

Tehnici de programare

Tema 3



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Alexandru Madalina-Elena

Grupa: 30228

Anul 2

Cuprins:

1. Obiectivul temei
2. Analiza problemei, modelare, scenarii, cazuri de utilizare
3. Proiectare
4. Implementare
5. Rezultate
6. Concluzii
7. Bibliografie

1. Obiectivul temei

Cerinta: Consider an application *OrderManagement* for processing customer orders for a warehouse.

Relational databases are used to store the products, the clients and the orders. Furthermore, the application uses (minimally) the following classes:

1. *Model classes* - represent the data models of the application
2. *Business Logic classes* - contain the application logic
3. *Presentation classes* – classes that contain the graphical user interface
4. *Data access classes* - classes that contain the access to the database

Other classes and packages can be added to implement the full functionality of the application.

a. Analyze the application domain, determine the structure and behavior of its classes and draw an extended UML class diagram.

b. Implement the application classes. Use javadoc for documenting classes.

c. Use reflection techniques to create a method `createTable` that receives a list of objects and generates the header of the table by extracting through reflection the object properties and then populates the table with the values of the elements from the list: `JTable createTable(List<Object> objects);`

d. Implement a system of utility programs for reporting such as: under-stock, totals, filters, etc.

Abordarea problemei: Am ales sa fac proiectez o baza de date cu 3 tabele, *Order*, *Client* si *Product*, si sa aleg pentru fiecare coloane care au relevanta pentru tabelul current: *Order*(`orderid`, `final_price`, `quantity`, `client_id` si `product_id`), *Product*(`product_id`, `price`, `product_name` si `stock`) si *Client*(`id`, `name`, `address`, `email`, `phone_number`). Aceste tabele sunt manipulate folosind metoda in proiectul Java prin intermediul a trei interfete grafice. Fiecare dintre acestea este usor de

utilizat, fiind proiectate clar pentru orice utilizator. Datorita conexiunii dintre baza de date si metodele proiectate, modificarile dorite de utilizator sunt aplicate direct tabelelor originale.

In general, to process any SQL statement with JDBC, you follow these steps:

- Establishing a connection.
- Create a statement.
- Execute the query.
- Process the ResultSet object.
- Close the connection.

First, establish a connection with the data source you want to use. A data source can be a DBMS, a legacy file system, or some other source of data with a corresponding JDBC driver. This connection is represented by a Connection object. See Establishing a Connection for more information. A Statement is an interface that represents a SQL statement. You execute Statement objects, and they generate ResultSet objects, which is a table of data representing a database result set. You need a Connection object to create a Statement object.

For example, CoffeesTables.viewTable creates a Statement object with the following code: `stmt = con.createStatement();`

There are three different kinds of statements:

- Statement: Used to implement simple SQL statements with no parameters.
- PreparedStatement: (Extends Statement.) Used for precompiling SQL statements that might contain input parameters. See Using Prepared Statements for more information.
- CallableStatement: (Extends PreparedStatement.) Used to execute stored procedures that may contain both input and output parameters. See Stored Procedures for more information.

2. Analiza

Scenariu de utilizare:

- Se deschid initial 3 ferestre, una pentru tabelul Order, una pentru tabelul Client si una pentru tablul Product
- In fereastra pentru clients avem mai multe optiuni
- Se apasa Find dupa introducerea unui id pentru a se aplica un filter si pentru a se gasi clientul cu id-ul introdus
- Se apasa List all pentru a vizualiza din nou toate liniile tabelului
- Dupa completarea celor 5 campuri, se apasa Add pentru a adauga un client nou
- Dupa completarea celor 5 campuri, se apasa Update pentru schimbarea informatiilor unui client(id-ul completat trebuie neaparat sa existe in table)
- Se apasa Remove pentru stergerea unui client la alegere, identificat dupa ID
- In fereastra pentru products avem mai multe optiuni
- Se apasa Find dupa introducerea unui id pentru a se aplica un filter si pentru a se gasi produsul cu id-ul introdus
- Se apasa List all pentru a vizualiza din nou toate liniile tabelului
- Dupa completarea celor 4 campuri, se apasa Add pentru a adauga un produs nou
- Dupa completarea celor 4 campuri, se apasa Update pentru schimbarea informatiilor unui produs(id-ul completat trebuie neaparat sa existe in table)
- Se apasa Remove pentru stergerea unui produs la alegere, identificat dupa ID
- In fereastra pentru orders avem mai multe optiuni
- Se apasa Find dupa introducerea unui id pentru a se aplica un filter si pentru a se gasi comanda cu id-ul introdus
- Se apasa List all pentru a vizualiza din nou toate liniile tabelului
- Dupa completarea celor 4 campuri, se apasa Add pentru a adauga o comanda nou

- După completarea celor 4 câmpuri, se apasă Update pentru schimbarea informațiilor unui client(id-ul completat trebuie neapărat să existe în table)
- Se are în vedere faptul că ID-ul produsului ales și ID-ul clientului ales trebuie să existe în tabelele pentru produs și client
- Se apasă Remove pentru ștergerea unei comenzi la alegere, identificate după ID
- Se alege un ID și se apasă finalize order pentru a se genera un bon pentru comanda respectivă în fișierul «bill.txt»
- Se pot adăuga oricâți clienți, oricâte produse și oricâte comenzi
- Se pot repeta toți pașii de câte ori se dorește

3. Proiectare

Descriere pachete:

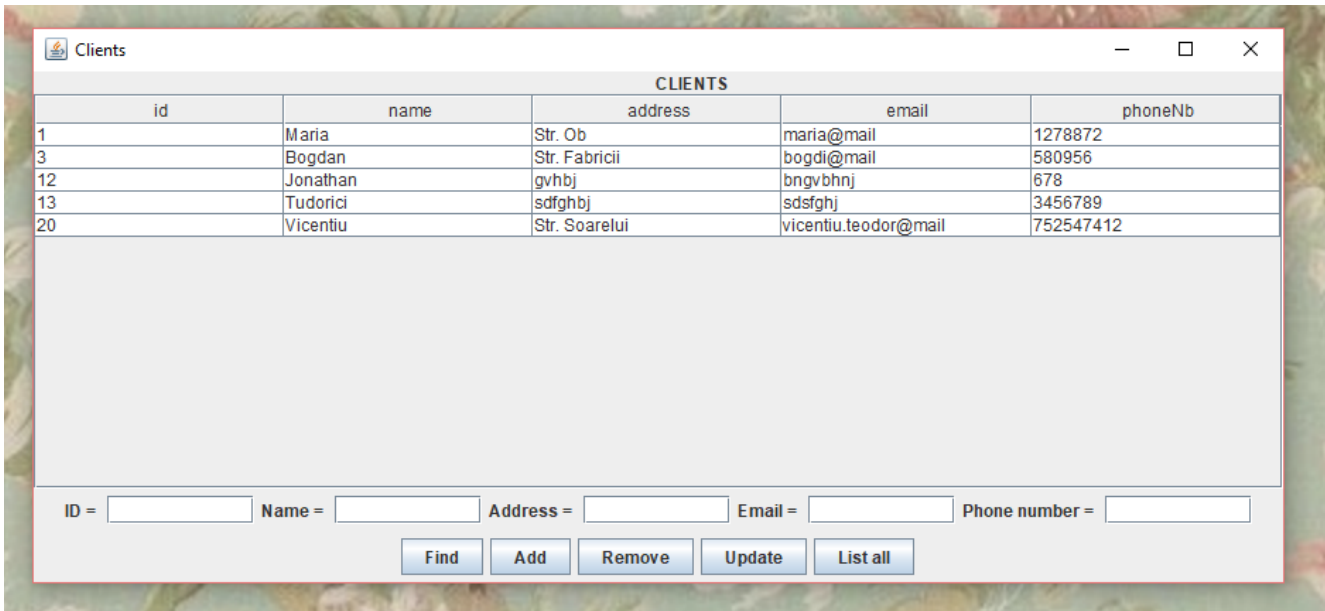
- *connection*: acest pachet conține o singură clasă, *ConnectionFactory*, iar în ea este rezolvată problema conexiunii dintre baza de date și proiect;
 - această clasă are un constructor și mai multe metode ce facilitează realizarea conexiunii: *CreateConnection*, *getConnection*, *close(Statement s)*, *close(Connection c)*, *close(ResultSet rs)*;
 - clasă conține și 5 parametri: *Logger LOGGER*, *String DRIVER*, *String DBURL*, *String USER*, *String PASS* și *ConnectionFactory singleInstance*;
- *model*: clasă în care sunt modelate tabelele din baza de date;
 - conține 3 clase: *Client*, *Order* și *Product*
 - *Client*: conține 5 atribute(*int id*, *String name*, *String address*, *String email* și *String phoneNb*); fiecare atribut are setate gettere și settere; sunt proiectați și 2 constructori, unul cu parametri și unul fără; clasă mai conține și o metodă *toString* adaptată la conținutul său;
 - *Product*: conține 4 atribute(*int productId*, *String productName*, *int stock*, *double price*); fiecare atribut are setate gettere și settere; sunt proiectați și 2 constructori,

unul cu parametri si unul fara; clasa mai contine si o metoda toString adaptata la continutul sau;

- *Order*: contine 5 atribut (int orderId, int clientId, int productId, int quantity, double finalPrice); fiecare atribut are setate gettere si settere; sunt proiectati si 3 constructori, 2 cu parametri si unul fara; clasa mai contine si o metoda toString adaptata la continutul sau; primul constructor primeste un orderId, un clientId, quantity si un obiect de tip Product si astfel instantiaza un nou obiect de tip Order, calculandu-i pretul in functie de produsul ales;
- *dao*: contine clasele folosite pentru manipularea a bazei de date
 - fiecare clasa, *ClientDAO*, *OrderDAO* si *ProductDAO*, are implementate interogari SQL clasice (Insert, Update, Delete, Select) prin intermediul a 5 metode: *findById*, *insert*, *delete*, *update* si *selectAll*;
 - in fiecare metoda se altereaza tabelul descries in functie de cerintele interogarii, memorata intr-un String, si se restabileste conexiunea la baza de date; dupa ce se executa modificarile dorite, se inchide conexiunea;
- *presentation*: pachet in care este inclusa partea de GUI
 - contine 7 clase: *GUI*, *OrderTable*, *ClientTable*, *ProductTable*, *OrderWindow*, *ClientWindow* si *ProductWindow*;
 - *GUI*: are 3 atribut: un ClientWindow, un ClientWindow si un ProductWindow care vor fi initializate in constructorul clasei; de asemenea, aceasta clasa mai contine o metoda (public static Vector<String> getFields(Object object)) care returneaza un array de field-uri corespunzatoare fieldurilor clasei obiectului transmis;
 - Clasele de tip *window* descriu interfata fiecarui tabel si stabilesc action listenerii de pe butoanele din interfata; de asemenea, acestea mai contin si 2 metode (*load*, care pune continutul bazei de date intr-un tabel, si *removeAllRows*, care sterge continutul tabelului afisat), acestea avand rolul de a reafisa tabelele de fiecare data cand baza de data este modificata

- Clasele de tip *table* proiectează tabelele ce trebuie afișate; metoda `addRow` adaugă un rând într-un tabel, în constructor fiind definite coloanele;
- *start*: conține `MainClass`; în această clasă se instantiază un nou obiect de tip GUI;

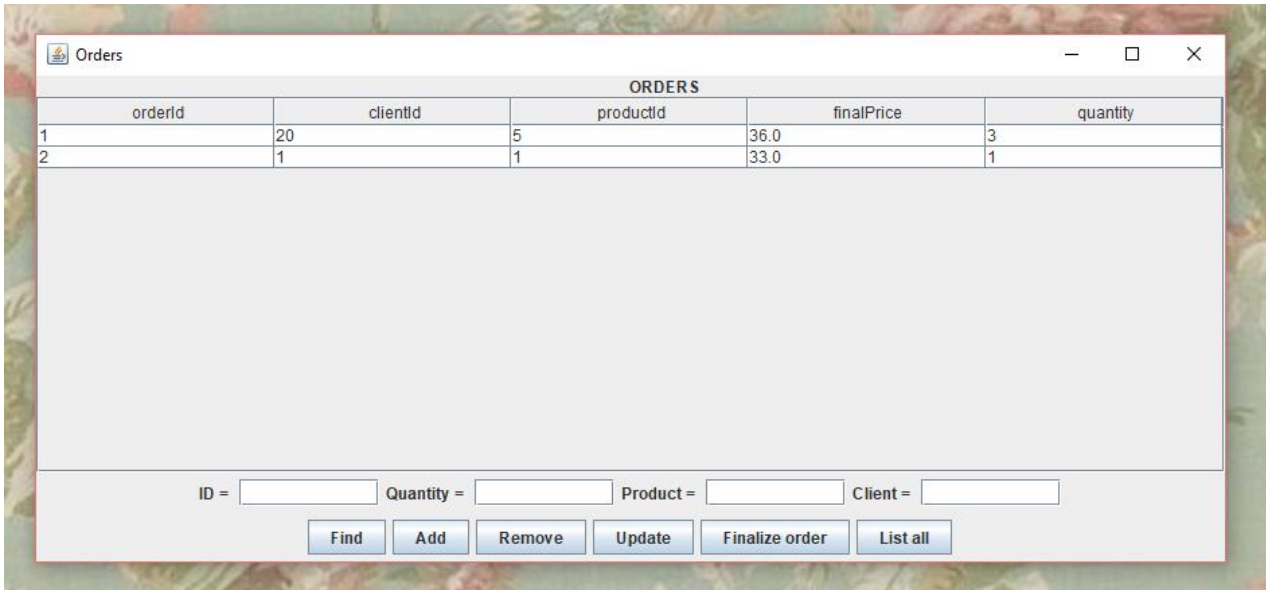
4. Implementare și testare



The screenshot shows a Java Swing window titled "Clients" with a standard Mac OS X title bar (red, yellow, green buttons). The window contains a table with the following data:

CLIENTS				
id	name	address	email	phoneNb
1	Maria	Str. Ob	maria@mail	1278872
3	Bogdan	Str. Fabricii	bogdi@mail	580956
12	Jonathan	gvhbj	bngvbhnj	678
13	Tudorici	sdfghbj	sdsfghj	3456789
20	Vicentiu	Str. Soarelui	vicentiu.teodor@mail	752547412

Below the table is a large empty rectangular area. At the bottom of the window, there are five text input fields labeled "ID =", "Name =", "Address =", "Email =", and "Phone number =", followed by five buttons: "Find", "Add", "Remove", "Update", and "List all".



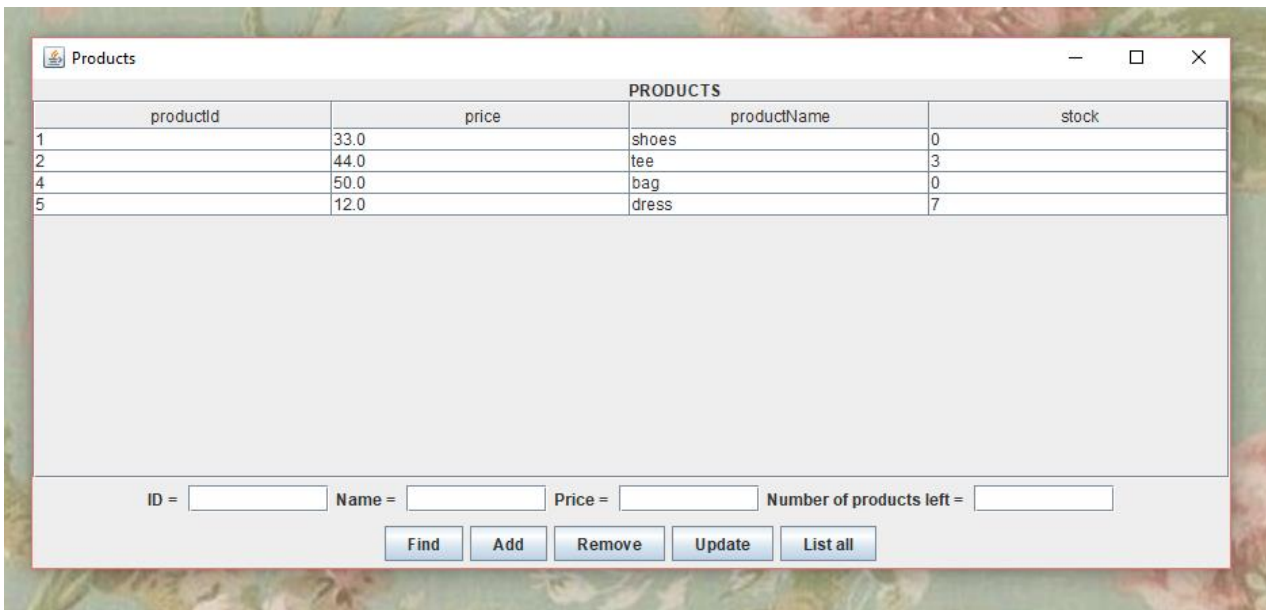
The 'Orders' window displays a table with the following data:

orderid	clientId	productId	finalPrice	quantity
1	20	5	36.0	3
2	1	1	33.0	1

Below the table, there are search and action controls:

ID = Quantity = Product = Client =

Buttons: Find, Add, Remove, Update, Finalize order, List all



The 'Products' window displays a table with the following data:

productId	price	productName	stock
1	33.0	shoes	0
2	44.0	tee	3
4	50.0	bag	0
5	12.0	dress	7

Below the table, there are search and action controls:

ID = Name = Price = Number of products left =

Buttons: Find, Add, Remove, Update, List all

Testare: Am testat fiecare parte a interfeței grafice pentru a verifica funcționarea corectă a operațiilor pe baza de date. Am verificat rezultatele afișând tabelele finale în interfata grafică, dar și în consola pentru a vedea urmările acțiunilor asupra tabelelor.

5. Rezultate obtinute

Pentru a putea folosi aplicatia, utilizatorul trebuie sa aiba JVM instalat si trebuie sa respecte regulile stabilite de introducere a datelor(in spatiile corespunzatoare id-urilor sau preturilor nu trebuie introduse decat cifre; client id si product id pentru order trebuie sa existe in tabelele product si client, altfel se va intampina o eroare; in cazul in care pentru o comanda se cere mai mult decat se afla in stoc, comanda nu va fi modificata sau nu va fi adaugata deloc; la apasarea unui buton de tip Finalize order, comanda va fi procesata si stearsa din tabel). Proiectul functioneaza pentru orice inputuri din partea utilizatorului, atat timp cat se respecta regulile.

6. Concluzii

Aplicatia are implementate operatii asupra unei baze de date ce contine trei tabele. Aceasta functioneaza pentru toate operatiile implementate si are un GUI usor de folosit si inteles. Aplicatia ar putea fi imbunatatita prin adaugarea inca unui tabel(de ex. Payment sau producer) pentru ca baza de date sa fie mai complete. De asemenea poate fi folosita o clasa abstracta pentru implementarea operatiilor pentru a nu fi repetat acelasi cod pentru fiecare tabel atunci cand este proiectata.

7. Bibliografie

1. Wikipedia
2. Tutoriale de Java
3. <https://stackoverflow.com/>
4. <https://www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/>
5. <https://dzone.com/articles/layers-standard-enterprise>
6. <http://tutorials.jenkov.com/java-reflection/index.html>
7. <https://alvinalexander.com/java/>