Technical University of Cluj-Napoca

Laboratory Work – Assignment 2

Order Management

Name: Florea Razvan

Group ID: 30425

Table of Contents

1. Introduction

1.1 Task objective

The objective of this task is rather permissive and is defined as follows: ”Consider an application OrderManagement for processing customer orders. The application uses (minimally) the following classes: Order, OPDept, (Order Processing Department), Customer, Product, and Warehouse. The classes OPDept and Warehouse use a BinarySearchTree for sorting orders”. Due to this reason, the students have full flexibility concerning the implementation and use of resources offered by the Java programming language. It is up to everyone’s decision rather to develop the algorithms in a particular way or another. Same for using a Graphic User Interface or not, or modeling in a specific way or not.

1.2 Personal approach

I personally developed a Java application for processing orders depending on the user’s state. The user may be either administrator or customer. Customers and administrators have different perspectives and therefore, different Graphic User Interfaces and actions that they can perform.

2. Problem Description

2.1 Problem analysis

The problem domain can be divided in smaller problems. In this way we only have to deal with problems of a smaller complexity that together form the main problem, the main objective of our application.

The problem can be seen from different perspectives and different points of view. The modeling of the orders, products and users to be recognized as structures, as independent classes, is very important because it is the core of almost every operation and every function that we want out application to perform. The user must not feel neglected and therefore the graphic user interfaces must not be forgotten and even more, they should be user friendly and efficient. The functionality is the back end “thinking process” of our application and is the most important aspect that we have to take into consideration.

2.2 Modeling

Problem modeling is done by means of object oriented programming and establishing classes, objects and relationships between the components of our problem.

2.3 Scenarios

A variety of scenarios have to be taken into consideration. That is, we must be able to predict what the user wants to do once the application is finished and ready to be used, so it should behave properly of any types of input and any operation that he or she may want to perform.

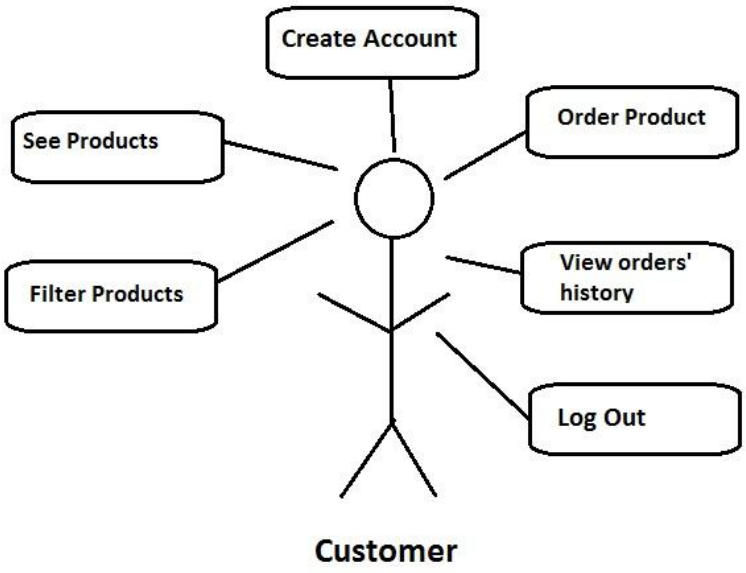
This is why I tried to take into considerations all the possible scenarios that may occur or a customer, as well as for an administrator when it comes to managing orders or stocks of products. Therefore I implemented the following functionalities: administrators are able to create accounts, authenticate as existing users, log out, remove customers, add products to the warehouse, remove products from the warehouse, modify the existing stock of products, see the stock of products, see the orders, see all the customers, manage the customers' orders.

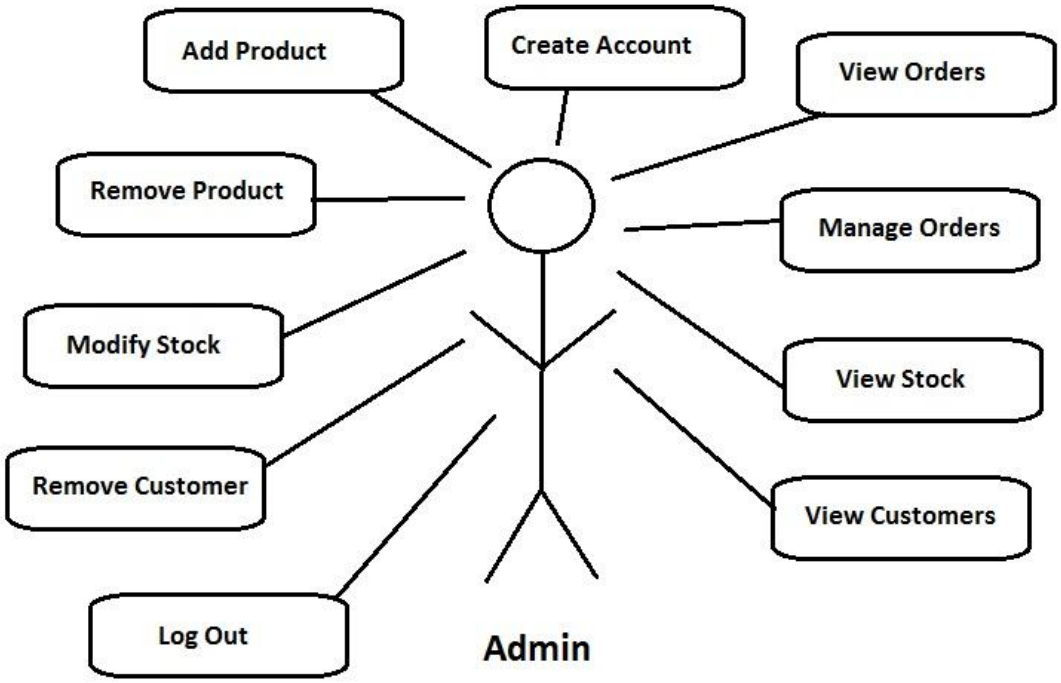
Customers are able to create accounts, authenticate as existing users, log out, see all the products in stock, filter products by make, by type or by price, ascending or descending, depending on what he or she wants, order a product or more, see the order's status, see the history of all of their orders.

3. Projection

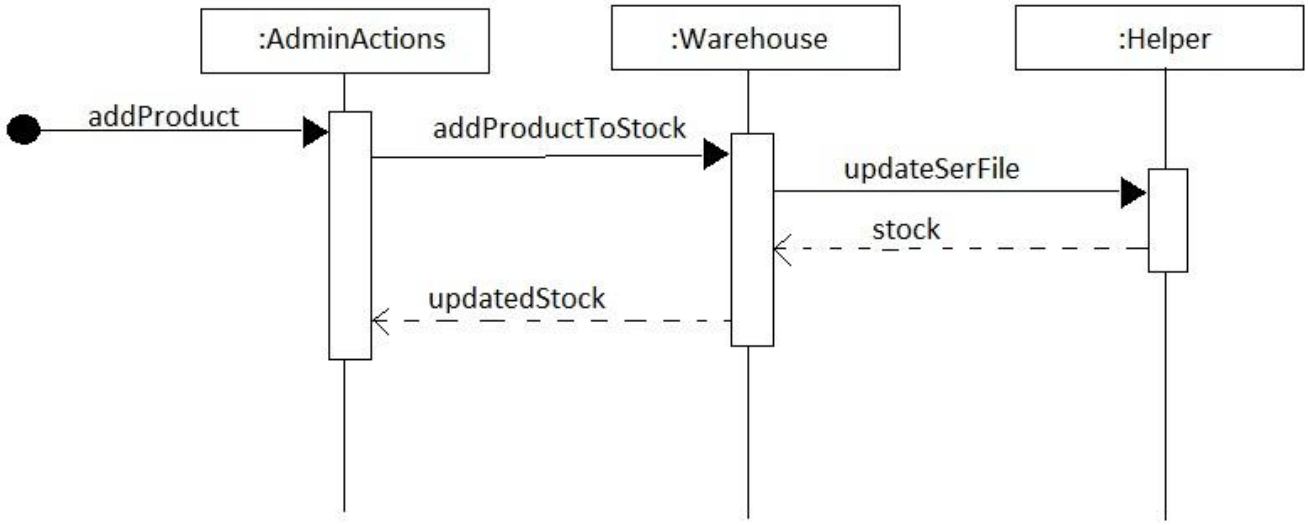
3.1 UML Diagrams

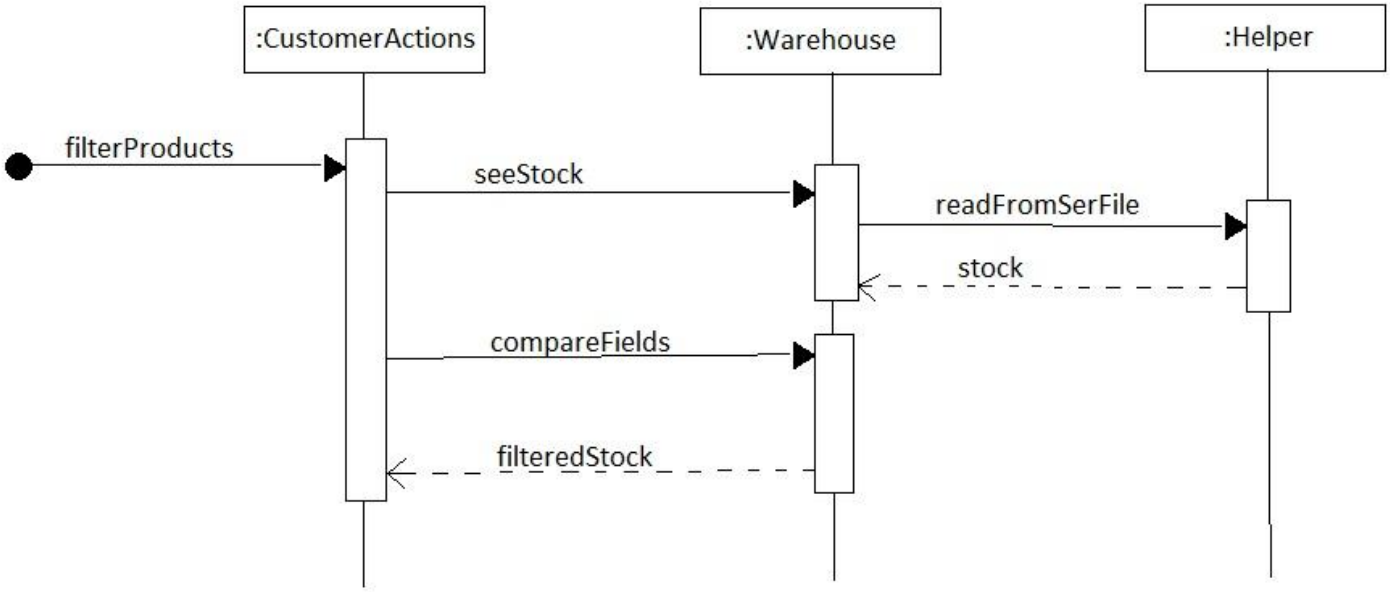
3.1.1 Use Case Diagrams



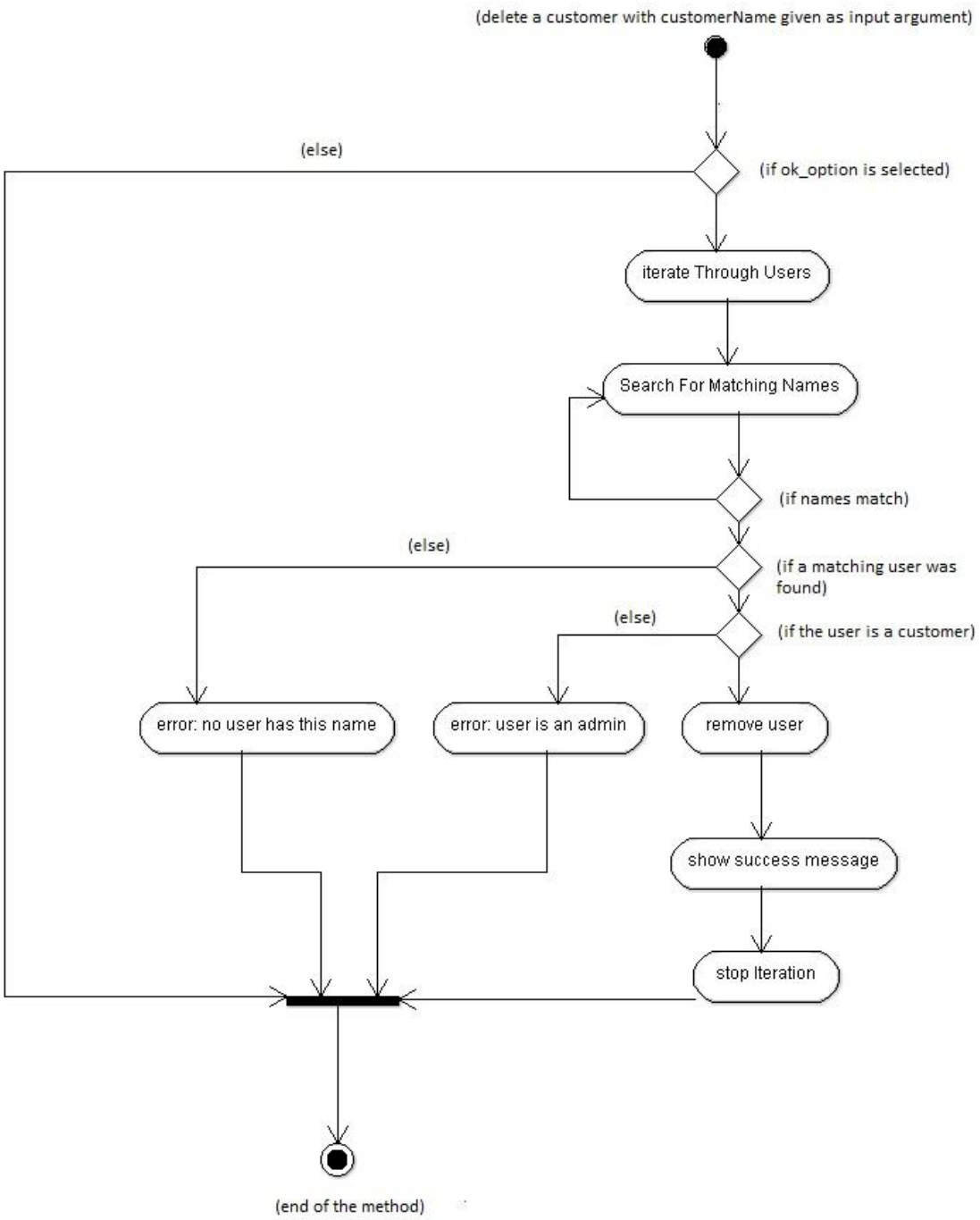


3.1.2 Sequence Diagrams

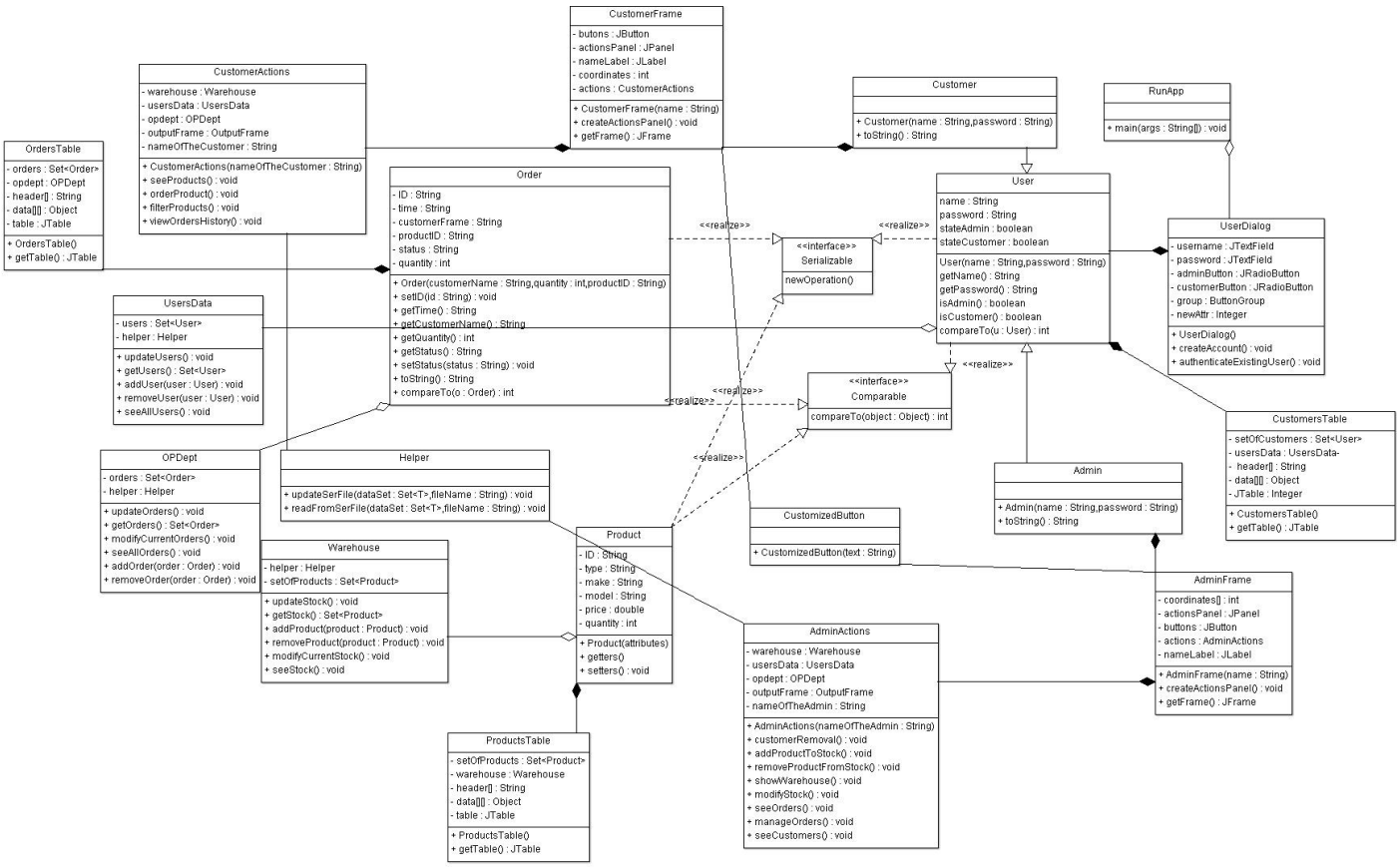




3.1.3 Activity Diagram



3.1.4 Class Diagram



The diagrams show how the problem is divided in smaller problems and how the modeling was done. The classes communicate between each other and therefore different types of relationships are established.

There are associations between the classes sharing the stock, users, and order models, aggregations between the classes that use mutual objects, inheritance from the class User which implements the interface Serializable and Comparable, in order to extend it and create the two types of users: administrators and customers. Compositionsis also used between classes that are absolutely necessary for other classes to work properly and extensions from classes provided by the java language(for example I extended th JButton class to create the CustomizedButton class, I extended the JFrame class in both cases when I created the AdminFrame, CustomerFrame and then the OutputFrame).

For the class diagrams I have used the ArgoUML environment.

3.2 Data Structures

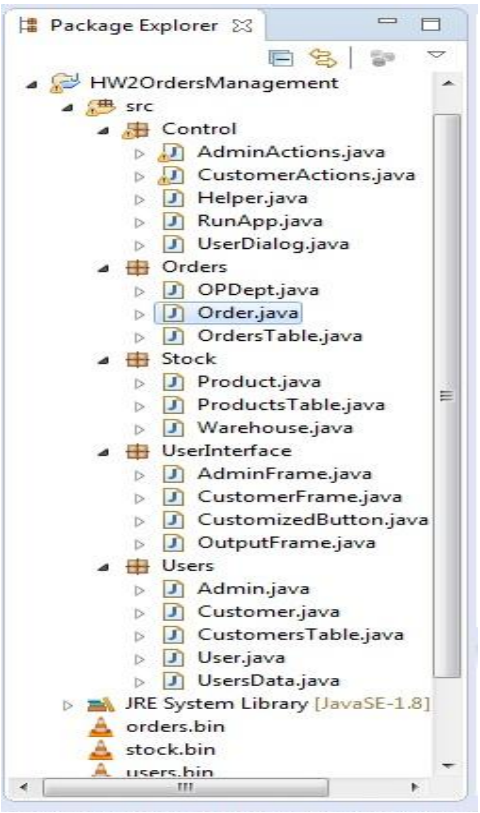
The data structures used are the primitive data types provided by the Java language such as String, integer, double, boolean, font, or other data types such as the customized button, JLabels, JTextFields, JTextAreas, fonts, JFrames.

I also used TreeSet to implement the BinarySearchTrees for sorting and rendering through orders, users or products in the stock.

3.3 Class Projection

The modeling and partitioning of the problem domain is done by means of dividing the problem and data types needed into several classes from which I instantiate objects of that type and use is it in another class. This connection between object of one class and methods of another class is done by means of relationships.

These classes are separated regarding their purpose into different packages. These packages are as follows:



1. Control Package:

The Control package contains the "thinking" done by the applications. That is the functionality of the application and the management of the user input and user dialogs.

It contains the following classes:

- AdminActions.java;
- CustomerActions.java;
- Helper.java;
- UserDialog.java;
- RunApp.java

2. Orders Package:

This package contains the classes responsible with representing the entities associated with the orders. These classes are as follows:

- Order.java;

- OPDept.java;

- OrdersTable.java

3. Stock Package:

This package contains the classes responsible with representing the entities associated with the warehouse. These classes are as follows:

- Product.java;

-Warehouse.java;

- ProductsTable.java

4. Users Package:

This package contains the classes responsible with representing the entities associated with the users. These classes are as follows:

- User.java;
- Admin.java;
- Customer.java;
- UsersData.java;
- CustomersTable.java

5. UserInterface Package:

The UserInterface package contains all the classes responsible for managing a nice display for the user and creating the graphic user interfaces. The classes that this package contains are the following:

- AdminFrame.java;
- CustomerFrame.java;
- OutputFrame.java;
- CustomizedButton.java

The packages communicate and everything work together in order to satisfy the user and all of his or her requirements. The algorithms are quite simple due to the good managements of resources, data available and modeling the problem domain.

3.4 User Interface

The user interfaces that I created for this application are the following ones:

4. Implementing and Testing

Implementation of the algorithms that I have used and modeling the problem was done by means of the Java programming language and the Integrated Development Environment (IDE) that I used is Eclipse. Even though I have not tested the application on another IDE, it should maintain its portability.

I have covered all the possible cases for user input for the dialogs with the one using the application so that no errors should occur to stop the running time of the applications. For this I used a lot of try - catch blocks to handle different exceptions that may occur.

I have tested all the possible cases that may appear when someone is using the application, and it was proven to be safe for everyone and is easy to understand how to interact with the application and the user interface.

5. Results

The Application is user friendly and very helpful in computing polynomials and different operations on polynomials. Being developed in the Java programming language the final product is highly portable and is able to run on different operating systems as long a java development kit is installed. The application is very straightforward and instructions are always given to the user whenever some wrong data is send as an input and is considered as not being valid.

6. Conclusions

This application meets all of its requirements but it can also be enhanced and developed even more. I have learned some new things and also I increased my java knowledge and experience with this programming language.

7. Bibligraphy

http://stackoverflow.com/

https://docs.oracle.com/javase/tutorial/uiswing/components/table.html

http://www.color-hex.com/color-names.html

https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html