DOCUMENTATIE

TEMA *3*

NUME STUDENT: Zaharia Robert
GRUPA: 30225

CUPRINS

1. Obiectivul temei

Obiectivul temei este realizarea unei aplicatii care administreaza un depozit cu produse. Aplicatia se va folosi de o baza de date in care va stoca informatii despre clienti, produse si comenzi .

Obiectivele secundare: realizarea interfatei grafice si utilizarea tehnicilor de reflexie pentru realizarea interogarilor pentru baza de date , realizarea de bonuri pentru fiecare comanda, structurarea codului in 4 pachete pentru o buna administrare a codului: dataAccessLayer, businessLayer, model si prezentare.

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

Use case: administrarea unui client nou sau deja existent

Primary actor: utilizatorul

Scenariul de succes:

1. Utilizatorul poate introduce valori pentru a insera un nou client, sau pentru a modifica sau a sterge datele despre un client deja existent.
2. Utilizatorul apasa pe buton in functie de operatia dorita
3. Aplicatia valideaza datele si ilustreaza un mesaj pentru a informa utilizatorul ca datele clientului au fost procesate.

Use case-uri alternative: administrarea unui produs nou sau deja existent.

Cerinte functionale:

Aplicatia trebuie sa permita utilizatorului sa introduca , modifice sau sa stearga datele despre un client.

Aplicatia trebuie sa permita utilizatorului sa introduca , modifice sau sa stearga datele despre un produs.

Aplicatia trebuie sa permita utilizatorului sa introduca , modifice sau sa stearga datele despre o comanda.

Cerințe nefuncționale:
- Aplicația de simulare ar trebui să fie intuitivă și ușor de utilizat utilizare de către utilizator .

In cadrul problemei vom folosi conexiune dintre Java si MySQL pentru a stoca datele intr-o baza de date. Pentru a realiza aceasta conexiune trebuie sa cream o clasa connector care va primi informatii despre adresa bazei de date, numele bazei de date la care trebuie sa se conecteze cat si numele de utilizator si parola pentru baza de date.

In afara de aceasta clas vom avea nevoie si sa introducem urmatoare dependenta in fisierul Pom.xml:

```
< dependency >
        < groupId > mysql </groupId>
        <artifactId> mysql-connector-java < /artifactId >
        <version> 8.0.23 </version>
    </dependency>
</dependencies>
```

Pentru a accesa informatiile din baza de date vom avea nevoie de cunostine de MySQL pentru a realizare interogarile potrivite pentru operatiile necesare. Interogarile necesare vor fi realizate in Java cu ajutorul tehnicilor de reflexie.

Tehnici de reflexie

Reflecția în Java este de a inspecta și schimba comportamentul unui program în timpul rulării.

Cu ajutorul acestui API de reflecție, puteți inspecta clase, constructori, modificatori, câmpuri, metode și interfețe în timpul rulării. De exemplu, puteți obține numele clasei sau puteți obține detalii despre membrii privați ai clasei.

Cu ajutorul reflectiei vom putea rezolva diferite probleme precum „extragerea campurilor si valorilor corespunzatoare dintr-un obiect dat” .

Pentru a obține acces la informațiile despre clasă, metodă și câmp ale unei instanțe, numim metoda getClass care returnează reprezentarea clasei de rulare a obiectului. Obiectul clasa returnat oferă metode de accesare a informațiilor despre o clasă.

3. Proiectare

Codul aplicatiei este structurat pe 5 pachete:
1) Pachetul Connection contine clasa ConnectionFactory care se ocupa de conectarea aplicatiei la baza de date “ Deposit ”.
2) Pachetul Model contine clasele: Client, Order si Product. Fiecare dintre aceasta clasa isi primeste numele si atributele de la tabelele cu acelasi nume din baza de date.
3) Pachetul Dao contine clasele: AbstractDao, ClientDao, OrderDao, ProductDao. Aceste clase se ocupa cu crearea interogarilor pentru baza de date folosind tehnici de reflexie.
4) Pachetul Bll contine clasele Clientbll, Orderbll, ProductBll si pachetul validators care contine validatorii pentru datele introduse de catre utilizator. Clasele din acest pachet combina

interogarile primate din pachetul Dao si cu verificarea datelor pentru a asigura buna functionare a aplicatiei.
5) Pachetul presentation contine clasele: Main, ClientView, OrderView, ProductView. Aceste clase creaza interfata grafica a aplicatiei si asigura functionalitatea aplicatiei.

4. Implementare

Pachetul Connection

Incepand cu pachetul Connection avem clasa ConnectionFactory. Aceasta clasa contine metodele folosite pentru conectarea la baza de date. Aceasta are ca attribute:

```
private static final Logger LOGGER =
Logger.getLogger(ConnectionFactory.class.getName());
private static final String DRIVER = "com.mysql.cj.jdbc.Driver";
private static final String DBURL = "jdbc:mysql://localhost:3306/deposit";
private static final String USER = "root";
private static final String PASS = "1234";
```

Pachetul Model

Pachetul Model contine clasele: Client, Order si Product. Fiecare dintre aceasta clasa isi primeste numele si atributele de la tabelele cu acelasi nume din baza de date. De asemenea in cadrul fiecarei clase am creat metodele : Setter si Getter pentru fiecare atribut , toString , Constructor.
Clasa Client contine atributele:
private int id ;
private String nume ;
private String adresa ;
private String telefon ;

Clasa Product contine atributele :
private int id ;
private String nume_produs ;
private int pret ;
private int cantitate ;

Clasa Order contine atributele :
private int id ;
private int id_clien t;
private int id_produs ;
private String data ;
private int cantitate ;

Pachetul Dao

Acesta contine clasele: AbstractDao, ClientDao, OrderDao, ProductDao. Aceste clase se ocupa cu crearea interogarilor pentru baza de date folosind tehnici de reflexie. Clasele ClientDao, OrderDao, ProductDao folosesc mostenirea pentru a crea interogarile pentru fiecare tabel.
Clasa AbstratDao contine mai multe metode :
Un constructor prin care obtinem numele clasei la care ne referim. Metodele SelectQuery, InsertQuery, UpdateQuery, DeleteQuery. Acestea realizeaza interogarile dar fara parametrii necesari. Metoda findAll() returneaza datele din tabelul cu numele Clasei < T > apeland metoda de conectare la baza de date. Metoda findByid(int id) primeste ca parametru un id si va returna datele liniei din tabel unde cheia primara este egala cu id. De asemenea aceasta metoda va apela functia de conectare la baza de date. Metoda Createobject sta la baza acestei clase, aceasta permite transformarea rezultatelor primate din interogarea din MySQl in clase potrivite. Aceasta foloseste reflexia pentru a converti in clasa < T >. Apoi avem si metodele Insert, Update si Delete Care primesc ca parametru un obiect al clasei <T> si apoi completeaza interogarea potrivita pentru ca apoi sa se conecteze la baza de date si sa realizeze operatia.

Pachetul Bll

Pachetul Bll contine clasele Clientbll, Orderbll, ProductBll si pachetul validators care contine validatorii pentru datele introduse de catre utilizator. Clasele din acest pachet combina interogarile primate din pachetul Dao si cu verificarea datelor pentru a asigura buna functionare a aplicatiei. In cadrul pachetului avem mai multi validatori:
Name validator verifica daca numele clientului contine doar litere.
Phone validator verifica daca numarul de telefon al clientului incepe cu “07” si are 10 cifre.
Data validate verifica daca data comenzii are formatul potrivit “YYYY-MM-DD”.
Cantitate validate verifica data cantitatea produsului este un numar intreg pozitiv.
Cantitate order verifica data cantitatea comenzii este un numar intreg pozitiv.
Clasa Clientbll primeste un client si verifica daca atributele respecta validatorii . Daca nu afiseaza un mesaj dar daca da, atunci va apela interogarea pentru baza de date.
Clasa Productbll primeste un produs si verifica daca atributele respecta validatorii . Daca nu afiseaza un mesaj dar daca da, atunci va apela interogarea pentru baza de date.
Clasa Orderbll primeste un order si verifica daca atributele respecta validatorii . Daca nu afiseaza un mesaj dar daca da, atunci va apela interogarea pentru baza de date.

Pachetul Presentation
Acest pachet realizeaza interfata grafica a programului. Avem 4 ferestre una pentru fiecare tabel din baza de date si una principala . Fereasta create de ClientView permite sa vedem toti clientii din baza de date si ne permite sa-I cautam dupa Id. De asemenea ne permite sa introducem clienti noi si sa modificam datele celor deja existenti.
Fereasta create de ProductView permite sa vedem toti clientii din baza de date si ne permite sa-I cautam dupa Id. De asemenea ne permite sa introducem clienti noi si sa modificam datele celor deja existenti.

Fereasta create de OrderView permite sa vedem toti clientii din baza de date si ne permite sa-I cautam dupa Id. De asemenea ne permite sa introducem clienti noi si sa modificam datele celor deja existenti. In cadrul clasei OrderView am rezolvat si obiectivul secundar al problemei: crearea de bonuri fiscale pentru comenzi in format txt sau pdf.

5. Rezultate

Rezultatul codului scris este o aplicatie functionala prin intermediul careia se pot modifica informatiile din baza de date a depozitului . Modificarile valorilor se pot observa atat in cadrul aplicatiei cat si in cadrul bazei de date .

6. Concluzii

Realizarea aplicatiei impus diverse probleme datorita conceptelor noi de reflexie cat si necesitatea cunostintelor de MySQL . Rezolvarea problemei a permis intelegerea acestor concepte cat si aprofundarea notiunilor de interfata grafica in java.

7. Bibliografie

https://www.geeksforgeeks.org/reflection-in-java/