme d'une chaîne

boolean equals(Object) : Deux opérations égaux.

getters/setters

La classe CompteCourant :

Un CompteCourant est une classe héritée de la classe Compte. C'est un compte qui tolère un découvert lors des retraits d'argent. On a implémentée la classe CompteCourant en définissant notamment un constructeur et en spécialisant la méthode **retirer()**. Le retrait maintenant est toléré à la limite du **découvert** autorisé.

La classe CompteEpargne:

Un CompteEpargne est une classe héritée de la classe Compte. C'est un compte rémunéré qui rapporte de l'argent à chaque dépôt avec un **taux d'intérêt** donné. On a implémenté la classe CompteEpargne en définissant notamment un constructeur et en spécialisant la méthode **déposer()**. Chaque montant **m** déposé implique une augmentation du solde de **m + m*taux**.

La classe Admin_Client:

Cette classe permet de créer la première interface graphique où l'utilisateur s'identifier. En effet il précise s'il est un client ou un admin, la **liste des comptes** est l'attribut de cette classe et Admin_Client() est son constructeur.

La classe Identify_Admin:

Cette classe permet de créer l'interface graphique qui permet à l'admin de s'authentifier avec un login et un mot de passe avant d'accéder à sa fenêtre. Cette classe contient un seul constructeur.

La classe Identify_Client:

Cette classe permet de créer l'interface graphique qui permet au client de s'authentifier avec un login et un mot de passe avant d'accéder à sa fenêtre. Cette classe contient un seul constructeur **Identify_Client()** et une **liste des comptes** comme un attribut.

La classe AdminFrame :

Cette classe permet de coder l'interface graphique qui représente la fenêtre de l'admin. Elle permet de créer de nouveaux comptes ou afficher l'ensemble des comptes de l'agence. Cette classe contient le constructeur **AdminFrame()**.

La classe ClientFrame :

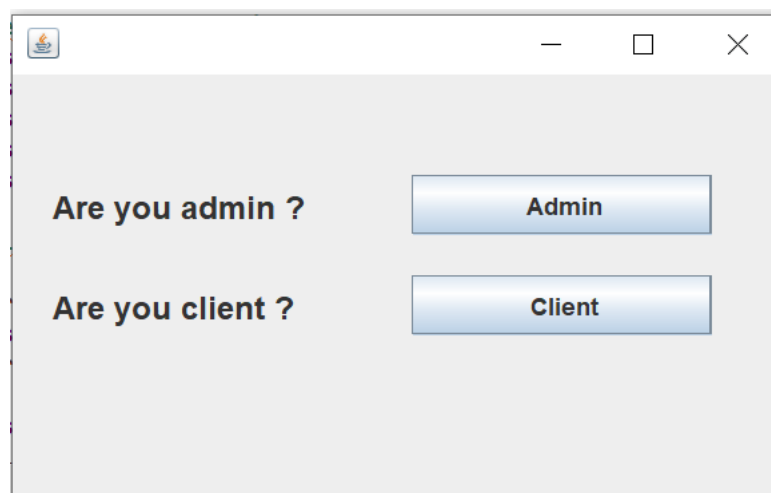
Cette classe permet de coder l'interface graphique qui représente la fenêtre du client. Elle permet de savoir des détails sur son compte et effectuer des opérations de dépôt et de retrait d'argent. Il peut également éditer les détails de ses opérations (relevé). Cette classe contient le constructeur **ClientFrame(compte)**.

III-Réalisation en Java:

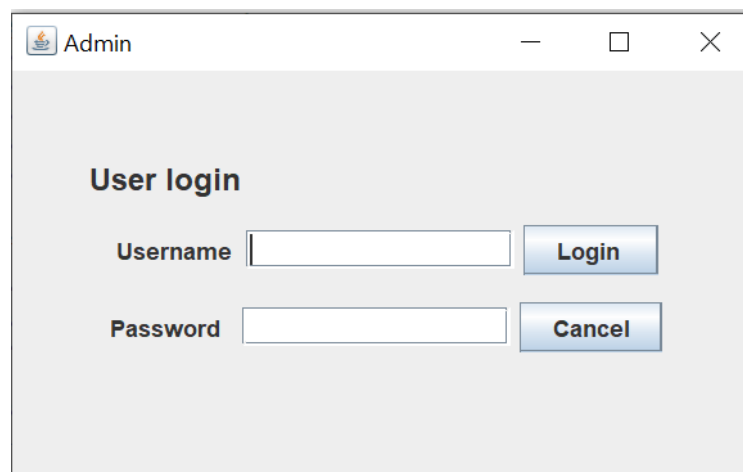
La réalisation de cette application a été fait sous eclipse , en utilisant le langage java. Les packages utilisés dans le code : Swing, JFrame, awt, time, util, ...

La première interface graphique a l'utilisateur de s'identifier. En effet il précise s'il est un client ou un admin. S'il est un client, il clique sur le bouton client, puis il accède à l'interface de s'authentification du client. S'il est un admin, il clique sur le bouton admin, puis il accède à l'interface de s'authentification de l'admin.

Le code de cette partie correspond à **La classe Admin_Client**.



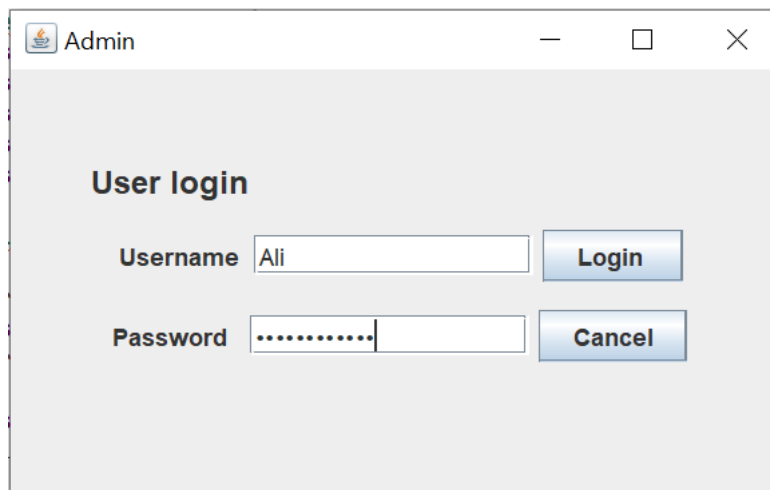
CAS1 : Apres que l'utilisateur clique sur le bouton Admin, il aura cette interface.



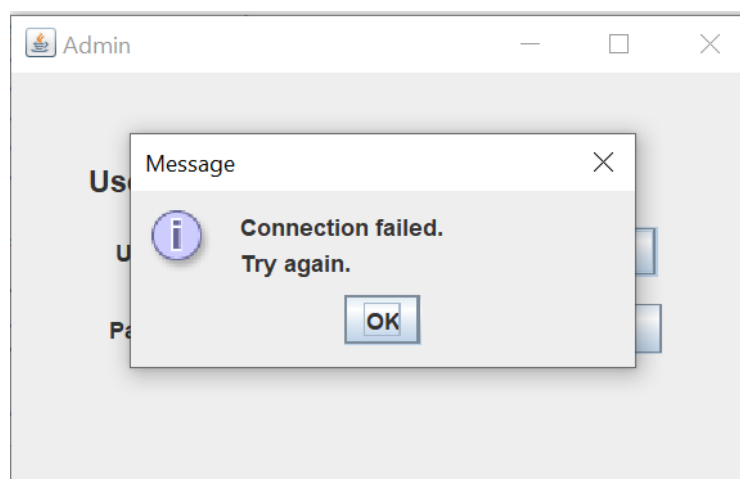
Cette interface est codé dans **La classe Identify_Admin**:

Admin fait entrer son nom est son mot de passe pour s'identifier. En effet, dans cette application, on a trois admins qui on le même mot de passe.

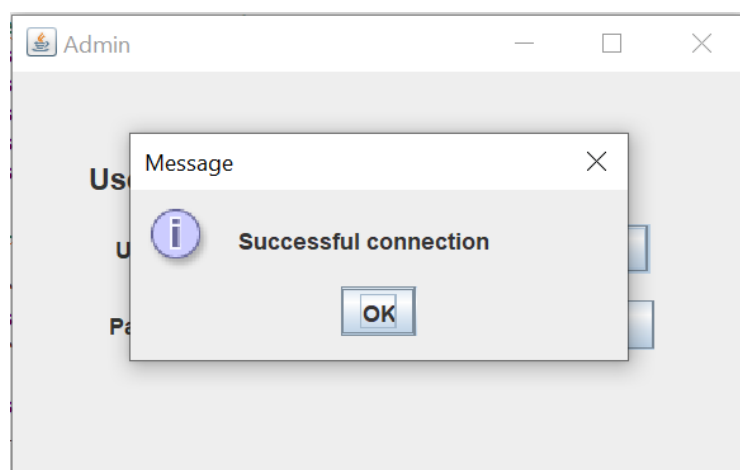
Nom de l'admin	Mot de passe
Fatima	123admin123
Ali	123admin123
Adam	123admin123



Après qu'il fait entrer le nom et mot de passe, il clique sur le bouton Login.
Si l'un de ces informations est fausse, il ne permet pas l'accès à l'interface suivante et il affiche ce message.



Si le nom et le mot de passe est correcte, il permet l'accès à l'interface suivante et il affiche ce message.



Bank Accounts

Account

Name

Balance

Description

Number	Name	Balance

Cette interface est codée dans **La classe AdminFrame**.

En effet, Cette interface graphique représente la fenêtre de l'admin. Elle permet de créer de nouveaux comptes ou afficher l'ensemble des comptes de l'agence.

Alors, pour ajouter un nouveau compte, on fait entrer le nom et la balance, puis on clique sur bouton new.

Bank Accounts

Account

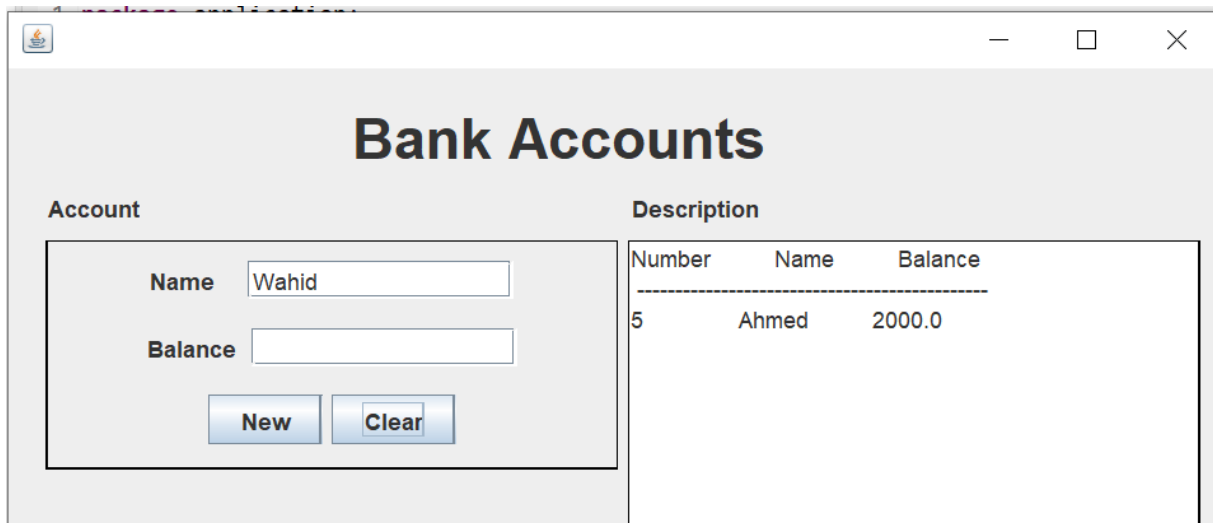
Name

Balance

Description

Number	Name	Balance
5	Ahmed	2000.0
6	Wahid	7000.0

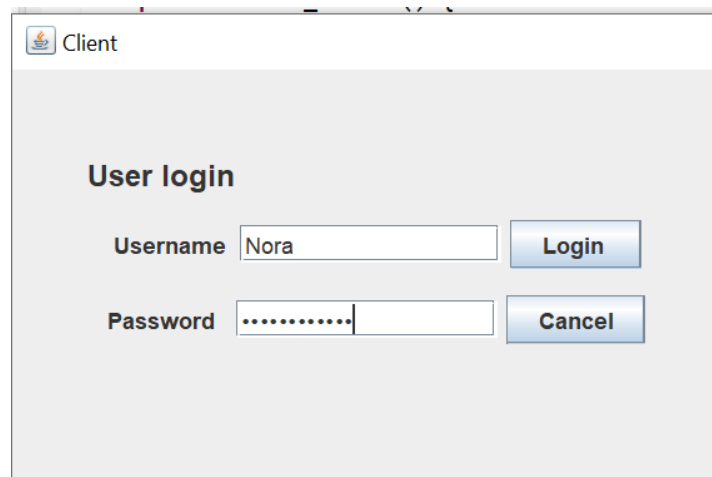
Et pour supprimer un compte, on fait entrer le nom , puis on clique sur bouton clear.



The 'Bank Accounts' window features a title bar with standard Windows controls. The main area is divided into two sections. On the left, under the 'Account' header, there is a form with a 'Name' field containing 'Wahid', an empty 'Balance' field, and two buttons labeled 'New' and 'Clear'. On the right, under the 'Description' header, there is a table with three columns: 'Number', 'Name', and 'Balance'. The table contains one row with the values '5', 'Ahmed', and '2000.0' respectively.

Number	Name	Balance
5	Ahmed	2000.0

CAS1 : Apres que l'utilisateur clique sur le bouton Client, il aura cette interface.

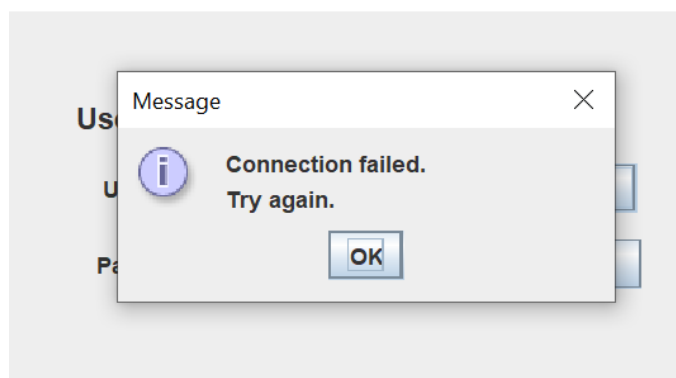


The 'Client' window has a title bar with a 'Client' icon and standard controls. The main area is titled 'User login' and contains two input fields: 'Username' with the text 'Nora' and 'Password' with masked characters '.....'. To the right of the 'Username' field is a 'Login' button, and to the right of the 'Password' field is a 'Cancel' button.

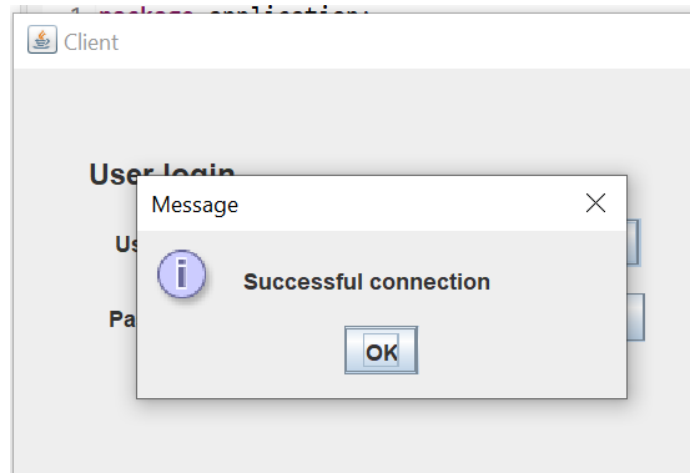
Cette interface est codée dans **La classe Identify_Client**

Client fait entrer son nom et son mot de passe pour s'identifier. En effet, tout client appartient à la liste des clients peut accéder à l'interface suivante avec **le mot de passe : 123client123.**

Si l'un de ces informations est fausse il ne permet pas l'accès à l'interface suivante et il affiche ce message.



Si le nom et le mot de passe est correcte, il permet l'accès à l'interface suivante et il affiche ce message.

A screenshot of a 'Bank Accounts' application window. The window has a title bar with standard OS controls. The main content area is divided into two columns: 'Account' and 'Description'. The 'Account' column contains a form with the following elements:

- IBAN**: A text input field containing the digit '6'.
- Name**: A label followed by the text 'Nora' in blue.
- Balance**: A label followed by the text '6000.0' in blue.
- A horizontal separator line.
- A text input field.
- Account**: A label.
- Two radio buttons: 'Deposit' (selected) and 'Withdr'.
- Another horizontal separator line.
- A 'details' button.

The 'Description' column is a large, empty rectangular area.

Cette interface est codée dans **La classe ClientFrame**.

En effet, Cette interface graphique représente la fenêtre du client. Elle permet de savoir des détails sur son compte et effectuer des opérations de dépôt et de retrait d'argent. Il peut également éditer les détails de ses opérations (relevé).

Le nom qui a été écrit dans l'interface précédente, permet d'écrire le nom, le numéro et la balance de ce client dans cette interface.

Alors, pour ajouter le montant et l'opération désirer, on clique sur bouton Details, puis il affiche les informations comme se voit dans cette interface.

Account	Description
IBAN 6	IBAN=6 Name=Nora Balance=10000.0
Name Nora	Operations
Balance 6000.0	2022-04-08 Deposit 4000.0
4000 <input checked="" type="radio"/> Deposit <input type="radio"/> Withdr	
Account	
details	

VI-Conclusion :

En conclusion, les interfaces bancaires est une application, dans l'intérêt du client et de la banque en même temps. En effet, cette application permet aux plusieurs clients l'accès à leurs comptes bancaires et effectuer leurs opérations au même temps. Elle est aussi en faveur du service bancaires, car il facilite, pour les admins, la gestion du processus des comptes bancaires.

Le codage des interfaces de cette application est en langage java.

Réalisée par : ANJAR FATIMA

BI&A, 1A