**APLICATIE BANCARA**

**BALUS DAN**

**Grupa 30227**

**1. Specificatia Problemei:**

*Cerinta*: Se va proiecta si se va implementa un sistem cu ajutorul caruia vom simula o aplicatie bancara, care ar realiza tranzactiile si operatiile obisnuite dintr-o banca. Se va construi o interfata cu ajutorul careia vom reda utilizatorului cum se desfasoara activitatea intr-o banca.

*Descriere* :

Aplicatia va avea 2 moduri de functionare. O metoda adresata unui client si o metoda adresata unui administrator.

Clientul are urmatoarele optiuni:

- sa creeze o persoana noua, sa se logheze in contul sau

- sa creeze un cont nou de tip Saving sau Spending sa-si editeze datele din cont

- sa faca tranzactii.

Administratorul poate vedea:

- toate persoanele inregistrate in banca

- toate conturile

- toate tranzactiile.

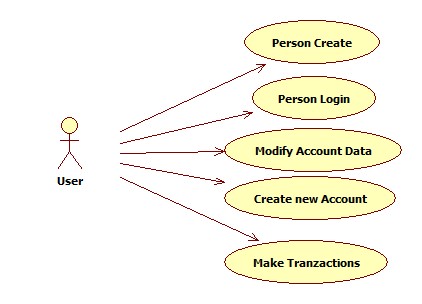
De asemenea administratorul mai poate sa sterge o persoana, un cont sau o tranzactie.

1. Analiza Problemei:

Majoritatea băncilor deţin aplicaţii pe care le folosesc la crearea conturilor şi la salvarea datelor despre clienţi, de asemenea operaţiile bancare din timpul nostru se realizate prin intermediul internetului permit accesul aproape imediat la datele şi soldul contuli nostru aproape din orice colţ al lumii, dacă avem la dispoziţie un calculator conectat la internet.

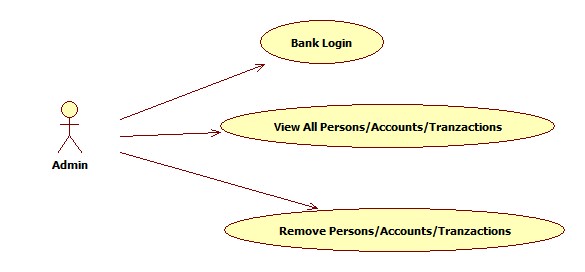
O aplicaţie de acest tip ar trebui sa permit accesul rapid la datele despre clienţi, oferind posibilitatea modificării imediate ale lor în timp real. Utilizatorul poate adăuga, *şterge* clienţi, *efectua* transferuri şi operaţii de *depunere* şi *retragere* dintre conruri.

1. **Diagrama Use-Case:**



Utilizarea aplicatiei: (**Client**)

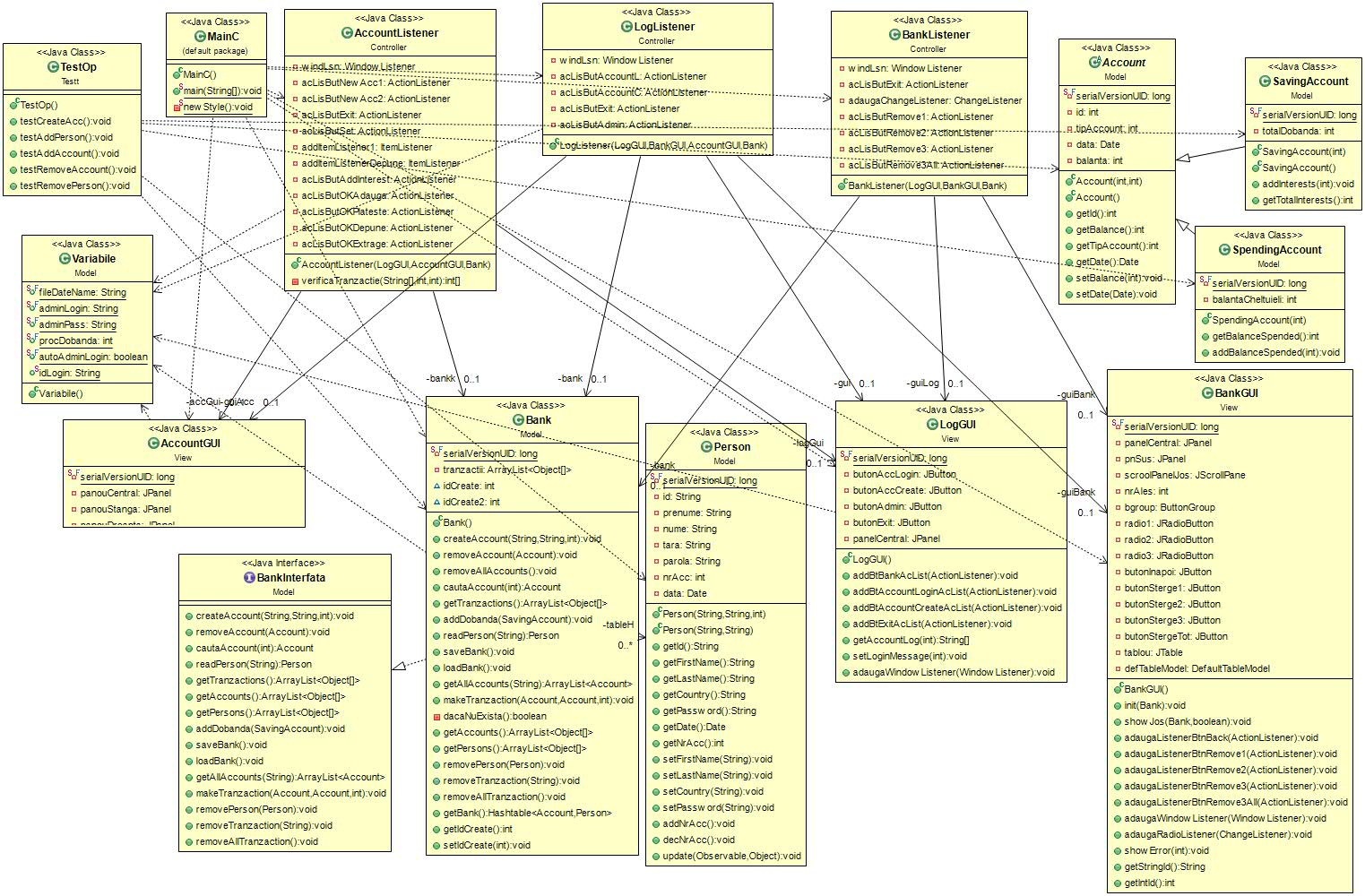
* Clientul poate sa creeze o noua persoana
* Clientul poate sa se logheze in propriul cont (persoana)
* Clientul poate sa modifice datele contului personal
* Clientul poate sa creeze mai multe conturi bancare, toate vor fi inscrise in numele sau.
* Conturile sunt de 2 feluri: Cont de economii si de cheltuieli.
* De pe contul de economii: Clientul poate sa-si scoata banii din contul de economii pe un alt cont de cheltuieli.
* De pe contul de economii: Clientul poate sa ceara si sa vada dobanda din contul de economii.
* De pe contul de cheltuieli: Clientul poate sa-si depuna bani pe cont.
* De pe contul de cheltuieli: Clientul poate sa faca o cheltuiala.
* De pe contul de cheltuieli: Clientul poate sa faca o tranzactie catre propriile conturi sau catre alte conturi.



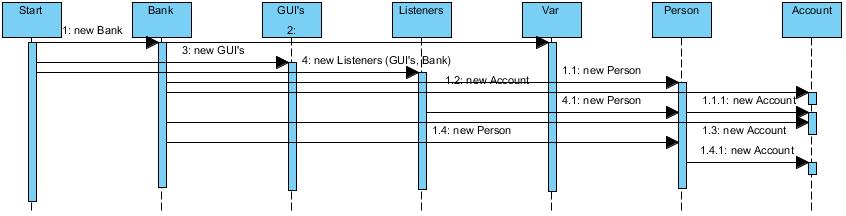
Utilizarea aplicatiei: (**Administrator**)

* Administratorul poate sa vada toate persoanele cu datele sale
* Administratorul poate sa vada toate conturile cu informatii despre ele
* Administratorul poate sa vada toate tranzactiile
* Administratorul poate sa stearga o persoana cu toate conturile sale
* Administratorul poate sa stearga un cont
* Administratorul poate sa stearga tranzactiile.

1. Diagrama UML:



1. Diagrama de secventa:



1. Implementare:

În cele ce urmează , vom prezenta amănunțit tehnica de implementare, atributele și metodele utilizate pentru soluționarea problemei, referindu-ne la fiecare clasă în particular.

Intrucat programul este folosit atat pentru administrator cat si pentru client, se vor implementa mai multe ferestre grafice.

**Prima** fereastra grafica va afisa modul de logare in program: administrator sau client, a **doua** fereastra va arata posibilitatile unui client de a folosi programul, o a **treia** fereastra grafica va exista pentru administrator, astfel incat el sa poata face toate operatiile in calitate de administrator. Fiecare fereastra va avea **Listener-uri**: de la apasarea unui buton, selectarea unui index, pana la inchiderea unei ferestre, toate interfetele grafice trebuie sa aiba listeneruri. Ca sa fie intr-un mod mai elegant, vom crea 3 listener-uri: cate unul pentru fiecare.

In pachetul model vom avea o clasa cele mai importanta: Bank pe care o vom serializa, adica datele respective vor fi salvate intr-un fisier la parasirea programului, iar la repornirea acestui program, datele vor fi citite din acel fisier, astfel incat folosirea programului sa fie intr-un mod elegant. Clasa Bank va implementa interfata clasei BankInterfata, care o sa contina metodele obligatoriu necesare pentru existenta unei banci (in limitele imaginatiei proprie).

Clasa Bank va contine un tabel Hashtable creat din 2 campuri: Account si Person. Astfel, pentru fiecare cont se va atribui o persoana. Tot in acest pachet, avem aceste 2 clase: Account si Person.

Clasa Account contine date precum ID-ul contului, tip-ul contului, soldul, data inregistrarii.

Clasa Person va contine date precum ID-ul unic al persoanei, Data crearii persoanei, numele, prenumele, parola, tara din care face parte. Intrucat avem 2 tipuri diferite de conturi: SavingAccount si SpendingAccount am creat in pachetul Model inca 2 clase.

Clasa Person implementează și interfața Observer , deoarece aplicația folosește principiul de Observer , clienții fiind observatori pentru propriile conturi.

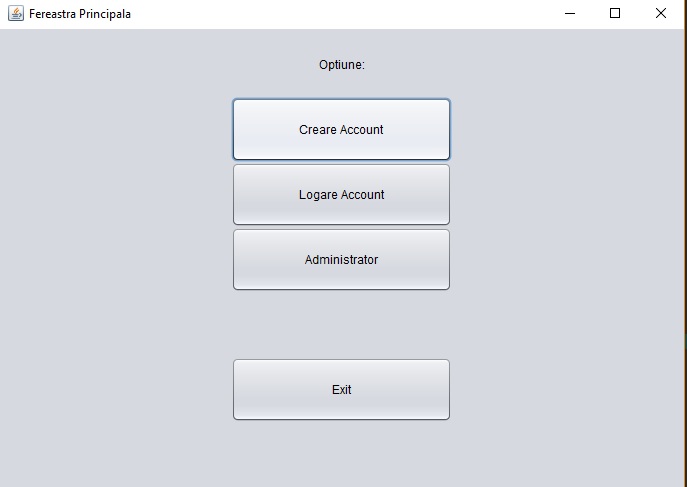
Mai avem, de asemenea, clasa TestOp care testeaza o parte din metodele proiectului folosint Junit 3.

Ambele clase mostenesc clasa Account, totusi se diferentiaza prin unele date. SavingAccount are ca propriile date: cantitatea de bani castigata ca dobanda, iar SpendingAccount cantitatea de bani cheltuita. Tot in aceast pachet mai avem o clasa Variabile care contine variabile statice precum: numele si parola administratorului, procentajul dobanzii, etc. Aceste date sunt statice intrucat vor fi folosite de celelalte clase si nu se vor schimba in mare parte pe parcursul intregului program.

Programul mai contine un pachet: View in care avem salvate 3 clase de tip GUI: Graffic User Interface. Acolo avem interfetele grafice pentru Utilizator, Administrator si Client. Utilizatorul alege un mod de utilizare intre cele 2: Administrator si Client. Aceaste clase afiseaza doar interfata grafica, operatiile principale fiind facute in clasele din pachetul Controller.

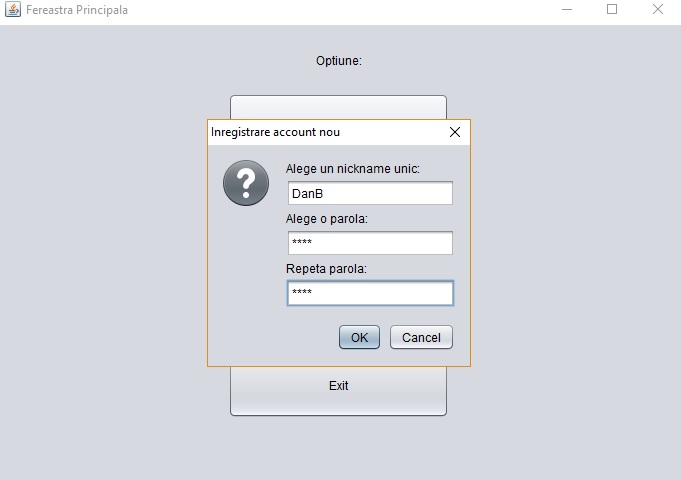
1. Lucru cu interfata:

1)



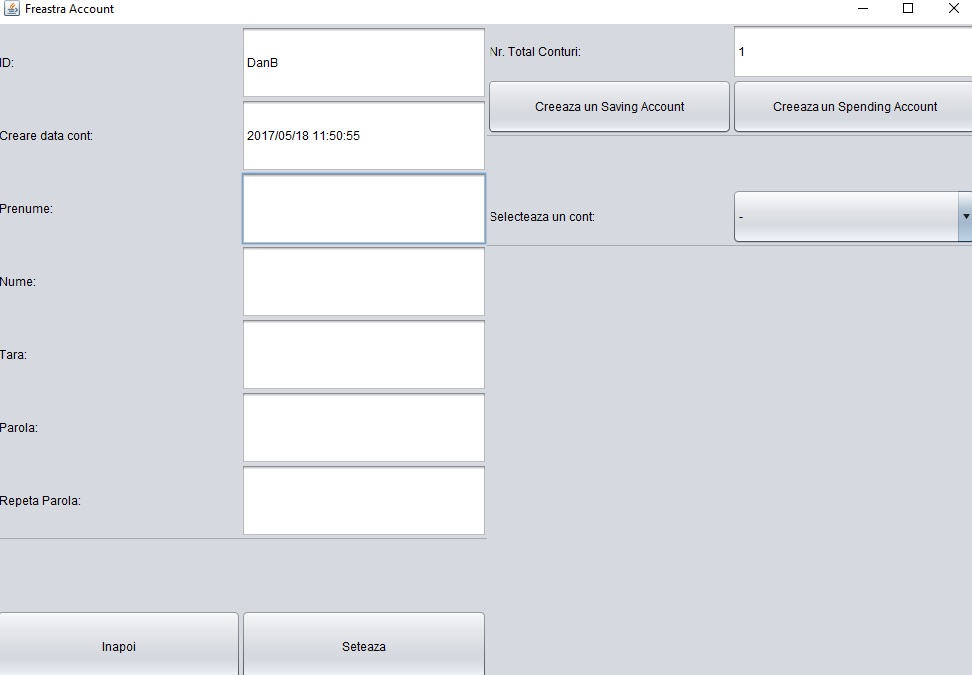
Cand se porneste aplicatia utilizatorul are 4 butoane: „Creare Account”, „Logare Account”, „Administrator” si „Exit”. Primul buton este pentru crearea unui nou cont, adica o persoana. Al doilea buton este pentru logarea unei persoane, iar al treilea buton este pentru administratorul bancii. Ultimul buton paraseste programul, nu inainte insa, de a salva toate datele bancii. De asemenea, cand se iese fortat din program (apasand butonul x din colt) toate datele se vor salva.

2)



In acest exemplu a fost apasat butonul „Creare Account”. Dupa cum vedem, a aparut o fereastra cu 3 campuri: primul camp este folosit pentru a citi un nume unic al contului (a persoanei). Acest camp unic, poate fi folosit si in calitate de CNP (de exemplu). In cazul in care exista deja o persoana cu acest camp, programul va afisa o eroare, si nu va permite inregistrarea acestuia. Campul 2 cere o parola, din cel putin 4 caractere. Campul 3 cere aceeasi parola. In cazul in care campul 2 si 3 nu coincid, se va afisa de asemenea o eroare. In cazul inregistrarii cu succes, fereastra LoginGUI va disparea (se va ascunde) si va aparea o noua fereastra: Account GUI (care de fapt exista dar era ascunsa).

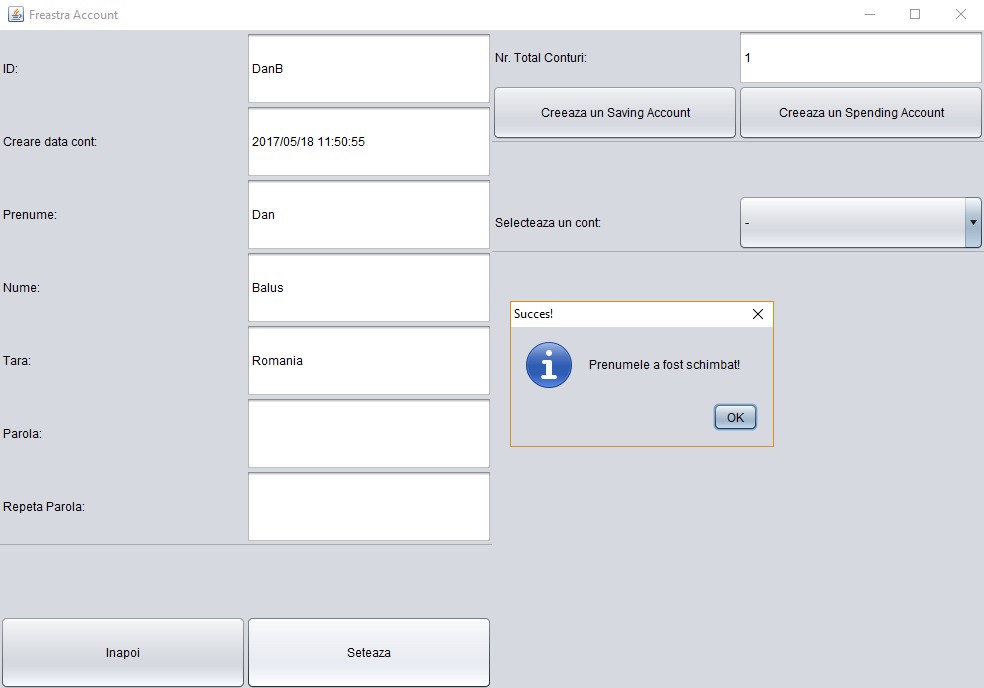
3)



In acest exemplu a aparut fereastra AccountGUI. Ea poate aparea in 2 cazuri: cand un utilizator se logheaza cu succes, sau isi face un cont cu succes. Aceasta fereastra, dupa cum vedem are cateva campuri in stanga cu datele personale ale utilizatorului. Campul ID (care nu poate fi editat) contine numele unic al utilizatorului. Campul „Creare data cont” (care nu poate fi editat) contine data in care a fost creat contul. Urmatoarele campuri: Prenume, Nume, Tara si Parola pot fi editate in orice moment. Butonul „Seteaza” se apasa pentru a confirma schimbarea noilor campuri. Butonul „Back” este pentru a ajunge din nou la fereastra LoginGUI.

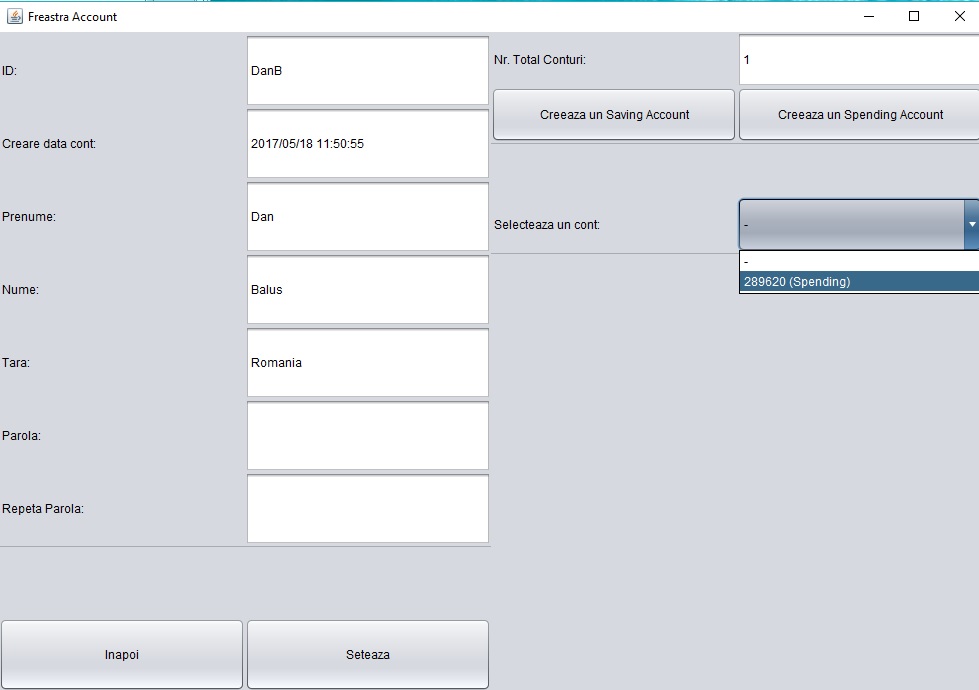
In partea dreapta avem un camp „Nr total conturi” (care nu poate fi editat) in care utilizatorul este informat cu numarul de conturi bancare pe care le are. Cand se creeaza contul personal, utilizatorul are deja si un cont bancar de tip Spending. Mai jos avem 2 butoane: „Creeaza un Saving Account” si „Creeaza un Spending Account”, fiecare poate fi folosit de cate ori are nevoie utilizatorul sa-si creeze conturi. Apasand pe unul din aceste butoane, se creeaza un nou cont pe numele acestui utilizator. Campul „Select an account” va fi studiat mai jos.

4)



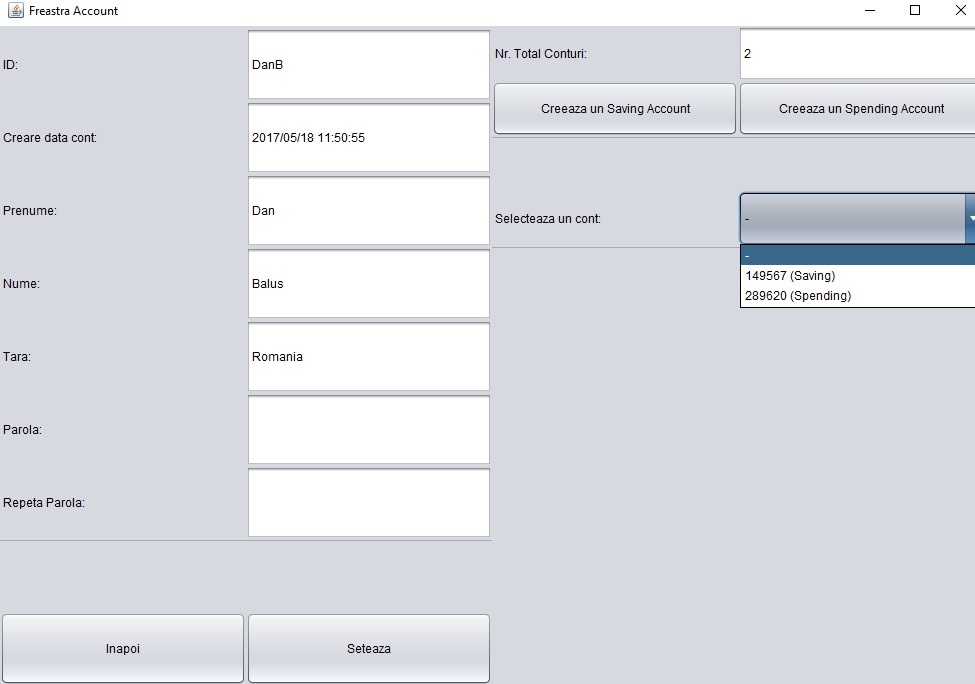
In acest exemplu, a fost apasat butonul „Seteaza”. Pentru fiecare camp care a fost editat apare un nou mesaj: „Prenumele a fost schimbat!” in cazul schimbarii cu succes, sau un mesaj de eroare in caz de eroare. Daca se doreste a schimba parola, ea trebuie scrisa in ambele campuri si sa fie aceeasi.

5)



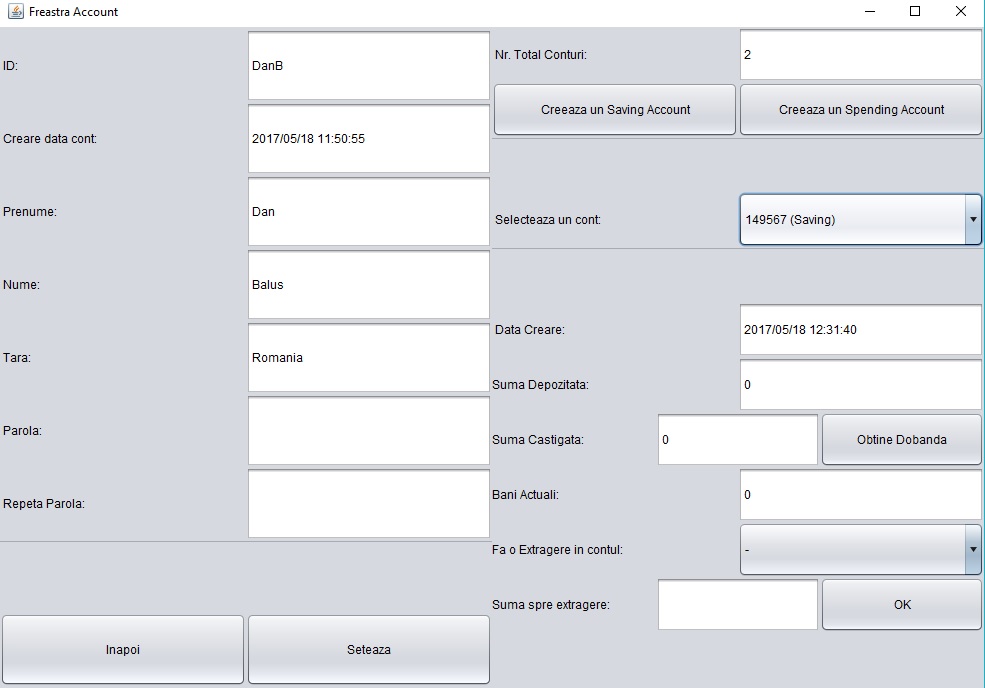
Dupa cum vedem, putem selecta mai jos un conturile bancare pentru a face operatii si a vedea informatiile din cont. La prima creare a contului personal, intr-un mod automat se creeaza si un cont bancar. Fiecare cont bancar are un ID unic generat aleator. In numele contului care urmeaza a fi selectat, se contine si tipul contului, care poate fi Spending sau Saving.

6)

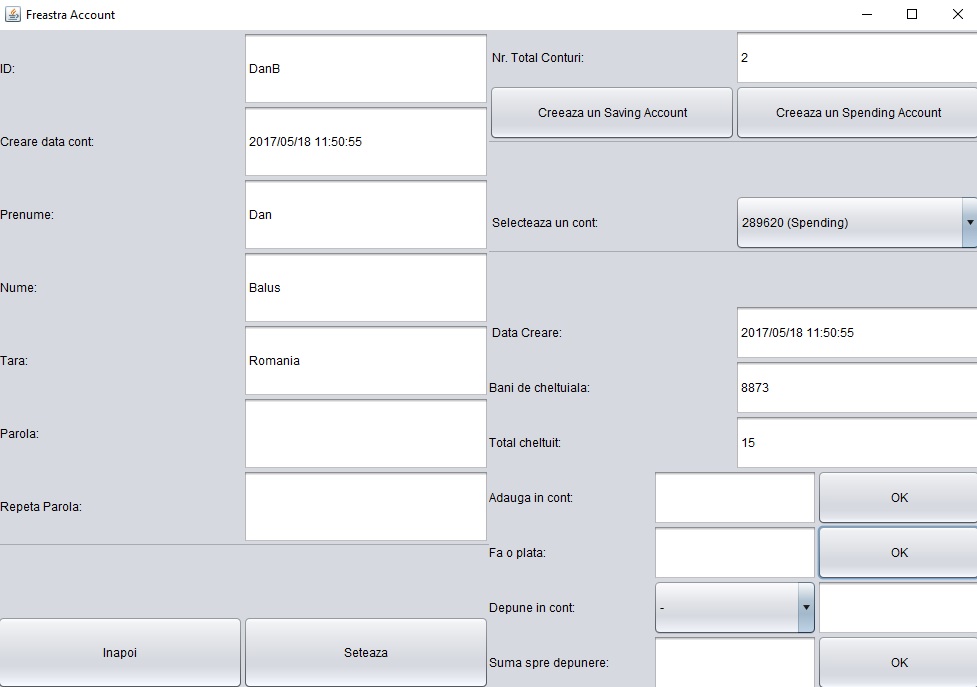


In acest exemplu, a fost apasat de odata pe butonul „Creeaza un Saving Account” si de 1 pe butonul „Creeaza un Spending account”.

7)



8)



In acest exemplu dorim sa vizualizam un cont de tip Spending.

Avem cateva campuri: „Data creare” unde avem data crearii acestui cont. Acest camp nu poate fi editat.

Avem campul „Suma depozitata” cu suma de bani disponibila acestui cont. Acest camp nu poate fi editat.

Mai jos avem campul „Total cheltuit”. Acest camp contine suma de bani care a fost scoasa din acest cont (scoasa pentru a face o plata, nu pentru depuneri sau tranzactii). Acest camp, de asemenea, nu poate fi editat.

Mai jos avem campul „Adauga in cont”, urmat de un buton „OK”. In momentul apasarii acestui buton, se adauga suma din campul alaturat in „bani de cheltuiala”. Astfel creste soldul acestui cont.

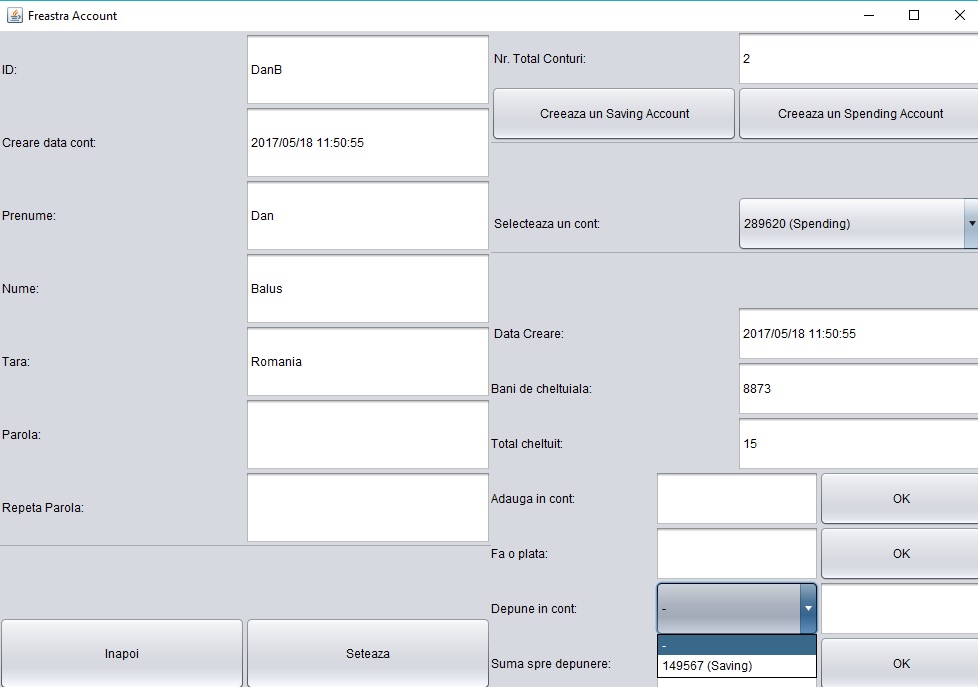
Mai jos avem un nou camp „Fa o plata”, urmat de un buton „OK”. In moemntul apasarii acestui buton, se scade din soldul acestui cont, suma alocata alaturi. In cazul in care contul nu dispune de bani suficienti, va aparea un mesaj de eroare, iar tranzactia nu va avea loc.

Mai jos avem un nou camp: „Depunere in cont”.

Toate conturile proprii pot fi selectate direct de catre un element ComboBox alaturat, si id-ul apare in campul mai la dreapta. Putem si manual sa scriem un id al unui alt cont in acel camp. Astfel putem transmite bani catre orice cont valid.

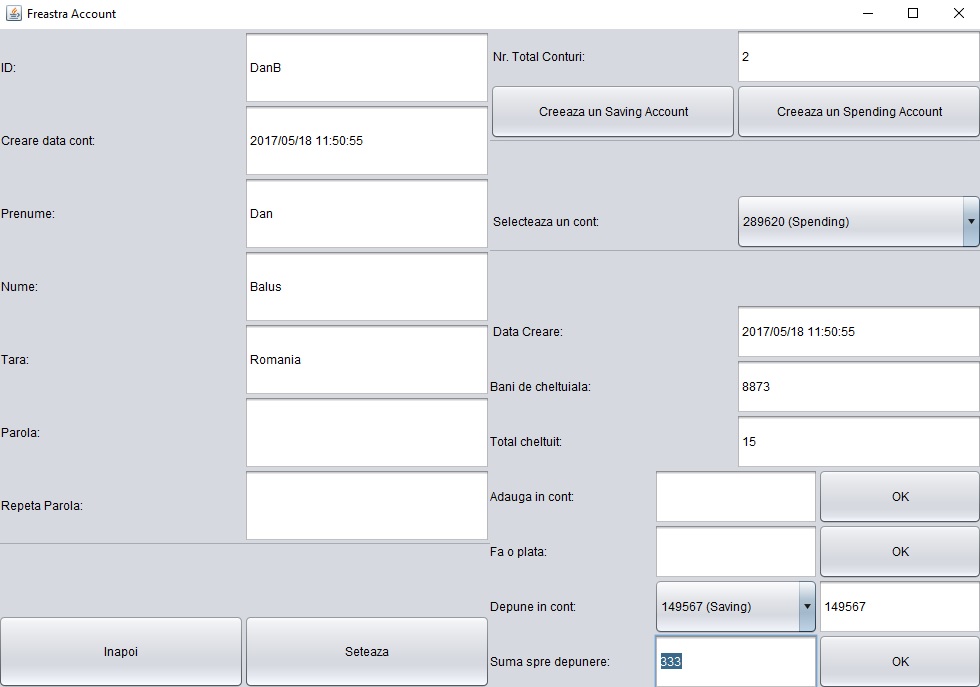
In acest exemplu, am depus in cont 8888 de lei in campul „adauga in cont. Si dorim sa facem o plata de 15 de lei. Dupa ce vom apasa OK, vom avea ca bani de cheltuiala doar 8873,si bani cheltuiti in total 15 .

9)



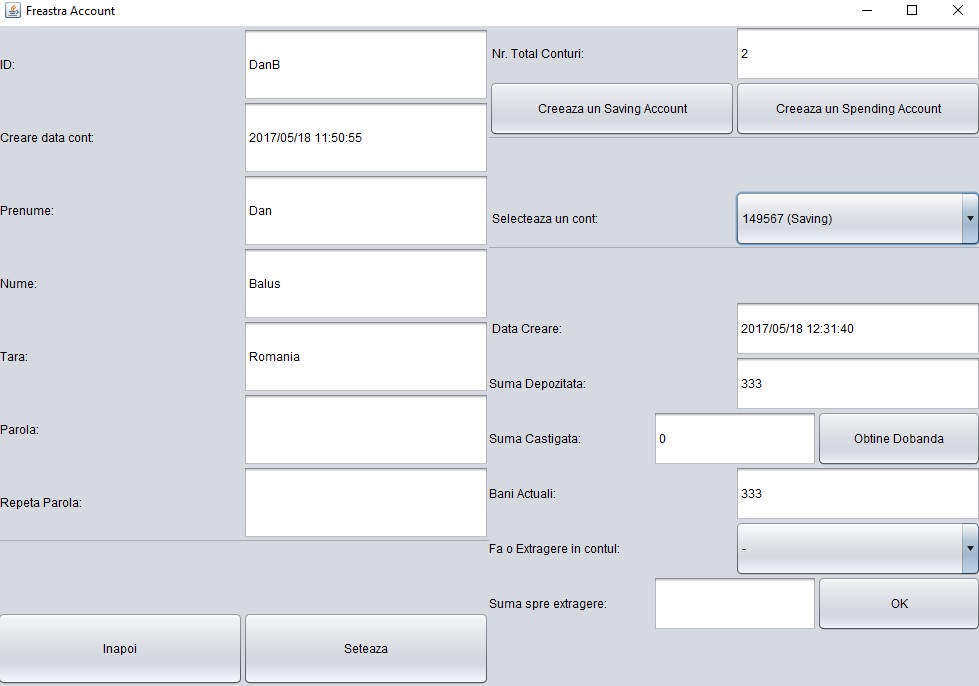
Dupa cum vedem, putem depune in cont in unul din celelalte 3 conturi personale, sau daca stim un ID de cont al altei persoane, il putem scrie in campul de alaturi.

10)



Dupa ce am selectat contul 149567 care e de tip Saving, in rubrica din dreapta a aparut acest id al contului in mod automat. Putem schimba aceasta rubrica cu orice alt id dorim noi sa fie. In cazul in care acest id este valabil, se vor trimite banii catre acel cont, in caz nevalabil se va afisa o eroare. In exemplu nostru, dorim sa transmitem 333 de lei din cei 8873 disponibili catre un cont de economii: saving account cu numarul 149567.

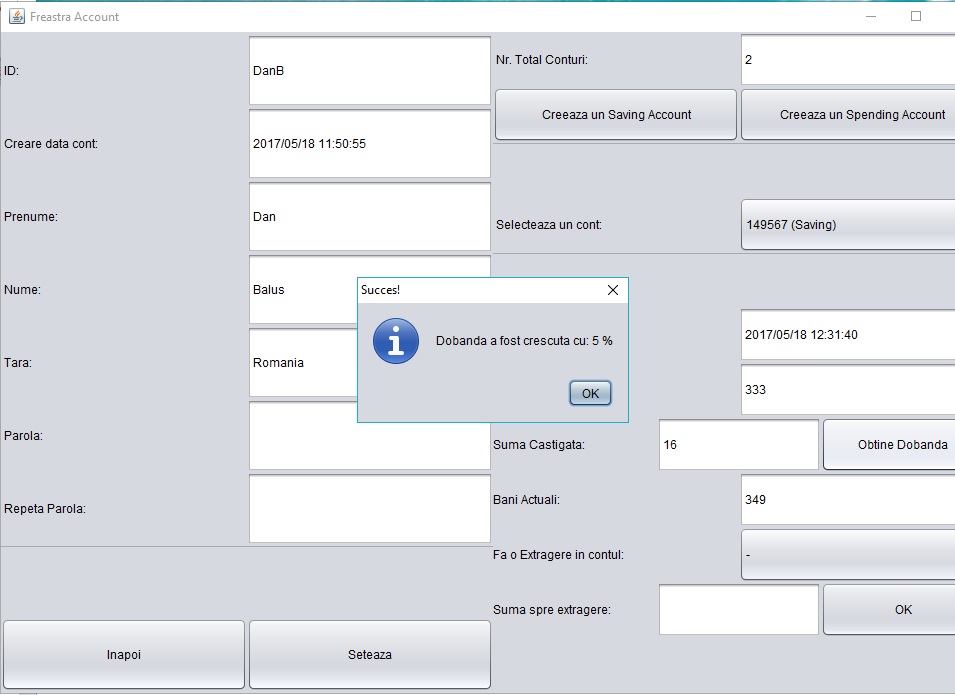
11)



In exemplu alaturat am intrat in contul 14567 al aceluiasi proprietar: DanB cu numele Dan Balus, pentru a vedea daca banii au ajuns. Dupa cum vedem un cont de tip Saving Account are alte rubrici decat un cont de tip Spending Account. Rubrica Created date coincide ca la Spending Account. Urmatoarea rubrica: Bani depozitati: avem banii depozitati in acest cont. Aceasta rubrica nu poate fi editata. Mai jos avem „Suma castigata”. Aici vor fi afisati suma de bani castigati ca dobanda. Urmatorul buton „Obtine dobanda” va adauga in acest cont o dobanda de 5%. Aceasta rubrica nu poate fi editata. Mai jos avem banii actuali care se calculeaza din suma banilor depozitati si cei castigati din dobanzi. Mai jos avem o rubrica „Fa o extragere in contul” si suma respectva care urmeaza a fi extrasa. Dupa cum vede, spre deosebire de un cont de Spending, acest cont este unul de economii si nu pot fi facute plati directe, si nici incasare a banilor direct, decat doar sa fie transmisi din alt cont.

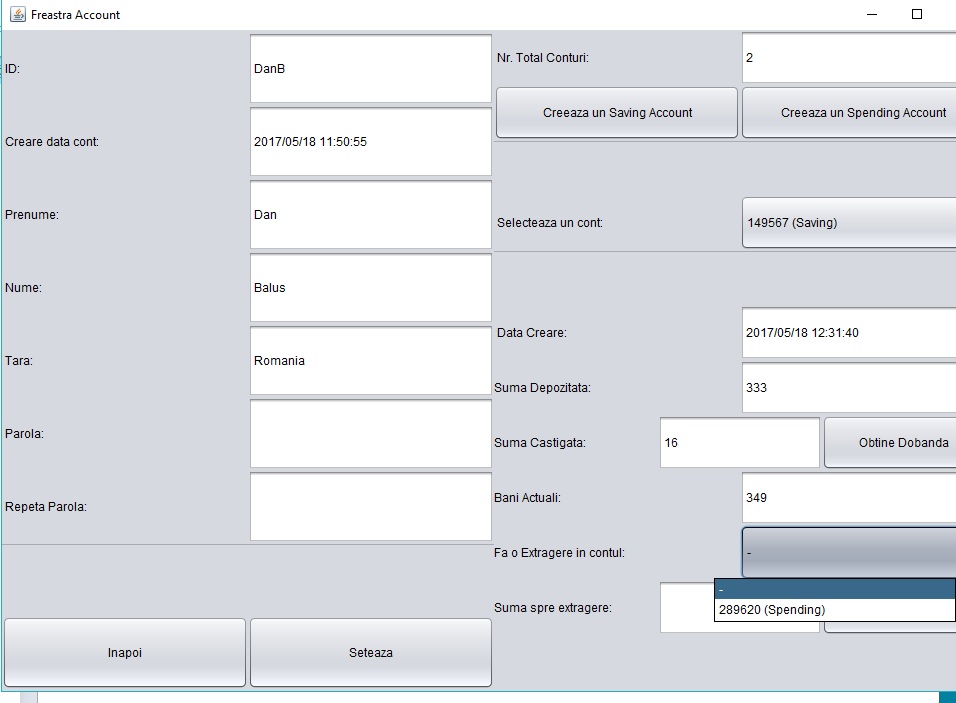
12)

”



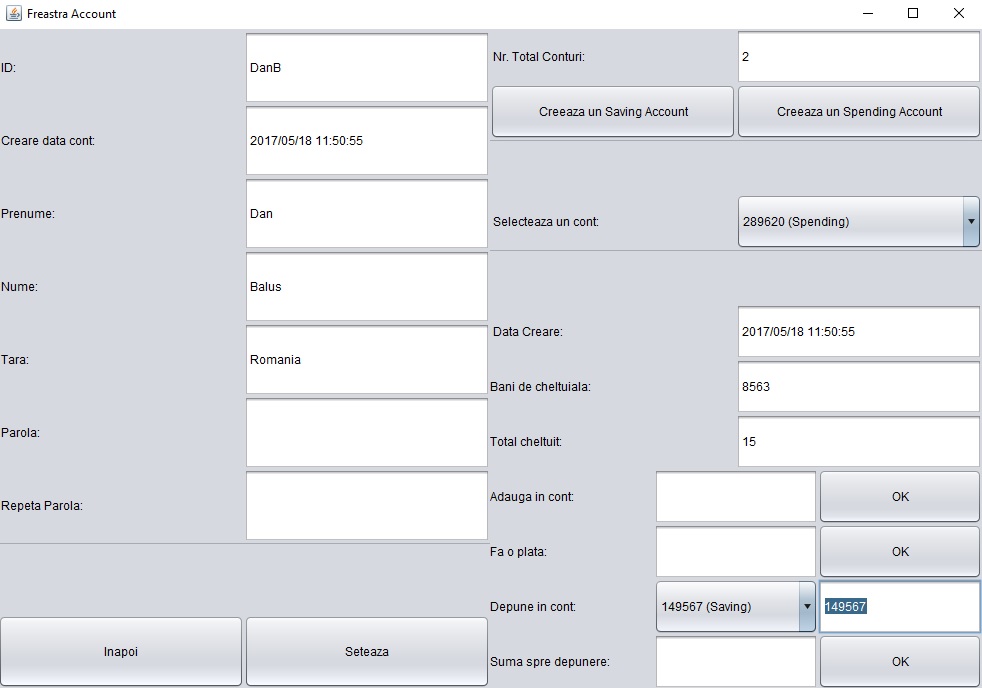
Aici a fost apasat butonul „Obtine Dobanda

13)



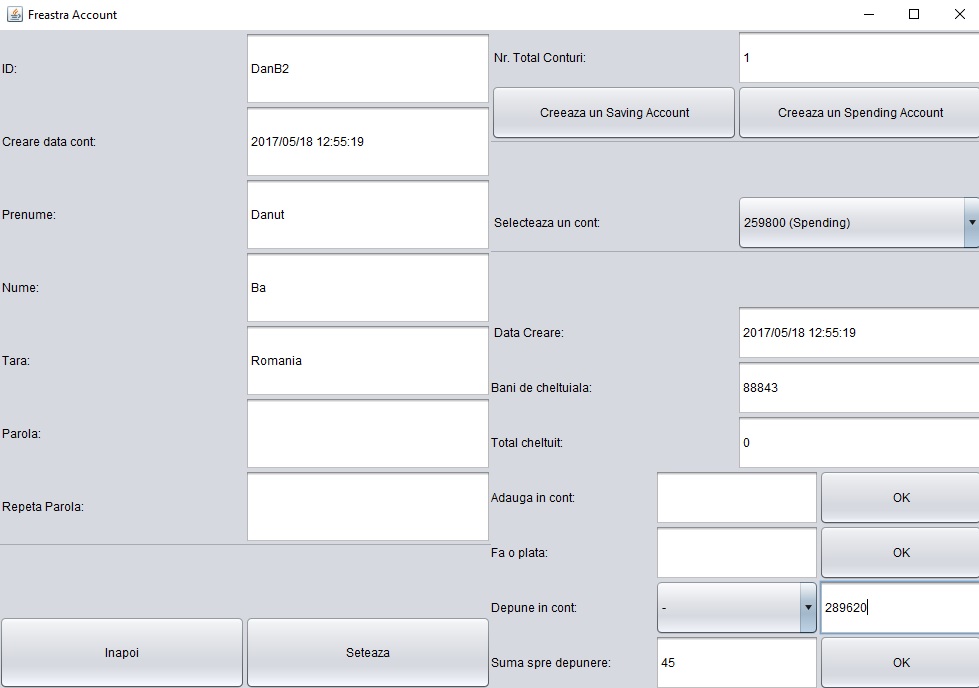
Dupa cum vedem, extragerea din acest cont se poate face doar catre alte conturi de tip „Saving”. Mai mult ca atat, extragerea se face doar catre conturile proprii, nu si catre alte conturi straine, precum se putea face la un cont de tip Spending. Vom incerca sa transmitem banii din acest cont catre ultimul cont de Spending: 289620.

14)



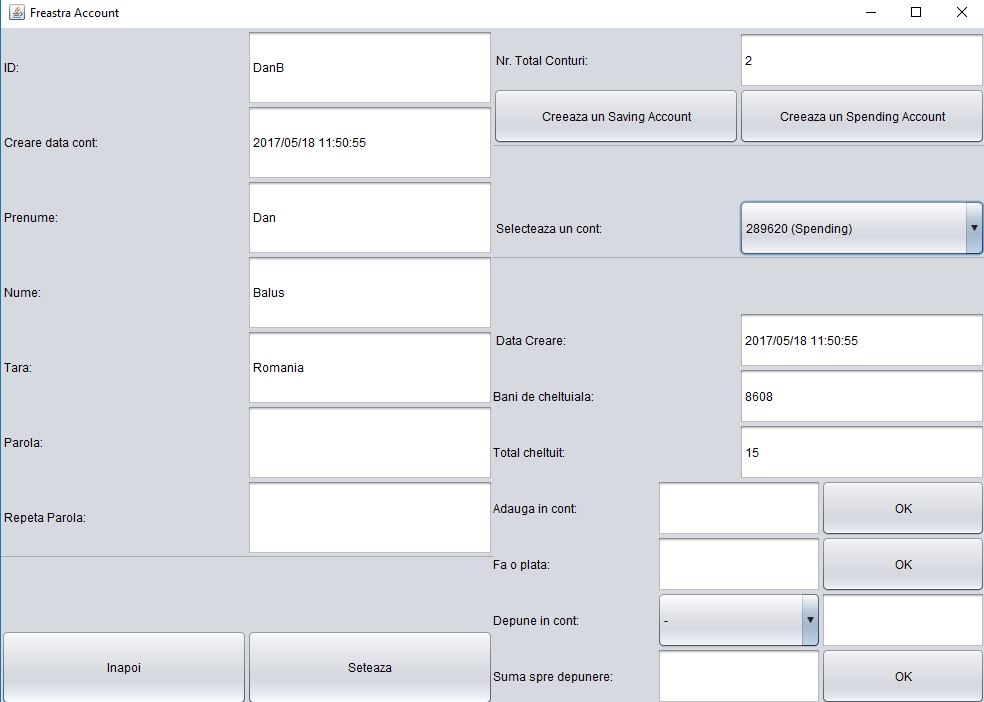
In exemplul alaturat am selectat acest cont 149567 pentru a urmari daca s-au transmis banii din contul de economii catre contul curent.

15)



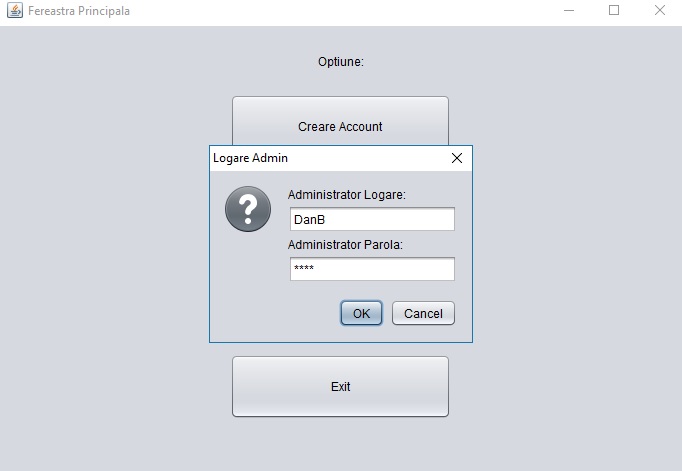
In exemplul alaturat, am creat un nou cont, cu ID-ul unic DanB2, avand numele Danut Ba. Am depus pe acest cont 88888 de lei ca bani de cheltuiala si incercam sa transmitem in contul ilustrat din exemplul de mai sus o suma de 45 lei. Contul de mai sus, avea ID-ul 289620.

16)



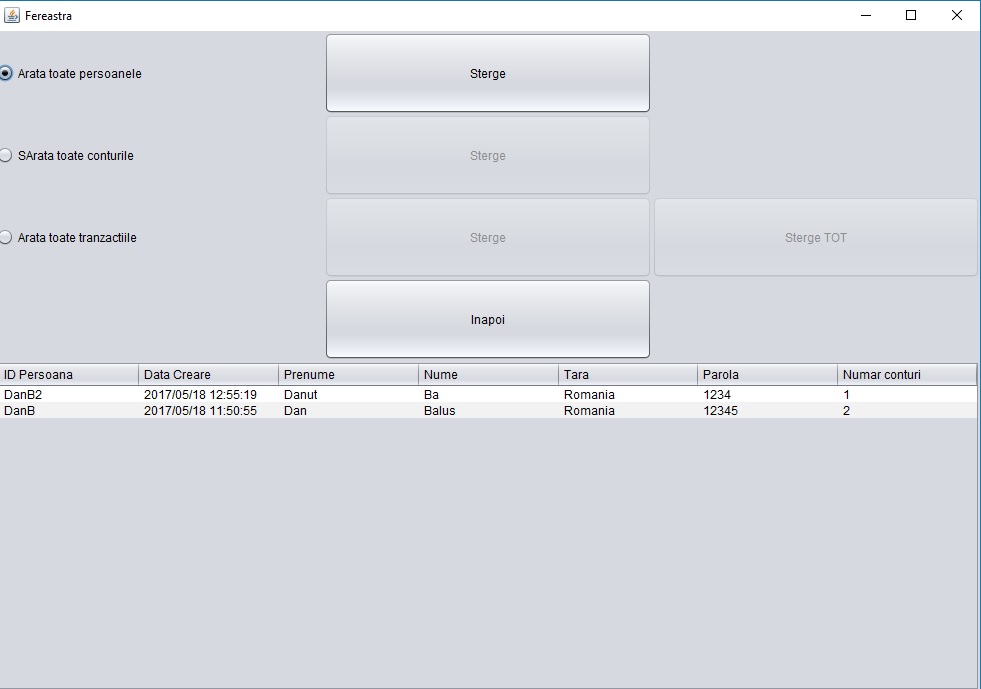
Aici ne-am logat inapoi cu contul DanB si-am selectat contul 289620 pentru a urmari daca acesti bani au ajuns cu bine. Intr-adevar, acum avem 8608 lei in cont.

17)



In acest exemplu vom incerca sa ne logam ca administratorul bancii. Logarea se face din interfata grafica Log GUI. Drept intrebare ne apare o fereastra cu loginul si parola administratorului. In programul nostru exista un singur login si o singura parola de administrator care este setata implicit in clasa Variabile din pachetul Model. In cazul in care utilizatorul nu o pune corect, va aparea un mesaj de eroare, astfel nu va avea acces catre aceasta clasa.

18)

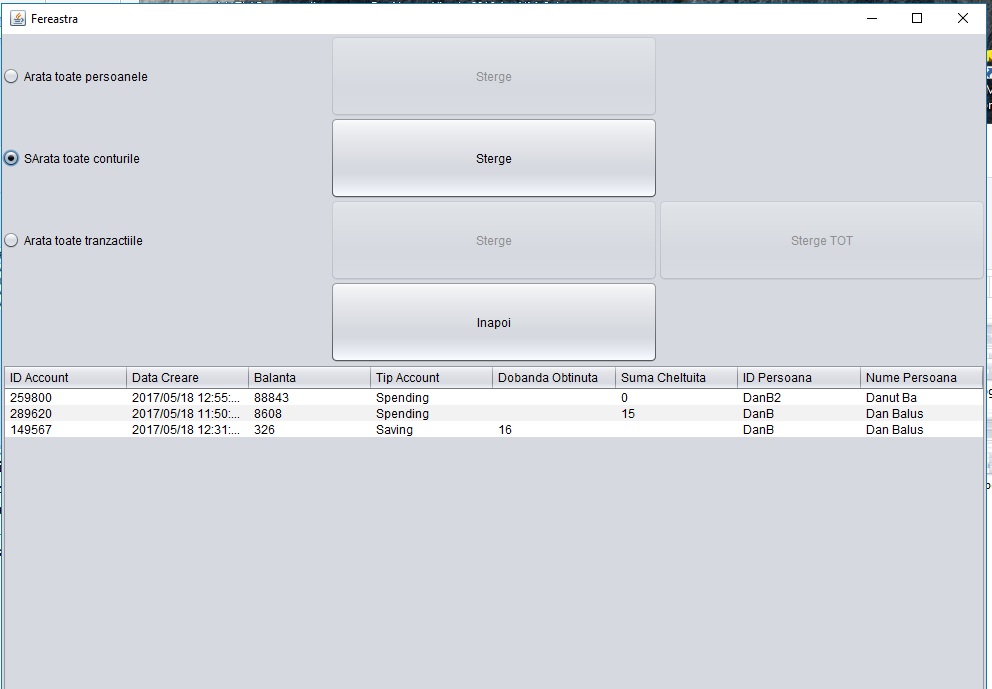


In exemplu de mai jos, dupa logarea cu succes al administratorului, a aparut fereastra cu 3 selectii: „Arata toate persoanele”, „arata toate conturi;e” si „arata toate tranzactiile”. Fiecare selectie are butoane care se activeaza doar dupa ce a fost selectata. In exemplu de mai sus, putem urmari fiecare persoana inregistrata in banca.

Putem vedea data inregistrarii fiecarei persoane, numele, prenumele lor, ID-ul personal, tara, parola, si numarul de conturi pe care le detine. In cazul in care o persoana nu detine nici un cont, ea este in mod automat eliminata din banca, si nu va mai fi afisata in acest tablou. In cazul apasarii pe butonul „sterge” se va sterge persoana din randul selectat.

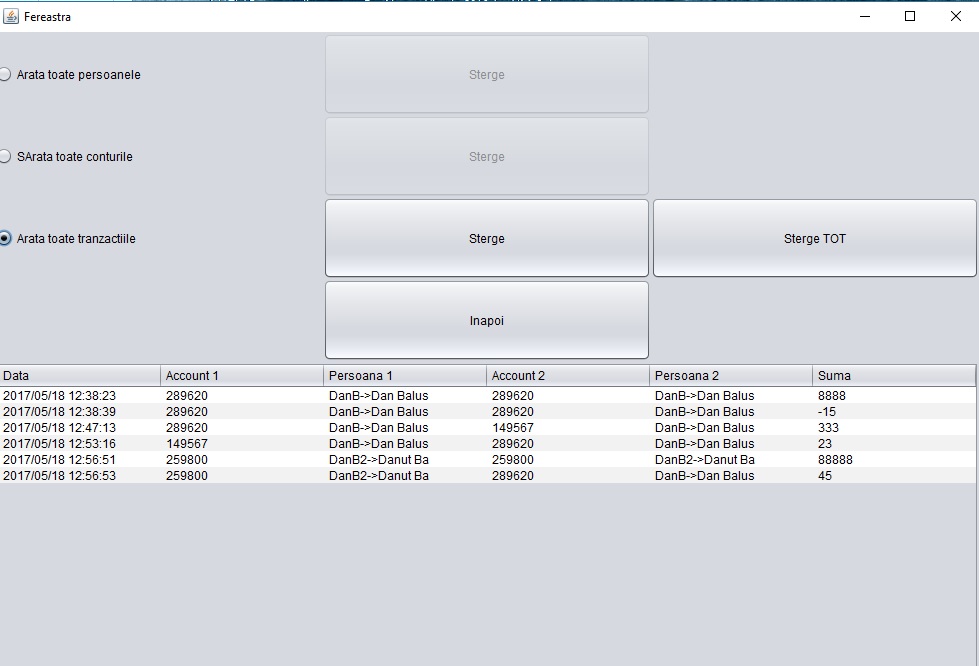
Daca nu a fost selectat niciun rand va aparea un mesaj de eroare.

19)



In acest exemplu, se vor afisa toate conturile din banca. Fiecare cont, are cate un id unic, o data de creare a contului, precum si banii depozitati in fiecare cont. Tipul contului, datele conturilor, si persoanele care le detine, impreuna cu numele, prenumele si ID-ul fiecaruia. In cazul apasarii butonului Sterge, care este activ, se va sterge persoana din randul selectat de catre utilizator. Daca niciun rand nu a fost selectat se va afisa un mesaj de eroare. In cazul in care stergem toate conturile unei persoane, in mod automat va fi eliminata si persoana respectiva din banca.

20)



In acest exemplu putem urmari toate tranzactiile care au avut loc in banca. Fiecare tranzactie are data efectuarii, ora la care a fost realizata, si cele 2 conturi intre care a avut loc tranzactia, cu id-ul si numele fiecaruia.

In cazul in care a avut loc o plata dintr-un cont de „cheltuieli” atunci contul 2 va avea acelasi ID ca primul cont, iar in rubrica Suma, se va afisa o suma negativa, ceea ce inseamna ca din acel cont au fost extrasi bani pentru cheltuiala. In cazul depunerii banilor catre aceluiasi cont, aceasta suma va fi pozitiva, iar contul 2 identic ca primul cont.

Administratorul bancii poate sa stearga cate o tranzactie apasand butonul „sterge”, sau sa le stearga pe toate deodata apasand butonul „sterge tot”. In cazul in care butonul „back” se se selecteaza, fereastra Bank Gui va fi ascunsa, adica va disparea, iar fereastra Log GUI va aparea. Pentru a intra din nou in aceasta fereastra, utilizatorul va trebui sa se logheze din nou cu contul administratorului.

1. Rezultate:

In urma lansarii aplicatiei am putut observa ca sunt multiple cai de utilizare a acesteia. Utilizatorul isi poate crea mai multe conturi si isi poate asuma doua roluri: in calitate de economist sau de cheltuitor. Fiecare dintre aceste roluri avand avantajele sale.

Pentru conturile de economii, clientul poate retrage dobanzi, iar pentru conturile de cheltuiala clientul poate face tranzactii intre conturile proprii sau a altor conturi. Clientul poate de asemenea sa-si schimbe datele personale, cum ar fi: numele, prenumele, parola, orasul. La rularea repetată a aplicației , conturile si tranzactiile nu se pierd , redeschiderea programului are loc in primul rand cu citirea conturilor si tranzactiilor vechi . Acest lucru este realizat prin intermediul interfeței „ Serializable ” .

În cadrul acestei aplicații se respectă configurația model- view – controller din cadrul arhitecturii Swing. Pentru o mai bună înțelegere a programului, s-au construit diagramele UML, de secvențe și use case, pentru a evidenția conexiunile dintre clase , obiecte , metode și utilizator .

S-a urmărit respectarea paradigmelor programării obiectuale , prin construcția programului utilizând clase, metode , relații între clase, respectând pe cât posibil principiul încapsulării .

1. Concluzii:

În cadrul acestui program , s-a urmărit implementarea codului utilizând interfete, mosteneri , respectarea paradigmelor OOP , precum și serializarea obiectelor .

Acest program poate fi dezvoltat ulterior , atât ca eficiență cât și ca vizualizare grafică. De exemplu, programul poate fi îmbunătățit astfel încât administratorul bancii sa confirme toate tranzactiile, crearea utilizatorilor sau a conturilor. Pentru conturile de economii s-ar putea de implementat o crestere a dobanzii in mod automat, in dependenta de timpul trecut. S-ar putea pune si o limita la dobanzi in dependenta de sumele alocate, etc. Pentru utilizator, s-ar putea pune si o vizualizare a tuturor tranzactiilor.