Assignment 1:

Food delivery system

Student: Coroian Tudor Florentin

Group: 30433/1

Date: 29.03.2021

1. Requirements analysis

1.1 Assignment specification

The purpose of this assignment is to provide us with an understanding of the Layered Architectural Pattern and the way it is implemented into various systems and applications.

The task is to develop a food delivery application in Java. The application should accommodate for two types of users: admin type and customer type. These users should provide a username and a password in order to access their respective accounts. In the same manner, the application should provide its user the ability to apply the basic operations on a database (Create, Read, Update, Delete), if the data from the input is correctly written.

1.2 Functional Requirements

The application should be linked to a database into which the user accounts, orders and products will be stored. The administrator should be able to view credentials and the profile of any user, in addition to adding products into the database.

In short terms, the program should be able to perform the CRUD operations on different tables from the database, as well as some other actions, defined in the formal statement of the problem.

Formal statement

Use JAVA/C# API to design and implement a food delivery application. The application should have two types of users (a regular user represented by the customer and an administrator user) which have to provide a username and a password in order to use the application.

Taking into account the formal statement of the problem as well as the specifications mentioned above, I present a table with the outline of the subtasks.

User type	Registration	Database operation	Other functionalities
Customer	Create an account by providing the necessary information. Login into the account, based on the username and password provided.	CRUD on the personal information (i.e. own account) CRUD on the personal shopping cart.	Chose a payment method Visualize historical data (i.e. past orders)
Administrator	Create an account by providing the necessary information. Login into the account, based on the username and password provided. Manage own account information.	CRUD on each of the customer personal information (i.e. their accounts). CRUD on the products table.	Generate activity reports for a certain date. Set a customer as loyal (which in term provides the customer with a discount).

1.3 Non-functional Requirements

The non-functional requirements mainly focus on the language used. In this regard, the designer should choose a language that allows the usage of the Object Oriented Programming paradigms in order to model the database.

Other constraints are the ones that come with the OOP paradigms, which will not be mentioned, due to their acknowledgement by all software designers.

In addition to these non-functional constraints, there are some presented in the formal statement of the problem.

First of all, the data will be stored in a database. Use the Layers architectural pattern to organize your application. Use a domain logic pattern (transaction script or domain model) / a data source hybrid pattern (table module, active record) and a data source pure pattern (table data gateway, row data gateway, data mapper) most suitable for the application.

Second, all the inputs of the application will be validated against invalid data before submitting the data and saving it in the database.

2. Use-Case model

The use case model was partly specified in the **Functional Requirements** subchapter of this documentation. As such, I will only present a list that will fully specify the action that a user of the application can take. I chose this list approach to show the use-cases of each actor because it is easier to read and understand, since it has only two actors, with just a few overlapping use-cases.

Customer Use-cases:

1. Log in
2. Sign up
3. View products
4. Add product to cart
5. View personal information
6. Choose payment method
7. Generate orders history
8. Delete item from cart
9. Submit order
10. Start order
11. See orders by date
12. Edit personal information

Admin Use-cases:

1. Log in
2. Sign up
3. View products
4. Add product
5. Update product
6. Delete product
7. View all products
8. Add customer
9. Update customer
10. Delete customer
11. View all customers
12. View all orders
13. View orders by date
14. View orders by customer
15. Make customer loyal

3. System architectural design

In the first chapter of this application, one of the requirements presented was the need to use the Layered Architectural Design. In order to manage this requirement (and fulfill the non-functional requirements of the assignment), I used a data source hybrid pattern (the table module) and a data source pure pattern (table data gateway) to organize the structure of the application.

The different layers correspond to the packages of the application. As such, we distinguish between the following layers/packages:

Database/connection: This is the layer of the database itself. For this application I chose MySQL as the database. The connection package contains one class, namely ConnectionFactory, which is designed with the Singleton principle in mind, such that the database is accessed through only point in the application.

Model/model: This is the layer that stores the classes with which the application was modeled. Each class in this package corresponds to a table in the database.

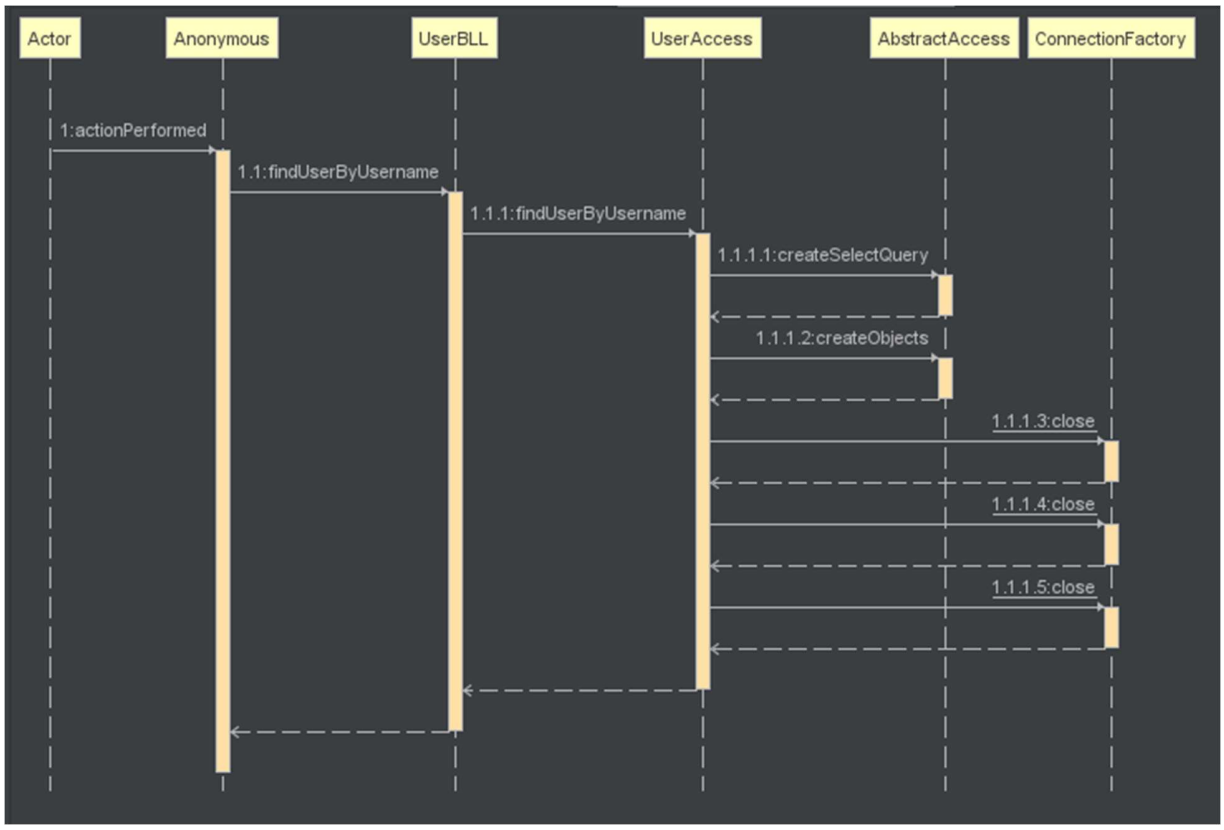
Repository/access: This is the layer that contains the actual implementations of the CRUD operations on the database. The classes in this package perform certain queries on the database and return their respective results.

Service/bussinesLogic: This layer is an extension of the repository layer. The classes in this package are meant to further process the data, such that only the expected answer (or the expected instruction) pass to and from the repository layer.

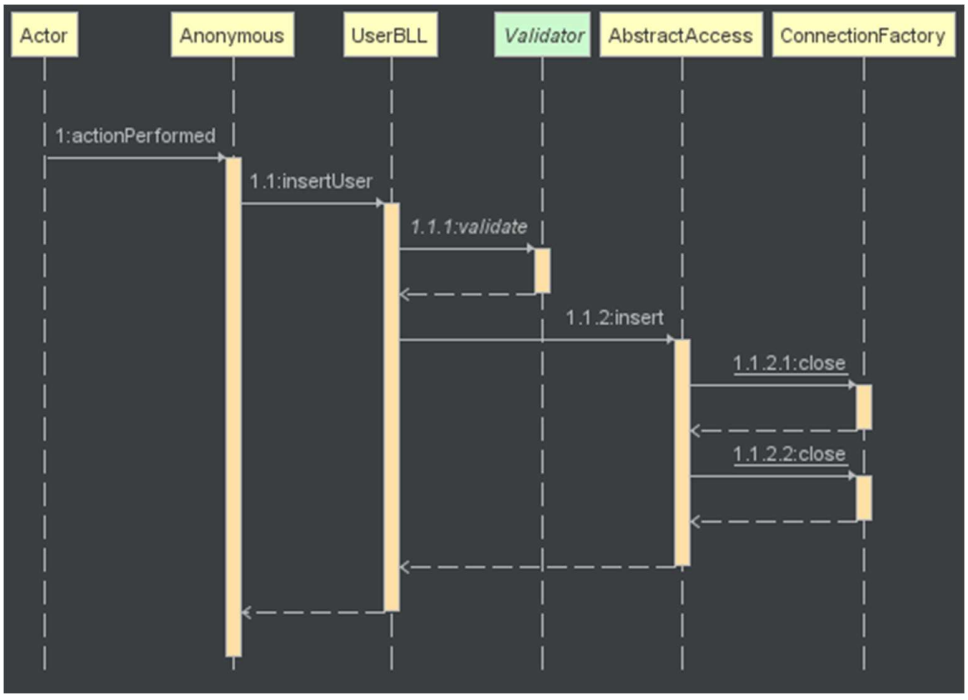
Controller and Presentation/presentation: For timeframe purposes, the Controller and Presentation Layers have been merged into a single package. However, their functionalities are separated, thus the structure required by the Layer Architectural Design is respected.

4. UML Sequence Diagrams

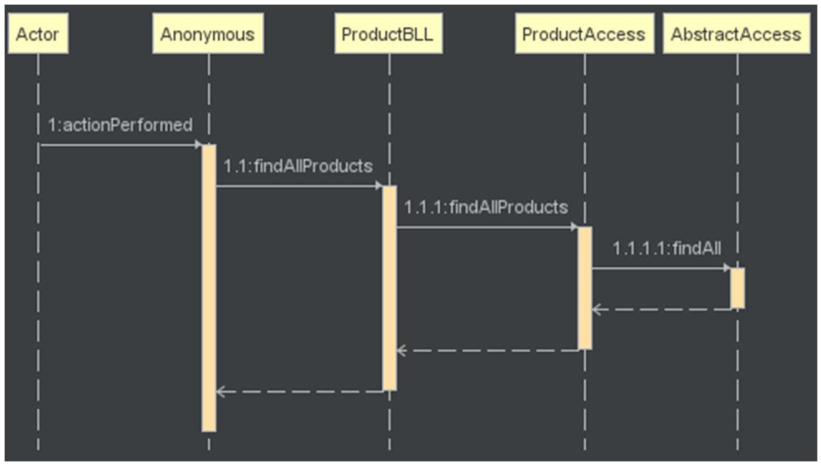
Login



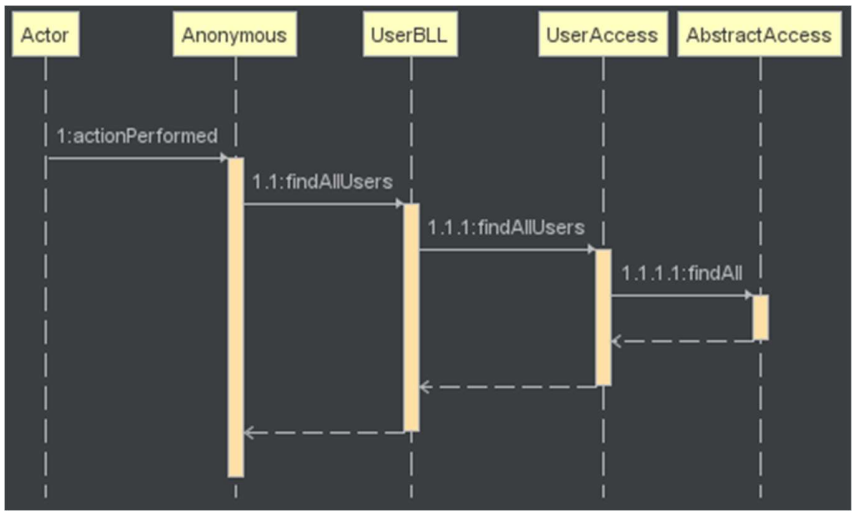
Signup



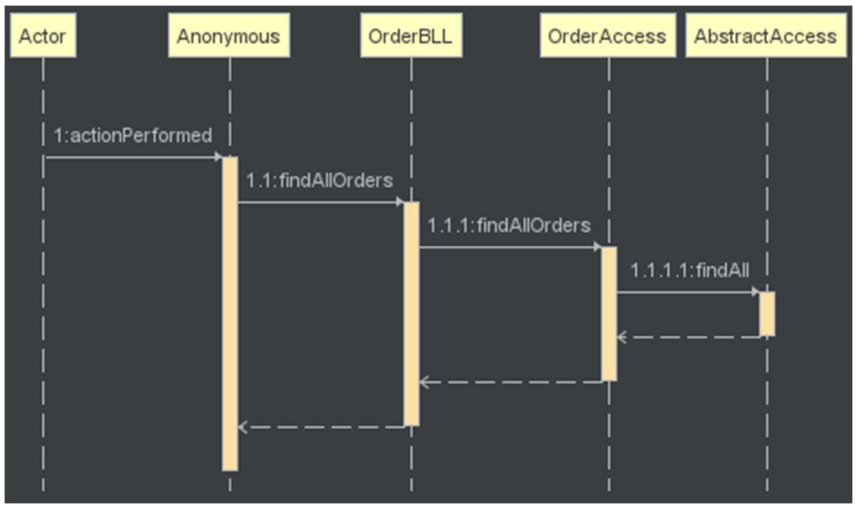
Show all products



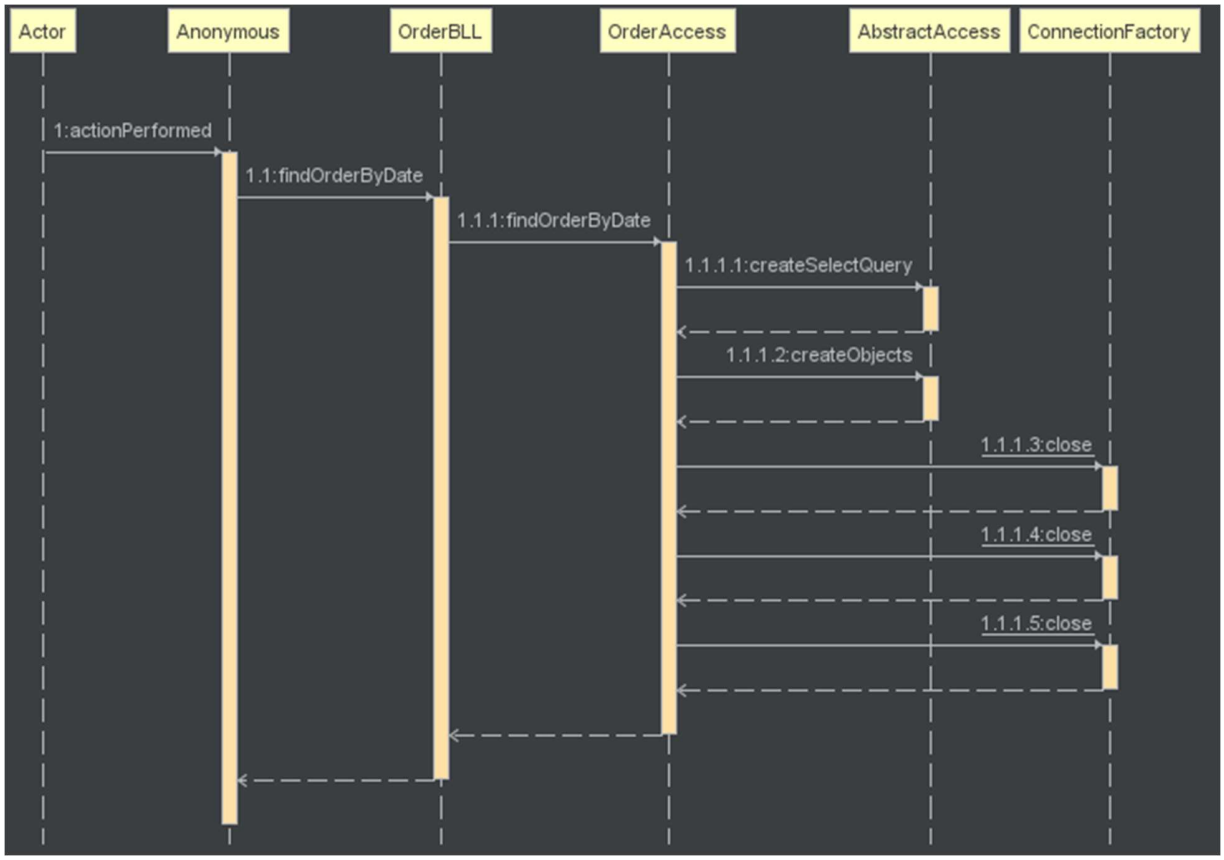
Show all users



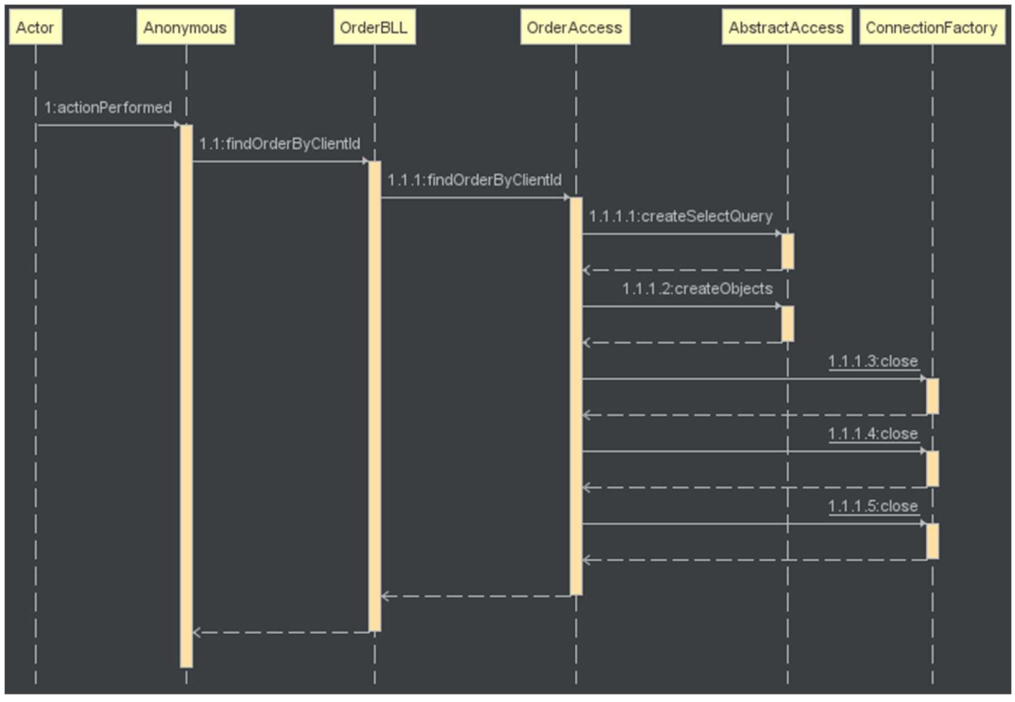
Show all orders



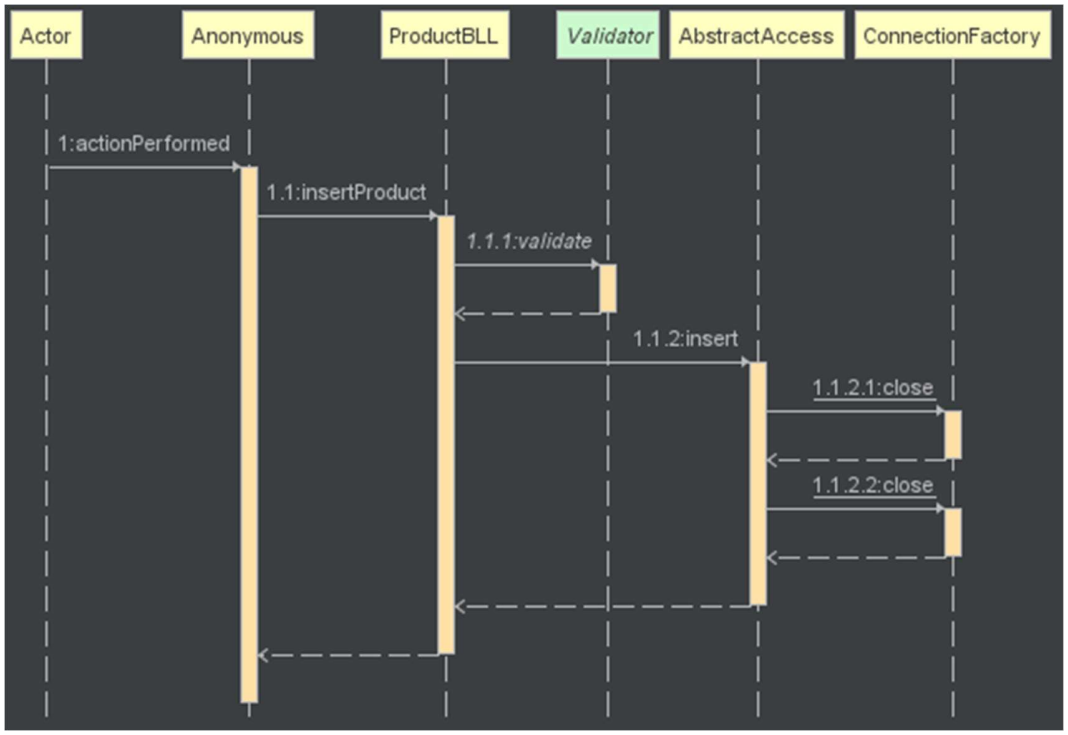
Show orders by date



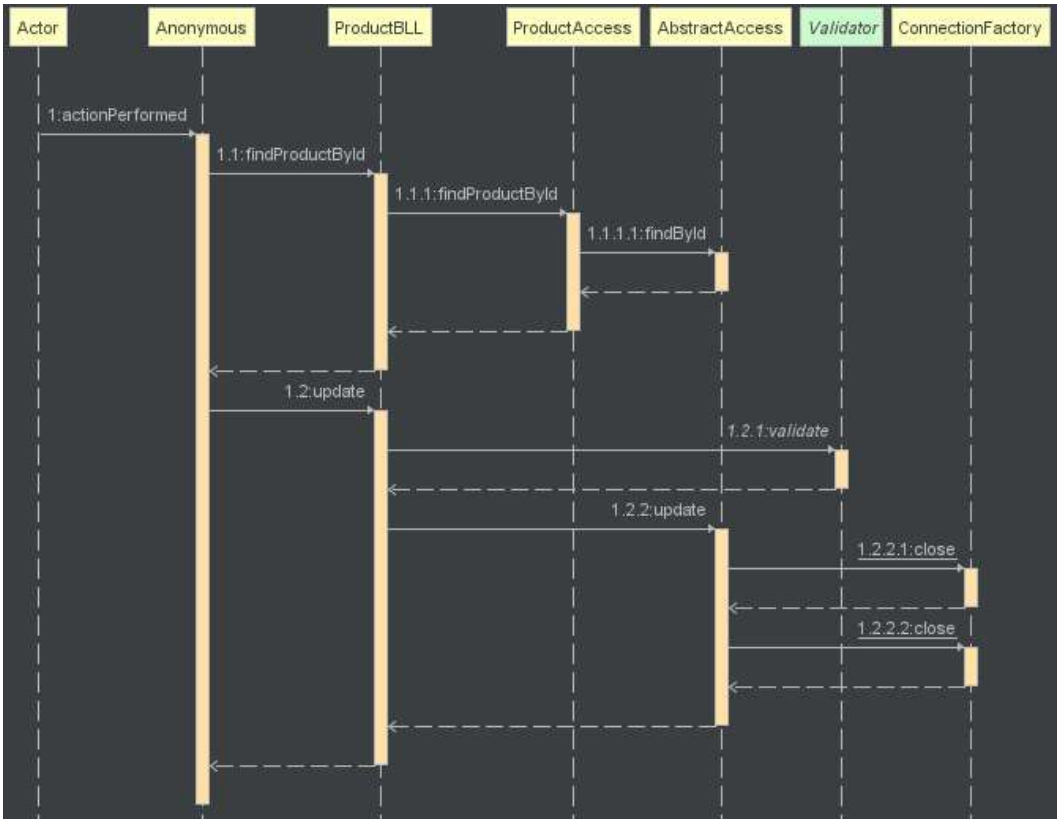
Show orders by customer



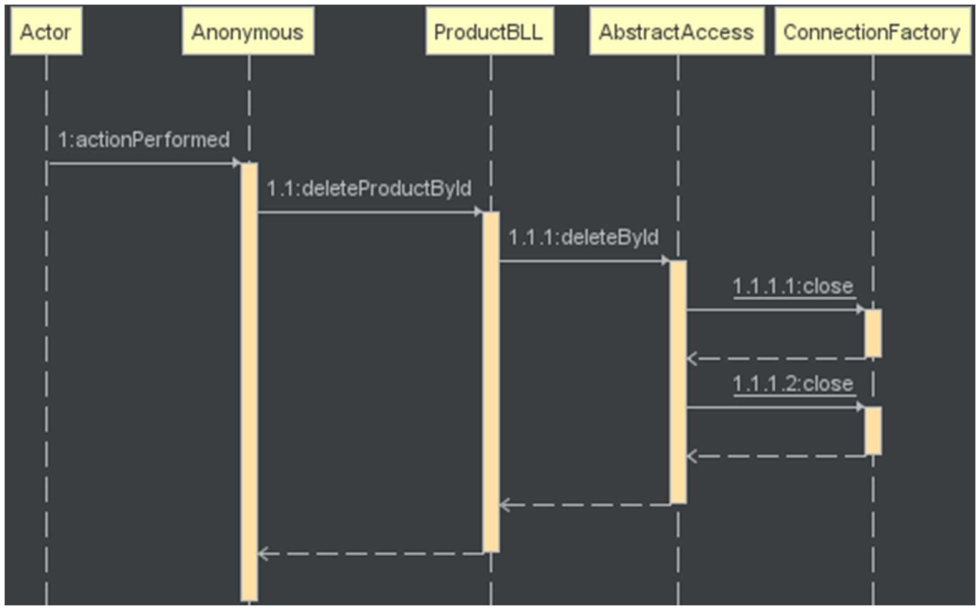
Add product



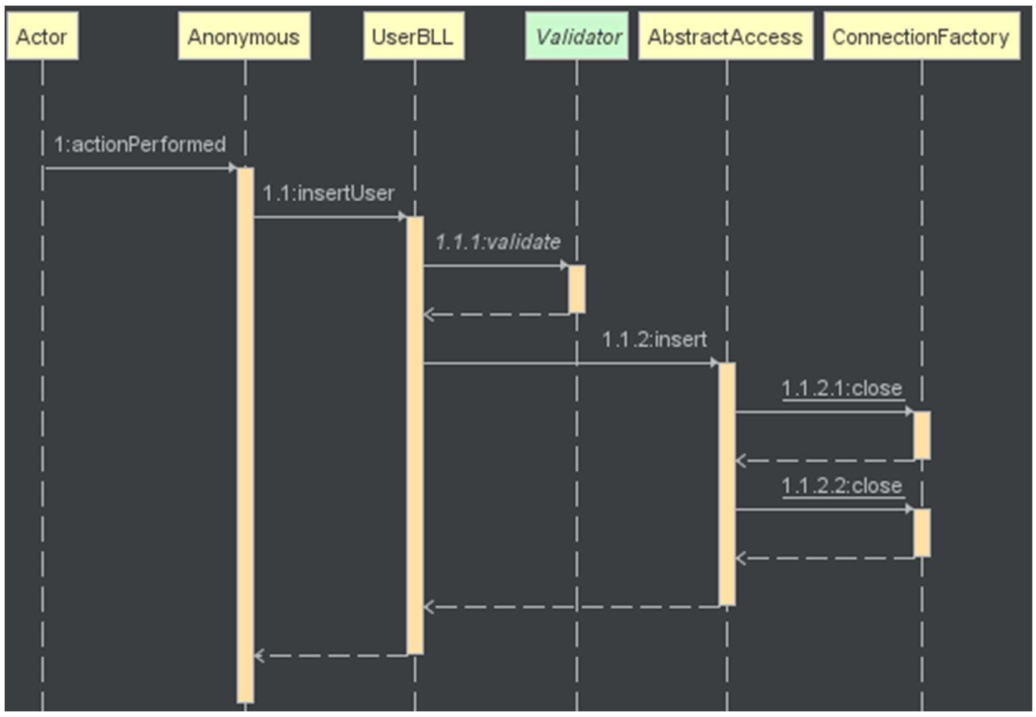
Update product



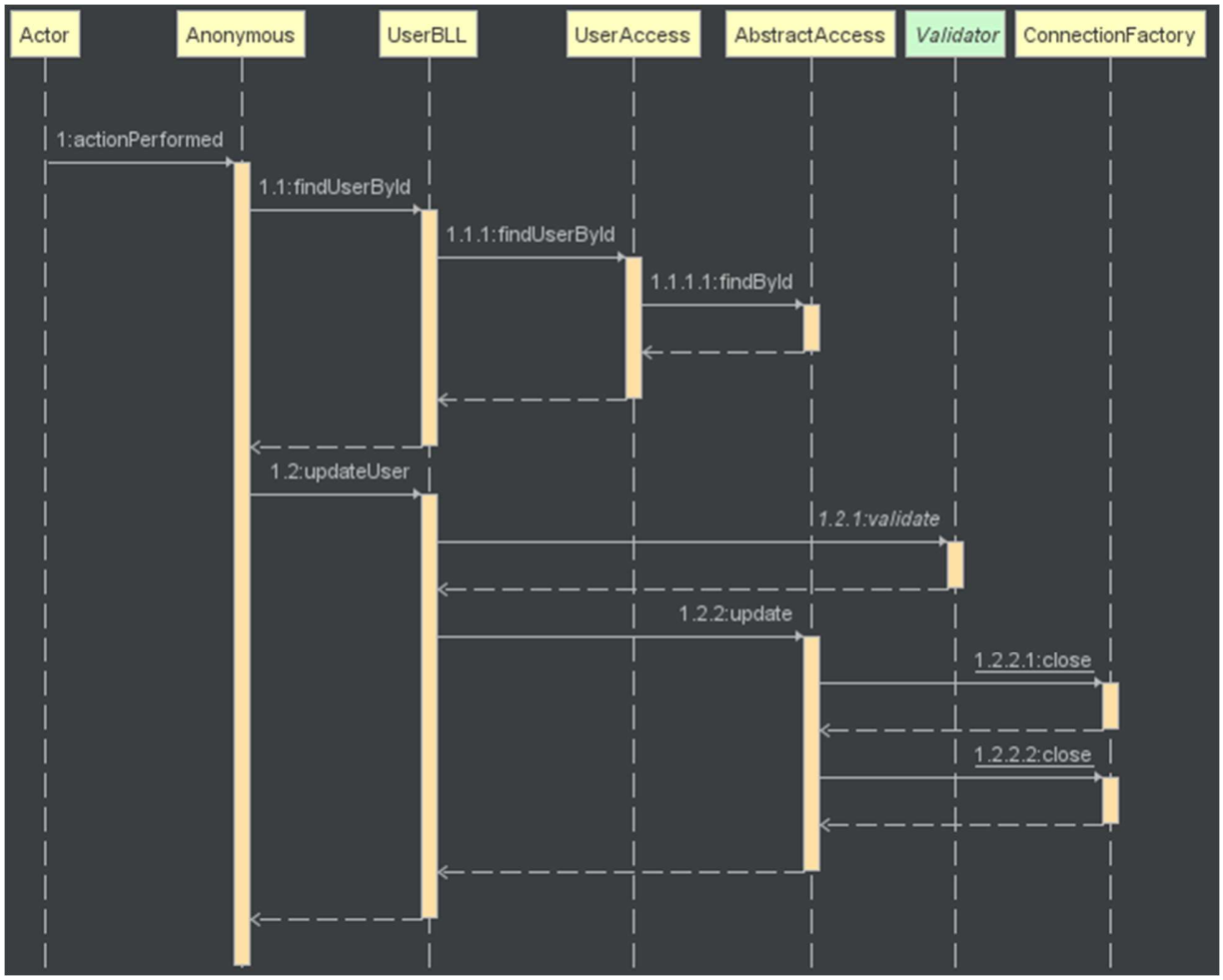
Delete product



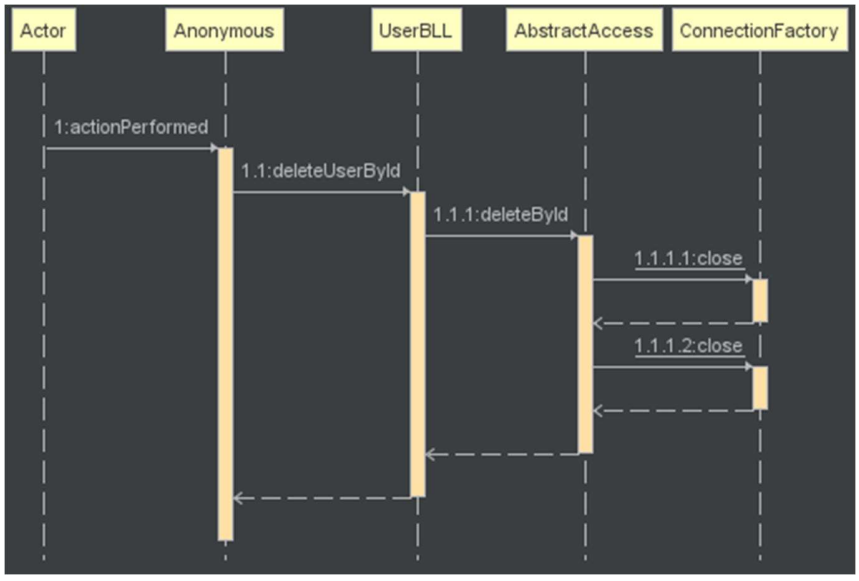
Add user



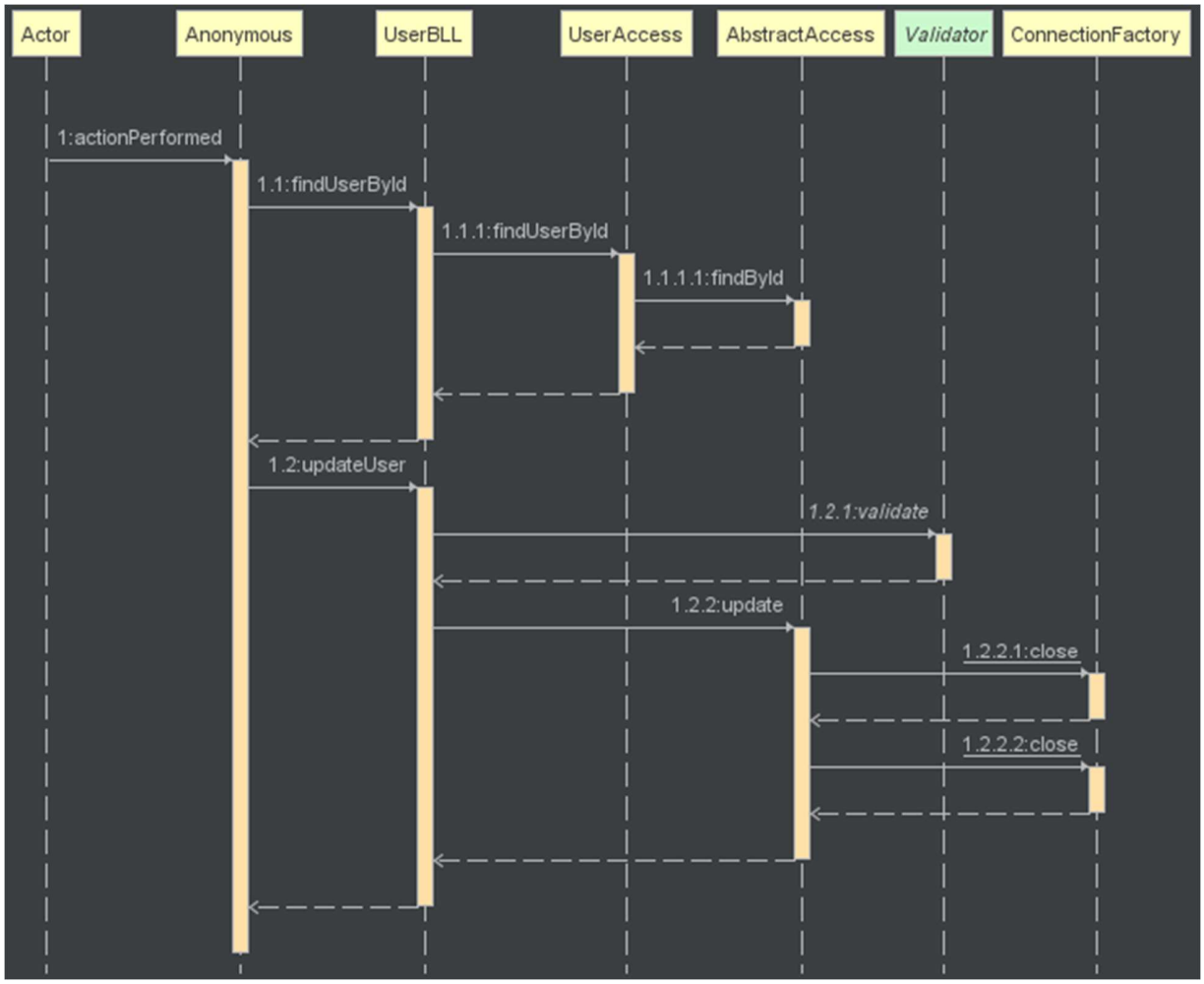
Update user



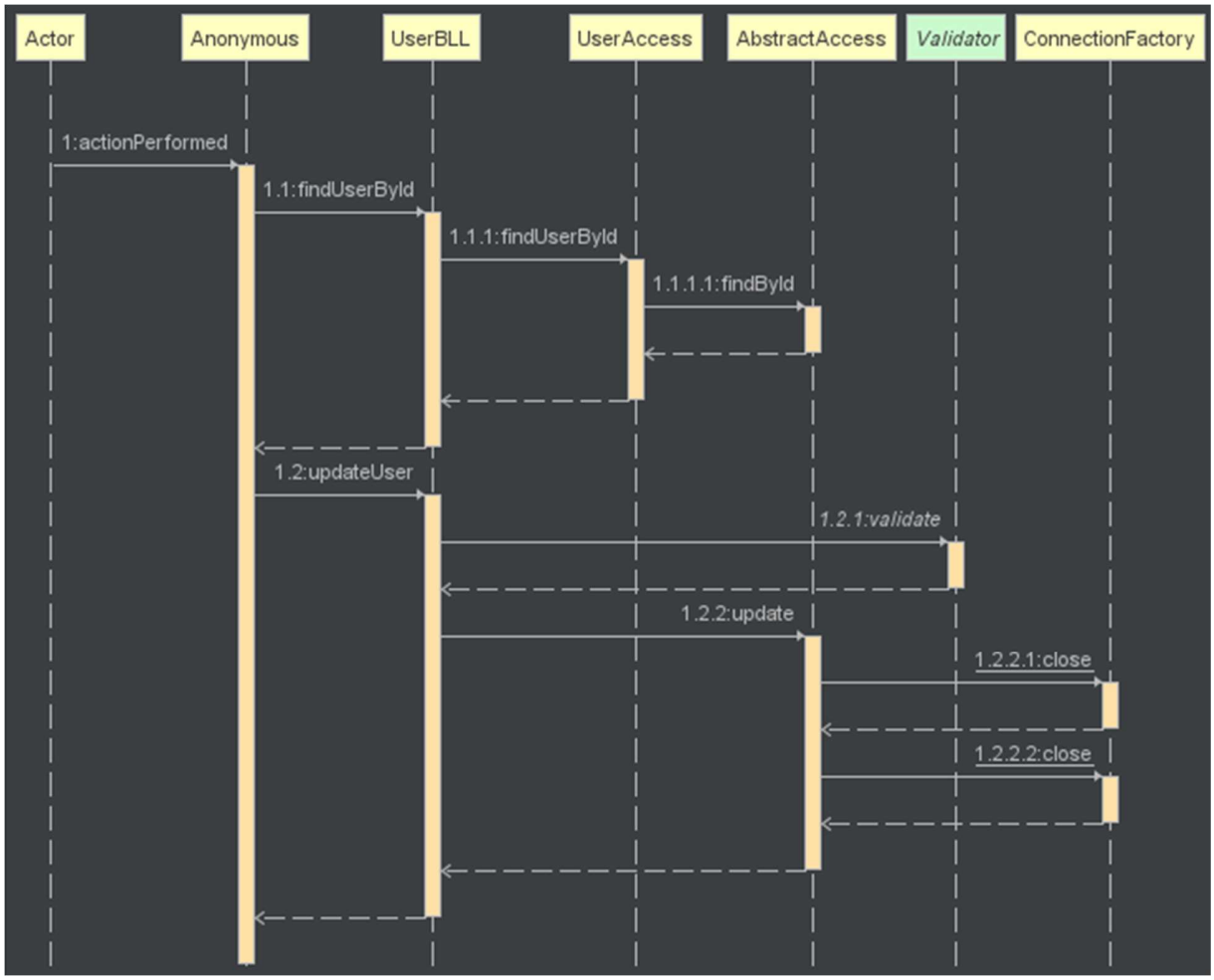
Delete user



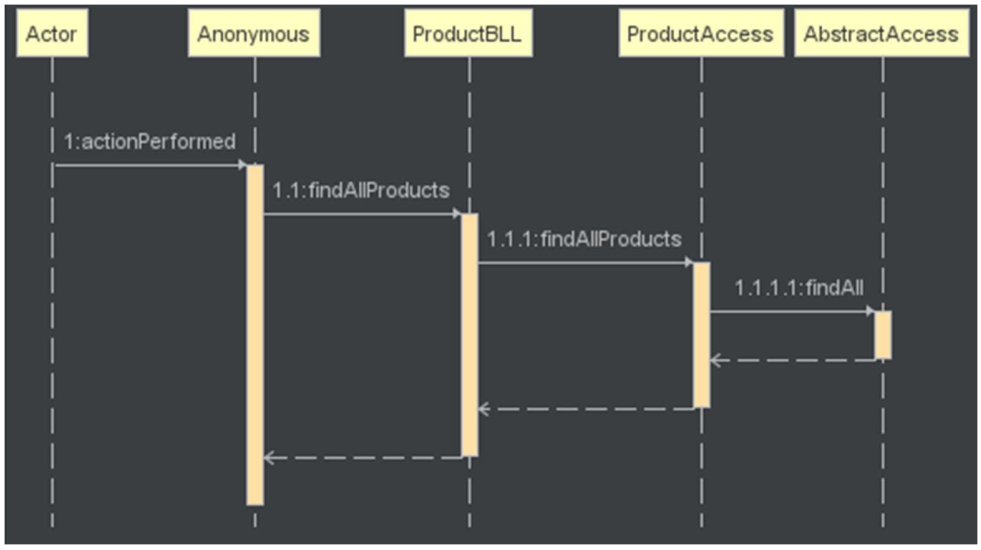
Make customer loyal



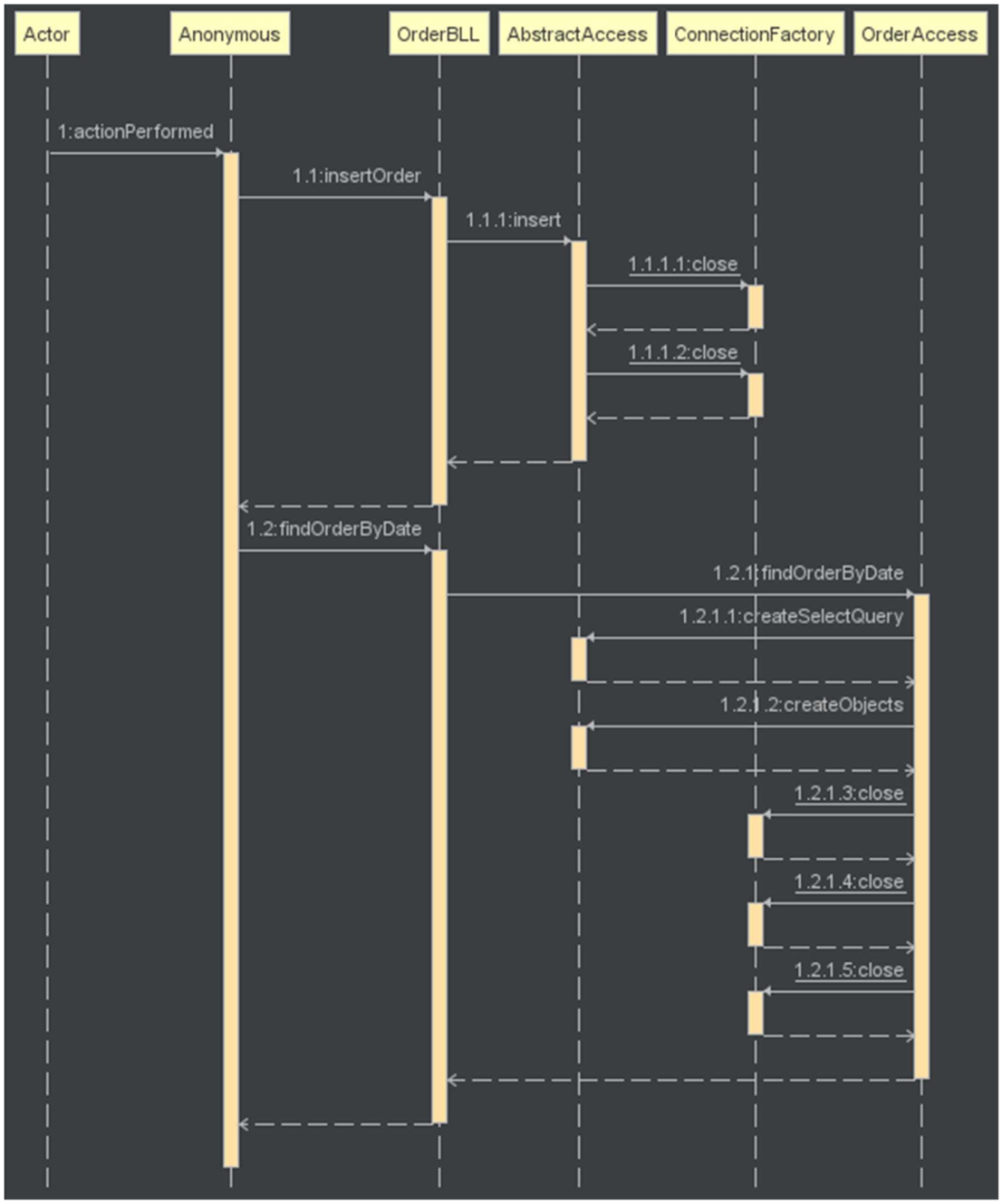
Update personal customer profile



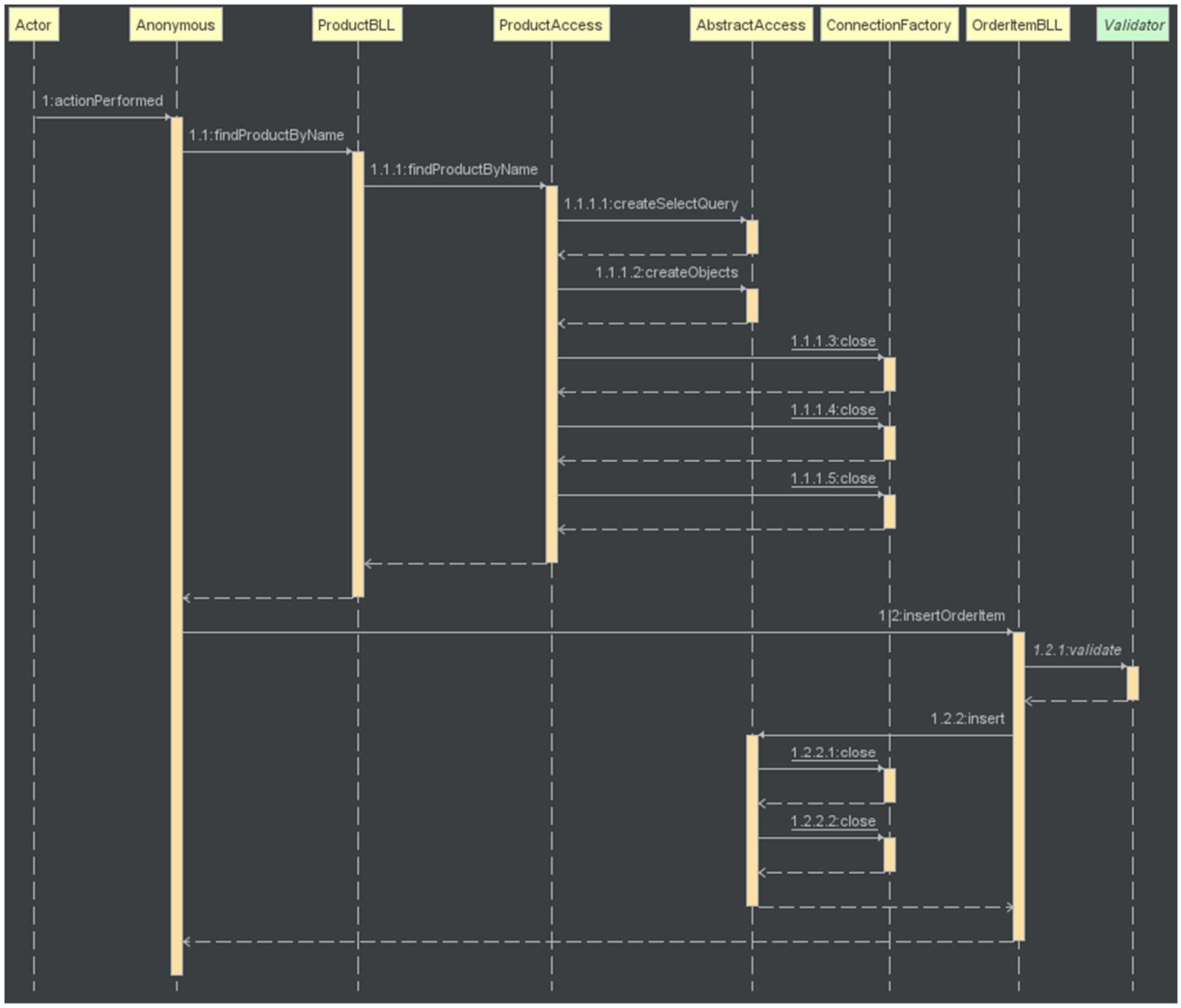
Show menu



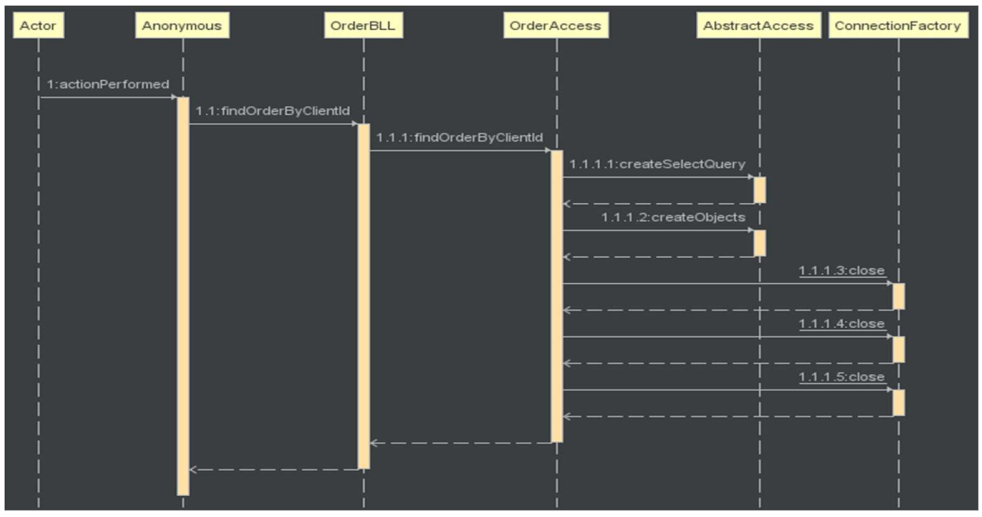
Start order


Actor
Anonymous
OrderBLL
AbstractAccess
ConnectionFactory
OrderAccess
1:actionPerformed
1.1:insertOrder
1.1.1:insert
1.1.1.1:close
1.1.1.2:close
1.2:findOrderByDate
1.2.1:findOrderByDate
1.2.1.1:createSelectQuery
1.2.1.2:createObjects
1.2.1.3:close
1.2.1.4:close
1.2.1.5:close

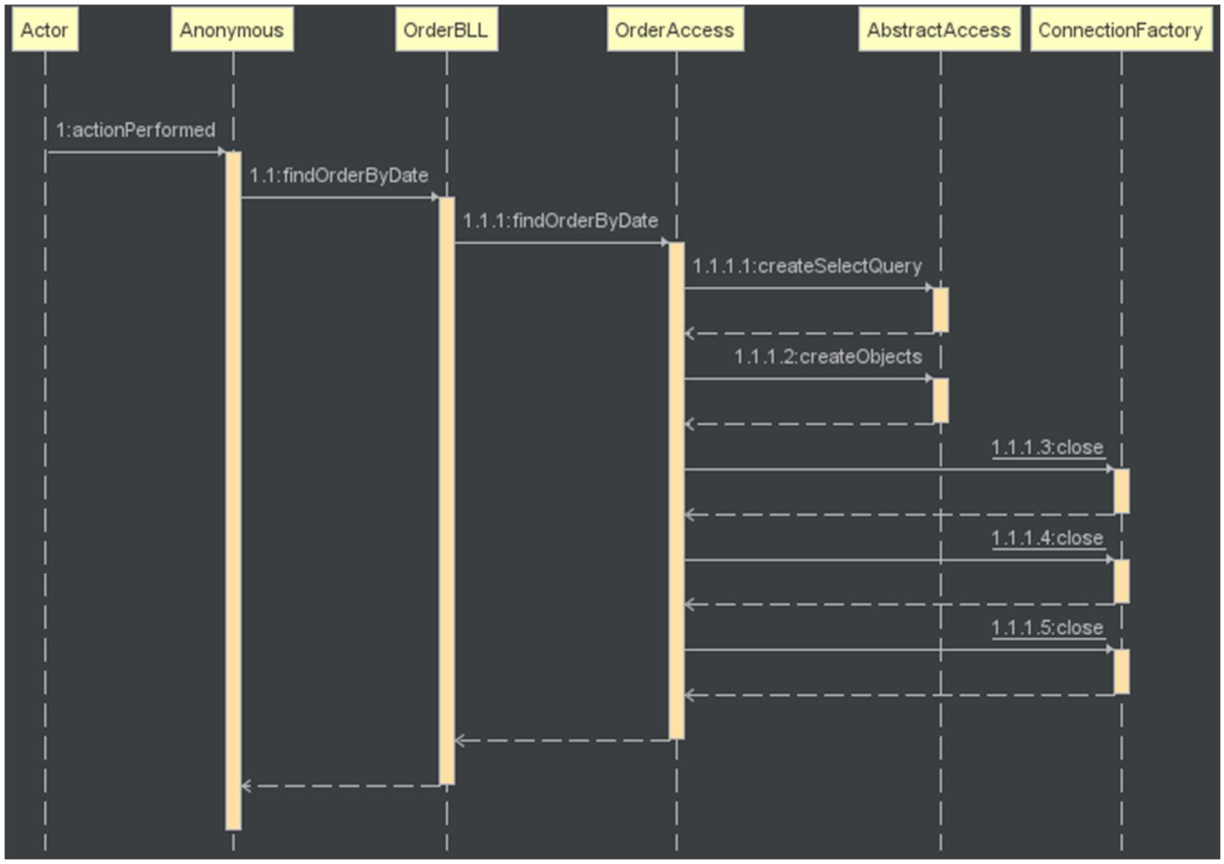
Add to cart



Show orders to customer



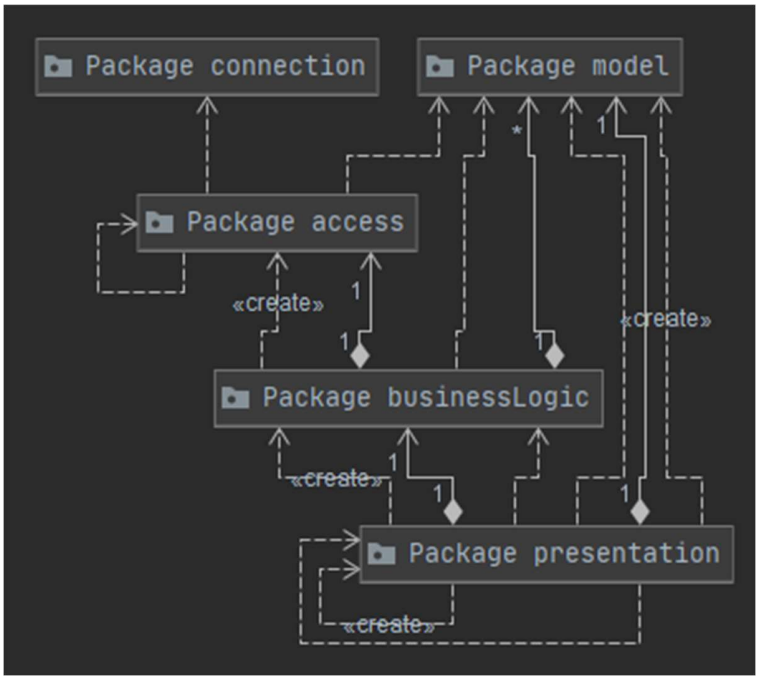
Show orders by date to customer



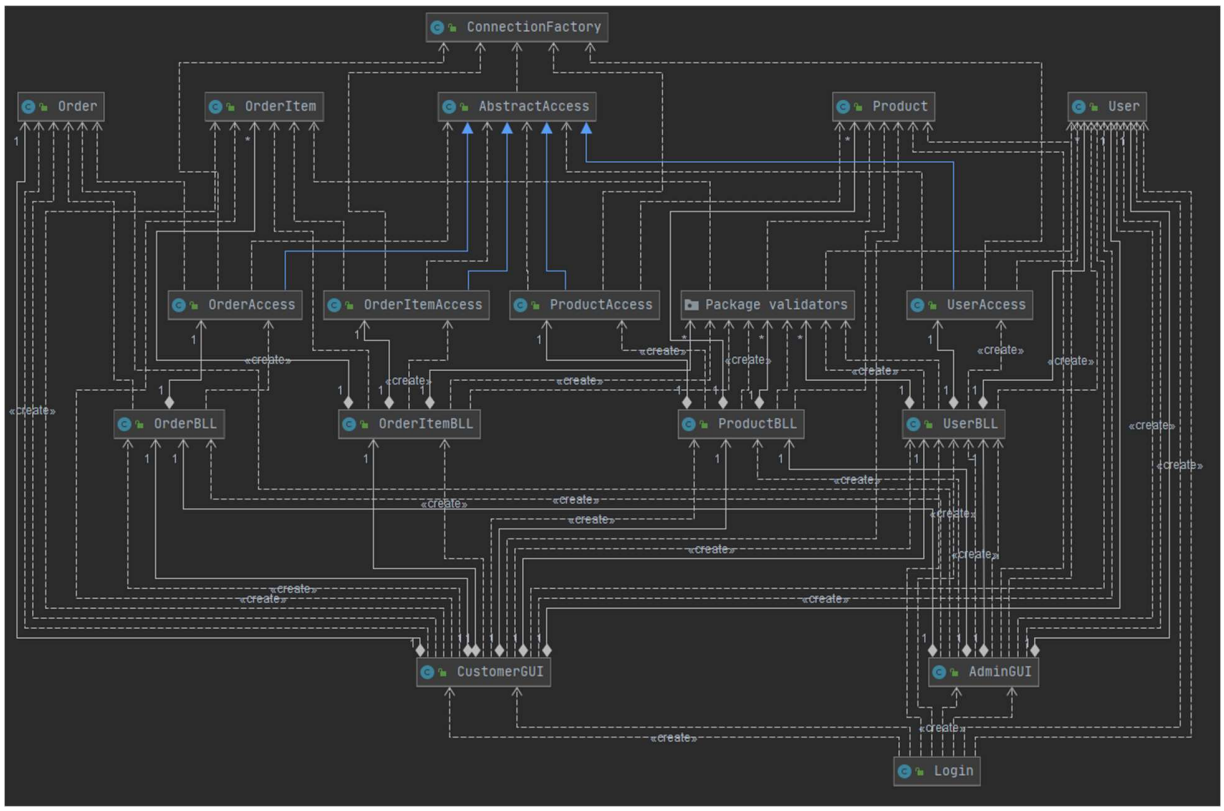
5. Class Design

This section presents a more detailed view on the classes involved in the execution of the food delivery application. For the sake of simplicity, only one class from each package will be presented, with the mention that the rest of them follow the same specific approach. In case of a major difference between the presented class and another class in the same package, said class will be also presented fully.

Before presenting the classes individually, I attached bellow the UML package diagram to further concrete the Layered Architectural Pattern.



Also, for a more in-depth view of the structure of the project, I attached bellow the UML Class diagram.



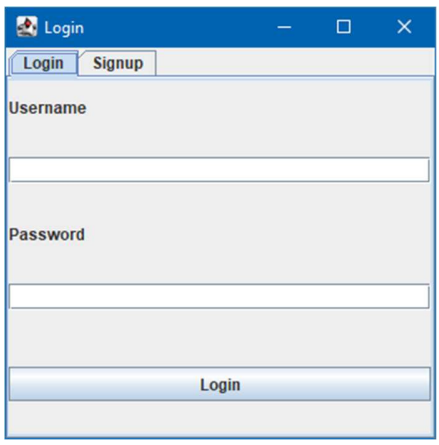
MainClass

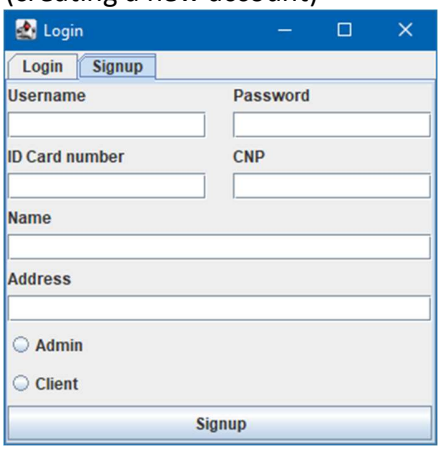
- application starting point; it only instantiates the LoginGUI

LoginGUI

- contains methods and fields responsible for receiving and displaying the necessary information from the user

- it has two tabs: one for login and another one for signing up (creating a new account)

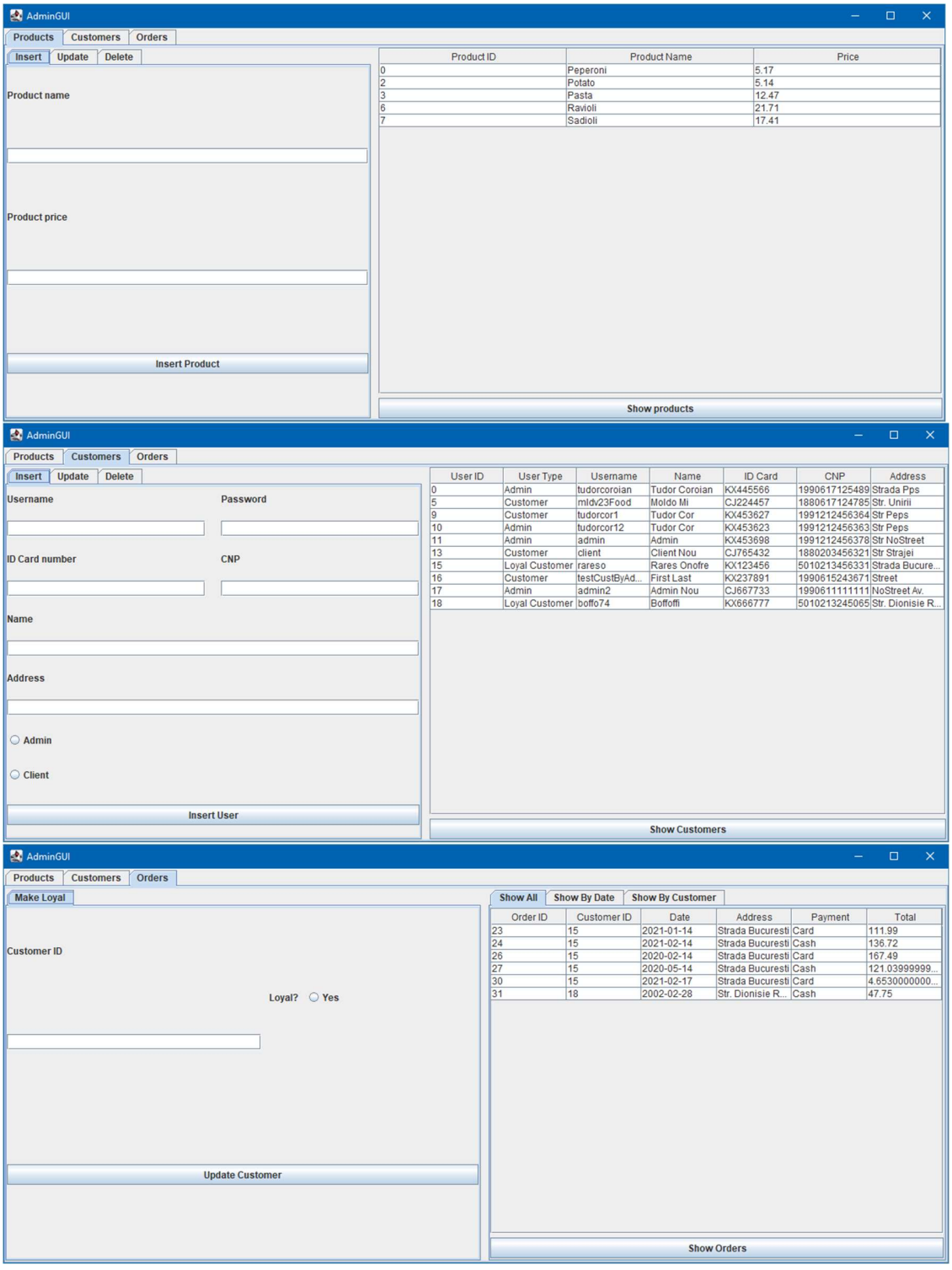




AdminGUI

- it is only displayed to an administrator of the application

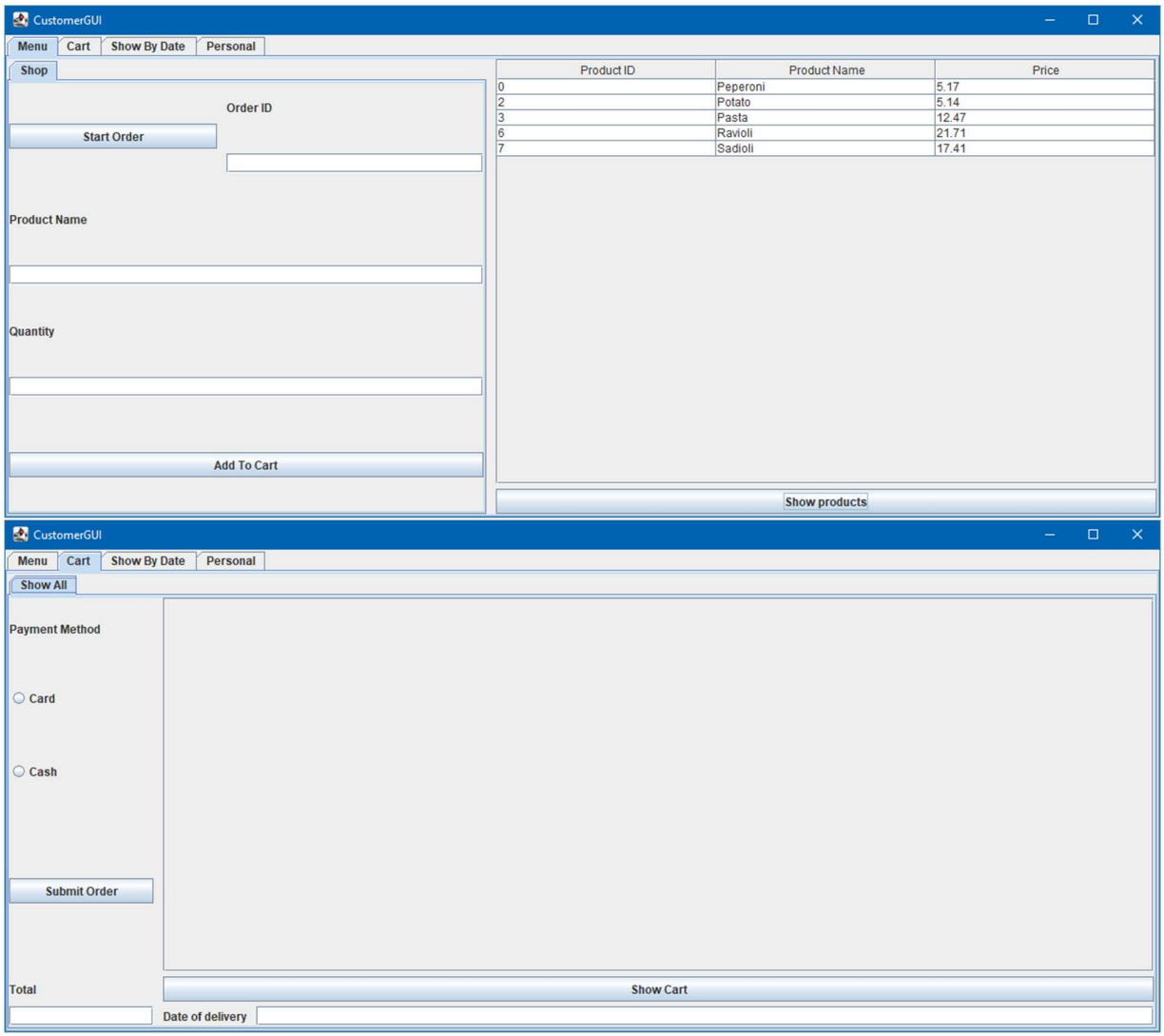
- has 3 tabs that correspond to products, customers and orders from the database

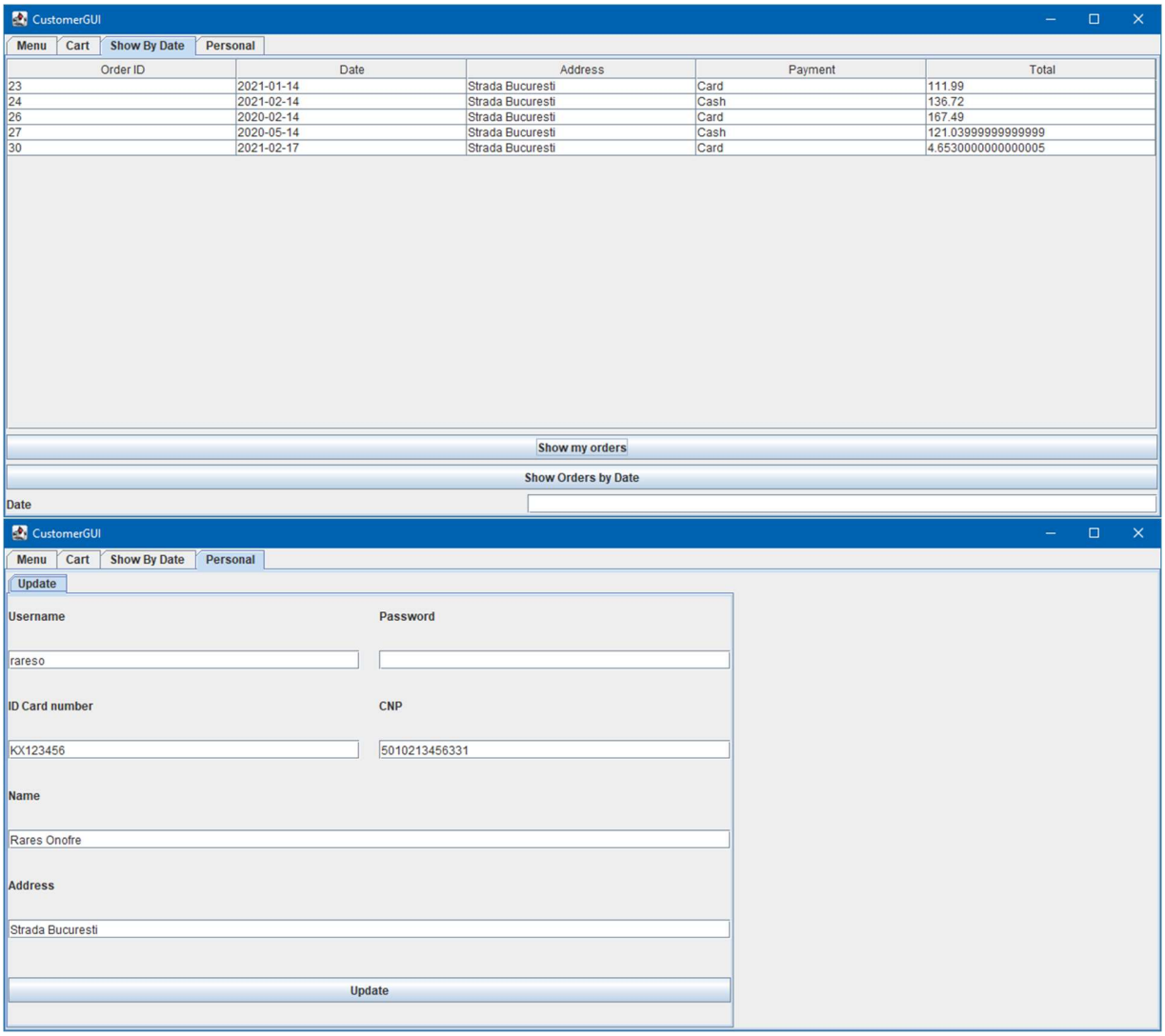


CustomerGUI

- it is only displayed to a customer

- it has 4 tabs: Menu, Cart, Show (orders) By Date, and Personal (profile information)





User

- model class for the application

Product

- model class for the application

OrderItem

- model class for the application

Order

- model class for the application

ConnectionFactory

- contains the link to the MySQL database

UserBLL

- contains an instance of the UserAccess class in order to process the results of the queries return by that class

ProductBLL

- contains an instance of the ProductAccess class in order to process the results of the queries return by that class

OrderItemBLL

- contains an instance of the OrderItemAccess class in order to process the results of the queries return by that class

OrderBLL

- contains an instance of the OrderAccess class in order to process the results of the queries return by that class

<<The validators package>>

- this package contains a series of validators class for the inputs of the application

- they use regex in order to validate against incorrectly written input and throw an exception in case the input is not as expected

AbstractAccess

- abstract class that implements the basic CRUD operations for any table in the database

- it uses reflexion techniques

- is the class responsible for creating the queries and sending them to the ConnectionFactory

UserAccess

- extends the AbstractAccess class and adds a series of particular method for executing more specialized queries

ProductAccess

- extends the AbstractAccess class and adds a series of particular method for executing more specialized queries

OrderItemAccess

- extends the AbstractAccess class and adds a series of particular method for executing more specialized queries

OrderAccess

- extends the AbstractAccess class and adds a series of particular method for executing more specialized queries

6. Data Model

The following class that are going to be presented are the ones used to model the application. Their use helps the designer have a clear distinction between each of the objects of the application as well as the information in the database. These classes can be found in the **model** package.

User

The user model class contains the following fields:

```
private int user_id;
private int user_type;
private String username;
private String password;
private String name;
private String id_card;
private String cnp;
private String address;
private int deleted;
```

Each of these fields has an equivalent column in the **user** table from the database (a column with the same name). It was modeled to accommodate for three types of users of the application: administrators, customers, and loyal customers. The loyal customer account type can only be created by an admin.

Special mentions about the **deleted** field which is used inside the database in order for the application to be able to perform a soft delete, instead of a hard one. This field is the same for all the other classes.

Product

The product model class contains the following fields:

```
private int product_id;
private String name;
private double price;
private int deleted;
```

This class was modeled specifically for a simple product having only the name of that product and its price.

Order

The order model class has the following fields:

```
private int order_id;
private int client_id;
private double total;
private int deleted;
private String dateBuy;
private String address;
private int payment;        // 0 => cash
                            // 1 => card
```

This class was modeled in order to allow the operation on order entities that have the date stored in them and also the type of payment the client chose. The total is computed by adding up all the entries in the orderItem table that have the same order_id as the one specified in the object of this class (and applying the discount for the loyal customer, if necessary).

OrderItem

The order item class has the following fields:

```
private int order_item_id;
private int order_id;
private String product_name;
private int quantity;
private int deleted;
```

The important fields of this class are the product name and the quantity. In them, the actual information of the order is stored (in terms of what products are bought). The order_id field corresponds to the order_id field presented in the order model class, based on which the indexation is done (with regard to which order a particular order item belongs to).

7. System Testing

The testing for this system was performed along with the development of this application. This means that when a new feature was added, that feature would be tested against an optimal scenario (i.e. a scenario that was guaranteed to succeed).

Finally, when the application was completely done, it underwent a set of more rigorous test, including some test scenarios that were guaranteed to fail. As a side note, the failure of each task is signal through an exception thrown inside the IDE itself.

8. Bibliography

https://www.tutorialspoint.com/mysql/mysql-update-query.htm

https://www.jetbrains.com/help/idea/2016.1/creating-and-disposing-of-a-form-s-runtime-frame.html

https://jetbrains.design/intellij/controls/checkbox/

https://regex101.com/r/E7d5qK/1/

https://docs.microsoft.com/en-us/dotnet/framework/develop-client-apps?redirectedfrom=MSDN

https://docs.oracle.com/javase/tutorial/jdbc/basics/gettingstarted.html

https://docs.oracle.com/javase/tutorial/uiswing/

http://index-of.es/Java/Creating%20a%20Custom%20Java%20Desktop%20Database%20Application%20-%20NetBeans%206.5%20Tutorial.pdf

https://plugins.jetbrains.com/plugin/8286-sequencediagram

https://www.w3schools.com/sql/sql_delete.asp

https://stackoverflow.com/questions/34329852/how-to-insert-date-values-into-table/34330258

https://www.javatpoint.com/java-string-to-date

https://docs.microsoft.com/en-us/sql/t-sql/data-types/datetime-transact-sql?view=sql-server-ver15

https://stackoverflow.com/questions/37772825/how-to-use-drag-and-drop-in-intellij-for-java-swing/37773316

https://stackoverflow.com/questions/53257073/java-9-replace-class-newinstance

https://stackoverflow.com/questions/56079042/fix-to-java-newinstance-deprecated

https://stackoverflow.com/questions/46393863/what-to-use-instead-of-class-newinstance

https://www.jetbrains.com/help/idea/work-with-maven-goals.html#trigger_goal

https://www.tutorialspoint.com/java/java_date_time.htm

http://coned.utcluj.ro/~salomie/PT_Lic/