

# UNIVERSITATEA TEHNICĂ CLUJ- NAPOCA

AUTOMATICĂ ȘI CALCULATOARE, AN II



## Lucrare laborator – Assignment IV Bank Application

Profesor curs: Prof. Dr. Ing. Cristina Pop

Profesor laborator: Teodor Petrican

Student: Jitaru Andrei

Grupa: 30229

# CUPRINS

1. Introducere – Obiectivul temei
2. Analiza problemei
  - a. Asumpții
  - b. Modelare
  - c. Scenarii
  - d. Cazuri de utilizare
  - e. Erori
3. Proiectare
  - a. Diagrama UML
  - b. Structuri de date utilizate în implementare
  - c. Proiectare clase
  - d. Algoritmi
  - e. Interfata Utilizator
4. Implementare
5. Testare
6. Rezultate
7. Concluzii și dezvoltări ulterioare

## **1. Introducere – obiectivul proiectului**

Obiectivul acestui proiect este de a simula operatiile care se pot realiza la nivelul unei banci. In alte cuvinte, s-a dorit implementarea in limbaj de programare JAVA a operatiilor de adaugare/stergere/editare persona, adaugare/stergere/ediate cont dar si a altor operatii caracteristice interactiunii cu o intreprindere de tip banca dintre care amintim retragerea sau adaugarea unor sume de bani in/din conturile anumitor persoane.

Aceste task-uri au fost impuse cu scopul de a demonstra, pe de o parte, înțelegerea asupra temei și cerintelor date, iar pe de alta parte pentru a oferi studentilor practicanti ocazia de a-si etale aptitudinile de implementare a acestora în contextul limbajului de programare mai sus menționat.

Toate aspectele enumerate anterior au fost verificate prin implementarea unei aplicații simple. Aceasta este realizată sub forma unui program interactiv cu un aspect rudimentar, care permite realizarea operatiilor mai sus mentionate printr-o utilizare simpla, care nu necesita cunostinte specializate. Principalul target al implementarii acestei teme a fost ingeniozitatea si capacitatea de a oferi utilizatorilor o experinta cat mai nonsalanta odata cu utilizarea aplicatiei.

## **2. Analiza problemei**

Proiectul, care în speță reprezintă un sistem de procesare a operatiilor fundamentale care pot fi realizate la nivelul unei aplicatii destinate unei intreprindere de tip banca, sugerează un proces complex care necesită atenție la fiecare pas, atât în contextual realizării fiecărei operații cât și în sensul testării cât mai multor exemple diferite în scopul verificării acestora. O astfel de abordare este absolut necesară deoarece fiecare operație este unică în felul ei și prezintă diferite excepții specifice, acestea trebuind tratate în parte cu scopul obținerii unei funcționări impecabile a programului.

### **a. Asumpții**

Pentru a obține rezultate optime în urma utilizării aplicației este necesară impunerea unor asumptii. În acest sens, presupunem că utilizatorul nu va introduce niciodată caractere străine în TextField-urile dedicate introducerii datelor de intrare (caracter străin = orice caracter care nu este o cifră, în cazul TextField-urilor a căror nume impune introducerea de caractere tip cifră, respectiv orice caracter care nu este o literă, în cazul TextField-urilor a căror nume sugerează introducerea de caractere de tip litere).

### **b. Modelare**

Cerința temei și anume de a se realiza un sistem de procesare a operațiilor realizabile la nivelul unei întreprinderi de tip bancă poate fi implementată în diferite moduri, desigur, fiecare cu eficiența sa. Prima și cea mai simplă metodă de implementare ar fi cea în care se utilizează două liste pentru memorarea persoanelor, respectiv a conturilor. Această soluție nu este însă cea mai indicată deoarece nu oferă o eficiență satisfăcătoare utilizării aplicației. Operațiile de ștergere (ne referim aici la ștergerea tuturor conturilor unei persoane), de exemplu, ar putea fi realizate cu dificultate din cauza necesității de a parcurge întreaga listă de conturi pentru a le identifica și a le șterge în consecință. Din această cauză s-a optat pentru o soluție mai simplă, care îndeplinește și una dintre cerințele temei, și anume utilizarea unui HashMap.

Dacă privim problema din punct de vedere al vitezei de lucru dar și a eficienței, atunci fără îndoială soluția mai sus menționată este cea corectă. Astfel, s-a optat pentru crearea unui hashMap de persoane și conturi în cadrul cărui persoanele reprezintă cheile map-ului iar conturile reprezintă valorile acestora. Privind această metodă de implementare mai simplu putem spune că fiecărei persoane îi corespunde o listă de conturi pe care le deține. Astfel, această soluție permite efectuarea mai rapidă a operațiilor dorite, operații cum ar fi cele de ștergere sau editare putând fi realizate cu foarte mare ușurință. Conturile prezintă și ele de asemenea modele variate cu semnificație diferite între acestea. Acestea se clasifică în conturi de cheltuieli și conturi de economii. Diferența dintre cele două este dată de faptul că un cont de cheltuieli ne permite

sa realizam mai multe retrageri si sa adaugam mai multe sume, in timp ce un cont de economii permite depunerea unei singure sume de bani (foarte mari insa), urmata de intreaga retragere a acesteia. De asemenea un cont de economii presupune aplicarea unei dobanzi pe perioada in care contractul este activ la banca respectiva.

### **c. Scenarii**

O serie de etape trebuie urmate pentru ca aplicatia să fie capabilă de a returna rezultate clare și corecte. Totalitatea acestor etape dau naștere unei serii de scenarii care definesc funcționalitatea programului. În acest sens, pentru a obține un anumit rezultat, diferite evenimente trebuie să aibă loc. In partea stanga interfata grafica a aplicatiei pune la dispozitie un meniu de comenzi pentru a putea naviga printe optiuni.

Pași de utilizare ai aplicației:

- Pentru adaugarea unei persoane:
  - Se selecteaza din meniul din partea stanga optiunea "ADD PERSON";
  - Se introduc datele referitoare crearii unei noi persoane in TextField-uri cu nume sugestiv;
  - Se realizeaza crearea persoanei prin apasarea butonului "ADD PERSON" din interfata principala (a nu se confunda cu butonul "ADD PERSON" din meniu).
- Pentru eliminarea unei persoane:
  - Se selecteaza din meniul din partea stanga optiunea "VIEW PERSONS" care va deschide un tabel (in cazul in care acest tablou nu este inca afisat) cu scopul vizualizarii persoanelor din banca;
  - Se alege din tabel persoana care se doreste a fi stearsa din banca prin apasarea butonului click oriunde pe randul dedicat acesteia in cadrul tabelului;
  - Se selecteaza din meniul din partea stanga optiunea "REMOVE PERSON".

- Pentru editarea unei persoane:
  - Se selecteaza din meniul din partea stanga optiunea "VIEW PERSONS" care va deschide un tabel (in cazul in care acest tablou nu este inca afisat) cu scopul vizualizarii persoanelor din banca;
  - Se alege din tabel persoana care se doreste a fi editate din banca prin apasarea butonului click oriunde pe randul dedicat acesteia in cadrul tabelului;
  - Se selecteaza din meniul din partea stanga optiunea "EDIT PERSON".
- Pentru vizualizarea persoanelor:
  - Se selecteaza din meniul din partea stanga optiunea "VIEW PERSONS" care va deschide un tabel populat de toate persoanele din banca.
- Pentru adaugarea unui cont:
  - Se selecteaza din meniul din partea stanga optiunea "VIEW PERSONS" care va deschide un tabel populat de toate persoanele din banca.
  - Se alege din tabel persoana careia se doreste a se adauga un nou cont prin apasarea butonului click oriunde pe randul dedicat acesteia in cadrul tabelului;
  - Se selecteaza din meniul din partea stanga optiunea "ADD ACCOUNT";
  - Se introduc datele referitoare crearii unui nou cont in TextField-uri cu nume sugestiv (doar in cazul crearii unui nou Saving Account este necesar a se introduce toate datele de intrare, in cazul crearii unui nou Speding Account fiind suficienta introducerea unui id unic si a tipului contului dorit, adica Spending);
  - Se realizeaza crearea contului prin apasarea butonului "ADD ACCOUNT" din interfata principala (a nu se confunda cu butonul "ADD ACCOUNT" din meniu).

- Pentru eliminarea unei cont:
  - Se selecteaza din meniul din partea stanga optiunea "VIEW ACCOUNTS" care va deschide un tabel (in cazul in care acest tablou nu este inca afisat) cu scopul vizualizarii conturilor din banca;
  - Se alege din tabel contul care se doreste a fi sters din banca prin apasarea butonului click oriunde pe randul dedicat acestuia in cadrul tabelului;
  - Se selecteaza din meniul din partea stanga optiunea "REMOVE ACCOUNT".
- Pentru editarea unui cont
  - Se selecteaza din meniul din partea stanga optiunea "VIEW ACCOUNTS" care va deschide un tabel (in cazul in care acest tablou nu este inca afisat) cu scopul vizualizarii conturilor din banca;
  - Se alege din tabel contul care se doreste a fi editat din banca prin apasarea butonului click oriunde pe randul dedicat acestuia in cadrul tabelului;
  - Se selecteaza din meniul din partea stanga optiunea "EDIT ACCOUNT".
- Pentru vizualizarea conturilor:
  - Se selecteaza din meniul din partea stanga optiunea "VIEW ACCOUNTS" care va deschide un tabel populat de toate conturile din banca.
- Pentru efectuarea unei retrageri:
  - Se selecteaza din meniul din partea stanga optiunea "VIEW ACCOUNTS" care va deschide un tabel (in cazul in care acest tablou nu este inca afisat) cu scopul vizualizarii conturilor din banca;
  - Se alege din tabel contul pentru care se doreste realizarea retragerii valutare prin apasarea butonului click oriunde pe randul dedicat acestuia in cadrul tabelului;
  - Se selecteaza din meniul din partea stanga optiunea "WITHDRAW".
  - Se introduce cantitatea valutara care va fi retrasa din cont;

- Se realizeaza retragerea sumei prin apasarea butonului "WITHDRAW" din interfata principala (a nu se confunda cu butonul "WITHDRAW" din meniu).
- Pentru efectuarea unei depuneri:
  - Se selecteaza din meniul din partea stanga optiunea "VIEW ACCOUNTS" care va deschide un tabel (in cazul in care acest tablou nu este inca afisat) cu scopul vizualizarii conturilor din banca;
  - Se alege din tabel contul pentru care se doreste realizarea depunerii valutare prin apasarea butonului click oriunde pe randul dedicat acestuia in cadrul tabelului;
  - Se selecteaza din meniul din partea stanga optiunea "DEPOSIT".
  - Se introduce cantitatea valutara care va fi depusa in cont;
  - Se realizeaza depunerea sumei in banca prin apasarea butonului "DEPOSIT" din interfata principala (a nu se confunda cu butonul "DEPOSIT" din meniu).

#### **d. Cazuri de utilizare**

Luând în considerare faptul că acest proiect realizează simularea operatiilor la nivelul unei intreprinderi de nivel banca aceasta ar putea fi utilizata cu singuranta si in scoputi practice in acest sens. Din momemnt ce dispune de o interfă grafică simplă cu o utilizare la fel de banală, se poate considera fără lipsă de incredere ca ar putea fi folosită de orice persoană care ar dori sa depuna sau sa retraga o anumita suma de bani din banca, sa-si creeze un cont, sa-si editeze datele personale sau sa renunte la contul respectiv.

#### **e. Erori**

Anumite evenimente pot determina apariția unor excepții care au fost tratate la implementare sub forma unor erori. Apariția acestor evenimente produc afișarea unei casuțe de dialog în care este prezentată eroarea pentru a fi mai facil de identificat.



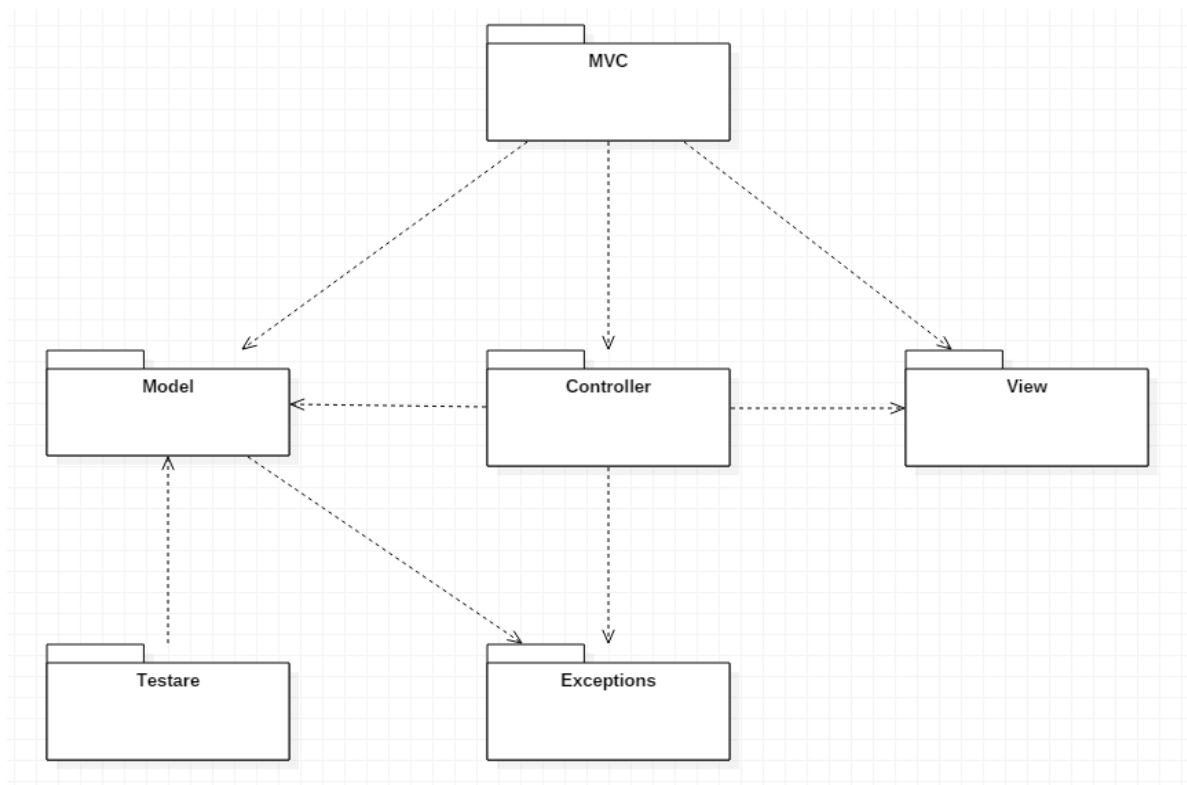
Cazuri care pot conduce la apariția erorilor:

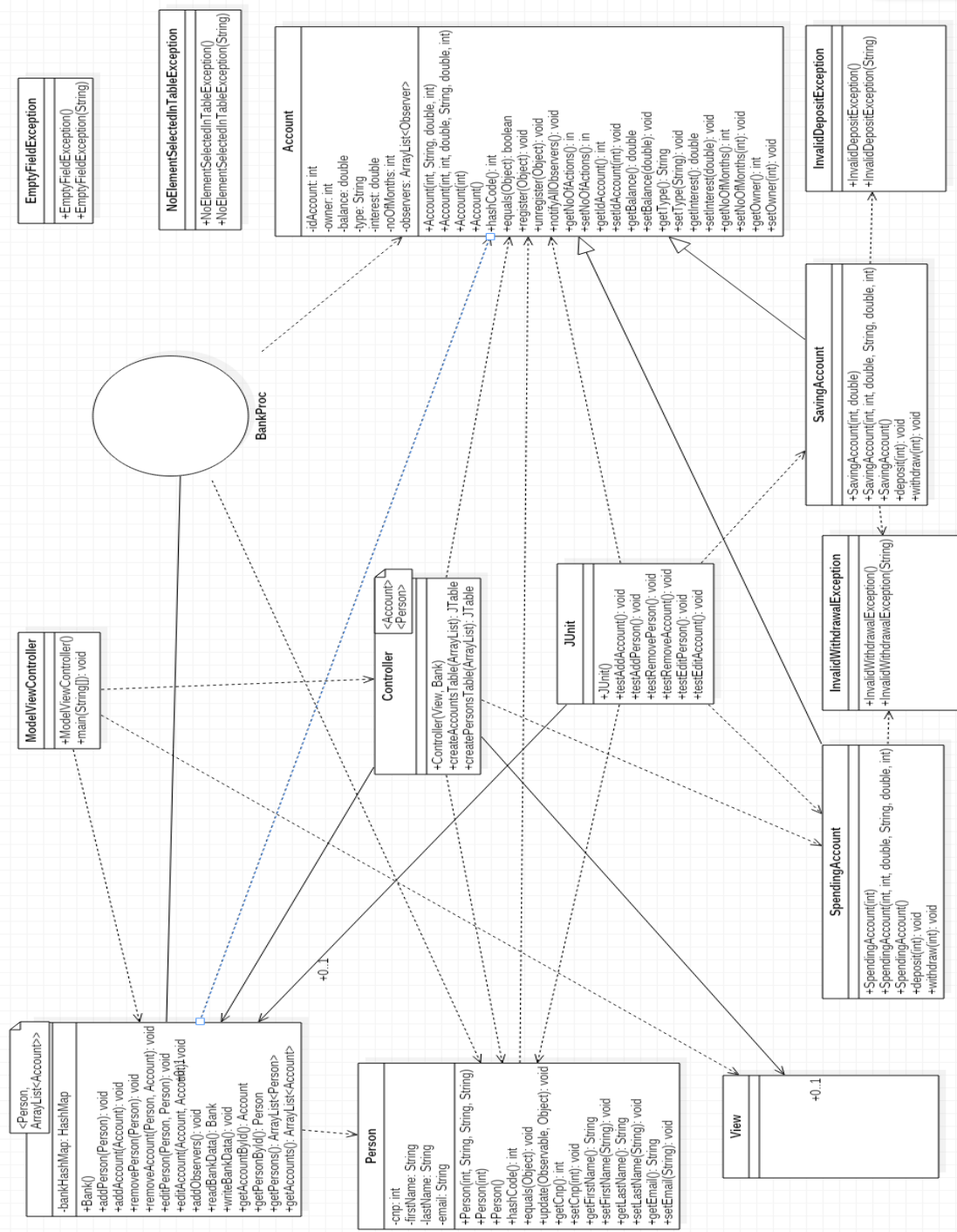
- Dacă detinatorul unui Saving Account dorește depunerea unei sume mai mici de 1000 UM (UM = Unitati Monetare), atunci va fi aruncata o exceptie de tipul InvalidDepositException cu mesajul "You need to deposit a larger sum!";
- Dacă detinatorul unui Saving Account dorește să realizeze mai mult de o depunere atunci va fi aruncata o exceptie de tipul InvalidDepositException cu mesajul "You can only deposit a single time!";
- Dacă detinatorul unui Saving Account dorește retragerea unei sume care nu reprezintă valoarea totală a UM asociate contului respectiv atunci va fi aruncata o exceptie de tipul InvalidDepositException cu mesajul "You can only withdraw the whole sum from the bank!";
- Dacă detinatorul unui Saving Account dorește să realizeze mai mult decât o retragere atunci va fi aruncata o exceptie de tipul InvalidDepositException cu mesajul "You can only withdraw a single time!";
- Dacă detinatorul unui Spending Account dorește să retragerea unei sume care depășește valoarea totală a UM asociate contului respectiv atunci va fi aruncata o exceptie de tipul InvalidWithdrawalException cu mesajul "Insufficient balance!";
- Dacă utilizatorul nu a selectat nicio linie din tabela Persons/Account în cazul efectuării operațiilor de Remove Person, Edit Person, Add Account, Edit Account, Remove Account, Withdraw sau Deposit conform instrucțiunilor prezentate în randurile de mai sus atunci va fi aruncata o exceptie de tipul NoElementSelectedInTableException cu mesajul "Nici un element nu a fost selectat în tabela Persons/Account pentru efectuarea operației de ștergere/editare/adaugare/depunere/retragere!";
- Dacă utilizatorul dorește crearea unui cont de tipul Saving Account sau editarea unui cont de tipul Spending Account astfel încât să devină de tipul Saving Account și nu va introduce date de intrare în TextField-urile "Age" și "Interest" va fi aruncata o eroare de tipul EmptyFieldException cu mesajul "Interest or Age field are empty!".

## 3. Proiectare

### a. Diagrame UML

În diagramele UML de mai jos sunt prezentate toate pachetele și clasele, cu atributele și relațiile dintre acestea. O analiză detaliată a acestor diagrame este prezentată în capitolul 4 al acestei documentații.





## **b. Structuri de date utilizate în implementare**

Funcționarea aplicației este strâns legată de utilizarea de structuri de date în cadrul implementării acesteia. Structura de baza pe care își are temeliile dezvoltarea proiectului este HashMap-ul care infatisează modalitatea de stocare a persoanelor și a conturilor aferente fiecăreia. Au fost utilizate de asemenea și ArrayList-uri pentru a eficientiza proprietatea de chaining a HashMap-urilor.

## **c. Proiectare clase**

Proiectul, ca un tot unitar, este divizat în 14 clase care permit transpunerea cerinței și temei în limbaj de programare Java. Clasele au dimensiuni, alcătuiri și roluri variate.

Clasa „Bank” se află la baza proiectului. Aceasta conține mijlocul de memorarea al conturilor și clienților dar și metodele care descriu diferite acțiuni care pot fi executate la nivelul clasei. Aceasta implementează interfața bankProc care stochează antetul catorva metode care trebuiesc implementate în clase Bank. Clasele SpendingAccount și SavingAccount extind ambele clase Account care prezintă descriere în limbaj a unui cont din aplicație. Fiecare cont are astfel ca atribut un id unic, un detinator, o cantitate de bani, un tip ("Saving" sau "Spending"), o dobândă, o perioadă de când a fost creat și un număr de acțiuni care pot fi executate asupra lui (în cazul unui SavingAccount cel mult două, iar în cazul unui SpendingAccount un număr nelimitat). Cele 2 clase, SpendingAccount și SavingAccount au implementări diferite atât pentru metoda withdraw cât și pentru metoda deposit.

Clasa Person descrie o persoană și atributele acesteia. Astfel, o persoană are un cnp unic, un prim nume, un al doilea nume și o adresă de email. Fiecărei persoane îi corespunde o listă de conturi în cadrul bancii. Clase EmptyFieldException, InvalidDepositException, InvalidWithdrawalException și NoSuchElementException ilustrează excepțiile care vor fi aruncate de către aplicație în anumite cazuri speciale. Clasa View este cea care realizează interfața grafică a programului și permite interacțiunea cu utilizatorul. Aceasta și clasele din pachetul Model sunt legate una de celelalte prin clase Controller. Astfel

conexiunea operatiilo la interfata este realizata. Clasa Mvc este cea care pune la loc toate componentele aplicatiei.

#### d. **Algoritmi**

În continuare sunt prezentați principalii algoritmi care permit buna funcționare a aplicației. Vor fi prezentate metodele din clasa Bank care permit operațiile realizate la nivelul aplicatiei:

- `public void addPerson(Person person)` - adauga o persoana in HashMap-ul bancii;
- `public void addAccount(Person person, Account accountToAdd)` - adauga un cont la cheia Person in HashMao-ul bancii. In prima faza se verifica daca account-ul exista deja in HashMap. Daca conditia este indeplinita se ia lista de la cheia person, se adauga contul in lista si se pune lista inapoi in HashMap la cheia respectiva;
- `public void removePerson(Person person)` - sterge o cheie (persona) din HashMap si toate conturile asociate acesteia;
- `public void removeAccount(Account accountToDelete)` - sterge account-ul unei anumite persoane. Persoana de la care se sterge account-ul se decide prin verificare campului owner a contului;
- `public void editPerson(Person oldPerson, Person newPerson)` - editeaza attributele unei persoane. In prima faza se copiaza intr-o lista auxiliara lista de account-uri a persoanei pe care dorim s-o editam dupacare se realizeaza stergerea persoanei si adaugarea unei noi persoane (care reprezinta vechea persoana insa care acum are campurile actualizate). Urmeaza o verificare a conturilor persoanei in scopul modificarii campului owner a acestora si reamplasarea listei la cheia reprezentata de noua persoana.
- `Public void editAccount(Account oldAccount, Account newAccount)` - editeaza attributele unui cont. Daca id-ul detinatorului contului se modifica atunci contul va fi sters din lista de conturi a vechiului detinator

- si pus in lista noului detinator cu valorile actualizate. In caz contrar, se cauta indexul contului vechi in lista de conturi a detinatorului acestuia si se pune la adresa contului vechi valoarea contului nou. Lista de conturi a detinatorului este actualizata.

#### **e. Interfata utilizator**

Interfața cu utilizatorul are un aspect minimalist pentru a putea fi utilizată cât mai ușor de orice persoană în parte. S-a optat doar pentru introducerea strictului necesar. Astfel, interfața conține un meniu cu 10 butoane în partea stanga, fiecare având un nume caracteristic pentru operația care o efectuează. Prin apăsarea acestor butoane vor fi deschise panouri cu tabele sau campuri de inserare a datelor și butoane pentru realizarea operațiilor.

### **4. Implementare**

Întreaga aplicație este implementată aferent modelului architectural “Model-View-Controller”. Alegerea acestui model se datorează izolării logicii față de considerentele interfeței cu utilizatorul, rezultând o aplicație unde aspectul vizual și nivelele inferioare sunt mai ușor de modificat, fără a afecta alte nivele. Clasa pivot a proiectului este fără de îndoială clasa Bank. Aceasta implementează operațiile executate la nivelul unei banci, dai și metode auxiliare care ajută la buna funcționare a acestora, utilizând atât instanțe ale clasei Person cât și instanțe ale clasei Account și a extensiilor acesteia. Cu alte cuvinte, aceasta clasă realizează legătura dintre persoane și conturi. De asemenea, clasa Bank a fost implementată utilizând metoda Design By Contract care presupune utilizarea de preconditionii, postcondiții și assert-uri. Precondițiile și postcondițiile au fost plasate în interfata bankProc iar assert-uri necesare funcționării Design By Contract au fost plasate în metodele descrise în interfata și implementate în clasa Bank. A fost realizat un Observer DP care informează detinatorii contului în legătura cu operațiile executate asupra acestora. Observerul este persoana iar obiectul Observable este un cont. Astfel, a fost implementată metoda update

care este apelata de oricate ori se realizeaza o schimbare in conturi. La nivel de interfata au fost realizati clickListeneri care usureaza intereactiunea utilizatorului cu programul.

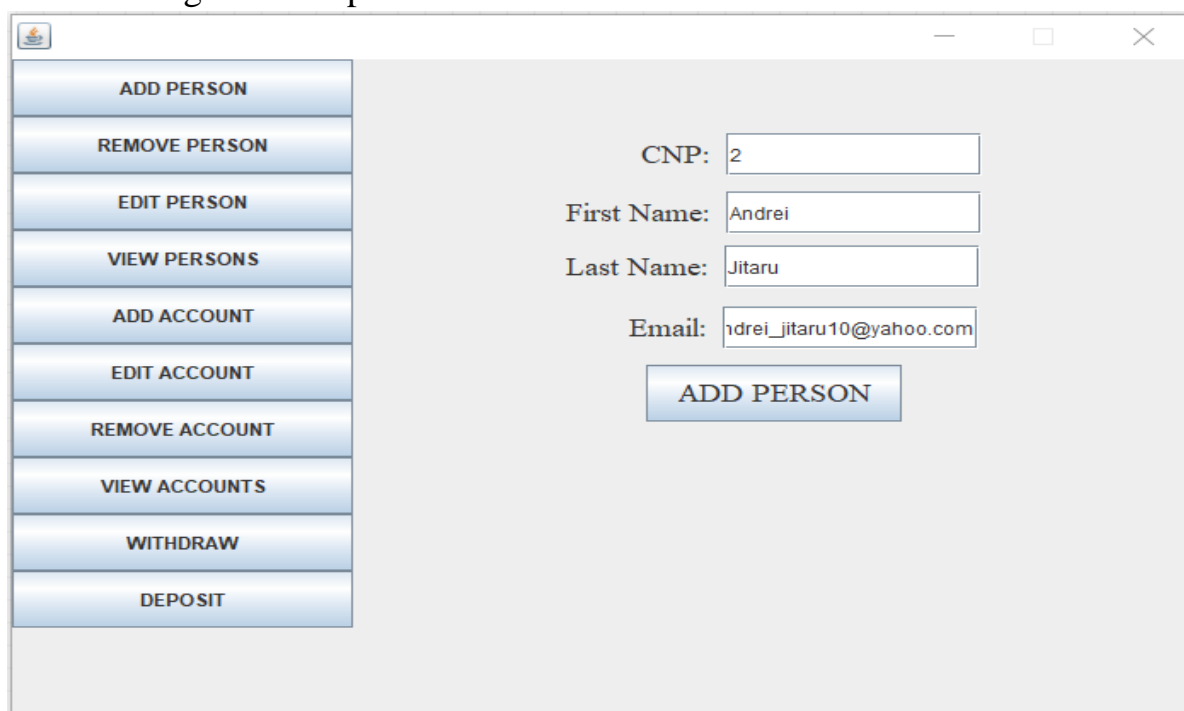
## 5. Testare

Testarea folosește Junit astfel încât fiecare metoda care definește o operație fundamentală este verificată din punct de vedere al funcționării sale corecte în clasa “JUnitTest” prin intermediu câte unui exemplu diferit. Rularea și execuția acestei clase va demonstra cu ușurința funcționarea corectă a operațiilor.

## 6. Rezultate

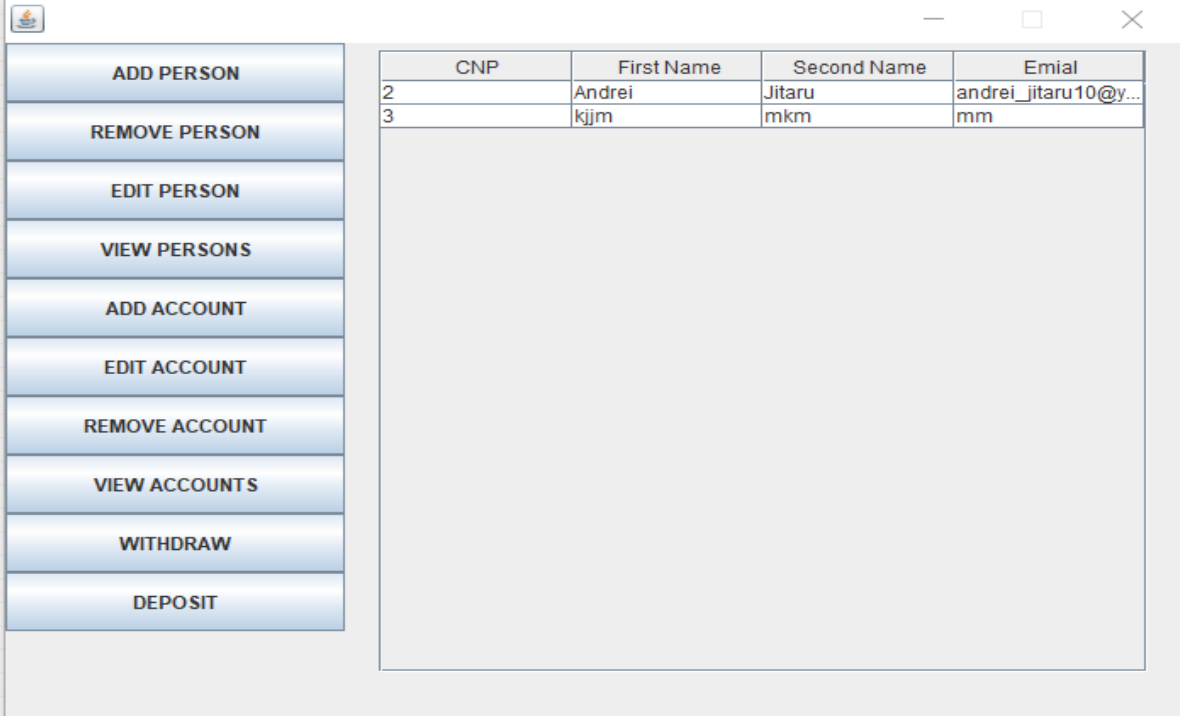
Au fost executate o serie de teste pentru a exemplifica diferitele rezultate care se pot obține în urma interacțiunii utilizatorului cu interfața dar și pentru a scoate în evidență eventualele erori care pot fi afișate.

- Adaugarea unei persoane



The screenshot shows a Java Swing window with a title bar. On the left, there is a vertical menu with buttons for various operations: ADD PERSON, REMOVE PERSON, EDIT PERSON, VIEW PERSONS, ADD ACCOUNT, EDIT ACCOUNT, REMOVE ACCOUNT, VIEW ACCOUNTS, WITHDRAW, and DEPOSIT. The main area of the window contains a form for adding a new person. The form has four labeled text input fields: CNP (containing '2'), First Name (containing 'Andrei'), Last Name (containing 'Jitaru'), and Email (containing 'ndrei\_jitaru10@yahoo.com'). Below these fields is a button labeled 'ADD PERSON'.

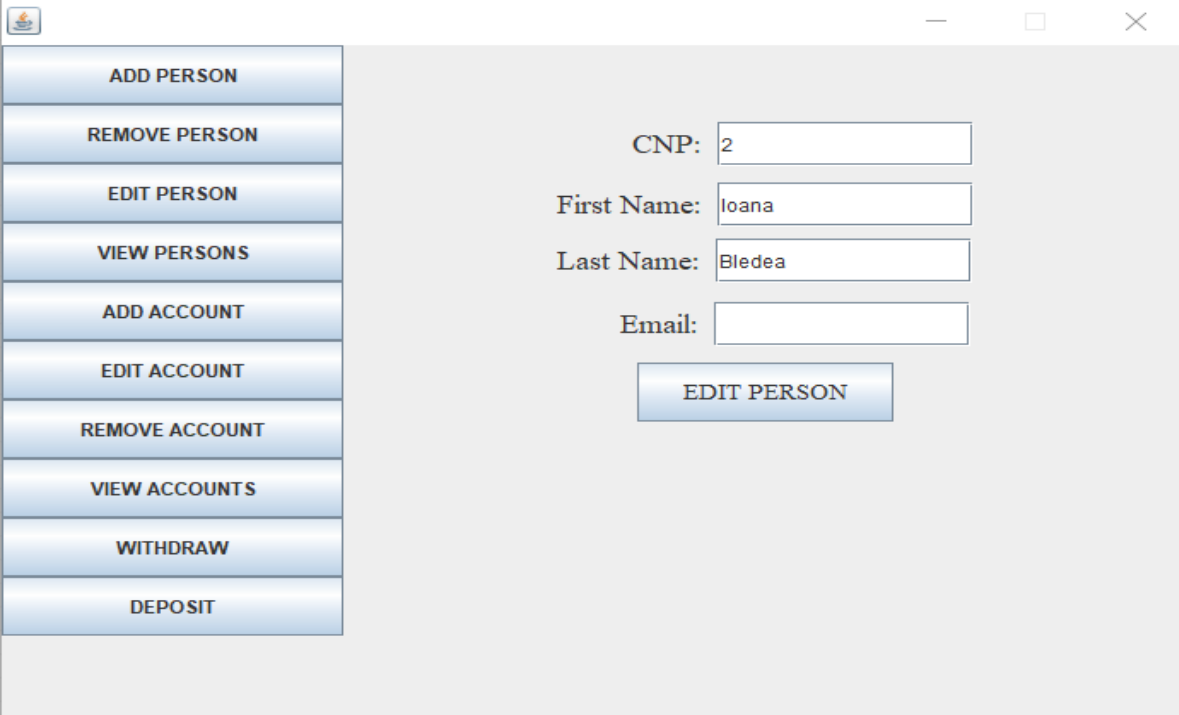
- Vizualizarea tabelului cu persoane



The screenshot shows a web application window with a menu on the left and a table of people on the right. The menu contains the following options: ADD PERSON, REMOVE PERSON, EDIT PERSON, VIEW PERSONS, ADD ACCOUNT, EDIT ACCOUNT, REMOVE ACCOUNT, VIEW ACCOUNTS, WITHDRAW, and DEPOSIT. The table has four columns: CNP, First Name, Second Name, and Email. The data in the table is as follows:

CNP	First Name	Second Name	Email
2	Andrei	Jitaru	andrei_jitaru10@y...
3	kjjm	mkm	mm

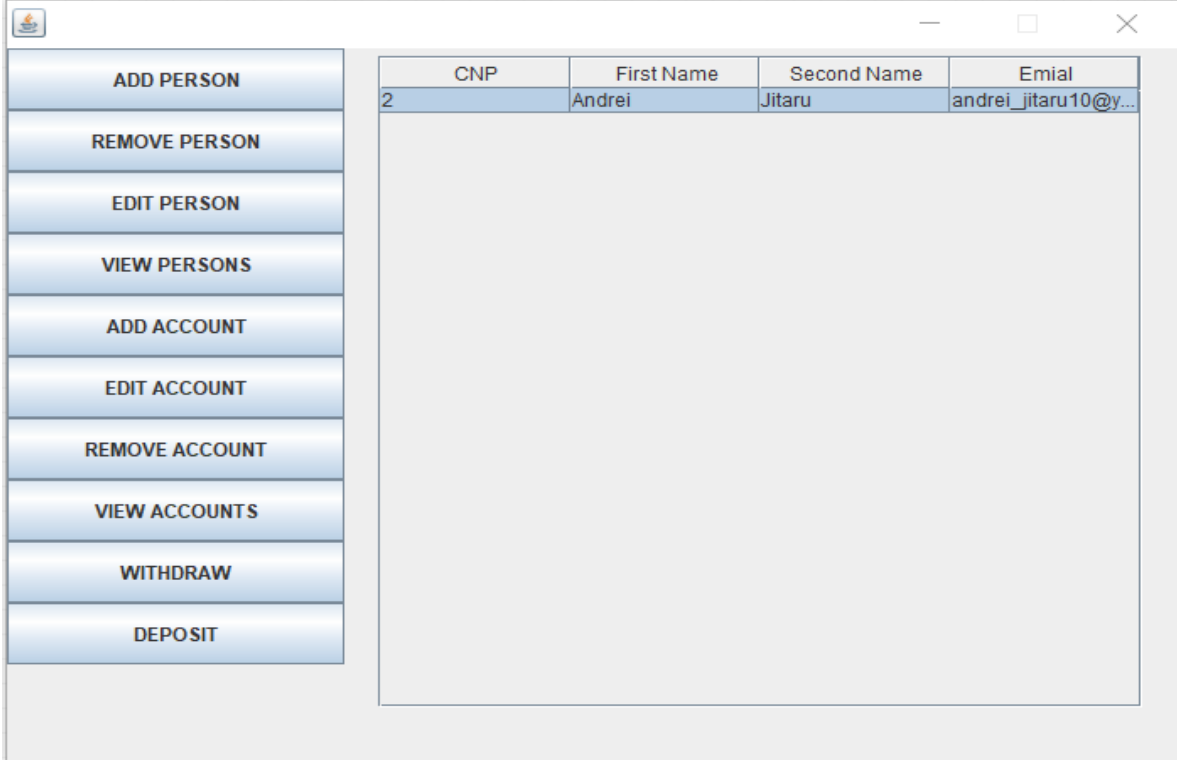
- Editarea unei persoane



The screenshot shows a web application window with a menu on the left and a form for editing a person on the right. The menu contains the following options: ADD PERSON, REMOVE PERSON, EDIT PERSON, VIEW PERSONS, ADD ACCOUNT, EDIT ACCOUNT, REMOVE ACCOUNT, VIEW ACCOUNTS, WITHDRAW, and DEPOSIT. The form has the following fields: CNP (2), First Name (Ioana), Last Name (Bledea), and Email (empty). There is an EDIT PERSON button below the form.

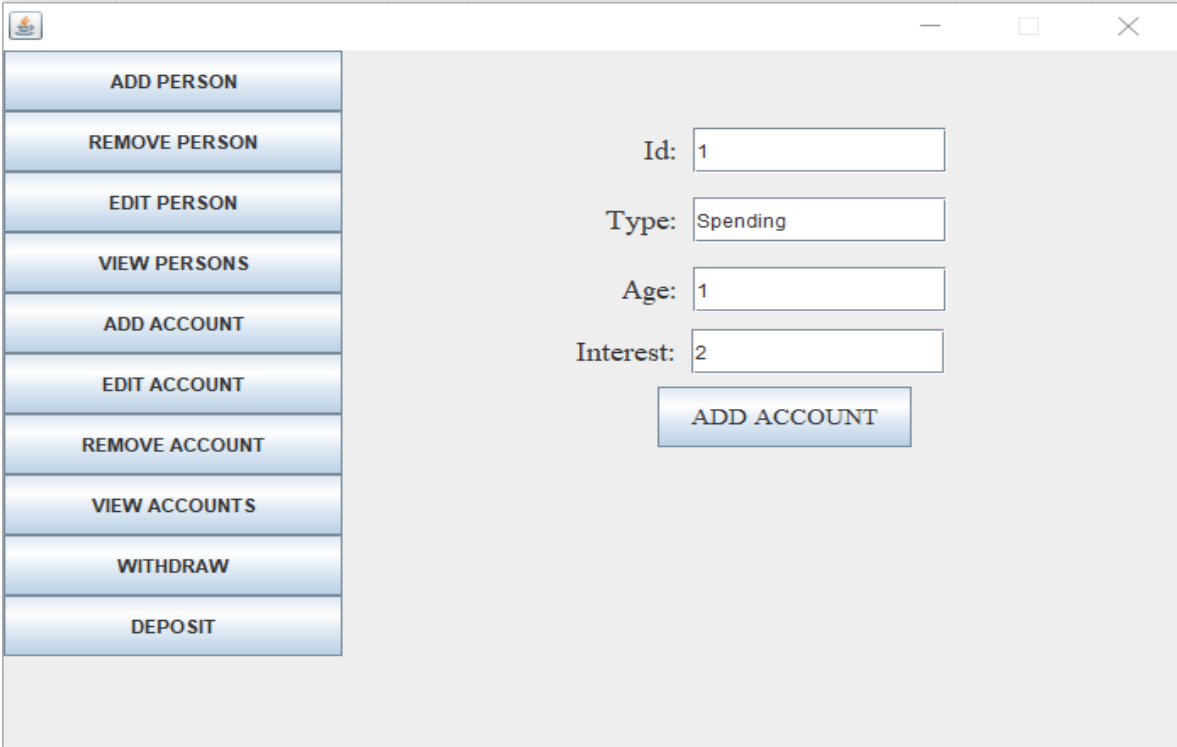


- Adaugarea unui cont



The screenshot shows a web application window with a menu on the left and a table of persons on the right. The menu includes buttons for ADD PERSON, REMOVE PERSON, EDIT PERSON, VIEW PERSONS, ADD ACCOUNT, EDIT ACCOUNT, REMOVE ACCOUNT, VIEW ACCOUNTS, WITHDRAW, and DEPOSIT. The table has columns for CNP, First Name, Second Name, and Email. The first row of the table contains the values 2, Andrei, Jitaru, and andrei\_jitaru10@y...

CNP	First Name	Second Name	Email
2	Andrei	Jitaru	andrei_jitaru10@y...



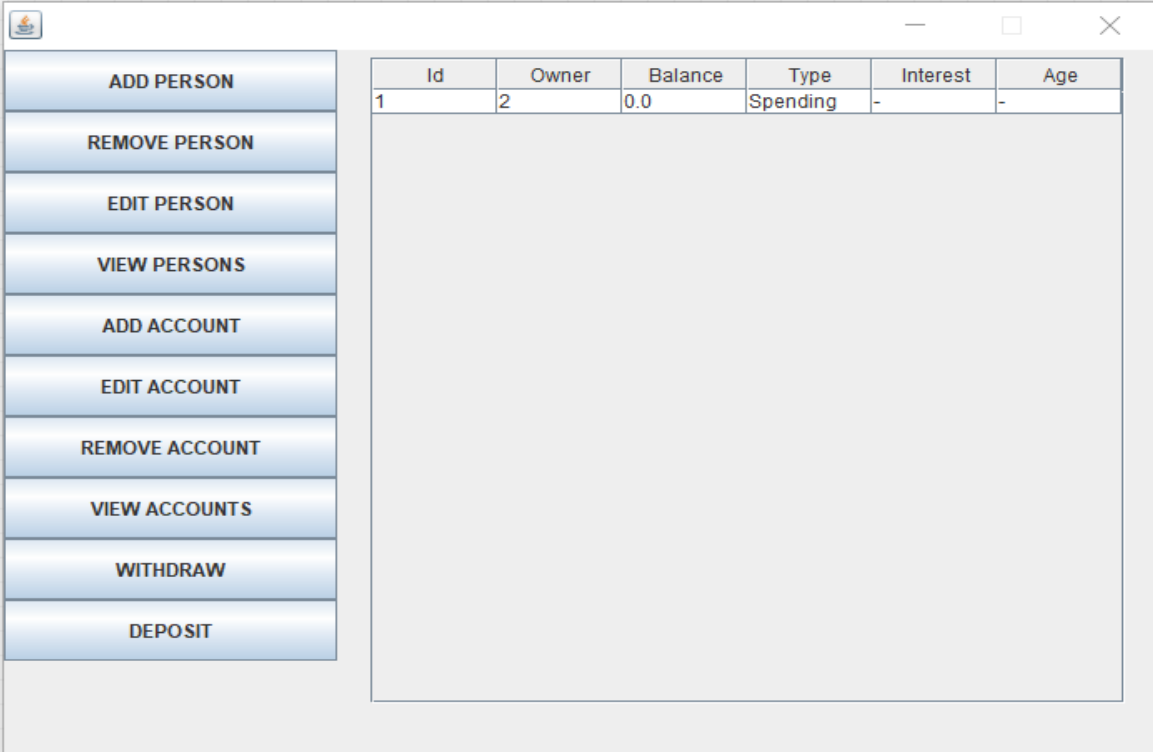
The screenshot shows a web application window with a menu on the left and a form for adding an account on the right. The menu includes buttons for ADD PERSON, REMOVE PERSON, EDIT PERSON, VIEW PERSONS, ADD ACCOUNT, EDIT ACCOUNT, REMOVE ACCOUNT, VIEW ACCOUNTS, WITHDRAW, and DEPOSIT. The form has fields for Id, Type, Age, and Interest, and an ADD ACCOUNT button.

Id:

Type:

Age:

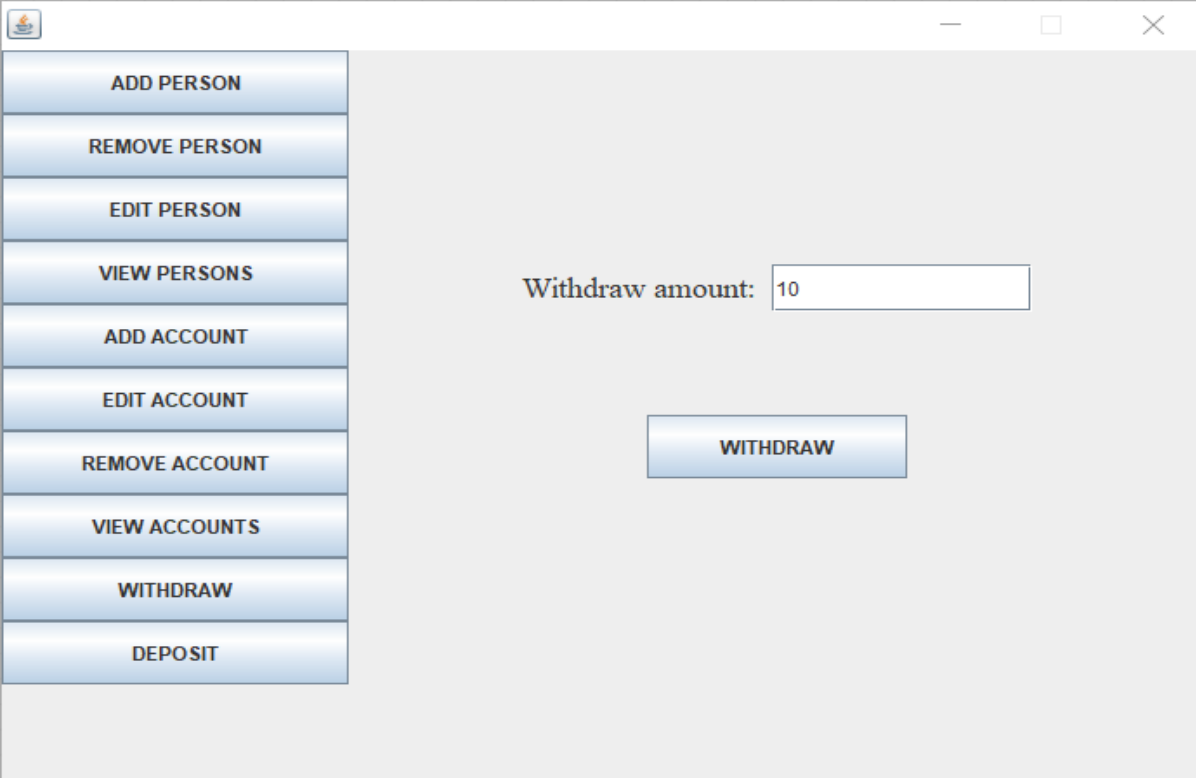
Interest:



The application window features a menu on the left with the following options: ADD PERSON, REMOVE PERSON, EDIT PERSON, VIEW PERSONS, ADD ACCOUNT, EDIT ACCOUNT, REMOVE ACCOUNT, VIEW ACCOUNTS, WITHDRAW, and DEPOSIT. The main area on the right displays a table with the following data:

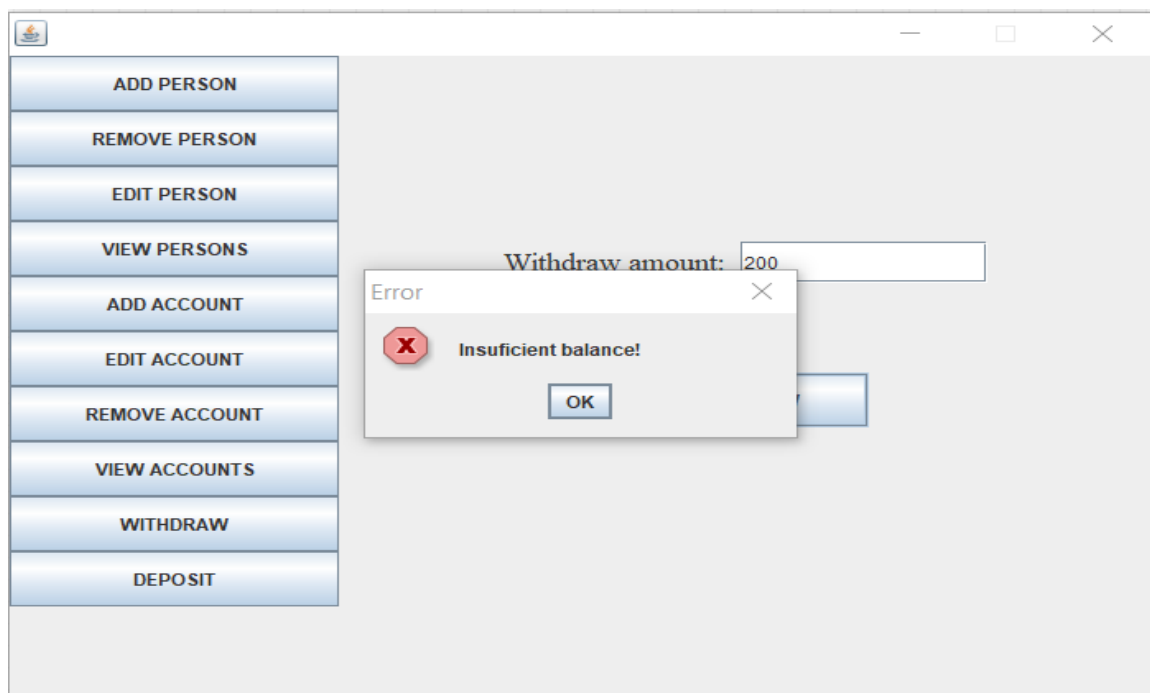
Id	Owner	Balance	Type	Interest	Age
1	2	0.0	Spending	-	-

- Retragerea dintr-un cont

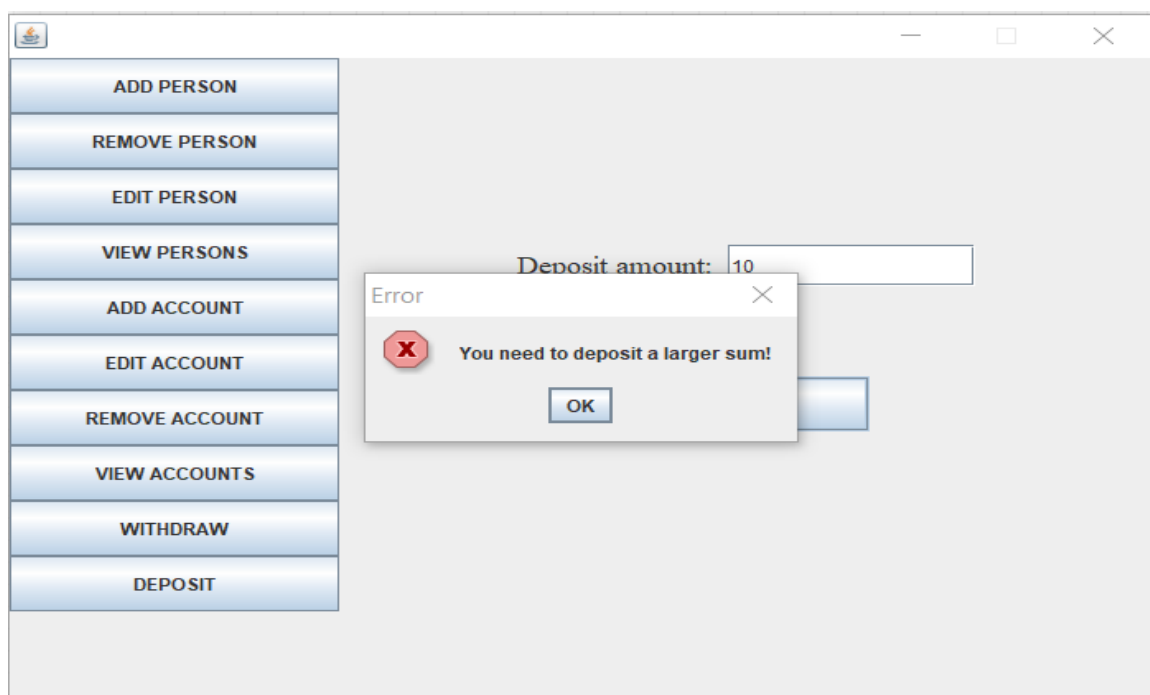


The application window shows the 'WITHDRAW' option selected in the menu. The main area contains a form with the label 'Withdraw amount:' followed by a text input field containing the value '10'. Below the input field is a button labeled 'WITHDRAW'.

- Withdrawal Error 1



- Withdrawal Error 2



## **7. Concluzii si dezvoltari ulterioare**

Sunt de parere ca aplicatia de fata prezinta posibilitatea de a fi utilizata cu usurinta de absolut orice persoana interesata simularea operatiilor realizate la nivelul unei intreprinderi de tip banca. Ca dezvoltari ulterioare as sugera crearea unei interefete utilizator cu culori mai calde si care sa implementeze unele operatii folosind componente Java mai inovitive dintre care amintim comboBoxurile.