Technical University of Cluj-Napoca

Programming techniques

Laboratory assignment 4

A picture containing text, clipart

Description automatically generated

**Managing orders system**

Student: Sicobean Alexandra Maria

Group: 30421

1. **Objective**

The objective of the assignment is to manage and implement an application for managing the food orders for a catering company.

The sub-objectives of the assignment are to:

* analyze the problem
* design the orders management application
* implement the orders management application
* test the orders management application

1. **Analysis, scenarios, use cases**

**Functional requirements:**

* The orders management application should allow the admin to import products from the menu
* The orders management application should allow the admin to add new products to the menu
* The orders management application should allow the admin to delete products from the menu
* The orders management application should allow the admin to modify products from the menu
* The orders management application should allow the admin to generate reports
* The orders management application should allow the client to search for certain products
* The orders management application should allow the client to create a new order
* The orders management application should allow the client to place a new order
* The orders management application should notify the employee each time a new order is placed

**Scenarios and use cases:**

Diagram

Description automatically generated

**Use Case:** sign in

**Primary Actor:** client

**Main User Scenario:**

1. The client inserts the credentials
2. The application validates the credentials and sends the client to the client page

**Use Case:** sign up

**Primary Actor:** client

**Main User Scenario:**

1. If the client does not have an account yet, he/she can create it by clicking on the sign up button
2. The client enters his/her full name, address, username and password
3. A new client is created
4. The client is redirected to the client page

**Use Case:** search through the menu

**Primary Actor:** client

**Main User Scenario:**

1. The client clicks on the display menu button
2. The client enters the searching data
3. The client clicks on the search button

**Use Case:** create a new order

**Primary Actor:** client

**Main User Scenario:**

1. The client clicks on the new order button
2. The client starts entering the name of each product he/she wants to order
3. The client clicks on add new product to confirm the previously entered product

**Use Case:** place an order

**Primary Actor:** client

**Main User Scenario:**

1. The client clicks on the place order button to validate the order.

**Use Case:** import products

**Primary Actor:** admin

**Main User Scenario:**

1. The admin logs in the application
2. The user clicks on the “Import products” button

**Use Case:** add base products

**Primary Actor:** admin

**Main User Scenario:**

1. The admin inserts data for the new product
2. The user clicks on the “Add base product” button

**Use Case:** modify products

**Primary Actor:** admin

**Main User Scenario:**

1. The admin inserts data for the modified product
2. The user clicks on the “Modify product” button

**Use Case:** delete products

**Primary Actor:** admin

**Main User Scenario:**

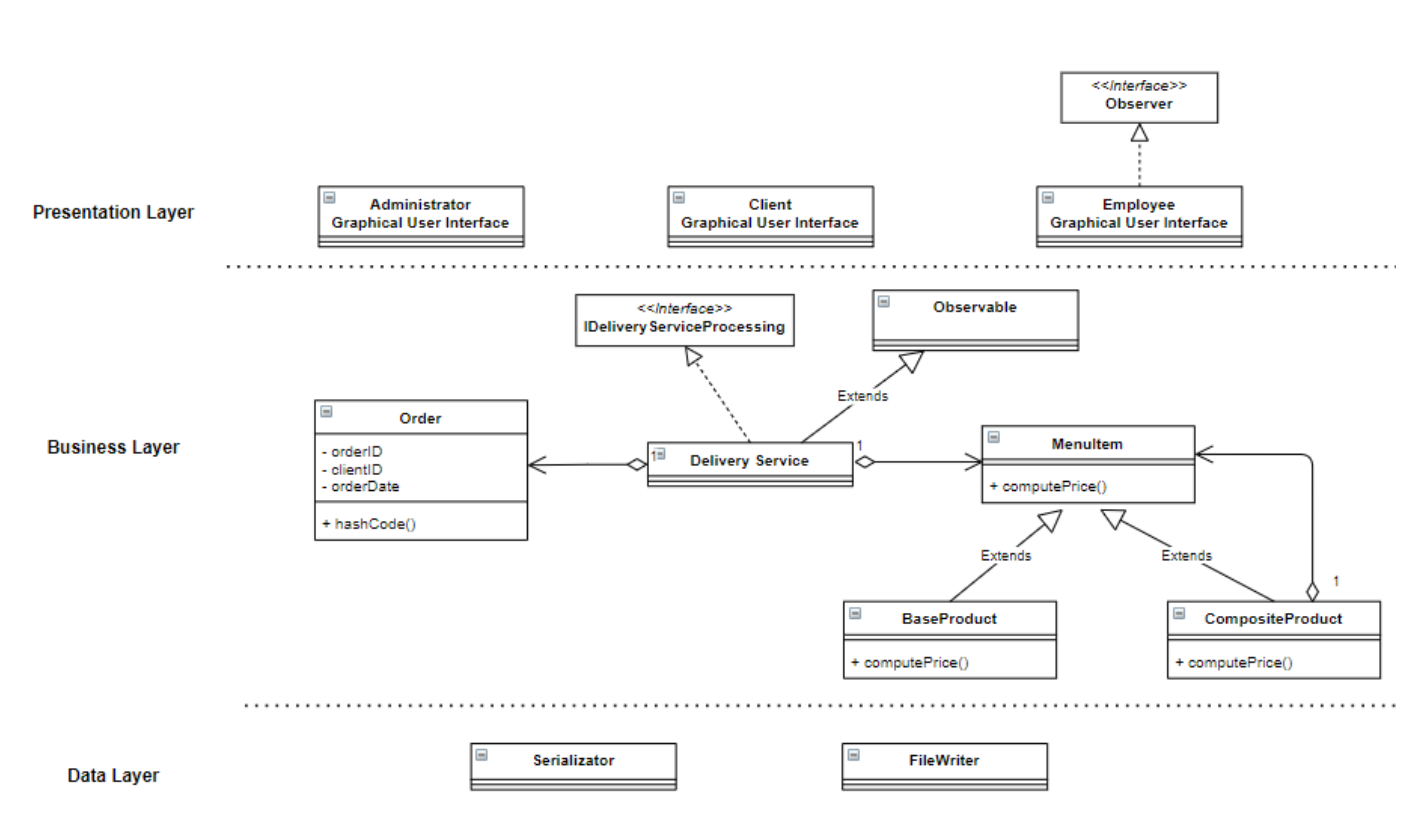
1. The admin inserts data for the to be deleted product
2. The user clicks on the “Delete product” button

**Use Case:** add composed products

**Primary Actor:** admin

**Main User Scenario:**

1. The admin inserts the name of the base products to be used for the composed product
2. The admin clicks on the “Add to composed product” button for each added base product
3. The admin validates the new composed product by pressing the “Add composed product” button.
4. **Design**

**UML class diagram:**

**UML package diagram:**

**Diagram

Description automatically generated**

**Data Structures:**

ArrayList – to manage data structures through deserialization

HashMap – to keep track of menu items, orders and clients. It does not allow duplicates, thus it can be a big help to eliminate duplicates.

Text

Description automatically generated

**Interfaces:**

Observer – in the EmployeeView class in order to notify the employee when a new order has been placed.

1. **Implementation**

**Serializator:**

Deserialize:

public static HashMap<String, MenuItem> deserializeMenu(){  
 HashMap<String, MenuItem> menuItemHashMap = new HashMap<>();  
 List<BaseProduct> menuList;  
 Path path = Path.*of*("products.csv");  
  
 try {  
 menuList = Files.*lines*(path)  
 .skip(1)  
 .map(line -> {return *getProduct*(line);})  
 .collect(Collectors.*toList*());  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
  
 int i = 1;  
 for(BaseProduct p : menuList){  
 p.setId(i);  
 i++;  
 menuItemHashMap.put(p.getTitle(), p);  
 }  
  
 return menuItemHashMap;  
  
}

Serialize:

public static void serializeMenu(HashMap<String, MenuItem> menuItemHashMap){  
 String endl = System.*getProperty*("line.separator");  
  
 try {  
 Writer writer = new FileWriter("products.csv");  
 writer.flush();  
 writer.append("Title,Rating,Calories,Protein,Fat,Sodium,Price");  
 writer.append(endl);  
 for(String m : menuItemHashMap.keySet()){  
 writer.append(menuItemHashMap.get(m).getTitle())  
 .append(',')  
 .append(String.*valueOf*(menuItemHashMap.get(m).getRating()))  
 .append(',')  
 .append(String.*valueOf*(menuItemHashMap.get(m).getCalories()))  
 .append(',')  
 .append(String.*valueOf*(menuItemHashMap.get(m).getProtein()))  
 .append(',')  
 .append(String.*valueOf*(menuItemHashMap.get(m).getFat()))  
 .append(',')  
 .append(String.*valueOf*(menuItemHashMap.get(m).getSodium()))  
 .append(',')  
 .append(String.*valueOf*(menuItemHashMap.get(m).getPrice()))  
 .append(',')  
 .append(endl);  
  
 }  
 writer.close();  
  
 }catch(IOException e){  
 e.printStackTrace();  
 }  
}

**Delivery service:**

Search:

public DefaultTableModel searchProduct(String title, double rating, double calories, double protein, double fat, double sodium, double price) {  
 assert title != null;  
 ArrayList<String> searchedTitles = new ArrayList<>();  
  
 menuItems.keySet().stream()  
 .filter(p -> menuItems.get(p).getTitle().contains(title) &&  
 (menuItems.get(p).getRating() == rating || rating == -1) &&  
 (menuItems.get(p).getCalories() == calories || calories == -1) &&  
 (menuItems.get(p).getProtein() == protein || protein == -1) &&  
 (menuItems.get(p).getFat() == fat || fat == -1) &&  
 (menuItems.get(p).getSodium() == sodium || sodium == -1) &&  
 (menuItems.get(p).getPrice() == price || price == -1))  
 .forEach(searchedTitles :: add);  
  
 for(String s : searchedTitles) {  
 searchedItems.put(s, menuItems.get(s));  
 }  
  
 return displayProducts(searchedItems);  
}

**UX:**

**Log in:**

**Graphical user interface, application

Description automatically generated**

**Sign up:**

**Graphical user interface, application

Description automatically generated**

**Admin view:**

**Graphical user interface, text, application, table

Description automatically generated**

**Client view:**

**Graphical user interface, application, table

Description automatically generated**

1. **Bibliography**

<https://www.geeksforgeeks.org/serialization-in-java/>

<https://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html#tag>

https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html