Programming Techniques

Assignment 2

Student : Rujita Alexandra

Group : 30422

Contents

[1. Problem specification. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . .. . . . .3](#_Toc447019469)

[2. Problem analysis, modeling, scenarios, use cases . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . 3](#_Toc447019470)

[2.1. Problem analysis 3](#_Toc447019471)

[2.2. Modeling 3](#_Toc447019472)

[2.3. Scenario and Use cases 4](#_Toc447019473)

[3. Design 5](#_Toc447019474)

[3.1. UML Diagram ( Class Diagram ) 5](#_Toc447019475)

[3.1.1 Customer Class 5](#_Toc447019476)

[3.1.2 Product Class 5](#_Toc447019477)

[3.1.3 Order Class 6](#_Toc447019478)

[3.1.4. CustomerBLL Class 6](#_Toc447019479)

[3.1.5 ProductBLL Class 6](#_Toc447019480)

[3.1.6 OrderBLL Class 6](#_Toc447019481)

[3.1.7. WarehouseBLL Class 6](#_Toc447019482)

[3.1.8 CustomerAccess Class 6](#_Toc447019483)

[3.1.9 OrderAccess Class 7](#_Toc447019484)

[3.1.10 ProductAccess Class 7](#_Toc447019485)

[3.1.11 WarehouseAccess Class 7](#_Toc447019486)

[3.1.12 Controller Class 7](#_Toc447019487)

[3.1.13 GUILogin 7](#_Toc447019488)

[3.1.14 GUIAdmin 7](#_Toc447019489)

[3.1.15 GUIUser 8](#_Toc447019490)

[3.1.16 mainC 8](#_Toc447019491)

[3.2. Package Diagram 8](#_Toc447019492)

[3.3. Data structures 8](#_Toc447019493)

[3.4. Relationships 9](#_Toc447019494)

[3.5. Architecture 9](#_Toc447019495)

[3.6. Algorithms 9](#_Toc447019496)

[3.7. User Interface 10](#_Toc447019497)

[4. Implementation 12](#_Toc447019498)

[5. Further developments and importance 14](#_Toc447019499)

1. Problem specification

Consider an application OrderManagement for processing customer orders. The application uses (minimally) the following classes: Order , OPDept (Order Processing Department) , Customer , Product, and Warehouse. The classes OPDept and Warehouse use a BinarySearchTree for storing orders.

The program will be able to perform the following operations:

* Add new customer
* Update existing customer address
* Delete exiting customer
* Place an order
* Delete an order
* Update an order
* Add new products
* Update exiting products
* Delete existing products
* Modify the content of warehouse according to the submitted orders

It will have a graphical user interface: first a frame corresponding to the login will appear, then according to the logged user ( admin or customer) another frame will appear in which the user can order products or update its account, or if the admin logged in the frame corresponding to the products access will appear. In both cases a table will appear containing information about the current order or the current sold products.

1. Problem analysis, modeling, scenarios, use cases

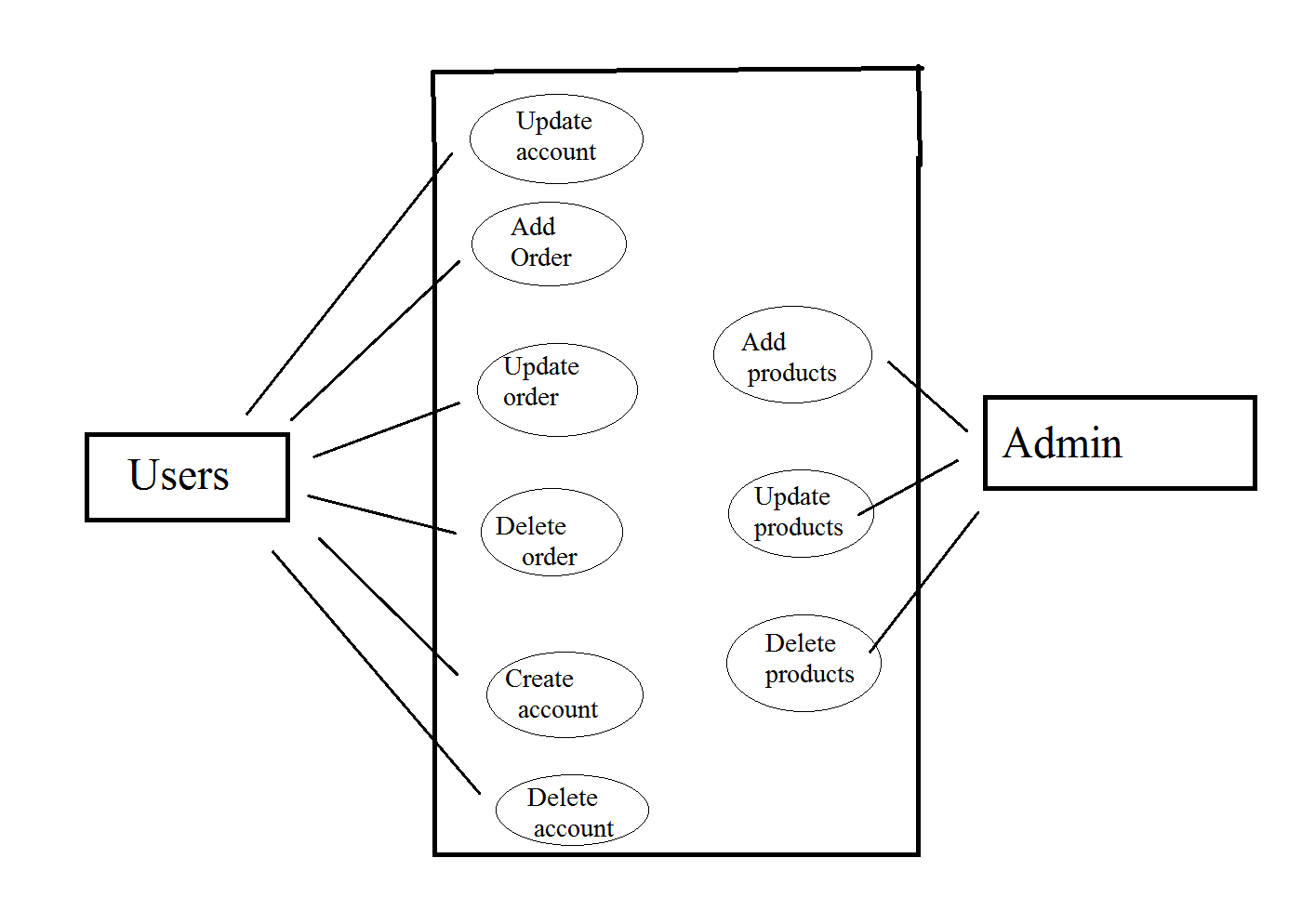
## Problem analysis

An order application can be modeled in several ways, it can store the available products in a database or in a BinarySearchTree, respectivley in a seriazible text file. The most efficient way is to work with a database, because you can store as many products you want and you will always have access to them. The downside is the fact that you have to know or learn the SQL query language ( syntax) . Beside the storing of the products you also have to think about the storing of the customer details, which is also easier to do when you have a database.

## Modeling

The implementation of this project is done by storing all the data ( the customer details and the products ) in a databse which has tables for each component that builds up the store concept. All the tables in the database have a corresponding model in the Java project, that are at the base of this implementation. The access to the database is done in separate classes, through the SQL query syntax.

## Scenario and Use cases

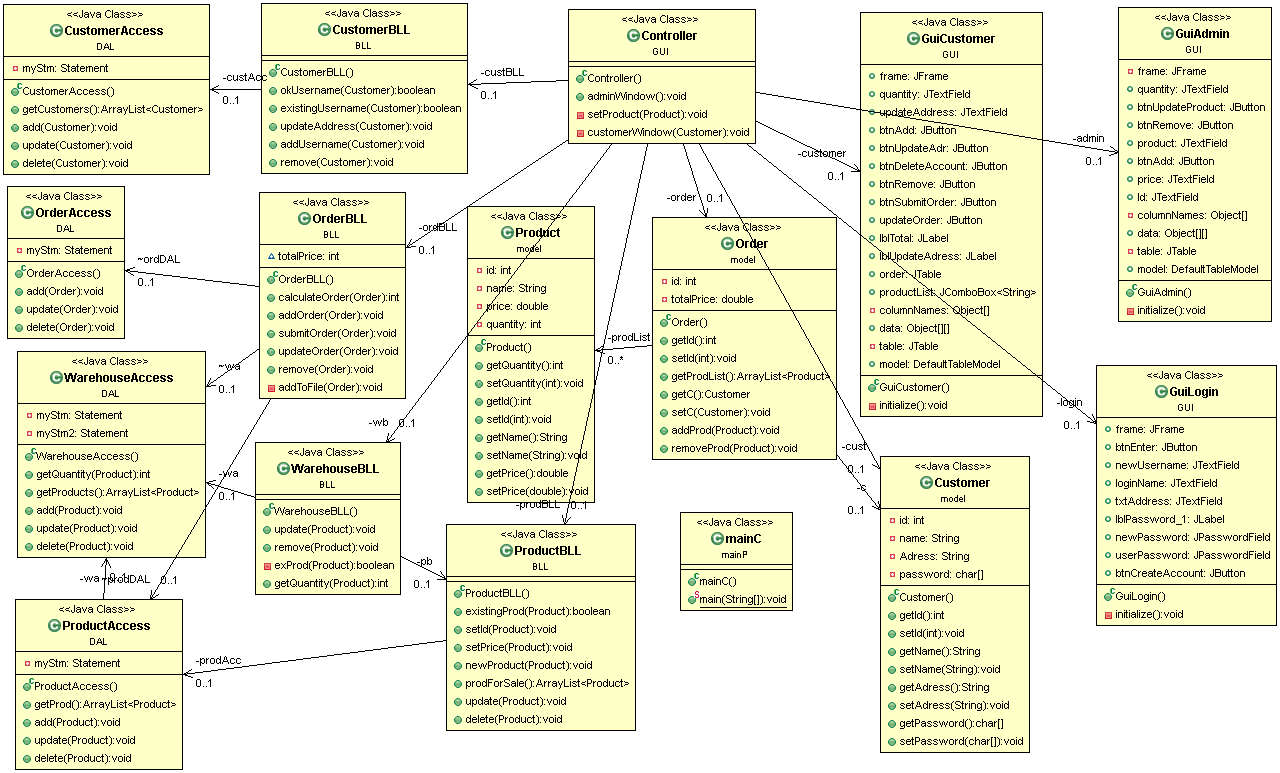


The users have access only to what personally concerns them. So they can update ( edit) the current order, they can add (submit ) a new order when they are sure about what they want to order, or they can delete the order. All the orders for one customer have the same ID as the customerId so it is easier to know who ordered what. After they click the submit button a message box will appear that will inform them what their order ID is.

The admin has the power to add new products by specifying the name, price and quantity, the id is not necessary because it will be automatically incremented. He or she can update an existing product quantity or can delete a certain product, but he or she has to specify the product id. The product id can be known by looking in the table next to the admin frame.

1. Design

## UML Diagram ( Class Diagram )

The UML diagram is a class diagram that shows a set of classes, interfaces and collaborations and their relationships. Class diagrams may also contain packages or sub-systems, both of which are used to group elements of the model.

### 3.1.1 Customer Class

The customer class has the details that describe a customer: the id , name , address and password which are all private, but you can get information about a certain customer by calling the get function or you can modify an information by calling the set function.

### 3.1.2 Product Class

The product class has details about the products such as id, name, price and quantity which you can access by calling the getters or setters for each one.

### 3.1.3 Order Class

The order class has the id , and the getter and setter for it, the total price and an array of products , so all the products for one order are stored togheter. It also has an add and remove method for modifying the array according to the customer needs.

### 3.1.4. CustomerBLL Class

This class is the layer between the user interface and the direct database access. It holds methods for preparing the customer to „enter” the database, it verifies if the user exists so you cannot create an username twice, or if you are an old customer it verifies if you really exist in the database. In essence it does all the necessary preparations to be sure that no error will occur when you add the customer details into the database.

### 3.1.5 ProductBLL Class

This class is the layer between the user interface and the direct database access. It holds methods for preparing the product to „enter” the database, it verifies if the product exists and if so how many are still available, it sets the id to the product so you can do certain updates. It also returns all the products that are for sale no matter how many are still in stock. In essence it does all the necessary preparations to be sure that no error will occur when you want to access the database.

### 3.1.6 OrderBLL Class

This class is the layer between the user interface and the direct database access. It holds methods for preparing the order to „enter” the database, it verifies if the products exists and if so how many are still available, it sets the id to the product so you can do certain updates. It also returns all the products that are for sale no matter how many are still in stock. In essence it does all the necessary preparations to be sure that no error will occur when you want to access the database.

It also calculates the total price for that order and after the „Submit” button is pressed it writes the current order into a text file.

### 3.1.7. WarehouseBLL Class

This class is the layer between the user interface and the direct database access. It verifies and return the value corresponfing to the number of available products. It also modifies the quantity of products after the admin access or after one order was submitted.

### 3.1.8 CustomerAccess Class

This class is responsible to the direct access to the database. It holds methods to all the operations that can be done to the customer table, such as update the address, delete the customer or insert a new customer. When creating a new customer it also sets to it the corresponding id from the database so it can perform further operations with it.

### 3.1.9 OrderAccess Class

This class is responsible to the direct access to the database. It holds methods to all the operations that can be done to the order table, such as: delete the order, insert a new order or update an already existing order. When creating a new order it also sets to it the corresponding id from the database so it can perform further operations with it.

### 3.1.10 ProductAccess Class

This class is responsible to the direct access to the database. It holds methods to all the operations that can be done to the product table, such as delete the product or insert a new product. When creating a new product it also sets to it the corresponding id from the database so it can perform further operations with it.

### 3.1.11 WarehouseAccess Class

This class is responsible to the direct access to the database. It holds methods to all the operations that can be done to the warehouse table, such as delete the product or insert a new product. When creating a new product it also sets to it the corresponding id from the database so it can perform further operations with it. You can also update the quantity corresponding to a certain product.

### 3.1.12 Controller Class

This is the main class for user interface operations. It holds all the actionListener for all the buttons or text fields. From this class “signals” are transmitted to the BLL classes so operations are performed on the database. Also here are done certain verifications such as: some filed is empty or the introduced value does not have the appropriate format. Also error or success messages are displayed when something goes wrong or right, when updating the address a pop-up message will appear informing the user what the new address is, or when not sufficient products are available it will inform the user how many he or she can order. This class updates the content of the tables that are displayed and the label corresponding to the total price of the current products that the customer has in basket.

It also controls which frame is available to whom. First the login frame appears then according to the logged user it will appear the user frame or the admin frame. If the username is not correctly introduced then a message error will appear and you will have another chance to introduce your username.

### 3.1.13 GUILogin

This class creates the user interface corresponding to the login phase of the store.

### 3.1.14 GUIAdmin

This class creates the user interface for the admin, which has the table of available products and how many are still in stock and the text fields for adding or updating the products.

### 3.1.15 GUIUser

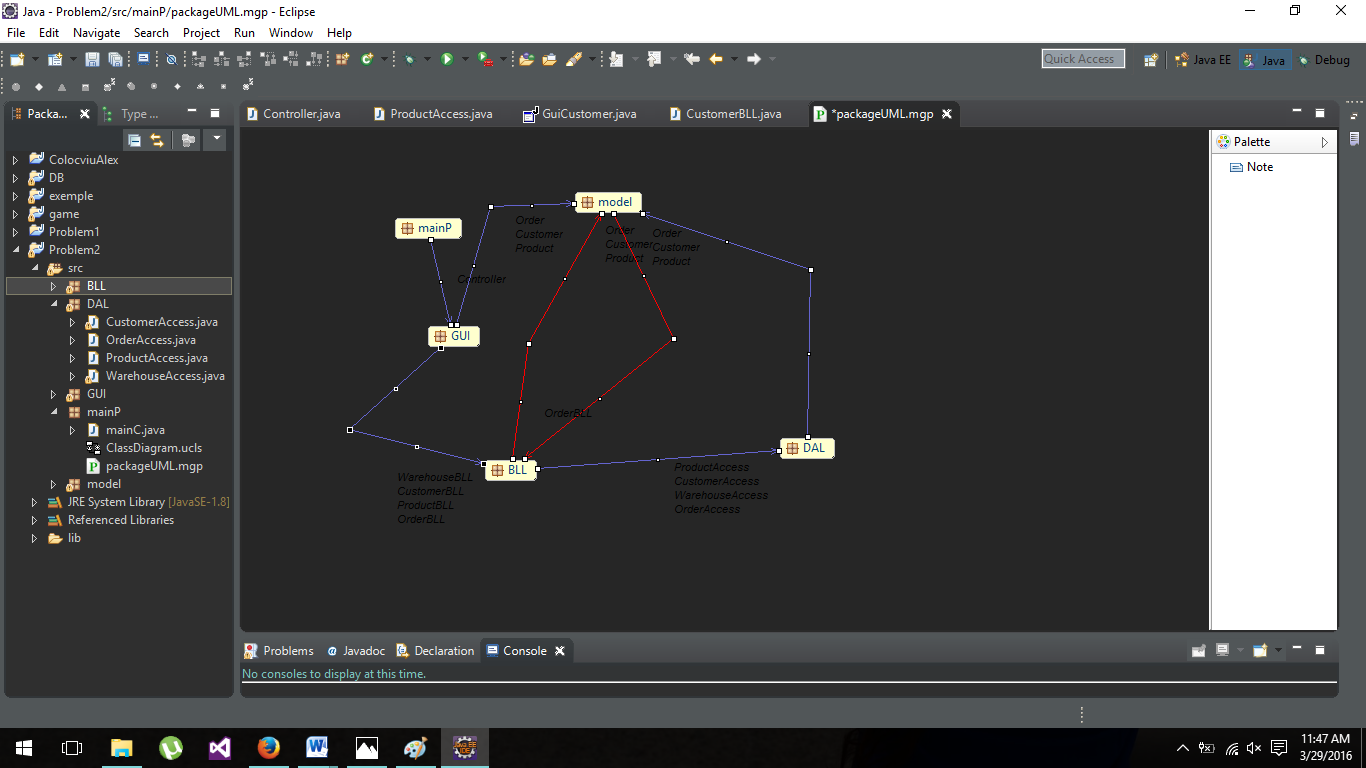
This class creates the user interface for the customer, , which has the table of currently ordered products , but it cannot remove a row when the remove button is pressed even if the product is subtracted from the total.

### 3.1.16 mainC

This class starts the program by calling the Controller class

## Package Diagram

Package Dependencies Diagram shows the dependencies between packages and highlights circular dependencies. Each dependency line also displays the involved classes.



## Data structures

The base data structures in this project are the Customer, Product and Order. With this daat structures we send information from the user interface to the database and back and no direct access is done. Every class depends on them to do the necessary operations.

The order object also has an arrayList of products so if one customer buys more products they are stored togheter and have the same order ID .

## Relationships

There are strong relationships between the Customer class and the CustomerBLL and CustomerAccess class, Product and the ProductBLL and ProductAccess class, Order class and OrderBLL and OrderAccess class.

Also there is a association relationship between the Controller class and the OrderBLL, ProductBLL and WarehouseBLL , the program will still somehow work but no updates will be done on the database. There is a strong dependecy relationship with the CustomerBLL class because if you do not pass this verification you cannot enter the buying / editing section.

Between the Controller and all the graphic user interface classes there is also a strong dependency but only in this direction.

## Architecture

The program uses the business logic layer architecture.

Business logic is the programming that manages communication between an end user interface and a database. The main components of business logic are [business rules](http://whatis.techtarget.com/definition/business-rule) and [workflows](http://searchcio.techtarget.com/definition/workflow). A business rule describes a specific procedure; a workflow consists of the tasks, procedural steps, required input and output information, and tools needed for each step of that procedure. Business logic describes the sequence of operations associated with data in a database to carry out the business rule.

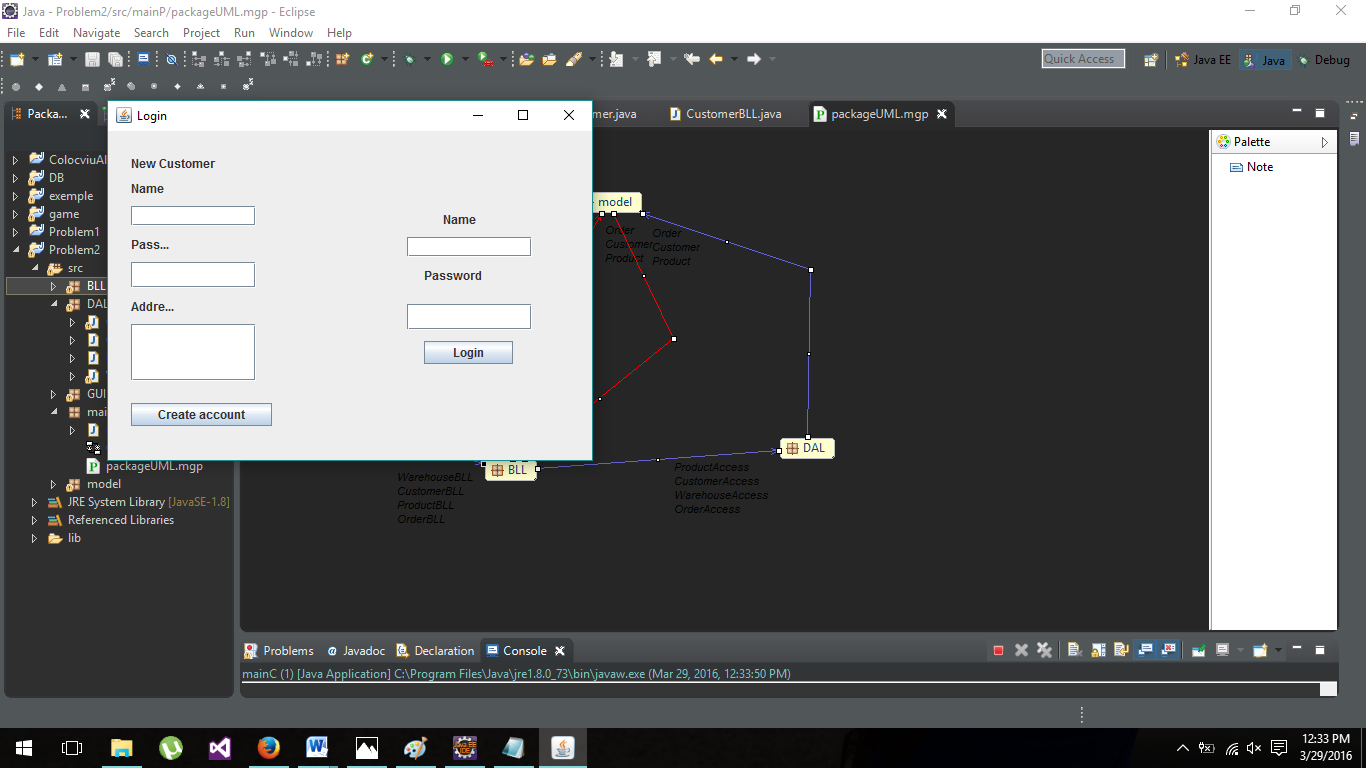
## Algorithms

The login phase can be done in 3 different ways: if the user wants to create a new account then the program will firts make sure that all the fields are completed and then it will verify if there is no such user in the database. If there is the user has to create a new username otherwise the user frame will appear to the user; if there is an old user then the program will verify if the user really exists and if so it will open the user frame, otherwise an error message will appear; if the Admin loggs in then a different frame will appear that allows the admin to make certain modifications and it will display the current products for sale and how many there are in the warehouse, this will be done by accesing the product table and also the warehouse table. All this verifications and accessing will pass throug a layer which will make sure all the objects fields have the corresponding format.

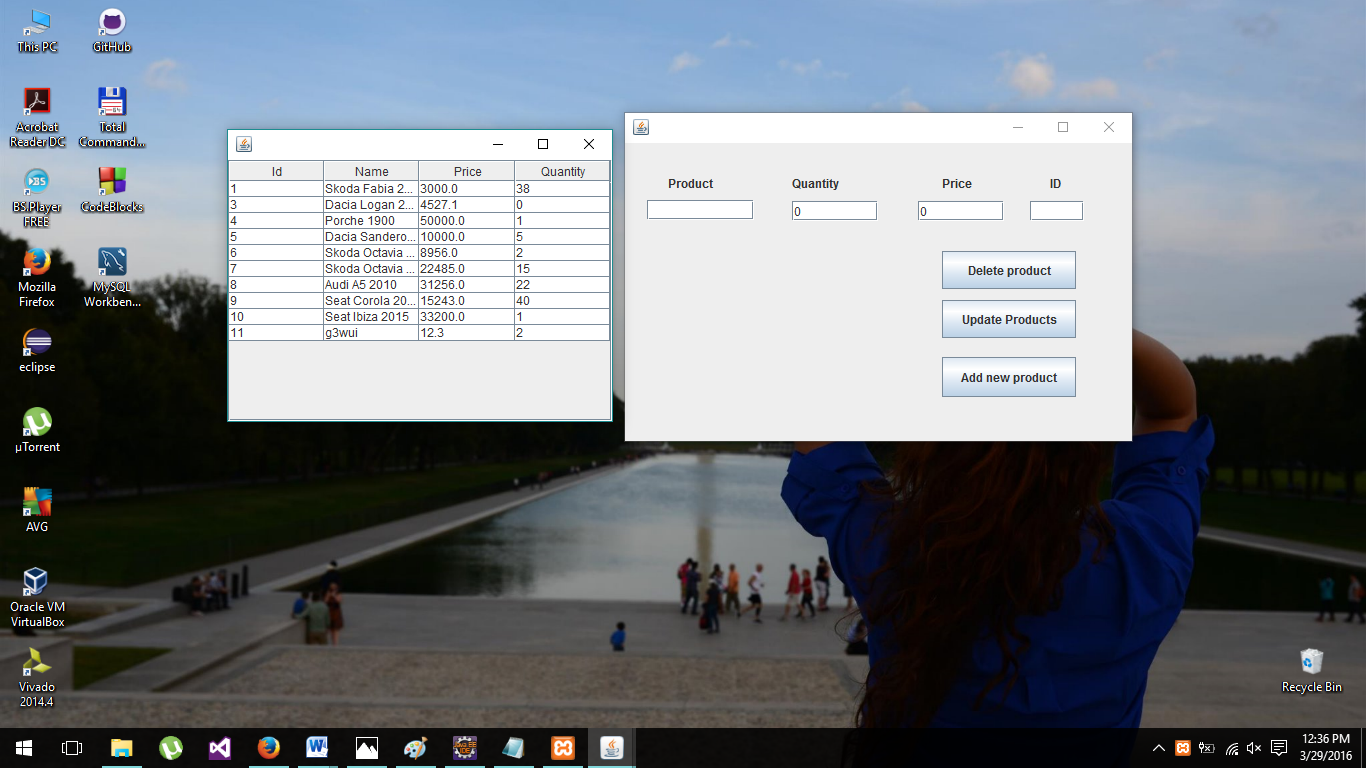
After a certain user is logged in then it can add products by chosing from a drop down list, when selecting the product and entering the quantity and pressing enter the program will search to see if there are so many products available as the user wants if not an message will appear. If there are enough then it will add the product and the quantity to the table, but it will add the order to the order table only when the submit button is pressed. Also when the button is pressed the orderd qunatity is removed from the warehouse table. The program also calculates the total price of the order accorinding to how many products are bought. After the order is submitted a text file containting the name of the customer, the products and its price per piece and the bought quantity is created.

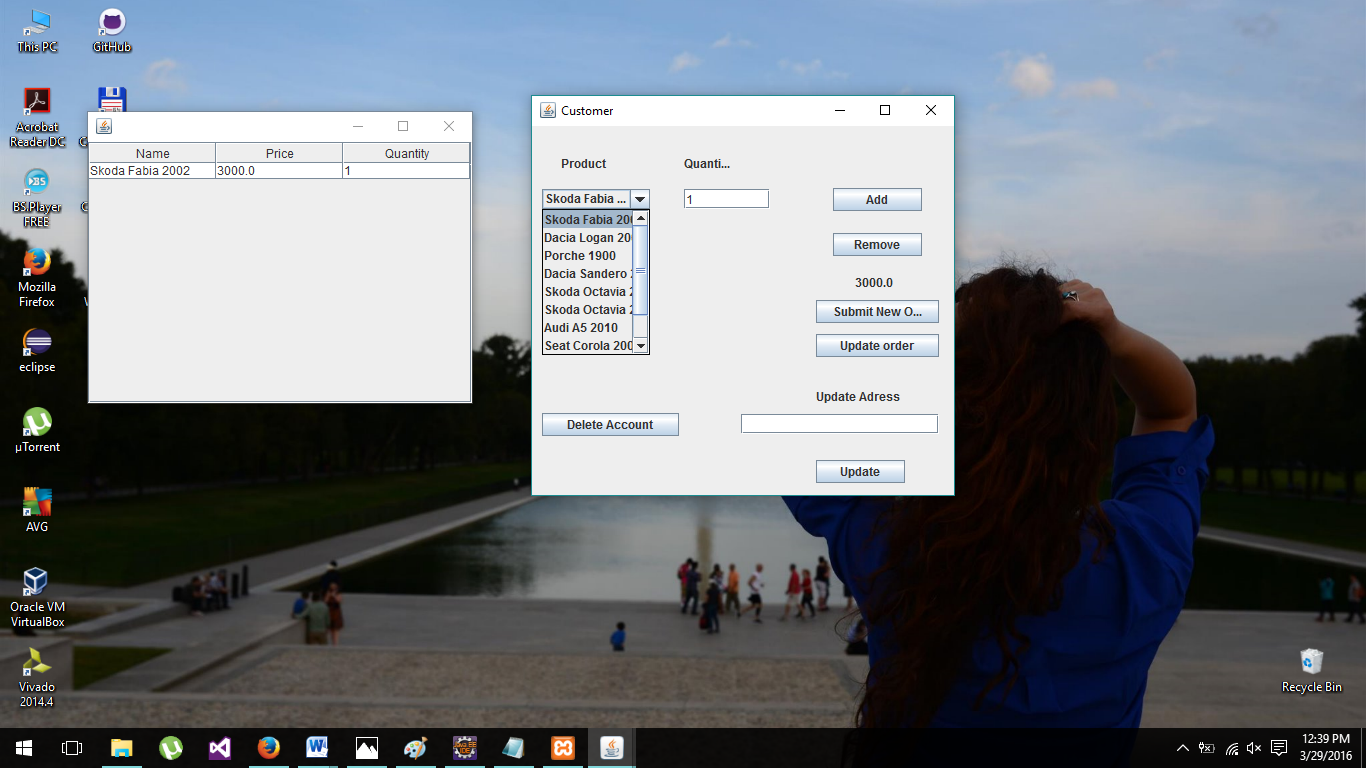
If the admin loggs in then it will appear a table with all the products and if he or she wants to add a new product the program will verify if the product does not exist yet and do the certain settings before transmitting the object to the database access. The same procedure is executed when the admin wants to update or remove a certain product. This operations do not affect only the product table from the database but also the warehouse table.

## User Interface



As it can be seen you can choose to create your account or if you already have one, you can enter it.



This is the admin window. On the right are the operations that can be executed and the corresponding text fields. On the left is the table associated to the database products and the database warehouse ( on the last column ), when adding or removing or updating a product the admin will see the result only on the next login.

The customer window has the drop down list that shows all the products that are for sale in the store , even if they are not available. The total price of the current order is shown in the label under the remove button. On the right side of the screen it is shown what the order contains and how many products were ordered in one step.

1. Implementation

public boolean existingUsername(Customer newCust) {

ArrayList<Customer> existingC = (ArrayList<Customer>) custAcc.getCustomers().clone();

for (Customer c : existingC)

if (c.getName().equals(newCust.getName())){

newCust.setAdress(c.getAdress());

newCust.setId(c.getId());

return true;

}

return false;

This method, from the CustomerBLL class, verifies if an username already exists and if it does it sets to the username the id and the address received from the database so you can do operations with it and also if an order is made to know where to ship it.

private void addToFile(Order order) throws IOException {

File fout = new File("Order.txt");

FileOutputStream fos = new FileOutputStream(fout);

BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(fos));

bw.write("Customer: " + order.getC().getName() + "\n");

for (Product p : order.getProdList()) {

bw.write(p.getName() + " Price/piece: " + p.getPrice() + " Quantity: " + p.getQuantity());

bw.newLine();

}

bw.write("Total price is: "+calculateOrder(order));

bw.close(); }

This method adds the order that was just submitted to an text file, creating an order summary which the customer can print it if he or she wants to .

public CustomerAccess() {

try {

Connection myConn = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/", "alexandra", null);

myStm = (Statement) myConn.createStatement();

} catch (SQLException e) {

e.printStackTrace();

}

This method connects the java project to the mySql server and desired account that manages the store tables.

public ArrayList<Customer> getCustomers() {

Customer c;

ArrayList<Customer> existingCustomer = new ArrayList<Customer>();

ResultSet myRes;

try {

myRes = myStm.executeQuery("Select idcustomer,username,password,address from `test`.`Customer`");

while (myRes.next()) {

c = new Customer();

c.setAdress(myRes.getString("address"));

c.setName(myRes.getString("username"));

c.setId(Integer.valueOf(myRes.getString("idcustomer")));

existingCustomer.add(c);

}

} catch (SQLException e) {

e.printStackTrace();

}

return existingCustomer;

}

This method selects all the customers from the database and sets their corresponding id and address and then stores them in an arrayList which it then trasnmites further to the CustomerBLL to do the wanted operations.

1. Further developments and importance

Working at this project I found out how to work with layered architecture and how important that is. Also I development my way of working with databases and connecting the user with it, without compromising any important data of my store.