Sabou Cezara
Computer Science, year 2
group 30421

Order Management System

1.Objectives

1.1 Main objective

The main objective of this project is to create a Java Application that could provide the user the ability to generate the simulation of purchasing an item and obtaining a bill after that. The clients and products are kept in tables. The user should only select the client, or update it (in case he/she is that client), or delete it in case the user does not want to be part of the table; after that the user should be able to press the place order button, in which it can select the wanted product, and insert the number of those items. If the number of items is less than what the manager has in stock, then the purchase is made successfully and the bill is printed. If not, an error message should appear, telling the customer that not enough items are available.

1.2 Secondary objectives

The secondary objectives derive from the main objective:

1. The ability to interact with multiple tables;
2. Handling the user input;
3. Successfully gather data and print the bill in the text file.
4. Implementing a GUI(Graphical User Interface)

2. Problem Analysis

Order management is used in many different branches, but the most important one should be the one in which the user interacts directly with the order. Therefore, the best option would be to create a GUI that not only is easy to use, but also offers correctness and precision.

I will describe the functionality of the system in the following paragraphs.

The system is given a set of database tables, which can be populated or not. There are three types of data we are interested in when it comes to order management : the client, the product and the order.

These tables are kept in a database and they are later converted into lists, in order to easily access them.

The clients present in the client table, can be altered by the user in the few following ways :

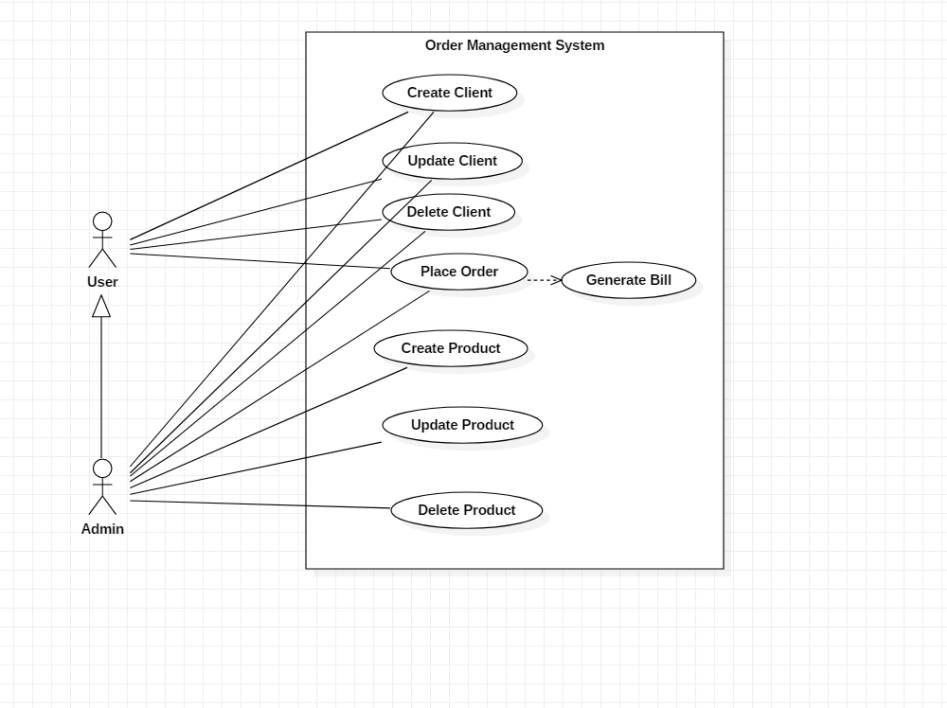
- Create a new client- with new information;
- Read data – simply get the most recent version of the table;
- Update a client – change information about existing client;

- Delete a client – remove user from table;

The products in the product table can also be altered in the same ways , but only by the manager of the warehouse, as he is the only one that can restock the products:

- Create a new product
- Read/see all the products in a table
- Update a product – change its price, the number of items in stock, or the description
- Delete a product – in case that product will not be selling anylonger.

The orders are placed whenever the user selects an existing client, an existing order and a correct number of items is inserted. If the order was successfully made, the bill will be printed.



The use case diagram presents :

Use case title: Place Order

- Summary: this case allows the user to place an order by selecting an order and a product
- Actor : User
- Preconditions : The item to be ordered existis in the stock
- Main success scenario:
  - User presses the Submit Order Button;
  - The Application opens the window with the clients and products;
  - The user selects an existing client;

- The user selects an existing product;
- The user inserts the desired quantity
- The user presses the Submit Order Button
- The System checks if the quantity is below the existing number of items in stock
- The order is successful and the bill is printed into a text file, specifying the date, time, name of client and the name of the purchased product;

Alternative sequences:

- The client was not selected
  - The application tells the user that they should select an existing client
- The product was not selected
  - The application tells the user that they should select an existing product to buy
- The number of items is greater than the existing items in stock
  - The application tells the user that the number of items is higher than the existing available stock number

Use Case Title : Create Clients

Summary : this use case allows the user to handle the client list(sign in with their information)

Actors : User

Preconditions : The user opens the application

Main success scenario :

- The user presses the see clients button
- The client window opens
- The client list and the form appears
- The user creates a new client using the form
- The user presses the create client button
- The client is successfully created

Use Case Title :Update Clients

Summary : this use case allows the user to handle the client list(sign in with their information)

Actors : User

Preconditions : The user opens the application

Main success scenario :

- The user presses the see clients button
- The client window opens
- The client list and the form appears
- The user selects an existing client
- The user modifies the data shown in the form
- The user presses the update client button
- The client is susccessfully created

Sabou Cezara
Computer Science, year 2
group 30421

Use Case Title :Delete Clients

Summary : this use case allows the user to handle the client list(sign in with their information)

Actors : User

Preconditions : The user opens the application

Main success scenario :

- The user presses the see clients button
- The client window opens
- The client list and the form appears
- The user selects an existing client
- The user presses the delete client button
- The client is successfully deleted

Alternative sequences : the data is not validated

Use Case Title : Restock

Summary : this use case allows the manager to handle the products

Actors : Warehouse Manager/ Admin

Preconditions : the admin opens the application

Main success scenario :

- The Admin presses the see products button
- The admin inserts the password
- The admin presses the submit password button
- The password is checked and validated
- The product window appears
- The admin creates/ updates/ deletes a product
- The product is correctly handled

Alternative sequences :

- The password was incorrect :
  - The application tells the admin that the password was not valid

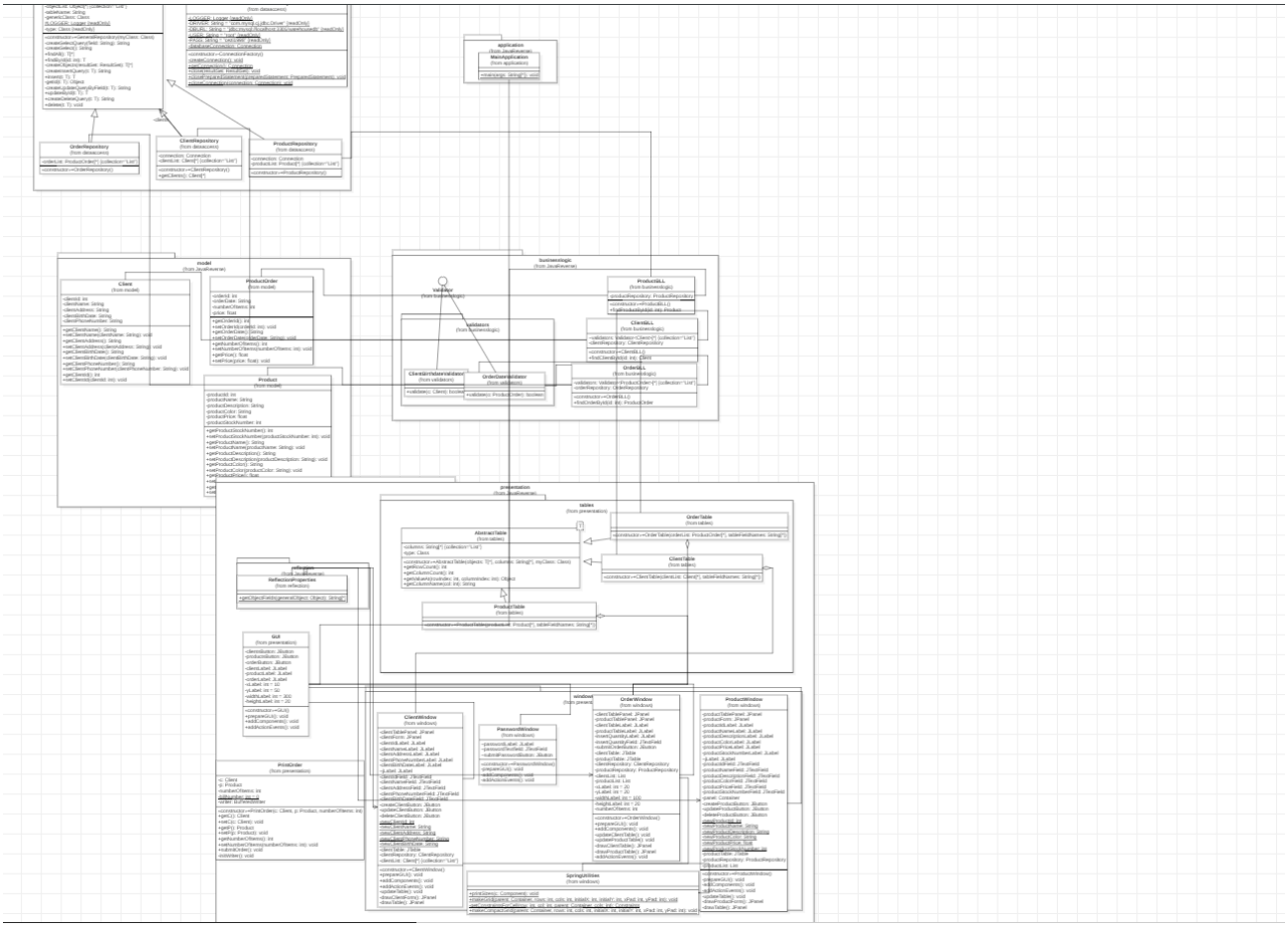
3.Design

The structure of the application is based on 6 different packages, based on a layered architecture

1. The ”model” package is the one that contains the following classes: Client, Product, Order .
2. The “dataacess” package is responsible for taking the data from the database and convert it to manageable information
3. The “businessLogicLevel” package links the model and the presentation, managing the interactions.
4. The “presentation” package is used to display data and also receive data;

5. The “reflection” package is meant to provide helpful classes and methods for obtaining data from unknown objects.
6. The “application” package simply starts the application

I created this UML diagram in order to show the main relationships of the application and to accentuate the layered architecture :



4.Implementation

In the Model package:

4.1 Client

```
The “Client” class is used to create all the clients that will be taken from the
database and keep the client structure with all its attributes :
private int clientId;
private String clientName;
private String clientAddress;
```

```
private String clientBirthDate;
private String clientPhoneNumber;
```

The class also holds the corresponding getters and setters.

4.2 Product

The “Product” class holds the structure of the products displayed in the tables and is used to keep the data obtained from the database. It has the same names as the column names from the database. The attributes are :

```
private int productId;
private String productName;
private String productDescription;
private String productColor;
private float productPrice;
private int productStockNumber;
```

4.3 ProductOrder

The Product order class is used for keeping track of the orders, and associating them with a client and a product.

Whenever a product is placed it uses the data from here, or it might set other data.

In the businessLogic package

4.4 Validators

The validators are created in order to verify the input given in the forms presented in the interface. They inherit the interface Validator which has only one method : validate().

4.5 BLL Classes(ClientBLL, OrderBLL, ProductBLL)

The business logic layer links the data access with the presentation, and only holds a few methods and it is used to create a new repository through which data is taken form the database and validate the input.

In the dataaccess package

4.6 General Repository

The general repository is by far the most complex one. It is a generic type class, which we need because we do not know which table the user is going to select.

The General Repository holds all the important methods, such as : findAll() which goes trough all elements from a table and returns them as an arrayList, the queries : Create, Update and Delete, which are done by first creating the query in a separate method and then connecting to the database and accessing the elements and the fields inside them. Moreover, the findall() method is using the createObjects method, which is actually the one that takes the values from the database and by reflection it obtains the fields, creates an in instance with the attributes found, and the inserts the instance into an arraylist; The getId() method takes an object and returns its id as an integer value.

The operations are done through reflection, the class is generic, and by receiving an object it will obtain its fields through reflection, therefore being able to recreate that product and access the data in the database.

Sabou Cezara
Computer Science, year 2
group 30421

The queries look like this one :

```
private String createSelectQuery(String field) {
    StringBuilder sb = new StringBuilder();
    sb.append("SELECT ");
    sb.append(" * ");
    sb.append(" FROM ");
    sb.append(type.getSimpleName());
    sb.append(" WHERE " + field + " =?");
    return sb.toString();
}
```

4.7 Deriving repositories

These classes : Client Repository, Order Repository and Product Repository are simply derived classes using the generic one, and they are created to give the class type to the generic class.

4.8 Connection Factory

The Connection Factory is responsible with creating the connection to the actual database, and it holds all the information aobut the driver, the user, the password and the dburl. The connection can be accessed by using the getConnection() method, and also the connection factory is important because it holds the methods that close the prepared statements used in the General Repository.

In the presentation package there is another package which holds all the tables:

- The most important one would be the Abstract Table because the other tables depend on it; it is a generic type of class and it receives a list of egenric objects and the class type, in order to generate an abstract table frame which can be displayed in the interface.
- The Client table has to generate the client table given from the client list of objects;
- The Product table should generate the Product table obtained from a list of products;
- The order table should do the same : receive a list, create a table from it

There is also the windows package which belongs to the presentation package too , and it holds the following classes :

- The Client Window, which is composed of a jtable created dynamically on the right side, and on the left side a form in which the user can insert information and the press the buttons under the form in order for it to be processed.
- The Product Window is very similar to the Client window;
- The Order window is more interesting because it displays both the clients and the product, and the user can select a product, and then a client, and the insert the number of items that it wants to purchase. An order is then created by using action listeners whenever the submit button Is pressed. The bill is to appear in the text field and it is meant to show a little bit of information about the purchase;
- The Password window is the one that is made out of a simple textfield that receives the password and checks if it is correct, then it will allow access to the Product window, which can actually be modified only by the manager himself;
- The Spring Utilities class is meant to help with the layout of the components of the graphical user interface.

Sabou Cezara
Computer Science, year 2
group 30421

In the reflection package :

4.9 Reflection Properties

There is only one, but very essential class, called ReflectionProperties, which takes an unknown object and returns the fields structure, name and type, all nicely packed into a list of fields.

5.Results

The results show the bill created, which proves that the database information was correctly accessed and obtained through reflection. It shows that the action listeners correctly interpreted the user's actions and it shows that the information has been validated.

6.Conclusions

In conclusion, the application uses OOP concepts and it serves as an Order Management system. A Graphical User Interface is created and easy to use, in order to offer the users an easy access to queue processing and to give back fast and correct results.

By developing this application I gained experience in trying to divide the classes into the layered architecture pattern, understanding its functionalities and obtaining a clearer perspective of the project. I became more familiar with the reflection concepts and also linking the methods designed by me to the front end. I learned how to connect to a database and modify the data through prepared queries.

I am aware that the application is not yet in its final form, and there are some adjustments that need to be done. The input conversion could be improved, as well as the interface. I also believe that there are more efficient ways in which I could implement the user interfae. I also would add other validators, so that the inserted data will be correct.

7.Reference

https://www.tutorialspoint.com/java/index.htm https://www.jetbrains.com/help/idea/maven-support.html

https://examples.javacodegeeks.com/desktop-java/swing/java-swing-layout-example/

https://docs.oracle.com/javase/tutorial/uiswing/events/intro.html

https://bitbucket.org/utcn_dsrl/pt-reflection-example/src/4fa035e84c34?at=master