**Assignment 3**

**Tehnici de programare**

**Tarta Manuel Vasile**

**Grupa 30225**

**Cuprins**

1. Obiectivul temei

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

3. Proiectare ( decizii de proiectare, diagrame, structuri de date, proiectare clase, relatii, interfata utilizator)

4. Implementare

5. Rezultate

6. Concluzii

7. Bibliografie

1. **Obiectivul temei**

Obiectivul temei 3 este realizarea unei aplicatii Order Management pentru procesarea clientilor unui warehouse. Pentru stocarea clientilor, produselor si a comenzilor facute se folosesc baze de date relationale. Totodata, aplicatia ar trebui sa fie structurata pe pachete folosind layered arhitecture si ar trebui sa aiba minim clase: Model classes, Business Logic classes, Presentation classes si Data access classes.

Printre obiectivele temei se mai numara folosirea javadoc pentru documentarea claselor, crearea unei interfata grafice care sa permita preluarea datelor si alegerea functiei dorite pe o tabela anume, folosirea Reflection pentru a implementa metoda generice care sa mearga pe toate cele 3 tabele.

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

Printre cerintele functionale se numara:

* Interfata ar trebui sa ii permita utilizatorului sa aleaga tabela pe care doreste sa realizeze operatia;
* Dupa, ar trebui sa ii permita sa introduca datele necesare pentru acea operatie;
* Apoi ar trebui sa ii permita utilizatorului sa selecteze operatia dorita si sa se realizeze acea operatie;
* Interfata ar trebui sa ii dea utilizatorului un feedback, daca operatia s-a realizat, daca nu s-a realizat care a fost cauza;
* Daca utilizatorul lasa anumite campuri goale, ar trebui sa se afiseze un mesaj care sa indice acest lucru, si sa nu se faca operatia selectata anterior;
* Daca utilizatorul selecteaza operatia de ViewAll, ar trebui sa se deschida o interfata ce contine un tabel cu toate obiectele de tipul dorit de utilizator (clienti sau produse).

Cerintele non-functionale sunt:

* Interfata ar trebui sa fie intuitiva, astfel incat utilizatorul sa isi poata da seama ce trebuie sa introduca si unde, pentru o comanda anume asupra unui tabel ce date trebuie sa introduca, pe ce buton trebuie sa apese.
* Daca operatia selectata a fost inserarea unui Order, programul ar trebui sa genereze o chitanta ca un fisier text.

Use case: inserarea unui client in baza de date

Primary actor: utilizatorul programului

Scenariul principal de succes:

* Utilizatorul apasa pe butonul Client care ne duce le interfata cu operatii pe clienti;



* Utilizatorul introduce toate datele corect (nume, adresa, email, varsta) pe interfata;
* Utilizatorul apasa butonul de add Client;
* Apare mesajul de „Inserare cu succes”;

Scenarii alternative de esec:

* Utilizatorul lasa campuri necompletate. In acest caz va aparea un mesaj de eroare care il va informa ca trebuie sa introduca toate datele;
* Este introdus un email care nu are patternul bun. Apare un mesaj de eroare care sugereaza ca trebuie introdus un email corect;
* Varsta nu este in intervalul cerut (14-99). Se deschide un mesaj de eroare care avertizeaza ca varsta nu este in interval;
* Varsta nu este un numar intreg. In acest caz apare un mesaj de eroare care ii indica utilizatorului ca datele au fost introduse gresit.

Use case: stergerea unui client din baza de date

Primary actor: utilizatorul programului

Scenariul principal de succes:

* Utilizatorul alege butonul de Client si este deschisa interfata pentru operatii pe tabela Client;
* Acesta introduce id-ul clientului pe care doreste sa il stearga din baza de date;
* Apasa pe butonul de delete Client;
* Clientul este sters din baza de date si utilizatorul primeste mesajul de confirmare cum ca stergerea clientului a avut succes

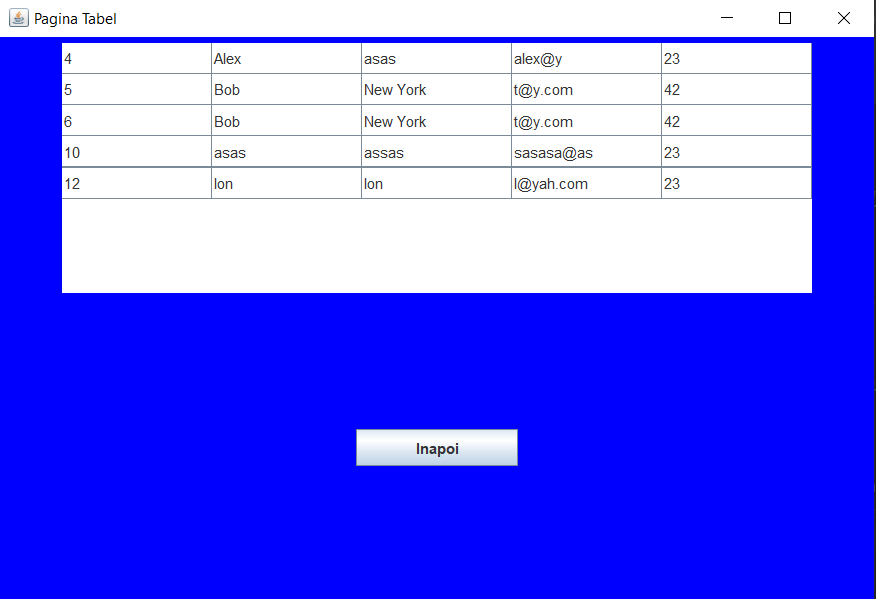
Scenarii alternative de esec:

* Utilizatorul nu insereaza id-ul corect, insereaza intr-un camp gresit sau nu insereaza deloc. In aceste cazuri se va afisa un mesaj de eroare care il va avertiza pe utilizator ca datele nu au fost introduse corect;
* Utilzatorul insereaza un id pe care operatia de findById apelata in spate nu il gaseste. In acest caz, stergerea nu se mai face si este afisat un mesaj de eroare cum ca clientul cu acel id introdus nu a fost gasit.

Use case: view all clients

Scenariul principal de succes:

* Utilizatorul apasa pe butonul care ne duce la interfata cu operatii pe clienti;
* Pe aceasta interfata, apasa pe butonul de View all clients;
* Se deschide o interfata care contine un tabel cu toti clientii din baza de date;



Scenariul de esec:

* Nu exista neaparat un scenariu de esec in acest caz, doar daca utilizatorul apasa pe alt buton dintre cele 3 de pe prima interfata, caz in care trebuie sa se intoarca la prima interfata si sa aleaga interfata buna.

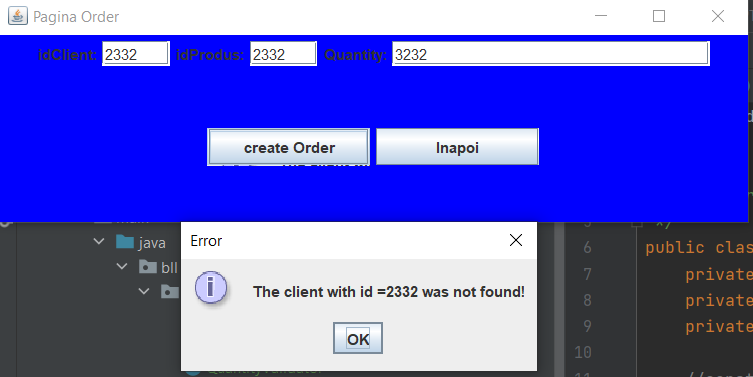
Use case: inserare Order

Scenariul principal de succes:

* Utilizatorul apasa pe butonul Order de pe prima interfata;
* Ajunge pe interfata de operatii pe tabela Order unde are optiunea de a insera un order nou;
* Utilizatorul insereaza datele necesare pentru inserarea unui Order, si anume id-ul clientului, id-ul produsului si cantitatea dorita din acel produs.
* Acesta apasa pe butonul de creare Order;
* Cantitatea dorita nu este mai mare decat stock-ul; stock-ul pentru acel produs este decrementat cu valoarea dorita pentru acel order;
* Se face update in Produs cu noua valoare a stock-ului;
* Apare mesajul care ii spune utilizatorului ca inserarea a avut succes.

Scenarii alternative de esec:

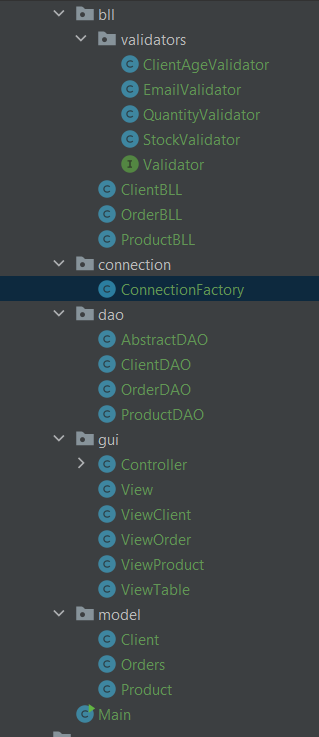
* Datele de intrare introduse nu sunt corecte, unul dintre id-uri sau cantitatea dorita nu sunt numare intregi, sau utilizatorul a uitat sa introduca una dintre valori, sau a introdus unde nu trebuie;
* Cantitatea dorita este mai mare decat stock-ul produsului respectiv, caz in care nu se face nici inserarea orderului, nici actualizarea produsului. Utilizatorul este avertizat de faptul ca produsul pe care il cere nu are stock-ul necesar;
* Unul dintre id-uri (client sau produs, sau chiar ambele) sunt id-uri care nu sunt existente in baza de date, caz in care utilizatorul va fi anuntat ca acel id nu a fost gasit si inserarea nu se va mai realiza



Celelalte use case-uri despre care nu am discutat functioneaza exact la fel ca acestea, doar ca pe alte tabele.

1. **Proiectare ( decizii de proiectare, diagrame, structuri de date, proiectare clase, relatii, interfata utilizator)**

Din punct de vedere al proiectarii, am impartit programul in mai multe pachete, folosindu-ma de ideea de Layered Arhitecture.



Am un pachet de Business Logic in care am clase care valideaza datele de intrare date de utilizator (verifica daca emailul are un pattern cerut, daca varsta este intr-un anumit interval, daca stock-ul si cantitatea dorita sunt numere pozitive). Tot in acest pachet am si clase care verifica pentru cele 3 clase care mapeaza baza de date daca datele sunt bune, si daca sunt bune, fac operatiile cerute ( adica cele de insert, update, find si view all). Aceste clase au nume cum ar fi ClientBLL. Nu aduc nicio logica in plus, doar combina logica din doua locuri diferite: validarea si realizarea operatiei propriu-zise. Urmeaza pachetul connection in care am metodele care realizeaza conexiunea cu baza de date din mySQL. Am metode pentru realizarea conexiunii si metode de close care inchid conexiunea cu baza de date, statementurile si resultSet-urile dupa ce am realizat operatia dorita. Apoi, am pachetul dao in care am clasa AbstractDAO. Aici am o implementare generica folosind Java Reflection pentru toate operatiile pe tabelele din baza de date. Facand o astfel de implementare generica, am putut folosi aceste metode pentru toate cele 3 tabele, trebuind doar sa le apelez pentru obiectele pe care trebuie.

In pachetul gui am clasele necesare pentru realizarea interfetei grafice. Am un controller in care am implementat ActionListenerii pentru butoane cu ajutorul carora leg interfata de functionalitatile pe care trebuie sa le realizeze. Pe langa asta am View-urile care reprezinta partea vizuala, locul in care utilizatorul poate sa introduca datele si sa apese pe butoane pentru a da o anumita comanda ce se va executa in spate.

In pachetul model am cele 3 clase care mapeaza tabelele din baza de date. Pe aceste clase facem operatiile implementate. Inseram clienti, produse sau orders, editam clienti sau produse, stergem clienti sau produse, vizualizam toti clientii sau toate produsele din tabele.

1. **Implementare**

In clasele Client, Product si Orders am doar campurile care sunt si in tabelele din mySQL, constructori care sa fie potriviti pentru tipul de operatii (pentru Reflection a fost nevoie de constructorul fara parametrii, pentru insert trebuie constructor fara id, pentru update constructor cu toti parametrii). Pe langa constructori in aceste clase mai sunt doar setteri si getteri.

In clasa AbstractDAO se gaseste implementarea metodelor care lucreaza pe baza de date. Pentru metoda de find am facut un select in functie de id (prima data am construit Stringul de query, pe care l-am dat la statement). Dupa ce am primit rezultatul in ResultSet, l-am trimis unei metode createObjects care construieste cu Reflection un obiect de tipul celui care a apelat metoda de find si il returneaza. Cream un obiect de tipul bun folosind metoda newInstance, dupa care am stocat intr-un obiect de tipul Method fiecare setter pe rand. La fiecare pas, apelam acest setter pe field-ul actual cu valoarea actuala din ResultSet. La final returnam obiectul(la findById) si lista de obiecte la FindAll.

Pentru insert, partea de Reflection a fost pentru a afla numele clasei, pentru a lua numele fieldurilor si a le pune in query, dar si valorile field-urilor ca sa le setam la statement-ul final de insert. Field-urile le-am parcurs intr-un for, le-am accesat cu getDeclaredFields si le-am luat valoarea cu metoda get(). Pentru a face cat mai generic, la setarea valorilor in statement am folosit setObject().

Metoda de Update merge pe acelasi concept ca si insert. Parcurgem toate fieldurile, prima data luam numele si il punem in query, dupa parcurgem si luam valorile si le setam in statement.

Pe de alta parte, Delete merge ca si FindById. Stergem in functie de un id deci punem in query clauza where in functie de id si in statement setam valoarea din id.

Clasele CliendDAO, ProductDAO si OrderDAO iau acele metode generale care merg pe orice tabele si le particularizeaza pe tipul lor. Apelandu-le cu tipul lor, transforma acel obiect general de tip T intr-o instanta a obiectului lor si asa Reflection o sa functioneze acum pe tipul clasei lor. Nu exista nicio diferenta intre aceste 3 clase, doar ca sunt tipuri diferite.

Clasele ProductBLL, ClientBLL si OrderBLL iau acele operatii pe bazele de date de inserare, update, delete si find si le particularizeaza si mai tare in functie de tipul obiectelor pe care se apeleaza. Aceasta particularizare se face prin includerea de validatori care trebuie sa fie indepliniti inainte de realizarea operatiei. De exemplu pentru client exista validatori pentru email si pentru varsta care trebuie sa fie in intervalul 14-99. Pentru Product, stock-ul trebuie sa fie >0, la fel pentru Orders in cazul cantitatii dorite. Daca aceste conditii nu sunt indeplinite, este aruncata o exceptie si operatia dorita nu se mai realizeaza. Utilizatorul trebuie sa insereze alte date, care sa treaca de aceste teste de validare.

1. **Rezultate**

De rezultate propriu-zis nu sunt foarte multe de zis. Inserarea trebuie sa insereze o linie in tabela pentru care se apeleaza, update-ul trebuie sa actualizeze o linie existenta, delete-ul trebuie sa stearga o linie existenta si find trebuie sa gaseasca un obiect in functie de un id dat ca parametru. Pentru a scoate in evidenta aceste rezultate, fara a trebui sa te uiti neaparat in baza de date, am facut sa se afiseze pe interfata niste mesaje care sa descrie statusul operatiei. Astfel ca, daca operatia s-a realizat cu succes am spus asta, daca nu s-a realizat am zis din ce cauza (parametrii nu sunt introdusi, sunt gresiti, nu exista un anumit obiect pe care se face operatia). Asa am facut pentru toate operatiile.

Tot din punct de vedere al rezultatelor se poate vorbi si despre acea chitanta care a fost ceruta o data cu inserarea unui order nou. Pentru ca un order nou sa poata fi inserat, trebuie ca clientul sa existe, produsul sa existe si cantitatea sa nu fie mai mare ca stocul disponibil pentru acel produs. Daca toate acestea sunt indeplinite, produsul este actualizat cu noul stoc ramas disponibil, order-ul este inserat in tabela si o chitanta este generata. Aceasta nu contine foarte multe date, doar datele despre client, datele despre produs si cantitatea ceruta.

Chitanta numarul 1  
Client [id=4, name=Alex, address=asas, email=alex@y, age=23]  
Product [id=24, name=pere, stock=1]  
Cantitatea dorita: 3

1. **Concluzii**

Aceasta tema a fost destul de interesanta datorita Reflection. Lucrul cu baze nu a fost ceva nou pentru ca am facut un proiect si semestrul trecut cu asta, dar Reflection a fost ceva nou. E chiar destul de folositor sa poti face metode care sa mearga pe mai multe tipuri de tabele (daca nu chiar toate).

1. **Bibliografie**

[1] materialul postat pe Teams cu privire la aceasta tema: ASSIGNMENT\_3\_SUPPORT\_PRESENTATION