DOCUMENTATION

ORDERS MANAGEMENT

STUDENT NAME: Mesesan Mara-Irina
GROUP: 30424

CONTENTS

1. Assignment Objective

The objective of the assignment was to make an Order Management using database connection and java. The purpose was to make inseration, deletion and update opeartions using queries and a set of classes so that an user interface can have a connection with an order system databsase.

2. Problem Analysis, Modeling, Scenarios, Use Cases

A. Problem Analysis

The problem requested that an admin user has to make an order entry using data from a client and also a list of products that the client is going to order.

Also, the clients and products have to be inserted by the user and after an order is made a Bill is generated showing the items that were bought and the total price.

3. Design

4. Implementation

Packages used:
A. Connection
A.1 Connection
Makes the connection between the database (DataGrip) and the project.

B.dao

B.1 AbstractDAO

Is a generic abstract Data Access Object (DAO) class . The class is generic and can work with any type T. The class contains methods for executing SQL queries, retrieving data, and mapping it to objects of type T. The class also includes helper methods to create specific types of SQL queries, such as createSelectQuery(), createSelectAllQuery(), createInsertQuery(), createUpdateQuery(), createDeleteQuery(), returnIdByNameQuery(), and returnNameByIdQuery()

B.2 ClientDAO

Represents a Data Access Object (DAO) class named ClientDAO, which is responsible for managing Client entities in a database. The ClientDAO class extends the AbstractDAO<Client> class, indicating that it inherits some common functionality for data access.

This DAO class provides methods for finding and updating client entities in the database using different criteria. It utilizes the ConnectionFactory class to establish and close database connections.

B.3 BillDAO

This class is responsible for interacting with the database to perform operations related to bills. The totalPrice method calculates the total price of an order based on the given order ID. It executes a SQL SELECT query that joins the productinorder, product, and order tables to retrieve the quantity, price, and order information. The total price is calculated by multiplying the quantity and price of each product in the order and summing them up. Overall, this DAO class provides methods for calculating the total price of an order and retrieving the price of a bill from the database. It helps in managing bill-related operations within the application.

B.4 ProductDAO

This class is responsible for interacting with the database to perform operations related to products. The findByName method retrieves a product by its name. The updateByName method updates a product by its name. It executes a SQL UPDATE query using the createUpdateQuery method inherited from the AbstractDAO class

The decrementStock method updates the stock quantity of a product by subtracting a given decrement value. It executes a SQL UPDATE query to decrement the stock value of the specified product.

The checkStock method retrieves the current stock quantity of a product based on its ID. It executes a SQL SELECT query and returns the stock quantity.

B.5 OrderDAO

This class is responsible for managing Order entities in the database. The createAddClientQuery method creates an SQL INSERT query for adding a client to an order. It returns the query as a string. The insertClient method inserts a client into an order. It executes an SQL INSERT query using the createAddClientQuery method to create the query. The method retrieves the new ID for the order using the setId method and sets the ID and client ID in the prepared statement.

The returnOrderId method returns the ID of the last order associated with a client. It executes an SQL SELECT query to retrieve the maximum ID from the "order" table where the client ID matches the provided ID. The method returns the retrieved ID value.

B.6 ProductInOrderDAO

It Represents a Data Access Object (DAO) class named ProductInOrderDAO. This class is responsible for managing ProductInOrder entities in the database.

C. bll

C.1 ClientBLL

A Business Logic Layer (BLL) class named ClientBLL. This class acts as an intermediary between the presentation layer (UI) and the data access layer (DAO) for performing operations related to the Client entity. The ClientBLL class has a dependency on the ClientDAO class, which is used to interact with the database he class encapsulates the operations related to the Client entity and interacts with the ClientDAO to perform database operations.

C.2 OrderBLL

The OrderBLL class has a dependency on the OrderDAO

The addClient method is used to insert a client into an order. It calls the insertClient method of the orderDAO object, passing the client ID as a parameter.

C.3 ProductBLL

C.4 BillBLL

C.5 ProductInOrderBLL

D. model

The classes for each table used

D.1 Client

D.2 Bill

D.3 Order

D.5 Product

D.6 ProductInOrder

The connection between order and product class having only foreign keys

E. view

The frames for each user interface that holdes specific actions for each table

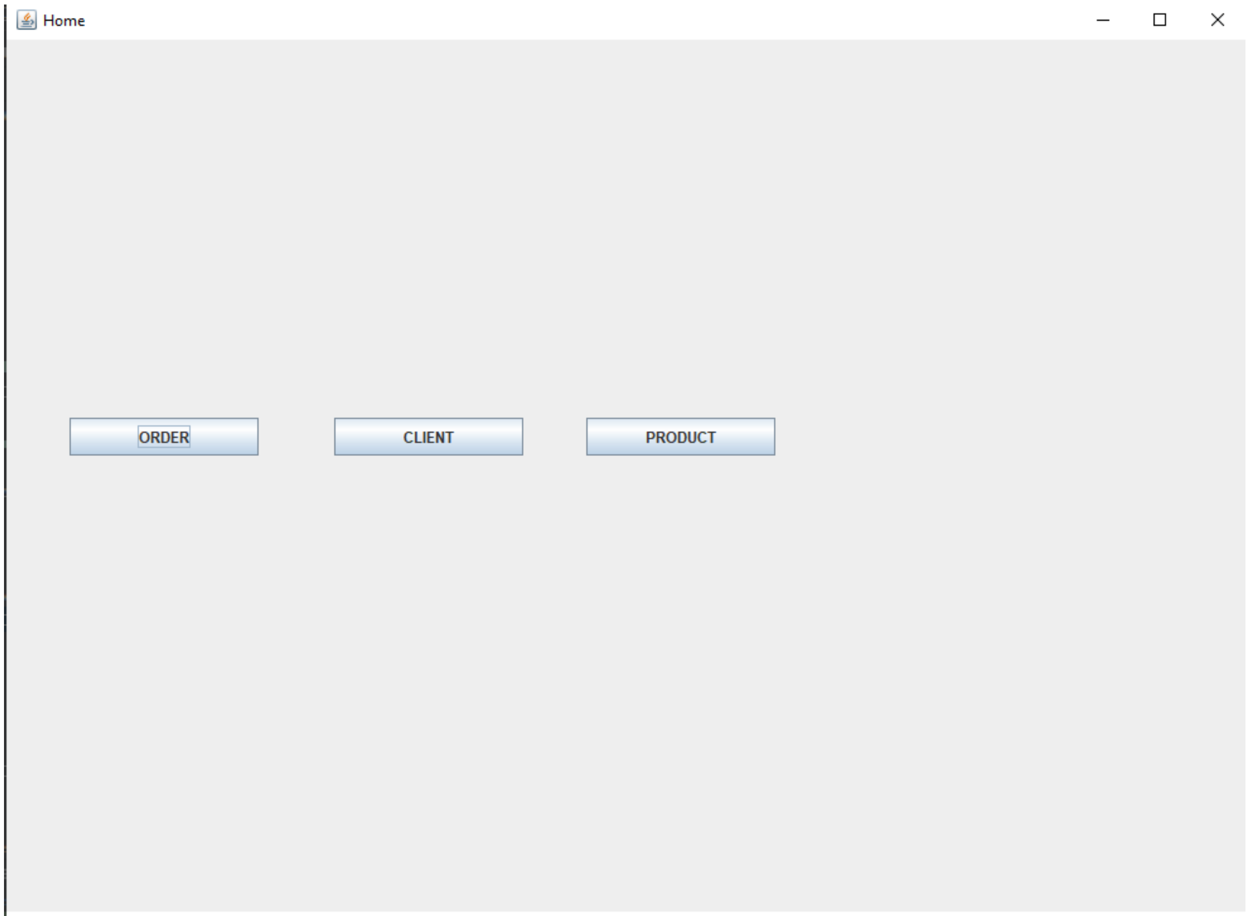
E.1 MainPage

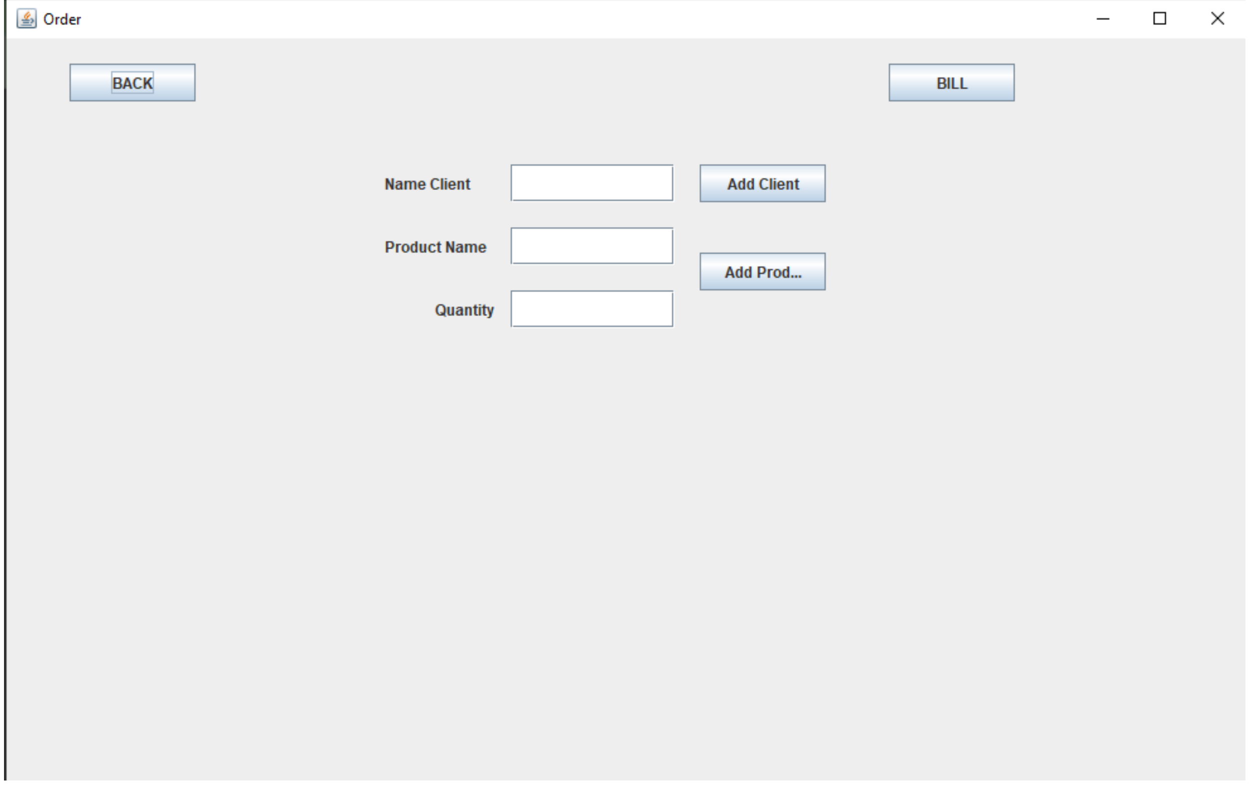
The provided code represents a graphical user interface (GUI) class named MainPage. It creates a window with buttons for order, client, and product functionalities.

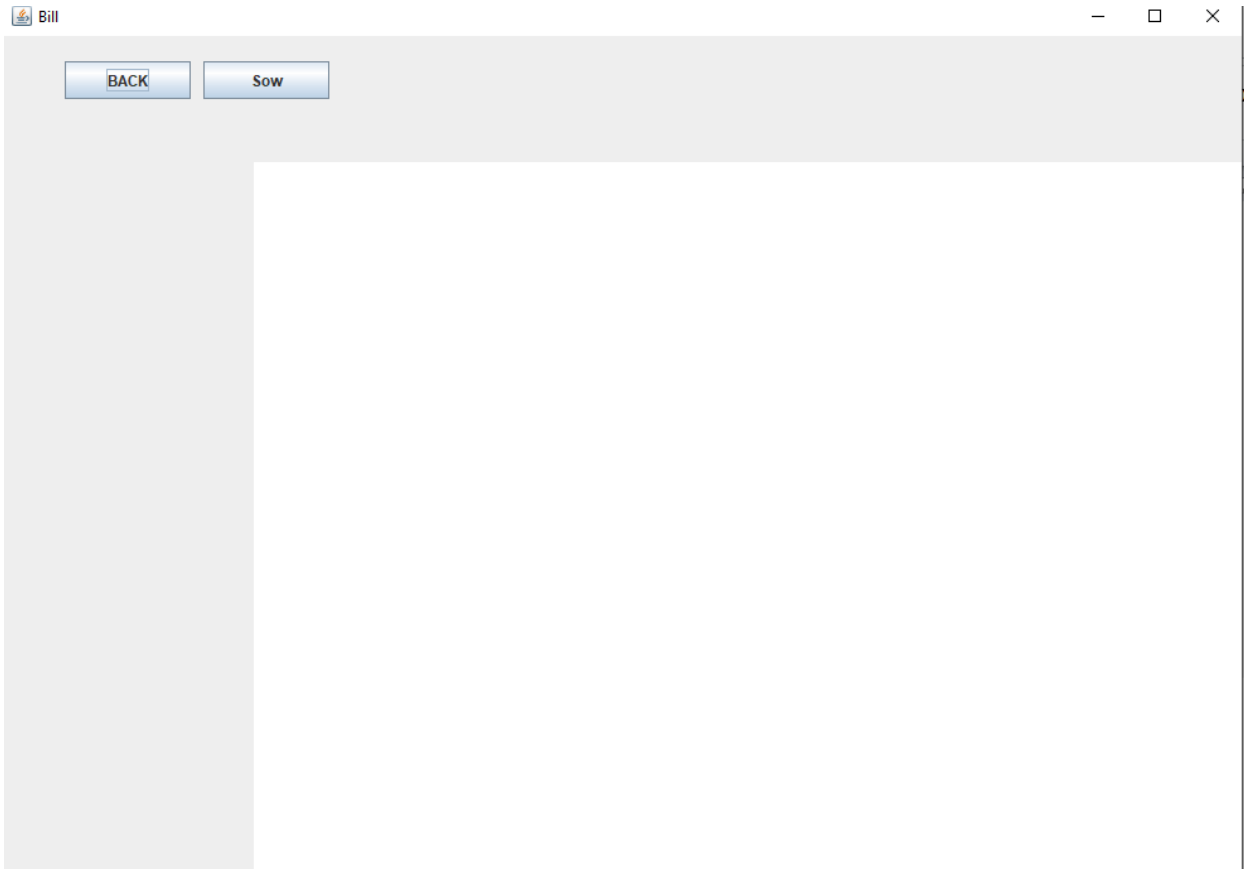
F. Controller

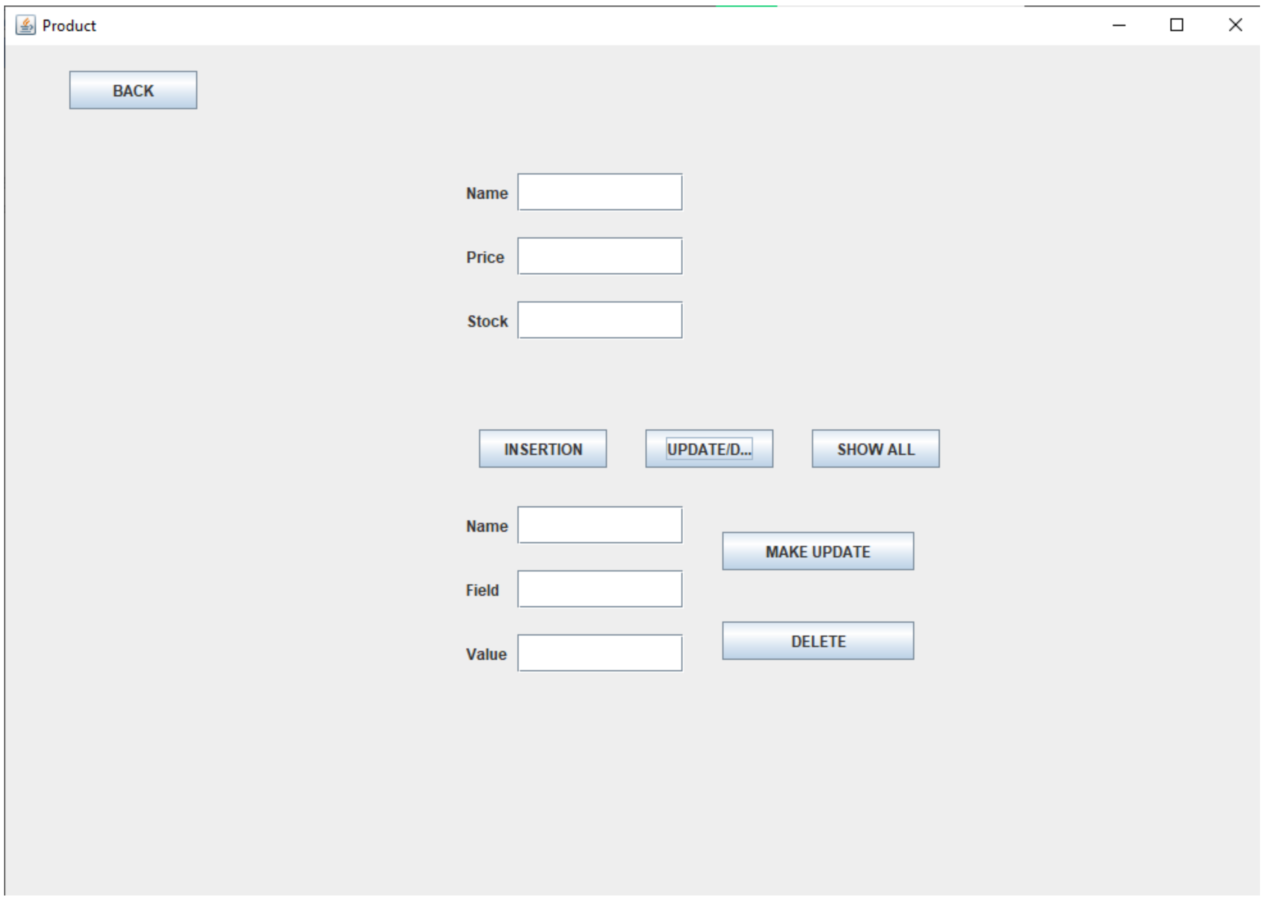
G. Model

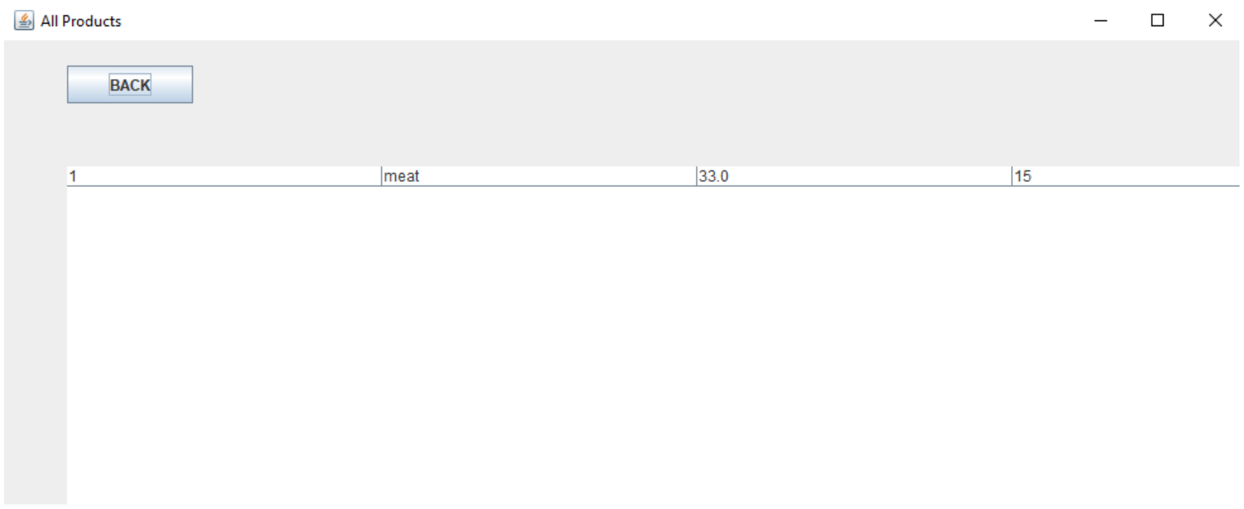
5. Results


Home
ORDER
CLIENT
PRODUCT

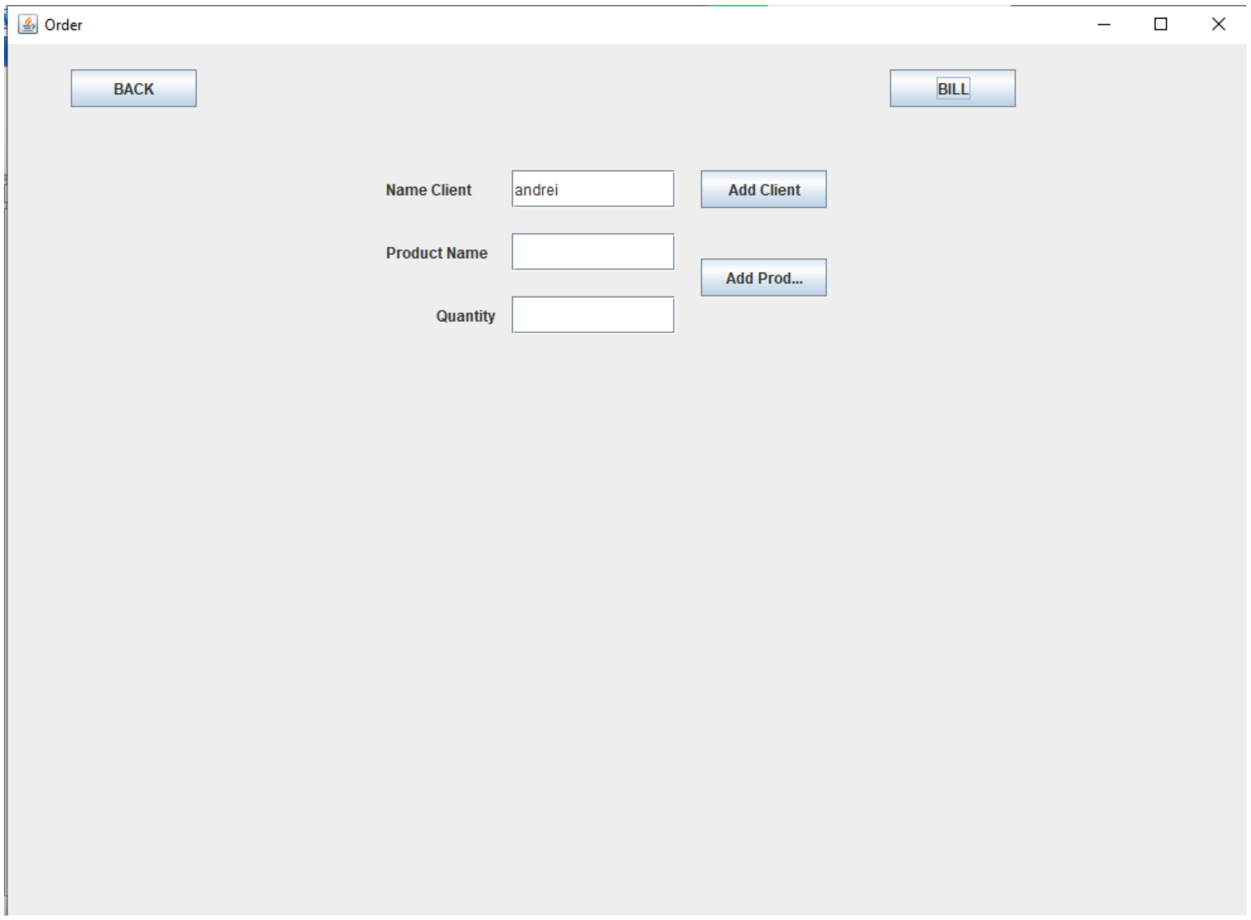

Order
BACK
BILL
Name Client
Add Client
Product Name
Add Prod...
Quantity


Bill
BACK
Sow


Product
BACK
Name
Price
Stock
INSERTION
UPDATE/D...
SHOW ALL
Name
Field
Value
MAKE UPDATE
DELETE


All Products
BACK

1	meat	33.0	15



6. Conclusions

Improved working with database in a more organized way. For future developments I want to make sure that a lot of exceptions are followed regarding the way the products and clients are added.