Project Report

Car Booking System

ICT Engineering

Group number: 2

Bianca Sgondea (240312)

Nikola Sevo (245494)

Nicholas Papas (245466)

Sergiu-Mihai Craciunescu (240329)

Course: SEP1

Supervisors:

Allan Henriksen

Mona Wendel Andersen

Date: 3 June 2016

Contents

List of figures and tables

Abstract

The main objective of this project is to develop a car booking system for the V-Rent company to help them to manage the reservations information and the vehicles. The system should be able to make the work easier for the employees and the manager to add reservations and bookings, see information about them and the customers. It is important for the user to see certain information about vehicles (make, model and mileage). The rental system must include the option to search through the reservations by a specific criteria (name, address, phone number).

To transform the project from just an idea and then to a finished product, it first needed to be divided into small steps. First step it was to transform the customer's wishes into requirements and right after, in diagrams that are describing the problems that the system should solve. The next step was to design the system. In the last step the final product was created using Java. The data created by the system is saved using binary files on the computer that it is running on.

The result of the project is a vehicle renting system which meets the customer's requirements and can be easily modified in the future to add more features without changing the whole system.

1. Introduction

Managing a vehicle renting company and keeping track of the customers and the reservations is a difficult task to accomplish without making an renting system where the employees can have access to all of that, so that is the reason for which in this project we want to develop a renting system for a renting company.

In the 21st century renting a car is a normal and very habitual thing for a lot of people. In the near past you could've rented a vehicle by going to their office and picking a vehicle that they had available at that particular moment, without knowing what type of vehicle it was or if is the vehicle you wanted. Nowadays the facilities of renting a vehicle are more diversified. It means you can choose the type of vehicle you want to rent, the make, the model and so on.

V-Rent is a vehicle rental company located in Horsens. It is owned by Van Motor who wants to upgrade his company with a rental software system. They are a small company who started a renting business three years ago and they are growing very fast. Because of that they want to upgrade their small company with a rental system where they can keep track of customers, vehicle and reservations.

The developed rental software system allows the employee to: make reservations by first name, last name, address, phone number, pick-up and delivery locations, pick-up and delivery date. Also the employee can see all information (make, model, mileage and/or load size) about the vehicles searching them by type. The employee will be able also to hand out vehicle for a customer who already made the reservation. The system will not allow a vehicle to be rented at the same time and all the data will be stored in a file in order to avoid data loss if the system is shut down. There is also a feature for employees, which allows them to search customer reservations be name, address or phone number. The system is made with a beautiful and easy to use graphical user interface.

The system's purpose is to serve a small car renting company's employees in managing the rents and the vehicles they have.

2. Analysis

During the analysis phase, the requirements must be understood in order to create a system that satisfies the client's needs.

This phase implies the following steps:

- A requirement list
- A use case model
- Use case description
- Activity diagrams

2.1 Requirements

The requirements are the client's wishes for the system.

2.1.1 Functional requirements

1. The employee must be able to rent out vehicles.
2. The employee can reserve a vehicle for a customer.
3. The employee can book a vehicle for a customer immediately.
4. The employee must be able to get a list of all the cars available or rented.

5. During making a reservation the employee should insert in the system: customers name, pick up time, returning time, pick up location, delivery location, an estimate of the amount of kilometres they will drive, type of vehicle.
6. During renting the employee should be able to search for a specific reservation.
7. The employee must be able to see when a car will become available if it is rented.
8. During picking up the car, the employee should be able to insert customers driving license number.
9. During returning a car, the employee should be able to insert the number of kilometers that the car has driven.
10. The system should store in a file all the information (that the employee is inserting) about the rentals, vehicle data and customer data.
11. The system will check the total number of driven kilometers. For every 20.000 km driven it will inform the employee if the car needs to be taken for service check.
12. If a car is booked for several days the system must be able to calculate the price with a discount of 30%, for the following days.
13. The camper cars can be booked only for a weekend or for a full week.
14. The employee should be able to delete a reservation if the client is canceling it.
15. The employee must be able to add or delete cars to the system.
16. The graphical interface system and the way the data is stored could easily be changed later on.

2.1.2 Non-functional requirements

1. The system will run on a single computer.
2. The system should be programed in java.

2.2 Use case diagram

The following diagram describes what the system must be able to do, filing the client's demands. Each of the use cases shown in the diagram represents an action that can be performed by the employee.

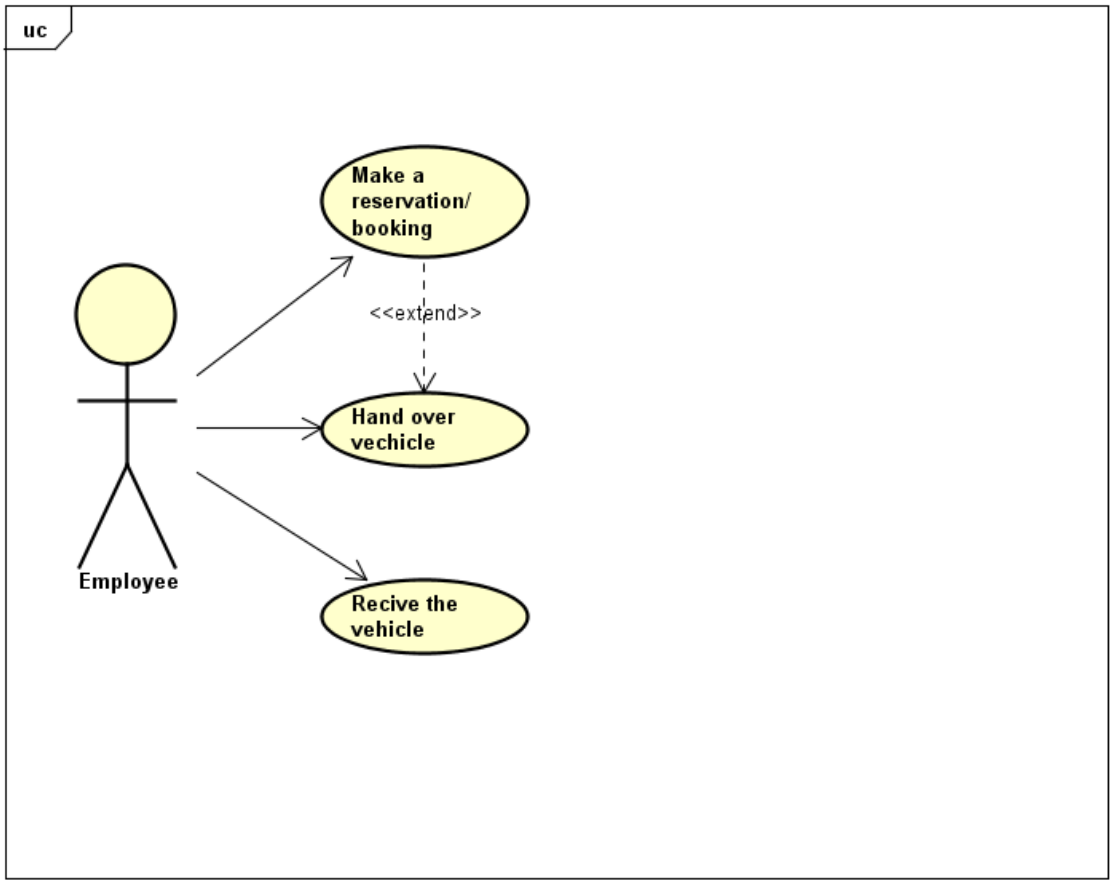

Figure 1: Use case diagram

In the picture above, we have a Use Case diagram which details the possible use cases when the customer representative is using the rental system.

See the use case diagram in Appendix A

2.3 Use case description

The use case description provides more details of every action that can be performed by the employee.

A use case diagram contains:

- A summary in which the action is described.
- The actor shows who is going to use the system.
- Precondition shows the condition before making the action.

- Post condition shows the result after the action is done.
- Base sequence diagram describes the action step by step.
- Exception sequence shows which base sequence could be an exception if the action does not work as excepted.
- Sub use case shows if the use case is connected to another action.

ITEM	VALUE
UseCase	Make a reservation/ booking
Summary	Employee books a car for a customer.
Actor	Employee
Precondition	Customer ask for a booking.
Postcondition	New reservation is created.
Base Sequence	1) Employee selects "Renting info". 2) Employee inserts information about costumer and renting. 3) Employee selects "Next". 4) Employee inserts the renting period and chose a vehicle type. 5) Employee selects "Update the upper list". 6) System prints out a list with all the vehicles of a specific type 7) Employee select a specific vehicle. 8) Employee selects "Save reservation". 9) System checks if the information is valid. 10) System prints out a confirmation "Reservation saved".
Exception Sequence	Inexistent date: 1-9) as Base sequence 8) The system will show warning "Insert information in all fields (*)"
Sub UseCase	Hand over vechicle
Note	

Figure 2: Use case description

The all use case descriptions can be found in Appendix B

2.4 UML Class diagram

The class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, methods and the relationships between objects.

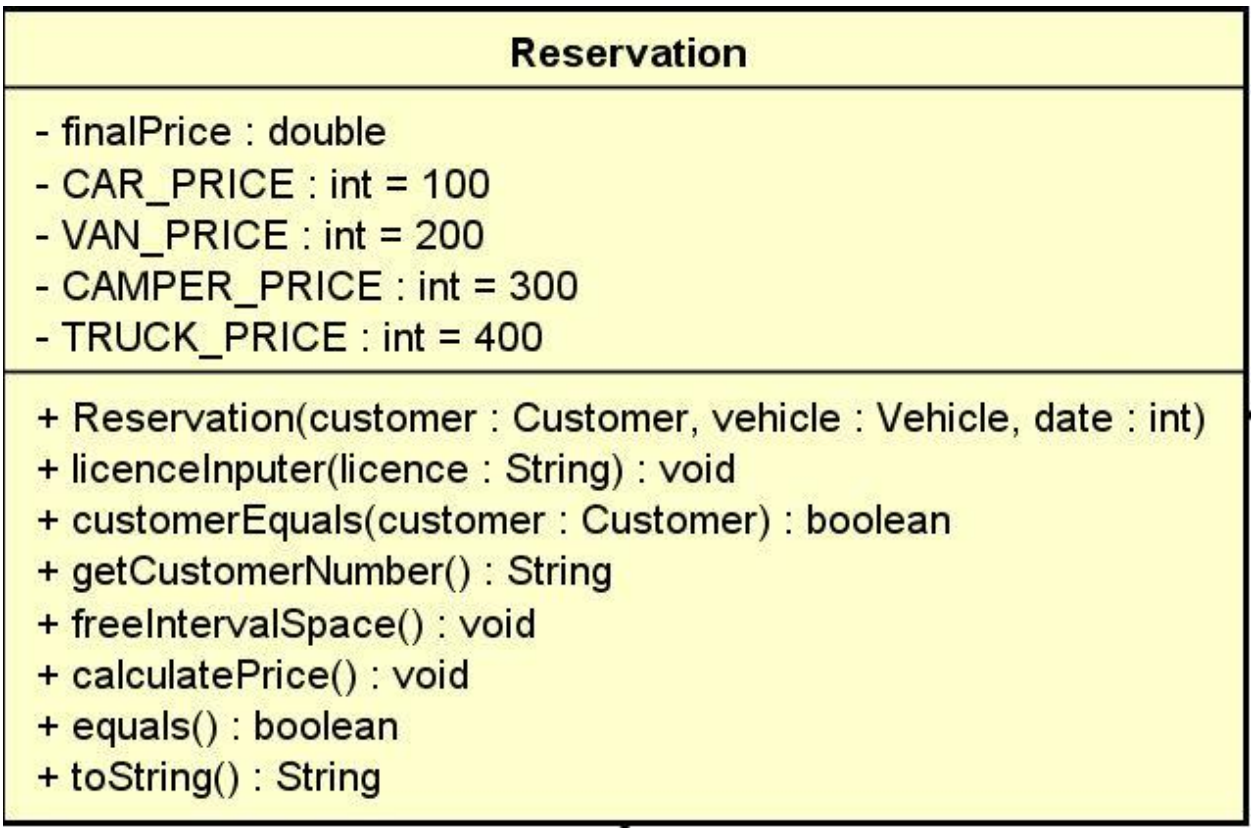

Figure 3: UML Class diagram

The above picture represents the UML class diagram for the "Reservation" class.

See all class diagrams in Appendix C

2.5 Activity diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. Activity diagrams are constructed from a limited number of shapes, connected with arrows.

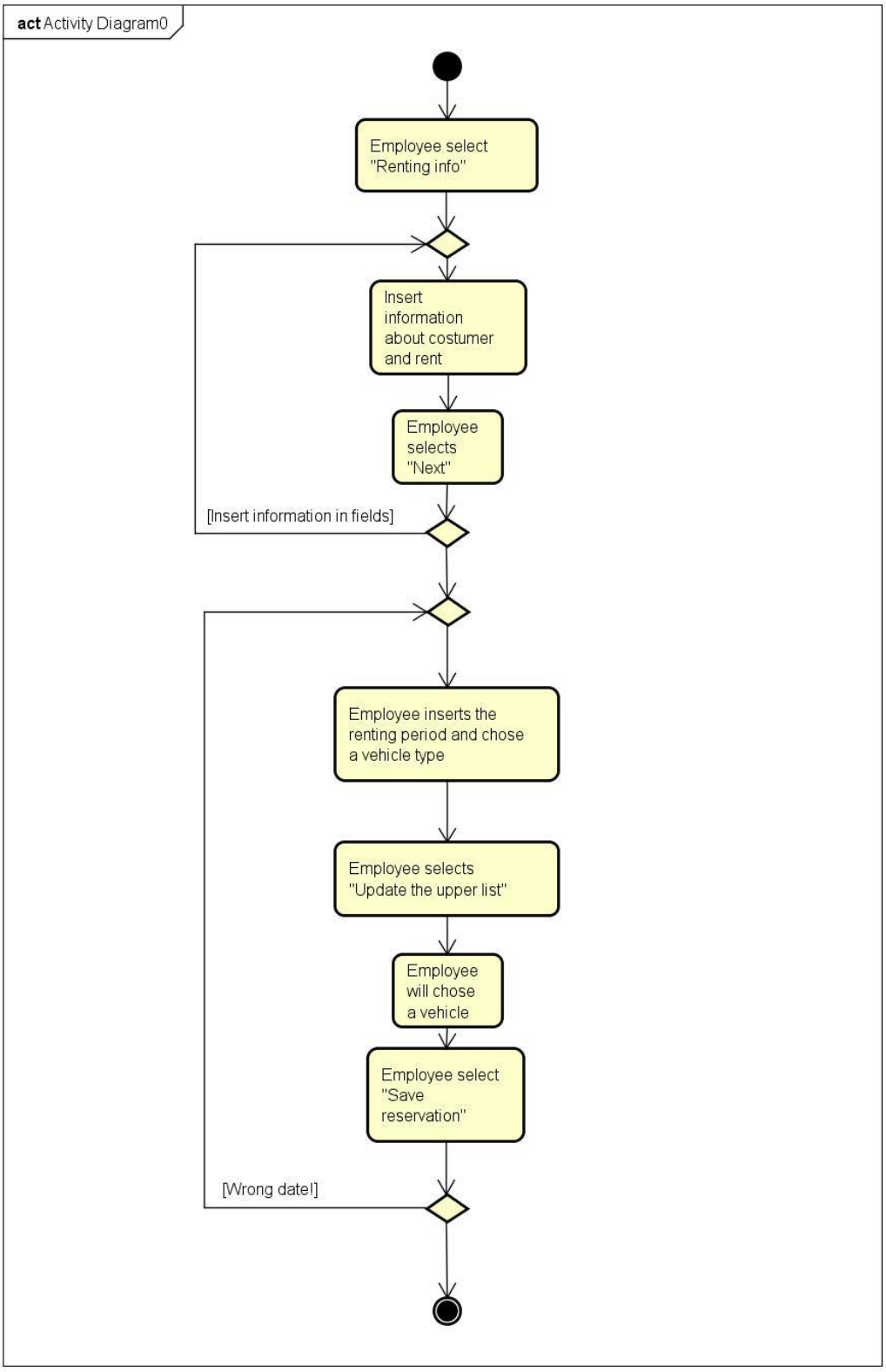

Figure 4: Activity diagram

See all activity diagrams in Appendix D

3. System design

The program consists of 3 parts: Java classes, adapters and the GUI.

Java classes

The Java classes make the functioning of the program possible by creating and controlling necessary Java objects, as well as connecting, retrieving and sending information to the files, through the adapters.

```
 9  public class ReservationList implements Serializable
10  {
11
12     private ArrayList<Reservation> reservations;
13
14     //
15     /**
16      * We add the provided reservation on the first null space in the reservations list
17      * @param reservation Specific Reservation object that we want to add to the list
18      */
19     public ReservationList()
20     {
21        reservations = new ArrayList<>();
22     }
23
24     public int size(){return reservations.size();}
25
26
27     public int indexOfCustomer(String licenseNumber)
28     {
29        for(int i =0;i<reservations.size();i++){
30
31           if((licenseNumber.equals(reservations.get(i).getCustomerNumber()))){
32
33              return i;
34           }
35        }
36        return -1;
37     }
38
39     public void addReservation(Reservation reservation)
40     {
41           if (indexOfCustomer(reservation.getCustomerNumber())==-1){
42              reservations.add(reservation);
43        }
44     }
45
46     public void deleteReservation(Reservation reservation)
47     {
48       if (indexOfCustomer(reservation.getCustomerNumber())!=-1){
49          reservations.remove(reservation);
50        }
51
52     }
53     // * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - *
54
```

Figure 5: ReservationList Class

Above you can see the ReservationList class which contains a list of all the reservations and also contains methods that help handling reservations.

See all source code in Appendix F

Adapters

Adapter is a pattern that provides default implementation of interface or abstract class. His main purpose is to read and write files.

GUI

The GUI (Graphical User Interface) it was created using a simple interface. All the tabs represent the functionalities of the system. For each tab that it's pressed the GUI will change the functionality and will display the specific proper one that it was pressed in order to be able to make an action afterwards and change the data from the file. The functions represent the Use Case of the system, each function works according to Use Case Descriptions.

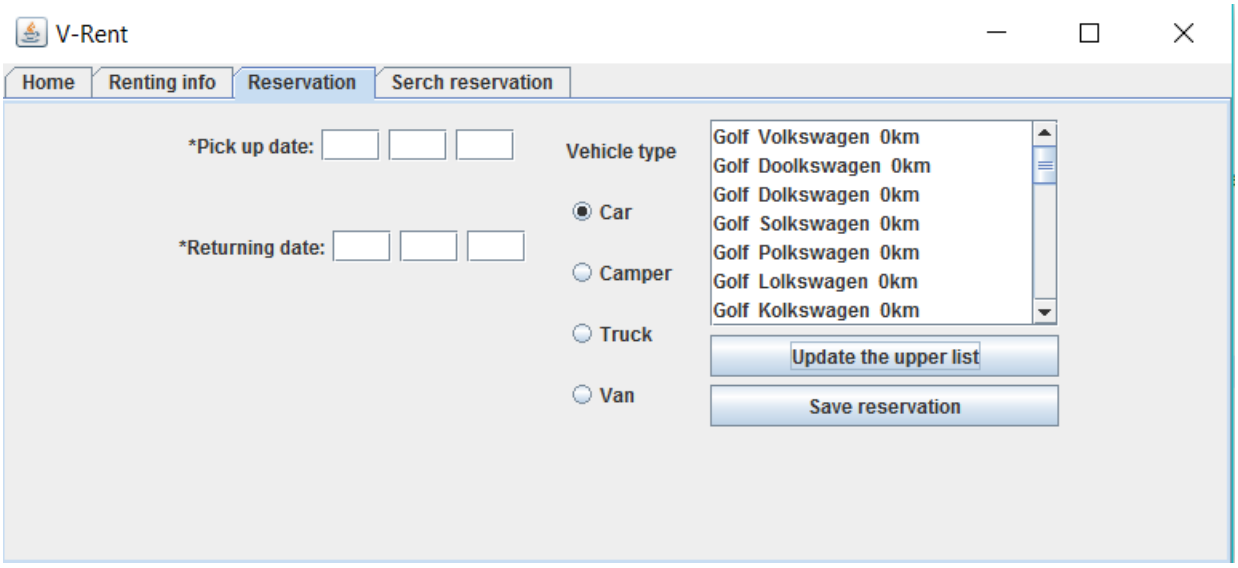

Figure 6: Reservation tab in GUI

Above you can see the main facility window of the system.

See full User Guide in Appendix E

4. Implementation

During the implementation phase some considerations were made for the most important requirements and also for the use case descriptions. The waterfall model was kept in mind and after that the implementation of the classes was started. Once all the classes were implemented the GUI implementing was started and also test the functionality of the classes. Modifications were needed in the class diagram in order to match with the functional code. In order to connect GUI with the Java methods some adapter classes were implemented. For storing information the system is using binary files.

5. Testing

5.1 Add customer/ make reservation

“Add reservation” function is the main and the most important function of the system, because reservations are the most important things for a rental company. The role of this function is to allow the user to add a reservation for a certain vehicle on a certain date with specific information about the customer.

The test of this functions was performed by filling all the fields with information on the second tab “Renting info”. After completing all the fields, the “Next” button is pressed and the next tab, called “Reservation” will appear where you need also to fill up some text fields, choose a type and click on the “Update the upper list” button and the specific models of that kind of vehicles are going to appear in the list. After that you select the model that you want and click on the “Save reservation” button and a small window it’s going to pop up with the message “Reservation saved” and the reservation is saved in the specific file with reservations.

Exception:

If you are not completing all the fields an error message saying “Insert information in all fields (*)” is going to pop out and you cannot move forward to the actual reservation. It’s going to be the same action for the “Reservation” tab also.

5.2 Show reservations

This function takes the data from the files and displays all the reservation with the name, the phone number of customers and also with the price that they have to pay for the rent. The test phase was performed by going on the last tab called “Search reservation”. After that, the user will press on the “Update reservation list” and all the reservations will appear in the list. Here the user can check more information for each reservation.

Exception:

1. We tested this several times and sometimes the list is filled with reservations, sometimes the system is just freezing and nothing can be done, not even selecting the close button for closing the program. The reason of the error could not be found.

2. When you are double clicking on a reservation a new window will pop out with the specific information about the customer. The feature wasn’t implemented because of the time limit.

6. Results

The vehicle renting system created for V-Rent is a simple renting system that will definitely make a positive difference.

As a result we have created a renting system which is able to make reservations for customers and also to show all the reservations that were created. The system is easy to use which is helping the employees and also it was created in order to be easily changed afterwards improving it with new features.

The system does not have for now implemented a method which checks if the dates of the reservations are overlapping and the employee is still not able to search for information about a specific reservation by name or phone. Also the system is not displaying the full information about a specific reservation or client.

There are also some functionalities of the system that were not implemented yet, such as: handing out a vehicle to a customer, not alerting the employee when a vehicle has passed 20.000 km.

7. Discussion

At the beginning of the project we created the system in Java following the requirements that were extracted and created after the interview. One of the system's main functions was to rent an available vehicle but as the work progressed in the project, this function had to be divided in two features - reserve a vehicle from a specific date and book a vehicle immediately.

After the group started testing the system towards the end of our project period, it was found out that the code doesn't match the GUI's functionality, so because of that the team needed to change almost the entire code in order to make a functional GUI. A new feature was added for searching customers by name in the system.

Besides those problems we mainly followed the requirements of our client and the basic structure of the project.

8. Conclusions

The aim of this project was to design and create a system that allows the company's employees to make renting procedures and easily manage the company's customers and vehicles.

At the beginning we focused on recognizing the requirements from the interview. The next steps where to make use case, use case descriptions and activity diagrams based on the requirements. Based on the use cases we created the class diagram with the most important methods. During implementation we found some errors and we needed to change the class diagram and also some of the use case description.

Overall, the result of the project is that some of the requirements are completely fellfield and some of them are not fully finished.

9. List of references

Links:

1. https://docs.oracle.com/en/
2. http://www.tutorialspoint.com/java/
3. http://www.allanhenriksen.dk/via/
4. https://studienet.via.dk/

Books:

5. Gaddis. T.: *Starting out with Java 3rd Edition*

10. Appendices

- Appendix A – Use Case Diagram
- Appendix B – Use Case Description
- Appendix C – UML Class Diagram
- Appendix D – Activity Diagram
- Appendix E – User Guide
- Appendix F – Java Source Code
- Appendix G – Daily Logs