MINISTRY OF EDUCATION AND RESEARCH

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

FUNDAMENTAL PROGRAMMING TECHNIQUES

ASSIGNMENT 3

ORDER MANAGEMENT

Student: Soponari Beatrice-Valentina

Grupa: 30221

Anul: 2

Cuprins :

1.Obiectivul temei

Obiectivul acestei teme a fost folosirea tehnicii Reflection din Java si lucrul cu clasele generice, pentru a invata un mod nou de a optimiza lucrul cu anumite obiecte care au in mare nevoie de aceleasi metode, iar rescrierea lor ar necesita mult mai mult timp decat folosirea claselor generice care optimizeaza timpul de lucru asupra aplicatiei. Am avut implementat o aplicatie pentru gestionarea unui depozit, care sa permita utilizatorului sa adauge produse noi in depozitul sau, sa managerieze listele de clienti si sa simuleze crearea unei comenzi intre un client si un produs din listele sale disponibile.

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

Problema presupunea crearea unei interfete care va urma sa fie folosita de utilizator, manipularea datelor dintr-o baza de date, adaugarea de noi obiecte, stergerea lor, editarea sau chiar vizualizarea acestora. Scenariile de folosire sunt urmatoarele: utilizatorul porneste aplicatia, unde apare o fereastra din care poate alege datele pe care vrea sa le manipuleze: clienti, produse sau comenzi. Daca acesta alege “client” se va deschide o fereastra care ii va permite sa introduca datele unui nou client, sa editeze datele unui client existent, sa stearga un client al carui id trebuie sa il introduca sau sa vizualizeze un tabel cu toti clientii din baza de date. Daca apasa butonul “product” se va deschide o fereastra care ii va permite sa adauge date pentru a adauga in baza de date un produs nou, sa editeze datele unui produs existent, sa stearga un produs din baza de date sau sa vizualizeze tabelul cu produse existent in MySQL. La alegerea variantei “order” se va deschide o fereastra care ii va permite sa simuleze o comanda realizata de un client din baza de date, care va comanda un produs, pentru ambele obiecte va trebui introdus id-ul de catre utilizator.La realizarea comenzii se va introduce si o cantitate care va fi scazuta din cantitatea actuala a produsului in baza de date, si se va calcula un pret total al comenzii, care, impreuna cu id-ul comenzii, al produsului si al clientului se va afisa intr-o factura in format .pdf.

3.Proiectare

Începem implementarea prin crearea clasei care realizează conexiunea cu baza de date. Aceasta este clasa ConnectionFactory care oferă un getter pentru conexiunea creată, precum și metode pentru închiderea conexiunii, declarației și resultSet. Apoi trecem la clasele care definesc baza de date în aplicație. Acestea sunt clasele care construiesc modelul și fiecare trebuie să fie mapat la unul dintre tabelele din baza de date. Identificăm toate tabelele din baza de date și creăm clasele: Clienți, Produse, Comenzi de comandă având fiecare ca câmpuri atributele din tabelul pe care le mapează.O ultimă clasă particulară este clasa Bill care se mapează pe o tabelă virtuală care este utilizată pentru generarea datelor pentru raportul comenzilor.

Am realizat aceasta aplicatie utilizand abstractizarea claselor si a metodelor, conform carora s-au format clasele:

Client, Product, Order, ClientDAO, ProductDAO, OrderDAO, Bill, ClientBLL, OrderBLL, ProductBLL, JTableClient, JTableProduct, ClientView, ProductView, OrderView, View, Controller si Main

Clasele Client, Product si Order modeleaza obiectele principale cu care lucreaza aplicatia, fiindu-le atribuite exact aceleasi campuri ca si tabelelor din baza de date. Dupa ce am conectat aplicatia la baza de date MyQL prin clasa ConnectionFactory(), am folosit query-uri specifice limbajului MySQL pentru interogarea datelor din baza de date (stergerea, editarea, inserarea de noi date sau vizualizarea acestora). Ultimul lucru pe care trebuie să îl poată face aplicația este să ofere o modalitate de comunicare cu utilizatorul. De obicei, acest lucru se face prin interfața grafică cu utilizatorul. In cadrul interfetei grafica am relizat un frame principal in cadrul careia se poate alege tipul de obiect

asupra caruia dorim sa realizam operatiile CRUD,iar in urma butonului apasat in cadrul acestei interfete se va deschide un nou frame corespunzator alegerii utilizatorului(client, produs sau comanda). Aceste capabilitati se realizeaza in cadrul claselor View, ClientView, ProductView si OrderView. De asemenea pentru realizarea optiunii de afisare a obiectelor din baza de data a fost necesar sa cream clasa AbstractJTAble in care realizam in med generic crearea unui Jtable in care ne sunt afisate field-urile si datele din baza de date corespunzatoare obiectului. Apoi cream clasele ProductTable si ClientTable prin care ne generam tabelele in mod particular. Obiecte ale acestor clase sunt apelate mai apoi in clasa Controller unde se produce de fapt simularea aceste aplicatii.

4.Implementare

Clasa Client contine campurile id, name, address, email, folosite pentru a identifica un client in baza de date. Id-ul fiecarui client este unic. Aceasta prezinta modelul principal pe care trebuie sa il gestionam, a carui date trebuie stocate, modificate sau chiar sterse cand e necesar.

Clasa Product contine atributele id, name, price si quantity, folosite pentru a identifica produsul in baza de date, id-ul fiecarui produs fiind unic. Produsul fiind legat de comanda, la realizarea unei comenzi, printr-o interogare MySQL se selecteaza cantitatea curenta dintr-un produs selectat dupa id, pretul acestuia, iar mai apoi din cantitate se va scadea cantitatea comandata de client, iar pretul este folosit pentru a afisa suma totala pe care un client o va avea de platit la final cand este generata factura, in functie de

cate produse de acel tip a comandat. Un client poate comanda un singur produs o data, nu este implementata capacitatea de a comanda mai multe produse in acelasi timp, dar ar putea fi implementata ca o capabilitate ulterioara pentru dezvoltarea proiectului.
Clasa Order contine atributele id, idClient, idProduct, campul id este de asemenea unic iar celelalte doua campuri sunt folosite pentru a face legatura in cazul unei comenzi intre tabelele Client si Product, de care este nevoie pentru a realiza o comanda. Cand se realizeaza o comanda se va genera o factura care va contine datele acelei comenzi si il va informa pe client de pretul total pe care trebuie sa il plateasca pentru produsul si cantitatea de produs pe care a comandat-o.
Clasa AbstractDAO implemeneaza folosind clase generice si metoda java reflection metodele care insereaza, editeaza sau sterg dintr-un tabel MySQL. Metodele implementate de aceasta clasa sunt createSelectQuery(String field) care creeaza o operatie de SELECT cu sintaxa specifica din MySQL, metoda findAll() care returneaza toate datele dintr-un tabel de tipul specificat printr-o Lista de obiecte, metoda findById(int id) care primeste ca parametru id-ul cautat si returneaza obiectul gasit in tabelul din baza de date cu toate atributele sale. Metoda createObjects(ResultSet resultSet) care primeste ca parametru un obiect de tipul ResultSet pentru a crea mai apoi obiectul folosit de noi in java, adica unul dintre tipurile: client, product sau order. Metoda insert(T t) primeste ca parametru un obiect de tip T si il insereaza in baza de date in tabelul de care apartine, gasim

tabelul folosindu-ne de numele clasei din care provine si inseram in baza de date folosindu-ne de un query de INSERT specific din MySQL. Metoda update(T t) primeste un obiect t care contine noile date care trebuie inlocuite in vechiul obiect de tip T din baza de date. Pentru a implementa operatia in baza de date ne folosim din nou de un query specific unui UPDATE in MySQL. Metoda delete(T t) primeste ca parametru un obiect t care urmeaza sa fie sters din baza de date din tabelul din care apartine, fiind identificat dupa id, folosindu-ne si de aceasta data de un query specific MySQL, si anume acela de DELETE.

Aceasta clasa este mai apoi extinsa de clasele ClientDAO, ProductDAO si OrderDAO care foloses fiecare metodele din aceasta clasa, demonstrand cat de folositor este lucrul cu clase generice, scutindu-ne de scrierea repetitiva a acestor metode in toate cele trei calse. Clasa JTable lucreaza de asemenea cu clase generice pentru a afisa un tabel din baza de date, fie ca e vorba de tabelul Client, Product sau Order, desi in aplicatia noastra vom folosi doar pentru Client si Product. La cererea utilizatorului toate datele prezente la momentul respectiv in tabelul intergogat in baza de date vor fi afisate si vor putea fi vizualizate.

Clasa OrderBLL creeaza metoda generateMessage care genereaza textul care trebuie scris in factura finala. Clasa Bill reprezinta factura care va fi generata automat intr-un fisier de tip .pdf o data cu crearea unei comenzi. La afisarea pdf-ului se vor oserva date precum id-ul comenzii, id-ul clientului, id-ul produsului si pretul total al comenzii in functie de cantitatea de produs comandata si pretul acestuia.

Clasa ConnectionFactory realizeaza conexiunea cu baza de date folosita prin metoda createConnection(). De asemenea aceasta se ocupa si cu inchiderea conexiunii, a unui statement sau a unui resultSet prin metodele close(Connection connection), close(Statement statement), close(ResultSet resultSet). Atributele de care mai avem nevoie desigur pentru ca aceasta clasa sa functioneze sunt URL-ul catre baza de date, numele acesteia, parola si path-ul catre DRIVER.
Clasa Controller este clasa in care se petrece de fapt simularea aplicatiei, in care se apeleaza toate metodele create (insert, upade, delete) la momentul potrivit si in care interfata grafica se leaga de restul codului.
Clasele ClientDAO, ProductDAO, OrderDAO extind clasa AbstractDAO si se folosesc mai departe de metodele din aceasta.
Clasa OrderDAO mai implementeaza si metoda generateMessage() in care se genereaza textul care urmeaza sa fie scris pe factura dupa realizarea unei comenzi. Pentru realizarea acestei utilizari a fost nevoie de folosirea pachetului **com.itextpdf.text.Paragraph** care a fost descarcat de pe internet.
Clasa View este prima fereastra care apare in intampinarea utilizatorului, de unde acesta poate alege tabelul din baza de date pe care doreste sa il modifice/vizualizeze.
Clasa ClientView realizeaza fereastra din interfata in care utilizatorul va manipula operatiunile ce trebuiesc realizate in tabelul Client.

Clasa ProductView realizeaza interfata grafica folosita pentru a modifica/vizualiza datele din tabelul Product din baza de date la nevoia utilizatorului.
Clasa OrderView a fost creata pentru realizarea unei comenzi facuta de un anumit client pentru a cumpara un anumit produs(a caror id se va introduce de catre utilizator).
La final toate ferestrele sunt folosite in metoda Controller, folosind actionListeners pentru a determina cand este necesara folosirea a carei interfete grafice.
Clasa Main creeaza un obiect de tipul Controller si astfel se deruleaza intreaga aplicatie.

5.Rezultate

Dupa rularea acestui program, indiferent ca e vorba de adaugarea unui noi client, a unui nou produs, de modificarea datelor unui client sau ale unui produs, de inserarea unei noi comenzi, se poate observa modificarea in baza de date din MySQL in care vor aparea clientii, produsele, comenzile, se vor sterge clientii sau produsele dorite si se decrementa stocul din tabelul de produse in functie de cantitatea comandata de fiecare client. De asemenea, dupa plasarea unei comenzi se va genera fisierul pdf care va contine datele referitoare la comanda si la clientul care a realizat-o, si si la pretul final al acesteia. Deci programul lucreaza corect si face conexiunile cu baza de date si tabele specifice acesteia, reusind sa insereze, sa editeze, sa stearga si sa afiseze datele din baza de date.

6.Concluzii

Concluzia mea este ca la aceasta tema ne-am familiarizat cu folosirea claselor generice si a tehnicii de reflexie din java, care este foarte utila pentru optimizarea cantitatii de cod scrise si mai ales a timpului necesar scrierii acesteia, deoarece chiar daca poate necesita mai multe cunostinte sa scrii metode folosind clase generice, este mult mai rapid si mai logic decat sa implementezi practic aceleasi metode de doua sau chair de trei ori. Pentru mine tema a fost foarte folositoare si pot spune ca chiar am invatat foarte multe lucruri noi in Java. De asemenea, fisierele pe care le-am avut pe GIT au fost foarte folositoare pentru a ne da seama ce se cere de fapt si pentru a intelege mai bine modul in care trebuie implementate operatiile. Capabilitatea de a comanda mai multe produse deodata pentru un client nu a fost implementata deoarece necesita mai multa munca la tabelele din baza de date, iar timpul nu a fost chiar suficient pentru o asemenea implementare, dar ar putea fi considerata ca o posibila dezvoltare ulterioara a aplicatiei.

7.Bibliografie

https://gitlab.com/utcn_dsrl/pt-layered-architecture

https://gitlab.com/utcn_dsrl/pt-reflection-example

https://docs.oracle.com/javase/tutorial/jdbc/basics/processingsqlstatements.html

https://www.w3schools.com/php/php_mysql_select.asp