DOCUMENTATION ASSIGNMENT 4

STUDENT: VĂLCĂUAN LAURA-MARIA

GROUP: 30422

Table of contents

1.OBJECTIVE OF THE TOPIC

The objective of the theme is the implementation and design of a management system for a restaurant with a dedicated graphical interface designed to facilitate and data entry (customers, products, orders) by the user as well as the possibility offered administrator to modify or delete existing entries. The management system must be easy to use and easy to understand for any user, regardless of their level of training in the field technology.

The sub-objectives considered in the implementation of the main objective are:

1. Separating the problem into easy-to-implement structures: making classes and packages that will take part in the formation of the computing system;
2. Making graphical interfaces: creating interfaces that will be used by user to view and process data;
3. Implementation of a method to convert and store data entered by the administrator;
4. Extract data from the file via Stream objects;
5. Validate the input data introduced by the user;
6. Implementation of certain functions using lambda expressions;
7. Testing for certain situations.

2.PROBLEM ANALYSIS, MODELING, SCENARIOS, USE CASES

A lambda function (anonymous function) is a function defined and called without being linked to an identifier. Lambda functions are a form of nested functions in the sense that allow access to the variables in the domain of the function in which they are contained.

Lambda expressions allow us to create instances of single-method classes in a much more compact way. A lambda expression consists of a comma-separated list of formal parameters enclosing possibly enclosed in round brackets, a directional arrow (->), a body consisting of an expression or a block of statements.

A functional interface is any interface containing only one method. Because of this we can omit the method name when implementing the interface and we can eliminate the use of anonymous classes. Instead we will have lambda expressions. An interface is annotated with @FunctionalInterface. To understand how to work with lambda expressions we built a small example where we created collections of objects sorted by various criteria. The implementation of the Comparator interface was done in an anonymous class, using lambda expressions. The implementation with lambda expressions was possible because in version 8 Comparator is annotated with @FunctionalInterface.

Introduced in Java 8, the Stream API is used to process collections of objects. A stream is a sequence of objects that supports various methods that can be pipelined to produce desired results. Java stream features are:

A stream is not a data structure, but takes input from collections, arrays or I/O channels.

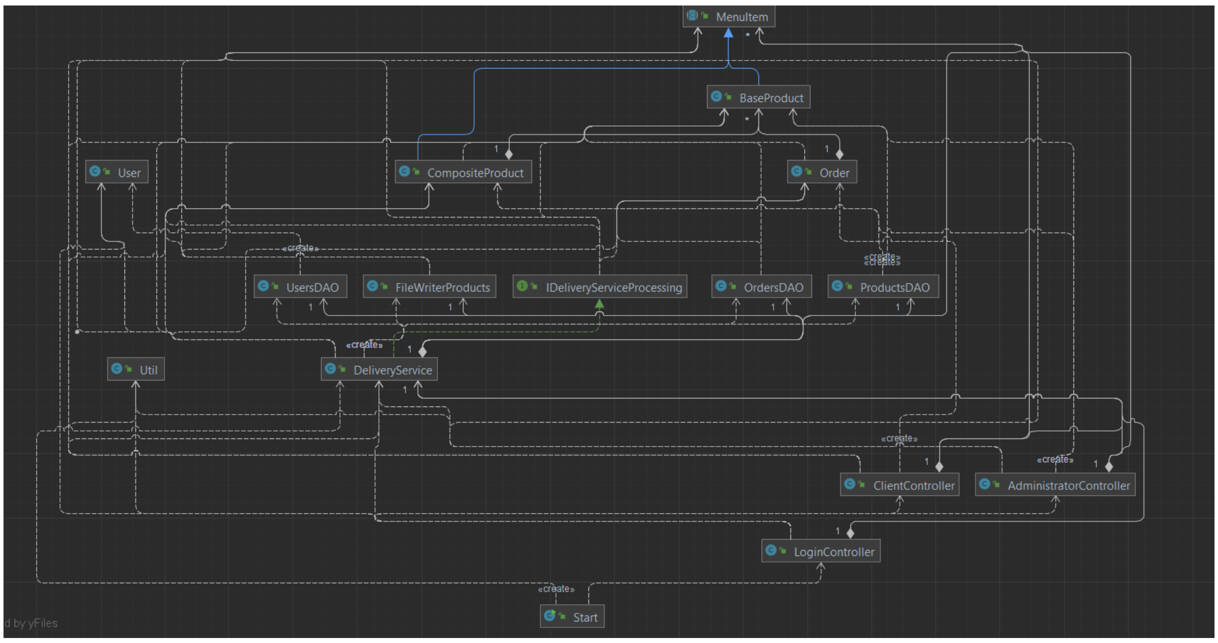
Streams do not modify the original data structure, they only provide the output according to pipeline methods.

Each intermediate operation is executed lazily and returns a stream as a result, via therefore various intermediate operations can be pipelined. Terminal operations mark the end of the flow and return the result.

Serialization is a method by which data can be saved together in a unitary manner. with the signature of an object. Using this operation one can save in a file, as a byte string, an instance of a class, at any time during execution. The object can also be restored from the file where it was saved following a serialization operation.

Design by Contract is an approach to designing robust but simple software. It provides methodological guidelines for achieving these goals without resorting to defensive programming. Instead, Design by Contract builds class invariants and pre/post validation of method arguments and returns values in the code itself.

z3.DESIGN



The application contains several main classes that are intended to model the application as well as the graphical interface and the tables in which the information stored in the database will be displayed.

What is GUI in Java? GUI (Graphical User Interface) in Java is an easy-to-use visual experience builder for Java applications. It is mainly made of graphical components like buttons, labels, windows, etc. through which the user can interact with an application.

For the implementation of this application, the classes have been organized as several models that are based on the transmission of information from one layer to another (class packages).

This model facilitates program maintenance and ensures data security and integrity by the fact that within the BusinessLogic package, the information entered by the user is verified before being updated or entered into the data table. The AdministratorGUI window will be similar to the ClientGUI, the only difference being that the administrator has the ability to perform several actions to modify the table and the data contained in it. The JavaFX TableView enables you to specify the default sort order of a TableView. The items in the TableView will be sorted according to this order - until the user clicks some TableColumn headers and changes the sort order.

The default sort order consists of an ObservableList of TableColumn instances. Items in the TableView will be sorted after the sorting set on the first TableColumn in this list. If the values of the first TableColumn are equal, the items will be sorted according to the sorting set on the second TableColumn, and if they are equal then the third TableColumn etc.

JavaFX ListView is a class used to choose one or more choices from the list. ListViewclass available within scene.control.ListView package. ListView allows us to add as many elements as we want. ListView can also be allowed the user to add elements horizontally or vertically. ListView can be allowed to add images in the list values. ListView used to select single or multiple values at a time.

Button class is a part of JavaFX package and it can have a text or graphic or both. When the button is pressed an Action Event is sent. This Action Event can be managed by an EventHandler. Buttons can also respond to mouse events by implementing an EventHandler to process the MouseEvent.

Introduced in Java 8, the Stream API is used to process collections of objects. A stream is a sequence of objects that supports various methods which can be pipelined to produce the desired result. A stream is not a data structure, instead it takes input from the Collections, Arrays or I/O channels.

A stream does not store data and, in that sense, is not a data structure. It also never modifies the underlying data source.

This functionality – java.util.stream – supports functional-style operations on streams of elements, such as map-reduce transformations on collections.

4.IMPLEMENTATION

In order to implement the chosen model for the application, we divided the classes into several packages, depending on their functionality and role:

- dataLayer: in this package we have defined 2 classes

- Serializer and FileReader.

ClassSerializer class has 2 methods - writeObject and readObject. These are intended to serialize and deserialize the DeliveryService object and write it, respectively read it from the file. The FileReader class has several methods that are used to read from the original file, all the products that we will insert into the table. Reading from the file is done via objects of type Stream.

- businessLogic: Inside the methods in the classes of this package is implemented logic of the application and how data is passed to the user through the GUI. Also, here will also be implemented Composite Design Pattern objects and objects of type Observable. An important component of this package is the lambda expressions and their functionalities as well as their compact form. An example of a lambda expression from is the search function.

- presentationLayer: this part of the application deals with the retrieval of data transmitted by the previous layer and processing it so that it takes a user-friendly form. In the case of this application, the data passed as a list of objects is placed in a table, and then displayed in a separate panel. The user can use a variety of buttons to select the type of operation they wish to perform. After selecting an operation, the user can enter any values in the fields specific to each table.

5.RESULTS

As a result we obtained an application for processing data from a file via Stream objects through which users have the possibility to view the data in a compact and easy to understand way. Verification of the application and of the data entered in the table is done through assertions and various tests whose result is transmitted and displayed in the graphical interface.

6.CONCLUSIONS

In this project I had the occasion to learn some new techniques of programming. It was a little hard at the beginning, everything was so confusing and so many tasks to do but the truth is it was not so bad and I am proud to say I realized and implemented this project. I improved my skills and knowledge about lambda expressions, learned about serializations and how to work in JavaFX. I wanted to try something new because I am a swing user in general.

7.BIBLIOGRAPHY

- https://www.baeldung.com/java-serialization#:~:text=Serialization%20is%20the%20conversion%20of,or%20transfer%20over%20a%20network.
- https://www.tutorialspoint.com/java/java_serialization.htm
- https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html
- https://openjfx.io/
- https://www.javatpoint.com/java-lambda-expressions
- https://www.geeksforgeeks.org/lambda-expressions-java-8/
- https://www.baeldung.com/java-observer-pattern
- https://www.tutorialspoint.com/design_pattern/observer_pattern.htm
- https://www.tutorialspoint.com/design_pattern/observer_pattern.htm
- https://www.geeksforgeeks.org/stream-in-java/#:~:text=Introduced%20in%20Java%208%2C%20the,to%20produce%20the%20desired%20result.&text=A%20stream%20is%20not%20a,Arrays%20or%20I%2FO%20channels.
- https://stackify.com/streams-guide-java-8/