Assignment 3

-Documentation-

A warehouse is a building for storing goods. Warehouses are used by manufacturers, importers, exporters, wholesalers, transport businesses, customs, etc. They are usually large plain buildings in industrial parks on the outskirts of cities, towns or villages.

They usually have loading docks to load and unload goods from trucks. Sometimes warehouses are designed for the loading and unloading of goods directly from railways, airports, or seaports. They often have cranes and forklifts for moving goods, which are usually placed on ISO standard pallets loaded into pallet racks. Stored goods can include any raw materials, packing materials, spare parts, components, or finished goods associated with agriculture, manufacturing, and production. In India, a warehouse may be referred to as a go down.

For a warehouse to be productive it need a way to process all the products and orders. In this way it can be a program that does this.

A screenshot of a map

Description automatically generated

Fig 1. Presentation Logic Package

This is the package from the java program where every frame, that the user uses for manipulating the Data base data, is placed.

A screenshot of a cell phone

Description automatically generated

Fig 2. Business Logic Package

This is the package that links the Presentation Logic with the data Logic package (By mistake I combined the presentation logic with the data logic which had to be the business logic).

A screenshot of a cell phone

Description automatically generated

Fig 3. Data Logic Layer – Package

This is the package that accesses the Data Base with all the queries.

A screenshot of a cell phone

Description automatically generated

Fig 4. Models Package

This Package contains all the classes that are needed to access the Data Base and collect data from the DB more easily.

A screenshot of a social media post

Description automatically generated

Fig 5. All Packages

**Models Package**

Client Class

package ro.utcn.pt.assignment3.Models;

public class Client {

protected int id;

protected String name;

protected String address;

public Client() {

}

public Client(int id, String name, String address) {

this.id = id;

this.name = name;

this.address = address;

}

public Client(String name, String address) {

this.name = name;

this.address = address;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

}

Product Class

package ro.utcn.pt.assignment3.Models;

public class Product {

public int product\_id;

public String name;

public double price;

public int quantity;

public Product() {

}

public Product(int product\_id, String name, double price, int quantity) {

this.product\_id = product\_id;

this.name = name;

this.price = price;

this.quantity = quantity;

}

public Product(String name, double price, int quantity) {

this.name = name;

this.price = price;

this.quantity = quantity;

}

public int getProduct\_id() {

return product\_id;

}

public void setProduct\_id(int product\_id) {

this.product\_id = product\_id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

}

Order Class

package ro.utcn.pt.assignment3.Models;

import java.util.ArrayList;

public class Order {

public int order\_id;

public int client\_id;

public String client\_name;

public double totalSum;

public int product\_id;

public String product\_name;

public int quantity;

public Order() {

}

public Order(int order\_id, int client\_id, String client\_name, double totalSum, int product\_id, String product\_name, int quantity) {

this.order\_id = order\_id;

this.client\_id = client\_id;

this.client\_name = client\_name;

this.totalSum = totalSum;

this.product\_id = product\_id;

this.product\_name = product\_name;

this.quantity = quantity;

}

public String getClient\_name() {

return client\_name;

}

public void setClient\_name(String client\_name) {

this.client\_name = client\_name;

}

public String getProduct\_name() {

return product\_name;

}

public void setProduct\_name(String product\_name) {

this.product\_name = product\_name;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public int getOrder\_id() {

return order\_id;

}

public void setOrder\_id(int order\_id) {

this.order\_id = order\_id;

}

public int getClient\_id() {

return client\_id;

}

public void setClient\_id(int client\_id) {

this.client\_id = client\_id;

}

public double getTotalSum() {

return totalSum;

}

public void setTotalSum(double totalSum) {

this.totalSum = totalSum;

}

public int getProduct\_id() {

return product\_id;

}

public void setProduct\_id(int product\_id) {

this.product\_id = product\_id;

}

}

**Data Layer Package**

DBConnection Class

*package* ro.utcn.pt.assignment3.DataLayer;  
*import* java.sql.*Connection*;  
*import* java.sql.DriverManager;  
  
*/\*\*  
 \* Creates the connection to the Data Base  
 \* \*/  
public class* DBConnection {  
  
 *static final* String JDBC\_DRIVER = "com.mysql.jdbc.Driver";  
 *static final* String DB\_URL = "jdbc:mysql://localhost/ordermanagement";  
  
 *static final* String USER = "root";  
 *static final* String PASS = "spune-miIshmael";  
  
 *public Connection* connection;  
  
 *public static* DBConnection *db*;  
  
 */\*\*  
 \* Constructor creates the database connection  
 \* \*/  
 private* **DBConnection**(){  
 *try*{  
 System.out.println("Connecting to DATABASE...\n");  
 connection = DriverManager.*getConnection*(DB\_URL, USER, PASS);  
 }*catch*(Exception e){  
 e.printStackTrace();  
 }  
 }  
  
 *public static synchronized* DBConnection getConnection(){  
 *if*(*db* == *null*){  
 *db* = *new* DBConnection();  
 }  
 *return db*;  
 }  
}

ClientOp Class

*package* ro.utcn.pt.assignment3.DataLayer;  
  
*import* ro.utcn.pt.assignment3.Models.Client;  
*import* sun.util.cldr.CLDRLocaleDataMetaInfo;  
  
*import* java.sql.*Connection*;  
*import* java.sql.*PreparedStatement*;  
*import* java.sql.*ResultSet*;  
*import* java.sql.SQLException;  
*import* java.util.ArrayList;  
  
*/\*\*  
 \* This Class operates the Data Base Table Client  
 \*  
 \* \*/  
  
public class* ClientOP {  
  
 */\*\*  
 \* This method returns all the clients from the Data Base  
 \** ***@param*** *connection - The connection to the Data Base  
 \** ***@return*** *An array list of clients  
 \* \*/  
 public* ArrayList<Client> returnClients(*Connection* connection) *throws* SQLException{  
  
 ArrayList<Client> foundClients = *new* ArrayList<>();  
  
 String stmt = "Select \* from client";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 *ResultSet* resultSet = preparedStatement.executeQuery();  
  
 *while* (resultSet.next()){  
 Client client = *new* Client(resultSet.getInt("client\_id"), resultSet.getString("name"), resultSet.getString("address"));  
 foundClients.add(client);  
 }  
 *return* foundClients;  
 }  
  
 */\*\*  
 \* This method return only the clients name (all clients names)  
 \** ***@param*** *connection - Data Base Connection  
 \** ***@return*** *An arrayList of clients  
 \* \*/  
 public* ArrayList<String> getAllClientsName(*Connection* connection) *throws* SQLException{  
  
 ArrayList<String> foundClients = *new* ArrayList<>();  
  
 String stmt = "Select \* from client";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 *ResultSet* resultSet = preparedStatement.executeQuery();  
  
 *while* (resultSet.next()){  
 String client = resultSet.getString("Name");  
 foundClients.add(client);  
 }  
 *return* foundClients;  
 }  
  
 */\*\*  
 \* This method returns a client based on the ID  
 \** ***@param*** *connection - Data Base Connection  
 \** ***@param*** *client\_id - The id of client in the Data Base  
 \** ***@return*** *The searched client  
 \* \*/  
 public* Client getClientbyID(*Connection* connection, *int* client\_id)*throws* SQLException{  
 Client foundClient = *new* Client();  
  
 String stmt = "Select \* from client where client\_id = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setInt(1, client\_id);  
 *ResultSet* resultSet = preparedStatement.executeQuery();  
  
 *while* (resultSet.next()){  
 foundClient.setId(resultSet.getInt("client\_id"));  
 foundClient.setName(resultSet.getString("name"));  
 foundClient.setAddress(resultSet.getString("Address"));  
 }  
  
 *return* foundClient;  
 }  
  
 */\*\*  
 \* This Method adds a client to the Data Base  
 \** ***@param*** *connection - Data Base Connection  
 \** ***@param*** *client - The client that needs to be inserted into the Data Base  
 \*  
 \* \*/  
 public void* addClient (*Connection* connection, Client client) *throws* SQLException{  
 String stmt = "Insert into Client (Name, Address) Values (?,?)";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setString(1, client.getName());  
 preparedStatement.setString(2, client.getAddress());  
  
 preparedStatement.executeUpdate();  
 }  
  
 */\*\*  
 \* This Method edits an existing client's name  
 \** ***@param*** *connection - Data Base connection  
 \** ***@param*** *client - the client that is being updated  
 \** ***@param*** *newName - The new name of the client  
 \* \*/  
 public void* editClientName (*Connection* connection, Client client, String newName) *throws* SQLException{  
 String stmt = "Update Client set name = ? where client\_id = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setString(1, newName);  
 preparedStatement.setInt(2, client.getId());  
  
 preparedStatement.executeUpdate();  
 }  
  
 */\*\*  
 \* This method edits an existing client's address  
 \** ***@param*** *connection - Data Base connection  
 \** ***@param*** *client - The client that is being updated  
 \** ***@param*** *newAddress - The new Address of the client  
 \* \*/  
 public void* editClientAddress(*Connection* connection, Client client, String newAddress) *throws* SQLException{  
 String stmt = "Update Client set Address = ? where client\_id = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setString(1, newAddress);  
 preparedStatement.setInt(2, client.getId());  
 preparedStatement.executeUpdate();  
 }  
  
 */\*\*  
 \* This method deletes and existing client  
 \** ***@param*** *connection - Data Base connection  
 \** ***@param*** *client - The client that is being deleted  
 \* \*/  
 public void* deleteClient(*Connection* connection, Client client) *throws* SQLException{  
 String stmt = "Delete From Client where client\_id = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setInt(1, client.getId());  
  
 preparedStatement.executeUpdate();  
 }  
  
 */\*\*  
 \* This method verifies if the client exists (by id)  
 \** ***@param*** *connection - Data Base connection  
 \** ***@param*** *id - The id of the client in the Data Base  
 \** ***@return*** *true or false  
 \* \*/  
 public boolean* existsClient(*Connection* connection, *int* id) *throws* SQLException{  
 String stmt = "Select \* from client where client\_id = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setInt(1, id);  
 *ResultSet* resultSet = preparedStatement.executeQuery();  
  
 *if*(resultSet.next())  
 *return true*;  
 *else  
 return false*;  
 }  
}

ProductOp Class

*package* ro.utcn.pt.assignment3.DataLayer;  
  
*import* ro.utcn.pt.assignment3.Models.Product;  
  
*import* java.sql.*Connection*;  
*import* java.sql.*PreparedStatement*;  
*import* java.sql.*ResultSet*;  
*import* java.sql.SQLException;  
*import* java.util.ArrayList;  
  
*/\*\*  
 \* This Class manipulates the products in the Data Base table product  
 \* \*/  
public class* ProductOp {  
  
 */\*\*  
 \* This method returns all the available products  
 \** ***@param*** *connection - Data Base Connection  
 \** ***@return*** *An arrayList with all the existing products  
 \* \*/  
 public* ArrayList<Product> viewAllProducts(*Connection* connection) *throws* SQLException{  
  
 ArrayList<Product> allProducts = *new* ArrayList<>();  
  
 String stmt = "Select \* from Product";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 *ResultSet* resultSet = preparedStatement.executeQuery();  
  
 *while*(resultSet.next()){  
 Product product = *new* Product(resultSet.getInt("product\_id"), resultSet.getString("name"),  
 resultSet.getDouble("price"), resultSet.getInt("quantity"));  
 allProducts.add(product);  
  
 }  
 *return* allProducts;  
 }  
  
 */\*\*  
 \* This method returns all the names of the available products  
 \** ***@param*** *connection - Data Base connection  
 \** ***@return*** *An arrayList with all the names of the available products  
 \* \*/  
 public* ArrayList<String> getAllProductsNames(*Connection* connection) *throws* SQLException{  
 ArrayList<String> allProducts = *new* ArrayList<>();  
  
 String stmt = "Select \* from Product";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 *ResultSet* resultSet = preparedStatement.executeQuery();  
  
 *while*(resultSet.next()){  
  
 String productName = resultSet.getString("Name");  
 allProducts.add(productName);  
  
 }  
 *return* allProducts;  
 }  
  
 */\*\*  
 \* This method returns the product by it's table id  
 \** ***@param*** *connection - Data Base Connection  
 \** ***@param*** *product\_id - The id of the product in the product table  
 \** ***@return*** *The searched product  
 \* \*/  
 public* Product getProductByID(*Connection* connection, *int* product\_id) *throws* SQLException{  
  
 Product product = *new* Product();  
  
 String stmt = "Select \* from product where product\_id = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setInt(1, product\_id);  
 *ResultSet* resultSet = preparedStatement.executeQuery();  
  
 *while* (resultSet.next()) {  
 product.setProduct\_id(resultSet.getInt("product\_id"));  
 product.setName(resultSet.getString("name"));  
 product.setPrice(resultSet.getDouble("price"));  
 product.setQuantity(resultSet.getInt("quantity"));  
 }  
  
 *return* product;  
 }  
  
 */\*\*  
 \* This method returns the product by it's name  
 \** ***@param*** *connection - Data Base Connection  
 \** ***@param*** *name - The name of the product  
 \** ***@return*** *The searched product  
 \* \*/  
 public* Product getProductByName(*Connection* connection, String name) *throws* SQLException{  
  
 Product foundProduct = *new* Product();  
  
 String stmt = "Select \* from product where name = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setString(1, name);  
 *ResultSet* resultSet = preparedStatement.executeQuery();  
  
 *while*(resultSet.next()){  
 foundProduct.setName(resultSet.getString("name"));  
 foundProduct.setPrice(resultSet.getDouble("price"));  
 foundProduct.setQuantity(resultSet.getInt("quantity"));  
 foundProduct.setProduct\_id(resultSet.getInt("product\_id"));  
 }  
  
 *return* foundProduct;  
 }  
  
 */\*\*  
 \* This method adds a product to the Data Base table product  
 \** ***@param*** *connection - Data Base connection  
 \** ***@param*** *product - The product to be added to the Data Base  
 \*  
 \* \*/  
 public void* addProduct(*Connection* connection, Product product) *throws* SQLException{  
 String stmt = "Insert into product(name, price, quantity) values (?, ?, ?)";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setString(1, product.getName());  
 preparedStatement.setDouble(2, product.getPrice());  
 preparedStatement.setInt(3, product.getQuantity());  
 preparedStatement.executeUpdate();  
 }  
  
 */\*\*  
 \* THis method checks to see if the product exists (by name)  
 \** ***@param*** *connection - Data Base Connection  
 \** ***@param*** *name - The name of the product  
 \** ***@return*** *true or false  
 \* \*/  
 public boolean* existsProductByName(*Connection* connection, String name) *throws* SQLException{  
 String stmt = "Select \* from product where name = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setString(1, name);  
 *ResultSet* resultSet = preparedStatement.executeQuery();  
  
 *if*(resultSet.next())  
 *return true*;  
 *else  
 return false*;  
 }  
  
 */\*\*  
 \* This method edits the name of the product with a given id  
 \** ***@param*** *connection - Data Base connection  
 \** ***@param*** *newName - The new name for the product  
 \** ***@param*** *id - The id of the product in the Data Base  
 \* \*/  
 public void* editProductName(*Connection* connection, String newName, *int* id) *throws* SQLException{  
 String stmt = "Update product set name = ? where product\_id = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setString(1, newName);  
 preparedStatement.setInt(2, id);  
 preparedStatement.executeUpdate();  
 }  
  
 */\*\*  
 \* This method edits the price of the product with a given id  
 \** ***@param*** *connection - Data Base Connection  
 \** ***@param*** *newPrice - The new price for the product  
 \** ***@param*** *id - The id of the product in the Data Base  
 \* \*/  
 public void* editProductPrice(*Connection* connection, Double newPrice, *int* id) *throws* SQLException{  
 String stmt = "Update product set price = ? where product\_id = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setDouble(1, newPrice);  
 preparedStatement.setInt(2, id);  
 preparedStatement.executeUpdate();  
 }  
  
 */\*\*  
 \* This method edits the quantity available for the product  
 \** ***@param*** *connection - Data Base connection  
 \** ***@param*** *newQuantity - The new quantity for the product  
 \** ***@param*** *id - The id of the product in the Data Base  
 \* \*/  
 public void* editProductQuantity(*Connection* connection, *int* newQuantity, *int* id) *throws* SQLException{  
 String stmt = "Update product set quantity = ? where product\_id = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setInt(1, newQuantity);  
 preparedStatement.setInt(2, id);  
 preparedStatement.executeUpdate();  
 }  
  
 */\*\*  
 \* This method deletes a product  
 \** ***@param*** *connection - Data Base Connection  
 \** ***@param*** *id - The id of the product to be deleted  
 \*  
 \* \*/  
 public void* deleteProductByID(*Connection* connection, *int* id) *throws* SQLException{  
 String stmt = "Delete from product where product\_id = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setInt(1, id);  
 preparedStatement.executeUpdate();  
 }  
  
 */\*\*  
 \* This method returns the quantity of a product  
 \** ***@param*** *connection - Data Base Connection  
 \** ***@param*** *id - The id of the product in the Data Base  
 \** ***@return*** *The available quantity for that product  
 \* \*/  
 public int* getProductQuantityByID(*Connection* connection, *int* id) *throws* SQLException{  
 String stmt = "Select quantity from product where product\_id = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setInt(1, id);  
 *ResultSet* resultSet = preparedStatement.executeQuery();  
 *int* quantity = 0;  
 *while* (resultSet.next()){  
 quantity = resultSet.getInt("quantity");  
 }  
 *return* quantity;  
 }  
}

OrderOp Class

*package* ro.utcn.pt.assignment3.DataLayer;  
  
*import* com.sun.org.apache.regexp.internal.RE;  
*import* com.sun.org.apache.xpath.internal.operations.Or;  
*import* ro.utcn.pt.assignment3.Models.Client;  
*import* ro.utcn.pt.assignment3.Models.Order;  
*import* ro.utcn.pt.assignment3.Models.Product;  
  
*import* java.sql.*Connection*;  
*import* java.sql.*PreparedStatement*;  
*import* java.sql.*ResultSet*;  
*import* java.sql.SQLException;  
*import* java.util.ArrayList;  
  
*/\*\*  
 \* This Class manipulates the Data base table productOrder  
 \* \*/  
public class* OrderOp {  
  
 */\*\*  
 \* This method returns all the orders registered in the Data Base  
 \** ***@param*** *connection - Data Base connection  
 \** ***@return*** *An arrayList of all the Orders  
 \* \*/  
 public* ArrayList<Order> getAllOrders(*Connection* connection) *throws* SQLException{  
  
 ArrayList<Order> allOrders= *new* ArrayList<>();  
  
 String stmt = "Select \* from productOrder";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 *ResultSet* resultSet = preparedStatement.executeQuery();  
  
 *while*(resultSet.next()){  
 Order foundOrder = *new* Order(resultSet.getInt("order\_id"), resultSet.getInt("client\_id"),  
 resultSet.getString("client\_name"), resultSet.getDouble("totalSum"), resultSet.getInt("product\_id"),  
 resultSet.getString("product\_name"), resultSet.getInt("quantity"));  
 allOrders.add(foundOrder);  
 }  
  
 *return* allOrders;  
 }  
  
 */\*\*  
 \* This method edits the quantity registered in the order  
 \** ***@param*** *connection - Data Base connection  
 \** ***@param*** *order - The order that it's being edited  
 \** ***@param*** *newQuantity - The new quantity for the order  
 \*  
 \* \*/  
 public void* editOrderQuantity(*Connection* connection, Order order, *int* newQuantity) *throws* SQLException{  
 String stmt = "Update productOrder set quantity = ? where order\_id = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setInt(1, newQuantity);  
 preparedStatement.setInt(2, order.getOrder\_id());  
 preparedStatement.executeUpdate();  
  
 String stmt2 = "Update product set quantity = quantity + ? - ? where product\_id = ?";  
 *PreparedStatement* preparedStatement1 = connection.prepareStatement(stmt2);  
 preparedStatement1.setInt(1, order.getQuantity());  
 preparedStatement1.setInt(2, newQuantity);  
 preparedStatement1.setInt(3, order.getProduct\_id());  
 preparedStatement1.executeUpdate();  
  
 }  
  
 */\*\*  
 \* This method deletes an existing order  
 \** ***@param*** *connection - Data Base Connection  
 \** ***@param*** *order - The order to be deleted  
 \* \*/  
 public void* deleteOrder(*Connection* connection, Order order) *throws* SQLException{  
  
 String stmt1 = "Update product set quantity = ? where product\_id = ?";  
 *PreparedStatement* preparedStatement1 = connection.prepareStatement(stmt1);  
 preparedStatement1.setInt(1, order.getQuantity());  
 preparedStatement1.setInt(2, order.getProduct\_id());  
 preparedStatement1.executeUpdate();  
  
 String stmt = "Delete from productOrder where order\_id = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setInt(1, order.getOrder\_id());  
 preparedStatement.executeUpdate();  
 }  
  
 */\*\*  
 \* Returns an order from the data Base based on the id in the table  
 \** ***@param*** *connection - Data Base connection  
 \** ***@param*** *id - The id of the order in the Data Bse  
 \** ***@return*** *The searched order  
 \* \*/  
 public* Order getOrderByID(*Connection* connection, *int* id) *throws* SQLException{  
  
 Order foundOrder = *new* Order();  
  
 String stmt = "Select \* from productOrder where order\_id = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setInt(1, id);  
 *ResultSet* resultSet = preparedStatement.executeQuery();  
  
 *while*(resultSet.next()){  
 foundOrder.setOrder\_id(resultSet.getInt("order\_id"));  
 foundOrder.setProduct\_id(resultSet.getInt("product\_id"));  
 foundOrder.setProduct\_name(resultSet.getString("product\_name"));  
 foundOrder.setClient\_id(resultSet.getInt("client\_id"));  
 foundOrder.setClient\_name(resultSet.getString("client\_name"));  
 foundOrder.setQuantity(resultSet.getInt("quantity"));  
 foundOrder.setTotalSum(resultSet.getDouble("TotalSUM"));  
 }  
  
 *return* foundOrder;  
 }  
  
 */\*\*  
 \* This method creates an order  
 \** ***@param*** *connection - Data Base connection  
 \** ***@param*** *product - The name of the product that's being ordered  
 \** ***@param*** *client - The client that makes the order  
 \** ***@param*** *quantity - The quantity of the product that is being ordered  
 \** ***@param*** *totalSum - The Sum to pay for the order  
 \* \*/  
 public void* placeOrder(*Connection* connection, Product product, Client client, *int* quantity, *double* totalSum) *throws* SQLException{  
  
 String stmt = "INSERT INTO productOrder (product\_id, product\_name, client\_id, client\_name, quantity, totalSum) values (?, ?, ?, ?, ?, ?)";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
  
 preparedStatement.setInt(1, product.getProduct\_id());  
 preparedStatement.setString(2, product.getName());  
 preparedStatement.setInt(3, client.getId());  
 preparedStatement.setString(4, client.getName());  
 preparedStatement.setInt(5, quantity);  
 preparedStatement.setDouble(6, totalSum);  
  
 preparedStatement.executeUpdate();  
  
 String stmt1 = "Update product set quantity = quantity - ? where product\_id = ?";  
 *PreparedStatement* preparedStatement1 = connection.prepareStatement(stmt1);  
 preparedStatement1.setInt(1, quantity);  
 preparedStatement1.setInt(2, product.getProduct\_id());  
 preparedStatement1.executeUpdate();  
 }  
  
 */\*\*  
 \* This method gets an order based on the Client name, Product name, amount to pay and quantity  
 \** ***@param*** *connection - Data Base Connection  
 \** ***@param*** *name - The name of the client that ordered  
 \** ***@param*** *product - The name of the product the client ordered  
 \** ***@param*** *sum - The sum the client has to pay for the order  
 \** ***@param*** *quantity - The quantity ordered by the client  
 \** ***@return*** *The order that satisfies these parameters  
 \* \*/  
 public* Order getExactOrder(*Connection* connection, String name, String product, Double sum, *int* quantity)*throws* SQLException{  
 String stmt = "Select \* from productOrder where client\_name = ? and product\_name = ? and totalSum = ? and quantity = ?";  
 *PreparedStatement* preparedStatement = connection.prepareStatement(stmt);  
 preparedStatement.setString(1, name);  
 preparedStatement.setString(2, product);  
 preparedStatement.setDouble(3, sum);  
 preparedStatement.setInt(4, quantity);  
  
 Order order = *new* Order();  
  
 *ResultSet* resultSet = preparedStatement.executeQuery();  
  
 *while* (resultSet.next()) {  
 order.setClient\_name(resultSet.getString("client\_name"));  
 order.setProduct\_name(resultSet.getString("product\_name"));  
 order.setTotalSum(resultSet.getDouble("totalSum"));  
 order.setQuantity(resultSet.getInt("quantity"));  
 }  
  
 *return* order;  
 }  
}