*DOCUMENTATION*

*ASSIGNMENT 3*

*STUDENT NAME: Chiorean Bianca Anamaria*

*GROUP: 3*

# *CONTENTS*

[*1. Assignment Objective 3*](file:///C:\Users\Bianca\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043139)

[*2. Problem Analysis, Modeling, Scenarios, Use Cases 3*](file:///C:\Users\Bianca\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043140)

[*3. Design 3*](file:///C:\Users\Bianca\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043141)

[*4. Implementation 3*](file:///C:\Users\Bianca\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043142)

[*5. Results 3*](file:///C:\Users\Bianca\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043143)

[*6. Conclusions 3*](file:///C:\Users\Bianca\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043144)

[*7. Bibliography 3*](file:///C:\Users\Bianca\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043145)

# *Assignment Objective*

*The main objective of the assignment is to manage orders, to work with data bases and to create a corresponding Graphical User Interface.*

*The sub-objectives are:*

* *Create the corresponding layers*
* *Create the 4 tables: Client, Product, Order, Bill*
* *Create 3 GUI:*
  + *Client*
    - *New client*
    - *Edit client*
    - *Delete client*
    - *View all clients*
  + *Product*
    - *New product*
    - *Edit product*
    - *Delete product*
    - *View all products*
  + *Order*
    - *New Order*
    - *Edit Order*
    - *Delete Order*
    - *View all Orders*
    - *Create Bill*
    - *View all Bills*
* *Link the tables*
* *Connecting the data base to the application*
* *Creating the java classes using reflection technique*
* *Print all the products in JTables*
* *Identify an appropriate way to add/edit/delete entries*
* *Test the application*

# *Problem Analysis, Modeling, Scenarios, Use Cases*

*The functional requirements should be presented together with the use cases (use case diagrams and use case description). The use cases’ descriptions can be done as a flow-chart or as a list containing the execution steps of each use case.*

***The functional requirements are:***

* *The application should allow the user to display first a menu where the user could choose to enter/modify/ delete a customer, product or order.*
* *The application should allow the user to enter a their data as a customer.*
* *The application should allow users to edit a client/product/order based on an id.*
* *The application should allow the user to see all the entries as a JTable.*
* *The application should allow the user to generate a bill.*
* *The application should allow the user to sell all the bills that existed.*

***Use Case****: start application*

***Primary Actor****: user*

***Main Success Scenarios****:*

1. *User enter the inputs the correct fields and the operation is done on the entry.*
   1. *For Client*
      1. *Input the name of the client.*
      2. *Input the email of the client (the email must have a valid form as:* [*nume@email.com*](mailto:nume@email.com)*).*
      3. *Input the id of the already existing client (only for Edit and Delete).*
      4. *User presses a button*
         1. *New button*
            1. *A new record is created.*
         2. *Edit button*
            1. *Based on the provided id the Client record is edited as requested (the input fields are the new values of the record).*
         3. *Delete button*
            1. *Based in the provided id the requested client is deleted.*
         4. *View all* 
            1. *All existing clients are printed in a new window in a JTable.*
      5. *User can exit the client window by pressing X in the top right corner of the window.*
   2. *For Product*
      1. *Input the name of the product.*
      2. *Input the price of the product.*
      3. *Input the stock of the product.*
      4. *User presses a button*
         1. *New button*
            1. *A new record of type product is created.*
         2. *Edit button*
            1. *Based on the provided id and the other text fields, the product is updated.*
         3. *Delete button*
            1. *Based on the provided id the product is deleted.*
         4. *View all button*
            1. *All the product are printed in a JTable in a new window.*
      5. *User can exit the product window by pressing X in the top right corner of the window.*
   3. *For Order*
      1. *Input the id of the order (required for Edit, Delete, Bill).*
      2. *Input the id of the client.*
      3. *Input the id of the order.*
      4. *Input the wanted quantity of the product*
         1. *In case the available product in stock are not enough for the wasted quantity requested by the user, an Exception message is shown and the order in not processed.*
      5. *User presses a button.*
         1. *Order button:*
            1. *A new order is created.*
         2. *Bill button:*
            1. *A bill is created and is displayed as a pop-up containing the date of the order, the name of the client, the ordered product, the price and the ordered quantity of the product.*
         3. *Edit button:*
            1. *Based on the id provided, the order id edited.*
         4. *Delete button*
            1. *Based on the id provided the order id deleted.*
         5. *Log Table Bill*
            1. *Prints all the bills.*
         6. *Show all orders.*
            1. *Shows all the existing orders.*
      6. *User can exit the Order window by pressing the top right x button.*

*2) User can modify something else ( go back to step 1)*

***Alternative Sequenc****e: Invalid values for the setup parameters*

*- The user inserts invalid values for the application’s setup parameters*

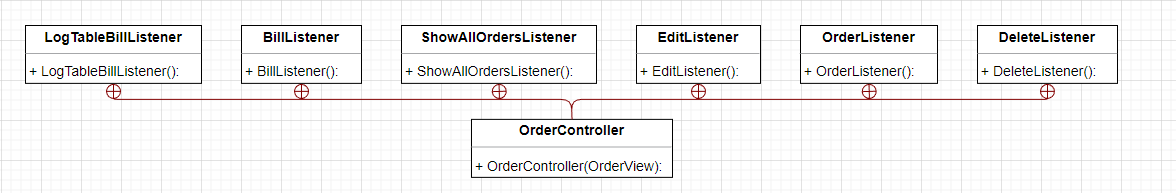
*- The application displays an error message and requests the user to insert valid values*

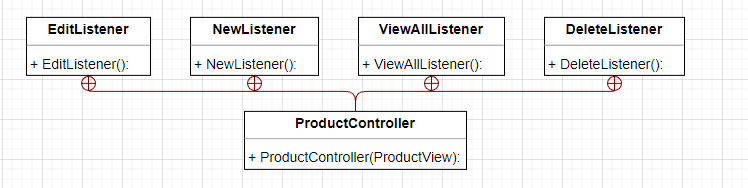
*- The scenario returns to step 1*

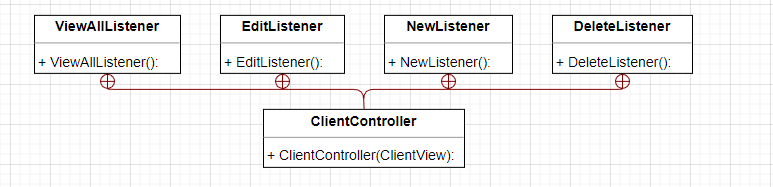
# *Design*

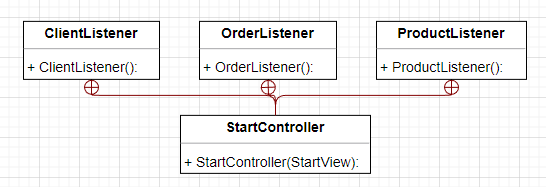
*The following should be presented: OOP design of the application, UML package and class diagrams, used data structures, defined interfaces and used algorithms (if it is the case).*

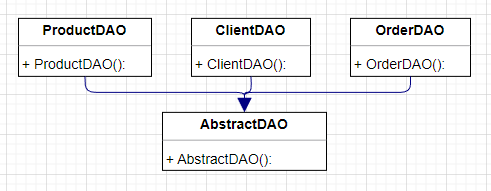
*These are the class diagrams:*

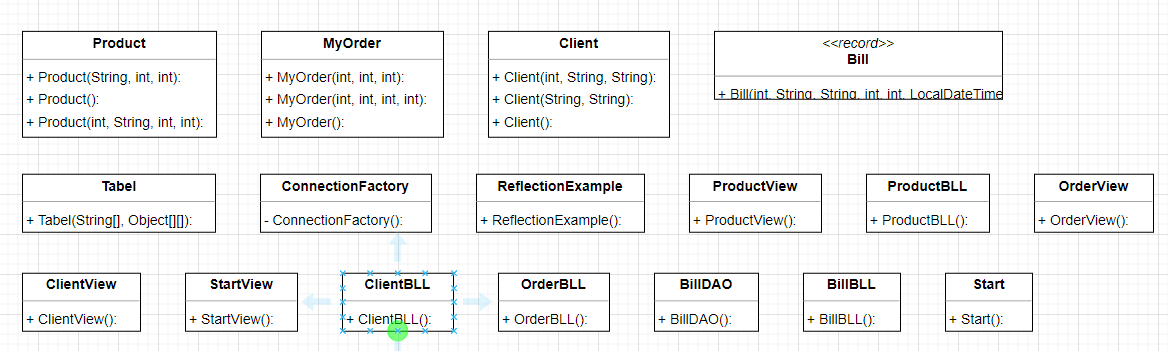
**

**

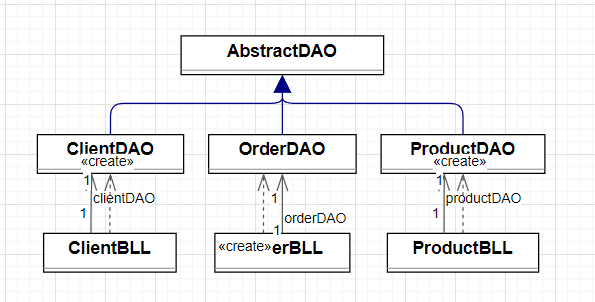
**

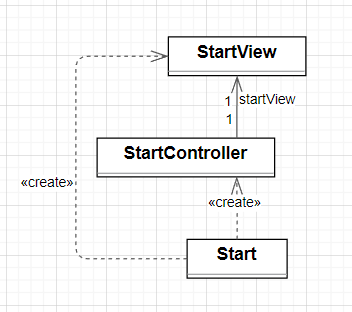
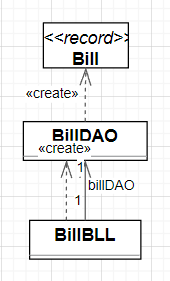
**

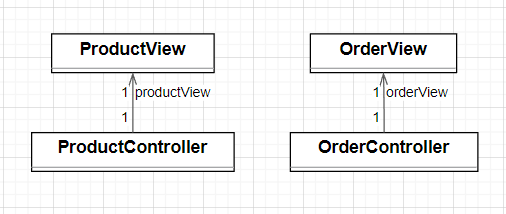
**

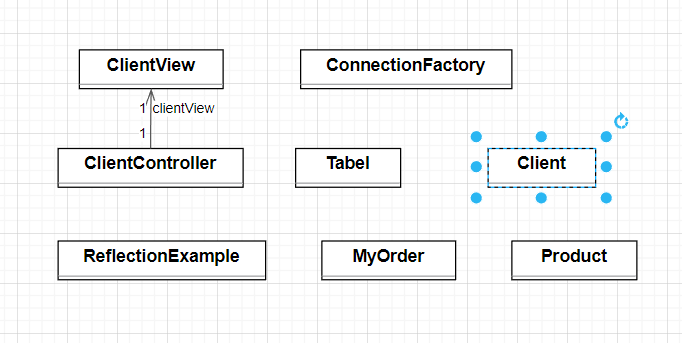
**

*Dependences:*

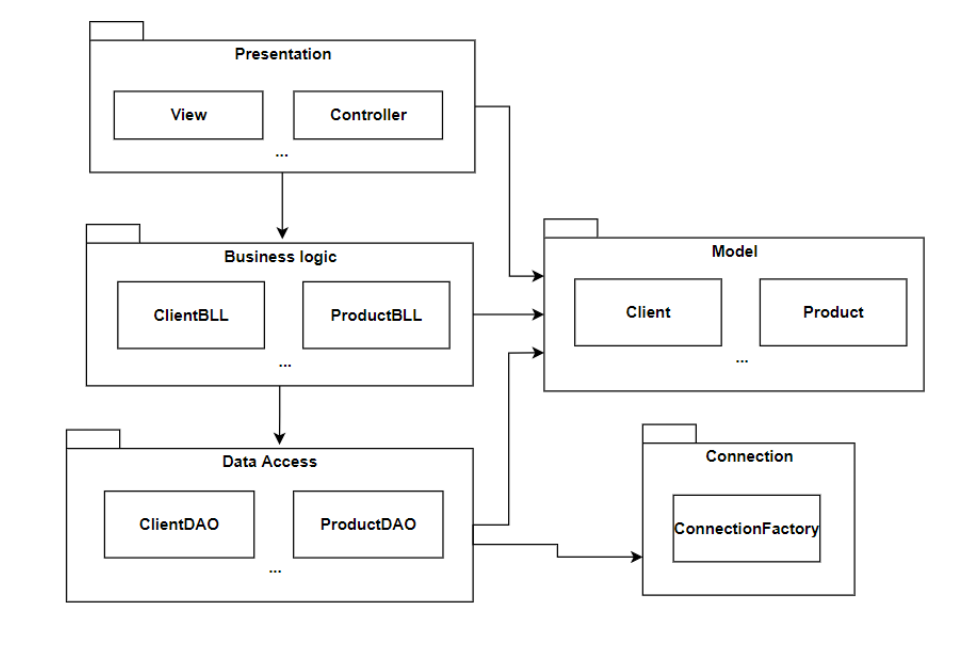
**

** ******

**

******

***Package diagram:***

******

# *Implementation*

***The StartView class*** *represents the start view of an application. It extends the JFrame class and provides a graphical user interface (GUI) for the application's main screen. The class creates a GUI window with buttons for different functionalities and provides methods to retrieve the buttons and attach listeners to them.*

*The class has the following components:*

* *lblNewLabel: A JLabel that displays the text "Order Management" in a large font at the center of the frame.*
* *btnClient, btnProduct, btnOrder: JButton components representing buttons for different functionalities (Client, Product, and Order) of the application.*
* *The buttons are positioned using absolute coordinates (setBounds) and have custom background colors and fonts.*
* *The background color of the frame is set to a specific RGB value.*
* *The layout of the frame is set to null, meaning the components are manually positioned.*
* *The frame is made visible using setVisible(true).*
* *The class provides getter methods (getBtnClient, getBtnProduct, getBtnOrder) to retrieve the button instances. It also provides listener methods (clientListener, productListener, orderListener) to attach ActionListener objects to the buttons, allowing actions to be performed when the buttons are clicked.*

***The OrderView class*** *is a GUI view class that represents the view for managing orders in an application. It extends the JFrame class.*

*The class has the following components:*

*textFieldProductId, textFieldClientId, textFieldQuantity, textFieldId: JTextField components used for inputting order-related information such as product ID, client ID, and quantity.*

*btnBIll, btnOrder, btnLogTableBill, btnEdit, btnDelete, btnShowAllOrders: JButton components representing buttons for different actions related to order management (Order, Bill, LogTable Bill, Edit, Delete, Show All Orders).*

*The background color of the frame is set to a specific RGB value.*

*The layout of the frame is set to null, meaning the components are manually positioned.*

*The frame is made visible using setVisible(true).*

*The class provides getter and setter methods for accessing and modifying the values of the text fields and buttons. It also provides several listener methods (orderListener, billListener, logTableBillListener, editListener, deleteListener, showAllOrdersListener) to attach ActionListener objects to the corresponding buttons.*

*Additionally, the class has methods for displaying error messages (showErrorMessage) and regular messages (showMessage) using JOptionPane dialogs.*

***The AbstractDAO (Data Access Object) class*** *that provides common functionality for accessing and manipulating data in a database.*

*The class has the following components:*

* *The class is defined as AbstractDAO<T>, where T represents the type of the entity being accessed.*
* *The class imports various dependencies, including connection.ConnectionFactory and several classes from the java.sql package.*
* *The class declares a Logger object for logging purposes.*
* *The class has a constructor that retrieves the actual type of the entity being accessed using reflection.*
* *The class contains several private methods that assist in creating SQL queries for different operations, such as createSelectQuery, createInsertQuery, createSelectAllQuery, updateQuery, and deleteByIdQuery.*
* *The class provides public methods to perform common database operations:*
* *findAll: Retrieves all entities of the specified type from the database.*
* *findById: Retrieves the entity with the specified ID from the database.*
* *insert: Inserts the specified entity into the database.*
* *update: Updates the specified entity in the database.*
* *delete: Deletes the specified entity from the database.*
* *The class includes a private method createObjects that maps the retrieved ResultSet to a list of objects of the specified entity type.*
* *The class uses the ConnectionFactory class to obtain and manage database connections.*

***The Bill class***

* *The class is declared using the record keyword, followed by the class name (Bill in this case).*
* *The class defines several fields, each of which represents a data attribute of a bill:*
* *orderId (type: int): Represents the unique identifier of the order associated with the bill.*
* *clientName (type: String): Represents the name of the client associated with the bill.*
* *productName (type: String): Represents the name of the product included in the bill.*
* *productPrice (type: int): Represents the price of the product.*
* *quantity (type: int): Represents the quantity of the product in the bill.*
* *timestamp (type: LocalDateTime): Represents the timestamp indicating when the bill was created.*
* *The record class automatically generates several methods based on the fields:*
* *Constructor: A constructor is automatically generated that takes arguments for each field in the same order they are defined.*
* *Getters: Getter methods are automatically generated for each field, allowing access to the field values.*
* *The record class promotes immutability, meaning that the field values cannot be modified after the object is created. To access the field values, you can use the generated getter methods.*

# *Results*

*The testing was done in the following manner:*

* *First, a valid objected was inputted*
* *Second, an invalid object was inputted*

*These steps were followed for each category (client, product, order).*

# *Conclusions*

*Developing this application I learned to prioritize my tasks, work in parallel (developing the interface and the operations), divide the work load, how work with abstract DAOs, use SQL statements, use JavaDoc, implement a JTable. I observed how important are the comments in the code for documentation, the importance of suggestive variable, methods, classes names.*

*.*

# *Bibliography*

1. *Bruce Eckel, Thinking in Java (4th Edition), Publisher: Prentice Hall PTRUpper Saddle River, NJUnited States, ISBN:978-0-13-187248-6 Published:01 December 2005.*
2. *What are Java classes? -* [*www.tutorialspoint.com*](http://www.tutorialspoint.com)
3. [*https://www.geeksforgeeks.org/java-swing-jtable/*](https://www.geeksforgeeks.org/java-swing-jtable/)
4. [*https://dsrl.eu/courses/pt/*](https://dsrl.eu/courses/pt/)
5. [*https://www.baeldung.com/javadoc*](https://www.baeldung.com/javadoc)
6. *https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html*