DOCUMENTATION

Assignment nr 3

Dora Cristian Adrian
30422

Contents

1. Objectives

1.1. Main Objectives

The main objective for this project is to design and implement an order management application for the client orders for a warehouse. The user has a couples of options, for example to add, delete, edit, and view a specific client or product and even to create an order. The application will store the client, product, and order data in the database. Some basic information about the clients, products and the orders are stored. Not all orders will be accepted, a filter will be implemented such that orders that do not match the criteria will no be accepted. This filter consists of checking if the warehouse has at least as many products of the type of product that is being ordered.

1.2. Secondary Objectives

- Analyse the problem and identify requirements
- Design the orders management application
- Implement the orders management application
- Create a PDF file that contains information about each order
- Use a layered architecture
- Create generic methods
- Test the orders management application

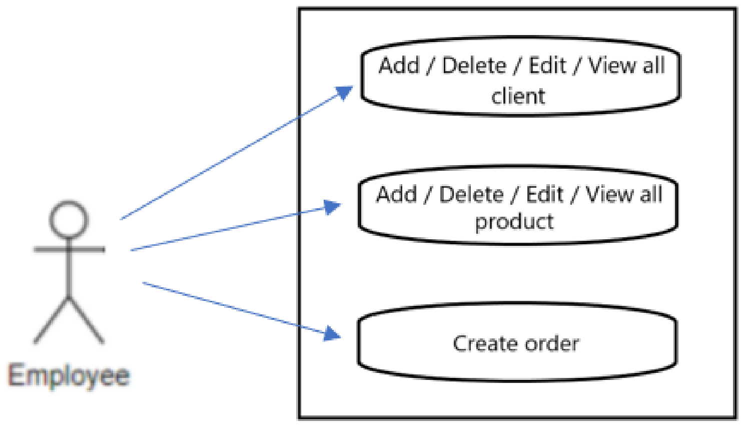
2. Functional and non-functional requirements

2.1. Functional requirements

2.1.1. MoSCoW classification

- Must have: A connection to a database which stores all the relevant information about each client, product, and order. A graphical user interface that allows the user to interact with the application and the database implicitly. CRUD operations on the database.
- Should have: Generic classes and methods that allow the use of the same methods on different types of objects, without duplicate code.
- Could have: Eye catching graphical user interface with nice colours and good-looking aspect regarding positioning of each element
- Won't have: Login page for client/employee. Lack of ability for the client to perform only client operations makes the app useable only by the employees

2.1.2. Use case diagram


Add / Delete / Edit / View all client
Add / Delete / Edit / View all product
Create order
Employee

2.1.3. Use case scenarios

The employee can choose between client or product operations or create an order.

- For the first two options, the user can: add a new client / product, delete them from the database, edit or update them. When 'view all' is chosen, a table with all the entries appears.
- For the create order operation, with the help of two JComboBox, one for clients and the other one for the products, the employee will have the ability to choose an option (a client/product from the list). Another field must be completed where the user is asked for the quantity the user wants to buy. The submit option will finish the order and save it into the database and into a PDF file

2.2. Non-functional requirements

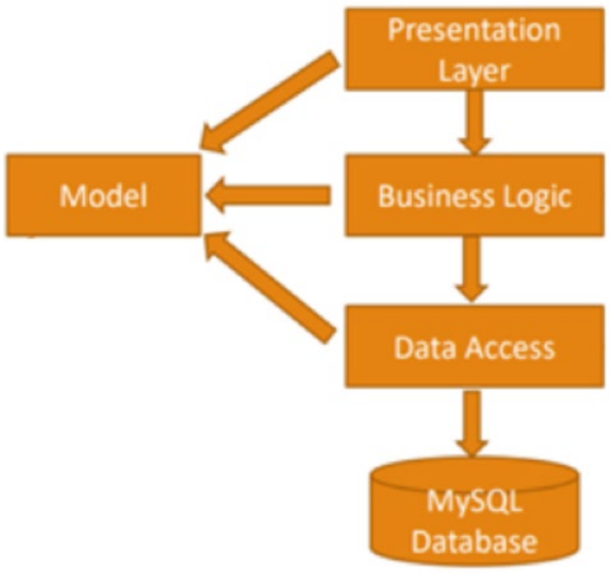
- Ease of use: the user interface should be simple and intuitive, with labels that indicate what the application is expecting from the user
- Compact design: the application's graphical interface should not have a greater size than necessary but should be big enough to display all the information that has to be shown
- The response time for each transaction should not exceed 2 seconds and should perform instantly for most of the transactions
- Reliability of database: the system should use a reliable database management system

3. Conceptual architecture

The code is written in OOP style, following the 4 pillars of Object-Oriented Programming paradigm: Encapsulation, Abstraction, Polymorphism, Inheritance.

4. Design and Implementation

4.1. Package diagram

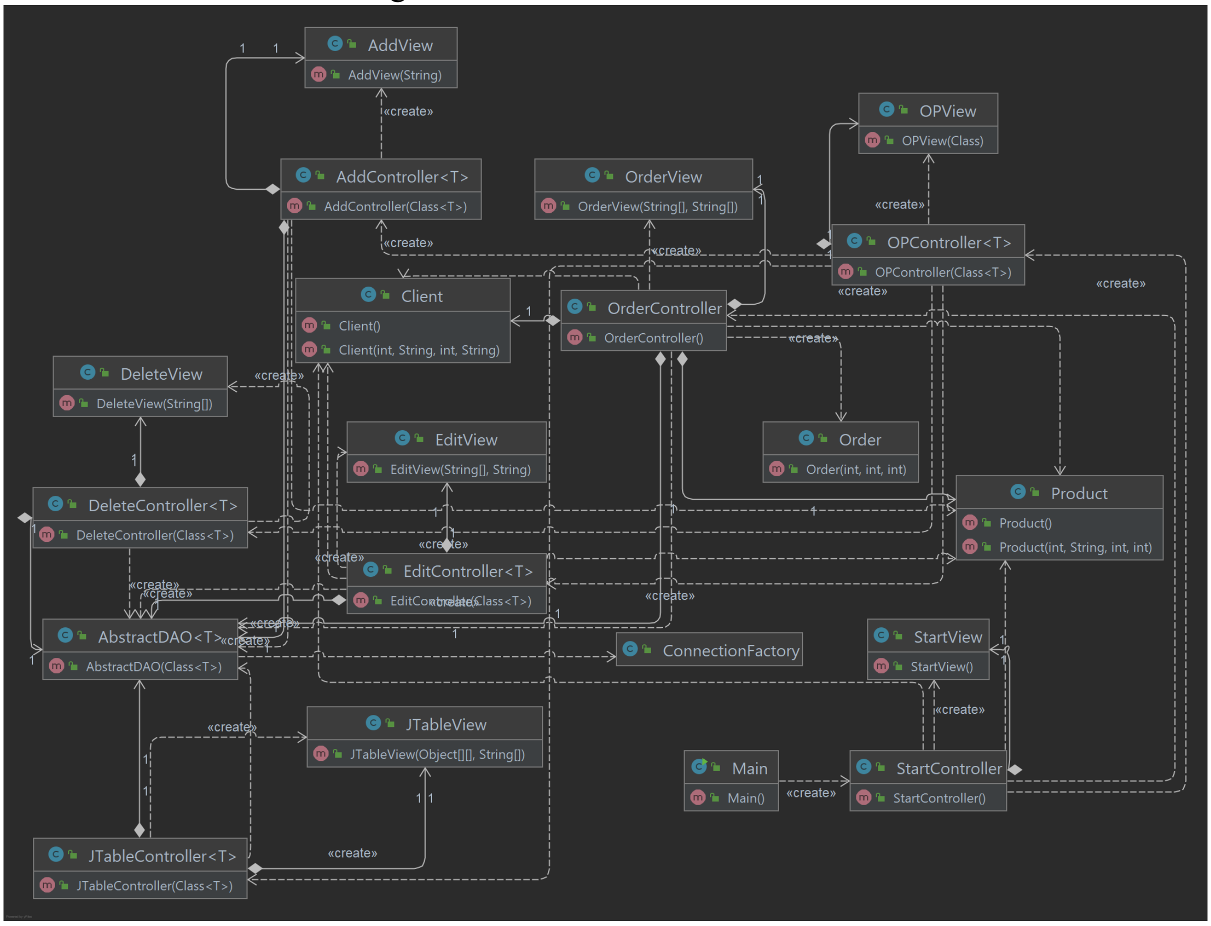


Layered architecture is considered the most common and widely used architectural framework in software development. It is an n-tiered pattern where the components are organized in horizontal layers. They work together as a single unit of software, but do not depend on each-other.

There are four layers in this architecture:

- The presentation layer – is the user interface and communication layer of the application, where the end user interacts with the application. Its main purpose is to display information to and collect information from the user. In the current application, it receives which is the next step the user wants to take and the data input (if it is the case). It also displays new windows bringing the information the user wanted.
- The business logic - handles the business rules, calculations, and logic within an application which dictate how it behaves. It performs data validations and decides which will be the next step taking into account the input from the user. It also communicates with the database in the sense that it uses the methods form the data access layer to perform specific operations on the database.
- Data Access – contains methods for accessing the underlying database data. It implements CRUD operations and supports any class due to the generic implementation of the methods. Another important aspect is the connection between the application and the database. It uses a singleton class that makes sure that only one connection is made to the same database.
- Model – this is where all the data structures are implemented. There is a class for clients, products and orders respectively.

4.2. Class Diagram



4.3. Classes implementation

4.3.1. Model

All classes from this package have the same fields as the tables from the database that correspond to each class.

Client – has an id, a name, an age, and an address. There are getters and setters and 2 constructors.

Product – has an id, a name, a cost, and a quantity. Getters and setters and also 2 constructors same as in Client

Order – has the id of the client and the one of the products and also the quantity ordered by the client.

4.3.2. Database access

Has two classes. ConnectionFactory class makes sure only one instance of the connection to the database is made during the execution of the program. It is a singleton class.

AbstractDAO implements methods that allow CRUD operations on the database. This class contains generic methods that also use reflection techniques such that the same code of the same method is used to perform same operation on the database, but any class is accepted. Reflection techniques assure extractions of the values for each filed of the given object and also the names and type of those fields. This technique brings reusability to the code since it can be used for any object type. The methods implemented in this class are used in the controller package to communicate with the database and perform the tasks requested by the user.

4.3.3. Controller

StartController – creates a new StartView and opens a new controller depending on the button that was pressed. From here the user can choose between client and product operations or placing a new order.

OPController – is a generic class that creates a new OPView window that brings CRUD options to the user. This controller is accessed via any of the product or client operation buttons. The class (Client or Product) is received from the StartController and is passed further to other generic classes that perform the CRUD operations. From here the user can choose between the operations available in the application: add, edit, delete, view all. Depending on the class passed as parameter, the operations displayed will have some differences.

AddController – is a generic class that creates a new AddView window. Depending on the type of action selected previously in the OPController, a view for the client addition or for the product addition is displayed. Here the user can introduce a new client or product and it will be added to the database if the data passed all the validation tests. A new random id is given to each client or product entered by the user. The addition is "sealed" when the user presses the submit button.

EditController – is also a generic class that creates a new EditView window. The user can choose which client or product should be edited and inserts data that is validated in this class. Should the tests be passed, the data is updated in the database and also on the JComboBox displayed in this window. Again, the submit button will perform the task.

DeleteController – is a generic class that creates a new DeleteView window. This class is very simple as it doesn't need to perform any validations to the data. The chosen element from the JComboBox is deleted from the database when the submit button is pressed. A confirmation message will be displayed.

JTableController – a generic class that creates a JTableView window. Here all the data from one of the clients or products tables will be displayed as a table also. Reflection techniques are again used to create the tables. Only a list of filed names and one of values needs to be passed to the JTableView class to display the data. Those lists are acquired from the database using the findAll method implemented in AbstractDAO.

OrderController – here the user can choose one client and one product, enter a quantity, and press the submit button to finish the order. A PDF file will be created with the details of the order such as the name of the client, the name of the product and the date and time of purchase alongside the quantity. A check is made here at the quantity. The ordered quantity should be smaller than the quantity on the stock. If the client needs to order more than the quantity available, a message will be displayed letting the user know a smaller quantity should be introduced and the max quantity is also displayed on that message box.

4.3.4. Presentation/View

StartView – a view that displays 3 buttons that will decide what will be done next: client operations, product operations, inserting a new order.

OPView – a view displays operations with names based on the class given by the StartController. It has 4 buttons representing the 4 basic operations that can be made on the database: add client or product, edit an existing client or product, delete a client or product and finally, view all clients or products in a table.

AddView – this view will display some labels and some text fields where the user can introduce the data. A submit button will send the data to the controller where the data will be validated and introduced in the table.

EditView – a JComboBox will give the user the opportunity to choose the desired client/product that should be edited. Again, some labels and some text fields will prompt the user to introduce the new data that will be passed to the controller in order to be validated and updated in the database.

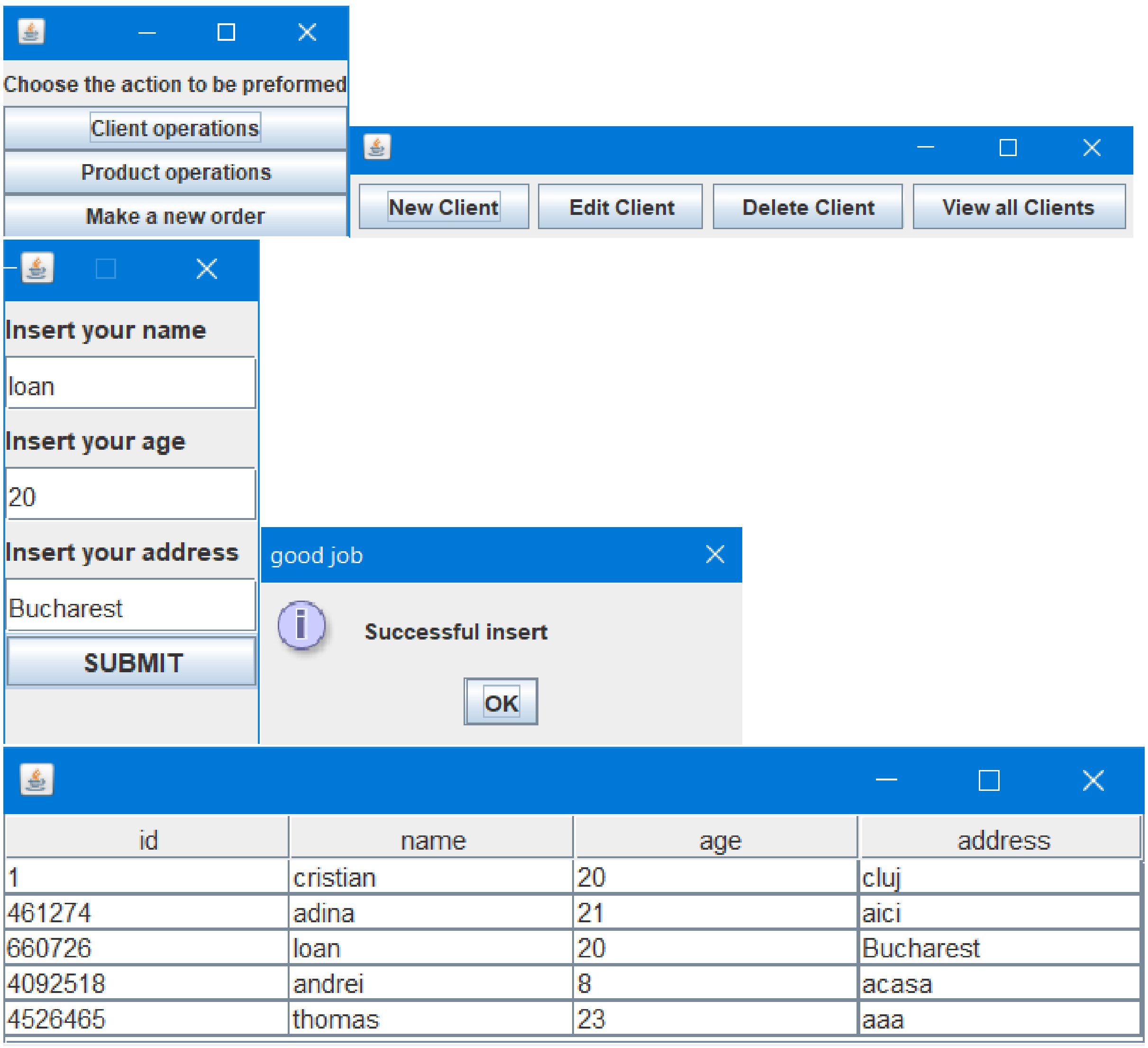
DeleteView – this window has only a JComboBox and a submit button. When the button is pressed, the selected object from the JComboBox is sent to the controller to perform the deletion method.

JTableView – this will display a table given a set of column names (extracted from the class filed names using reflection techniques) and a set of values (extracted using the findAll method from the Database Access layer). This information will be displayed as a table on a new window.

5. GUI presentation

5.1. Successful operations

Below are presented some of the windows that were created during a successful addition of a client to the database and viewing all the clients



Insert your name: Ioan

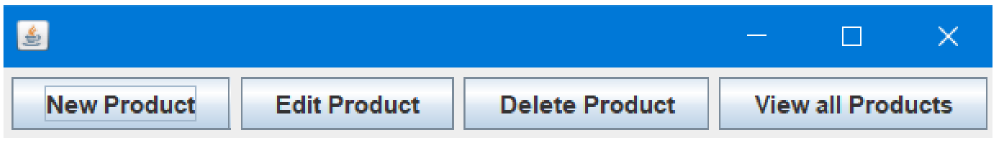
Insert your age: 20

Insert your address: Bucharest

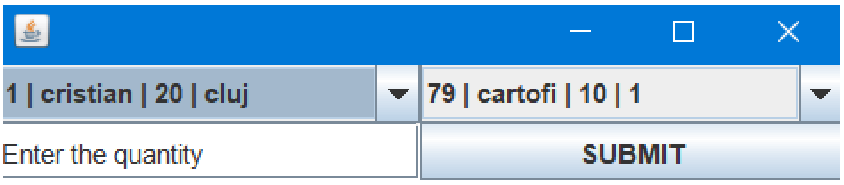
good job

Successful insert

id	name	age	address
1	cristian	20	cluj
461274	adina	21	aici
660726	Ioan	20	Bucharest
4092518	andrei	8	acasa
4526465	thomas	23	aaa



Here is how an order window looks like. The text in the text field should be replaced with the quantity. The user can select any existing client and any existing product form the JComboBoxes.

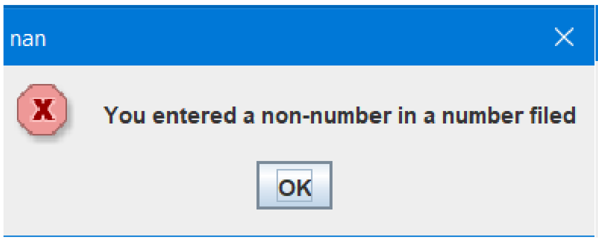

1 | cristian | 20 | cluj

79 | cartofi | 10 | 1

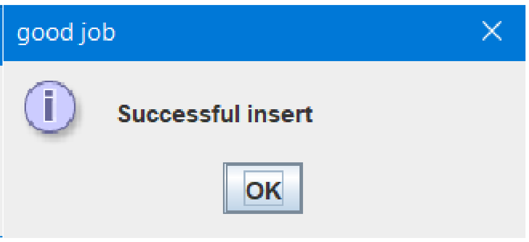
Enter the quantity

5.2 Validation

Here are some examples of the validation tests. If a field that should contain only numerical values contains some non-numeric value, a message will be displayed with a suggestive warning. In case all data is correctly introduced, a confirmation message will be displayed.


nan

You entered a non-number in a number filed


good job

Successful insert

6. Conclusion

This project helped me discover new functionalities in Java programming language, for instance, how to use reflection techniques, generic classes, and methods and about the layered architecture, which creates a cleaner and more decoupled code. Moreover, I learned about MySQL Workbench and how to use it. All the main and secondary requirements have been accomplished. At the end of the assignment, I consider that my Java programming skills evolved since the last project.

7. Bibliography

- the lectures from the course
- support presentation
- Wikipedia
- Geeks for Geeks
- W3schools
- Stack Overflow
- Quora
- freeCodeCamp
- javaTpoint
- Tutorials point
- IBM
- priyalwalpita.medium
- Baeldung
- FourWeekMBA
- EasyTechJunkie
- Academia
- Discovery Science Center