

# PROJECT REPORT

## ICT engineering, sem. 1

Vehicle Rental System for V-Rent

Course: SEP I1

Supervisors: Allan Henriksen, Mona Wendel Andersen

Date: 03/06/2016

Group 1: Ana Iulia Chifor [245484], Andreea Carst [240286]  
Filip Hudec[245495], Signe Lange Rasmussen[246676]

## *Abstract*

*V-Rent is an uprising rental company located in Horsens. Only 3 years old, the company is already facing a lot of orders which the employees cannot keep track of manually, as they used to do in the beginning. Therefore, the owner of V-Rent is looking for a digital solution that will ease the access to information for both his employees and the company's clients.*

*The following report is trying to come up with a solution for the current problem, by analyzing, designing and implementing a new product which is specially customized for the company's needs. The findings are then shown and discussed.*

*The report contains information about how the solutions were thought through and applied.*

*The end product meets all the requirements asked by V-Rent and it is therefore a realistic solution for the issues that the company is facing at the moment.*

# Table of Contents

<i>Abstract</i> .....	1
Introduction .....	3
Analysis .....	4
Requirements .....	4
Use Case modelling.....	5
Activity diagrams .....	8
Design .....	10
Class diagrams.....	10
GUI design .....	11
Sequence diagrams.....	12
Implementation .....	13
Test & Results .....	18
Discussion .....	19
Conclusion.....	20
References .....	21
Appendices .....	21
APPENDIX A – The Interview .....	21
APPENDIX B – Use Case Descriptions and Activity Diagrams .....	23
APPENDIX C – Class Diagram .....	29
APPENDIX D – User Guide .....	29

## Introduction

V-Rent is a vehicle rental company located in Horsens that is in need of a software system for their developing business. The owner of the company, Van Motor, wants to replace the old booking system with a digital one, in order to make the renting process faster and easier for both the clients and the employees.

So far, the company has been using a non-functional spreadsheet where the secretary would introduced data about the rented vehicle and the client. This booking process is now causing problems for the company which has increased the number of vehicles it owns and has an increasing demand for rentals. At the moment, all the checking for rented vehicles has to be done manually, which is a great issue for V-Rent. In addition to that, the secretary is also in charge of manually updating data regarding the vehicle's present situation (i.e. whether the vehicle is in service at the moment, the kilometers driven of a vehicle etc.).

V-Rent is therefore interested in a solution with the following main features: a booking option where the employee is able to go through all the steps necessary for introducing, saving, comparing and changing information when a client wants to rent a vehicle.

Furthermore, the system should be able to suffer adjustments in the future, in case the company decides to start selling some of the vehicles and at the same time the system should be flexible enough to be able to merge information from a holiday house renting company in the next years.

The following report aims to present a potential solution for the system the company is asking for. The report will contain analysis and design of the system, together with the implementation, testing and the results of the findings.

# Analysis

## Requirements

The following lists of requirements have been set up according to V-Rent's expectations for the new system, which were withdrawn from an interview with them. The full interview with the company can be found in Appendix A. Both the functional and non-functional lists have been ordered in a prioritized sequence.

The orange block of the functional list represents the top priority requirements which will be analyzed in the report.

Table 1 - List of functional requirements

	FUNCTIONAL REQUIREMENTS
TOP PRIORITY REQUIREMENTS	<ol style="list-style-type: none"> <li>1. The user should be able to set pick up time</li> <li>2. The user should be able to set return time</li> <li>3. The user should be able to set vehicle type</li> <li>4. The user should be able to set up pick up location</li> <li>5. The user should be able to set up return location</li> <li>6. The user should be able to see prices</li> <li>7. The user should be able to set customer name</li> <li>8. The user should be able to set driver license number</li> <li>9. The user should be able to set estimated kilometers for route</li> <li>10. The system should calculate discount prices for vehicles rented for a longer period</li> <li>11. The user should be able to set actual return time and location of the vehicle</li> <li>12. The user should be able to see list of all available vehicles</li> <li>13. The user should be able to see list of all vehicles</li> <li>14. The user should be able to see list of rented vehicles</li> <li>15. The user should be able to see list of active clients</li> <li>16. The program has pop up message for too many kilometers</li> </ol>
NON-PRIORITY REQUIREMENTS	<ol style="list-style-type: none"> <li>17. The user should be able to see special renting time for auto camper.</li> <li>18. The user should be able to set renting price</li> <li>19. The user should be able to remove a vehicle</li> <li>20. The user should be able to add a vehicle</li> </ol>

	21. The user should be able to change data easily later on 22. The user should be able to set price for a sale vehicle
--	---

Table 2 - List of non-functional requirements

NON-FUNCTIONAL REQUIREMENTS
1. The program should be made in Java

## Use Case modelling

The actor of the use case diagram is the user of the system, which in this case represents any employee from V-Rent. Figure 1 contains an overview of all the use cases which should be applied in the system. The first three cases are further described in the project, since they represent the most frequently used scenarios. A full description of the last three may be found in Appendix B.

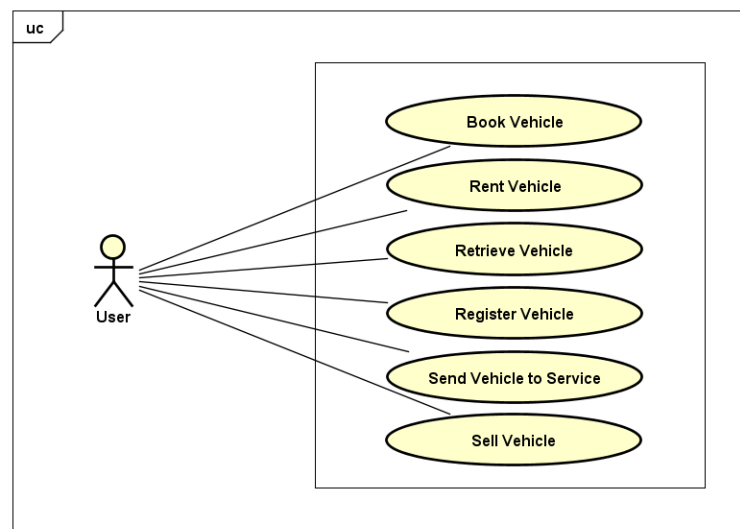


Figure 1 - An overview of all the use cases

The diagram in Figure 1 shows each possible use of the system. The user should be able to choose what they need to do. The first three use cases are depending on each-other, meaning that the user can only rent a vehicle if a booking was created first and the user can only retrieve a vehicle if a rent was registered. The last three cases are independent and can be used at any time.

*Table 3 - Book Vehicle - Use Case Description*

Use Case	Book Vehicle
Summary	User inserts details about the booked vehicle
Actor	User
Precondition	There are available vehicles
Post-condition	User has successfully introduced and saved data about a booked vehicle.
Base Sequence	<ol style="list-style-type: none"> <li>1. User inserts type of vehicle.</li> <li>2. User inserts pick up time.</li> <li>3. User inserts return time.</li> <li>4. User inserts pick up place.</li> <li>5. User inserts return place.</li> <li>6. User inserts estimated kilometers.</li> <li>7. User clicks "Search" button.</li> <li>8. System shows estimated price of the rental.</li> <li>9. System shows available vehicles.</li> <li>10. User selects vehicle.</li> <li>11. User inserts name of the client.</li> <li>12. User confirms booking.</li> </ol>
Exception Sequence	<p>User applies filters:  1-9 as base sequence.  10. User applies filters  11. User clicks "Search" button.  12. System shows available vehicles.  13. User selects vehicle.  14. User inserts name of the client.  15. User confirms booking.  Use case ends.</p>

The use case description from Table 3 represents the first use case of the system. This case will come in use whenever the employee from V-Rent gets a new order for booking a vehicle. This first step represents the basic information that the user needs to save from a client, in order to make a booking. The first functionalities of the system will already be activated here. The user will no longer need to manually check whether a vehicle is available or not. The system will automatically select the available vehicle according to the filters selected and it will save the information inputted in order to be used for the next step. Moreover, the system will generate a reservation number for the booking, which will ease the process of finding a specific booking information later on.

*Table 4 - Rent Vehicle - Use Case Description*

Use case	Rent vehicle
Summary	The user has successfully went through the whole process of renting a vehicle to a client
Actor	User
Precondition	The user has access to data from 1 <sup>st</sup> step of the renting process - the "Book Vehicle" use case
Post-condition	User has successfully introduced and saved data about a rented vehicle.
Base Sequence	<ol style="list-style-type: none"> <li>1. User inserts reservation number.</li> <li>2. System loads full name of the client and vehicle information.</li> <li>3. User inserts client information.</li> <li>4. User confirms order.</li> </ol>

The second use case of the system – Rent Vehicle – which is described in Table 4, takes place later on, when the client comes to the company to pick up a booked vehicle. At this step, the user will check and save the client information necessary for renting out the vehicle (i.e. driving license number and phone number of the client).

*Table 5 - Retrieve Vehicle - Use Case Diagram*

Use Case	Retrieve Vehicle
Summary	User inserts details about the returned vehicle
Actor	User
Precondition	The user has access to information about the rented vehicle
Post-condition	User has successfully introduced and saved data about a returned vehicle.
Base Sequence	<ol style="list-style-type: none"> <li>1. User inserts the reservation number.</li> <li>2. System loads client and vehicle information.</li> <li>3. User inserts the kilometers driven of the vehicle.</li> <li>4. User inserts information about the state of the vehicle upon return.</li> <li>5. System sets return date to current date.</li> <li>6. User inputs the final price of the rental.</li> <li>7. User confirms the return of the vehicle.</li> </ol>

Once a client returns a vehicle, the user will have to access the third use case described in Table 5 – Retrieve Vehicle. At this step the system will provide features such as setting the



return date to current date, saving the condition (good/damaged) of the vehicle upon return, or updating the actual driven kilometers for the vehicle.

## Activity diagrams

The following activity diagrams are a visual representation of the main three use case descriptions, including the exception sequences. The figures represent step by step actions from both the user and the system.

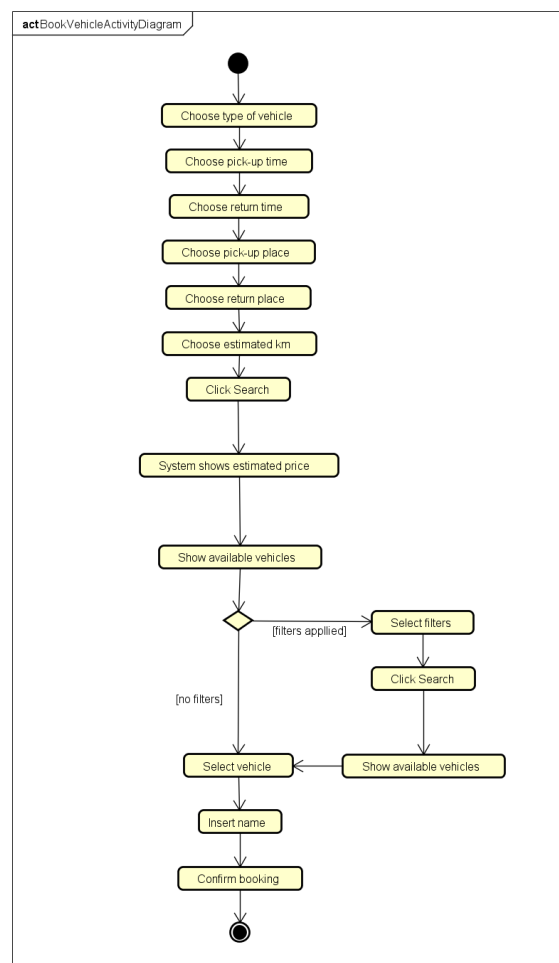


Figure 2 - Book Vehicle Activity Diagram

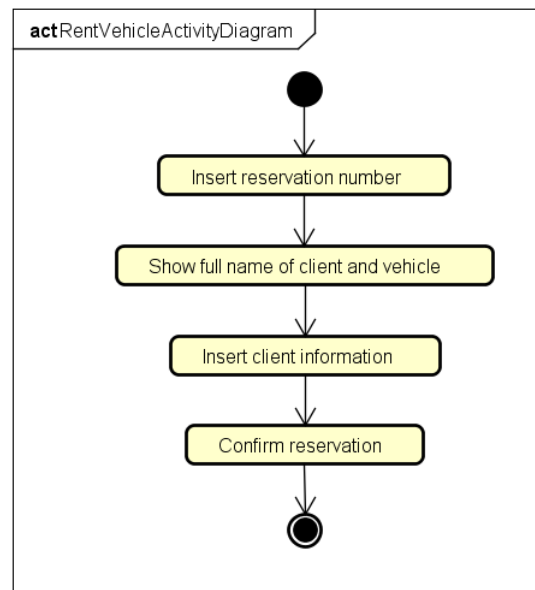


Figure 3 - Rent Vehicle Activity Diagram

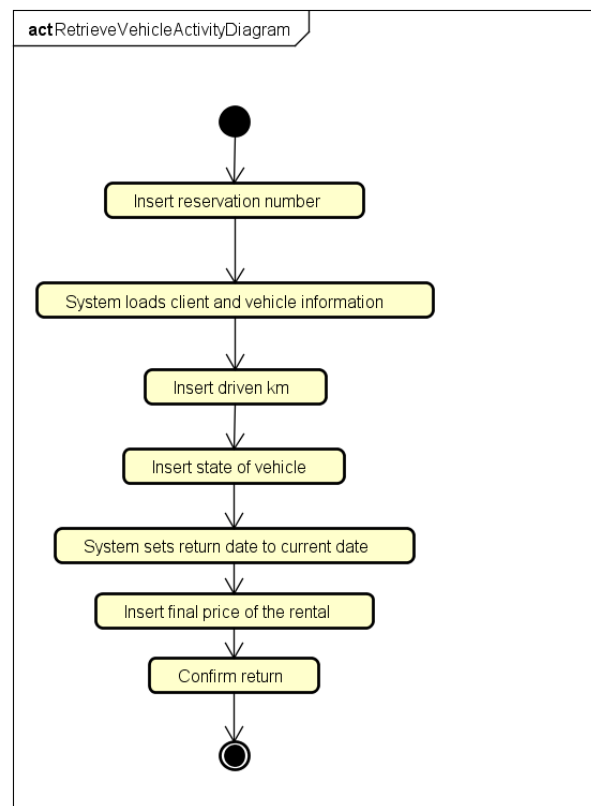


Figure 4 - Retrieve Vehicle Activity Diagram

## Design

### Class diagrams

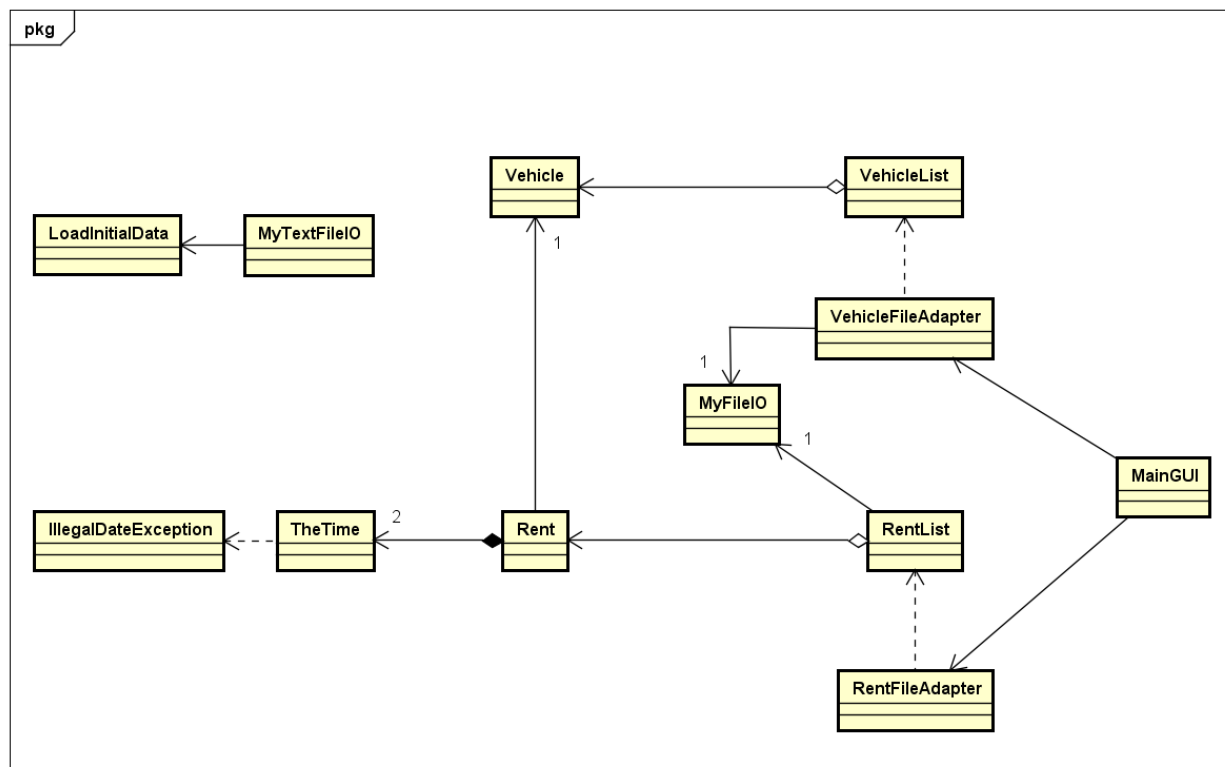


Figure 5 Class Diagram of the System

Figure 5 is a minimalistic representation of the classes used for the system. For a detailed version of the diagram, please check Appendix C. There are different types of relationships that are being used throughout the system. One of them is a composition relationship, which can be found between *TheTime* class and *Rent* class – by using this kind of relationship, it is ensured that whenever a date is used in the system, it will not be overwritten but rather make a copy of the original date. Next there are two aggregation relationships between *RentList* class and *Rent* class as well as *VehicleList* class and *Vehicle* class. The dotted arrows represent dependency relationships and the rest of the classes use association relationships.

## GUI design

The graphical user interface of this project is focused on having as many filters as possible, in order to make the search of certain types of vehicles easy for the user. The full interface can be found in Appendix D.

The program was organized in tabs, according to the activity the user wants to access. This way, the user gets maximum visibility of all the actions available at all times – see Figure 6.

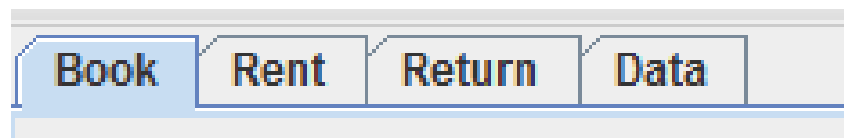


Figure 6 - Options for the user

Moreover, the program provides features such as searching for vehicles according to types, availability, model, colour etc. In Figure 7 there is a representation of the possible filters that the user can access when making a booking.

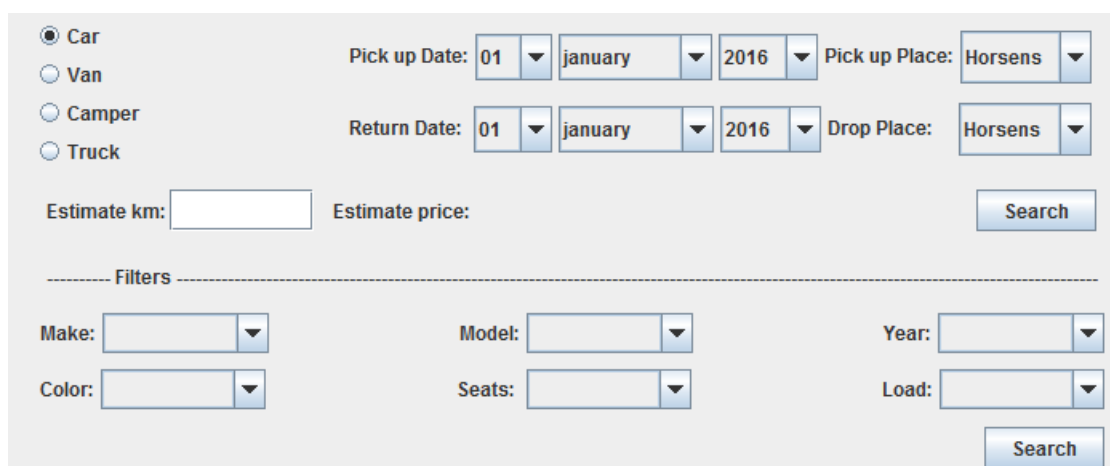


Figure 7 shows a form for filtering vehicle search results. It includes radio buttons for vehicle types: Car (selected), Van, Camper, and Truck. There are date pickers for 'Pick up Date' and 'Return Date', both set to 01 January 2016. There are also dropdown menus for 'Pick up Place' and 'Drop Place', both set to Horsens. Below these are input fields for 'Estimate km' and 'Estimate price', followed by a 'Search' button. A dashed line separates this section from the 'Filters' section. The 'Filters' section includes dropdown menus for 'Make', 'Model', 'Year', 'Color', 'Seats', and 'Load', followed by another 'Search' button.

Figure 7 - Filters for booking a vehicle

Another example of filters available to the user are the ones down in Figure 8. Here the user can easily access information about all the booked vehicles, the rented ones or even check the ones which are currently in service.

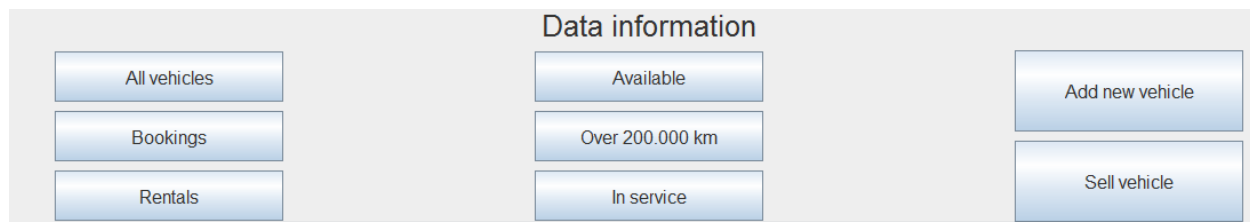


Figure 8 - Filters for retrieving data about the vehicles

## Sequence diagrams

Some of the most used methods when using the program are the filters applied for booking vehicles or for getting information about a certain type of vehicles. The following sequence diagram is representing the first nine steps of the Booking use case. Figure 9 shows the different classes used for detecting the filtered options from the user.

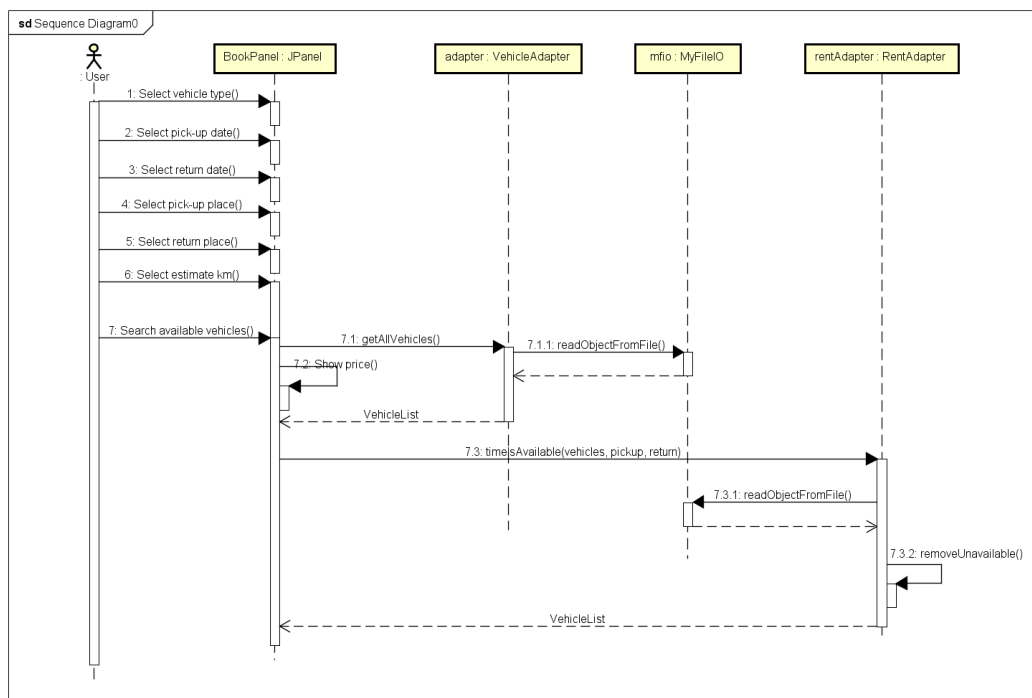


Figure 9 - Sequence diagram of booking a vehicle

## Implementation

When implementing the code, most of the functionalities went to the File Adapter classes, where different filtering methods were created. One example of these methods can be seen in the code snippet from Figure 10, where the method returns a list of vehicles of type *VehicleList* after the method runs a for loop that will check if the make of the vehicle introduced is equal to any vehicle in the list. The method can be found in the *RentFileAdapter* class, from the class diagram, in Appendix C.

```
public VehicleList filterByMake(VehicleList list, String make,
                                String noneSelector)
{
    if (make.equals(noneSelector))
    {
        return list;
    }

    VehicleList tempList = new VehicleList();
    String[] tempArray;

    for (int i = 0; i < list.size(); i++)
    {
        tempArray = list.get(i).toString().split(",");

        if (tempArray[2].equals(make))
        {
            tempList.add(list.get(i));
        }
    }

    return tempList;
}
```

Figure 10 – Filter by Make Method

Another functionality of the system that is frequently used by the program, is the *timeIsAvailable* method, which can be found in *RentFileAdapter* class as well. The method takes as parameters a list of type *VehicleList* and two dates of type *TheTime*. Then the method checks step by step every possibility whether the dates introduced do not interfere with the other dates which have been saved for renting the vehicle previously. This method comes to use whenever the system has to check whether a vehicle is currently rented. The method will return a list with the available vehicles, of *VehicleList* type.

```
public VehicleList timeIsAvailable(VehicleList list, TheTime pickUpTime2,
    TheTime returnTime2)
{
    RentList rentals = getAllRents();
    for (int i = 0; i < rentals.size(); i++)
    {
        Rent rent = rentals.get(i);
        if (rent.getPickUpTime().equals(pickUpTime2)
            || rent.getReturnTime().equals(returnTime2)
            || rent.getPickUpTime().equals(returnTime2)
            || rent.getReturnTime().equals(pickUpTime2))
        {
            list.remove(rent.getVehicle());
        }

        else if (rent.getPickUpTime().isBefore(pickUpTime2)
            && pickUpTime2.isBefore(rent.getReturnTime()))
        {
            list.remove(rent.getVehicle());
        }
        else if (pickUpTime2.isBefore(rent.getPickUpTime())
            && rent.getPickUpTime().isBefore(returnTime2))
        {
            list.remove(rent.getVehicle());
        }
        else if (rent.getPickUpTime().isBefore(pickUpTime2)
            && returnTime2.isBefore(rent.getReturnTime()))
        {
            list.remove(rent.getVehicle());
        }
        else if (pickUpTime2.isBefore(rent.getPickUpTime())
            && rent.getReturnTime().isBefore(returnTime2))
        {
            list.remove(rent.getVehicle());
        }
    }
    return list;
}
```

*Figure 11 - timeIsAvailable Method*

An extra feature that the system provides is checking whether the vehicles have more than 200.000 km driven. This way the user can keep track of the over driven vehicles and can avoid possible extra repairing taxes.

```
public VehicleList checkKm()
{
    VehicleList vehicles = new VehicleList();

    VehicleList result = getAllVehicles();
    for (int i = 0; i < result.size(); i++)
    {
        if (result.get(i).getDrivenKm() >= 200000)
        {
            vehicles.add(result.get(i));
        }
    }
    return vehicles;
}
```

Figure 12 - checkKm Method

Furthermore, comes a code example of Button Listeners, for a better understanding of what happens when the user filters options for booking a vehicle. The code is part of a private class and it is divided in several pictures which will be explained step by step. The void method *actionPerformed* will detect which of the two search buttons from the GUI was pressed. Next, the program will set the date and the kilometers to the ones selected by the user. Then method will try to convert the text from the kilometers field to integers. In case of error, the exception is caught and the warning message will inform the user that the estimated kilometers must be a number.

```
private class MyButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == searchButton1) {
            String selectedSday = (String) dayBox1.getSelectedItem();
            String selectedEday = (String) dayBox2.getSelectedItem();

            String selectedSmonth = (String) monthBox1.getSelectedItem();
            String selectedEmonth = (String) monthBox2.getSelectedItem();

            String selectedSyear = (String) yearBox1.getSelectedItem();
            String selectedEyear = (String) yearBox2.getSelectedItem();

            String selectedEstKm = (String) estimateKmText.getText();
            int estimatedKmInt;
            double estimatePrice;

            try // exception handler for "bad" estimated Km input
            {
                if (selectedEstKm.equals("")) {
                    estimatedKmInt = 0;
                } else {
                    estimatedKmInt = Integer.parseInt(selectedEstKm);
                }
            } catch (NumberFormatException f) {
                JOptionPane.showMessageDialog(null,
                    "Estimated km must be a number!",
                    "Warning message", JOptionPane.WARNING_MESSAGE);
            }
            return;
        }
    }
}
```

Figure 13 - MyButtonListener class



Next the type of the vehicle will be set according to the user's choice from the radio button (Figure 14).

```
String type = "";

if (rType1.isSelected()) {
    type = "Car";
}

else if (rType2.isSelected()) {
    type = "Van";
}

else if (rType3.isSelected()) {
    type = "Camper";
}

else if (rType4.isSelected()) {
    type = "Truck";
}
```

Figure 14 - Radio Buttons

The method will further on check if the dates introduced by the user are eligible, otherwise, the system will display a warning message.

```
SDate = new TheTime(selectedSday, selectedSmonth, selectedSyear);
EDate = new TheTime(selectedEday, selectedEmonth, selectedEyear);

if (!(SDate.isBefore(EDate))) {
    JOptionPane.showMessageDialog(null,
        "Pick up date must be before return date",
        "Warning message", JOptionPane.WARNING_MESSAGE);
    return;
}

TheTime thisDate = new TheTime(thisDay, thisMonth, thisYear);

if (!(thisDate.isBefore(SDate)) || !(thisDate.isBefore(EDate)))
{
    JOptionPane.showMessageDialog(null,
        "Both pickup and return dates must be after current date",
        "Warning message", JOptionPane.WARNING_MESSAGE);
    return;
}
```

Figure 15 - Time Check

The system will then automatically calculate the price for the rental by calling the *discount* method, according to the period rented and the estimated driven kilometers. Then the system will select the available vehicles by type and dates.

```
estimatePriceDouble = discount(estimatedKmInt, SDate, EDate);
estimatePriceText.setText(estimatePriceDouble + "");

vehicles = vehicleAdapter.getAllVehicles();
vehicles = rentAdapter.timeIsAvailable(vehicles, SDate, EDate);
vehicles = vehicleAdapter.filterByType(vehicles, type,
    NoneSelector);
update(vehicles);
addItemToBoxes(vehicles, getNoneSelector());
}
```

Figure 16 - Calculating the price

The second big condition checked by the method is whether or not the second search button is used. In that case, the method *ApplyFilters* is called.

```
if (e.getSource() == searchButton2 && vehicles != null) {
    vehicles = ApplyFilters(vehicles);
    update(vehicles);
}
```

Figure 17 - Button Search 2

The last big if condition checks whether the confirm button was pressed. In that case a new panel will pop up, where the user will introduce the client name before confirming the booking.

```
if (e.getSource() == finalConfirm) {
    String selectPickUp = (String) locationBox1.getSelectedItemAt();
    String selectDrop = (String) locationBox2.getSelectedItemAt();

    Vehicle selectedVehicle;
    selectedVehicle = (Vehicle) listArea.getSelectedValue();

    if (selectedVehicle == null) {
        return;
    }

    BookConfirm confirm = new BookConfirm(rentAdapter,
        selectedVehicle, SDate, EDate, selectPickUp,
        selectDrop, selectedVehicle.getDrivenKm(),
        estimatePriceDouble);
}
```

Figure 18 - Confirm button

## Test & Results

The table below shows the results of testing the system. The tests were both conducted in the console and GUI. Therefore the program was debugged and most issues that may arise have been checked.

All of the proposed cases worked successfully.

*Table 6 - Test results*

Book a vehicle		Result
Get available vehicles after selecting type of vehicle, dates for pick-up and return and place of pick-up and return		Works
Get estimated price according to renting period and estimated kilometres		Works
Get selected vehicle information with picture		Works
Save booking with name of the client and booking data		Works
Rent a vehicle		Result
Find a booking by reservation number and show the vehicle and client information		Works
Save renting information		Works
Return a vehicle		Result
Update the actual driven kilometres on return		Works
Update actual return date		Works
Send a vehicle to service		Result
Get all available vehicles list		Works
Send selected vehicle to service		Works
Register a vehicle		Result
Insert all information needed for registering a new vehicle		Works

Save the new vehicle in the system	Works
Sell a vehicle	Results
Get all available vehicles list	Works
System asks for confirmation of sale	Works
System deletes the selected vehicle from database	Works

## Discussion

In relation to what the client has asked for the new system, most of the features have been implemented. Moreover, the system was designed in such a manner that features can easily be added in the future when there will be a need for it.

The program offers a large variety of filters, so the user has access to as much information about a vehicle as possible. This will ease communication between V-Rent's employee and their client. The program offers all details needed about a vehicle, including a picture of the vehicle – therefore anyone can make a booking and offer details about a vehicle, without needing to actually be a connoisseur in the matter. On the other hand the information of the vehicles when displayed in the scroll panel is not easily understood by a user who has not read the user guide.

Another feature that the program offers is generating a reservation number for each booking. This way, the client information will quickly be accessed by the employee when needed.

Last but not least, in the design of the program it was ensured that the main activities that the employee will use are at sight at all times and thus will ease the access and browsing from one place to another.

## Conclusion

V-Rent's decision to readjust the way they store and select data about the rentals has led to a whole process of analyzing the requirements, designing a solution, implementing and testing the findings.

A list of opportunities for the new system has been withdrawn in the first part of the project, where the list of requirements served as main instructions for what the end product should be able to do. With the help of the use case descriptions and activity diagrams, a thorough overview of the system's features took shape.

When designing the product, the project's main focus was both functionality and efficiency. The system is aiming to achieve most of the requirements from V-Rent, whereas the GUI offers accessibility for any kind of user, no matter whether they are a vehicle expert or not. Moreover, the system was designed in such a way that it is easy to add extra features if needed in the future.

The implementation of the design was the most time consuming part of the project. The different classes and methods had to be thought through and implemented in the system. Most of the methods focused on filtering the options, reading from the user input, updating and saving information.

Further on, the testing of the product was a successful breakthrough, where all the methods have been checked and debugged.

All in all, the final product can be considered an eligible candidate for a replacement of the manual system that V-Rent is currently using. The company's demands have been accomplished and V-Rent can start using the new product right away.

## References

Allanhenriksen.dk. (2016). *SDJI*. [online] Available at:

<http://www.allanhenriksen.dk/via/sdj/index.html> [Accessed 1 Jun. 2016].

Docs.oracle.com. (2016). *How to Use Lists (The Java™ Tutorials > Creating a GUI With JFC/Swing > Using Swing Components)*. [online] Available at:

<https://docs.oracle.com/javase/tutorial/uiswing/components/list.html> [Accessed 25 May 2016].

Gaddis, T. (2007). *Starting out with Java*. 3rd ed. New Jersey: Pearson Education (US).

Stackoverflow.com. (2016). *Stack Overflow*. [online] Available at:

<https://stackoverflow.com/> [Accessed 1 Jun. 2016].

## Appendices

### APPENDIX A – The Interview

***“First of all, can you tell us something about your company?”***

Van Motor: *“Yes, I started the rental of cars three years ago and it is an excellent business already. My cousin*

*Pete Diesel now owns 25% of the company and is, like me, working full time. We also have three part time workers, Jack, Jim and Gina doing some work for us. Today we no longer only rent out regular cars, but also other vehicles. In total we have 27 family cars, 12 vans, 4 moving trucks and a single auto camper. Once a customer liked the car he rented so much that he offered to buy it. We have talked about if this could be one of our future goals – I mean to also to sell cars. Pete is however not into this idea. Instead, Pete wants us to rent out motorcycles, but I am not sure that this is such a good idea.”*

***“OK, good to know. Are all the vehicles here in Horsens?”***

Van Motor: *"Well, Horsens is the main location, but the customers can actually pick up cars in different locations in Denmark. The boys Jack and Jim are hired to drive vehicles between our main location in Horsens and these pick up destinations. This service is widely used for the moving trucks because when moving to another part of the country, no one likes to drive back with an empty truck afterwards. Sometimes the boys also drive vehicles to and from auto repair."*

***"What about you, your cousin and the last one of the employees, what do you do?"***

Van Motor: *"Gina and I are mostly working in the office renting vehicles to customers, while Pete is the handy man cleaning, refueling and doing some simple repairs."*

***"How does a customer rent a vehicle now?"***

Van Motor: *"They either come into the office or call us on the phone to reserve a vehicle. We then ask them for their name, pick up time, return time, pick up location and delivery location, an estimate of the amount of kilometers they will drive, and of course which type of vehicle to rent. When they pick up the vehicle we also need to see their driver license and write down the driver license number. Often a customer will ask for a specific car model or for a van with a certain load size."*

***"Do they rent a car per hour or for a full day?"***

Van Motor: *"We have prices for one day rent including 100 km. If you rent a vehicle for several days in a row the following days you get a 30% discount. The auto camper is rented for one weekend or a full week at a time."*

*Unfortunately, the auto camper is only rented out in the summer period and some other vacations. The paradox is that in these periods we could perhaps rent out five auto campers. We haven't found a solution for this yet."*

***"Have you thought about how you would like your new rental system to work?"***

Van Motor: *"Yes, we talked about that, and we want it to make it easy to book vehicles. We imagine maybe also being able to see a list of all vehicles, and see which of these are available at a specific time. Then it will have to, in some way, store information about things like the rentals, vehicle data and customer data. Gina doesn't have much knowledge about cars, and I know that she would like to have lots of information about the vehicles on the screen when a customer calls. For every 20.000 km a car is driven to auto repair for a check, and it would also be nice to have some message when it is time to do that."*

***"Would you like the customer to book on the Internet instead of calling you on the phone or stopping by?"***

Van Motor: *"No, not at this time. We don't want to start out with an online booking system. What we need for now is a program running on a single computer at the front desk, so whoever is at work can make the booking."*

***“Any other things we should consider?”***

Van Motor: *“Yes actually, we are considering buying a summer house rental company in a few years and at this time our system and theirs have to be combined. Therefore, it would be nice if you could make the program so that the graphical interface and the way the data is stored could easily be changed later on. If you do this, then we might just hire you again for making the upcoming changes!”*

***“OK, sounds great. So if there is nothing else, then we will get to work...”***

Van Motor: *“Oh, actually there is one more thing. I have a nephew who is learning to program at the moment, and I think it could be helpful for him to see a big program made in the same language as he is using. If only I could remember the name of it... could it be... Coffee?”*

***“That is actually a thing, but maybe you are thinking of Java?”***

Van Motor: *“Yes, that’s the one! Can you make the program in that?”*

***“Sure. Thank you very much for your time. We will try our very best to create a system that will fulfil your requirements.”***

Van Motor: *“Thank you, I look forward to trying out the program!”*

## **APPENDIX B – Use Case Descriptions and Activity Diagrams**



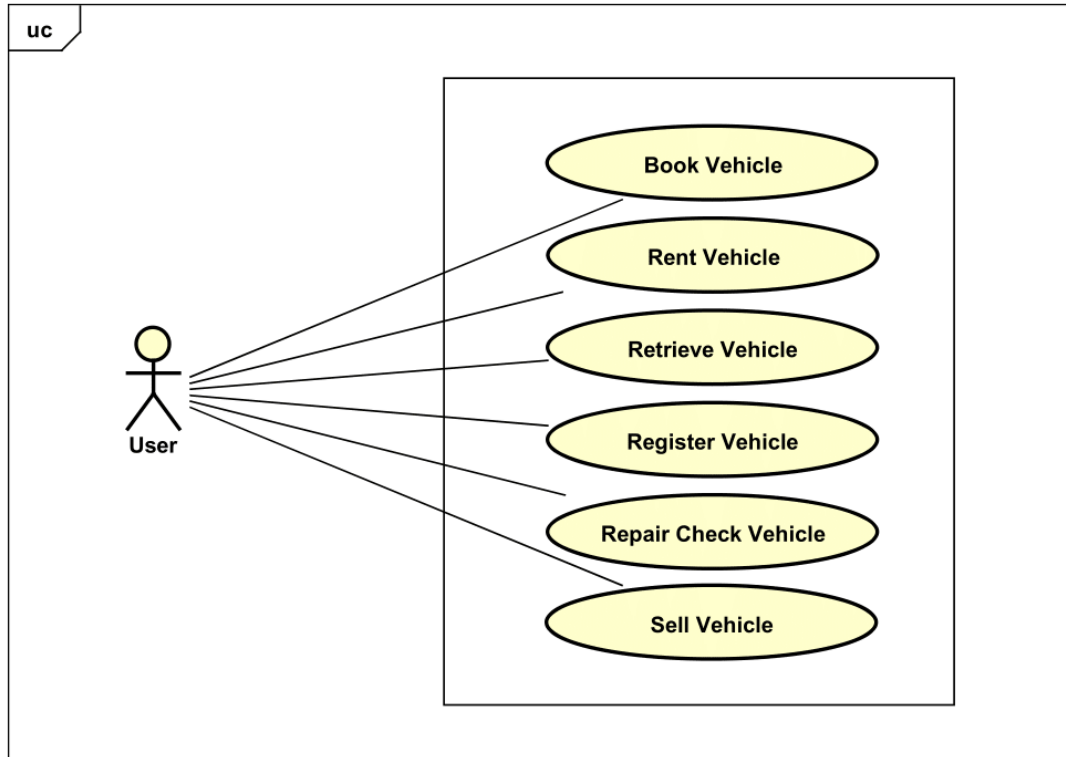


Figure 19 - Use Case Diagram

Table 7 - Register Vehicle Use Case Description

Use Case	Register Vehicle
Summary	User inserts details about a newly acquired vehicle
Actor	User
Post-condition	User has successfully introduced and saved data about a registered vehicle.
Base Sequence	<ol style="list-style-type: none"> <li>1. User inserts the registration number.</li> <li>2. User inserts the type of the vehicle.</li> <li>3. User inserts the make of the vehicle.</li> <li>4. User inserts the model of the vehicle.</li> <li>5. User inserts the year of the vehicle.</li> <li>6. User inserts the color of the vehicle.</li> <li>7. User inserts the seats of the vehicle.</li> <li>8. User inserts the driven km of the vehicle.</li> <li>9. User inserts the load size of the vehicle.</li> <li>10. System sets condition of the vehicle automatically to false.</li> <li>11. System saves information about the new vehicle.</li> </ol>
Use Case	Send Vehicle to Service

Table 8 - Send Vehicle to Service Use Case Description

Summary	User registers a vehicle as in service
Actor	User
Post-condition	User has successfully introduced and saved data about a vehicle sent to service.
Base Sequence	<ol style="list-style-type: none"> <li>1. User selects to see al vehicle list.</li> <li>2. System shows all vehicle list.</li> <li>3. User selects the vehicle he/she wants to send to service.</li> <li>4. System registers the vehicle as sent to service.</li> </ol>

Table 9 - Sell Vehicle Use Case Description

Use Case	Sell Vehicle
Summary	User inserts information about a sold vehicle
Actor	User
Post-condition	User has successfully introduced and saved data about a sold vehicle.
Base Sequence	<ol style="list-style-type: none"> <li>1. User selects to see all vehicle list.</li> <li>2. System shows all vehicle list.</li> <li>3. User selects the vehicle he/she want to sell.</li> <li>4. User selects "Sell vehicle" button.</li> <li>5. System asks the user to confirm the sale.</li> <li>6. User confirms the sale.</li> <li>7. System deletes selected vehicle from the system.</li> </ol>

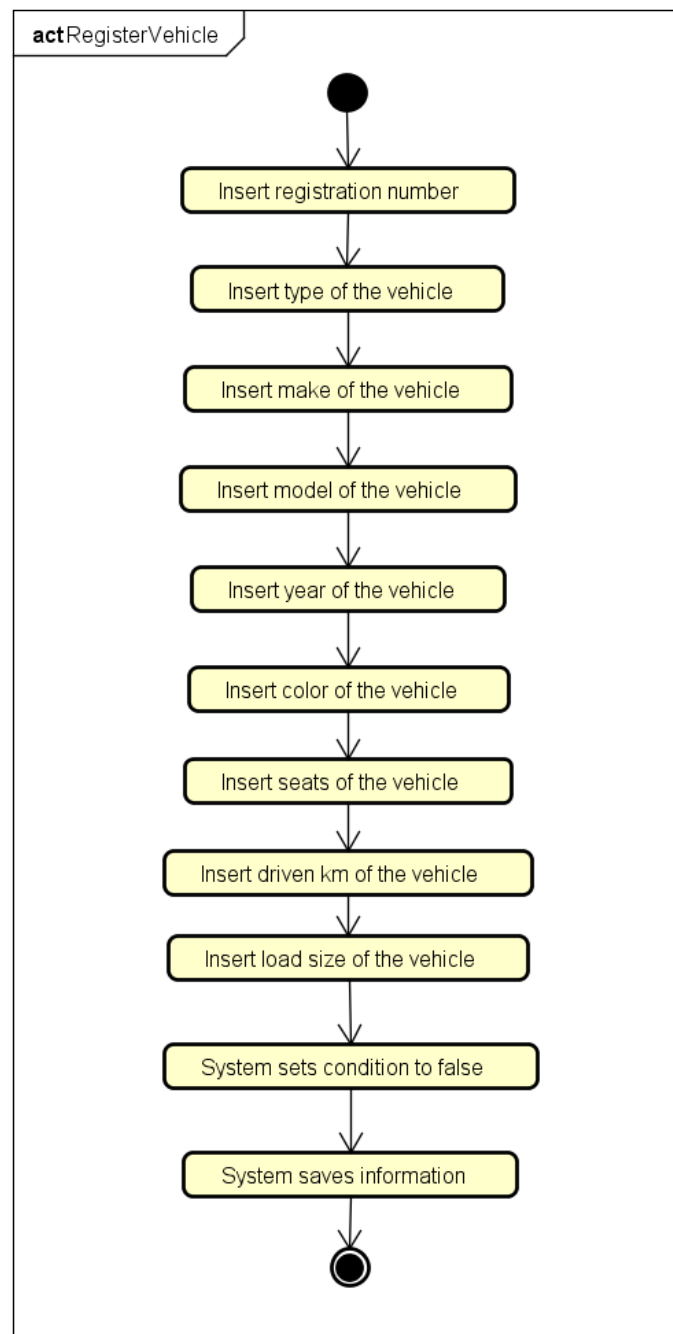


Figure 20 - Register Vehicle Activity Diagram

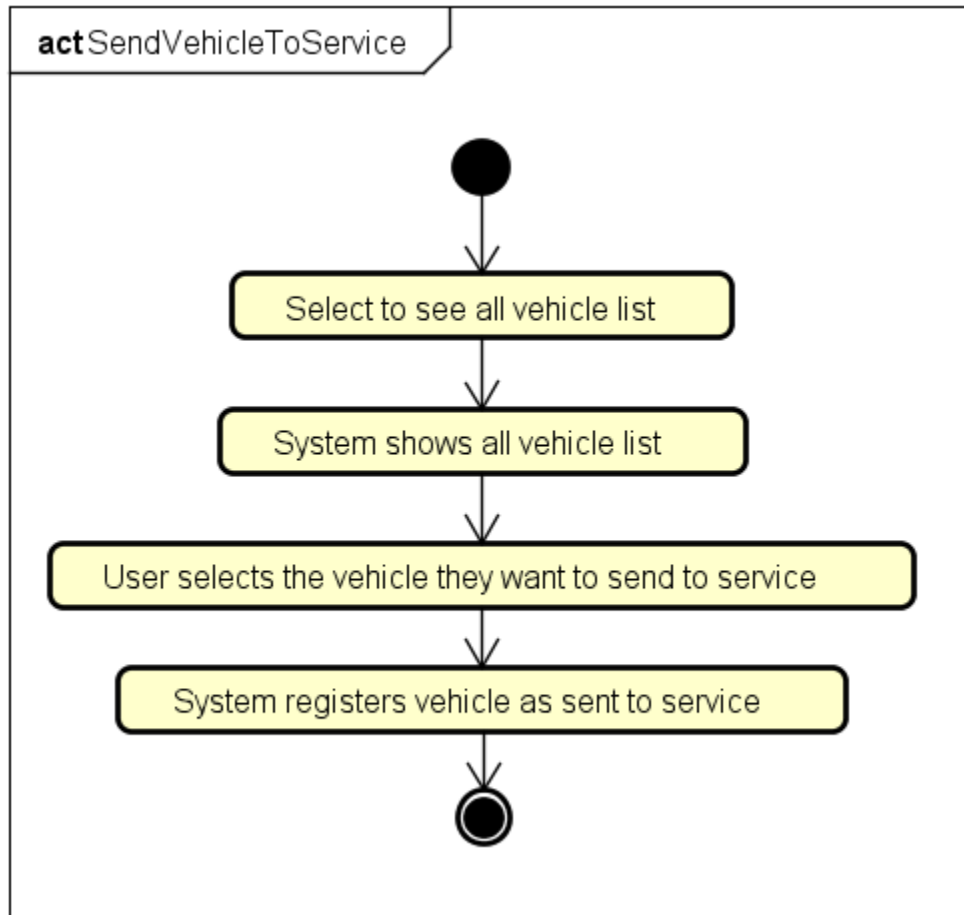


Figure 21 - Send Vehicle to Service Activity Diagram

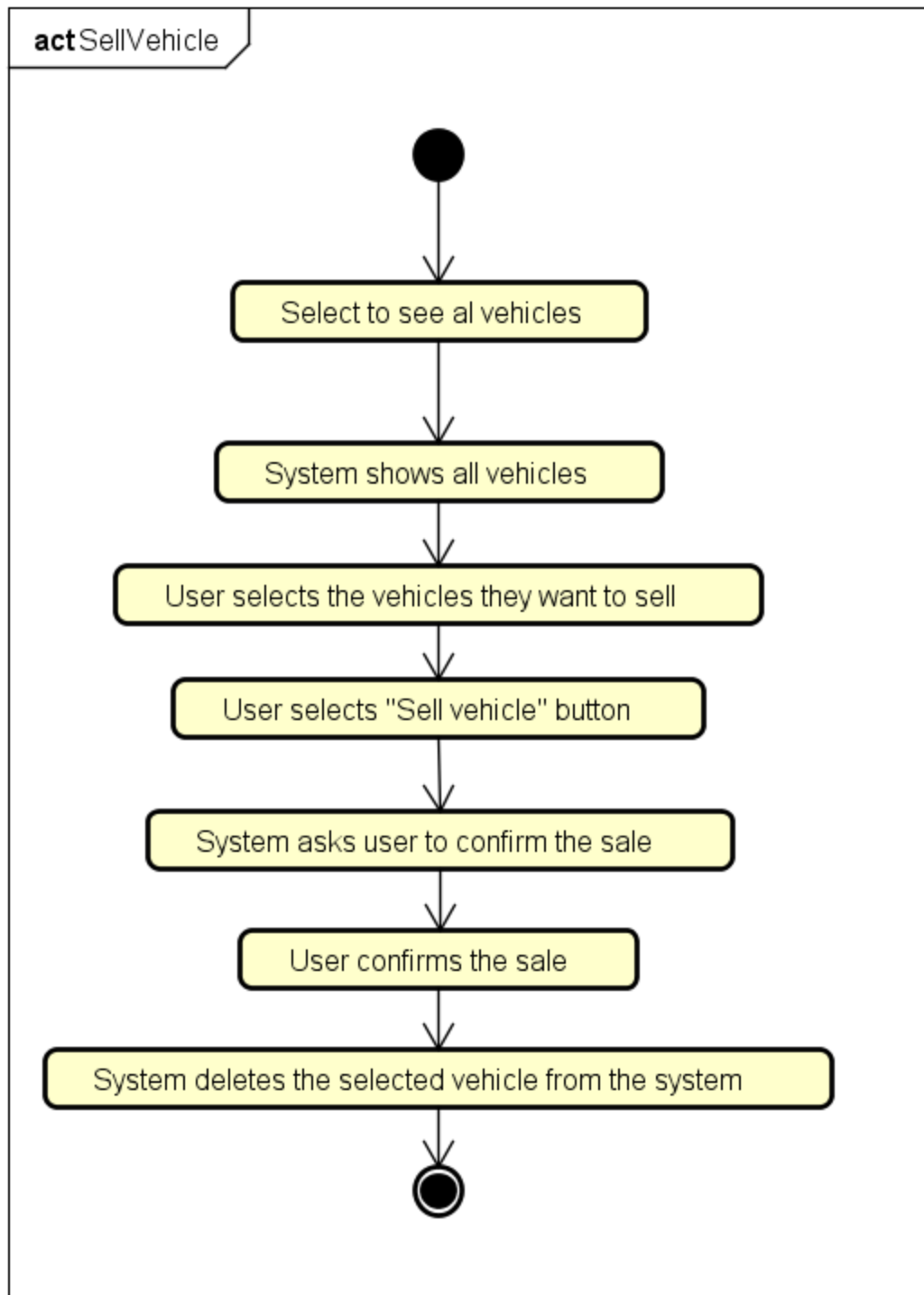


Figure 22 - Sell Vehicle Activity Diagram



<input checked="" type="radio"/> Car	Pick up Date:	01	▼	january	▼	2016	▼	Pick up Place:	Horsens	▼
<input type="radio"/> Van										
<input type="radio"/> Camper	Return Date:	01	▼	january	▼	2016	▼	Drop Place:	Horsens	▼
<input type="radio"/> Truck										

- Enter the estimated amount of what the costumer will drive as a whole number.
- Press the first *Search* button.

Estimate km:	<input type="text" value="100"/>	Estimate price:	<input type="button" value="Search"/>
--------------	----------------------------------	-----------------	---------------------------------------

- View the estimated price and available vehicles.
- The information about the vehicle is written the following order:

Registration number, vehicle type, make, model, year of production, color, amount of seats, kilometers the vehicle has driven, load in kilograms, service status (false if out from service, true if vehicle is currently in service).

Estimate km:  Estimate price: 100.0

----- Filters -----

Make:  Model:  Year:

Color:  Seats:  Load:

AL19742,Car,Ferrari,F50,1997,red,2,6000,0,false
HE23477,Car,Audi,A7,2014,black,5,125332,0,false
IC98532,Car,Audi,TT,2015,blue,2,0,0,false
XY54679,Car,Toyota,Yaris,2007,blue,5,0,0,false
TC78898,Car,Toyota,Corolla,2015,white,5,0,0,false
TP52614,Car,Toyota,Prius,2016,greys,5,0,0,false
JA72921,Car,Seat,Mii,2012,green,4,50020,0,false
SL35899,Car,Seat,Leon,2015,white,5,0,0,false
RT90456,Car,Mercedes,W212,2010,white,5,0,0,false
CR43683,Car,Mercedes,C220,2014,black,5,0,0,false
CA06145,Car,Tesla,P85D,2016,red,5,0,0,false
AI35328,Car,Mazda,6,2015,greys,5,0,0,false
KG63245,Car,Mazda,3,2016,white,5,0,0,false
GN88959,Car,Mazda,2,2011,greys,4,0,0,false
FT45101,Car,Ford,Mondeo,2015,greys,5,0,0,false

- Choose the needed filters by navigating the drop boxes.
- Press the *Search* button to apply the filters.

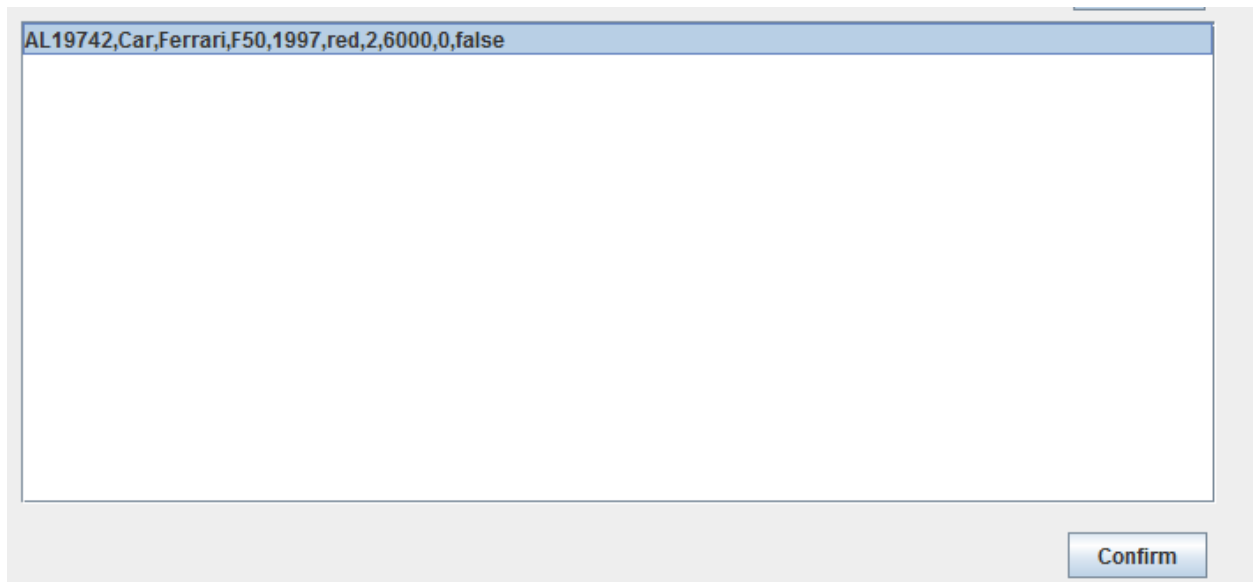
----- Filters -----

Make:  Model:  Year:

Color:  Seats:  Load:



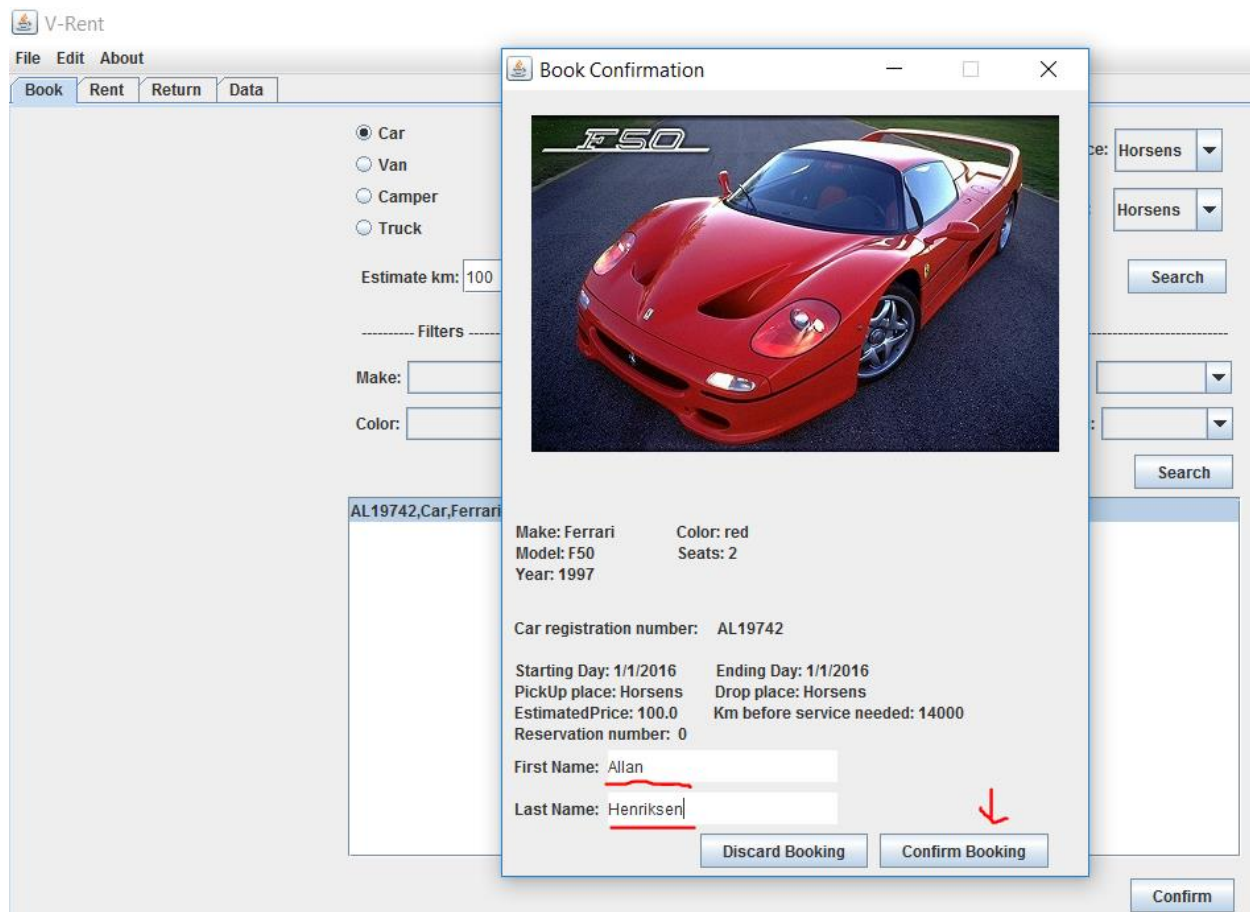
- Select the vehicle to rent.
- Press *Confirm*.



AL19742,Car,Ferrari,F50,1997,red,2,6000,0,false

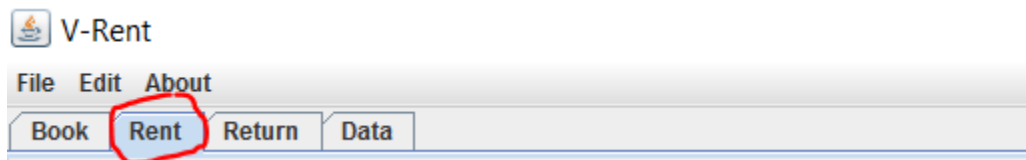
Confirm

- A pop up window will appear.
- Make sure the right vehicle is selected.
- Fill in the first name and last name of the client.
- Inform the client of the reservation number.
- Press *Confirm Booking* to store the booking information.



## How to rent a vehicle

- Go to the *Rent* tab.

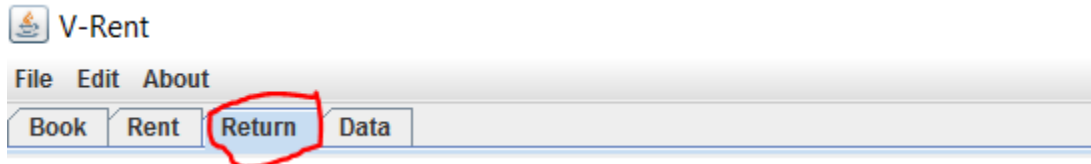


- Fill in the reservation number of the client and press *Enter* on the keyboard.
- The client name and vehicle information will now appear.

A screenshot of a web form titled 'Renting the car'. The form is divided into two main sections: 'Client information' on the left and 'Car details' on the right. The 'Client information' section contains fields for 'Reservation number:' (with '0' entered), 'First name:' (with 'Allan' entered), 'Last name:' (with 'Henriksen' entered), 'Driver licence number:', and 'Phone number:'. The 'Car details' section contains fields for 'Registration number:' (with 'AL19742' entered), 'Type:' (with 'Car' entered), 'Make:' (with 'Ferrari' entered), 'Model:' (with 'F50' entered), 'Year:' (with '1997' entered), 'Color:' (with 'red' entered), 'Seats:' (with '2' entered), 'Driven km:' (with '6000' entered), and 'Load size:' (with '0' entered). At the bottom right of the form are 'Save' and 'Cancel' buttons.

- Fill in the driver license number and phone number.
- Press *Save* to store the renting information.

- Go to the *Return* tab.



- Fill in the reservation number of the client and press *Enter* on the keyboard.
- The client information and vehicle information will now appear.

A screenshot of the 'Client information' form. The form has a title 'Client information' and a label 'Reservation number:' followed by a text input field containing the number '0'.

- Fill in how many kilometers the vehicle has driven.
- Choose the condition the vehicle was returned in.
- Fill in the final price. This should include the estimated price plus additional fees for returning the vehicle late or in a damaged condition.
- Press *Save* to store the returning information.

A screenshot of the 'Return information' form. The form has a title 'Return information' and several fields: 'Driven km:' with a text input field containing '6100'; 'Return date:' with three dropdown menus showing '02', 'june', and '2016'; 'Car condition:' with two radio buttons, 'Damaged' and 'Good', where 'Good' is selected; and 'Final price:' with a text input field containing '100'. At the bottom of the form, there are two buttons: 'Save' and 'Cancel'. A red arrow points down to the 'Save' button.