UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

GESTIONAREA COMENZILOR INTR-UN DEPOZIT

-TEMA 3-

Nume: Bolduț Denisa-Teodora

Grupa: 30225

Cuprins:

Tema 3:

Consider an application OrderManagement for processing customer orders for a warehouse. Relational databases are used to store the products, the clients and the orders. Furthermore, the application uses (minimally) the following classes: • Model classes - represent the data models of the application • Business Logic classes - contain the application logic • Presentation classes – classes that contain the graphical user interface • Data access classes - classes that contain the access to the database

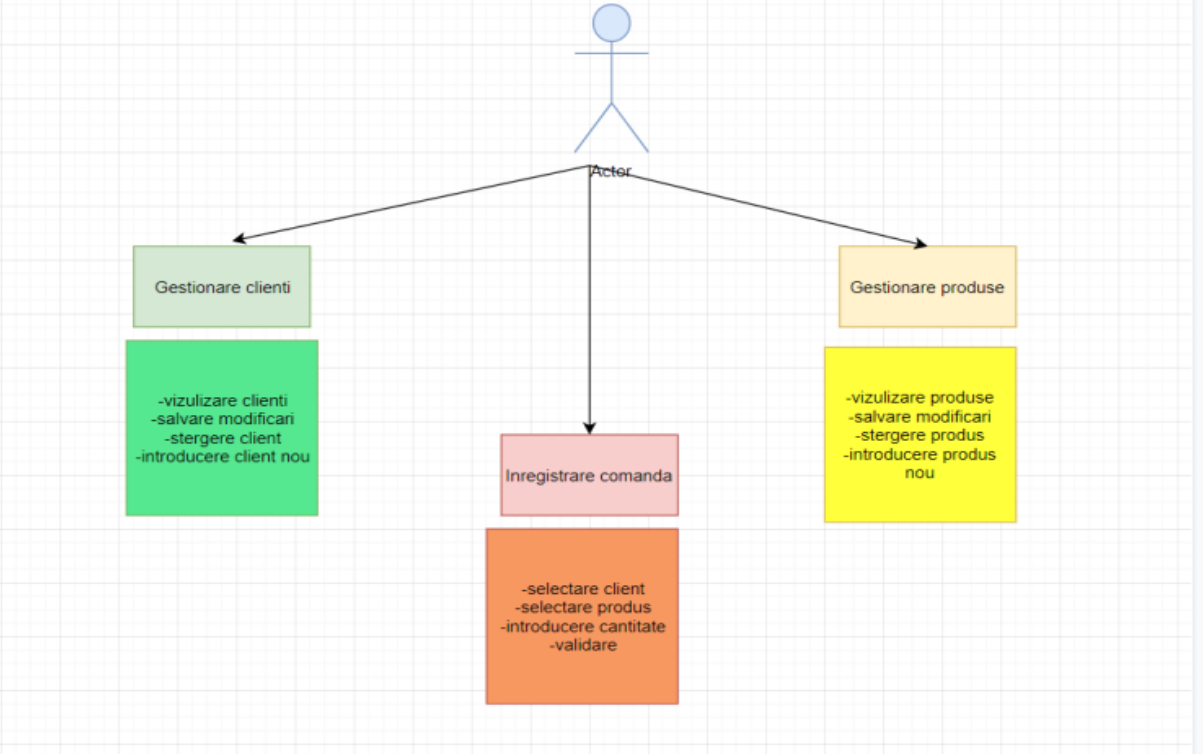
1. Obiectivul temei

Aceasta tema are scopul de a gestiona functionalitatile necesare intr-un depozit. Pentru a indeplini aceste functionalitati avem nevoie de optiunea de a insera noi clienti, noi produse, de a le edita sau sterge si bineinteles de a face o comanda. Proiectul astfel scris va avea la baza deci niste tabele realizate in baza de date prin MySQL pentru a pastra informatiile existente si pentru a ajuta prelucrearea lor.

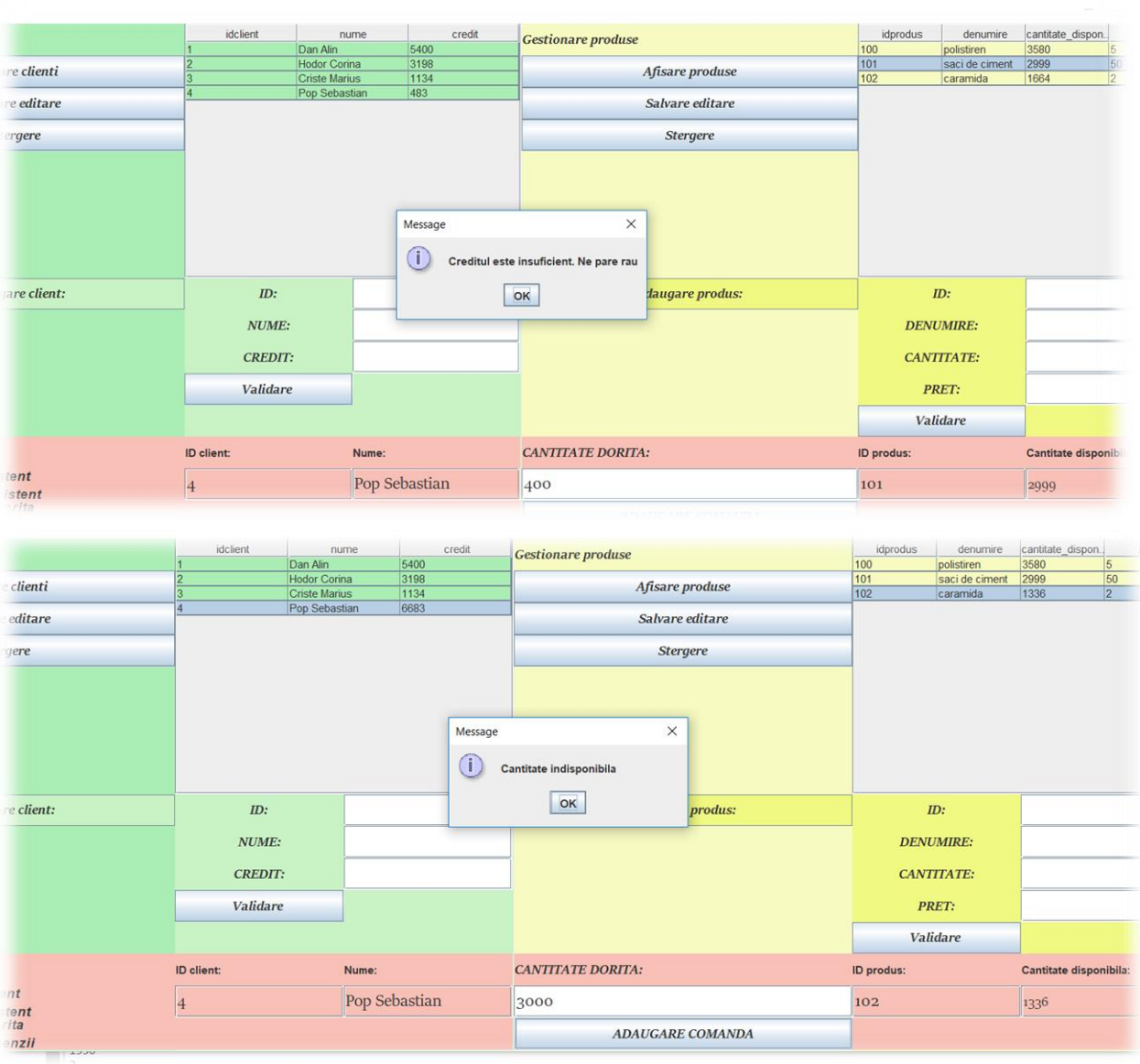
Astfel avem reprezentati sub forma de tabel clientii si produsele din baza de date care vor putea fi vizualizate si selectate pentru diversele operatii. Ne vom folosi de Jtable pentru a interactiona cu datele existente, fiind un tool care ne prermite sa selectam mai usor randurile tabelelor. Tabelele pot fi gandite ca un mod de reprezentare a datelor bidimensionale. De fapt, clasa JTable are chiar un constructor care primeste ca argument un masiv bidimensional de obiecte din clasa Object si afiseaza continutul acestora in liniile si coloanele unui tabel.

2. Analiza problemei, modelare, scenarii, cazuri de utilizare

Analizand problema acestui proiect observam faptul ca utilizatorul trebuie sa interactioneze si sa prelucreze informatii stocate in baza de date. Prin interactionarea utlizatorului intelegem alegerea unei operatii pe care o doreste sa o implementeze, si anume vizualizarea cleintilor, salvarea modificarii aduse unui anumit client , stergerea unui client , introducerea unui nou client, vizulaizarea produselor, salvarea modificarii asupra unui produs,stergerea unui produs, introducerea unui nou produs si nu in cele din urma posibilitatea de a inregistra o comanda . Pentru validarea unei comenzi , utilizatorul este nevoit sa selecteze un client existent in tabela, un produs existent, sa introduca cantitatea dorita si sa apese pe butonul de validare . Acest lucru este valid si se inregistreaza in baza de date daca si numai daca cantitatea dorita de comandare a produsului este disponibila in depozit si daca si numai daca acea valoare a cantitatii in functie de pretul produsul este posibila sa fie achitata de catre client, clientul avand un credit inregistrat in tabela corespunzatoare lui .

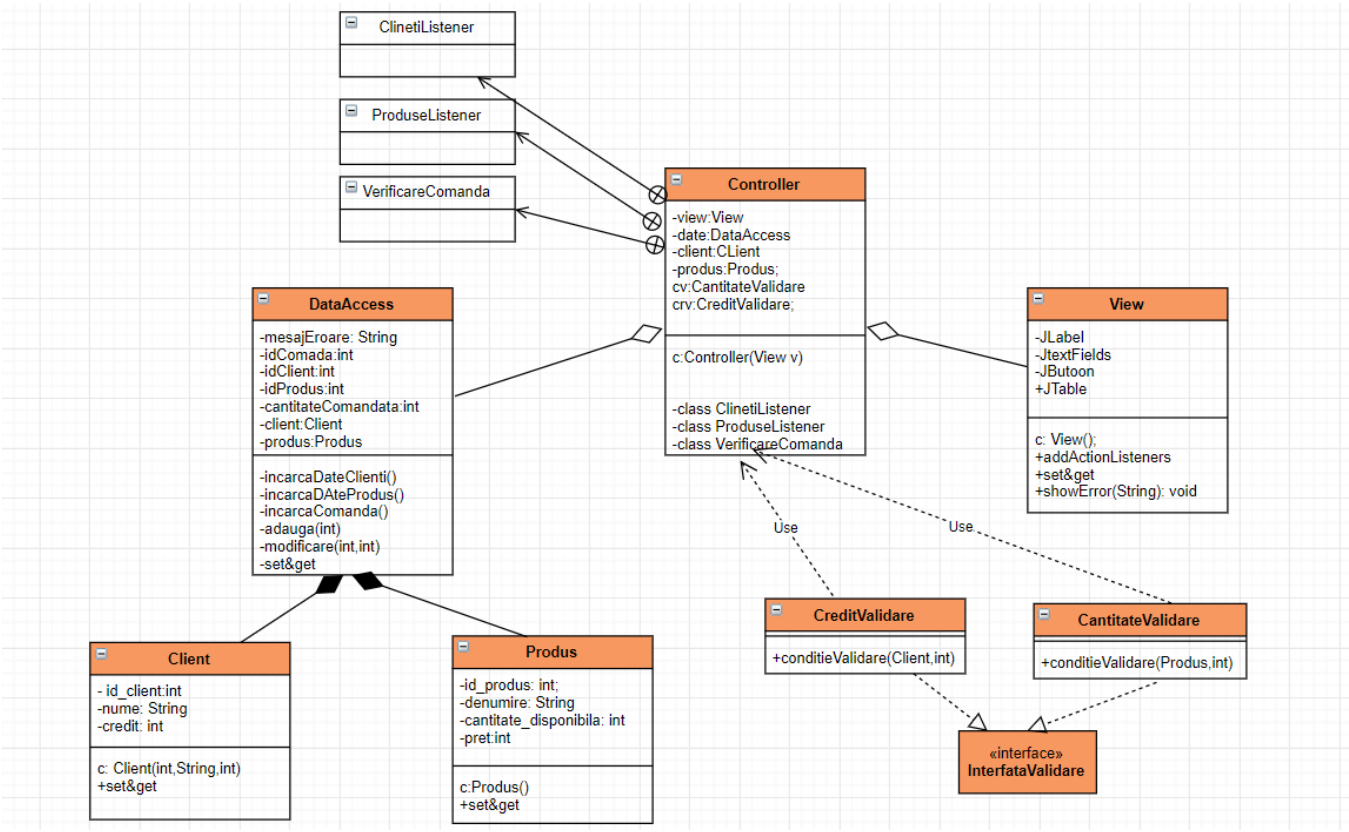


In cazul in care validarea comenzii nu a reusit, se arunca o exceptie , acest lucru atentionand utilizatorul cu privire la exceptia aruncata, putand vizuliza cauza erorii. Se poate intampla acest lucru in cazul in care cantitatea produsului nu este disponibila, fiind mai putine produs in depozit decat cele care se cer ca cantitate dorita. In acelasi timp, clientul trebuie sa furnizeze un credit indeajuns pentru a achizitona produsul in cantitatea dorita. Aceste mesaje il intampin ape utilizator intr-o fereastra cum este in exemplul urmator, semnaland ambele excpetii ce pot fi intalnite:



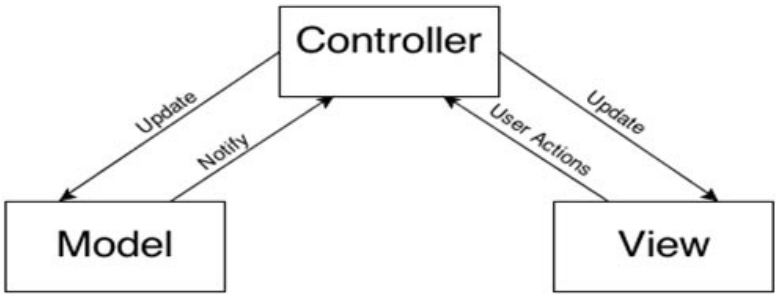
3. Proiectare

DIAGRAMA UML



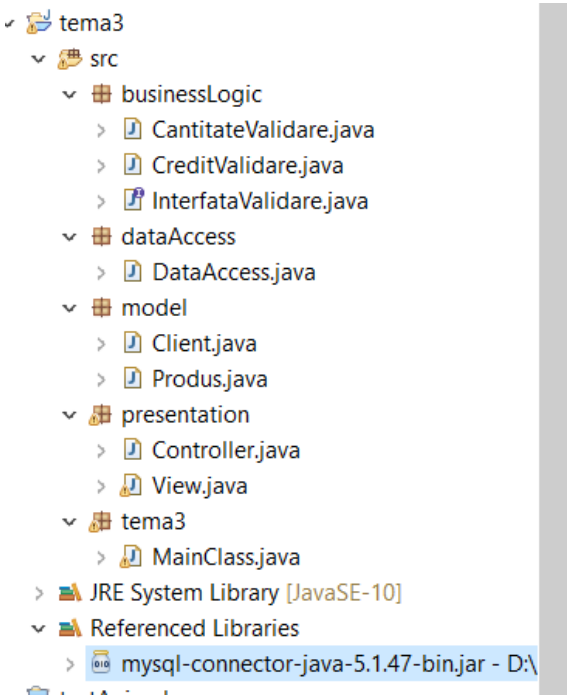
Astfel reprezentant diagrama UML se pot observa legaturile dintre clase si interfate. UML este un limbaj standard pentru descrierea
de modele s specificatii pentru software. Limbajul a fost creat de către consortiul Object Management Group (OMG) care a mai produs printre altele si standardul de schimb de mesaje intre sisteme CORBA. UML a fost la bază dezvoltat pentru reprezentarea complexităţii programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase, si instantele acestora (numite si obiecte). Cu toate acestea, datorită eficientei si claritătii în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT. Asa se face că există aplicatii ale UML-ului pentru management de proiecte, pentru business Process Design etc.

Structura proiectului este gandita sub modelul arhitectural ***Model-view-controller (MVC)***:



Structuri de date & PROIECTARE CLASE

Pentru realizarea proiectului principala intrebuintare este conexiunea cu baza de date. Pentru a indeplini acest lucru este nevoie sa includem connectionul.



JDBC (Java Database Connectivity) este o interfaţă standard (API) de programare a accesului la baze de date fiind constituită dintr-un set de clase si interfeţe scrise în Java, furnizând mecanisme standard pentru

dezvoltarea aplicaţiilor Java cu baze de date . Folosind facilitătile oferite de JDBC este usor să transmitem secvente SQL către baze de date relationale (MYSQL ,MS-SQL,Oracle) . Pachetele Java care ofera suport pentru lucrul cu baze de date sunt incluse in java.sql , ce reprezinta nucleul tehnologiei JDBC oferind posibilitatea lucrului cu instructiuni SQL si procesarea rezultatelor obtinute în urma interogărilor , respectiv javax.sql ,ce oferă capabilităti pe parte de server cum ar fi tranzactii distribuite si conexiuni multiple, în conditiile în care într-o arhitectură client-server baza de date se poate afla pe aceeasi masină sau pe o altă masină cu care clientul este conectat dintr-un intranet sau Internet. Utilizând API-ul JDBC aplicatiile pot fi programate în mod independent de SGBD, iar apelurile SQL către baza de date vor fi trimise ,preluate si transformate de către diverse drivere scrise specific pentru fiecare SGBD. Specificatia JDBC pune la dispozitia producătorilor de drivere reguli de construire a driverelor, de-alungul timpului atât specificatia cât si driverele au cunoscut o evolutie continuă, sub aspectul functionalităților oferite.Se vor verifica caracteristicile/functionalitatile oferite de fiecare driver pentru a putea dezvolta in mod corect aplicatia client Java.

Principii OOP

In programarea procedurala, datele sunt descrise separat de prelucrari, astfel ca legatura dintre ele se realizeaza numai la nivel conceptual. Pentru tipurile de date primitive s-a adoptat conceptul, conform caruia prin tip de date se intelege o multime de valori, careia i se asociaza o multime de operatii care se pot efectua asupra valorilor respective. Constatam deci ca, inca de la inceputurile programarii in limbaje de nivel superior, s-a facut

o legatura stransa intre date si operatii, legatura incorporata in fiecare limbaj de programare.

Un obiect care apartine unei clase se numeste si instanta a clasei (este o instantiere, o realizare particulara a clasei respective). De exemplu, clasa Barbat si clasa Femeie sunt subclase ale clasei Om. In schimb, Ion_Popescu este o instanta a clasei Barbat (un obiect care apartine acestei clase), iar Maria_Preda este o instanta a clasei Femeie.

Polimorfismul (engleza: polymorphism) permite ca aceeasi operatie sa se realizeze in mod diferit in clase diferite. Sa consideram, de exemplu, ca in clasa Figura_geometrica exista metoda arie(), care calculeaza aria figurii respective. Clasele Cerc, Triunghi, Patrat sunt subclase ale clasei Figuri_geometrice si vor mosteni, deci, de la aceasta metoda arie(). Este insa evident ca aria cercului se calculeaza in alt mod decat aria patratului sau cea a triunghiului. Pentru fiecare din instantele acestor clase, la calcularea ariei se va aplica metoda specifica clasei sale.

Incapsularea (engleza: encapsulation) este proprietatea obiectelor de a-si ascunde o parte din date si metode. Din exteriorul obiectului sunt accesibile ("vizibile") numai datele si metodele publice. Putem deci sa ne imaginam obiectul ca fiind format din doua straturi.

Agregarea (engleza: aggregation) este proprietatea obiectelor de a putea incorpora alte obiecte. Asa dar, "datele" continute intr-un obiect pot fi nu

numai date primitive, ci si obiecte. Se pot astfel crea obiecte cu structuri din ce in ce mai complexe.

Mostenirea (engleza: inheritance) este proprietatea unei clase de a contine toate atributele (variabilele) si metodele superclasei sale. In consecinta, trecerea de la clasa la subclasa se face prin adaugarea de atribute si/sau de metode. De exemplu, clasa Barbat si clasa Femeie au ambele toate atributele clasei Om, dar fiecare din acestea are si atribute specifice.

Accesarea unei baze de date folosind JDBC este simplă şi implică următorii paşi:

1. Obţinerea unui obiect de tip Connection ce încapsulează conexiunea la baza de date (în acest pas se realizează deci conexiunea la baza de date).

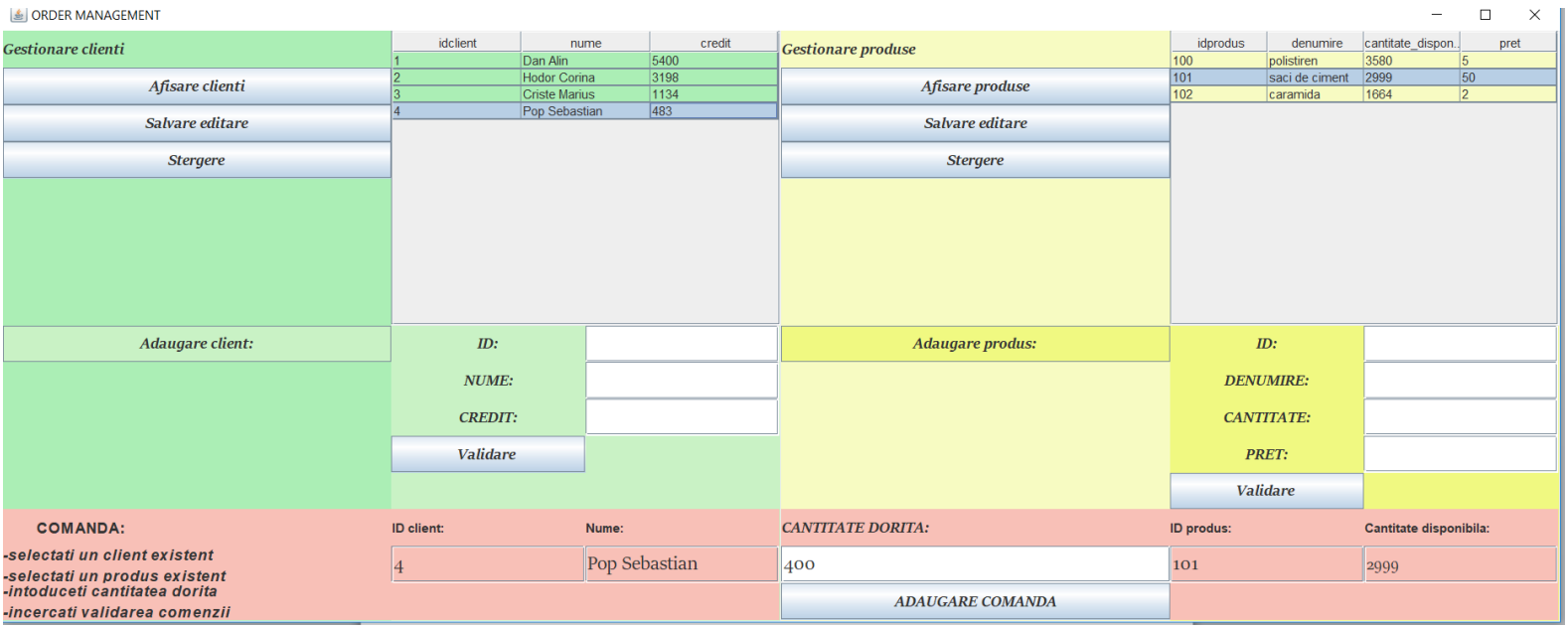
2. Obţinerea unui obiect Statement dintr-un obiect de tip Connection. Acest obiect este folosit pentru a transmite spre execuţie comenzi SQL către baza de date. Prin intermediul acestui obiect sunt efectuate operaţii de interogare şi modificare a bazei de date.

3. Obţinerea unui obiect ResultSet dintr-un obiect Statement. Obiectul ResultSet încapsuleaz rezultatele operaţiilor de interogare.

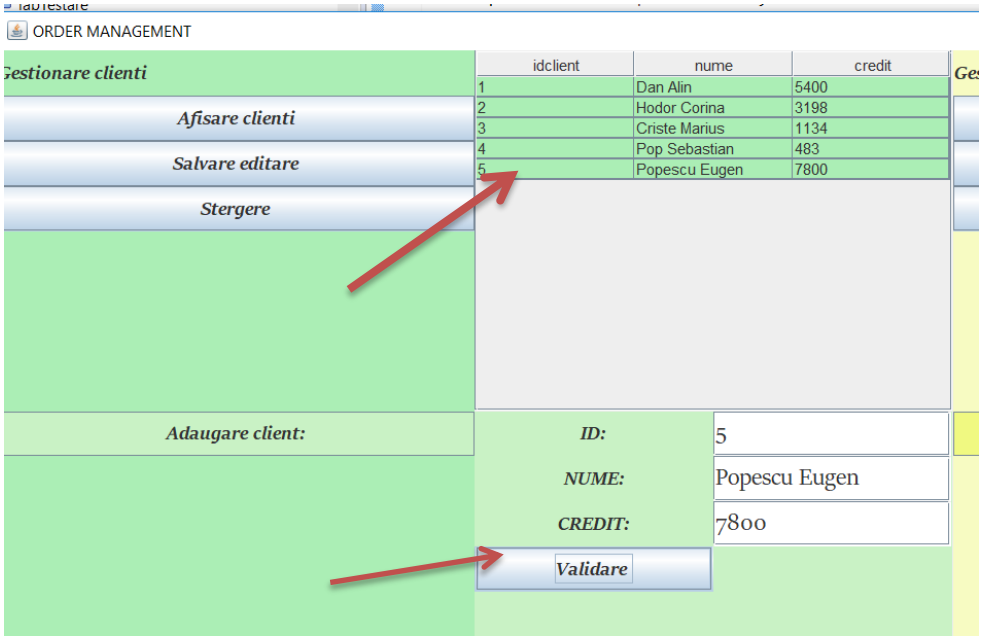
4. Procesarea rezultatelor încapsulate în obiectul ResultSet. Procesul de conectare la o bază de date implică înregistrarea unui driver corespunzăto si realizarea unei conexiuni cu baza de date. O conexiune (sesiune) la o baza de date reprezintă un context prin care sunt trimise secvente SQL catre baza de date si primite rezultate la nivel de aplicatie.

4. Implementare

Interfata cu utilizatorul

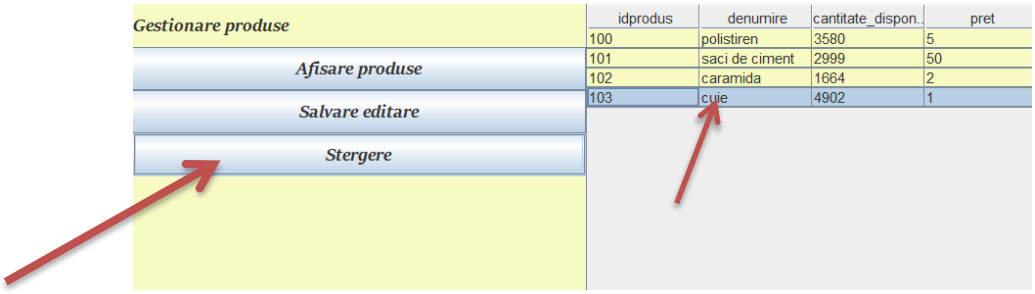


Prin intermediul culorilor vedem distribuirea gestiunilor. Cu verde este reprezentata gestionarea glientilor, acestia fiind vizulizati in tabelul alaturat reprezentat tot pe fond verde . Cu galben sunt ilustrate operatiile privind produsele. Afisarea produselor fiind vizulizata in tabelul corespunzator, salvarea modificarii facandu-se direct in tabel, iar stergerea selectand un produs si apasand butonul corespunzator stergerii. Avem si o zona dedicata introducerii unui nou produs/ client fiind necesare introducerea proprietatiilor fiecaruia iar apoi se apasa butonul de validare, rezultatul putand fi vizulizat instat in tabela. Vom lua un exemplu pentru a clarifica acest aspect. Presupunem ca avem tabelul de clienti prezentat anterior. Acum vrem sa introducem un client cu id-ul urmator (acesta este unic), de exemplu 5, vrem sa il numim Popescu Eugen si sa ii acordam un credit in valoare de 7800.



In momentul in care am apasat pe butonul de validare, clientul poate fi vizulizat deja in tabel.

Acelasi lucru se intampla si in cazul produselor. La fel daca vrem sa stergem de exemplu un produs, acesta trebuie selectat din tabel iar apoi apasat pe butonul de stergere, rezultat observat imediat.



lar cu portocaliu avem „ fereastra ” destinata comenzilor, cu indicatiile necesare in partea stanga.

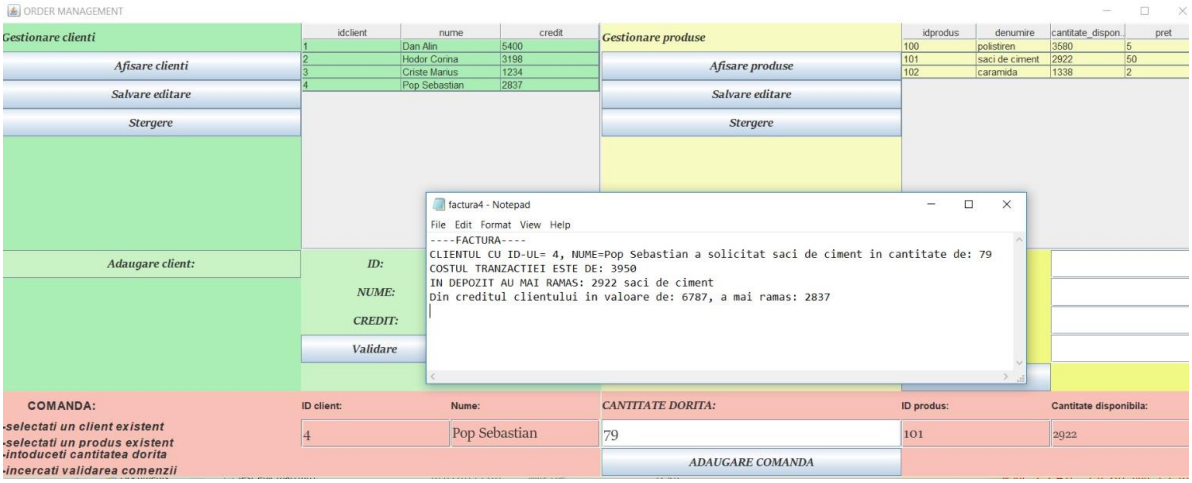
Acest mecanism ne este oferit de catre pachetul java.lang.reflect. Reflectia inseamna autoexaminare, adica permite determinarea structurii clasei. In limitele managerului de securitate putem afla metodele clasei, constructorii clasei si restul membrilor clasei. Cateodata putem sa modificam starea obiectlui prin apelul metodelor specifice sau putem construi obiecte noi . Mecanismul de reflectie este utilizat de componentele Java (Java beans) pentru determinarea capabilitatilor obiectelor pe timpul executiei. Membrii clasei sunt atributele si metodele. Metodele se impart la randul lor in doua categorii, constructori si metode obisnuite. Principalele clase care definite in acest pachet sunt:

- java.lang.reflect.Field
- java.lang.reflect.Method
- java.lang.reflect.Constructor

Clasa Class ne ofere cate o pereche de metode pentru obtinerea atributelor, metodelor respectiv ai contructorilor. Una dintre aceste metode permit accesul la metodele publice, incluzand si cele mostenite, iar cealalalta metode permite accesul doar la metodele publice si nepublice declarate in cadrul clasei. De exemplu getFields() va fi metoda care ne da lista tuturor campurilor de date publice, iar getDeclaredFields() returneaza doar cele declarate in cadrul clasei.

5. Rezultate

In urma validarii undei comenzi se genereaza o un fisier text care reprezinta factura clientului. In aceasta se poate vizuliza cine si ce si cat cumpara, la fel si pretul pe care este necesat sa il achid ein urma achizitiei.



Un alt exemplu de factura care se poate genera in urma operatiilor este acasta:

----FACTURA----

CLIENTUL CU ID-UL= 3, NUME=Criste Marius a solicitat saci de ciment in cantitate de: 2

COSTUL TRANZACTIEI ESTE DE: 100

IN DEPOZIT AU MAI RAMAS: 2999 saci de ciment

Din creditul clientului in valoare de: 1234, a mai ramas: 1134

Sau acesta :

----FACTURA----

CLIENTUL CU ID-UL= 5, NUME=Popescu Eugen a solicitat cuie in cantitate de: 23

COSTUL TRANZACTIEI ESTE DE: 23

IN DEPOZIT AU MAI RAMAS: 4879 cuie

Din creditul clientului in valoare de: 7800, a mai ramas: 7777

6. Concluzii, ce s-a invatat din tema, dezvoltari ulterioare

In urma implementarii acestui proiect am invatat sa lucrez mai bine cu bazele date, sa gestionez si sa prelucrez informatiile din tabele pentru a le putea afisa si pentru a putea interactiona utilizatorul cu ele.

Am mai invatat sa folosesc JTable, operatiile de selectare a clientului sau a produselor putand fi astfel usurate. Jtable iti permite sa selectezi un rand si poate stoca informatiile din acesta, usurand astfel lucru cu datele existente.

Dezvoltari ulterioare a acestei pot fi mai multe tabele in baza de date pentru a gestiona mai multe aspecte a unui depozit, apropiindu-ne astfel cat mai mult de viata reala.

7 . Bibliografie

http://www.lec-academy.ro/utilizarea-jtable-in-swing-partea-i/

https://ms.sapientia.ro/~manyi/teaching/oop/oop_romanian/curs16/curs16.html

https://howtodoinjava.com/oops/object-oriented-principles/

https://ibytecode.com/blog/jdbc-mysql-create-database-example/