Proiect PPOO

Gestiunea tranzacțiilor unui sistem de e-banking

Coordonator științific,

Prof. Univ. Dr. Cristian Ciurea

Student,

Lazăr Ștefan

Cuprins

[Introducere 3](#_Toc119092450)

[Implementarea și prezentarea aplicației 3](#_Toc119092451)

# Introducere

Această lucrare are ca scop descrierea un sistem informatic care gestionează tranzacțiile bancare realizate de către clienți prin intermediul unei aplicații de e-banking. Acest sistem informatic ușurează modalitatea prin care utilizatorii își pot vedea tranzacțiile, aceștia nemaifiind nevoiți să se ducă la bancă să ceară extras de cont.

Aplicația este destinată tuturor utilizatorilor unei instituții bancare, atât angajați cât și clienți.

Conținutul lucrării este structurat într-un capitol care realizează descrierea tehnologiilor informatice folosite în realizarea aplicației cât și modul de implementare și proiectare a acesteia.

Deci, acest capitol sintetizează tehnologiile informatice folosite în realizarea aplicației, prezentarea aplicației și conține detaliile informatice cu privire la modul în care aceasta a fost implementată, un succint manual de utilizare cât și imagini sugestive.

# Implementarea și prezentarea aplicației

Aplicația informatica este o aplicație de tip consola, realizată in Java. Java reprezintă un limbaj de programare orientat pe obiecte, puternic tipizat, în care sunt construite cele mai multe aplicații de pe piața informatică.

Aplicația dispune de un meniu afișat în consola cu **8 opțiuni** și cu posibilitatea de ieșire din aplicație. Meniul este implementat prin intermediul unui **WHILE** care așteaptă ca utilizatorul să aleagă o opțiune, până în momentul în care acesta apasă 0, adică alege să iasă din aplicație.

Persistența datelor este realizată prin intermediul unor fișiere binare **listaConturiBancare.dat** si **listaClienti.dat** care sunt deserializate în momentul rulării aplicației, înainte de afișarea meniului. La alegerea opțiunii de ieșire din aplicație, aceste fișiere sunt restaurate cu modificările asupra datelor de lucru, modificări care se fac în memorie prin intermediul masivelor si colecțiilor.

![Text

Description automatically generated]()

Figură 1 Meniul principal

**Opțiunea numărul 1** a meniului afișează toate **conturile bancare** care se află în sistem (în fișierul **listaConturiBancare.dat**). În/Din fișier este serializat/deserializat un obiect de tipul **ListaConturiBancare** care este un **singleton** si care conține un atribut de tipul **ArrayList<ContBancar>** și, de asemenea, cărțile sunt adăugate și în colecția de tipul **LinkedHashMap<Integer, ContBancar>** a cărui scop va fi prezentat ulterior. Am ales folosirea design pattern-ului creational **Singleton**, deoarece obiectul cu lista de conturi bancare din baza de date trebuie sa fie unic la nivelul aplicației. De asemenea, clasa implementează interfața **OperatiiFisier** care conține cele 2 metode de salvare și populare a listei. Astfel, lista de conturi bancare este parcursă cu un **for-each** si este afișată fiecare cont bancar în parte.

![Graphical user interface, text

Description automatically generated]()

Figură 2 Conturile și tranzacțiile aferente din sistem

**Opțiunea numărul 2** a meniului permite adăugarea unui nou cont bancar în sistem. Adăugarea se face în cadrul listei care ulterior va fi serializată în fișier la închiderea aplicației. Datele contului bancar se introduc pe rând cu posibilitatea de anulare a adăugării contului. Fiecare input trebuie să îndeplinească diferite criterii, de exemplu **IBAN-ul trebuie să aibă fix 24 de caractere**, **BIC-ul** **trebuie să aibă fix 24 de caractere etc.** De asemenea, ele sunt preluate prin intermediul mecanismului de gestiune al excepțiilor, cât și prin excepția custom **InvalidValueException**.

![Text

Description automatically generated]()

Figură 3 Adaugă cont bancar

**Opțiunea numărul 3** a meniului permite crearea unui raport pe baza numelui unui client. Utilizatorul comunica clientul dorit, iar dacă acesta este găsit, se afișează datele contului bancar al respectivului și tranzacțiile făcute, dacă acestea există. Daca nu este găsit, utilizatorul va fi întors la meniul principal. Acest raport poate fi salvat la dorința utilizatorului într-un fișier text.

![Text

Description automatically generated]()

Figură 4 Creare raport cont bancar

**Opțiunea numărul 4** a meniului permite actualizarea unui cont bancar. Așa cum a fost precizat mai sus, la popularea listei, conturile mai sunt adăugate și într-o colecție **LinkedHashMap<Integer, ContBancar>** a cărei cheie este un index folosit pentru gestionarea contului care se dorește a fi actualizat. Mai întâi este afișată la consolă lista cu toate conturile bancare aflate în sistem. Utilizatorul alege numărul contului pe care dorește să îl actualizeze, iar apoi alege atributul pe care îl va actualiza. Și aici acesta trebuie sa fie atent, deoarece dacă nu respectă validatorii care se află și în setteri, aplicația se va opri, deoarece a fost aruncată o excepție.

![Text

Description automatically generated]()

Figură 5 Actualizare cont bancar

**Opțiunea numărul 5** a meniului permite ștergerea unei cont bancar. Mai întâi este afișată la consolă lista cu toate conturile bancare aflate în sistem. Utilizatorul alege numărul contului pe care dorește să îl șteargă. Dacă acesta există, contul bancar cu numărul respectiv va fi șters cu succes. Dacă nu există, va fi afișat un mesaj care va menționa introducerea unui input valid.

![Text

Description automatically generated]()

Figură 6 Ștergerea unui cont bancar

**Opțiunea numărul 6** a meniului permite filtrarea conturilor bancare după o anumită categorie. Validarea tipului de cont bancar, tip care este realizat printr-un **enum,** se face prin intermediul unei colecții de tipul **HashMap<String, TipContBancar>** care este inițializat cu fiecare valoare din enum și cu elementul respectiv pentru căutare printr-un bloc static din cadrul enum-ului. Mai întâi vor fi afișate toate tipuri de conturi bancare. Dacă valoarea inputului nu este egală cu una dintre tipurile prestabilite, se va afișa un mesaj corespunzător. După ce s-a trecut un input valid, se vor afișa toate conturile și tranzacțiile, dacă acestea există, de tipul respectiv. De asemenea, putem să și salvăm acest raport într-un fișier text.

![Text

Description automatically generated]()

Figură 7 Filtrare cont bancar după tip

**Opțiunea numărul 7** a meniului permite căutarea unui client după id și afișarea tuturor tranzacțiilor sale. Dacă nu este găsit un client cu id-ul din input, utilizatorul va fi întors la meniul principal. Dacă este găsit, se va afișa un mesaj personalizat cu numele clientului și apoi se va putea genera un raport cu tranzacțiile sale. Dacă acesta nu are tranzacții, va fi afișat un mesaj corespunzător. Dacă acesta are tranzacții, se va afișa numele și id-ul clientului, respectiv lista tranzacțiilor și soldul curent. Apoi există și opțiunea de a salva acest raport într-un fișier text.

![Text

Description automatically generated]()

Figură 8 Căutare client și afișare tranzacții existente

**Opțiunea numărul 8** a meniului permite afișarea tuturor tranzacțiilor pentru fiecare utilizator. După cum se poate vedea mai jos, se afișează numele fiecărui client sub care se află toate tranzacțiile făcute de acesta. De asemenea, acest raport se poate salva într-un fișier text.

![Text

Description automatically generated]()

Figură 9 Vizualizare listă tranzacții