

***Activité Pratique-POO-Java- BI&A-  
2022/2023***

***Rapport du Projet : la gestion des  
comptes bancaires***

***Réalisée par : Amin Benali***

## **Introduction :**

La Bank est généralement un système informatique qui permet de gérer les comptes, les opérations et les soldes. Elle permet également de créer différents types de comptes, tels que les comptes d'épargne et les comptes courants, qui ont des droits différents. Les utilisateurs peuvent effectuer diverses opérations, comme déposer de l'argent dans un compte spécifique, retirer de l'argent et transférer des fonds entre différents comptes.

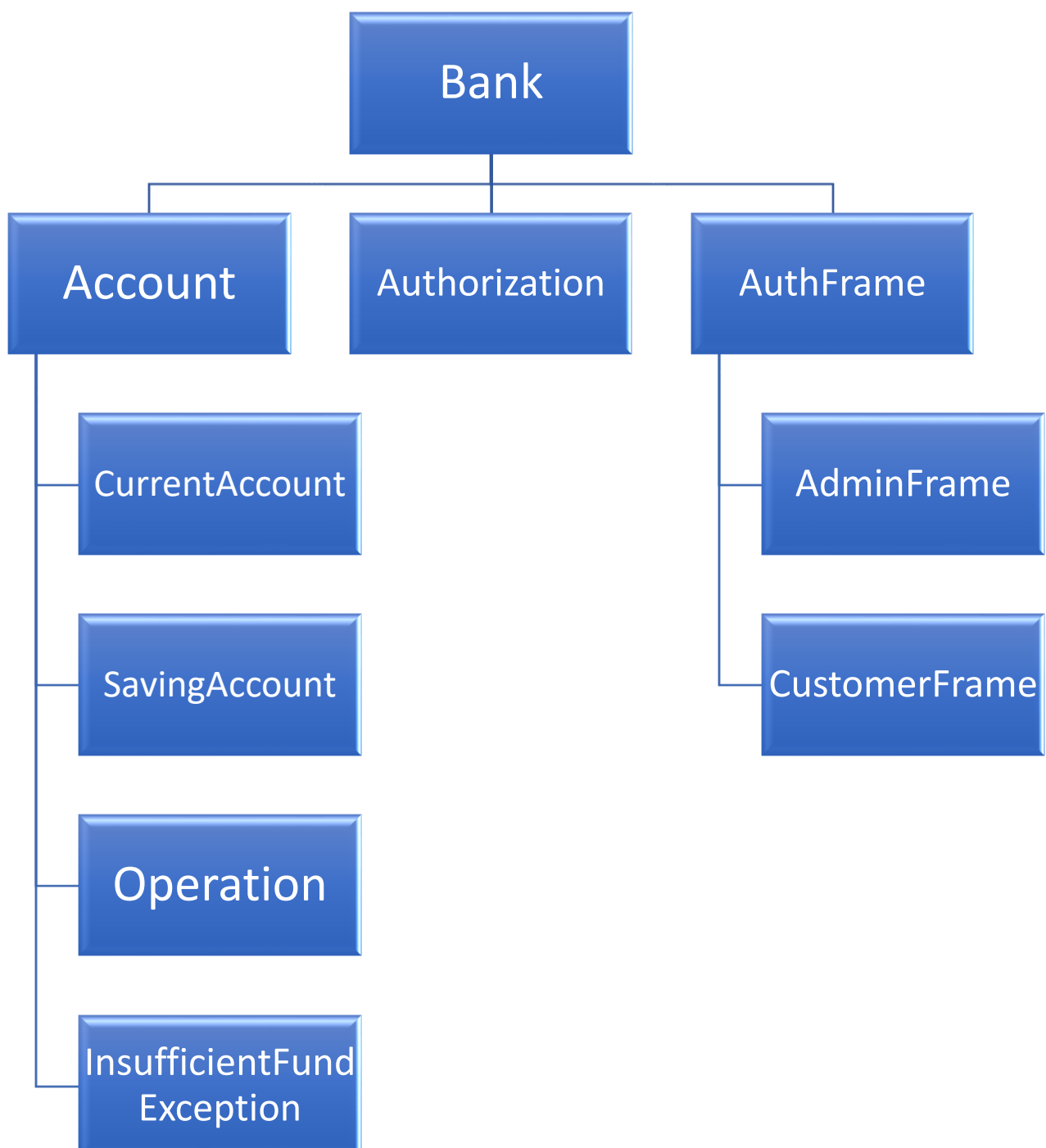
Ce projet vise à simuler ces fonctionnalités élémentaires et essentielles de la Banque en utilisant la programmation orientée objet en Java. De plus, pour assurer une bonne expérience utilisateur, on va ajouter la bibliothèque graphique Swing dans ce projet.

## **Analyse et conception :**

Pour définir le fonctionnement de base, on a besoin d'implémenter des classes qui regroupent des fonctionnalités élémentaires :

- Bank : c'est la classe principale à partir de laquelle le programme commence, elle permet d'exécuter une simulation qui crée des comptes et les associe des opérations, et de sauvegarder ces données en appliquant le principe de sérialisation.
- La classe Authorization : permet de faire l'authentification
- Une classe AuthFrame : qui s'étend de la Classe JFrame, c'est la première fenêtre qu'un utilisateur va rencontrer, elle prend en charge l'authentification, elle accepte comme nom de l'utilisateur et password une seule combinaison : admin/123.
- AdminFrame : aussi s'étend de la Classe JFrame, elle apparaît juste après l'authentification, elle prend en argument une instance de la Bank, et permet de visualiser tous les comptes et ajouter ou supprimer un compte.

- CostumorFrame : aussi s'étend de la Classe JFrame, elle apparait juste après l'authentification, elle prend en argument une instance de la Bank, et permet d'afficher les données d'un compte défini par un numéro unique IBAN.
- La classe Opération : elle permet de spécifier l'opération, de l'effectuer, et de la sauvegarder dans le relevé du compte bancaire.
- La classe InsufficientFundException : une exception pour le cas du retrait d'un montant plus grand que le solde.
- La classe Account : pour la création des comptes dans la Bank.
- 2 sous-classes de la classe Account : CurrentAccount et SavingAccount. Elles permettent de spécifier le type du compte pris en charge.



- Remarque : j'ai essayé de travailler avec JDBC, j'ai créé une classe qui permet la gestion de base de données, mais plusieurs bugs et problèmes ont apparus au cours de développement, la structure avec laquelle on a construit ce programme ne permet pas d'une intégration simple de JDBC.

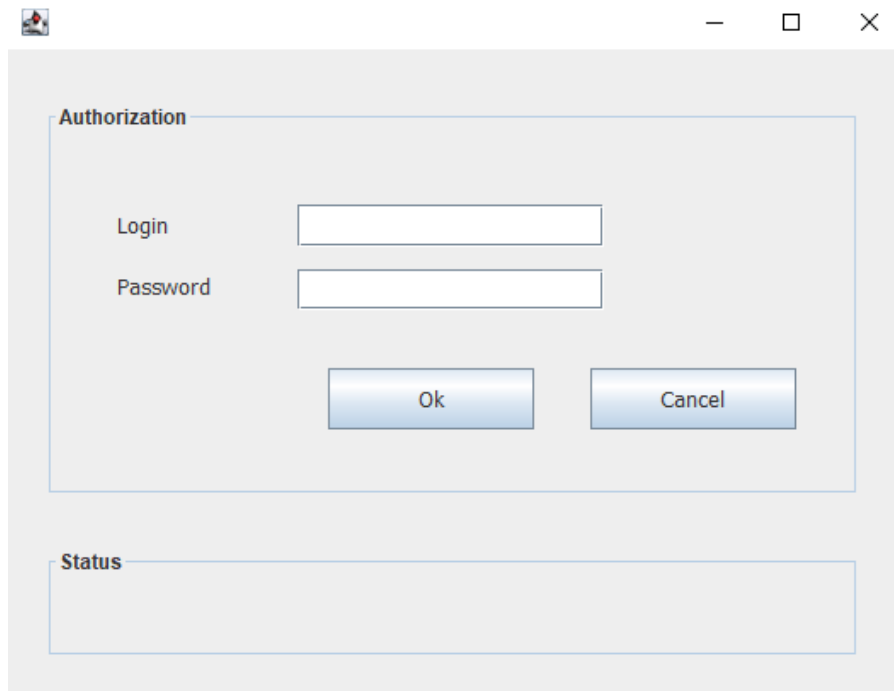
## **Réalisation en Java :**

La réalisation de ce programme a eu besoin d'importer des packages qui facilitent l'implémentation, voilà les packages utilisés :

- `java.io.FileOutputStream`; : pour créer un fichier de sortie et écrire des données.
- `java.time.LocalDate`; : Fournit des classes pour gérer les dates et les heures.
- `java.util.ArrayList`; : Fournit une implémentation de la structure de données de liste basée sur un tableau dynamique.
- `java.util.List`; : Interface pour les collections de données ordonnées et indexées.
- `java.util.Random`; : Génère des nombres aléatoires.
- `java.util.concurrent.ThreadLocalRandom`; : Fournit des fonctionnalités pour générer des nombres aléatoires dans des threads.
- `java.io.BufferedOutputStream`; : Ajoute un tampon à un flux de sortie pour améliorer les performances.
- `java.io.IOException`; : Permet de gérer les erreurs d'entrée/sortie.
- `java.io.ObjectOutputStream`; : Permet d'écrire des objets Java sérialisables dans un flux de sortie.
- `java.io.OutputStream`; : Classe abstraite pour les flux de sortie.
- `java.io.FileInputStream`; : Permet de lire des données à partir d'un fichier.
- `java.io.ObjectInputStream`; : Permet de lire des objets Java sérialisables à partir d'un flux d'entrée.
- `javax.swing` : fournit des composants graphiques pour la création d'interfaces utilisateur (UI) en Java.

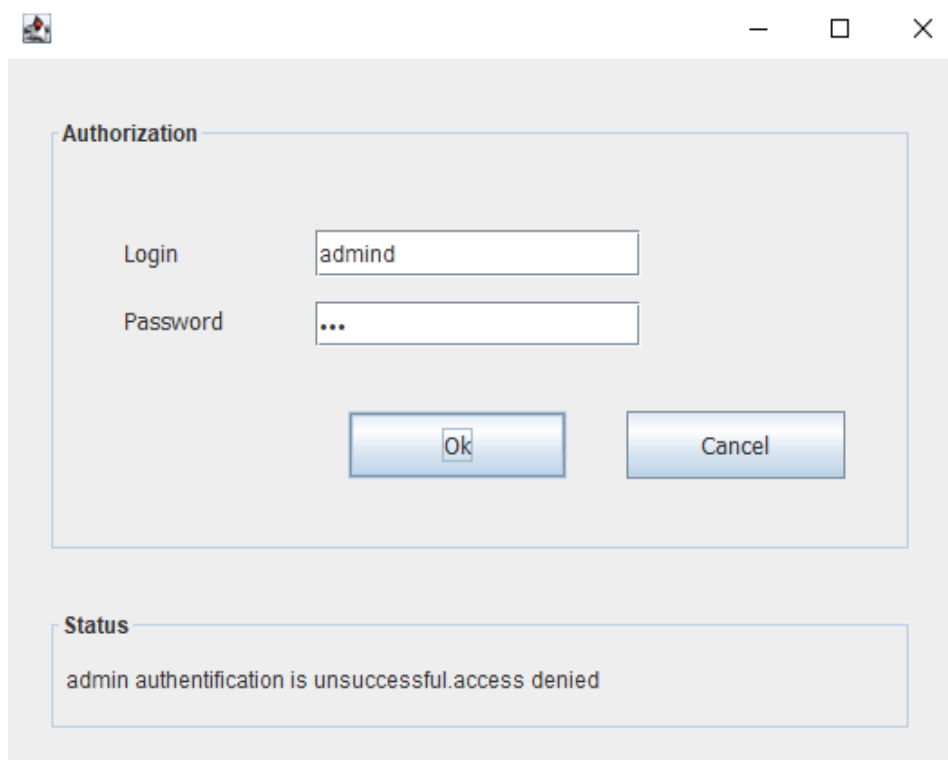
## Exécution :

- À l'exécution, la fenêtre d'authentification apparaît :



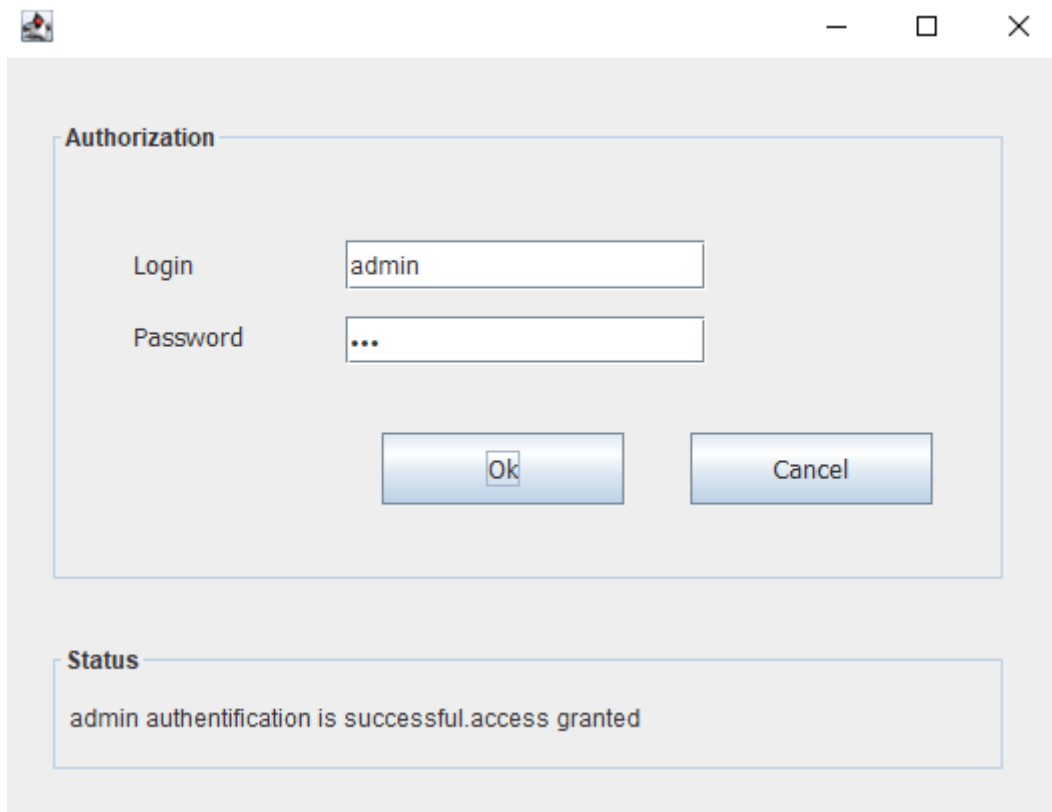
The screenshot shows a standard Windows-style dialog box titled "Authorization". It features a "Login" label next to an empty text input field, and a "Password" label next to another empty text input field. Below these fields are two buttons: "Ok" and "Cancel". At the bottom of the dialog, there is a "Status" label followed by an empty text area.

- Le cas de fournir des données incorrecte renvoie un status de « Access denied » :



This screenshot shows the same "Authorization" dialog box, but with the "Login" field containing the text "admind" and the "Password" field containing three dots "...". The "Ok" button is highlighted, indicating it has been clicked. In the "Status" text area at the bottom, the message "admin authentication is unsuccessful.access denied" is displayed.

- Dans le cas de fournir les correctes informations, le status renvoie « Access granted » :

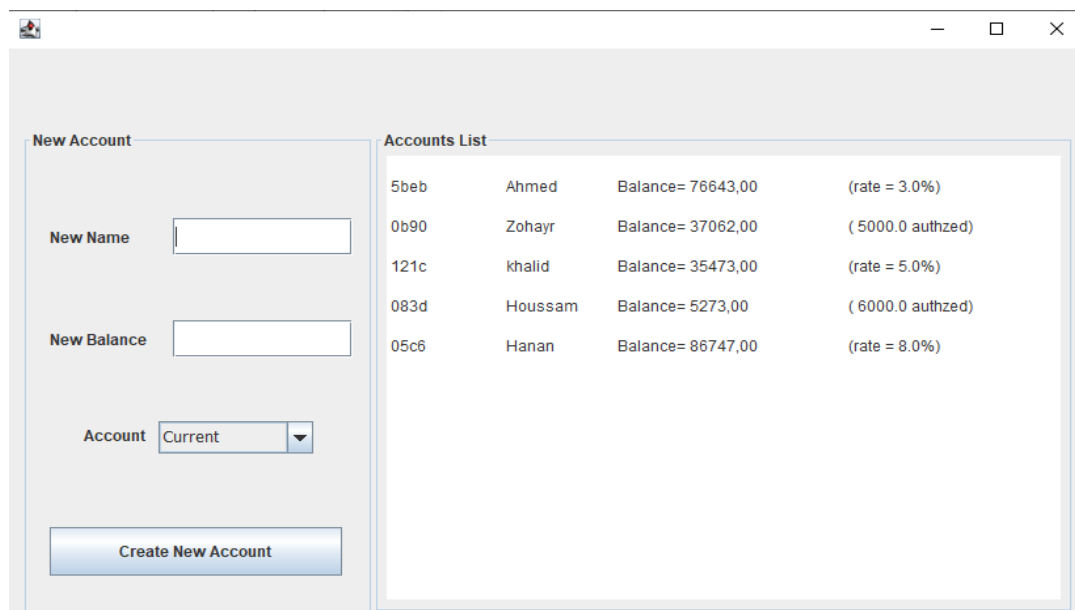


The image shows a Windows-style dialog box titled "Authorization". It contains two input fields: "Login" with the text "admin" and "Password" with three dots. Below these are "Ok" and "Cancel" buttons. At the bottom, a "Status" section displays the message "admin authentication is successful.access granted".

Authorization	
Login	admin
Password	...
<div>Ok Cancel</div>	
Status	
admin authentication is successful.access granted	

- Après l'authentification, les deux fenêtres suivantes apparaissent :

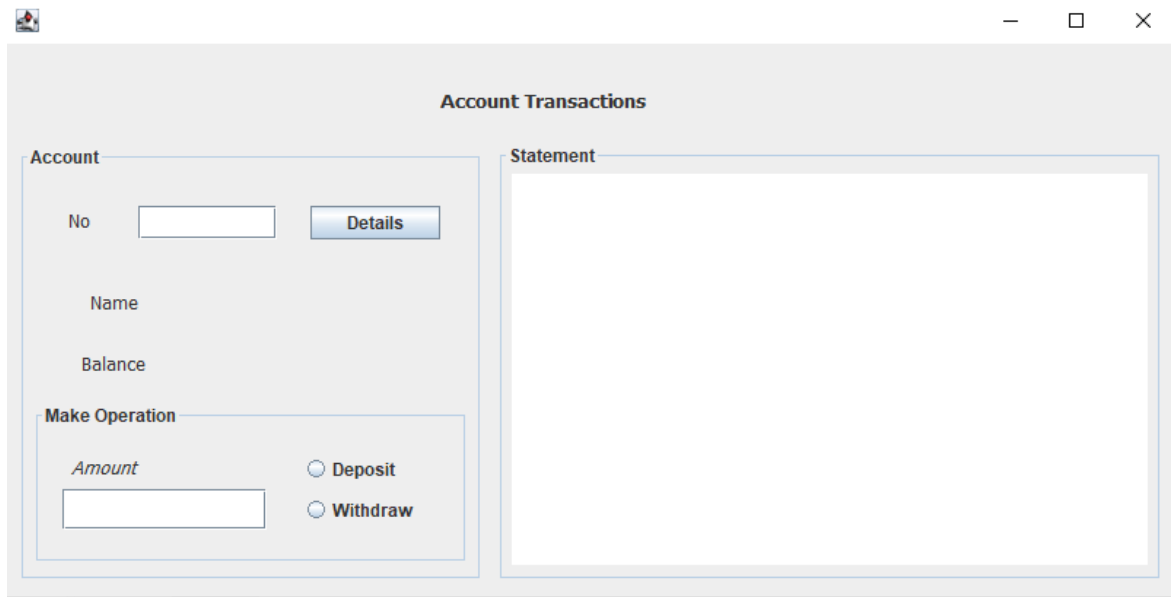
- AdminFrame :



The image shows a Windows-style window titled "AdminFrame". It is divided into two main sections. The left section, titled "New Account", contains input fields for "New Name", "New Balance", and a dropdown menu for "Account" (currently set to "Current"), along with a "Create New Account" button. The right section, titled "Accounts List", displays a table of existing accounts.

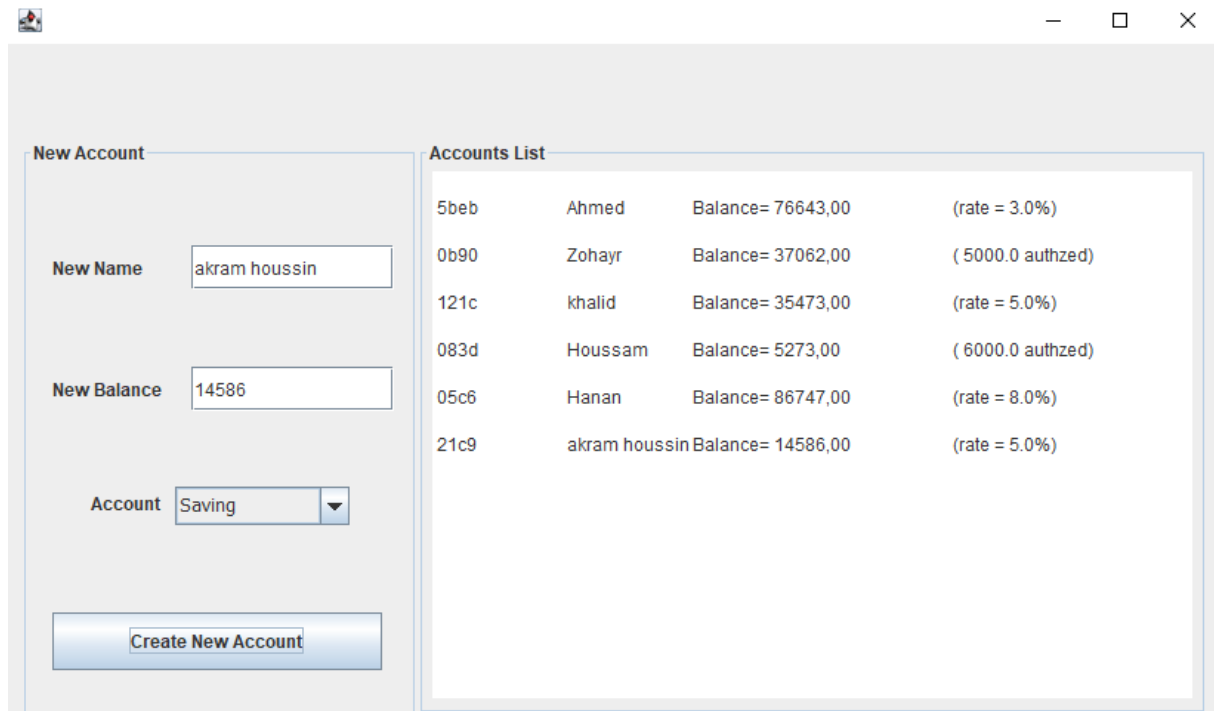
Accounts List			
5beb	Ahmed	Balance= 76643,00	(rate = 3.0%)
0b90	Zohayr	Balance= 37062,00	( 5000.0 authzed)
121c	khalid	Balance= 35473,00	(rate = 5.0%)
083d	Houssam	Balance= 5273,00	( 6000.0 authzed)
05c6	Hanan	Balance= 86747,00	(rate = 8.0%)

➤ CostumerFrame :



The screenshot shows a window titled "Account Transactions". It is divided into two main sections. The left section, titled "Account", contains fields for "No" (with an input box), "Name", and "Balance". Below these is a "Make Operation" section with an "Amount" input box and two radio buttons: "Deposit" and "Withdraw". A "Details" button is located next to the "No" field. The right section, titled "Statement", is a large empty rectangular area.

➤ AdminFrame Affiche tous les comptes de la banque et permet d'ajouter d'autres en spécifiant le nom, la balance et le type de compte :



The screenshot shows a window titled "AdminFrame". It is divided into two main sections. The left section, titled "New Account", contains fields for "New Name" (with input "akram houssin"), "New Balance" (with input "14586"), and "Account" (with a dropdown menu showing "Saving"). A "Create New Account" button is at the bottom. The right section, titled "Accounts List", displays a table of existing accounts.

ID	Name	Balance	Rate/Status
5beb	Ahmed	Balance= 76643,00	(rate = 3.0%)
0b90	Zohayr	Balance= 37062,00	( 5000.0 authzed)
121c	khalid	Balance= 35473,00	(rate = 5.0%)
083d	Houssam	Balance= 5273,00	( 6000.0 authzed)
05c6	Hanan	Balance= 86747,00	(rate = 8.0%)
21c9	akram houssin	Balance= 14586,00	(rate = 5.0%)

- CustomerFrame permet de donner plus de details sur un compte dont le numero est afficher dans la liste de la fenetre AdminFrame :

The screenshot shows a window titled "Account Transactions". It is divided into two main sections: "Account" and "Statement".

**Account Section:**

- No:** 5beb (text input)
- Details:** (button)
- Name:** Ahmed
- Balance:** 76643.0
- Make Operation:**
  - Amount:** (text input)
  - Deposit:** (radio button, selected)
  - Withdraw:** (radio button)

**Statement Section:**

5beb Ahmed Balance= 76643,00 (rate = 3.0%)

Operations :

2005-06-02	DEPOSIT	17799.11
2006-06-28	DEPOSIT	76378.59
2014-04-06	WITHDRAW	95848.64
2018-05-16	DEPOSIT	49750.06
2020-11-17	DEPOSIT	1639.24
2021-10-01	DEPOSIT	97404.24
2022-12-07	TRANSFERT	27014.35
2022-12-27	WITHDRAW	43823.85
2022-12-30	WITHDRAW	59202.91
2022-12-30	TRANSFERT	62807.85

- Aussi elle permet de faire des operations deposter/retirer en actualisant le relevé bancaire :

The screenshot shows the same "Account Transactions" window, but with updated information after a withdrawal operation.

**Account Section:**

- No:** 5beb (text input)
- Details:** (button)
- Name:** Ahmed
- Balance:** 6643.0
- Make Operation:**
  - Amount:** 70000 (text input)
  - Deposit:** (radio button)
  - Withdraw:** (radio button, selected)

**Statement Section:**

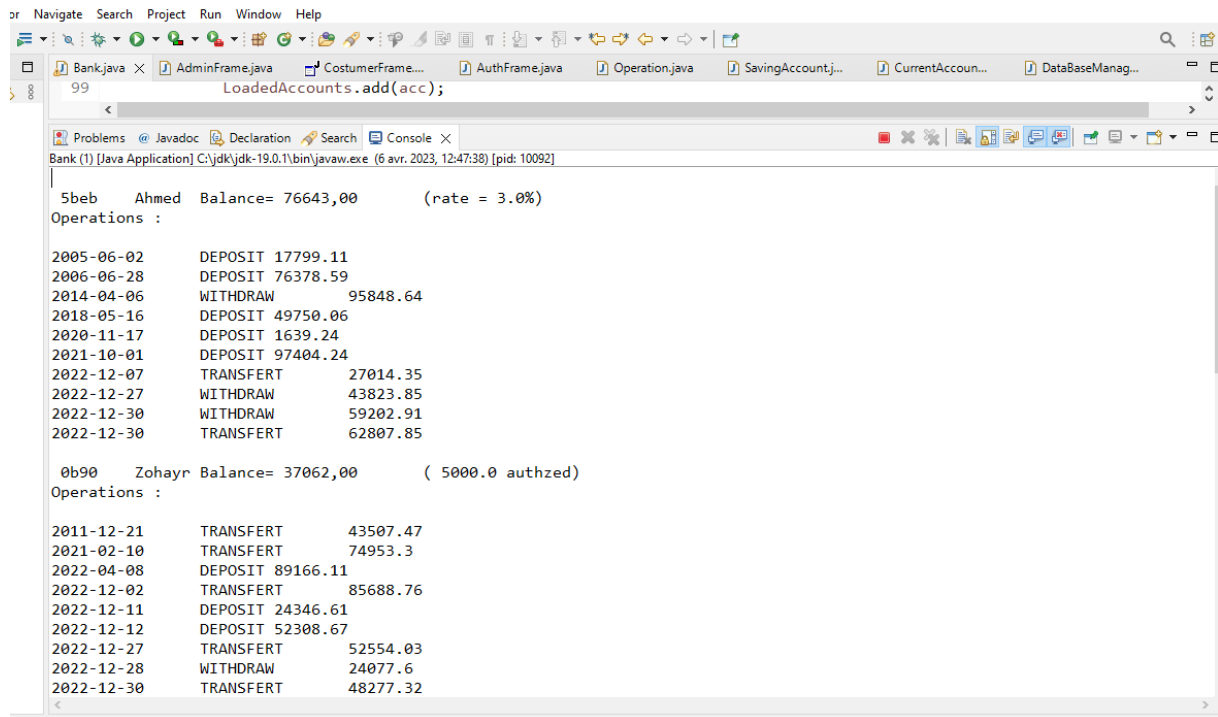
5beb Ahmed Balance= 6643,00 (rate = 3.0%)

Operations :

2005-06-02	DEPOSIT	17799.11
2006-06-28	DEPOSIT	76378.59
2014-04-06	WITHDRAW	95848.64
2018-05-16	DEPOSIT	49750.06
2020-11-17	DEPOSIT	1639.24
2021-10-01	DEPOSIT	97404.24
2022-12-07	TRANSFERT	27014.35
2022-12-27	WITHDRAW	43823.85
2022-12-30	WITHDRAW	59202.91
2022-12-30	TRANSFERT	62807.85
2023-04-05	WITHDRAW	70000.0



- Le programme fonctionne avec le mécanisme de sérialisation, il enregistre les données dans un fichier "bank.dta" et puis il peut lire c'est donné :



The screenshot shows a Java IDE with a console window displaying the output of a Java application. The application is titled "Bank (1) [Java Application] C:\jdk\jdk-19.0.1\bin\javaw.exe (6 avr. 2023, 12:47:38) [pid: 10092]". The console output shows two accounts and their operations.

```
5beb  Ahmed  Balance= 76643,00      (rate = 3.0%)
Operations :

2005-06-02    DEPOSIT 17799.11
2006-06-28    DEPOSIT 76378.59
2014-04-06    WITHDRAW      95848.64
2018-05-16    DEPOSIT 49750.06
2020-11-17    DEPOSIT 1639.24
2021-10-01    DEPOSIT 97404.24
2022-12-07    TRANSFERT    27014.35
2022-12-27    WITHDRAW     43823.85
2022-12-30    WITHDRAW     59202.91
2022-12-30    TRANSFERT    62807.85

0b90  Zohayr  Balance= 37062,00      ( 5000.0 authzed)
Operations :

2011-12-21    TRANSFERT    43507.47
2021-02-10    TRANSFERT    74953.3
2022-04-08    DEPOSIT 89166.11
2022-12-02    TRANSFERT    85688.76
2022-12-11    DEPOSIT 24346.61
2022-12-12    DEPOSIT 52308.67
2022-12-27    TRANSFERT    52554.03
2022-12-28    WITHDRAW     24077.6
2022-12-30    TRANSFERT    48277.32
```

(Les fichiers sources du code comprend une classe supplémentaire DataBaseManager, c'est ma tentation d'intégrer le JDBC avant abandonner à cause de ce qu'elle demande de changement complexe dans le programme.)

## **Conclusion :**

Ce projet est une opportunité pour la compréhension et l'application des différents concepts de la programmation orientée objet (POO), il m'a permis de construire une application qui simule la gestion des comptes bancaires par les admins des agences bancaire, et aussi les services que ces agences proposent aux clients.