**Facultatea de Automatica și Calculatoare**

**Calculatoare și Tehnologia Informației**



**Tehnici de Programare**

**Documentație**

**Tema 3:**

**Order Management Application**

**Profesor îndrumator: Realizat de:**

**Cristina Pop Câmpean Casiana Ștefana**

**Grupa: 30228**

**Cuprins**

**1. Obiectivul temei**

**2. Analiza problemei, asumptii, modelare, scenarii, cazuri de utilizare, erori**

**3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase,**

**interfete, relatii, packages, algoritmi, interfata utilizator, modul de tratare a erorilor)**

**4. Implementare**

**5. Testare**

**6. Rezultate**

**7. Concluzii**

**8. Bibliografie**

**1. Obiectivul temei**

Realizarea unei aplicatii pentru managemenul comenzilor facute de clienti. Se va realiza cu ajutorul unei baze de date care stocheaza date despre clienti si produse. Aplicatia va fi impartita pe layere care vor contine mimin 4 pachete:

Model: contine clasele model(Order, Customer, Product)

Bussiness Logic: contine logica aplicatiei

Presentation: contine interfata grafica

Data access: contine accesul la baza de date

**Obiective secundare**

Conectarea la o baza de date aflata in MySql;

Realizarea tehnicilor de reflectie;

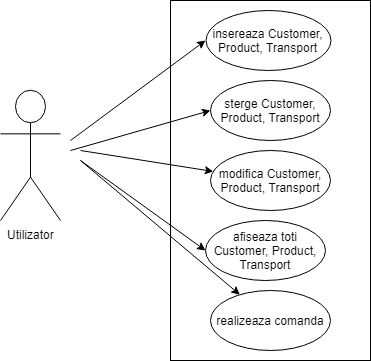
Crearea de JTable utilizand tehnici de reflectie;

Realizarea ferestrelor separate pentru operatiile pe tabelele bazei de date, mai exact inserare, stergere, modificare si selectare ;

Realizarea proiectului folosind javadoc

**2. Analiza problemei, asumptii, modelare, scenarii, cazuri de utilizare, erori**

Pentru a realiza proiectul, a fost necesar sa analizam cu atentie cerintele temei. Din enunt si din barem reiese ca va trebui sa creem minim patru pachete cu clasele corespunzatoare, deaorece va trebui sa avem un proiect organizat pe layere. Pe langa aceasta, se cere cere realizarea unei metode pentru a construi un JTable cu ajutorul tehnicilor de reflexie, pentru care este necesara o aprofundare a acestor noi notiuni. De asemenea, pe langa dcumentatia propriu-zisa, trebuie folosit javadoc pentru care este la fel necesara informarea asupra acestui concept.



Utilizatorul poate realiza mai multe operatii, una dintre ele fiind discutata in cele ce urmeaza:

Use case: Realizare comanda

Actorul: utilizatorul aplicatiei

Asumptii:date introduse corect

Scenariu succes:

1. Utilizatorul apasa pe butonul Make Order

2. El introduce corect id-urile clientului, produsului, transportului si al cantitatii dorite

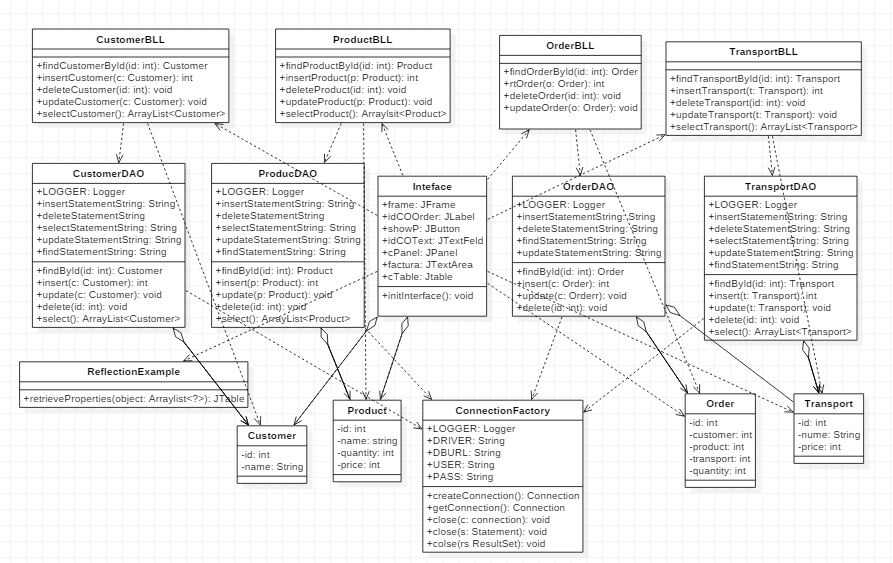
3. In functie de cantitate, se va afisa pe ecran bonul facturii sau un mesaj de indisponibilalte a produsului comandat.

**3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator, modul de tratare a erorilor)**

**Decizii de proiectare**

Pentru a implementa un poriect care utilizeaza o baza de date am ales si pentru a respecta cerintele temei am ales sa impart proiectul in 4 pachete, pentru care am implementat clasele model care corescpund cu tabelele din baza de date, clasele de dataAccess care realizeaza coneiunea cu baza de date si realizaeaza operatiile, pachetul de businessLogic si pachetul presentation care contine interfata grafica.

**Diagrame UML**



**Structuri de date**

La acest proiect am folosit structuri de date de tip ArrayList. Acestea au fost folosite pentru a retine date despre obiectele de tip Customer, Product, Transport si Order, fiind folosite mai ales in procesarea datelor din baza de date si pentru afisarea JTable.

**Proiectare clase**

Pentru acest proiect a fost necesara proiectarea unui total de 15 clase. Aceste clase sunt organizate in 4 pachete, mai exact model, dataAccess, bussinessLayer si presentation. Clasele din model corespund cu tabelele aflate in baza de date, mai exact Customer, Transport, Product si Order. Fieacare atribut al claselor corespunde cu coloanele tabelelor din baza de date.

Clasele din dataAcces realizeaza conexiunea la baza de date si realizeaza operatiile cerute folosite pe elementele din baza de date. Logica de business este un layer superior care va apela metodele din clasele din pachetul model si pachetul dataAccess, iar clasa Interface contine implementarea interfetei grafice.

**Packages**

Pentru acest proiect au fost necesare 4 pachete, conform cerintei. Primul pachet numit Model contine clasele care reprezinta tabelele din baza de date, acestea fiind: Customer, Order, Product, Transport. Urmatorul pachet este cel de dataAccess. Acest pachet contine conexiunea la baza de date si procesarea operatiilor cerute, mai exact INSERT,UPDATE, DELETE, SELECT ALL. Pachetul bussinessLayer contine logica aplicatiei, iar pachetul presentation contine interfata grafica. Acest proiect este structurat pe layere, fiecare layer de sus folosindu-se de metodele din layerele de mai jos, de exemplu BusinessLayer se va folosi de layerele dataAccess si model pentru a realiza operatiile dorite.

**Interfata utilizator**

Interfata utilizator este alcatuita din mai multe paneluri care corespund cerintelor proiectului.Cand se deschide interfata utilizator, apare meniul principal si in functie de ce buton vom apasa se va deschide alt panel. Meniul contine 4 butoane: Customer Operations, Product Operations, Transport Operations si Make Order. Primele 3 butoane daca vor fi apasate vor aparea mai multe elemente pentru realizarea operatiilor de inserare, strergere, modificare si view all. Utilizatorul trebuie mai intai sa introduca in campurile de pe ecran datele dorite conform informatiilor din tabel si sa selecteze actiunea dorita. Ultimul buton are rolul de a realiza o comanda. Utilizatorul trebuie sa introduca id-urile clientului, al produsului, al transportului si al comenzii si sa scrie cantitatea dorita. Daca produsul are stocul mai mic decat cantitatea dorita se va aisa un mesaj de understock, altfel se va afisa bonul comenzii cu pretul total si alte informatii suplimentare.

Ultimul pachet contine clasa care implementeaza interfata grafica. Mai intai am declarat toate componentele necesare pentru datele de intrare si de iesire ale proiectului. Proiectul fiind mai complex, am realizat design-ul a 5 paneluri care contin mai multe elemente, in felul urmator:

-MenuPanel contine 4 butoane care ne vor trimite fiecare la unul dintre celeletepaneluri, in funtie de actiunea dorita

-CustomerPanel contine 5 butoane, 2 Label-uri, 2 JtextField-uri si un JscrollPane. Aici putem realiza inserarea, stergerea modificare si afisarea clientilor din aza de date in JTable.

Urmatoare doua panel-uri, TransportPanel si ProductPanel sunt pe aceeasi idee.

Ultimul panel, OrderPanel contine 2 butoane, 4 Label-uri si 4 JtextField-uri si un JTextArea. In functie de datele completate in TextField-uri se va afisa in JTextArea bonul fiscal aferent comezii sau un mesaj cu stoc indisponibil. Pe acest bon se vor afisa date despre numele clientului, numele produsului, numele transportului, cantitatea ceruta, pretul total al produselor bazat de cantitatea ceruta, pretul transportului, iar la final se calculeaza suma totala.

In singura metoda initInteface() am initializat toate componentele necesare si am realizat design-ul. Fiecare Panel are Layout absolut, asfel fiecare cmponenta fiind pusa cu atentie dupa pixeli si dupa intaltime sau latime. Fiind nevoie de ma multe paneluri, am implementat constructori pentru fiecare buton, apasarea butoanelor de Done din fiecare Panel va modifica vizibilitatea panel-urilor.

**4 . Implementare**

Proiectul conține 15 clase organizate in 4 pachete :

**Pachetul businessLayer :**

* CustomerBLL
* ProductBLL
* OrderBLL
* transportBLL

**Pachetul dataAccess :**

* ConnectionFactory
* CustomerDAO
* OrderDAO
* TrasnportDAO
* ReflectionExample

**Pachetul model :**

* Customer
* Order
* Product
* Transport

**Pachetul presentation :**

* Interface

Pachetul model contine clasele care reprezinta tabelele din baza de date. Clasa Customer contine atributele id si name, Order contine id, customerId, productId si idTransport, Product contine id, name, quantity si price, iar clasa Transport contine id, name si price. Toate aceste clase sunt alcatuite din constructori si meotdele de set si get care vor fi folositoare in procesarea datelor din baza de date pentru operatiile necesare.

Pachetul dataAccess contine in primul rand conexiunea la baza de date realizata in clasa ConnectionFactory. Acesta conexiune la MySql se va realiza cu ajutorul obiectului de tip Logger si a Stringurilor DRIVER, DBURL ,USER, PASS. Pe langa metoda de creare a coenxiunii, aceasta clasa mai contine si alte metode de inchidere a conexiunii, a Statement-ului sau a ResultSet-ului.

Urmatoarea clasa, CustomerDAO contine metodele care ne vor ajuta sa facem operatiile necesare pe baza de date, mai exact INSERT, DELETE, SELECT, SELECT ALL. Aceasta clasa contine de asemenea un obiect de tip Logger si va contine mai multe Stringuri care vor reprezenta cate un stament, ca de exemplu:

**private** **final** **static** String deleteStatementString = " DELETE FROM customer where id = ? " ;

Fiecare dintre statement-urile declarate in clasa vor fi folosite pentru metoda de findById, insert, delete, update si select. In fiecare din aceasta metoda se va realiza conexiunea la baza de date si se va declara un obiect de tip ResultSet, care ne va ajuta sa preluam, sa schimbam sau sa determinam valorile campurior din baza de date. Daca de exemplu vom face delete afler id, atunci cu ajutorul ResultSet-ului noi vom selecta din deleteSteemetString id-ul corespunzator,ar cu ajutorul obiectului de tip preparedStatement vom executa operatia.

deleteStatement.executeUpdate();

La finalul fiecarei dintre aceste metode vom inchide conexiunea, ResultSet-ul si Statement-ul cu ajutorul metodelor din connectionFactory.

ConnectionFactory.close(deleteStatement);

ConnectionFactory.close(dbConnection);

Celelate clase din acest pachet, mai exact ProductDAO, OrderDAO si TransportDAO au fost realizate in exact aceeasi idee ca si clasa discutata mai sus. Spre deosebire de celelalte, OrderDAO are cu o metoda mai putin decat celellate, mai exact cea care realizeaza „ SELECT \* FROM order ” . Aceasta nu este implementata deoarece nu ne este cerut sa vedem ce avem in tabelul order, doar in celelalte, si de aceea la fel in OrderBLL vom avea cu o metoda in minus.

Ultima clasa din acest pachet, ReflexionExample utilizeaza tehnicile de reflexie, metoda retrieveProperties primeste ca paramentru o lista de obiecte si returneaza un JTable.

**public** **static** JTable retrieveProperties(ArrayList<?> object,**int** c)

In aceasta **metoda se vor realiza** doua lisede tip Object pentru a putea realiza un nou JTable pe baza lor.

JTable table=**new** JTable ( rand , columns );

Un JTable se defineste astfel, rand va contine o matrice compusa din toate randurile aflate in tabelul corespunzator, iar colmns va contine numele coloanelor tabeleului. Ai vom parcurge cu doa foreach-uri pentru a afla datele dorite. Ne vom folosi de obiectul de tip Field pentru aflarea datelor necesare astfel:

columns[ i++ ] = field.getName();

value = field.get ( ob );

rows.add ( value.toString() );

Dupa acestea, pentru aflarea listei de Object[][]rand vom parcurge cu 2 for-uri datele din rows pentru a le pune corespunzator in lista noastra.

La final, vom initializa jtable si vom returna acest tabel. Aceasta functie va fi folosita mai apoi in interfata cand va fi dorita actiunea de afisare a tuturor datelor din tabel.

Pachetul BusinessLayer contine logica aplicatiei. Acest pachet se va folosi de layerele discutate mai sus, mai exact model si dataAcces. Clasa CustomerBLL contine metodele findCustomerById, insertCustomer, deleteCustomer, updateCustomer si selectCustomer, toate aceste metode apeland metodele din CustomerDAO aferente cu operatiile corespunzatoare procesarii elementelor din baza de date.

Toate celelate clase sunt asemanatoare cu cea discutata mai sus, fiecare BLL folosindu-se de metodele din clasele corespunsatoare cu pachetul dataAccess.

**5. Testare**

Testarea a fost realizata la inceput in main. Dupa realizarea conectarii la MySql la baza de date creata, am verificat pe rand daca operatiile de inserare, stergere, selectie si modificare merg corespunzator. Am verificat in MySql daca se modifica conform operatiilor verificate. Mai apoi am realizat metoda de creare a unui JTable folosit tehnicile de reflexie. Acesta metoda a fost de asemenea testata intr-un main. Pentru a crea un nou JTable am folosit constrctorul JTable(Object[][] rowData, Object[] columnNames), pentru care am testat in main corectitudinea continului celor 2 liste de obiecte.

Dupa ce fieacreoperatie a fost testata in consola, cu atenta observatie asupra bazei de date din MySql, am trebut mai departe la testarea operatiilor si a afisarii JTable-ului in interfata grafica, iar acesta, dupa mai multe modificari a fost reusita.

**6. Rezultate**

In urma realizarii acestui proiect,am reusit sa obtin o aplicatie de gestionare a comenzilor realizate de clienti. Am putut insera, sterge, selecta si modifica conform asteptarilor, toate actiunile efectuate avand efect asupra bazei de date din MySql.

**7. Concluzii**

In concluzie, in urma acestui proiect am reusit sa realizez o aplicatie mai buna care sa utilizeze o baza de date. Am invatat despre tehnicile de reflexie si depre organizarea pe layere, si de asemenea am invatat cum ar trebi realizata documetatia javadoc.

Mi-am imbunatatit proiectarea interfetei grafice, am realizat pentru prima oara o interfata care are Layout absolut, iar deoarece am fost nevoita sa fac mai multe panel-uri, am realizat astfel mai multe asezari pe ecran in functie de elementele necesare, iar tot in cadrul interfetei grafice am reusit sa extrag datele din baza de date intr-un JTable.

**8. Bibliografie**

1. <https://docs.oracle.com/javase/tutorial/uiswing/components/table.html>
2. <http://www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/>
3. <http://christoph-burmeister.eu/?p=1556>
4. <https://dzone.com/articles/layers-standard-enterprise>
5. <http://tutorials.jenkov.com/java-reflection/index.html>
6. <https://bitbucket.org/utcn_dsrl/pt-layered-architecture/src/master/src/main/java/start/>
7. <https://stackoverflow.com/questions/5600051/java-swing-how-to-toggle-panels-visibility?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa>
8. <http://www.codejava.net/ides/eclipse/how-to-generate-javadoc-in-eclipse>