MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

GESTIONAREA COMENZILOR

Alin Matean

Facultatea de Automatica si Calculatoare

Grupa 30228

Cuprins

1.Obiectivul temei.....3

2.Analiza problemei, modelare, scenarii, cazuri de utilizare.....3

3.Proiectare.....5

4.Implementare.....8

5.Rezultate.....10

6.Concluzii.....11

7.Bibliografie.....11

Obiectivul temei

Obiectivul acestei teme este proiectarea și implementarea unei aplicații de de gestionare a unor comenzi. Comenzile și toate datele problemei vor fi stocate într-o bază de date. Programul trebuie să lucreze cu baze de date realționare și trebuie să-i permită utilizatorului să insereze, editeze sau să șteargă clienți și produse, dar și să poată plasa o comandă. Mai mult, fiecare comandă trebuie să poată fi văzută sub forma unei facturi.

Analiza problemei, modelare, scenarii, cazuri de utilizare

Analiza problemei

Problema care se impune este cea de a gestiona comenzile și de a se crea facturi pentru fiecare comandă. Acest lucru se realizează cu ajutorul unei baze de date unde se păstrează toate datele introduse de utilizator.Toate datele și modificările se vor actualiza în timp real.

Scenarii

Scenariul de utilizare a aplicației presupune interacțiunea cu interfața grafică creată. Astfel utilizatorul are trei panouri principale, fiecare corespunzător unui tabel din baza de date: client, produs și comenzi. Utilizatorul poate introduce clienți noi, poate să șteargă un client sau poat să facă update unui client, de asemenea poate să introducă sau să șteargă produse și să facp update uni produs. În cazul comenzilor poate doar să introducă comenzi noi. Toate aceste date pot fi văzute pe fiecare panou corespunzător tabelului din baza de date create.

Scenariu de utilizare:

- se pornește aplicația
- inițial se deschide panoul de la Client, unde avem câmpurie pentru a adăuga, a edita sau a șterge un client și mai jos avem tabelul cu toți clienții care se va actualiza la fiecare operație făcuta

- același lucru ca și la client, avem și în al doilea panou, la produs, putem adăuga, șterge și edita produse, iar în partea de jos este tabelul cu toate produsele din baza de date
- al treilea panou este cel cu comenzi, aici avem câmpurile pentru a efectua o nouă comandă, comenzile se pot observa în partea de jos în tabelul corespunzător
- mai mult, la fiecare comandă nou introdusă se va genera o factură de tip fișier.txt cu numele clientului, numele prdusului, cantitatea și prețul final
- tot aici avem și cazurile în care nu se pot efectua anumite operații asupa tabelelor din baza de date, de exemplu: se vrea introducerea unei comenzi cu o cantitatsea care nu este disponibilă în stoc, în acest caz se va afișa un mesaj de eroare. Alte cazuri care ar putea genera excepții sunt cele asupa tabelelor de genul chei primare / chei străine sau de exemplu introducerea unui nume prea lung.

Cazuri de utilizare – diagrama Use-case

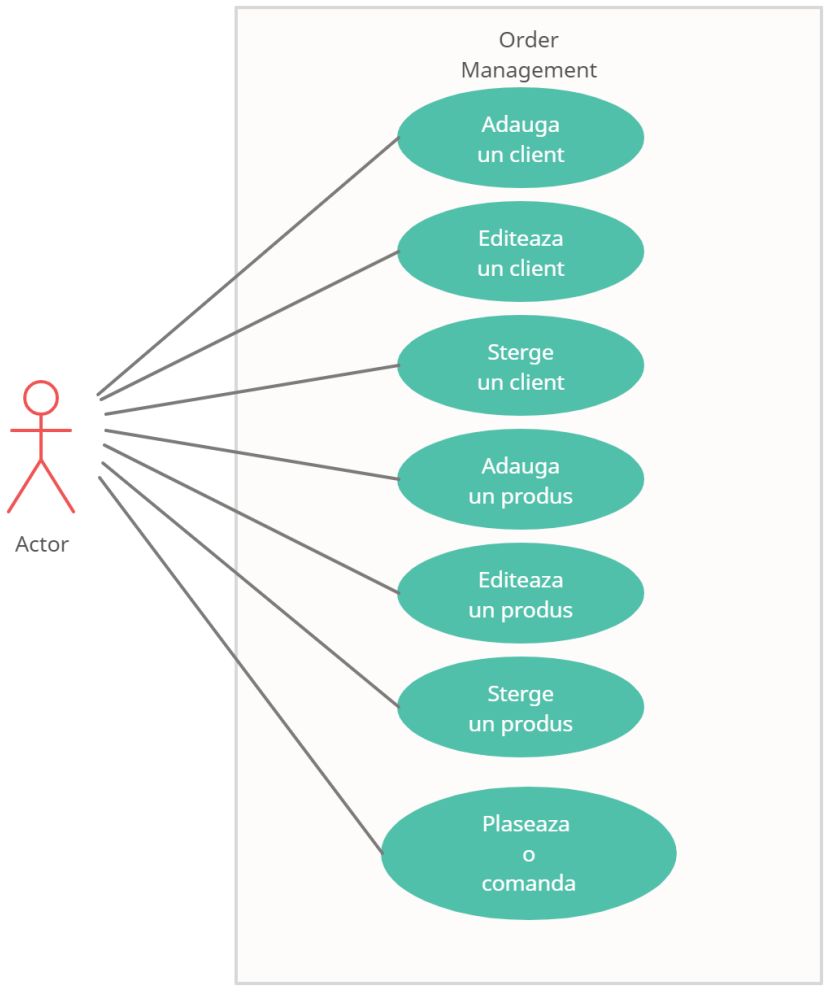

Figura 1. Diagrama use-case

Utilizatorul va efectua practice o singura operație și anume va porni simularea după ce introduce datele de intrare și va crea un fișier unde se vor stoca datele simulării.

Proiectare

Decizii de proiectare

Aplicația a fost proectată folosind baze de date cu ajutorul MySql Workbench. Este un instrument vizual pentru baze de date, acesta permite modelarea datelor, dezvoltarea SQL si instrumente de administrare complete pentru configurarea serverului, administrrea utlizatorului, backup si altele. Este disponibil pe diferite sisteme de operare. Cu ajutorul acestui instrument am creat baza de date unde am făcut tabelele:

- client(idClient (INT), name (VARCHAR(45)), address(VARCHAR(45))) – idClient – primary key
- product(idProduct(INT), name(VARCHAR(45)), quantity(INT), price(DOUBLE)) – idProduct – primary key
- orders(idOrder (INT), idClient(INT), idProduc(INT), quantity(INT), total(DOUBLE)) – idOrder – primaryKey, idClient – foreign key către idClient din client și idProduct foreign key către idProduct din product

În cadrul pachetului “src” unde se află tot codul necesar funcționării și testării aplcației se găsește directorul “main”. În directorul main găsim folder-ul “java” care conține pachetul prinipal și anume “tuc.tp.tema3”, în care se află de fapt tot conținutul pentru funcționarea corectă a calculatorului. Acest pachet conține la rândul lui alte 7 pachete în total. Celelalte pachete sunt:

- connection – unde se realizează conexiunea la baza de date
- dao – care se ocupă de interogările asupra baze de date, conține AbstractDAO, ClientDAO, ProductDAO și OrderDAO
- bll – aici se încapsulează logica aplicației, există câte o clasă pentru fiecare table: ClientBLL, ProductBLL, OrderBLL. Aici se mai află și pachetul Validators care conține o interfață și alte 3 clase care implemetează interfața respectvă
- model – conține câte o clasă de model pentru fiecare tabel din baza de date, se face maparea acestor tabele
- presentation – conține clasele necesare pentru interfața grafică GUI: view si controller
- start – pachetul în care se află o clasă pentru a extrage anumite proprietăți și pentru a crea tabelul ce va fi afișat și clasa Start care pornește efectiv aplicația

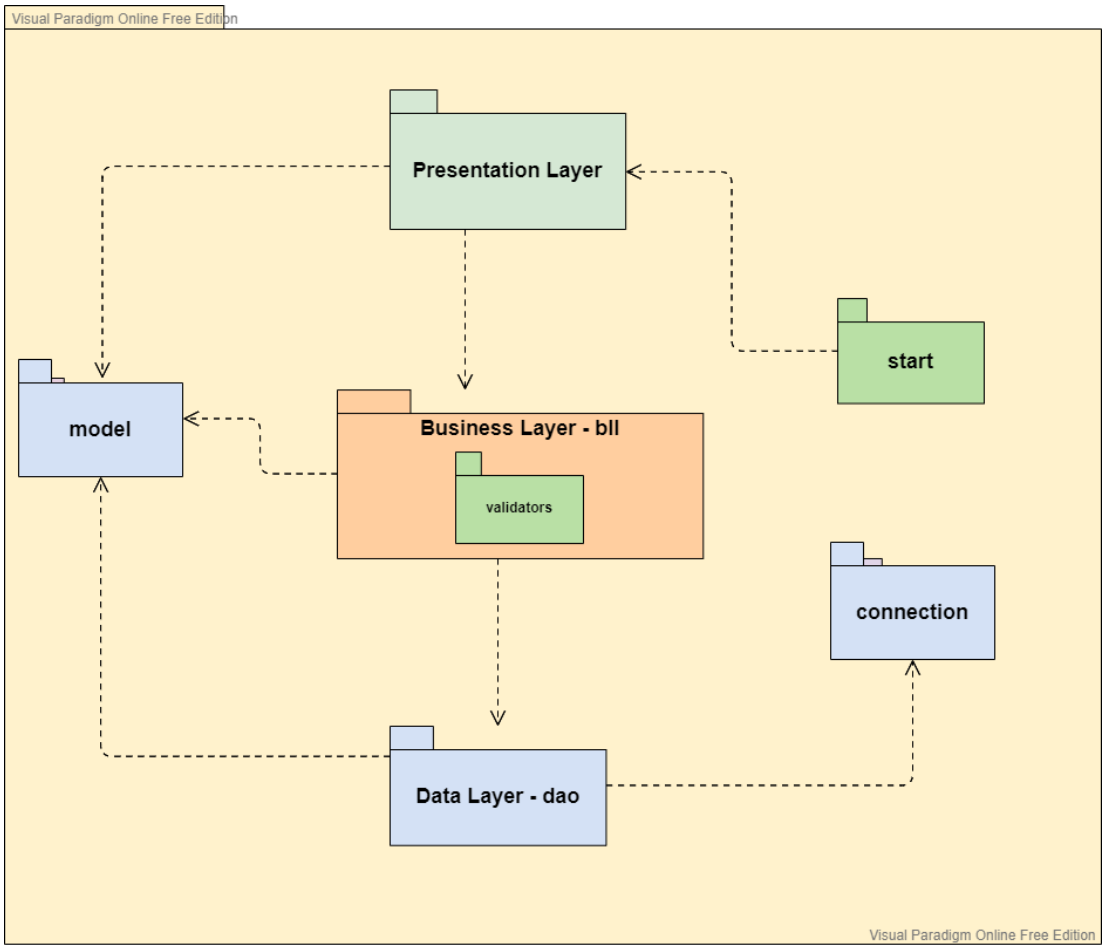

Figura 2. Diagrama de pachete [2]

Diagrama UML de clase

Clasele proiectate:

În pachetul tuc.tp.tema3 se găsesc:

- ClientNameAddressValidator – pachetul validators din pachetul bll
- OrderQuantity - pachetul validators din pachetul bll
- ProductPriceValidator - pachetul validators din pachetul bll
- Validator - pachetul validators din pachetul bll
- ClientBLL – pachetul bll
- OrderBLL - pachetul bll
- ProductBLL - pachetul bll
- ConnectinFactory – connection
- Client – pachetul dao
- Orders - pachetul dao
- Product - pachetul dao
- Controller – pachetul presentation

- View - pachetul presentation
- App – pachetul start
- ReflectionExample – pachetul start

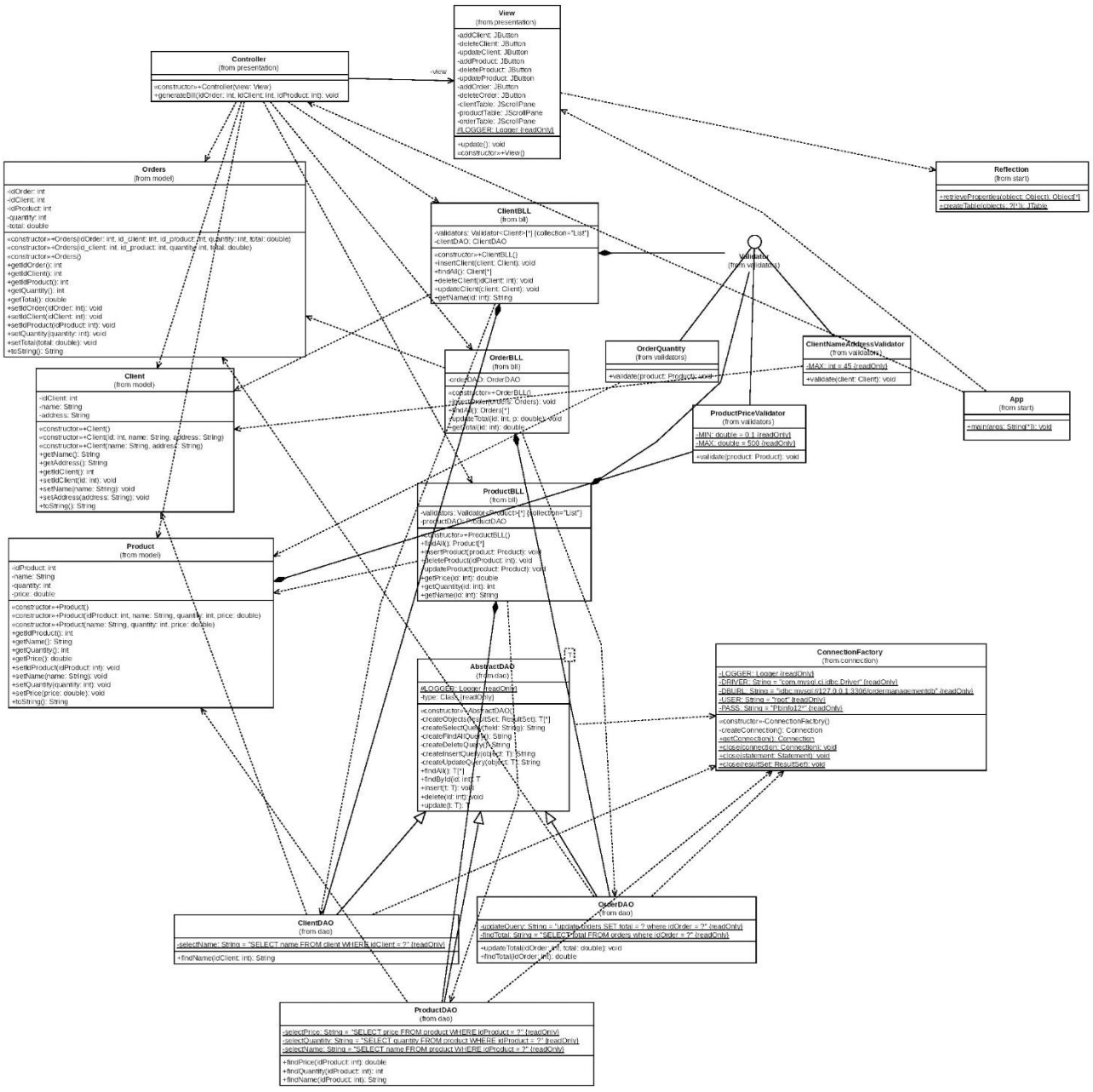

Figura 3. Diagrama UML de clase

Interfața grafică cu utilizatorul:

Utilizatorul are la dispoziție 3 panouri fiecare corespunzător unui tabel din baza de date. Pe fiecare panou utilizatorul poate să adauge, editeze sau să șteargă un element din tabel, cu excepția tabelului orders unde se poate doar adăuga o comandă nouă. De asemenea, pe fiecare panou se poate vedea tabelul curent din baza de date.

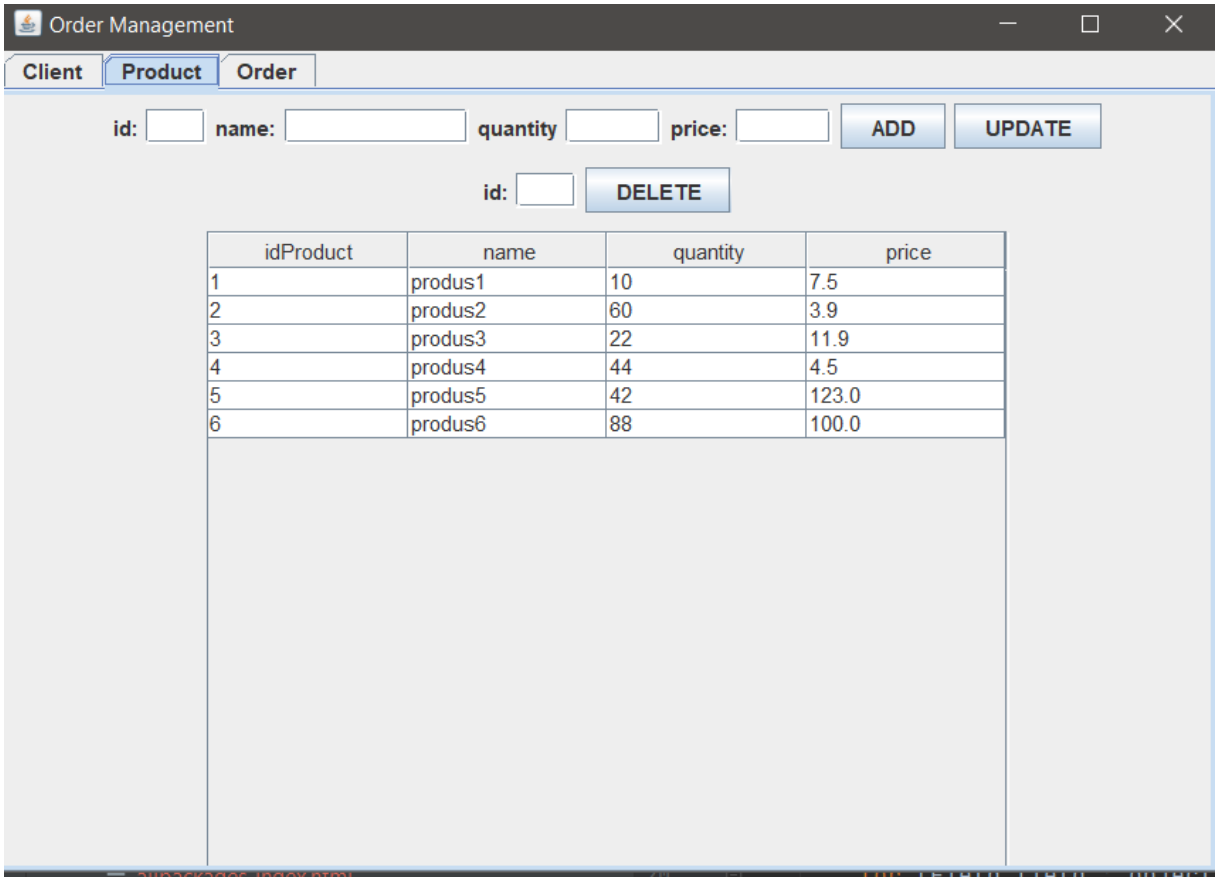

Figura 4. Interfața grafică cu utilizatorul

Implementare

Clasele

Pachetul bll:

- Pachetul validators care conține o interfață și trei clase care implementează interfața necesare validării unei noi intrări în baza de date

- ClientBLL – încapsulează operațiile care se fac asupra clienților, aici avem o lista de validatoare care s verifică în constructorul clasei. Conține metodele insertClient, deleteClient, findAll, updateClient, getName, fiecare dintre acestea apelând metodele din AbstractDAO
- ProducttBLL – încapsulează operațiile care se fac asupra produselor, aici avem o lista de validatoare care s verifică în constructorul clasei. Conține metodele insertProduct, deleteProduct, findAll, updateProduct, getName, getPrice, getQuantity, fiecare dintre acestea apelând metodele din AbstractDAO sau ProductDAO
- OrdersBLL – încapseulează operațiile care se fac asupra tabelei Orders din baza de date. Astfel putem adăuga o comandă noua, putem seta totalul în urma calculării și putem găsi toate intrările în baza de date

Pachetul connection:

- ConnectionFactory – este clasa care realizează conexiunea la baza de date. Conține metodele createConnection () face conexiunea cu baza de date , getConnection() returneaza conexiunea close (Connection), close (Statement), close (ResultSet) inched conexiunea si sunt apelate dupa fiecare interogare a bazei de date

Pachetul dao:

- AbstractDAO – aici sunt implementate principalele operații care se fac asupra tabelelor: insert, delete, update, select, utilizând tehnica Reflection celelalte clase moștenesc aceste metode și nu mai sunt implemetate altele noi. Această clasă este o clasă generică, obiectul urmând sa fie de tipul Client, Product sau Orders. Metoda createObjects este cea care creeaza o instanță de tipul corespunzător dintr-un ResultSet. Mai sunt metodele *query care creeaza interogările de care vom avea nevoie mai departe și care vor fi apelate pentru a crea un statement care va fi executat. Metodele de insert, update, delete și findAll care execută interogările necesare. Fiecare metodă creează un statemet care va executa interogarea potrivită
- ClientDAO, ProductDAO, OrderDAO moștenesc AbstractDAO cu genericul Client, Product și Order. Am mai implementat doar anumite interogări specifice pentru fiecare tabel, în rest le moștenesc pe cele din AbstractDAO.

Pachetul model:

- Client – este clasa care face maparea și conține toate câmpurile tabelei Client din MySQL cu tipul de date corespunzător. În plus, mai conține metode de get și set pentru fiecare câmp.
- Orders – este clasa care face maparea și conține toate câmpurile tabelei Orders din MySQL cu tipul de date corespunzător. În plus, mai conține metode de get și set pentru fiecare câmp.
- Product – este clasa care face maparea și conține toate câmpurile tabelei Product din MySQL cu tipul de date corespunzător. În plus, mai conține metode de get și set pentru fiecare câmp.

Pachetul presenation:

- Controller – aici sunt implementatea funcționalitățile butoanelor de pe interfața grafică. Cu ajutorul unui ActionListener pentru fiecare buton îi dată câte o funcționalitate. Tot aici avem metoda de generare a unei facturi, după fiecare comandă nou adăugată. Această comandă se creează automat după ce se adaugă o comandă nouă pe numele clientului, cu produsul ales, în ce cantitate și la ce preț total. Comanda este reprezentată sub forma unui fișier text.

- View – aici este creată partea vizuală a aplicației, cea cu care interacționează utilizatorul. În constructor sunt create cele trei panouri, fiecare cu tabelul și cu butoanele corespuunzătoare. Cu ajutorul metodei update (), după fiecare modificare a tabelei, de exemplu ștergere, update sau inserare se face update și în interfață, nu doar in baza de date.

Pachetul start:

- Reflection – în această clasă se creează tabelul în urma extragerii datelor din baza de date. Se pun coloanele corespunzătoare și datele din fiecare tabel în parte
- App – este clasa care pornește efectiv aplicația. Se creează un view și un controller pe baza view-ului creat

```
public static JTable createTable(List<? extends Object> objects)
throws IllegalAccessException{
    ArrayList<Object> rowData = new ArrayList<>();
    ArrayList<String> columns = new ArrayList<>();

    for(Field field: objects.get(0).getClass().getDeclaredFields()){
        field.setAccessible(true);
        columns.add(field.getName());
    }

    String[] columnNames = new String[columns.size()];
    columnNames = columns.toArray(columnNames);
    DefaultTableModel tableModel = new DefaultTableModel(columnNames,
0);

    for(Object obj: objects){
        rowData.clear();
        for(Field field: obj.getClass().getDeclaredFields()){
            field.setAccessible(true);
            rowData.add(field.get(obj));
        }
        Object[] dataObjects = new Object[rowData.size()];
        dataObjects = rowData.toArray(dataObjects);
        tableModel.addRow(dataObjects);
    }
    return new JTable(tableModel);
}
```

Rezultate

Pentru testare am inserat, șters și făcut update pe fiecare tabel pentru a vedea rezultatele și in baza de date și in interfața grafică. Aplicația funcționează corespunzător. Mai mult, după o inserare în tabela Orders se va genera și o factură cu toate datele necesare de tipul:

```
Comanda: 11
Nume client: client14
Produs: produs3
```

```
Cantitate: 2
Total: 23.8
```

Concluzii

Dezvoltări ulterioare:

- S-ar putea dezvolta interfața grafică cu utilizatorul
- Implementarea unui nou tabel în baza de date care să țină stocul
- Adăugarea mai multor câmpuri în baza de date
- Crerea unui mod pentru a căuta în baza de date un anumit client sau produs, din interfață
- Posibilitatea de a adăuga mai multe produse pe aceeași factură
- Crearea unui mod prin a recompense clienții fideli, de exemplu la un anumit număr de comenzi sau dacă se depășește o anumită sumă, clientul să beneficieze de o reducere.

Concluzii:

Cu ajutorul acestui proiect am învățat să lucrez cu bazele de date într-un proiect java. Am învățat să utilizez MYSql WorkBench. Consider că este chiar util având în vedere că în lumea reală majoritatea aplicaților interacționează cu o bază de date. Am început să comentez codul pentru a se putea genera un document Javadoc după terminare pentru o bună documentație a proiectului.

Bibliografie

[1] https://users.utcluj.ro/~igiosan/teaching_poo.html

[2] https://online.visual-paradigm.com/

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