MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

DOCUMENTATIE

TEMA *3*

Teglă Lidia-Sorina

Grupa 30226 | Anul 2 semestrul 2

CUPRINS

1. Obiectivul temei

Scopul principal al proiectului este de a proiecta si implementa o aplicatie pentru gestionarea comenzilor clientilor intr-un depozit.

Obiectivele secundare includ analizarea problemei si identificarea cerintelor, proiectarea aplicatiei de gestionare a comenzilor, implementarea aplicatiei de gestionare a comenzilor si testarea acesteia.

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

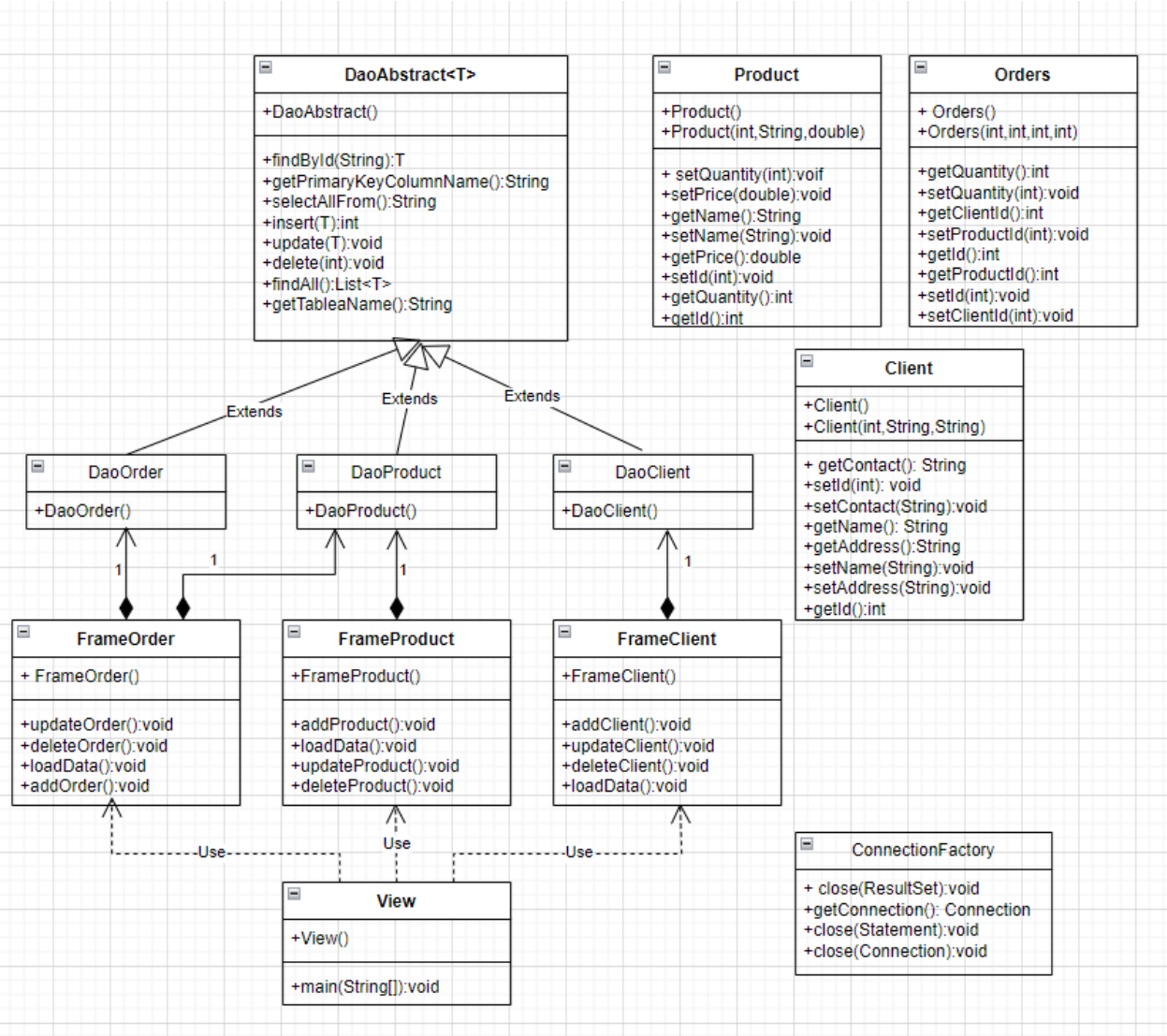
Caz de utilizare: Adaugare produs
Actor principal: angajat
Scenariul de succes principal:
1. Angajatul selecteaza optiunea de a adauga un nou produs
2. Aplicatia va afisa un formular in care detaliile produsului trebuie introduse
3. Angajatul introduce numele produsului, pretul si stocul curent
4. Angajatul face clic pe butonul "Adaugare"
5. Aplicatia stocheaza datele produsului in baza de date si afiseaza un mesaj de confirmare

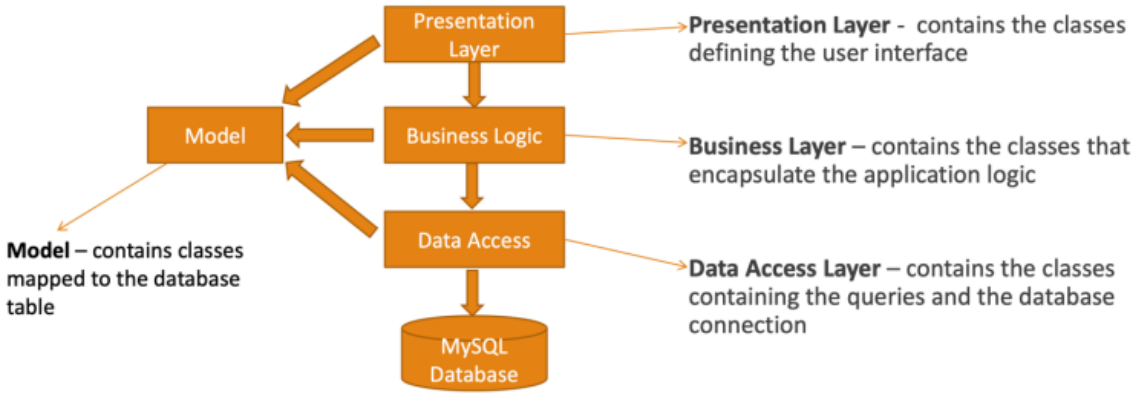
Secventa alternativa: Valori invalide pentru datele produsului
- Utilizatorul introduce o valoare negativa pentru stocul produsului
- Aplicatia afiseaza un mesaj de eroare si solicita utilizatorului sa introduca un stoc valid
- Scenariul revine la pasul 3.

Cerinte functionale:
- Aplicatia ar trebui sa permita unui angajat sa adauge un client nou
- Aplicatia ar trebui sa permita unui angajat sa adauge un produs nou
-Aplicatia ar trebui sa permita adaugarea de comezi noi
-Cantitatea din tabelul product va fi decrementa in functie de cantitatea comandata
-Daca o comanda va fi stearsa, cantitatea va reveni inapoi in tabela product

Cerinte non-functionale:
- Aplicatia ar trebui sa fie intuitiva si usor de utilizat de catre utilizator.

3. Proiectare





4. Implementare

Clasa DaoAbstract reprezintă o clasă abstractă pentru a crea obiecte DAO (Data Access Object) generice, care pot fi utilizate pentru a accesa datele dintr-o bază de date relațională. Aceasta utilizează reflectia Java pentru a accesa și manipula datele din baza de date, astfel încât clasele DAO care o extind să nu aibă nevoie să implementeze cod duplicat.

Clasa conține metode pentru a executa interogări de selectare, inserare, actualizare și ștergere a datelor din baza de date, și pentru a transforma obiectele din baza de date în obiecte Java și invers. Metoda executeSelectQuery este folosită pentru a executa interogări de selectare și returnează un ResultSet. executeInsertQuery, executeUpdateQuery și executeDeleteQuery sunt utilizate pentru a executa interogări de inserare, actualizare și ștergere, respectiv. Metoda createObjects este utilizată pentru a transforma obiectele din baza de date în obiecte Java.

Clasa este generică și acceptă orice tip de clasă ca argument de tip. Aceasta utilizează informațiile despre tipul clasei pentru a genera interogările SQL necesare, precum și pentru a accesa proprietățile obiectelor. Clasele DaoClient, DaoProduct și DaoOrder moștenesc clasa DaoAbstarct.

Clasa ConnectionnFactory conține numele Driver-ului (inițializat prin reflecție), locația bazei de date (DBURL) și utilizatorul și parola pentru a accesa serverul MySQL. Conexiunea la baza de date va fi plasată într-un obiect Singleton*. Clasa conține metode pentru crearea unei conexiuni, obținerea unei conexiuni active și închiderea unei conexiuni, a unui Statement sau a unui ResultSet.

Clasa FrameClient este o fereastră grafică de utilizator (GUI) care gestionează operațiile de adăugare, actualizare și ștergere a datelor clientului într-un sistem de gestionare a clienților. Aceasta utilizează o clasă DaoClient pentru a accesa baza de date și a efectua operațiile CRUD (create, read, update, delete) asupra obiectelor Client.

Fereastra conține patru câmpuri text pentru introducerea datelor clientului: idField, firstNameField, addressField și emailField, și trei butoane: addButton, updateButton și deleteButton. O tabelă clientTable afișează datele clienților existenți din baza de date și poate fi actualizată după ce se efectuează o operațiune CRUD.În constructor, se realizează inițializarea interfeței grafice a utilizatorului și se încarcă datele clientului în tabel.

Metodele addClient(), updateClient() și deleteClient() sunt utilizate pentru a adăuga, actualiza și șterge datele clientului din baza de date, respectiv. În plus, metoda loadData() este utilizată pentru a încărca datele clientului în tabel.

Clasa FrameOrder face parte din pachetul GUI. Ea reprezintă o fereastră grafică pentru gestiunea comenzilor dintr-un sistem de management al comenzilor. În cadrul clasei sunt utilizate mai multe elemente de interfață grafică precum JTable, JTextField, JButton și JLabel.

În constructorul clasei sunt create obiectele DaoOrder și DaoProduct pentru a permite accesul la baza de date. Apoi, sunt create câmpurile de text pentru introducerea datelor comenzii, butoanele pentru adăugarea, actualizarea și ștergerea comenzii, precum și o tabelă pentru afișarea datelor comenzilor. În

ceea ce privește layout-ul, acesta este definit prin intermediul unor panouri, astfel încât elementele sunt poziționate într-un mod coerent.

Mai multe metode sunt definite în cadrul clasei. Metoda addOrder() adaugă o comandă nouă în baza de date și actualizează stocul produsului corespunzător comenzii. Metoda updateOrder() actualizează o comandă existentă în baza de date. Metoda deleteOrder() șterge o comandă existentă în baza de date și actualizează stocul produsului corespunzător comenzii șterse. Metoda loadData() încarcă datele comenzilor din baza de date și le afișează în tabelul corespunzător.

Clasa FrameOrder o fereastră grafică (GUI) pentru administrarea unui sistem de management al produselor. Clasa conține un tabel în care se afișează informațiile despre produse, precum și câmpuri pentru introducerea, editarea și ștergerea de produse. Interacțiunea cu baza de date se face prin intermediul clasei DaoProduct, care conține metode pentru inserarea, actualizarea și ștergerea de produse, precum și pentru obținerea de produse din baza de date.

În ceea ce privește componente, clasa conține câteva câmpuri JTextField pentru introducerea datelor de produs, câteva butoane pentru interacțiunea cu baza de date și un tabel JTable pentru afișarea datelor. Componentele sunt organizate într-un JPanel, care este apoi adăugat ca conținut al ferestrei principale JFrame.

Clasa conține, de asemenea, metodele addProduct(), updateProduct(), deleteProduct() și loadData(), care sunt apelate atunci când utilizatorul interacționează cu componentele GUI. Aceste metode apelează metodele corespunzătoare din clasa DaoProduct pentru a actualiza baza de date și apoi încarcă datele actualizate în tabelul JTable prin intermediul clasei DefaultTableModel.

5. Rezultate.

Pentru a testa aplicatia, am testat fiecare scenariu de utilizare posibil, rezultatul obtinut fiind cel asteptat in fiecare caz. De exemplu, la inserarea sau actualizarea oricarui produs sau client, aplicatia duce la bun sfarsit operatiile, dar trebuie reactualizat tabelul (apasand pe butonul de refresh) pentru a se observa modificarile. In cazul stergerii din tabelele Client sau Product, rezultatul se va observa imediat. De asemenea, pentru adaugarea comenzilor, aplicatia duce la bun sfarsit operatiile de inserare a comenzilor si de actualizare a stocului produsului comandat.

6. Concluzii

Prin realizarea celei de a treia teme la disciplina Tehnici de programare am reușit să înțeleg necesitatea tehnicilor de reflexie și modul in care acestea sunt folosite, și de asemenea să mă obișnuiesc să le folosesc pentru a implementa clase generice. Totodată, am învățat să integrez o bază de date în proiect, lucru ce se va dovedi folositor în viitor, și să documentez atât clasele cât și metodele folosind javadoc.

În viitor, aplicația poate fi dezvoltată prin adăugarea mai multor tabele în baza de date dar și prin extinderea operațiilor de insert, update, delete pentru restul tabelelor. De asemenea, se poate extinde prin adăugarea posibilității ca un client să comande mai multe produse, acest lucru fiind posibil prin crearea unui nou tabel ce va reține informațiile intermediare..

7. Bibliografie

1. *https://www.baeldung.com/java-jdbc*
2. *http://www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/*
3. *http://www.tutorialspoint.com/java/util/timer_schedule_period.htm*
4. *https://dzone.com/articles/layers-standard-enterprise*
5. *http://tutorials.jenkov.com/java-reflection/index.html*
6. *https://www.baeldung.com/java-pdf-creation*
7. *https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html*