DOCUMENTATIE

TEMA *3*

queuesManagement

NUME STUDENT: Dimitriu David
GRUPA: 30218

CUPRINS

1. Obiectivul temei

Obiectivul temei este implementarea unei aplicatii pentru gestiunea unor stock-uri dintr-un depozit,a comenzilor si gestionarea bazei de date in care se afla clinetii,produse si comenzile.

Obiective secundare :

1) Analiza problemei si identificarea cerintelor
2) Proiectare aplicatiei de gestiune a comenzilor
3) Implementare aplicatiei de gestiune a comenzilor

Cerințele proiectului sunt :

Requirement	Grading
Minimum to pass • Object-oriented programming design, classes with maximum 300 lines, methods with maximum 30 lines, Java naming conventions • Use *javadoc* for documenting classes and generate the corresponding JavaDoc files. • Use relational databases for storing the data for the application, minimum three tables: Client, Product and Order. • Create a graphical user interface including: ○ A window for client operations: add new client, edit client, delete client, view all clients in a table (JTable)	5 points

○ A window for product operations: add new product, edit product, delete product, view all product in a table (JTable) ○ A window for creating product orders - the user will be able to select an existing product, select an existing client, and insert a desired quantity for the product to create a valid order. In case there are not enough products, an under-stock message will be displayed. After the order is finalized, the product stock is decremented. • Use reflection techniques to create a method 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. • Good quality documentation covering the sections from the documentation template.	
Layered Architecture (the application will contain at least four packages: dataAccessLayer, businessLayer, model and presentation)	2 points
Create a bill for each order as a text file or .pdf file	1 point
Use reflection techniques to create a generic class that contains the methods for accessing the DB: create object, edit object, delete object and find object. The queries for accessing the DB for a specific object that corresponds to a table will be generated dynamically through reflection.	2 points

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

Funtionalitatile proiectului sunt :

a)Permiterea operatiilor de inserare/stergere/actualizare si afisare in baza de date a tabelul de clienti.
b) Permiterea operatiilor de inserare/stergere/actualizare si afisare in baza de date a tabelul de produse.
c) Permiterea operatiilor de inserare/stergere/actualizare si afisare in baza de date a tabelul de comenzi.
Cerinte non-functionale ale proiectului :

a) Interfata grafica a aplicatiei este intuitiva si usor de folosit de catre user.

Este necesar urmarea următorilor pași :

Selectarea interfetei de client,produs sau admin.

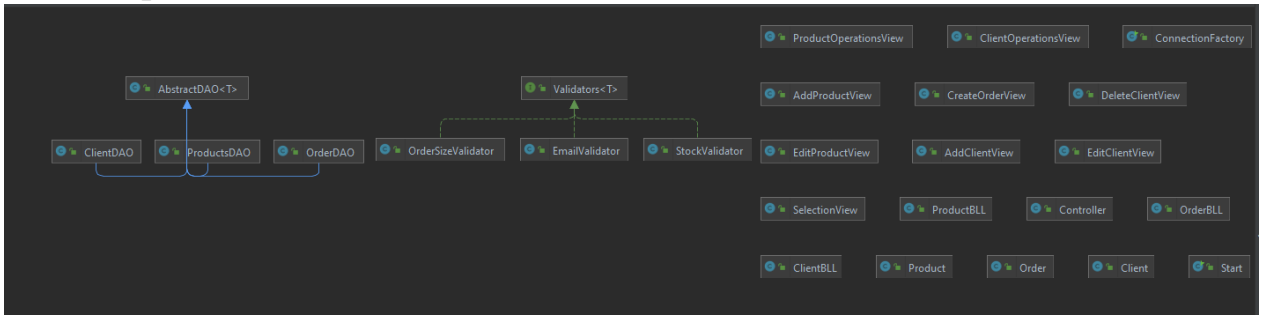
Daca a fost selectata interfata de client,user-ul poate introduce un client,edita,sterge si sa vada tot tabelul de clienti. In cazul apasa pe butonul de add client i se va afisa alt view in care introduce datele pentru client. La fel si pentru update. Pentru delete trebuie sa introduca id ul user-ului pe care vrea sa-l stearga. Daca apasa pe butonul de find all i se va arata pe view tabelul de clienti.

Daca a fost selectata interfata de produs,pasii sunt aceeasi,

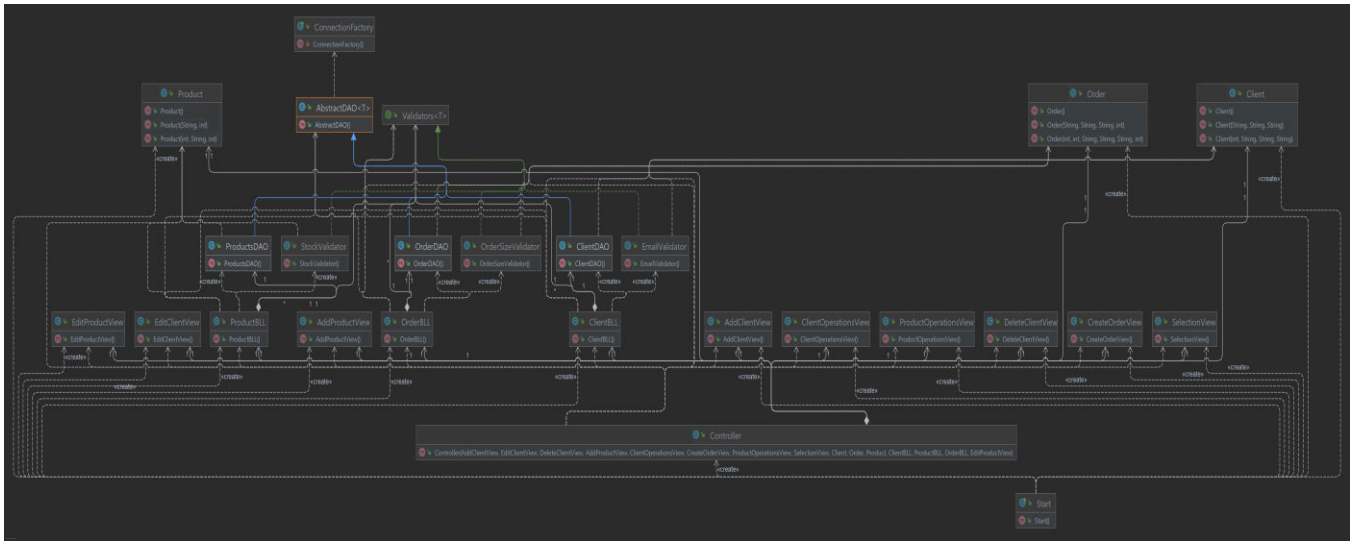
Daca a fost selectata interfata de admin,de orders, user-ul trebuie sa selecteze un client, ce produs vrea sa cumpere si cantitatea introdusa intr-un txtField

3. Proiectare

Proiectul este structurat in pachetele bll care contine pachetul validators,in connection, dao, model, presentation si start. Pentru realizarea calculatorului de polinoame a fost necesar implementarea următoarelor clase : ClientBLL, OrderBLL, ProductBLL, din pachetul bll, ConnectionFactory din pachetul connection,AbstractDAO, ClientDAO, OrderDAO, ProductsDAO din pachetul dao, Client,Order si Product din clasa model , AddClientView, AddProductView, ClientOperationsView,Controller,CreateOrderView,DeleteClientView,EditProduct View, ProductOperationsView, SelectionView din pachetul presentation si clasa Start din pachetul start.



Următoarea figură prezintă principalele proprietăți și metode ale unei clase printr-o diagramă UML, evidențiind în același timp conexiunile dintre ele:



Pentru implementarea controller-ului ne-am folosit de inner clasele cu care am facut legatura cu butoanele din view,am implementat functionalitatile butoanelor :

4. Implementare

Vom prezenta fiecare clasa cu câmpuri si metodele importante. Se va descrie implementarea interfeței utilizator.

Clasele din pachetul Model: Client, Product, Oders

Aceste 3 clase modeleaza tabele cu aceasi nume din baza de date.

In clasa Polynomial am implementat urmatoarele operatii. Campurile lor sunt identice cu coloanele tabelului corespunzator. Ca metode contin doar getters, setters pentru campurile lor.

Clasele din pachetul DAO

AbstractDAO

Este o clasa gennerica se foloseste de reflection pentru a creea query-uri in specifice in

functie de tipul obiectului cu care lucreaza si mai apoi le executa.

Metodele createSelectQuery, createFindAllQuery si createDeleteQuery creaza queryuri

pentru delete si select in fuctie de genericul T.

Metoda findAll executa un query care returneaza toate randurile dintr-un tabel.

Returneaza o list cu obiecte de tip T.

Metoda findById primeste un id si executa un query care returneaza randul dintr-un

tabel al carui id este egal cu parametrul metodei. Returneaza o valoare generica de tip T.

Metoda insert primeste un parametru generic T si executa un query de inserare pe unul

din tabele bazei de date. Se insereaza valorile campurilor parametrului generic. Metoda

returneaza id-ul randului inserat.

Metoda delete primeste un id si executa un query asurpa unuia dintre tabele pentru stergea randului cu id-ul respectiv. Returneaza id-ul randului sters.

Metoda update primeste un parametru de tip generic T si executa un query de actualizare asupra unui tabele. Randul actualizat va avea valorile coloanelor egale cu valorile

campurilor parametrului generic t.

Codul de la metoda care implementeaza insertul din AbstractDao:

```
/**
     * Method that execute an insert query
     * @param t inserted Object
     * @return int
     */
    public int insert(T t) {
        Connection dbConnection = ConnectionFactory.getConnection();

        PreparedStatement insertStatement = null;
        int insertedId = -1;
        ResultSet rs = null;
        try {

            insertStatement = dbConnection.prepareStatement(createInsertQuery(),
Statement.RETURN_GENERATED_KEYS);
            insertValues(insertStatement,t);
            insertStatement.executeUpdate();

            rs = insertStatement.getGeneratedKeys();
```

```
            if (rs.next()) {
                insertedId = rs.getInt(1);
            }
            return insertedId;
        } catch (SQLException e) {
            LOGGER.log(Level.WARNING, "AbstractDAO:insert " +
e.getMessage());
        } finally {
            ConnectionFactory.close(rs);
            ConnectionFactory.close(insertStatement);
            ConnectionFactory.close(dbConnection);
        }
        return -1;
    }
```

ClientDAO

Clasa mosteneste AbstractDAO.

Clasa aceasta creaza query-uri specifice pentru tabelul Client.

Metoda createInsertQuery creaza un query de inserare, iar metoda createUpdateQuery
creaza un query de actualizare. Ambele nu au parametri si returneaza un string. Metoda setInsertValues primeste ca parametru un statement si un client. Valorile campurilor clientului sunt puse pe pozitile valorilor ce trebuie inserate din statement.

Metoda setUpdateValues primeste aceasi 2 parametrii ca si metoda setInsertsValues,

doar ca valorile campurilor clientului sunt adaugate intr-un statement de update.

ProductDAO

Clasa mosteneste AbstractDAO.

Clasa aceasta creaza query-uri specifice pentru tabelul Product.

Metoda createInsertQuery creaza un query de inserare, iar metoda createUpdateQuery

creaza un query de actualizare. Ambele nu au parametri si returneaza un string.

Metoda setInsertValues primeste ca parametru un statement si un product. Valorile

campurilor produsului sunt puse pe pozitile valorilor ce trebuie inserate din statement.

Metoda setUpdateValues primeste aceasi 2 parametrii ca si metoda setInsertsValues,

doar ca valorile campurilor produsului sunt adaugate intr-un statement de update.

Pachetul BLL: pachetul Validators

In acest pachet sunt clase care implementeaza interfata Validator si implicit metoda validate. Clasele de validare pentru obiectele de tip Client sunt: AgeValidator, care se asigura

ca clientul are varsta minima de 10 ani, si EmaiValidator, care se asigur ca email-ul clientului respecta un anumit format.

Clasele de validate pentru obiectele de tip Product sunt: PriceValidator, care se asigur ca pretul produsului nu e negativ, si QuantityValidator, care se asigura ca cantitatea

produsului nu e negativa.

Clasa de validate pentru obiectele de tip Orders e QuantityOrderValidator, care se

asigura ca cantitate de produse de pe comanda nu e negativa.

Toate aceste clase arunca o exceptie in cazul in care obiectul nu e valid.

Pachetul Bll: clasa ClientBLL

Aceasta clasa are ca si campuri o lista de validatori specifici pentru obiectele de tip

client si un obiect instanta a clasei ClientDAO.

Sunt implementeate metode care apleaza metodele de inserare, update, delete, findAll si findById din din clasa ClientDAO. Metodele de insert si update primesc ca parametru un

client pe care il si valideaza folosind obiectele din lista de validare a clasei.

Pachetul BLL : clasa ProductBLL

Aceasta clasa are ca si campuri o lista de validatori specifici pentru obiectele de tip

Product si un obiect instanta a clasei ProductDAO. Sunt implementeate metode care apleaza metodele de inserare, update, delete, findAll si findById din din clasa ProductDAO. Metodele de insert si update primesc ca parametru un

produs pe care il si valideaza folosind obiectele din lista de validare a clasei.

Pachetul BLL : clasa OrderBLL

Aceasta clasa are ca si campuri o lista de validatori specifici pentru obiectele de tip Orders si un obiect instanta a clasei OrderDAO.

Sunt implementeate metode care apleaza metodele de inserare si findAll din clasa

OrderDAO. Metoda de insert primeste ca parametru o comanda pe care o si valideaza folosind obiectele din lista de validare a clasei. Metoda createPDF primeste ca paramentru o comanda si o valoare de tip int numita totalPrice. Daca comanda e valida atunci creaza un pdf care reprezinta factura pentru comanda respectiva.

Pachetul Presentation: Clasa View

Clasa mosteneste JFrame si este fereastra principala a aplicatiei.

In aceasta clasa sunt 3 butoane pentru a selecta tabelul asupra caruia vrem sa facem

operatii: Client, Product sau Orders. Clasa are ca metode doar getters pentru cele 3 butoane. Pachetul Presentation: Clasa ClientView Clasa mosteneste JFrame.

In aceasta clasa, care implementeaza interfata grafica pentru operatiile din tabelul

Client, avem textField-uri pentru a introduce id-ul, numele, varsta si email-ul si butoane pentru operatii : insert, delete, update, findById, findAll. De asemenea avem si un buton numit exit care inchide aceasta fereastra si o deschide pe cea principala. Are metode de get pentru butoane si textField-uri.

Pachetul Presentation: Clasa ProductView

Clasa mosteneste JFrame. In aceasta clasa, care implementeaza interfata grafica pentru operatiile din tabelul Product, avem textField-uri pentru a introduce id-ul, numele, cantitatea si pretul si butoane pentru operatii : insert, delete, update, findById, findAll. De asemenea avem si un buton numit exit care inchide aceasta fereastra si o deschide pe cea principala. Are metode de get pentru butoane si textField-uri.

Pachetul Presentatiom: Clasa OrderView

Clasa mostenteste JFrame. In aceasta clasa, care implementeaza interfata grafica pentru operatiile din tabelul Orders, avem un textField pentru a introduce cantitate si butoane pentru operatiile de creatOrder si findAll. De asemenea avem si un buton numit exit care inchide aceasta fereastra si o deschide pe cea principala. Are metode de get pentru butoane si textField.

Pachetul Presentation: Clasa TableView

Clasa mostenteste JFrame. Aceasta clasa contine tabelul in care se afiseaza rezultatele operatiilor de select. Metoda createTable primeste ca parametru o lista de obiecte generice(de tip T) si folosind reflection creaza un nou tabel pe care il returneza.

Pachetul Presentation: Clasa Controller

Aceasta clasa are ca si campuri un view, un clientView, un productView, un

orderView , 2 tableView-uri, unul pentru obiecte de tip Client si unul pentru obiecte de tip Product, si cate un obiect din clasele ClientBLL, ProductBLL,

OrderBLL. Implementeaza interfata ActionListener si se adauga pe sine ca si ascultator pentru obiectele din ferestrele interfetei grafice.

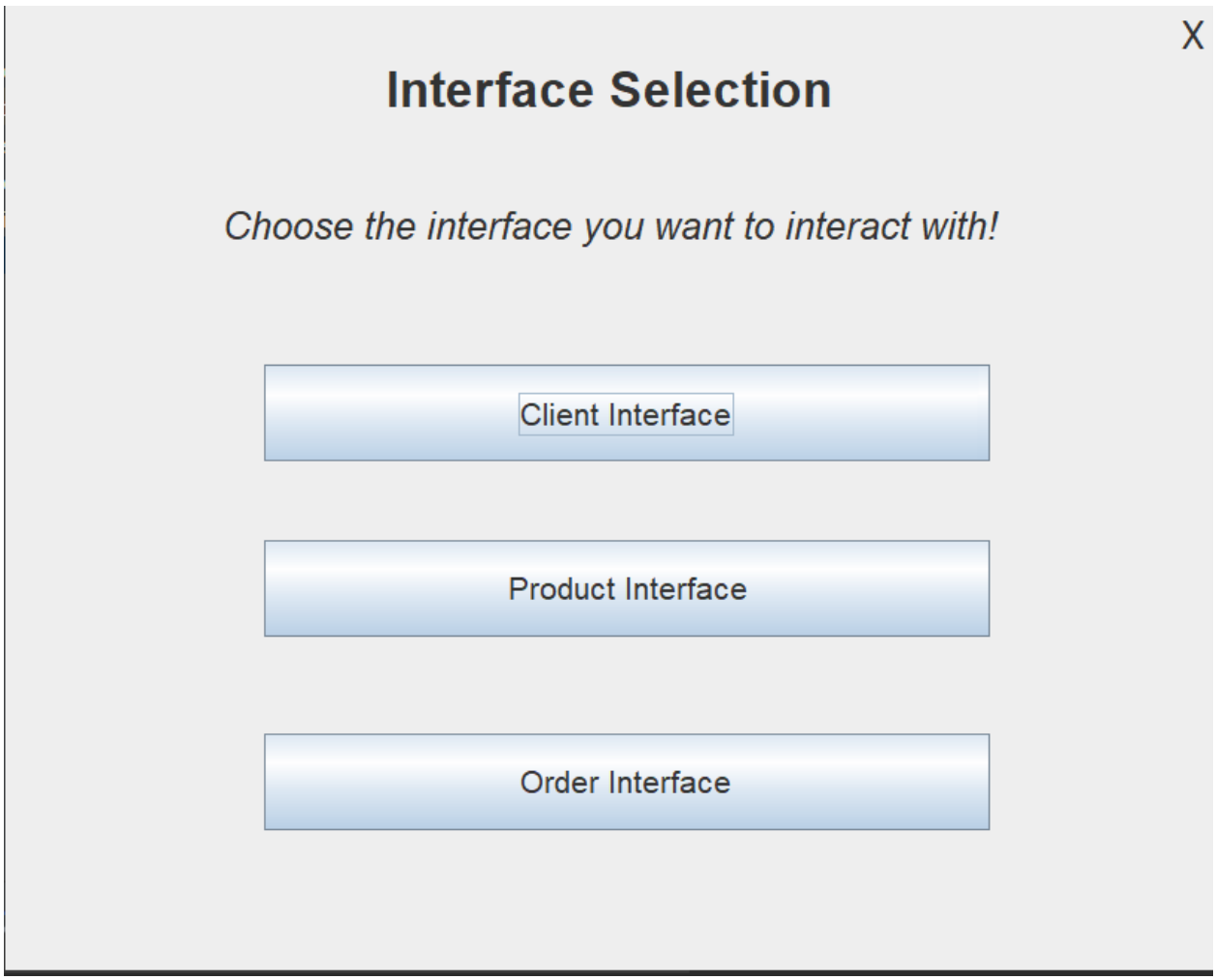
Pachetul Start: Clasa Start

In aceasta clasa ne creeam obiecte de timpul potrivit pentru a instantia un obiect de tipul Controller in metoda main.

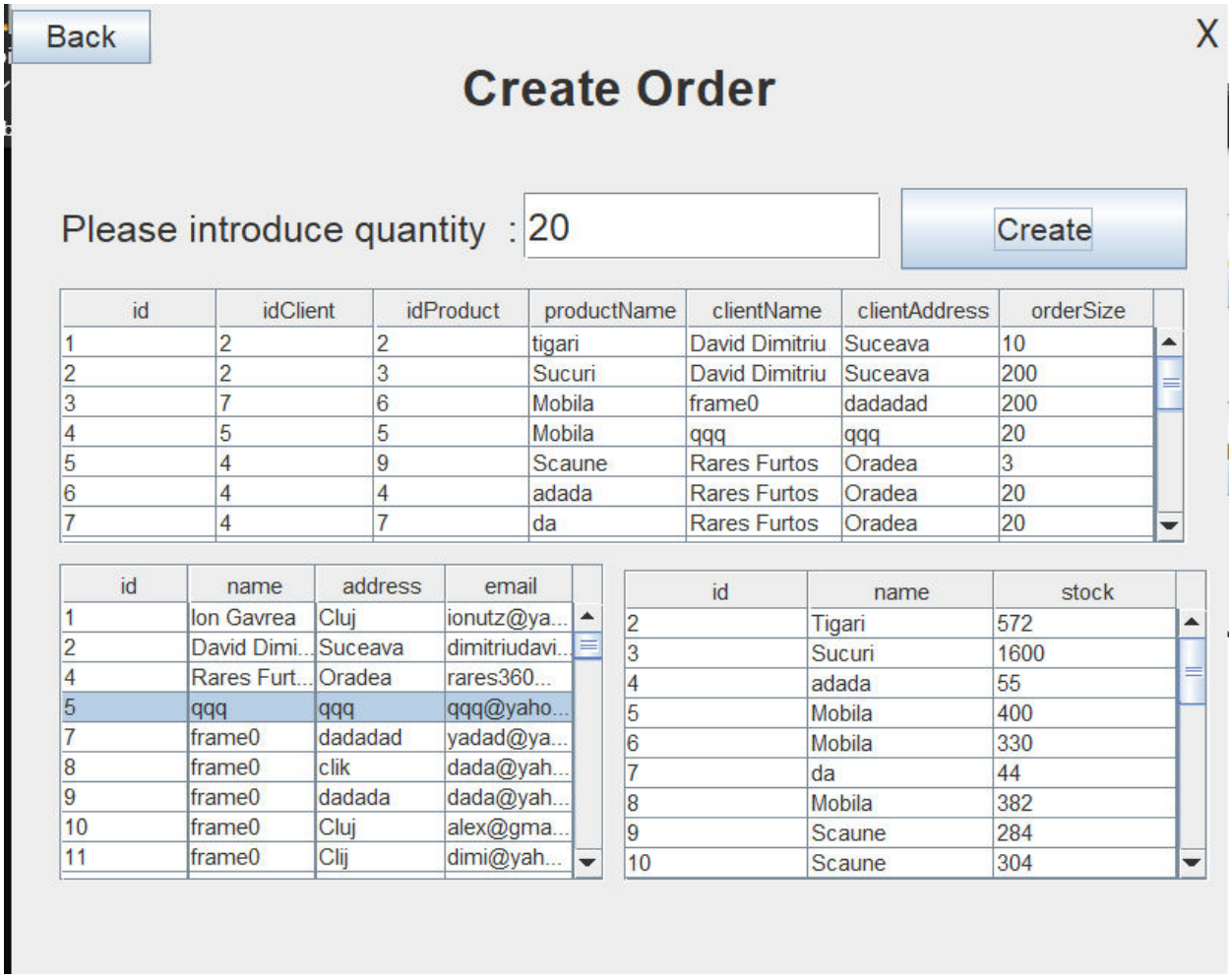
```
package start;


import bll.ClientBLL;
import bll.OrderBLL;
import bll.ProductBLL;
import model.Client;
import model.Order;
import model.Product;
import presentation.*;


/**
 * Class that contain main method
 */
public class Start {


    /**
     * main method
     * @param args
     */
    public static void main(String[] args) {
```

```
        Client client = new Client();
        Order order = new Order();
        Product product = new Product();
        ClientBLL clientBLL = new ClientBLL();
        ProductBLL productBLL = new ProductBLL();
        OrderBLL orderBLL = new OrderBLL();
        AddClientView addEditClientView = new AddClientView();
        DeleteClientView deleteClientView = new DeleteClientView();
        EditProductView editProductView = new EditProductView();

        AddProductView addProductView = new AddProductView();
        EditClientView editClientView = new EditClientView();
        ClientOperationsView clientOperationsView = new
ClientOperationsView();
        CreateOrderView createOrderView = new CreateOrderView();
        ProductOperationsView productOperationsView = new
ProductOperationsView();
        SelectionView selectionView = new SelectionView();
        Controller controller = new
Controller(addEditClientView,editClientView,deleteClientView,
addProductView,clientOperationsView,createOrderView,productOperationsView,s
electionView,client,order,product,clientBLL,productBLL,orderBLL,editProductVie
w);
    }
}
```

Utilizatorul va interacționa cu acest view :



Interfata pentru Orders:

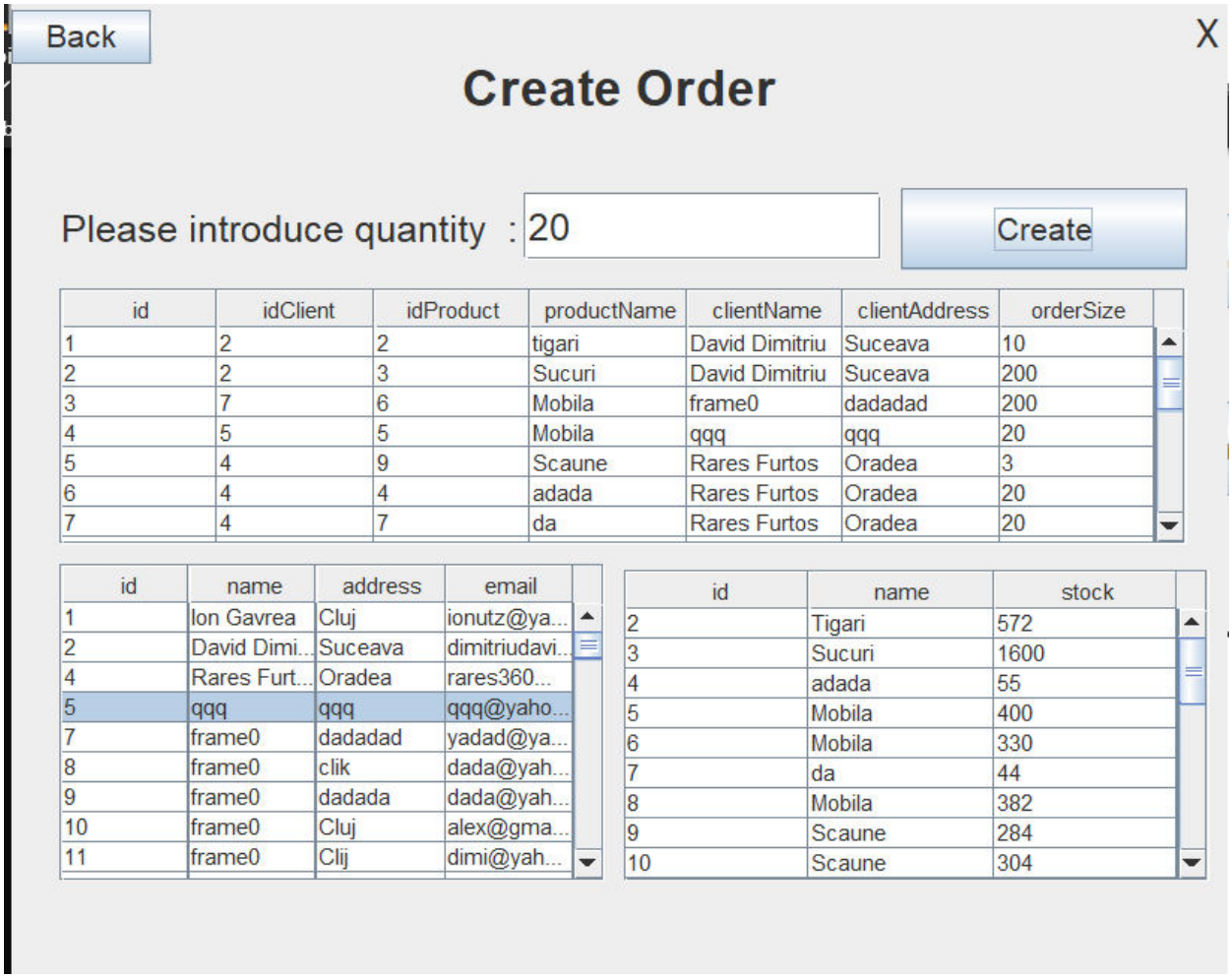


5. Rezultate

Ca mod de testare, aplicația se conectează la o bază de date cu 3 tabele:

client, vizualizare și Comenzile și operațiunile implementate care pot fi efectuate pentru fiecare tabel sunt testate a lui. Pentru tabelele Client și Product, operațiuni: inserați, ștergeți, actualizați, findById și findAll. Pentru tabelul Comenzi, operațiuni: insert (creează comandă) și findAll. Toate aceste operațiuni au

Succes deoarece tabelul din baza de date s-a schimbat conform așteptărilor după fiecare operațiune.



6. Concluzii

După acest proiect am învățat cum să conectez aplicația java la baza de date

Și cum putem extrage sau modifica informații din baza de date din interiorul aplicației. De asemenea, am învățat cum să folosesc reflecția și clasele generice.

Este un pic complicat în comparație cu alte subiecte, dar cred că acestea din urmă

În acest scop, am dobândit câteva abilități noi care mi-au aprofundat înțelegerea

programare.

7. Bibliografie

https://users.utcluj.ro/~igiosan/teaching_poo.html - cursuri OOP Ionel Giosan

https://dsrl.eu/courses/pt/materials/A3_Support_Presentation.pdf - prezentarea proiectului

https://www.w3schools.com/java/java_arraylist.asp - informatii despre ArrayList<>

https://dsrl.eu/courses/pt/materials/PT2021-2022_Assignment_3.pdf

https://regex101.com/r/nI5oA5/1 -regex demo