

ORDER MANAGEMENT

Documentație

Pentru: Tehnici de Programare

Student: Luncian Lidia, grupa 30227

Cuprins:

1. Obiectivul temei
2. Analiza problemei, modelare, scenarii, cazuri de utilizare
3. Proiectare
4. Implementare
5. Rezultate
6. Concluzii
7. Bibliografie
8. Obiectivul temei

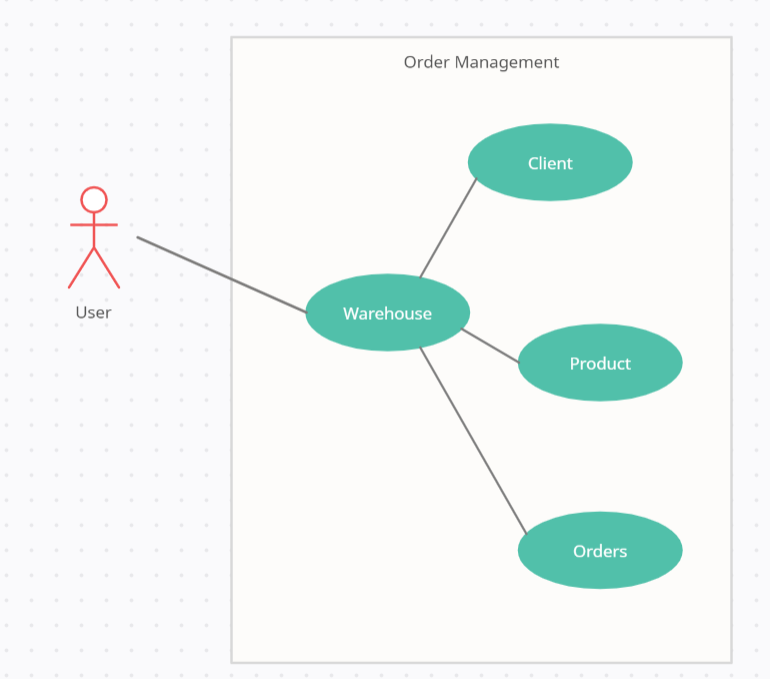
Obiectivul acestei teme a fost acela de a realiza o aplicație cu o intefață „User Friendly” pentru gestionareaa unui depozit. Această aplicație lucrează cu o bază de date ce permite adăugarea de noi clienți, produse sau comenzi în limita stocului disponibil. Aceasta mai permite și editarea sau ștergerea din baza de date a clienților, a produselor sau a comenzilor. În plus pentru comenzi se poate genera și o chitanță în format .txt.

Această aplicație este proiectată conform paradigmelor de programare orientată pe obiecte, folosind limbajul de programare Java. Am folosit modelul arhitectural Layers pentru modelarea pachetelor și a claselor. Acest proiect conține principalele pachete dataAccessLayer, businessLayer, model și presentation. De asemenea am folosit tehnici de reflexie pentru a crea metode pentru a accesa baza de date și pentru a genera tabelele cu date.

1. Analiza problemei

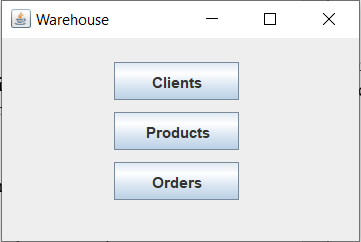
În această secțiune voi prezenta modul de interacționare al utilizatorului cu interfața grafică. Această interfață este intuitivă, ușor de folosit oricine ar fi utilizatorul un client, un lucrător sau managerul, dar de preferat ar fi managerul.

Mai jos este prezentat use-case-ul acestei teme.

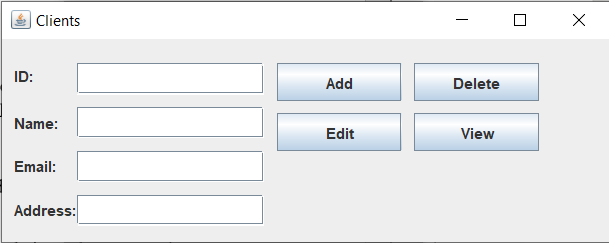


Userul interacționează prima dată cu interfața principală. Aceasta conține 3 butoane ce îi permit utilizatorului să aleagă unde dorește să facă modificări, în tabela client, product sau orders.

Această interfață arată astfel:



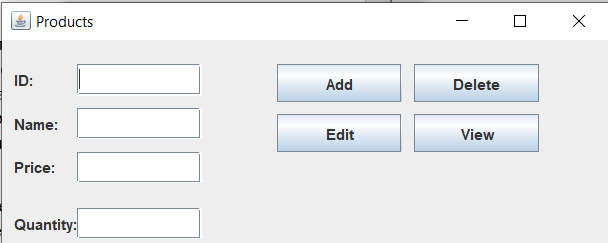
Voi lua pe rând butoanele și voi prezenta interfața pentru fiecare tabel ce corespunde bazei de date. Dacă apăsăm butonul Clients va apărea interfața dedicată acestora ce arată astfel:



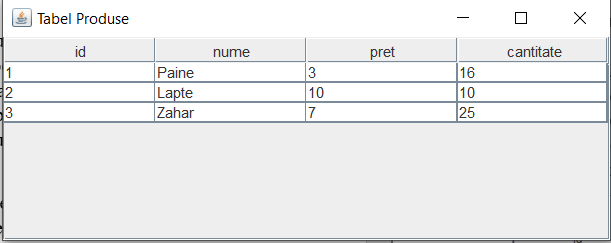
Aceasta conține 4 etichete ce corespund numelor datelor din tabelul client din baza de date: ID - id, Name - nume, Email - email, Address - adresa, 4 text field-uri unde utilizatorul trebuie să introducă datele și 4 butoane. Butonul Add corespunde funcției de insert din limbajul SQL, și inserează un nou client. Id-ul clientului în baza de date este auto increment, adică utilizatorul nu trebuie neapărat să introducă id-ul pentru a insera un nou client. Cu toate acestea id-ul este absolut necesar pentru funcțiile de edit și delete. Dacă acesta nu a fost introdus va apărea un mesaj de eroare care vaa cere introducerea id-ului. Butonul Edit corespunde funcției update din SQL, iar butonul Delete corespunde funcției delete din SQL. Folosind aceste butoane utilizatorul poate modifica un client, sau îl poate șterge complet din baza de date. Butonul View generează un nou frame care cuprinde toate datele din tabelul clienților. Toate metodele pe care aceste butoane le apeleză sunt implementate folosind reflexia. Pentru id-ul și email-ul clienților s-au folosit validatori.



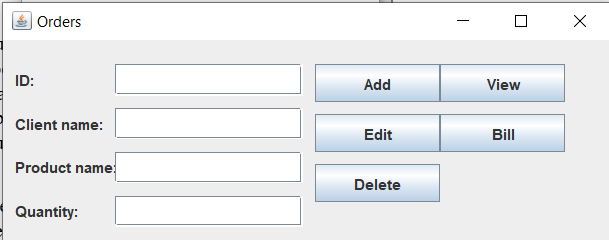
Pentru butonul Products din interfața principală apare interfața destinată produselor, ce arată astfel:

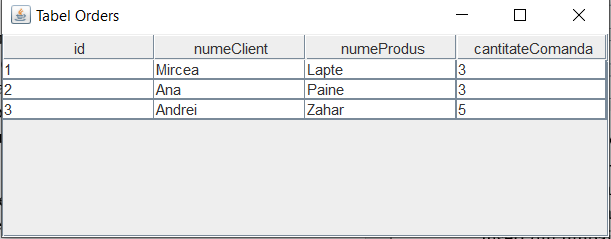


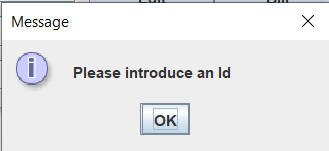
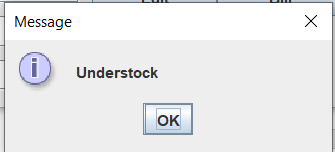
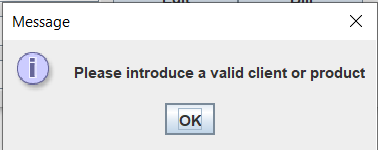
Aceasta conține 4 etichete ce reprezintă datele din tabelul din baza de date: ID - id, Name - nume, Price - preț, Quantity - cantitate, 4 text field-uri unde utilizatorul va introduce datele și 4 butoane. La fel ca și la client butoanele implementează aceleași funcții. Butonul Add corespunde funcției de insert din limbajul SQL, și inserează un nou produs. Id-ul produsului în baza de date este auto increment, adică utilizatorul nu trebuie neapărat să introducă id-ul pentru a insera un nou produs. Cu toate acestea id-ul este absolut necesar pentru funcțiile de edit și delete. Dacă acesta nu a fost introdus va apărea un mesaj de eroare care va cere introducerea id-ului. Butonul Edit corespunde funcției update din SQL, iar butonul Delete corespunde funcției delete din SQL. Folosind aceste butoane utilizatorul poate modifica un produs, sau îl poate șterge complet din baza de date. Butonul View generează un nou frame care cuprinde toate datele din tabelul poduselor. Toate metodele pe care aceste butoane le apeleză sunt implementate folosind reflexia.



Ultimul buton este cel de Orders. Apăsând acest buton utilizatorului îi va apărea o nouă interfață specifică comenzilor. Aceasta arată astfel:



 Aceasta conține 4 etichete ce reprezintă câmpurile din tabelul orders a bazei de date: ID - id, Client name - numeClient, Product name - numeProdus și Quantity – cantitateComandata, text field -uri unde utilizatorul poate introduce datele și 5 butoane. Butoanele Add, Edit, View și Delete au aceleași caracteristici ca și la client și produs. Butonul Add corespunde funcției de insert din limbajul SQL, și inserează o nouă comandă. Id-ul comenzii în baza de date este auto increment, adică utilizatorul nu trebuie neapărat să introducă id-ul pentru a insera un nou produs. Cu toate acestea id-ul este absolut necesar pentru funcțiile de edit și delete. Dacă acesta nu a fost introdus va apărea un mesaj de eroare care va cere introducerea id-ului. În plus dacă clientul sau produsul nu au fost găsite în baza de date va apărea un mesaj care va cere introducerea unui client sau produs valid. Iar dacă cantitatea pe care utilizatorul vrea să o comande nu există va apărea un mesaj de Understock. Butonul Edit corespunde funcției update din SQL, iar butonul Delete corespunde funcției delete din SQL. Folosind aceste butoane utilizatorul poate modifica o comandă, sau o poate șterge complet din baza de date. Butonul View generează un nou frame care cuprinde toate datele din tabelul comenzilor. Toate metodele pe care aceste butoane le apeleză sunt implementate folosind reflexia. Această interfață mai conține și butonul Bill care va genera chitanța pentru comanda cu id-ul specificat. Este obligatoriu să introducă id-ul.



1. Proiectare

Am folosit modelul arhitectural Layers pentru modelarea pachetelor și a claselor. Pachetul bll (Business Layer) cuprinde pachetul validators și clasele ClientBLL, OrderBLL și ProductBLL ce implementează logica de business a aplicației pentru fiecare dintre clasele pachetului model. Pachetul validators conține clasele: EmailValidator, IdClientValidator și interfața Validator.

Clasa ClientBLL implementează logica de business pentru clasa Client din pachetul model.

Clasa OrderBLL implementează logica de business pentru clasa Orders din pachetul model.

Clasa ProductBLL implementează logica de business pentru clasa Product din pachetul model.

Clasele EmailValidator și IdClientValidator implementează funcția validate declarată în interfață.

Pachetul connection conține doar clasa ConnectionFactory care realizează conexiunea cu baza de date folosind o singură instanță de obiect Connection prin implementarea modelului arhitectural Singleton.

Pachetul dao implementează logica DataAccess. Conține clasa abstractă AbstractDAO și clasele ClientDAO, ProductDAO și OrderDAO care extind clasa AbstractDAO pentru obiecte de tip Client, Product și respectiv Orders. Clasa AbstractDAO implementează metodele ce vor interoga sau modifica baza de date, de insert, delete, update, folosind tehnica reflexiei astfel încât aceste funcții se vor face folosind aceleași metode, indiferent de tipul obiectului instanță al unei clase din pachetul model.

Pachetul model conține clasele Client, Orders și Product ce modelează obiectele principale ale aplicației. Acestea corespund tabelelor din baza de date, au aceleași câmpuri, aceleași tipuri de date și în ceeași ordine.

Pachetul presentation cuprinde clasele ce implementează view și controller din modelul MVC. Conține clasele: BillWriter unde este implementată metoda de generare a unei chitanțe, ClientInterface unde este creată interfața pentru client, OrderInterface unde este creată interfața pentru orders, ProductInterface unde este creată interfața pentru product, TableInterface unde este creată interfața pentru tabelele de date, View unde este creată interfața principală cu cele trei butoane și clasa Controller unde sunt implementate clasele Listener pentru fiecare buton din fiecare interfață.

Pachetul start conține clasa Start unde este implementată metoda Main, de unde începe execuția programului.

Diagrama UML:

Chart, timeline

Description automatically generated

Pe segmente:

Pachetele dao și bll:

Diagram

Description automatically generated

Pachetele connection și model:

Diagram

Description automatically generated

Pachetul presentation:

Diagram

Description automatically generated

Diagram

Description automatically generatedDiagram

Description automatically generated

Diagram

Description automatically generated

1. Implementare

Voi analiza clasele din pachetele dao și bll și voi explica alegerile făcute în ce privește implementarea claselor din acestea, precum și clasa BillWriter.

Clasa AbstractDAO este clasa ce implementează operațiile de insert, update, delete și select, specifice bazelor de date, pentru obiectele instanță ale claselor din pachetul model. În această clasă se folosește tehnica reflexiei în Java pentru a nu fi nevoie să scriu de mai multe ori metode similare particularizate pe fiecare clasă ce modelează obiectele temei. În acest sens am fost definit un tip generic asociat clasei AbstractDAO. Acest tip se va obține în mod dinamic la runtime în funcție de nevoie. În aceasă clasă am implementat în plus față de funcțiile cerute o metodă care caută în tabele un obiect după numele său, și nu după id. Importat la această funcție este să găsească acest nume pentru a putea face o comandă. Dacă nu se găsește numele clientului care vrea să faca comanda și al produsului comandat apare un mesaj de eroare. Această metodă folosește interogarea ”SELECT \* FROM table WHERE name = name”. O altă metodă este cea de generare a unui tabel folosind o listă de obiecte. Această listă se obține apelând metoda findAll implementată tot în această clasă. Metoda createTable returnează un JTable.

Clasa ClientDAO poate implementa operațiile definite în AbstractDAO pentru Client. Clasa ProductDAO poate implementa operațiile definite în AbstractDAO pentru Product. Clasa OrderDAO poate implementa operațiile definite în AbstractDAO pentru Orders.

Pachetul bll conține clasele ce implementeză logica business pentru fiecare obiect în parte.

Clasa ClientBLL implementează logica business pentru client, apelând metodele, de insert, update, delete, findById, findByNume și createTable, pe un obiect de tipul ClientDAO. În această clasă sunt instanțiați și validatorii pentru client.

Clasa ProductBLL implementează logica business pentru product, apelând metodele definite în clasa AbstractDAO pe un obiect de tip ProductDAO.

Clasa OrderBLL implemetează logica business pentru orders. Apelează metodele descrise în clasa AbstractDAO pe un obiect de tipul OrderDAO.

Clasa EmailValidator implementează metoda validate din interfața Validator. Este folosit un cod regex pentru a asigura scrierea corectă a unui email pentru client.

Clasa IdClientValidator implementează metoda validate din interfața Validator. Sunt declarate 2 constante care determină limita inferioară și cea superioră a intervalului din care id-ul poate face parte. Am declarat limita inferioră -1, pentru ca utilizatorul să poată să nu introducă niciun id, ci să se auto incrementeze în tabel și să nu apară eroare sau chiar mai rău, să nu apară eroare, dar nici să se adauge.

Clasa BillWriter creează un fișier text în care vor fi scrise numele clientului care a plasat comanda, numele produsului comandat, prețul produsului la bucată, cantitatea de produs comandată și în final prețul total care se calculează înmulțind prețul pe unitate cu cantitatea.

Metoda findByNume:



Metoda createTable:



1. Rezultate

Am reușit să bifez toate subpunctele temei. Am folosit javadoc pentru documentarea proiectului și pentru generarea fișierelor JavaDoc aferente. Am creat conexiunea cu baza de date din MySQL unde au fost create 3 tabele: client, product și orders. Am creat interfața grafică ”User Friendly” ce permite utilizatorului să interacționeze ușor cu baza de date. Acesta poate să gestioneze baza de date prin introducerea de noi clienți, produse sau comenzi. Poate modifica datele deja introduse în baza de date, su le poate șterge. Utilizatorul poate genera și o chitanță daca dorește.

Am folosit tehnica reflexiei în Java pentru crearea metodelor ce interoghează baza de date, precum și pentru generarea tabelelor ce conțin datele din baza de date.

1. Concluzii

În urma realizării acestui proiect am învățat mai multe despre utilizarea modelelor arhitecturale Layers și Singleton. Am învățat despre modul de folosire al tehnicii de reflexie în Java pentru a scrie metode CRUD ce interacționează cu baza de date și pentru a crea tabele pe baza datelor dintr-o bază de date. De asemenea am învățat cum să lucrez cu această bază de date, cum să mă conectez, la ea și să creez metode atât pentru query cât și pentru implementarea funcției propriu-zise. Am învățat să creez și să folosesc validatori. Am aprofundat modul de creare al unui fișier text, lucrul cu metode abstracte, precum și materia predată anul trecut la baze de date, pentru construirea query-urilor.

Întotdeauna este loc pentru mai bun. Acesta este cazul și pentru acest proiect. Ar putea fi implementate metode care să afișeze doar clienții căutați, nu toată baza de date....

1. Bibliografie

[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/)

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

[Introduction to JavaDoc | Baeldung](https://www.baeldung.com/javadoc)

[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)

[Java: JOptionPane showMessageDialog examples (part 1) | alvinalexander.com](https://alvinalexander.com/java/joptionpane-showmessagedialog-examples-1/)

[java - Retrieving Data from JDBC Database into Jtable - Stack Overflow](https://stackoverflow.com/questions/27815400/retrieving-data-from-jdbc-database-into-jtable/43772751)

[java - Trying to display all data from database table into Jtable using reflection - Stack Overflow](https://stackoverflow.com/questions/49831334/trying-to-display-all-data-from-database-table-into-jtable-using-reflection)