MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ȘI SPORTULUI
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Order Management

Gherghel Daniel-Andrei

Grupa 30224

Îndrumător de laborator: Claudia Pop

Cuprins

1. Cerințe funcționale

Să se implementeze o aplicație care are scopul de a procesa comenzile clienților dintr-un depozit. Se va folosi o bază de date relațională pentru a reține clienții, produsele și comenzile.

2. Obiective

2.1. Obiectiv principal

Obiectivul principal al proiectului este de a crea o aplicație prin intermediul căreia să se faciliteze accesul la o bază de date relațională.

2.2. Obiective secundare

- Gruparea claselor în pachete
- Folosirea unor tehnici noi de programare (reflexie)
- Alegerea structurilor de date
- Manipularea bazei de date prin intermediul aplicației

3. Analiza problemei

Sistemele de prelucrare a tranzacțiilor (SPT) procesează și înregistrează datele rezultate din operațiile și activitățile firmei. Exemple tipice sunt sistemele informatice care prelucrează vînzările, cumpărările și mișcările de stocuri. Rezultatele acestor prelucrări sunt utilizate pentru a actualiza bazele de date privind clienții, stocurile și alte baze de date ale organizației. Acestea oferă apoi resurse de date ce pot fi prelucrate și utilizate de sistemele informaționale pentru management.

Aceste sisteme produc, de asemenea, o varietate de produse informaționale destinate uzului intern sau extern, precum facturi, chitanțe, state de plată, comenzi, ordine de plată, situații financiare. Sistemele de prelucrare a tranzacțiilor procesează tranzacțiile în două moduri principale, și anume:

- prelucrare în loturi – datele sunt acumulate de-a lungul unei perioade de timp și prelucrate periodic;
- prelucrare în timp real (sau online) – datele sunt prelucrate imediat după efectuarea unei tranzacții.

De exemplu, sistemele punctelor de vînzare (POS) ale magazinelor cu amănuntul pot utiliza terminale electronice la casele de marcat, pentru a culege și transmite electronic datele privind vînzările, prin intermediul legăturilor de

telecomunicații, către centrele informatice regionale în vederea prelucrării imediate (în timp real) sau pe timpul nopții (în loturi).

4. Proiectare

4.1. Diagrama de clase UML

4.2. Clase și algoritmi folosiți

Pachetul **Business** conține clasele care se ocupă de partea de verificare a datelor din aplicației. În pachet se găsesc clasele: *ClientCheck*, *Product Check si Order Check*. In toate cele trei clase se gasesc metodele de verificare pentru operatiile de adaugare, update si delete.

```
public void update(Client c)
	{
		if(c.getIdClient()!=0)
		{
			rep.update(c);
		}
	}

	public void delete(Client c)
	{
		if(c.getIdClient()!=0)
		{
			rep.delete(c);
		}

	}
```

`Pachetul` **`Data`** contine clasele: RepClient, RepProduct, RepOrder si Conection.

In primele 3 clase sund definite si implementate operatiile de inserare, update si delete, metode care returneaza un string care contine instructiunile executabile in MySQL.

Clasa Conection realizeaza conexiunea la baza de date.

```
		public String insert(Client o)
	{
		String s="Insert into Clients(idClient,nameClient,adresa) values ('" +
o.getIdClient() + "','" + o.getNameClient() + "','" + o.getAdresa() + "')";
```

```
        return s;

}
```

Pachetul **Model** contine 3 clase: Client, Product si Order. Cele 3 clase contin informatiile din tabele din baza de date si implementeaza metodele de get si set.

```
public class Client {

        private int idClient;
        private String nameClient;
        private String adresa;


public class Order {

        private int idClient;
        private int idProduct;
        private int cantitate;
        private int idOrder;

public class Product {

        private int idProduct;
        private int stoc;
        private String nameProduct;
        private double pret;
```

Pachetul PT2019 contine clasa App care contine functia principala de unde se deschide conexiunea si interfata cu utilizatorul, care este reprezentata printr-o fereastra de comada de unde se pot deschide fereastra dorita prin selectarea unei obtiuni.

Pachetul **Presentation** contine 7 clase unde se modeleaza interfata si transferul de date dintre aplicatie si baza de date.

Clasa CLIENT are rolul de a realiza Frame-ul unde sunt disponibile operatiile pe tabelul Client. Frame-ul are 3 textField-uri unde se introduce datele legate de client si 3 butoane de unde se pot executa cele 3 operatii.

Clasa contine, de asemenea, si un JTable care este populat cu clientii existenti in baza de date. Popularea JTable-ului se face cu ajutorul functiilor:

```
private static void getTableData(DefaultTableModel dtm) {
          Object[] dataForTable = new Object[3];
```

```
            Extract e=new Extract();
            ArrayList<Client> c=e.getClientData();
        for (int i = 0; i < c.size(); i++) {


                dataForTable[0] = c.get(i).getIdClient();
                dataForTable[1] = c.get(i).getNameClient();
                dataForTable[2] = c.get(i).getAdresa();
                dtm.addRow(dataForTable);

        }


    }

    public void getColumns() {

    table=new JTable();
String[] columns = {"idClient","nameClient","Adresa"};

DefaultTableModel defm=new DefaultTableModel(columns,0);
getTableData(defm);
//return columns;


    contentPane.add(table);
    JScrollPane scrollPane = new JScrollPane();
    scrollPane.setBounds(38, 162, 345, 89);
    this.getContentPane().add(scrollPane);
    table = new JTable(defm);
    scrollPane.setViewportView(table);
```

Clasa COMAND creeaza Frame-ul principal care contine 3 butoane de unde se selecteaza tabelul asupra caruia se doreste sa se efctueze operatii.

Clasa PRODUCT la fel ca si clasa CLIENT, are rolul de a realiza Frame-ul unde sunt disponibile operatiile pe tabelul Product. Frame-ul are tot 3 textField-uri unde si 3 butoane. Populatea JTable-ului se face cu ajutorul acelorasi functii.

Clasa ORDER are exact aceeasi structura ca si cele anterioare cu exceptia faptului ca are 4 textField-uri.

Clasa Execute are rolul de a returna un ArrayList care contine informatiile extrase din baza de date din tabelul Client.

```
static ArrayList<Client> getClientData()
```

```
{
       ArrayList<Client> c=new ArrayList<Client>();
       String s="select * from Clients";
       con=cn.createConection();
       ResultSet rs=null;
       Statement statement=null;
    try {
        //con.createConnection();
        statement= (Statement) con.createStatement();

        rs = statement.executeQuery(s);
        Client client;
        while (rs.next()) {
             String nume,adresa;
             int id;

            id=rs.getInt("idClient");
            nume=rs.getString("nameClient");
            adresa=rs.getString("adresa");
            client=new Client(id,nume,adresa);
            System.out.println(client);

            c.add(client);
        }


    } catch (Exception e) {
        System.out.println(e);
    }
    finally
    {
       try {
                    con.close();
             } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
             }
    }
    return c;


             }
```

Clasa ExecuteProduct extrage informatiile din baza de date din tabelul Product. Am deschis un statement care executa un query, mai precis intructiunea echivalenta din MySQL “SELECT * (ALL)”. Toate informatiile vor fi returnate sub forma unu array de obiecte, produse.

```
ArrayList<Product> p=new ArrayList<Product>();
       String s="select * from Product";
       con=cn.createConection();
       ResultSet rs = null;
```

```
   Statement statement = null;
try {
    //con.createConnection();
    statement= (Statement) con.createStatement();

    rs = statement.executeQuery(s);
    Product product;
    while (rs.next()) {
         String nume;
         int id,stoc;
         double pret;

        id = rs.getInt("idProduct");
        nume = rs.getString("nameProduct");
        prêt = rs.getDouble("pret");
        stoc = rs.getInt("stoc");

        product= new Product(id,nume,pret,stoc);


        p.add(product);

     }
```

Clasa ExecuteOrder se ocupa de partea de extragere de informatii din din tabelul Order. Merge pe acelasi principiu, se creaza un statement care se executa printr-un query pe un string, valoare care este pusa intr-un ResultSet. Cu un while se parcurge intreg ResultSet-ul si se extrag valorile, care mai apoi sunt adaugae in array-ul care va fi returnat.

```
   ArrayList < Order > o=new ArrayList < Order >();
   String s= "select * from Orders";
   con=cn.createConection();
   ResultSet rs = null;
   Statement statement = null;
try {
    //con.createConnection();
    Statement = (Statement) con.createStatement();

    rs = statement.executeQuery(s);
    Order order;
    while (rs.next()) {

         int ido,idc,idp,cant;


        idp = rs.getInt("idProduct");
        ido = rs.getInt("idOrder");
        idc = rs.getInt("idClient");
        cant = rs.getInt("cantitate");
```

```
            order = new Order(ido,idc,idp,cant);

            o.add(order);

        }
```

5. Concluzii și dezvoltări ulterioare

Aplicația implementată este ușor de folosit și realizeză toate operațiile necesare pentru lucrul cu o bază de date: insert, update, delete. Aplicația poate fi dezvoltată adăugând mai multe tabele în baza de date, precum și adăugarea mai multor funcții pentru utilizator. Este ideala pentru utilizare in cadrul unui magazine online sau si ca suport chiar si pentru un magazine fizic.

Bibliografie

https://ro.wikipedia.org/wiki/Sistem_de_prelucrare_a_tranzac%C8%9Biilor

https://examples.javacodegeeks.com/core-java/sql/retrieve-data-example/?fbclid=IwAR3W0c1zN0XrzaYlAThw7w9OrnmztKhT_LTJNxj-1fW2vq3R3Z8hLQBUeKQ

https://stackoverflow.com/questions/13123811/populating-jtable-from-array?fbclid=IwAR3-7evctclc5PWgFU_n-AVBPQICP6jViKH9umjp-FCK66jKC2vEy0p_2qE

https://stackoverflow.com/questions/45263672/how-to-populate-jtable-with-arraylist-values?fbclid=IwAR3W0c1zN0XrzaYlAThw7w9OrnmztKhT_LTJNxj-1fW2vq3R3Z8hLQBUeKQ

https://www.logicbig.com/tutorials/java-swing/generate-jtable-model-with-annotation.html?fbclid=IwAR1kvICXRI3cemR0SzJ7KfXarePdHorYNVz8wMToEcgpHhqE8mTFoxJedMY

http://theopentutorials.com/tutorials/java/jdbc/jdbc-mysql-create-database-example/

https://dzone.com/articles/layers-standard-enterprise

http://baseprogramming.com/blog1/2017/08/24/automating-jdbc-crud-operations-with-reflection/?fbclid=IwAR0Y11LSJ3tgxP0DL3d_LfKgUCwCgCFWs-8auOok9yNNd4-jIoHE8OeuOZw