**Orders Management**

**Tema 3**

Logo, company name

Description automatically generated

Student: Alexa Andrei Stefan

Grupa: 30225

1. **Obiective**

Proiectati si implementati o aplicatie pentru administrarea unor comenzi plasate de clienti intr-un depozit.

**Sub-obiective:**

* Analizati problema si identificati cerintele
* Proiectati aplicatia de administrare a comenzilor
* Implementati aplicatia de administrare a comenzilor
* Testati aplicatia de administrare a comenzilor

**De ce avem nevoie de o astfel de aplicatie de administarre a comenzilor?**

Acest proiect simuleaza lucrul cu comenzile, clientii si produsele unui deposit. Un astfel de tip de aplicatie ar putea fi folosita de un depozit real pentru a lucre cu toate datele intr-o maniera automata si mult mai organizata.

1. **Analiza problemei, modelare, scenarii si cazuri de utilizare**

**2.1 Analiza Problemei**

Baza de date pentru depozit contine 3 tabele: Client, Produse si Comenzi.

Noi vrem sa cream clase care reflecta aceste tabele (Client, Produse si Comenzi) si sa cream metode prin care putem sa lucram cu datele acestora.

Pentru a plasa/crea o comanda trebuie sa putem selecta un client existent, un produs existent si sa specificam cantitatea pe care dorim sa o comandam. Aceasta cantitate va fi extrasa din stocul disponibil al produsului. Daca nu sunt destule produse in stock, un mesaj va aparea pe ecran.

**2.2 Modelare**

Am creat 8 interfete diferite pentru o mai buna prezentare a aplicatiei. Fiecare interfata are cateva butoane pentru diferite functii(adaugare, stergere, selectare, update).

Cateva din cele 8 interfete contin View-uri asupra tabelelor pe care se vor executa operatiile alese.

Pentru interfata de plasare a unei comenzi am folosit 2 campuri in care se afiseaza date despre produsele pe care vrem sa le comandam si pe care le vom putea viziona in cosul de cumparaturi.

**2.3 Scenarii si cazuri de utilizare**

Folosirea acestei aplicatii este putin mai complexa datorita faptului ca este utilizat un numar mai mare de campuri text si butoane, afisand si ceva mai multe informatii pe ecran, dar in acelasi timp ea ramane foarte usor de folosit pentru orice utilizator.

Cand aplicatia se deschide, o fereastra care ne permite sa selectam ce vrem sa facem in continuare, apare pe ecran. Din aceasta fereastra noi putem sa ne alegem pe ce vrem sa facem urmatoarele operatii(Client, Produse, Comenzi). Fiecare dintre cele 3 interfete contine butoane si campuri text.

Cateva exmeple sunt:

* Add – in functie de interfata pe care o alegi(Client sau Produs) te va trimite la fereastra in care treuie sa introduci date pentru a adauga un client sau un produs.
* Delete – pentru a sterge un produs sau client, se va selecta din campul text alaturat produsul sau clientul pe care dorim sa il stergem.
* Update – te va trimite la fel ca si functia Add la una dintre interfetele alese.
* Adauga/Sterge - adauga sau sterge un produs din lista de cumparaturi.
* Plaseaza comanda – Confirmi comanda selectata.
* Alege clientul – Alegi clientul cu care se doreste a se plasa comanda.

Butonul de Add ia informatia de la toate campurile exceptand id-ul (care este generat automat in baza de date) si incearca sa creeze un nou obiect in tabelul respective, cu informatiile date.

Butonul de update are nevoie de un id pentru a gasi obiectul pe care vrem sa il editam (fie el client sau produs) si va lua informatia din restul campurilor , inlocuindu-le cu informatiile curente ale obiectului cu cele noi.

1. **Proiectare**

**3.1 Decizii de proiectare**

Structura pe nivele, data in prezentare a fost o mare inspiratie pentru proiectul meu, avand urmatoarele pachete: Connection, Dao, Model, Presentation si Start.

Interfata trebuie sa cat mai usor posibil de folosit si sa aiba toate cerintele functionale, insa ultima parte cu crearea bonului nu a mai putut fi realizata.

**3.2 Diagrama UML si Clase**

**Graphical user interface

Description automatically generated**

**3.3 Structuri si tipuri de date**

Listele observate sunt folosite obiectele pe care vrem sa le afisam.

Start-ul a fost folosit pentru testarea initiala a functionalitatii codului, insa run-ul principal pentru rularea aplicatiei se face din clasa Select.

Butoanele sunt utilizate pentru a selecta diferite operatii.

Clasele Client, Produs si Comanda sunt create imitand campurile din baza de date.

StringBuild-urile le-am folosit pentru a crea interogari si alte mesaje.

ResultSet-ul este folosit pentru a stoca datele pe care le obtinem prin interogarile facute asupra bazei de date.

buildTbable o folosim pentru a contrui un table in interiorul unui JPanel, ne folosim de ResultSet pentru a stoca datele, iar mai apoi de un metaData pentru a le putea procesa individual si pentru a adauga coloanele si randurile.

**3.4 Proiectarea Claselor, Interfetelor si Pachetelor**

Ideea cu care am inceput aici a fost sa am clase compacte, bine organizate, care sa aiba metode specific utilizarii operatiilor si sa fie numite corespunzator.

O sa incep sa explic cateva din clase:

AbstractDao:

* Este cea mai importanta clasa din acest proiect. Este folosita pentru metodele reflexive din acest proiect.
* Aceasta clasa lucreaza cu cele 3 clase din baza noastra de date si anume: Client, Produs si Comanda.
* In aceasta metoda am implementat metodele de interogare speficifice (adaugare, stergere, find, afisare) si de asemenea metodele care folosesc aceste metode de creare a interogarilor (insert foloseste metoda adaugare, stergere foloseste metoda stergere, update foloseste metoda find).
* Tot in aceasta clasa a implementat si o metoda numita buildTable pe care o folosim pentru a contrui un table in interiorul unui JPanel.

ConnectionFcatory:

* Aceasta clasa este folosita pentru a stabili conectarea sau deconectare la sau de la baza de date aleasa.

Client, Produs si Comanda:

* Aceste clase au aceleasi campuri ca si tabelele din baza de date. Acest tip de obiecte vor reprezenta o intrare in baza de date. While-ul din clasa Start creaza un reprezentare a unui String cu obiectele de acel tip, acesta a fost insa folosit doar pentru teste .

Select:

* Aceasta clasa este cea unde este main-ul principal cu ajutorul caruia este rulata aplicatia.

Pachetul Presentation contine “creierul ” acestei aplicatii, aici se intampla toate eventimentele care se intampla la apasarea unui buton (interactiunea dintre utilizator si aplicatie ). Fiecare buton apeleaza o metoda diferita care are un effect in cascada asupra tuturor celorlalte metode din celelalte clase, putandu-l asemana cu efectul domino.

Pachetele reprezinta structura pe nivele.

Baza de date este in cel mai de jos nivel al arhitecturii.

**3.5 Algoritmi folositi:**

Cel mai important algoritm pentru a lucre cu o baza de date este crearea unei interogari bazata pe cee ace ai nevoie (select \* from table etc ) si folosirea acelei interogari pentru a obtine un set de rezultate care corespunde cu datele obtinute prin folosirea acelei interogari. Cu resultSet-ul poti merge mai departe in a crea obiecte in programul tau care pot fi folosite mai departe pentru alte operatii.

Ideea reflexivitatii vine de la dorinta de utilizare a aceluiasi cod care sa lucreze cu mai multe tipuri de date sau obiecte. Daca ai 3 tipuri diferite de tabele, este mult mai eficient sa scrii o singura metoda care poate sa extraga datele din toate 3 tabelele decat sa scrii metode similiare pentru toate cele 3 tabele. Codul reflexiv poate fi deasemenea utilizat in mai multe proiecte care folosesc idei similare cu acestea.

**3.6 Interfata**

Pentru creare interfetei am folosit Eclipse deoarece mi se pare foarte usor de folosit si de realizat un schelet a cee ace vrei sa faci in proiectul tau. Spre exemplu, eu am facut codul “mama” care a continut doar butoanele si campurile specifice fara a le oferi si functiile caracteristice, iar mai apoi in cod-ul meu am inserat fix cee ace aveam nevoie.

Interfata este implementata folosind javaSwing, este foarte usor de folosit de catre toti utilizatorii.

Interfata apeleaza operatiile care trebuie sa fie executate cand un utilizator apasa pe unul dintre butoane (dinamic).

Interfata este usor de citit si de inteles de catre fiecare utilizator. Butoanele si campurile sunt denumite specific si intuitive fiecarei misiuni pe care o indeplinesc.

La rularea programului, prima fereastra care apare este urmatoarea:

Graphical user interface

Description automatically generated

Aceasta fereastra ne ofera optiunea de a accesa Tabela Client, la apasarea butonului denumit in acelasi fel, tabela Produs la apasarea butonului denumit “Produs”, iar daca se doreste plasarea unei comenzi, se va apasa pe butonul “Comanda”.

Daca spre exemplu am ales tabela Client, utilizatorul va fi redirectionat catre fereastra urmatoare care ii permite sa Adauge un nou client in table, sa stearga sau sa faca update unui dintre clientii deja existenti

Graphical user interface

Description automatically generated Pentru stergerea si pentru a face un update unui client din tabel, acesta trebuie selectat cu mouse-ul.

Daca se dreste adaugarea unui client nou in tabela, se va apasa pe butonul Add, iar utilizatorul va fi redirectionat catre urmatoarea fereastra:

Graphical user interface, application

Description automatically generated Se vor completa campurile denumite specific, iar apoi se va apasa butonul “Add Client” care il va adauga direct in tabelul nostru.

Tabelul se va reactualiza instant, iar valoarea va putea fi observata pe ultima pozitie libera a tabelului.

Daca acelasi buton de Add ar fi apasat dup ace a fost selectata categoria Produs la aparitia primei ferestre, urmatoarea fereastra va fi afisata, iar felul de functionare al butoanelor este similara cu cea de la tabela Clienti:

Table

Description automatically generated with low confidence Butoanele de Delete si Update functioneaza la fel ca si cele de la Client, insa la apasarea butonului de Add va aparea o fereastra specifica pentru un produs:

Graphical user interface, application

Description automatically generated

Daca se doreste plasarea unei comenzi, butonul “Comanda” va fi apasat, iar utilizatorul va fi redirectionat catre urmatoarea fereastra:

Graphical user interface

Description automatically generated

Pentru a plasa o comanda trebuie selectat un produs din tabelul din partea dreapta, iar de jos trebuie ales id-ul clientului cu care se doreste a se face achizitionarea. Tot pentru procesarea unei comenzi trebuie introdusa si o cantitate care in cazul depasirii stocului acelui produs, un mesaj de eroare va fi afisat. Aceasta cantitate se scade din stoc la fiecare adaugare a unui produs in cos. Daca se doreste revocarea comandarii unui produs, acesta trebuie selectat din cos si apasat pe butonul sterge care il va elimina automat.

1. **Implementare**

In aceasta sectiune o sa vorbesc despre metodele pe care le-am implementat.

Text

Description automatically generated

Aceasta metoda este folosita ca si o interogare, folosita mai departe in functia insert folosita

pentru insert-ul asupra unui tabel.

Text

Description automatically generated

Text

Description automatically generatedAceasta metoda o folosim ca si o interogare pentru metoda delete pe care o folosim pentru a sterge un rand dintr-o tabela.

Text

Description automatically generated

Aceasta este metoda cu ajutorul careia stergem un Obiect cu id specific dintr-un tabel.Se aseamana destul de mult cu celelalte metode, sigura diferenta este a querry-ul folosit.

Text

Description automatically generatedAceasta metoda este folosita pentru a obtine o interogare asupra tabelului de care apartine obiectul. Aceasta metoda o vom folosi mai tarziu in cadrul metodei de update.

Text

Description automatically generated

Text

Description automatically generated

Acesta metoda o folosesc pentru a stoca datele obtinute prin interogarile asupra bazei de date, folosindu-ma mai apoi de ea si in metoda buildTable pentru a putea afisa tabelele.

Text

Description automatically generated

1. **Rezultate**

Aplicatia functioneaza conform cerintelor, exceptand eliberarea unei chitante la fiecare comanda.

Am testat fiecare metoda ca si cum eu as fi un utilizator rau-intemeiat pentru a-i afla defectele, insa am fost placut surrins sav ad ca merge. Aplicatia nu te lasa sa introduci date eronate care ar putea sa strice baza de date sau intructiunile. Toate campurile care trebuie sa contina doar numere, nu permit introducerea de caractere si asa mai departe.

O eroare pe care nu am apucat sa o corectez e faptul ca la stergerea unui produs din cosul de cumparaturi, stocul nu se reactualizeaza la cel initial dinaintea introducerii acestuia in cos.

1. **Concluzii**

Acest proiect a fost o buna experienta de invatare deoarece a fost abia a doua oara cand am lucrat cu o baza de date in Java. Mi-am reactualizat cunostinele de cum sa conectez o baza de date cu aplicatia measi spre mirarea mea a fost putin complicat, dar am reusit sa ma descurc.

Javadoc a fost o metoda destul de interesanta si folositoare de a-mi documenta metodele prin comenatrii inaintea codului propriu zis al metodei.

Reflexivitatea a fost un alt minunat concept pe care l-am invatat si am incercat sa creez metode pe care sa le folosesc pentru proiectul viitor.

Structurarea pe nivele m-a ajutat sa creez un cod mai lizibil sim ai organizat.

1. **Bibliografie**

**-**Documentatia si presentarea pentru proiect si cursul

**-**[Introduction to JDBC | Baeldung](https://www.baeldung.com/java-jdbc)

**-**[Connect to MySQL with JDBC driver - Mkyong.com](https://mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/)

**-**Reflection in Java

-[Java Reflection Tutorial (jenkov.com)](http://tutorials.jenkov.com/java-reflection/index.html)

- SQL dump file generation

-[MySQL :: MySQL Workbench Manual :: 6.5.2 SQL Data Export and Import Wizard](https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html)