Semester Process Report

Michal Turcel 208276

Maria Avramescu 208208

Iustin Hilca 208230

Artemijs Zarovs 208279

DATE: 12.12.2013

Table of contents

1. Group policy

The members of the group decided to create and follow these rules :

- Everyone should attend the group meetings!
- If someone can't attend a group meeting, he / she should inform the rest of the group as soon as possible.
- All project team members confront issues directly and promptly.
- All members should be prepared for every meeting.
- Each meeting should have its own working agenda.
- Each member should pay attention to working process at group meetings.
- No games during the group meetings except smoke breaks.
- Try to be punctual, so the other members of the group do not have to repeat what was already said.
- Always keep the objective of the meeting in mind. Avoid talking about unrelated subjects.
- Every member should have expressed her or his opinions, and once convinced, should accept the conclusions.
- Be clear and short. Keep in mind that you are using the time of all the members.
- Every violation of these rules will be met with penalty of one pack of cookies in favour of rest of the group.

2. SWOT analysis

Michal Turcel

Strengths : Good in all programming courses, highly motivated, willing to help others.
Weakness : No experience with projects, forgetting about reports, reflections.
Opportunities : Teach other group members java, get used to do reports.
Threats : Bad weather, illness .

Maria Avramescu

Strengths : Good communicative skills, cooperative, hard-working, knowledge about Java, punctual, time-keeper.
Weakness : Not very good knowledge in databases, no experience to work in a team and to work for a project.
Opportunities : Improve my skills in English language, team-work and databases, interact with people from different cultures background, better understanding how I learn and how I can improve my performance, personal development.
Threats : I become impatient if I can not finish a task in the period of time that I have planned for it, work part-time, get exhausted sometimes.

Iustin Hilca

Strengths : Sociable, always up for something new , very ambitious.
Weakness : Database.
Opportunities : To improve my database knowledge.
Threats : Games.

Artemijs Zarovs

Strengths：Highly-adaptive and good logic thinking.
Weakness : Laziness, lack of concentration.
Opportunities : Improve my team-work and make me more hard-working person.
Threats : Weather, alcohol, games.

Team SWOT:

Strengths:

We are motivated because we all want to become engineers.
We are willing to help each other.
We divide our work and make a good schedule that fits to everybody in our team.
We are committed to the project and hard working.

Weakness :

We do not have experience with projects.
We did not work in a team so far.

Opportunities:

To improve skills in team-work .
To meet people from different culture backgrounds.
To improve our team skills in working for the projects.

Threats :

The personal conflicts that can appear between the members of the same group.
Bad weather.
Tendance to get distracted while we are working as a group.

3. Group roles

Michal Tucel

As a coordinator, my job is to plan group meetings. Before 22nd of November we met after courses in our class and after 22nd we met in our class at 10-11 in the morning nearly every day. Usually we stayed until late afternoon.

Maria Avramescu

As a recorder, I wrote down the notes that I considered necessary. I compiled the ideas of my mates and put them together on the table. That helped us to organize and understand the tasks we need to accomplish. At every meeting we had a conclusion about the subject that we studied.

As a checker I was looking for spelling mistakes when we had to send an assignment. I also made sure that the final solution meets all the requirements. I decided with my team how many hours we need to work for every hand-in and I made a schedule that fits everybody. One of my strengths is to be a good time-keeper, so the attribution to keep the members of the team aware of the time we spend in every meeting was a favorable one for me.

Iustin Hilca

As a monitor, I ensured that everyone was helped when he / she asked even if he / she asked ten times. I was more than glad to help my group mates and in the end everyone understood the task.

Artemijs Zarovs

As a presenter I want to be sure that communication in our group is easy and all tasks, solutions and strategies are discussable. Also I participated in all discussions and helped people make choice how they should do their work.

4. Reflections

Reflections as a team

From the beginning, every member from the team tried to have a contribution in the project. In this period of time, we worked on our project on daily basis. We did all the work together, one task at the time.

Usually we met in our classroom few hours after courses. In some days we get easily distracted and our work was not efficient. After our second hand-in, we realized that we can work faster if we divide the tasks. Every member could work independently in the time that fits him the best. We still met in school to put our work together, briefly explain to the rest of the group what we did and decide what are the next task that we need to accomplish. After the third hand-in, we realized that this way of completing our tasks is more efficient that the first one.

The motivation of the group was very strong in the beginning of the project. However, one of the group members started to lack the motivation to do his best and did only the necessary tasks, sometimes even late. Fortunately, in the report writing period, he started to work harder and did good job on project report.

After our first hand-in, we were required to create a group policy that had a good impact on our team at the beginning. After a short period of time, some of the members did not care so much about all the rules. In the end, the group contract did not improve our teamwork. However, the content of group contract is good, so the way to improve its positive impact on the group is to respect it.

Michal Turcel

From my point of view the team work in our group was good. In the beginning we were little confused about how we should work together because nobody of us had an experience with project like this. A few times I had a feeling that I am the one (sometimes the only one) who knows what to do. In the beginning it was about the code, later about diagrams. I not only explained to others but also showed them exactly what needs to be done and how. In worse case I did it myself and explained others (the code for first hand-in). However they do not learn much this way, so we started to write the code from scratch and later I tried to let others do and especially to manage the work on their own. For example I did all the diagrams (all two of them) for the second hand-in and for the last hand-in, Iustin was responsible for updating existing ones and creating new ones. I just explained him some details. Later we had some gaps, or things to improve in the way we cooperate. I would say it was really acceptable, since this was our first project and we managed to solve them (for example finding the way to split the work).

On the other hand we did not have any serious problems in our group. Never happened that nobody knew what we are supposed to do, or that somebody did not participate at all. Artyom did not do his best for the second and third hand-in. He lacked the motivation and tried to do as little as possible. However, if he was explicitly asked (or twice) to do something, he did it well and without any problem. Sad is the fact that he has a potential but the way he uses it is that he does very little and still has no problem to pass or to do what is necessary. Fortunately for last hand-in he started to do a lot more and I think it increased everybody's motivation.

Also I think that we worked faster when we met in school compared to the period before third hand-in. We managed to focus on our current task and we fulfilled new wishes really fast. Little tricky was when we thought that our program is done, and we have plenty time to finish reports and required appendices. So we slowed down a little and started with reports, but the problem was that the program was not done yet. There were many details that needed to be fixed, what took some time itself but also complicated already ongoing work,

mainly on project report (diagrams, use case descriptions, user guide).

I think this project was meant to teach us by doing. Although we did not do everything we were supposed to do (work logs, take responsibility for certain tasks / wishes, meetings), now we know what to do and for the next semester project we can start working the right way from the beginning.

Maria Avramescu

The biggest challenge for me in this project was to learn how to work in a team whose members belong to different cultures background. From the beginning of the project, I was very motivated to work hard and to support my team. All the period I felt very responsible for every task I needed to perform. I enjoyed keeping my sense of humor, to cooperate and socialize with the members of my team because I am aware of the fact that a good atmosphere in the group can improve dramatically the results.

One of the positive aspects of working in a group is that we can divide the work, which can help the job get done faster than if it was all done by one person. Sharing all my ideas with my mates and listen to their opinions helped me a lot to take the right decision for every task.

At the beginning, we met to work for the project every day after courses. This schedule exhausted me at a moment and I could feel that is not an efficient one for us. After a period of time, we decided that all the members can perform at home some specific tasks and meet in school to combine them and to take the final decisions together.

There were moments when some of us were more dedicated to the project than others. I had few arguments with Artemijs when I felt that we met and we wasted our time because he was more in the mood of playing games than working for the project. Mike is very skilled in programming languages and he made the first hand-in alone and explained to us. I felt bad about the fact he

wanted to do everything without any help. Afterwards I realized that he did not sacrifice for the team and it was just a goal that we wanted to achieve. We spoke with him about it and we started everything at the beginning, this time as a team. He became my best friend and I enjoyed working for the project more than I could expect. With the rest of group I had a very good relationship, although there were moments when we discussed about how to schedule the time when everyone can work together.

The group roles really made me to be aware of the fact that I am part of the team and I am responsible of the proper performance of the activities.

In the next project, I think I can improve my team-work through define the responsibilities of the members better and write down on paper the date of the meeting and the most important aspects of it.

Iustin Hilca

Since I am here I have learnt how to be responsible. By responsible, I understand that you need to respect a schedule and when you have to do a task, you need to do it because maybe your task depends on someone's work and the worst thing that you can do in a group is to waste somebody's time. When my coordinator, Michal Turcel, told me that I have to fix something in our program, I immediately started to work on the issue.

I gave my best in every situation. When we had a problem everyone started google for it and we did a good job. To be honest, we were lucky with Mike because he saved us most of the time when we had an issue with the program.

Everyone from our group is motivated because we all want to become it engineers. Like any group we had some moments when we were down, because a lot of things had to be done. When we found out that we need to make those appendices we all said "oh no, it can't be possible" but our motivation to become engineers helped us and we started working on the appendices.

We have been taught to share our culture with our group and everyone learnt something for everybody. When I was writing this, I asked mike something about Slovakia and I found out that Slovakians try to be a good person with everyone from other countries. When I talked to Arthyom, I found out that

Latvian people like to have some fun every day, and they like to drink a lot.

From working together I've learnt that Slovakians wants to work more than Latvians and sometimes more than Romanians. To best part in our group was that we worked a lot and we met every day to work for the SEP.

I took the responsibility for every task assigned to me. I took care of some panels for gui, project report, and some use cases, activity diagrams, use case descriptions, and some methods from our program.

In my opinion, we all showed interest and took on the responsibility for each task assigned to us, and we solved all the problems just in time.

In my group I was the monitor. As a monitor I provided a balanced opinion on all ideas and options. I was all the time ready to explain what actions I prefer. Every time when someone did not understand something about the program, I was there to help him, and make sure he understood.

Artemijs Zarovs

During this semester I was working with Maria, Mike and Iustin. It was long period and we got used to each other and I can say that I liked working with them of course we had some problems with each other and I want to be completely honest about it. Here I want to describe how I we worked together and what I think about my group-mates.

Mike was coordinator and main programmer in our team. His programing skills are on higher level than others. This was our advantage because he always did everything fast and we did not had any serious problems with code but also there is other side of it. For good learning student have to find his own way of learning, but with us there was no need to search solution for problems because Mike was fixing things before we even start to look for solution. We had some problems with understanding Mike's code and we needed to force him to explain everyone his methods of coding. As coordinator I think he did pretty good work, but also I think in our team there was no need for strong leader because in our team everybody knew that other are relying on them.

Maria was a checker and recorder. She did work I hate the most. Taking notes and documentation actually is more hard work than it sounds but I think she did it pretty well. As group-member I want to say that working with women is sometimes really hard but I also want to say that maybe it was harder for her to work with three guys. Well I got some experience working with her and it might help me in future.

Iustin was monitor in our team. He did great job because he actually cared about that everyone understands what we are doing and how everything is working. I enjoyed working with him the most and I really want to work with him in the future. One thing about him with double-sided edge is that he wants to work a lot sometimes there is need in break and forcing himself to action can bring zero results.

To sum up our teamwork I want to say that our team is really good balanced everybody has something special about him and here I mean not the way they work but the way they behave. Also we got used to each other and understood each other's needs. If somebody has some problems in his private field we try to help him. Sometimes people need some time to sort out his problems and everyone understands it.

About our project I want to say that only difficulty was to make priorities, choosing what is more important right now and what can be done later. When we figured that out we understood that as long as we do our work there will be no problems with schedule and there will be no lack of time.

During project period I received a lot of new experience and knowledge and I believe that in future it will help me a lot. I want to thank to all my team for this semester and to wish them good luck in the future.

5. List of tasks

In the beginning of our work for the project we did not decide who should be responsible for certain tasks. After second hand-in we started to divide work (wishes) and later also project / process report. However we did not clarify who is responsible for each wish or task.

For the next semester project it will be better for each member to have the responsibility for some specific tasks. That will motivate everybody to do the work as good as possible.

Michal Turcel

-database connection
-order vehicles by repair in
-graphical user interface
-vehicles to drive
-calculating price
-set price per additional km
-set discount and after how many days it applies
-process report
-requirements
-use case descriptions and diagrams until we added graphical user interface

Maria Avramescu

-today reserved vehicles
-graphical user interface
-auto campers availability
-Customer class
-set rent for vehicle
-process report
-javadoc comments

Iustin Hilca

-cancelling the reservation
-project report
-updating diagrams after adding graphical user interface
-adding new diagrams and use case descriptions (setting discount and rent, list of vehicles to drive).

Artemijs Zarovs

-list of returned vehicles
-project report
-testing
-user guide

6. Supervisor meetings

The meetings with the supervisors were not necessary at the beginning because we understood what we need to prepare for the assignments. We met them occasionally just to clarify what are the expectations of product owner and to have explained some of the wishes from the list.

In the period of working for the process and project report, we needed to meet the supervisors more often. None of our member had an experience with the reports before, so we had a lot of questions.

At every meeting, the supervisors were willing to help us and give good tips for every task.

01.12.2013

Consultation of use case diagram, activity diagrams, class diagram and SWOT analysis. We were required to add consequences to our group policy.

5.12.2013

At this meeting, the supervisor explained to us what should be included in our appendices. We had questions about javadoc comments.

10.23.2013

At this meeting, the supervisor gave us pieces of advice about the class diagram and the contents and structure of the process report (for example, we need to include the list of tasks and Bloom's forms).

7. Bloom's forms

Michal Turcel

Fill in this form – include it in your portfolio – discuss it with the rest of the group Date.....	Bloom's level	Keeping a portfolio	Reflecting on learning	System development	SCRUM	Java Programming	Object-oriented design and programming	UML	Web Programming	Database design	Written English	Spoken English	Team working	Sharing knowledge	Project planning	Presentation / exam skills
Excellent	6															
	5															
Good	4															
	3															
	2															
Basic	1															
No knowledge	0															

Maria Avramescu

Fill in this form – include it in your portfolio – discuss it with the rest of the group Date.....	Bloom's level	Keeping a portfolio	Reflecting on learning	System development	SCRUM	Java Programming	Object-oriented design and programming	UML	Web Programming	Database design	Written English	Spoken English	Team working	Sharing knowledge	Project planning	Presentation / exam skills
Excellent	6															
	5															
Good	4															
	3															
Basic	2															
	1															
No knowledge	0															

Iustin Hilca

Fill in this form – include it in your portfolio – discuss it with the rest of the group Date.....	Bloom's level	Keeping a portfolio	Reflecting on learning	System development	SCRUM	Java Programming	Object-oriented design and programming	UML	Web Programming	Database design	Written English	Spoken English	Team working	Sharing knowledge	Project planning	Presentation / exam skills
Excellent	6															
	5															
Good	4															
	3															
	2															
Basic	1															
No knowledge	0															

Artemijs Zarovs

Fill in this form – include it in your portfolio – discuss it with the rest of the group Date.....	Bloom's level	Keeping a portfolio	Reflecting on learning	System development	SCRUM	Java Programming	Object-oriented design and programming	UML	Web Programming	Database design	Written English	Spoken English	Team working	Sharing knowledge	Project planning	Presentation / exam skills
Excellent	6															
	5															
Good	4															
	3															
Basic	2															
	1															
No knowledge	0															

SEPI1 A 12 – Our Rent company

Project Report

Group 2:	Michal Turcel	208276
	Iustin Hilca	208230
	Artemijs Zarovs	208279
	Avramescu Maria	208208

Supervisors: Steffen Vissing Andersen

Mona Wendel Andersen

Hand in date: Friday, 13th of December 2013

Table of Contents

Table of Contents

1. Abstract

V-Rent is a Car Rental Company located in Horsens. It is owned by Van Motor who wants us to create a rental software system for his company. System based on Java must meet the requirements of the customer.

Program must be able to create reservation, rent already created reservations, list all cars, list all available cars, list specified cars, add car to the system, list information about cars and calculate payment. Also our program must organize and store information about reservations.

Software which we developed specifically for this customer matches all the requirements and can be used by different type of staff, like customer representative, manager, clerk, owner and handyman. The program also has database integration and possibility of future extension.

2. Introduction

Renting a car seems to be simple task both for the customer and employee at a car rental company. A car rental agency is a company that rents automobiles for short periods of time (generally ranging from a few hours to a few weeks) for a fee. It is often organized with numerous local branches (which allow a user to return a vehicle to a different location), and primarily located near airports or busy city areas and often complemented by a website allowing online reservations.

The V-Rent is a Car Rental Company located in Horsens. It's owned by Van Motor who wants us to create a rental software system for his company because the old method using pen and paper is not efficient anymore.

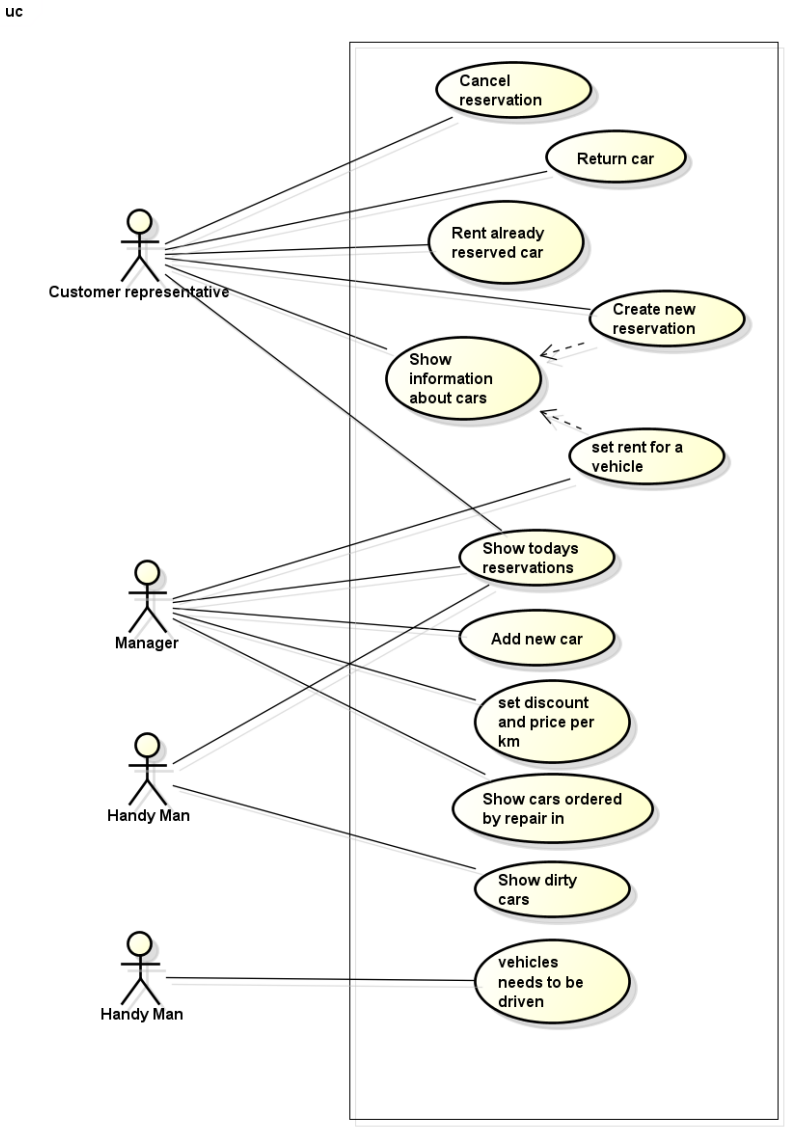
The V-Rent Company wants a single user system, to handle the rentings for their customers. The manager wants to have an easy way to reserve cars, see a list of all vehicles, see which of these are available at a specific time, and to store information like vehicle data and customer data in a database. The manager does not want to have an online renting system.

3. Analysis

3.1 Requirements

- A customer representative can reserve a car.
- The system needs to store information about vehicles.
- Customer representative can see details about the vehicles.
- The system needs to store name of the customer, car, pick-up and return time, pick-up and return location and approximate amount of kilometers for every reservation.
- The system will check if the car is available.
- The system will not allow the customer to rent a car if it is not available.
- The system will be able to filter cars based on the preferences of the customer: vehicle type, model, availability during a specific period or load size if it is a van.
- The system will show information about reservations done by a specific customer based on phone number or name.
- Manager should be able to add a vehicle.
- The system should be able to read and write information about vehicles and reservations into database.
- Customer representative can see details about reservations for today.
- Manager can see list of cars ordered by amount of kilometers left until next service.
- Customer representative can cancel a reservation.
- Handy man can see list of returned (dirty) vehicles.
- System must have Graphical User Interface.
- Driver can see which vehicle needs to be driven from which location to which location.
- Clerk must be able to compute the price of the rental.
- Manager must be able to set on how many days discount applies.
- System must allow renting of auto camper only for one weekend or one full week at a time.
- System must be able to receive and save customer data.
- Rent price per day for every vehicle, and additional prices for kilometers.

3.2 Use Case Diagram



Cancel reservation: Customer representative can cancel a reservation.

Return car: The car is made dirty, and reservation is deleted.

Create new reservation: Customer representative can make a reservation.

Show information about cars: Customer representative is asked to choose first preference to be specified and information about all the vehicles meeting the requirements will be printed.

Rent already reserved car: Customer representative will choose a customer by entering his phone number, a list of all reservations of the customer is printed on the screen and customer representative is asked to choose one of them.

Show todays reservations: System shows all reservations for today.

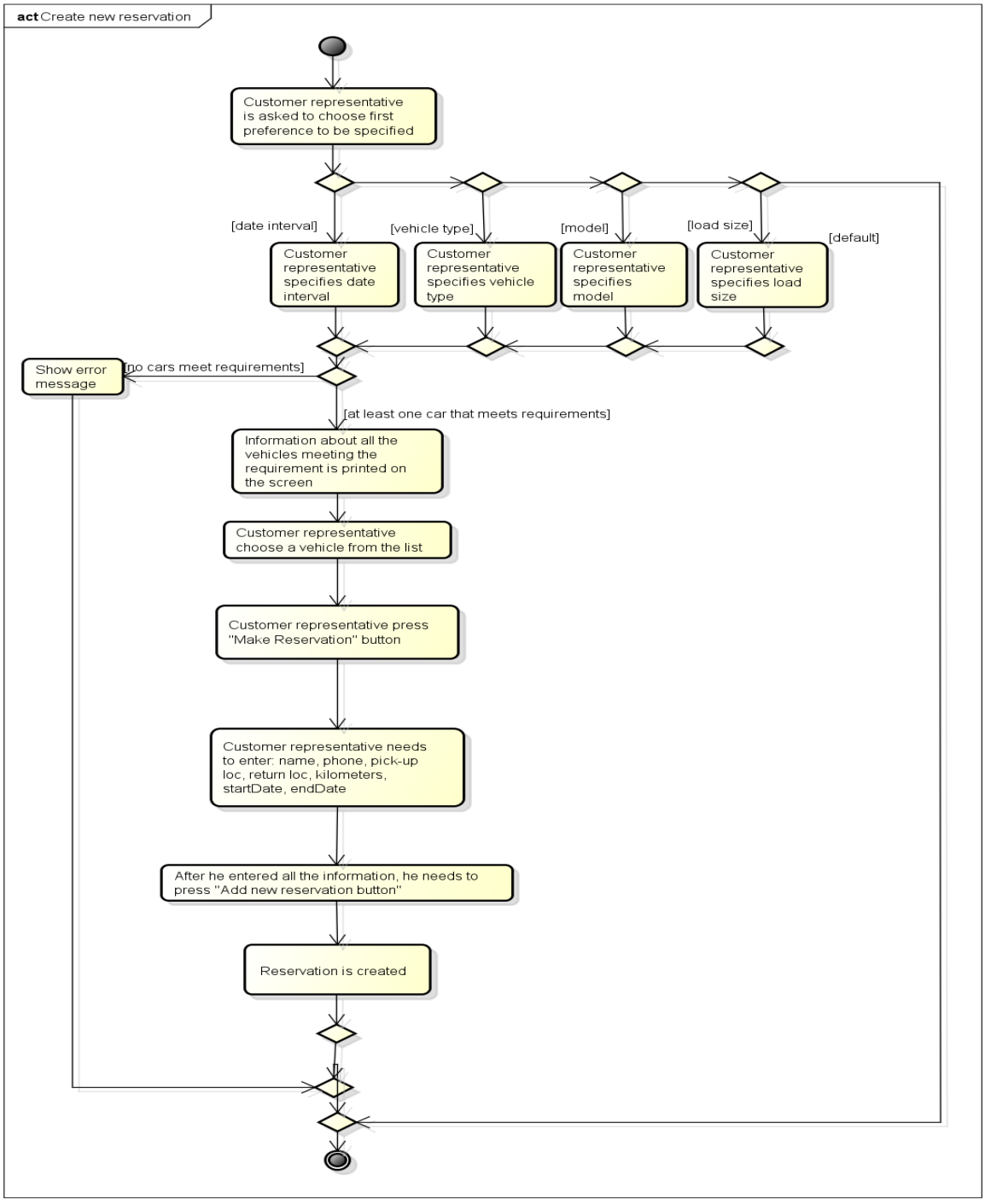
Add new car: Manager is asked to enter information about the new vehicle: type, registration number, make, model, year, mileage, rent price for day and load size for vans and the vehicle is added to the list.

Show cars ordered by repair in: Information about all the cars ordered by amount of kilometers left to next service is shown on the screen.

Show dirty cars: All dirty cars will be shown.

Check Appendix 2 – Use Case Description

3.3 Activity Diagram



"Create new reservation" is the most essential features of the system because the purpose of this project is to make a program that rents cars. The Create new reservation activity diagram displays step-by-step the process of creating new reservation.

Check Appendix 1 - Activity Diagrams.

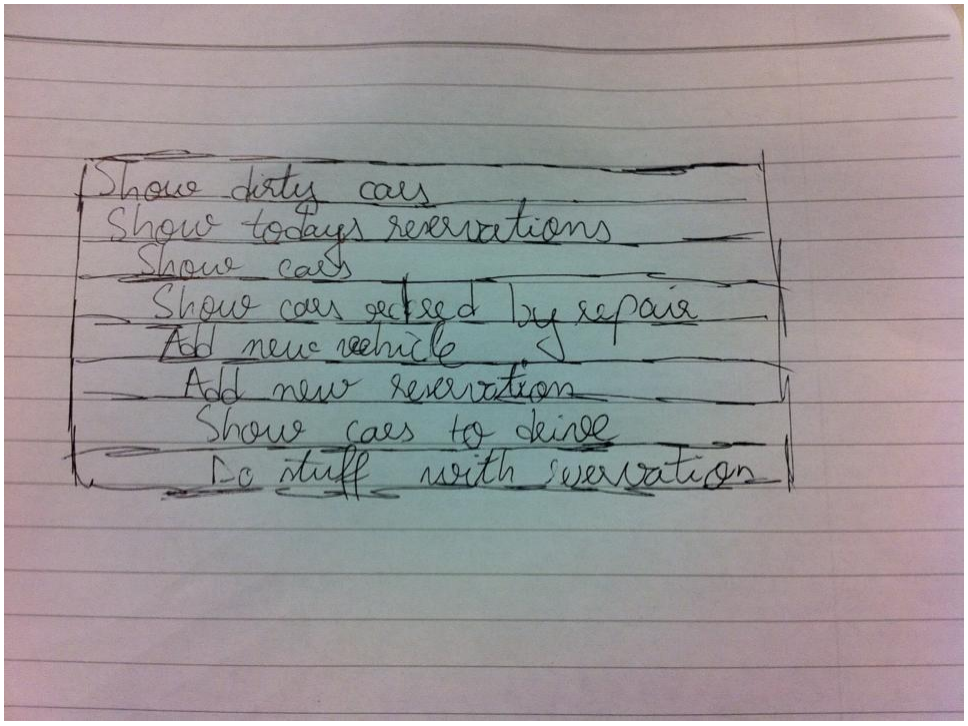
4. Design

4.1 Graphical User Interface (GUI)

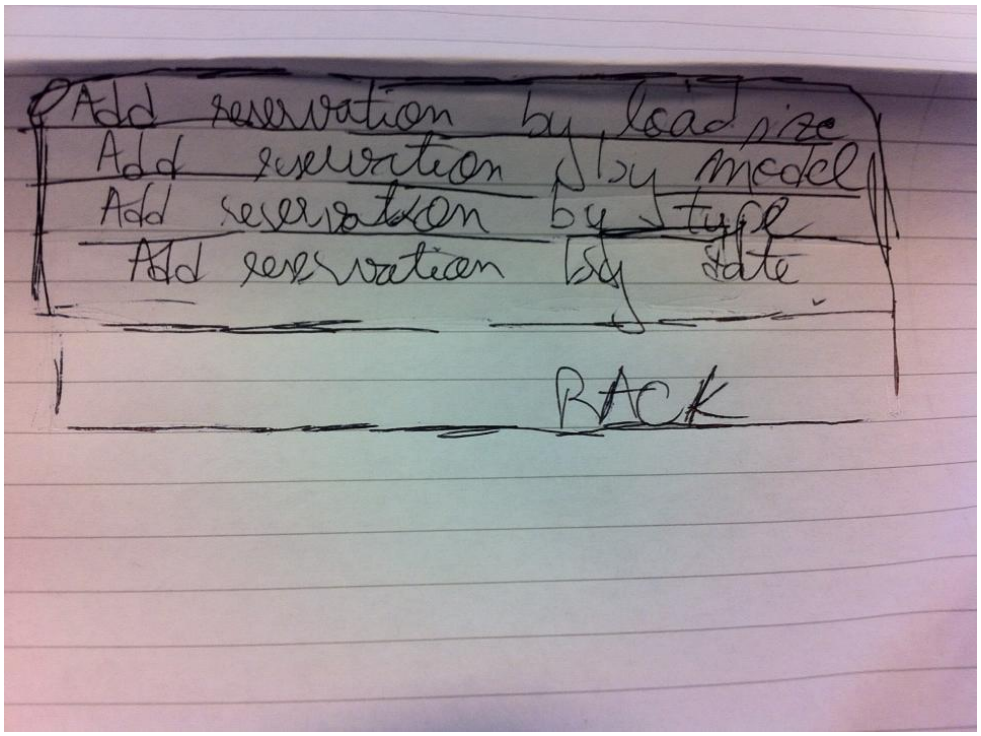
Main menu:

In the beginning of the GUI implementation, the sketch of the GUI was drawn on the paper, which consist the main menu with cases representing the main features of the system.

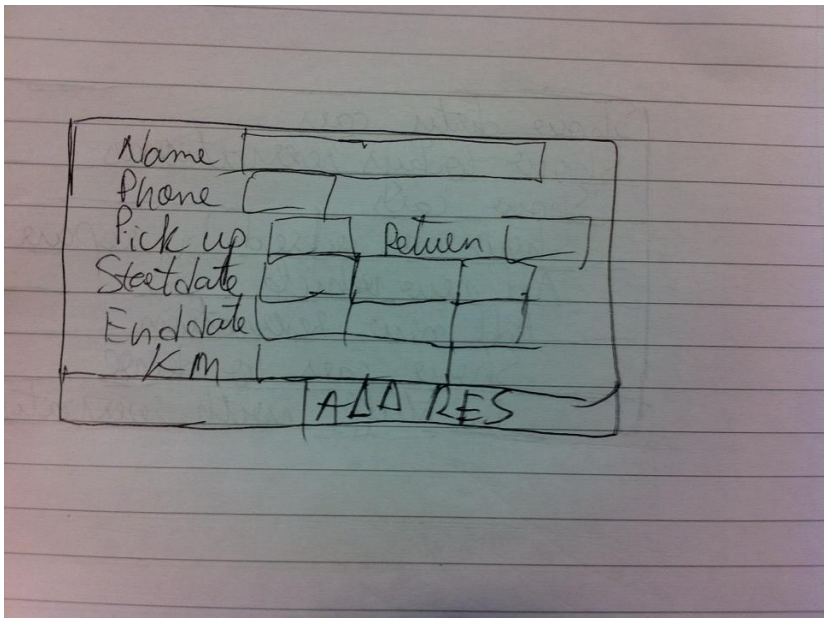
A simple interface was chosen for the main menu, with a grid list of buttons. Each button has a name which displays his name. In this way the user can interact with the system very easily.



The most important method from our program is “Add new reservation”. This method was designed to display 4 buttons. The user is allowed to make a reservation by load size, model, type, date.

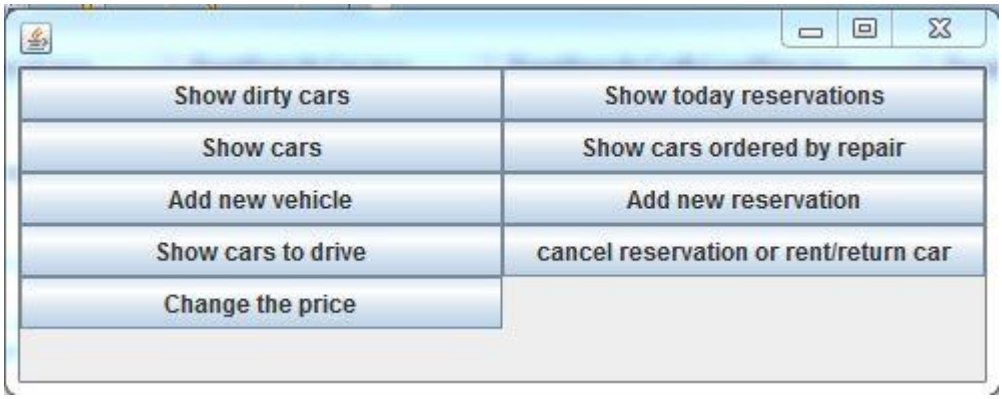


After the customer representative chooses what kind of reservation he wants to add, he will be redirected to a new panel which consists some labels, text fields and one button “Add Res”.



Result

The result looks almost the same as the sketch looks with another organization of the buttons.



4.2 Database Design

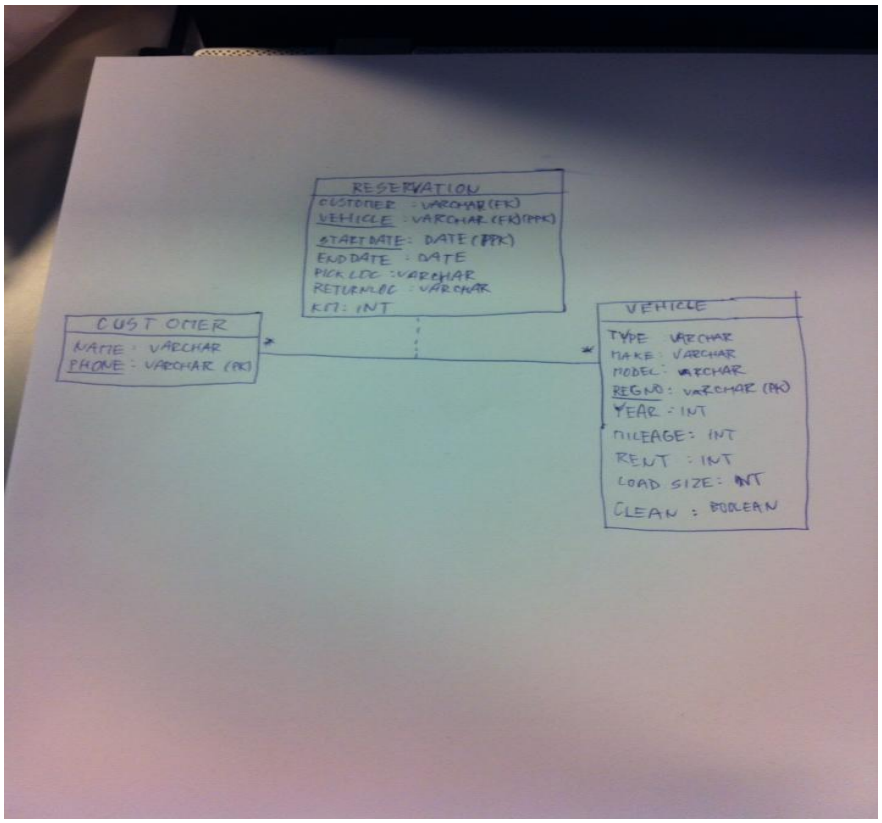
The first thing that we realized is that we need 3 tables: reservation, vehicle and customer. In order to do this we had to draw it on paper and to set the foreign and primary keys.

The first table created was “customer” which consists of: name and phone, where phone is primary key.

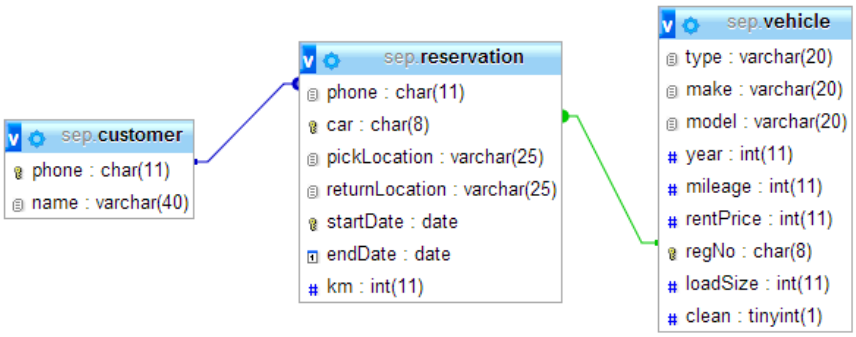
The second table created was “vehicle” which consists of: type, make, model, regNo, year, mileage, rent, loadSize, clean, where regNo is primary key.

The last table created was “reservation” which consists of: customer, vehicle, startDate, endDate, pickLoc, returnLoc, km, where customer is foreign key, vehicle is foreign key and part of primary key and startDate is part of primary key.

Here is the database drawn on the paper.



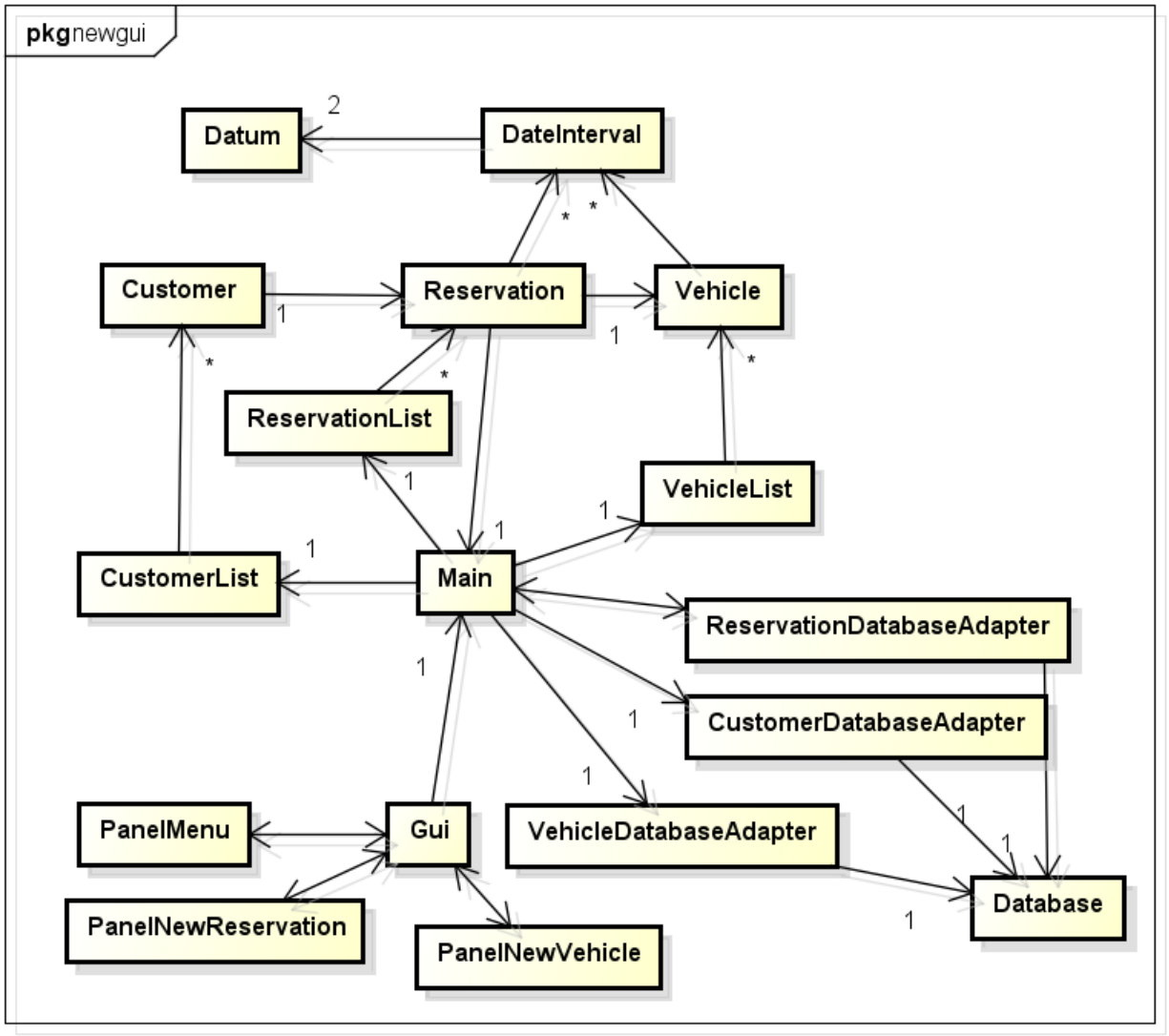
Final result:


sep.customer
phone : char(11)
name : varchar(40)
sep.reservation
phone : char(11)
car : char(8)
pickLocation : varchar(25)
returnLocation : varchar(25)
startDate : date
endDate : date
km : int(11)
sep.vehicle
type : varchar(20)
make : varchar(20)
model : varchar(20)
year : int(11)
mileage : int(11)
rentPrice : int(11)
regNo : char(8)
loadSize : int(11)
clean : tinyint(1)

4.3 Model class diagram

This basic model shows us the structure of the system. “Vehicle” class is used to create new vehicle in the system and information about it. “Vehicle list” class stores all vehicles and information about them. Every object of class “Reservation” stores information about a single reservation. “Reservation list” is used to hold a list of reservations and it can filter them by various criteria. “Datum” and “Date interval” are used to add dates and date intervals to reservations. Also we have 3 adapter classes that connects system to our database. “Main” class contains main functions of our system. “VehicleDatabaseAdapter”, “ReservationDatabaseAdapter” and “CustomerDatabaseAdapter” are used for connecting database with the “Main” class and help us to add maintain a constant connection with database when we do some changes like add cars/reservation. “GUI” is used to create a friendly way to communicate with the user. “PanelNewVehicle”, “PanelNewReservation” creates an interesting way to interact with the user, while he/she wants to add a new vehicle or a new reservation into the system. “PanelMenu” helps us to control all the panels from “GUI”. There are many other panels but we showed only 3 of them.

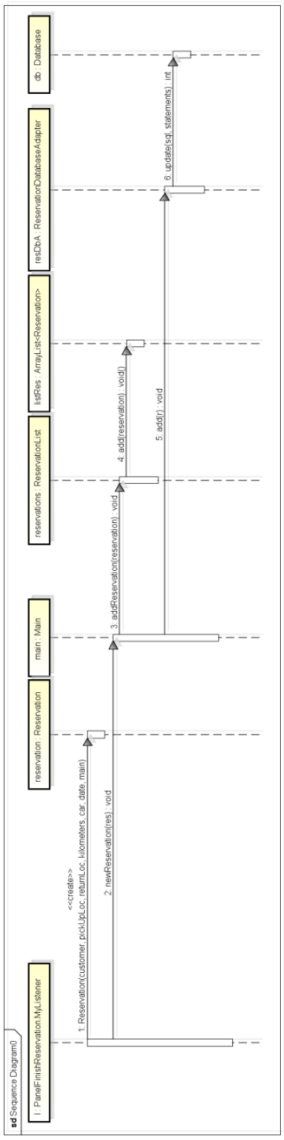
4.4 Class diagram



Check Appendix 3 for extended Class Diagram

4.5 Sequence diagram

The sequence diagram shows process of saving a new reservation in our system. Fisrtly, a new instance of the Reservation is created, using contents of text fields in PanelFinishReservation. This object is passed to newReservation method in Main class, which firstly saves it in Java model and then it tries to save it in the database. Saving in Java model is dony by calling addReservation method in the ReservationList. That will add the Reservation in the ArrayList. Saving in the database is done by calling add method in ReservationDatabaseAdapter, which calls update method in the Database class to insert new customer into our database.



5. Implementation

One of the most important methods of our program is Create new reservation.

In order to create a new reservation we need to press the button “Add new reservation”. The code below displays 4 buttons which offers us the possibility to make a reservation by specifying the car by load size, model, type, date interval.

```
package newgui;

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JPanel;

/**
 * A panel allowing the user to choose a property to specify when specifying
a
 * car.
 *
 * @author Maria Avramescu
 * @version 1.0 05/12/2013
 */

public class PanelSpecifyCar extends JPanel
{
   /**
    * An inner class that handles the button clicks
    *
    * @author Maria Avramescu
    * @version 1.1 05/12/2013
    */

   private class MyListener implements ActionListener
   {

      /**
       * A method that shows panel to specify vehicle by selected property
       */

      @Override
      public void actionPerformed(ActionEvent e)
      {
         // TODO Auto-generated method stub

         switch (e.getActionCommand())
         {
            case "specify car by load size":
               gui.showPanelSpecifyCarByLoadSize(create);
```

```
                break;
             case "specify car by model":
                gui.showPanelSpecifyCarByModel(create);
                break;
             case "specify car by type":
                gui.showPanelSpecifyCarByType(create);
                break;
             case "specify car by date interval":
                gui.showPanelSpecifyCarByDateInterval(create);
                break;
          }

       }
    }

    private Gui gui;
    private JButton addResByType;
    private JButton addResByModel;
    private JButton addResByLoadSize;
    private JButton addResByDateInterval;

    private boolean create;

    /**
     * One argument-constructor that creates panel to choose property of
vehicle
     * to specify.
     *
     * @param gui
     *            reference to instance of the Gui class
     */

    public PanelSpecifyCar(Gui gui)
    {
       super(new GridLayout(3, 1, 3, 3));
       this.create = true;
       this.gui = gui;
       addResByLoadSize = new JButton("specify car by load size");
       addResByModel = new JButton("specify car by model");
       addResByType = new JButton("specify car by type");
       addResByDateInterval = new JButton("specify car by date interval");
       this.add(addResByLoadSize);
       this.add(addResByModel);
       this.add(addResByType);
       this.add(addResByDateInterval);
       MyListener l = new MyListener();
       addResByLoadSize.addActionListener(l);
       addResByModel.addActionListener(l);
       addResByType.addActionListener(l);
       addResByDateInterval.addActionListener(l);
    }

    /**
     * A method setting the create flag
     *
     * @param create
     *            true if a reservation should be created after choosing a
```

```
     *            vehicle, false if user should be able to change the rent of
     *            vehicle instead
     */

   public void setCreate(boolean create)
   {
      this.create = create;
   }
}
```

If you want to specify a car by load size you need to press the button " specify by load size" button. By pressing the button, the program will go to PanelSpecifyCarByLoadSize panel. In this panel you need to specify a car by entering the load size in the JTextField. After the load size is inserted , you have 4 option: Update, Back, Set rent, Make reservation.

```
package newgui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JTextField;

/**
 * A panel allowing the user to see the list of vehicles with specified load
 * size.
 *
 * @author Maria Avramescu
 * @version 1.0 05/12/2013
 */

public class PanelSpecifyCarByLoadSize extends JPanel
{

   /**
    * An inner class that handles the button clicks
    *
    * @author Maria Avramescu
    * @version 1.1 05/12/2013
    */

   private class MyListener implements ActionListener
   {

      /**
       * A method that updates the list of vehicles, shows panel to set the
       * rent, finish reservation or menu.
       */

      @Override
      public void actionPerformed(ActionEvent e)
```

```
    {
        // TODO Auto-generated method stub
        if (e.getActionCommand().equals("Update"))
        {
            update();
        }
        else if (e.getActionCommand().equals("back"))
        {
            gui.showPanelMenu();
        }
        else if (e.getActionCommand().equals("set rent"))
        {
            Object o = list.getSelectedValue();
            if (o != null)
                gui.showPanelChangeCarRent((Vehicle) o);
        }
        else
        {
            Object o = list.getSelectedValue();
            if (o != null)
                gui.showPanelFinishReservation((Vehicle) o);
        }
    }

}

private JLabel loadSizeL;
private JTextField loadSizeT;
private Gui gui;
private JButton update;
private JButton reserve;

private JButton back;

private JList<Object> list;

/**
 * One argument-constructor that creates panel to show vehicles with
 * specified load size.
 *
 * @param gui
 *            reference to instance of the Gui class
 */

public PanelSpecifyCarByLoadSize(Gui gui)
{
    this.gui = gui;
    back = new JButton("back");
    loadSizeL = new JLabel("Specify load size: ");
    loadSizeT = new JTextField(5);
    update = new JButton("Update");
    reserve = new JButton("Make Reservation");

    list = new JList<Object>();
    list.setVisibleRowCount(5);
    MyListener l = new MyListener();
    reserve.addActionListener(l);
```

```
        update.addActionListener(l);
        back.addActionListener(l);

        add(loadSizeL);
        add(loadSizeT);
        add(update);
        add(reserve);
        add(back);
        add(list);

    }

    /**
     * A method setting the create flag
     *
     * @param create
     *            true if a reservation should be created after choosing a
     *            vehicle, false if user should be able to change the rent of
     *            vehicle instead
     */

    public void setCreate(Boolean create)
    {
        if (create)
            reserve.setText("Make Reservation");
        else
            reserve.setText("set rent");
    }

    /**
     * A method updating the list of vehicles
     */

    private void update()
    {
        int s = Integer.parseInt(loadSizeT.getText()); // take the load from
the
                                                       // textField
        list.setListData(gui.getMain().getCarsByLoad(s).toArray());
        gui.pack();
    }

}
```

To continue with creating the reservation you need to press “Make reservation” button. By pressing the button, the program will go to “PanelFinishReservation” panel. In this panel we can find 11 JTextFields: 1 textfield for “name”, 1 textfield for “phone”, 1 textfield for “pick-up location”, 1 textfield for “return location”, 1 textfield for “kilometers”, 3 textfields for “start date”, and 3 textfields for “end date”. The button “Find” allows us to search a customer by name/phone in the database. If the customer is found , his name or phone will be in the JList “customers”. In case we didn’t find any customer with that name or phone we can add a new one in the system. In order to finalize the reservation process you need to press the “Add new

reservation" button. By pressing the "add new reservation" button, it means that the following code line will be accessed "gui.getMain().newReservation(res);"

```
package newgui;

import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JTextField;

/**
 * A panel that allows user to create a new reservation after choosing a car.
 *
 * @author Maria Avramescu
 * @version 1.0 07/12/2013
 */

public class PanelFinishReservation extends JPanel
{

   /**
    * An inner class that handles button clicks for filtering / creating new
    * customers
    *
    * @author Maria Avramescu
    * @version 1.1 07/12/2013
    */

   private class CustomerListener implements ActionListener
   {

      /**
       * A method that updates list of customers, or saves new customer in
Java
       * model and database
       */

      @Override
      public void actionPerformed(ActionEvent e)
      {
         // TODO Auto-generated method stub
         switch (e.getActionCommand())
         {
            case "name":
               customers.setListData(gui.getMain().getByName(nameT.getText())
                     .toArray());
               gui.pack();
               break;
            case "phone":
               customers.setListData(new Object[] { gui.getMain()
                     .getCustByPhone(phoneT.getText()) });
               gui.pack();
```

```
                break;
             case "Add":
                String name = nameT.getText();
                String phone = phoneT.getText();
                if (name.length() > 1
                      && gui.getMain().getCustByPhone(phone) == null)
                {
                   gui.getMain().newCustomer(new Customer(name, phone));
                }
          }
       }

   }

   /**
    * An inner class that handles button clicks for creating reservation or
    * returning to menu
    *
    * @author Maria Avramescu
    * @version 1.1 05/12/2013
    */

   private class MyListener implements ActionListener
   {

      /**
       * A method that shows error message, creates and saves a new
reservation
       * in Java model and database or shows the menu.
       */

      @Override
      public void actionPerformed(ActionEvent e)
      {
         // TODO Auto-generated method stub
         if (e.getActionCommand().equals("back"))
         {
            reset();
            gui.showPanelMenu();
            return;
         }
         Object o = customers.getSelectedValue();
         if (o == null)
         {
            congrat.setText("select customer");
            return;
         }
         Customer customer = (Customer) o;
         String pickUpLoc = pickUpLocationT.getText();
         String returnLoc = returnLocationT.getText();
         int kilometers = Integer.parseInt(kmT.getText());
         int startDay = Integer.parseInt(startDayT.getText());
         int startMonth = Integer.parseInt(startMonthT.getText());
         int startYear = Integer.parseInt(startYearT.getText());
         int endDay = Integer.parseInt(endDayT.getText());
         int endMonth = Integer.parseInt(endMonthT.getText());
         int endYear = Integer.parseInt(endYearT.getText()); // take the load
```

```
                                                                    // from the
                                                                    // textField
        Datum startDate = new Datum(startDay, startMonth, startYear);
        Datum endDate = new Datum(endDay, endMonth, endYear);
        DateInterval interval = new DateInterval(startDate, endDate);
        if (!v.isAvailable(interval))
        {
           congrat.setText("vehicle unavailable");
           return;
        }
        Reservation res = new Reservation(customer, pickUpLoc, returnLoc,
              kilometers, v, interval, gui.getMain());
        gui.getMain().newReservation(res);
        congrat.setText("Congrulations!");
        addNewReservation.setText("back");
     }

  }

  private Gui gui;
  private JLabel startDateL;
  private JTextField startDayT;
  private JTextField startMonthT;
  private JTextField startYearT;
  private JLabel endDateL;
  private JTextField endDayT;
  private JTextField endMonthT;
  private JPanel panelStart;
  private JPanel panelEnd;
  private JTextField endYearT;
  private JList<Object> reserved;
  private JButton nameUpdate;
  private JButton phoneUpdate;

  private JList<Object> customers;
  private JButton newCust;
  private Vehicle v;
  private JButton addNewReservation;
  private JLabel congrat;

  private JTextField nameT, phoneT, pickUpLocationT, returnLocationT, kmT;

  private JLabel nameL, phoneL, pickUpLocationL, returnLocationL, kmL;

  /**
   * One argument-constructor that creates panel for creating a reservation.
   *
   * @param gui
   *           reference to instance of the Gui class
   */

  public PanelFinishReservation(Gui gui)
  {
     super(new BorderLayout());
     JPanel main = new JPanel(new GridLayout(6, 2, 3, 3));
     JPanel customer = new JPanel(new BorderLayout());
     JPanel customerSouth = new JPanel(new BorderLayout());
```

```
        JPanel customerNorth = new JPanel(new GridLayout(2, 2));
        JPanel nameTextButton = new JPanel(new BorderLayout());
        JPanel phoneTextButton = new JPanel(new BorderLayout());
        nameUpdate = new JButton("Find");
        nameUpdate.setActionCommand("name");
        phoneUpdate = new JButton("Find");
        phoneUpdate.setActionCommand("phone");
        customers = new JList<Object>();
        newCust = new JButton("Add");
        CustomerListener o = new CustomerListener();
        nameUpdate.addActionListener(o);
        phoneUpdate.addActionListener(o);
        newCust.addActionListener(o);
        this.gui = gui;
        panelStart = new JPanel(new GridLayout(1, 4, 3, 3));
        panelEnd = new JPanel(new GridLayout(1, 4, 3, 3));
        addNewReservation = new JButton("Add new reservation ");
        nameT = new JTextField(7);
        phoneT = new JTextField(7);
        pickUpLocationT = new JTextField(7);
        returnLocationT = new JTextField(7);
        kmT = new JTextField(7);
        nameL = new JLabel("Name :");
        phoneL = new JLabel("Phone :");
        pickUpLocationL = new JLabel("Pick up location :");
        returnLocationL = new JLabel("Return location :");
        kmL = new JLabel("Kilometers :");
        startDateL = new JLabel("Start date :");
        endDateL = new JLabel("Name :");
        MyListener l = new MyListener();
        addNewReservation.addActionListener(l);
        congrat = new JLabel("");
        reserved = new JList<Object>();
        customerNorth.add(nameL);
        nameTextButton.add(nameT, BorderLayout.CENTER);
        nameTextButton.add(nameUpdate, BorderLayout.EAST);
        customerNorth.add(nameTextButton);
        customerNorth.add(phoneL);
        phoneTextButton.add(phoneT, BorderLayout.CENTER);
        phoneTextButton.add(phoneUpdate, BorderLayout.EAST);
        customerNorth.add(phoneTextButton);
        customerSouth.add(customers, BorderLayout.CENTER);
        customerSouth.add(newCust, BorderLayout.EAST);
        customer.add(customerNorth, BorderLayout.NORTH);
        customer.add(customerSouth, BorderLayout.SOUTH);
        main.add(pickUpLocationL);
        main.add(pickUpLocationT);
        main.add(returnLocationL);
        main.add(returnLocationT);
        main.add(kmL);
        main.add(kmT);
        startDateL = new JLabel("Start date :");
        startDayT = new JTextField(2);
        startMonthT = new JTextField(2);
        startYearT = new JTextField(4);
        endDateL = new JLabel("End date :");
        endDayT = new JTextField(2);
```

```
        endMonthT = new JTextField(2);
        endYearT = new JTextField(4);
        panelStart.add(startDayT);
        panelStart.add(startMonthT);
        panelStart.add(startYearT);
        panelEnd.add(endDayT);
        panelEnd.add(endMonthT);
        panelEnd.add(endYearT);
        main.add(startDateL);
        main.add(panelStart);
        main.add(endDateL);
        main.add(panelEnd);
        main.add(addNewReservation);
        main.add(congrat);
        add(customer, BorderLayout.NORTH);
        add(main, BorderLayout.CENTER);
        add(reserved, BorderLayout.SOUTH);
    }

    /**
     * A method that clears the text fields
     */

    public void reset()
    {
        addNewReservation.setText("Add new reservation ");
        nameT.setText("");
        phoneT.setText("");
        pickUpLocationT.setText("");
        returnLocationT.setText("");
        kmT.setText("");
        congrat.setText("");
    }

    /**
     * A method saving a vehicle that will be reserved
     *
     * @param v
     *            the vehicle
     */

    public void saveCar(Vehicle v)
    {
        this.v = v;
        reserved.setListData(v.getReserved().toArray());
        gui.pack();
    }
}
```

6. Testing

6.1 Testing methods according to Use Case

In testing part the main goal was to check every stage of work. Our goal is to validate the design of GUI and check all our implementations. During system development system was checked for different bugs errors. For full system functionality GUI and database design and implementations had to be completed. The complete system was tested at the end of development. All test actions were performed according to user guide.

Create new reservation

This is the main function of our program and created reservations are used in other functions. The test was performed by going through algorithm from user guide. Short description of actions performed:

1. Press button "Create reservation" in main frame of GUI.

2. Specify the method how car will be chosen (by load size, by model, by type, by date interval).

3. Fill asked information and press create button.

Show cars

"Show cars" is similar to "Create new reservation" little difference in actions is that we don't have 3rd step. Here is short description of actions:

1. Press button "Show cars" in main frame of GUI.
2. Specify the method how cars will be chosen (by load size, by model, by type, by date interval).

Add new vehicle

"Add new vehicle" is used for adding car to the system. Cars are used in other functions and what is more important is used in creating new reservations. Actions to perform:

1. Press button "Add new vehicle".
2. Fill all fields with asked information about the cars.
3. Press add button and the car is added to system.

Show dirty/Show cars to drive / Show today reservations / Show cars ordered by repair

These methods are similar to use because there is only one action for each, simply choose button in main frame and system will list you asked information.

All the rest actions were also tested using algorithm from users guide. Here are displayed all asked functions and their test status and results. All these functions are performed according to use cases.

Cancel reservation	Tested	Passed
Return car	Tested	Passed
Rent already reserved car	Tested	Passed
Create new reservation	Tested	Passed
Show information about cars	Tested	Passed
Set rent for a vehicle	Tested	Passed
Show todays reservations	Tested	Passed
Add new car	Tested	Passed
Set discount and price per km	Tested	Passed
Show cars ordered by repair in	Tested	Passed
Show dirty cars	Tested	Passed
Vehicles needs to be driven	Tested	Passed

Test showed that all requirements are complete and program works exactly as expected.

6.2 Test of GUI

During full test we found out that GUI needs some modifications. All software performed very well and all requirements were complete, but we found that the user might have some problems because of lack of “Back” buttons.

Simple miss-click can start some procedure which cannot be stopped until you close the program or finish the action. This small problem was fixed and now GUI is easy to use, simple and does exactly what customer requires.

7. Results

The program was created in order to facilitate the processes involved in a renting a car system. When the employee creates a reservation for the customer representative, a car is reserved for a specific time period. When the employee wants to create a reservation he can look into the database to see if the customer representative is already in the database. The main purpose of the system is to create an easy way to reserve cars, see a list of all vehicles, see which of these are available at a specific time, to store information like vehicle data and customer data in a database and to keep track of the reservations in order to avoid double bookings.

The GUI will be changed and we should only make it to work well for the company the next couple of years. It has a grid layout in the main menu with buttons which represent many different actions.

7.1 Known errors in the system

A known error in the system is that the “rent” button doesn’t do anything.
Another error is that you can return a car without renting it.

8. Conclusion

Main goal of our project was to create software for car renting company providing electronic reservation and other facilities, easy-to-use systems which is needed for this specific customer. Program is designed for company staff and can be used by different actors for different actions. Our goal was reached and created program fulfils all customers' requirements. All requirements are complete the system is ready-to-use.

During development period some problems were faced. These problems can be divided in two parts, technical and logical. Technical problems are mainly problems with Java code and different types errors in program. Logical problems are because we always needed to decide "how this should work" and other problems with algorithm and class connections.

As a conclusion, project is successfully completed and our system fulfils all the requirements. The final version of program is designed and created unique for specific customer.

9. References

9.1 Books

- Gaddis. T. Starting out with Java. 3rd Edition.

- Database Systems A Practical Approach to Design, Implementation and Management, Thomas Connolly & Carolyn Begg, Fifth Edition

9.2 Additional resources

Presentations from classes

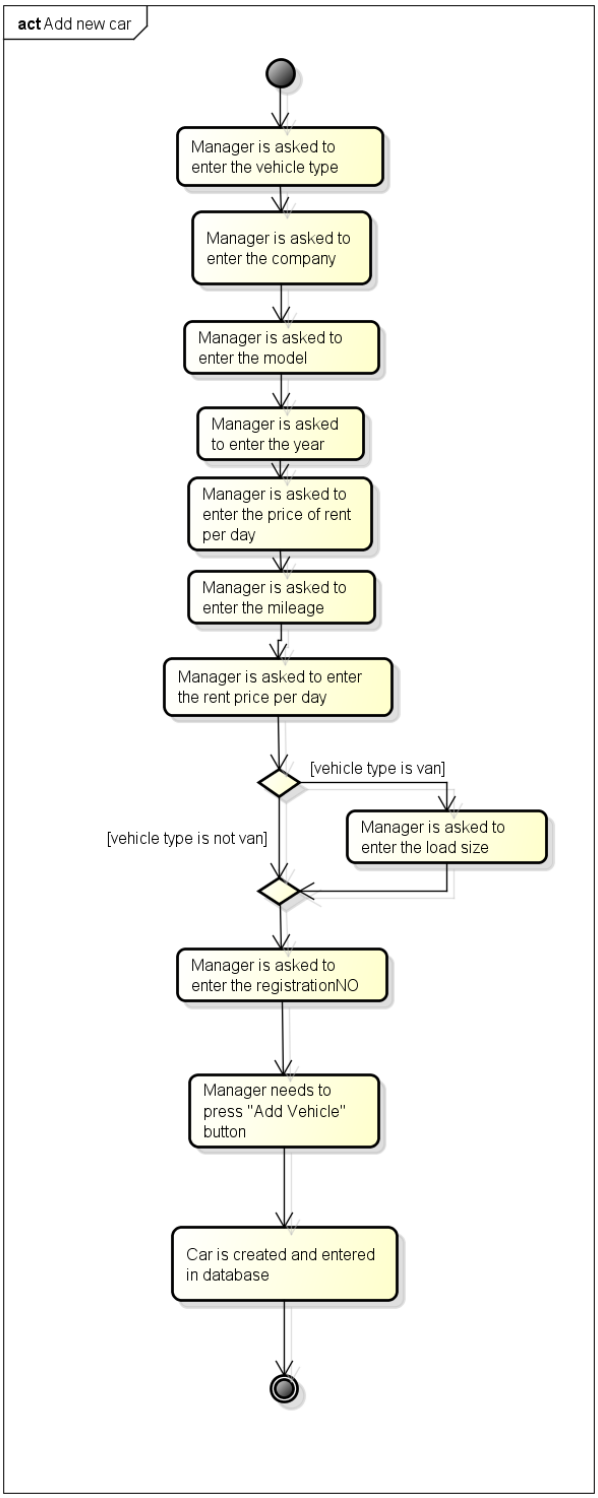
Website and Database design (WDDI1)

JAVA 1 and UML (SDJ1)

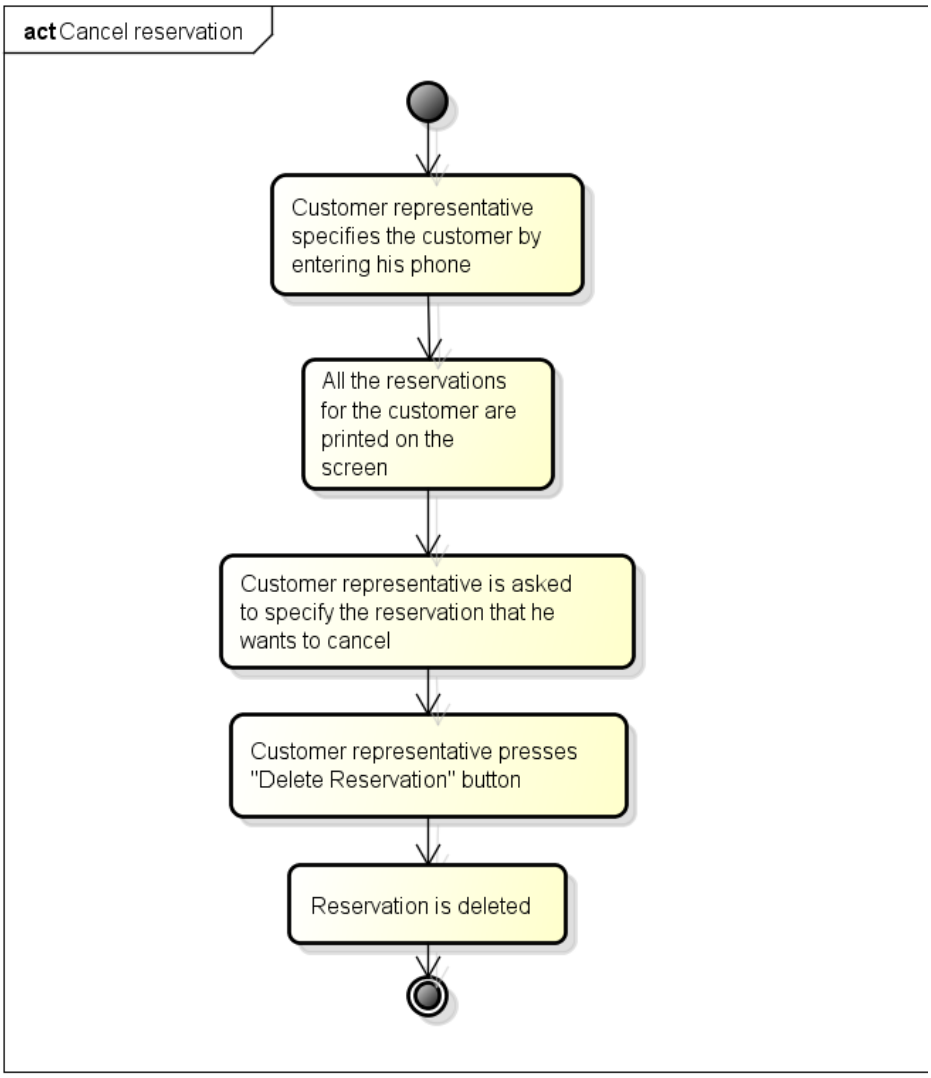
10. Appendices

10.1 Appendix 1 – Activity Diagram

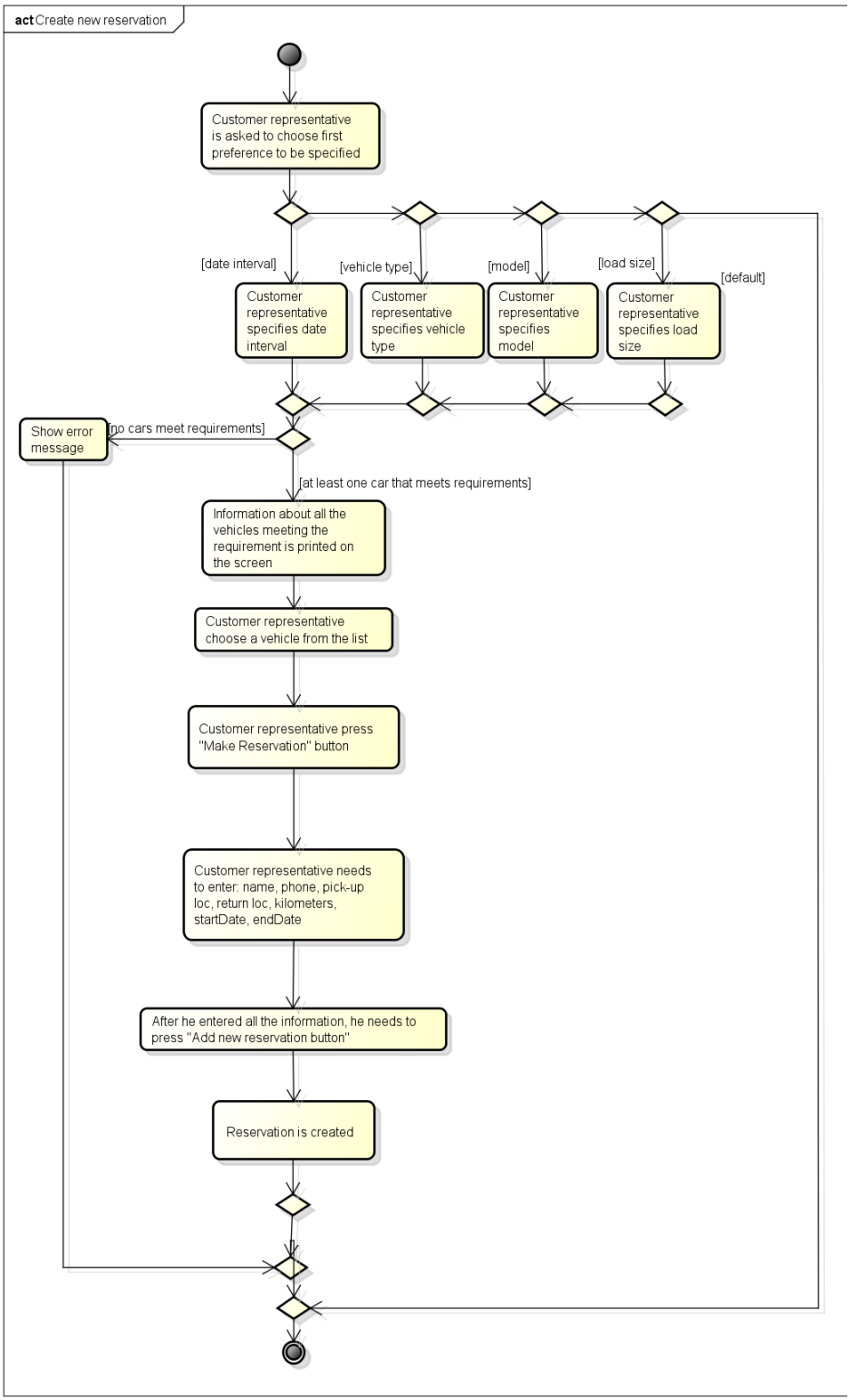
Add new car:



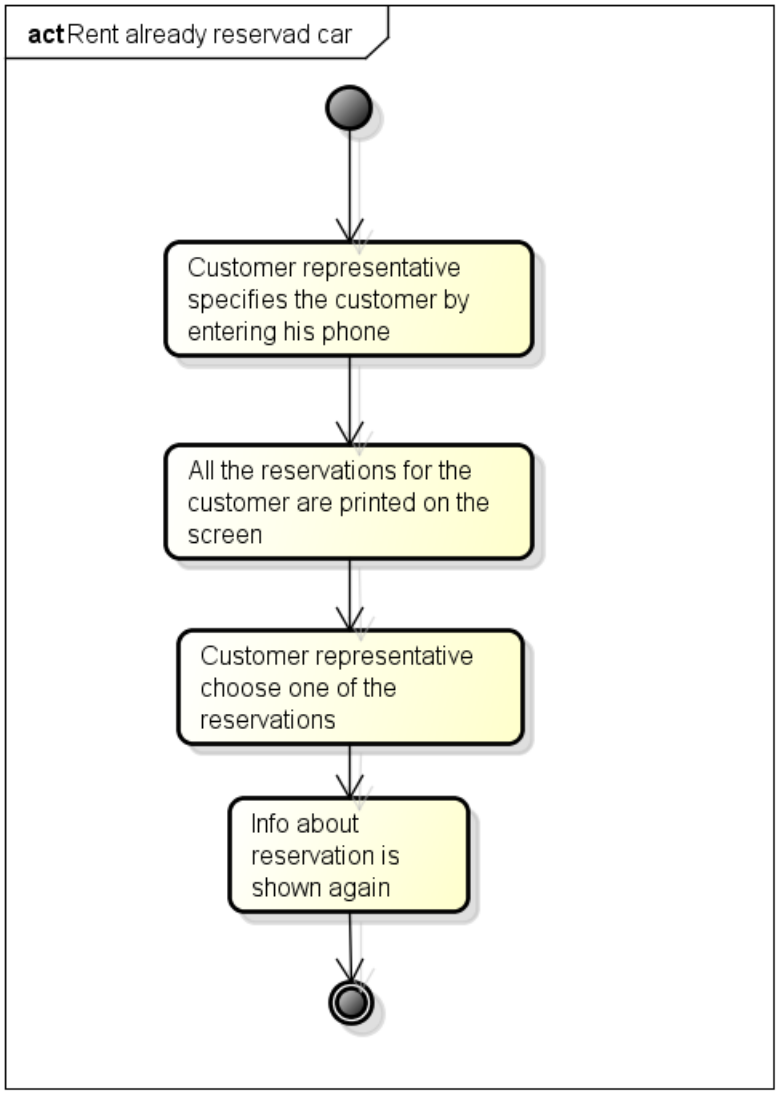
Cancel reservation:



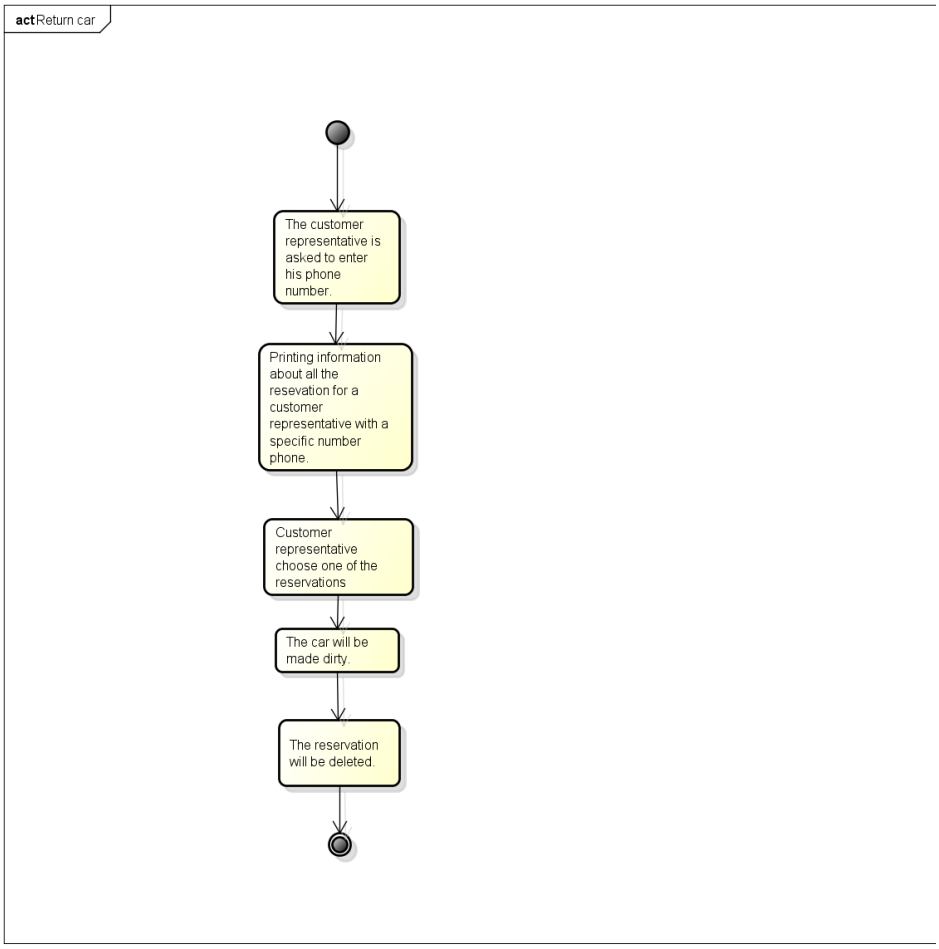
Create new reservation:



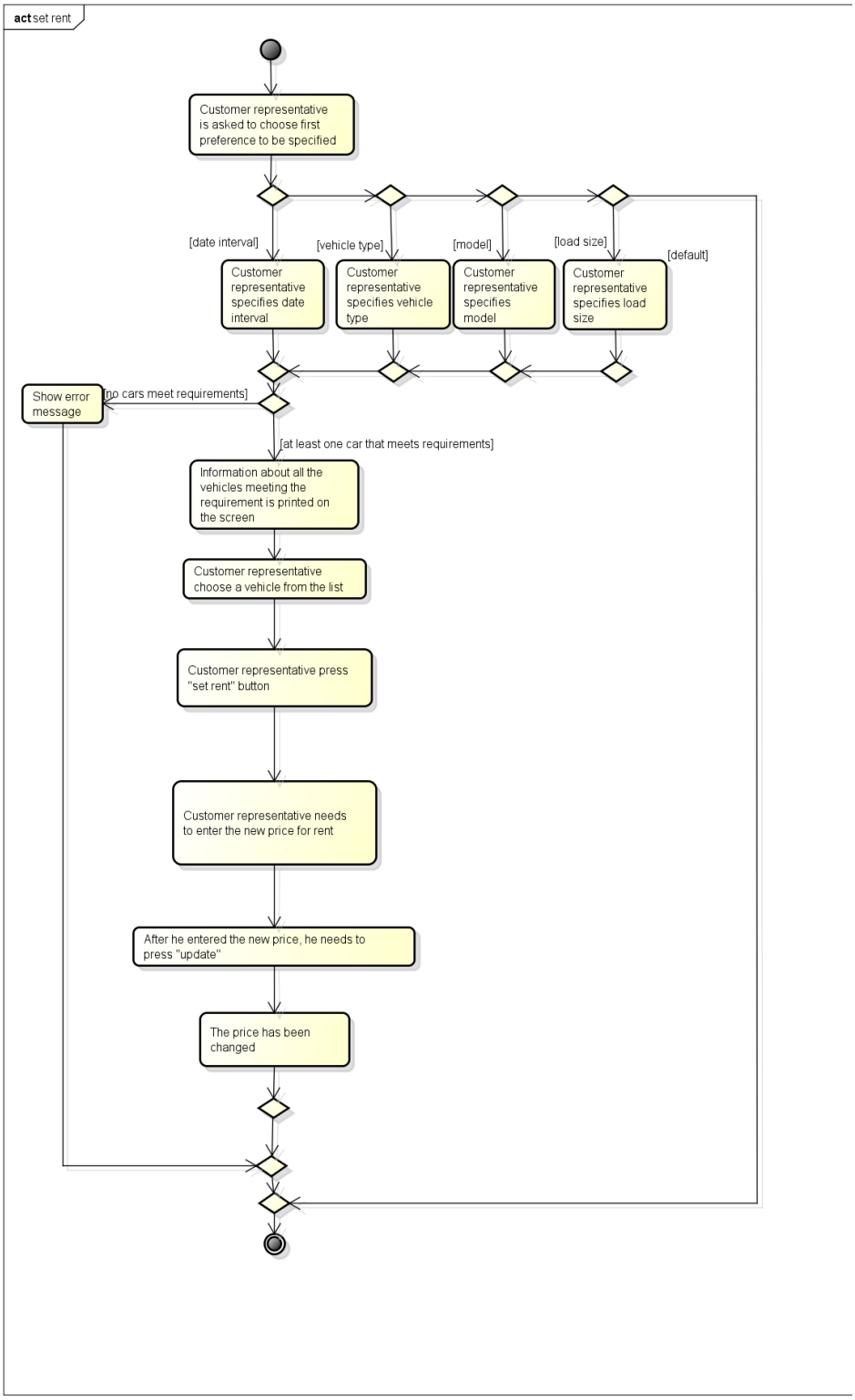
Rent already reserved car:


actRent already reservad car
Customer representative specifies the customer by entering his phone
All the reservations for the customer are printed on the screen
Customer representative choose one of the reservations
Info about reservation is shown again

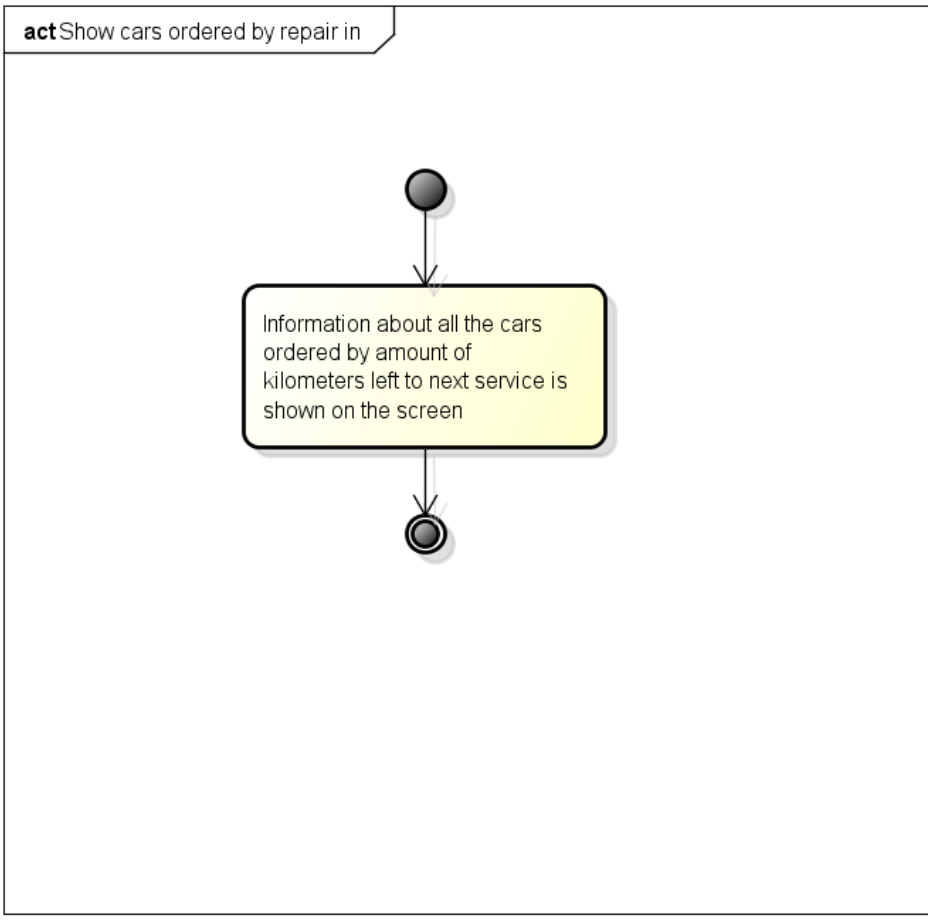
Return car:


actReturn car
The customer representative is asked to enter his phone number.
Printing information about all the resevation for a customer representative with a specific number phone.
Customer representative choose one of the reservations
The car will be made dirty.
The reservation will be deleted.

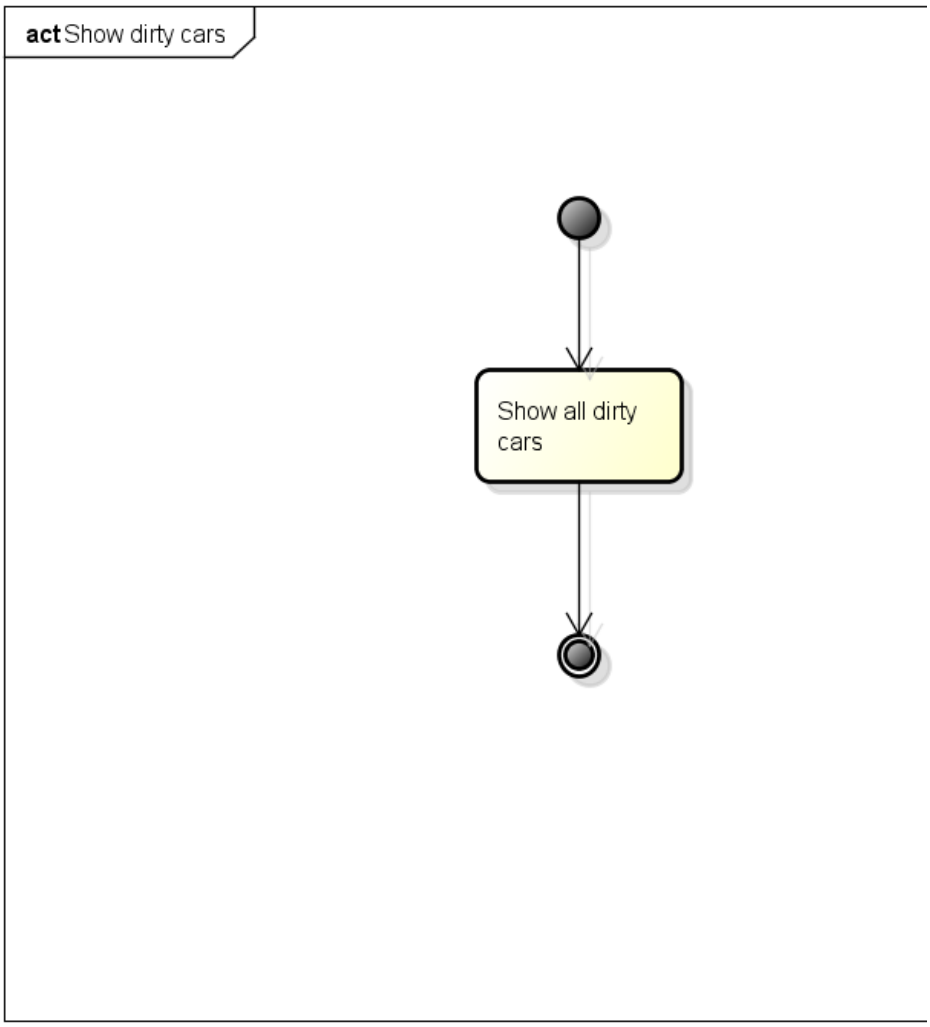
Set rent:



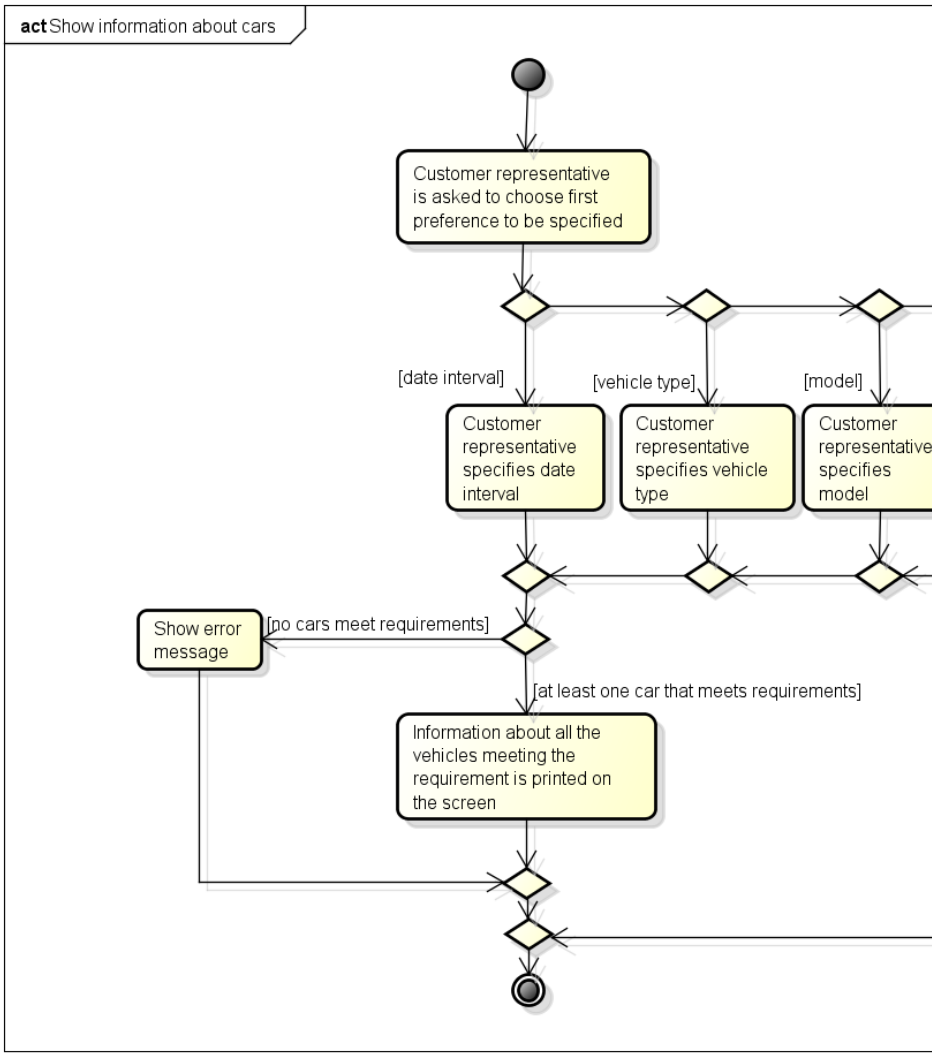
Show cars ordered by repair in:



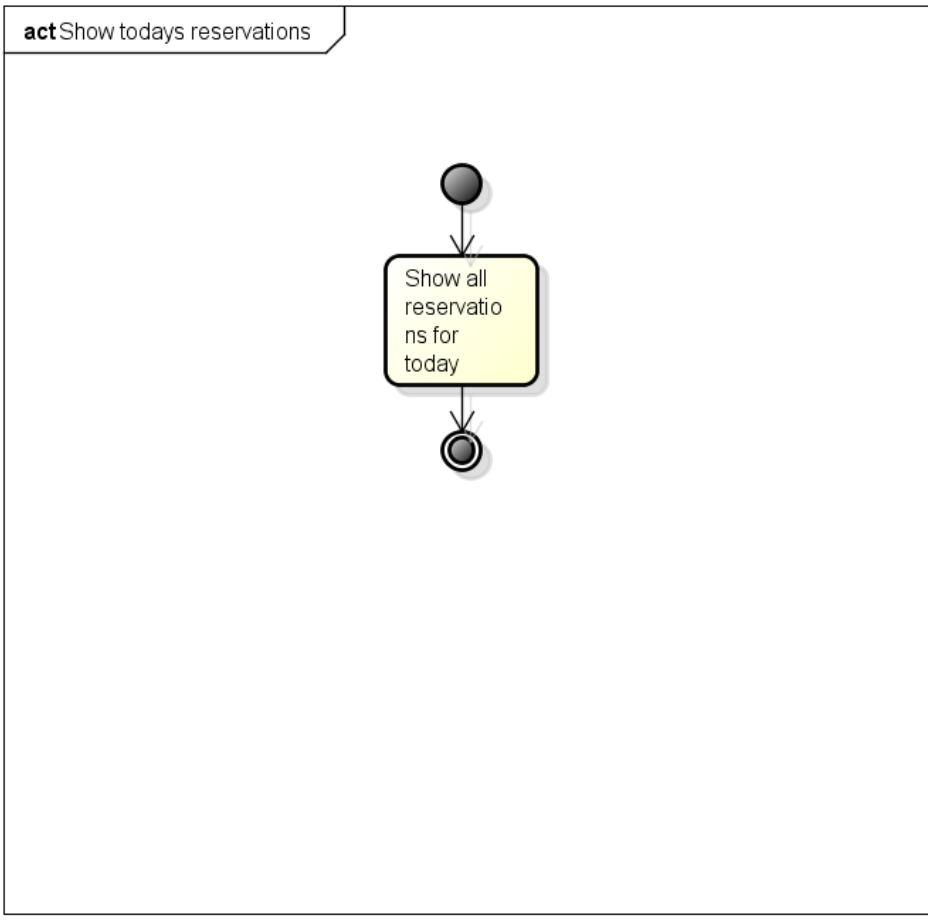
Show dirty cars:



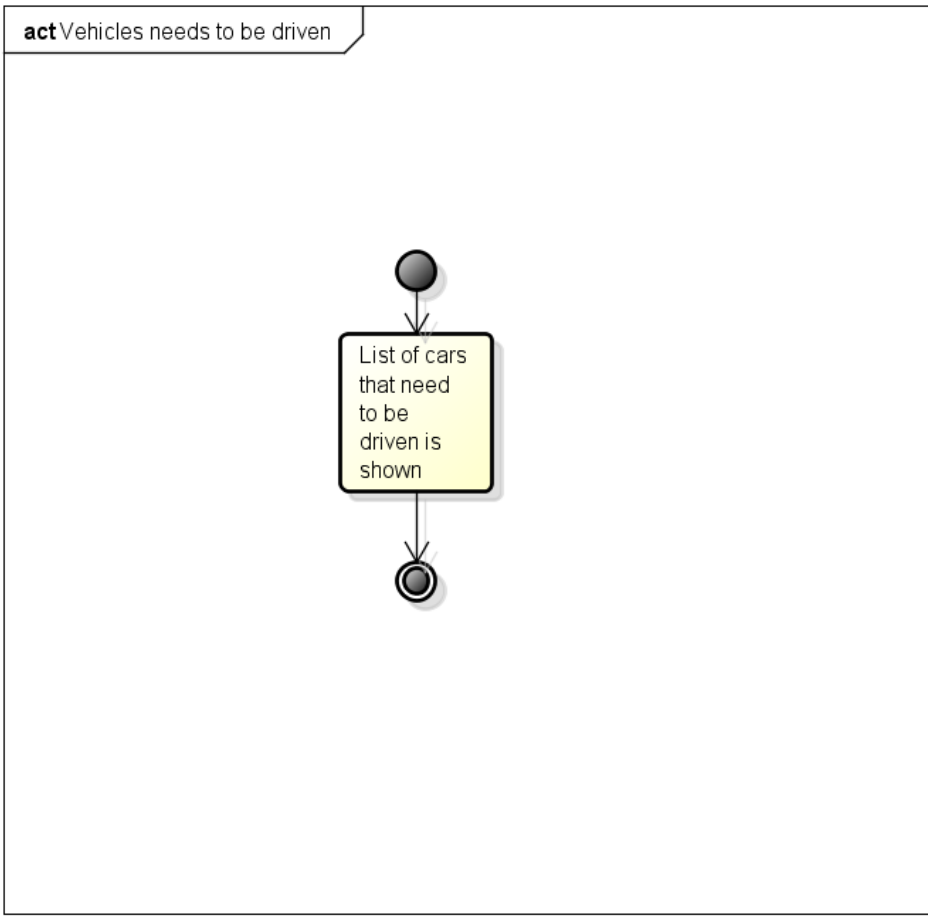
Show information about cars:



Show todays reservations:



Vehicles needs to be driven:



10.2 Appendix 2 – Use Case Description

Add new car:

ITEM	VALUE
UseCase	Add new car
Summary	
Actor	Manager
Precondition	
Postcondition	A new vehicle is created and added to th
Base Sequence	1. Manager is asked to enter information vehicle: type, company, model, year, mil price for day and load size for vans and number. 2. The vehicle is created and added to d
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

Cancel reservation:

ITEM	VALUE
UseCase	Cancel reservation
Summary	
Actor	Customer representative
Precondition	
Postcondition	
Base Sequence	1. Customer representative will choose a entering his phone number. 2. List of all reservations by the custo on the screen and customer representativ choose one of them. 3. Specified reservation is deleted
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

Create new reservation:

ITEM	VALUE
UseCase	Create new reservation
Summary	
Actor	Customer representative
Precondition	
Postcondition	New reservation is created
Base Sequence	1. Customer representative is asked to c preference to be specified. 2. Customer representative specifies the information about all the vehicles meeti requirement will be printed on the scree 3. Customer representative choose one ca list. 4. Customer representative is asked to s name/phone of existing customer or he ne new customer,than he will specify pick-u return location, kilometers, start date, 5. Reservation is created
Branch Sequence	
Exception Sequence	1. Customer representative failed to cho to specify (enters invalid number) - end without creating reservation 2. No vehicles meets requirement - end u creating reservation 4. The vehicle is not available at speci customer representative needs to enter d
Sub UseCase	
Note	

Set discount and price per km:

ITEM	VALUE
UseCase	set discount and price per km
Summary	
Actor	Manager
Precondition	
Postcondition	
Base Sequence	1. Manager needs to press the "Change the price button" 2. Manager is asked to enter after how many days the discount applies, the discount and price per km.
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

Rent already reserved car:

ITEM	VALUE
UseCase	Rent already reserved car
Summary	
Actor	Customer representative
Precondition	The customer has a reservation
Postcondition	Reservation's status is changed to rented.
Base Sequence	1. Customer representative will choose a customer by entering his phone number. 2. List of all reservations by the customer is printed on the screen and customer representative is asked to choose one of them.
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

Return car:

ITEM	VALUE
UseCase	Return car
Summary	
Actor	Customer representative
Precondition	
Postcondition	
Base Sequence	1. Customer representative will choose a entering his phone number. 2. List of all reservations by the custo on the screen and customer representativ choose one of them. 3. The car is made dirty, and reservatio
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

Set rent:

ITEM	VALUE
UseCase	set rent for a vehicle
Summary	
Actor	Manager
Precondition	
Postcondition	
Base Sequence	1. The manager needs to press "show cars 2. The manager needs to specify the car model or date interval. 3. After he enter his preference, he ne the car that meets the specified require 4. He needs to press "set rent" button, new rent per day of that car. 5. After he enters the new price he need "update" button
Branch Sequence	
Exception Sequence	
Sub UseCase	UseCase2
Note	

Show cars ordered by repair in:

ITEM	VALUE
UseCase	Show cars ordered by repair in
Summary	
Actor	Manager
Precondition	
Postcondition	
Base Sequence	1. Information about all the cars ordere kilometers left to next service is shown
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

Show dirty cars:

ITEM	VALUE
UseCase	Show dirty cars
Summary	Show all dirty cars
Actor	Handy Man
Precondition	
Postcondition	
Base Sequence	All dirty cars will be shown
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

Show information about cars:

ITEM	VALUE
UseCase	Show information about cars
Summary	
Actor	Customer representative
Precondition	
Postcondition	
Base Sequence	1. Customer representative is asked to c preference to be specified. 2. Customer representative specifies the information about all the vehicles meeti requirement will be printed on the scree
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

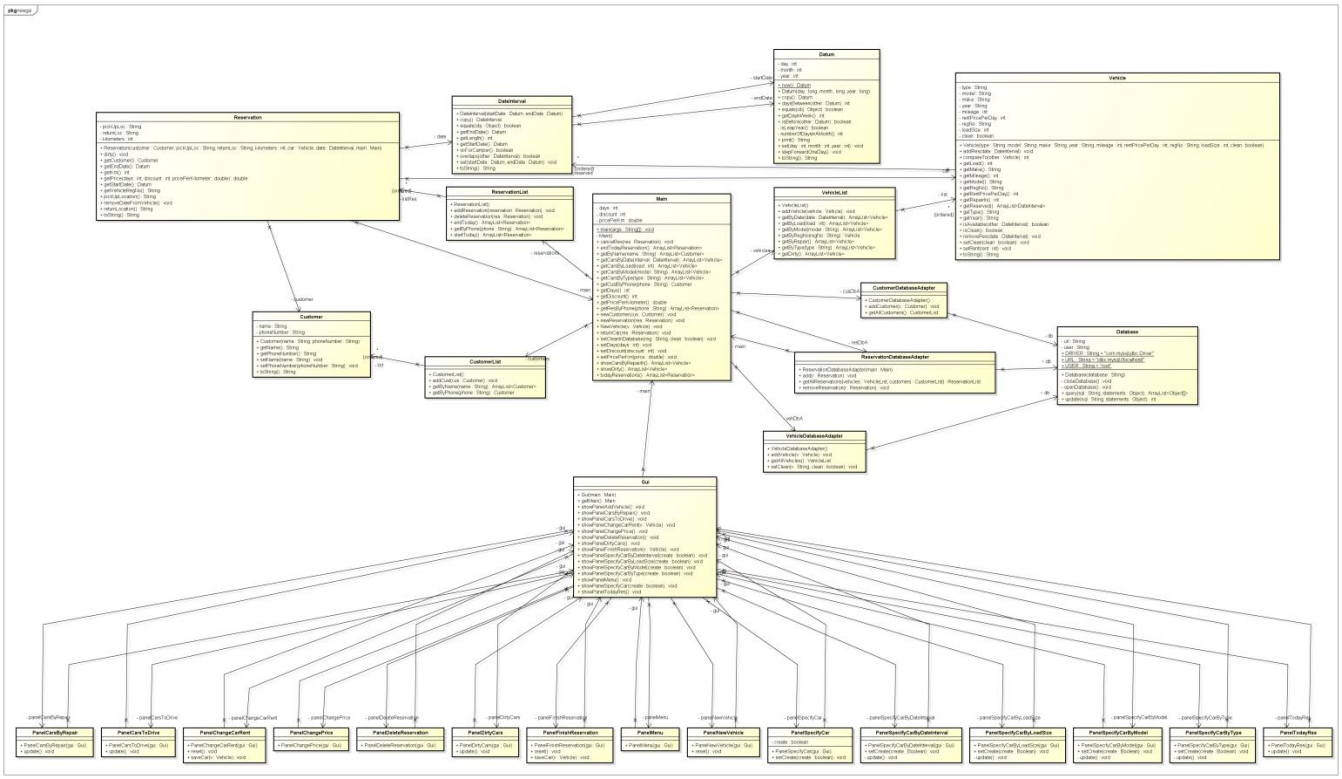
Show todays reservations:

ITEM	VALUE
UseCase	Show todays reservations
Summary	
Actor	Handy Man Manager Customer representative
Precondition	
Postcondition	
Base Sequence	Show all reservations for today
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

Vehicles needs to be driven:

ITEM	VALUE
UseCase	vehicles needs to be driven
Summary	
Actor	Handy Man
Precondition	
Postcondition	
Base Sequence	1. The user needs to press "Show cars to 2. A list of cars that need to be driven somewhere
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

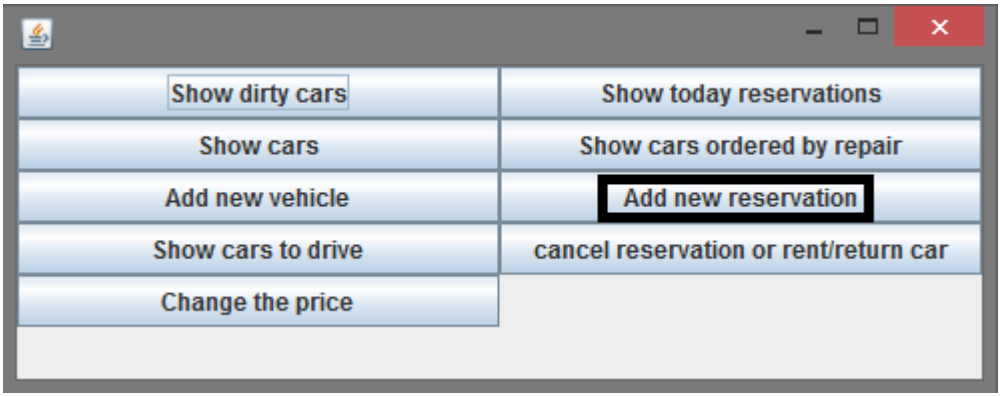
10.3 Appendix 3 – Class Diagram



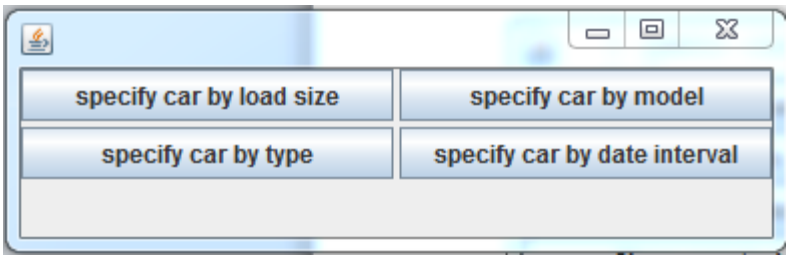
10.4 Appendix 4 – User Guide

Create new reservation

To create a new reservation in main menu of the program click “Add new reservation” button.

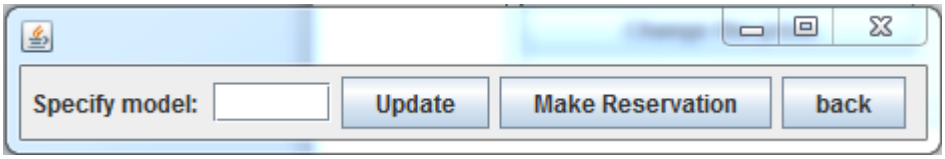


In new frame specify how you want to choose a car.



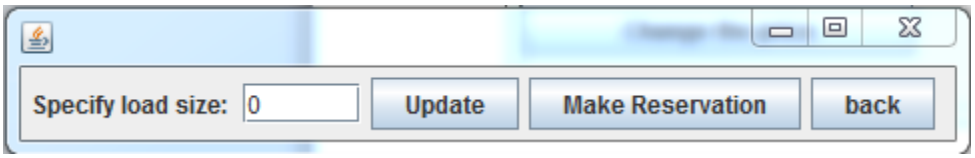
Specify a car byway you chose before

1

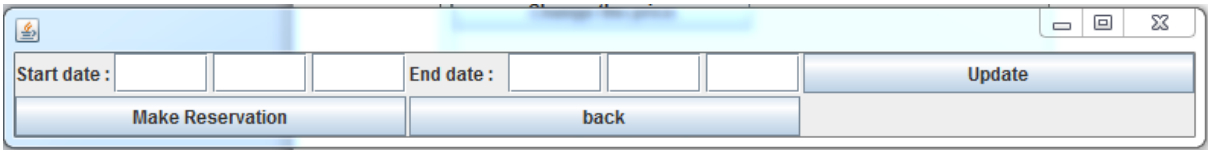


2

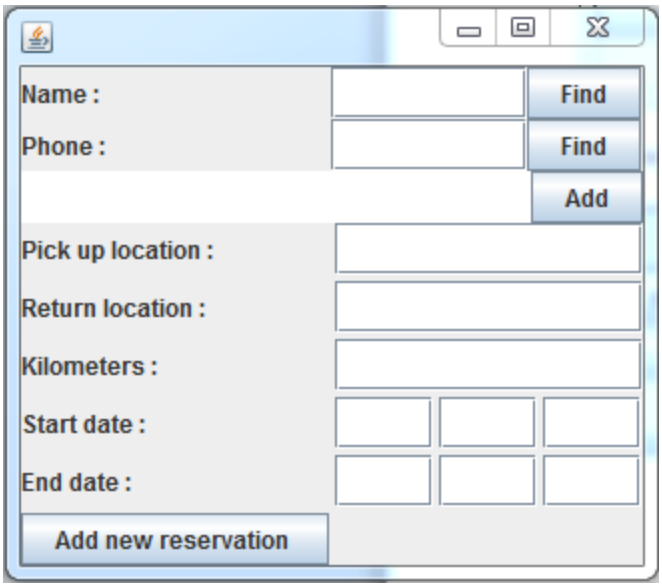
3



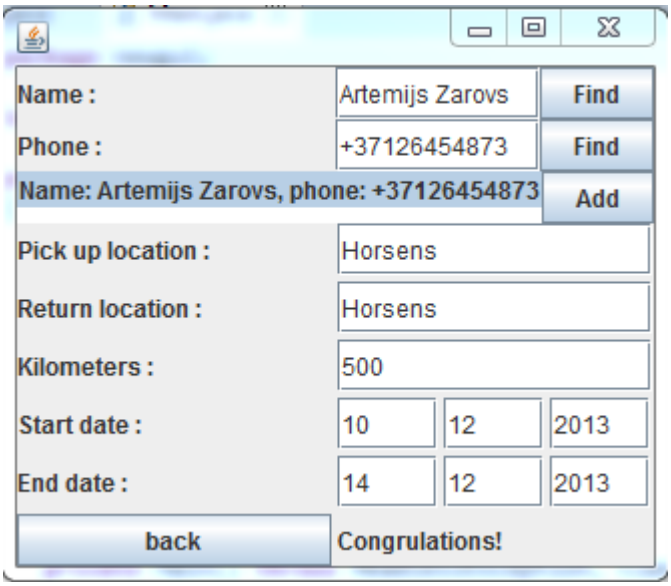
4



The way to choose a car for reservation. After that new window will appear. There fill all asked information about reservation.



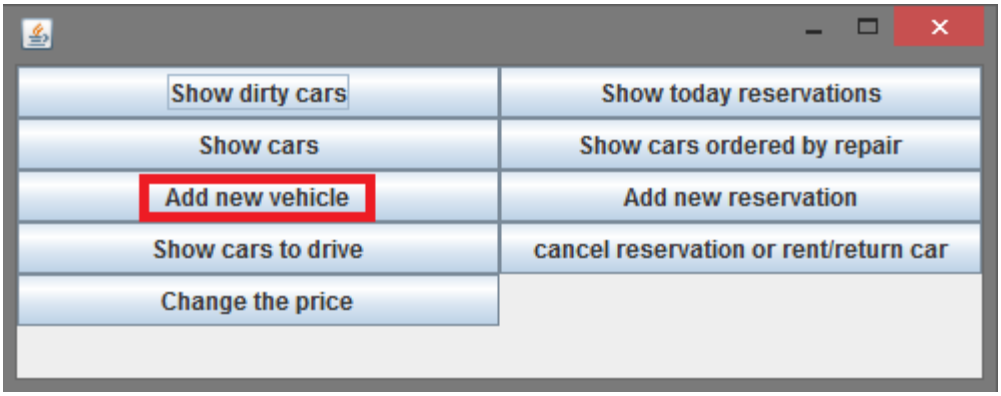
First choose a customer from the list. Customer can be found by looking for his name or phone number. To create a new customer, just fill in both fields and click Add button.



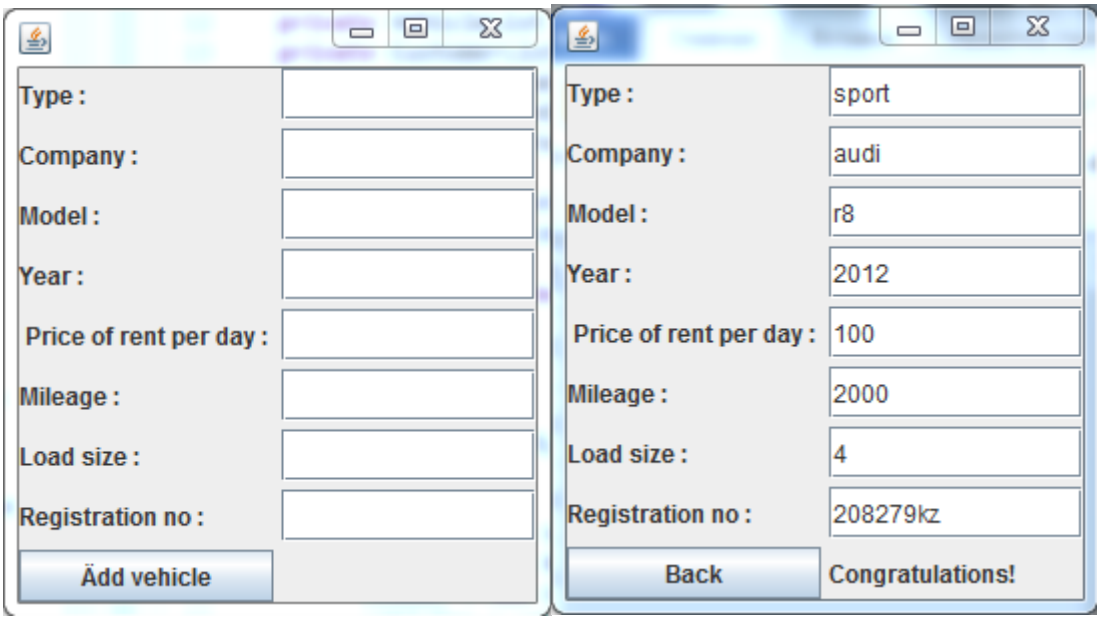
After filling all the information click on “Add new reservation” button and reservation will be created and saved to database.

Adding new vehicle

To add new vehicle please click on “Add new vehicle” in main menu of the program.



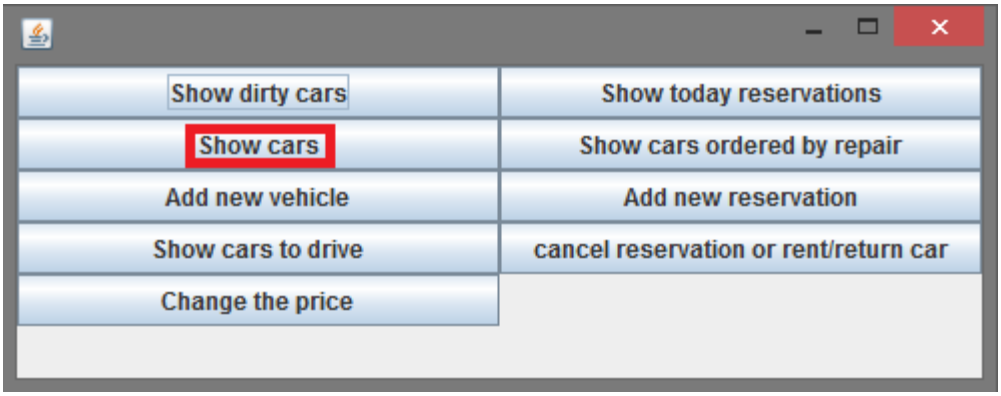
In new window enter all the asked information:



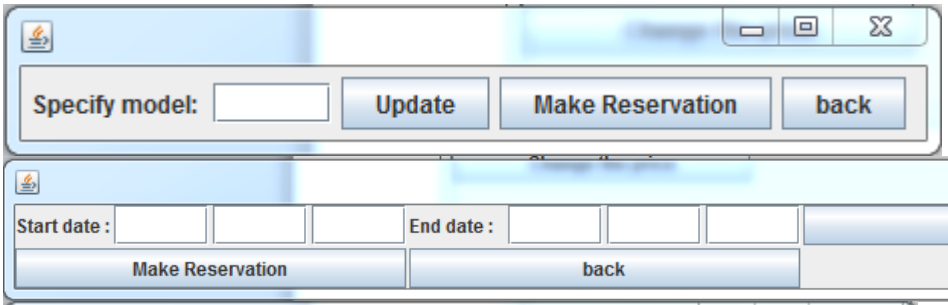
After filling all the fields click on “Add vehicle” button and vehicle will be saved to database.

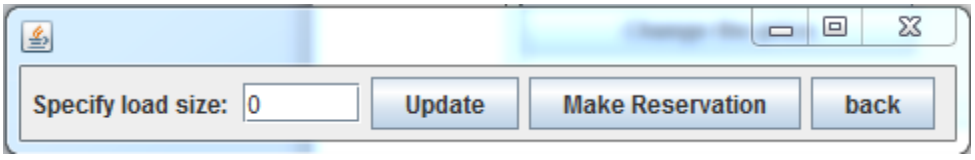
Show a car

This function lists cars specified by user to list cars in main menu click on “Show a car”.



In new window choose how you want to specify a car.

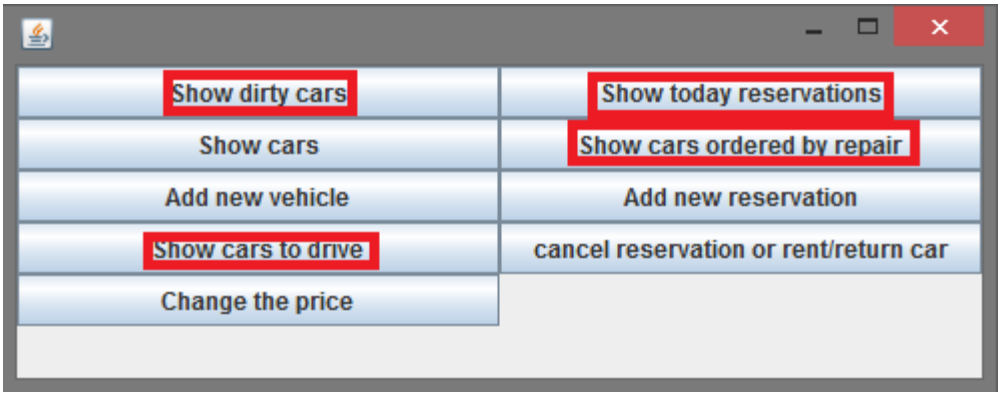




After you specify a car by method you chose a list of cars will appear

Show dirty cars / show cars to drive / Show today reservations / Show cars ordered by repair

These functions work very similar. Simply choose option in main menu.

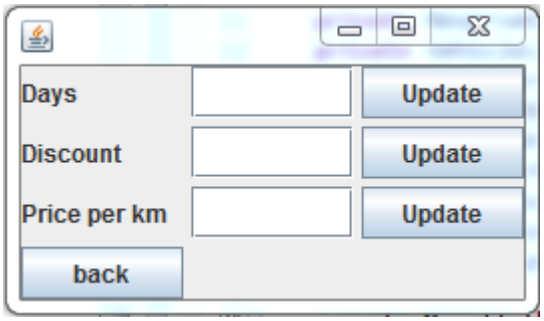


After choosing option program will list you asked information.

Change the price

Click button “Change the price”

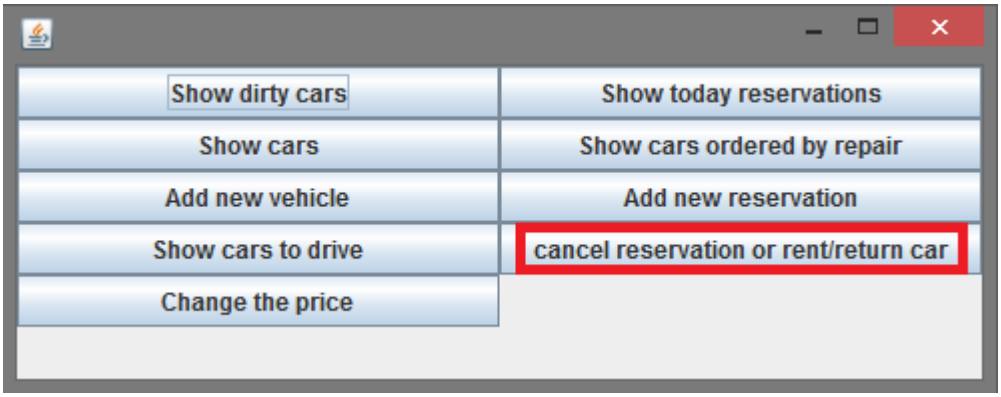
And then:



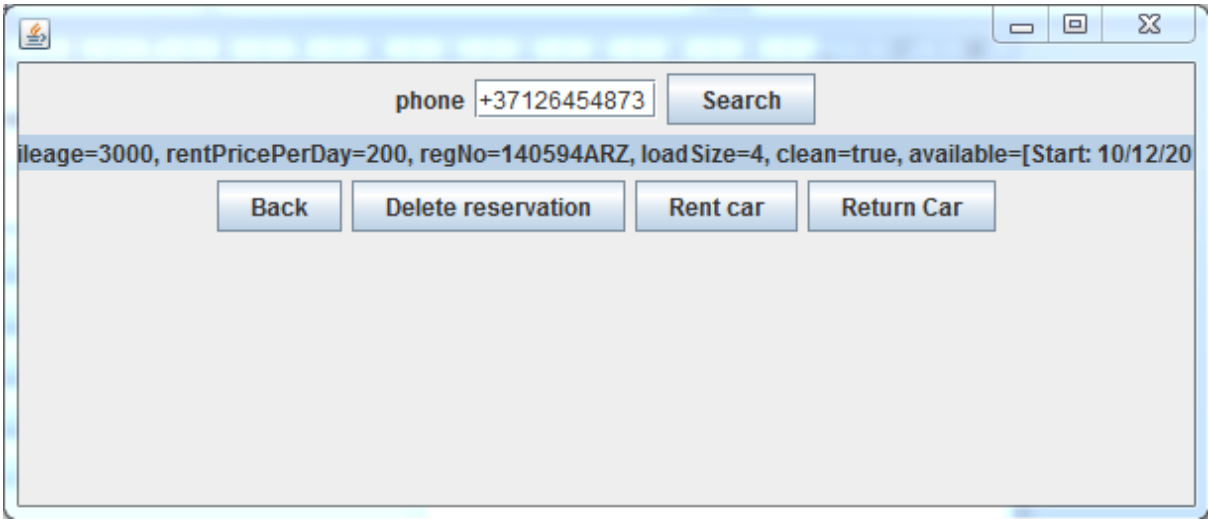
In new window specify number of days, discount and price per kilometer.

Cancel reservation or rent/return car

Click on do "**Cancel reservation or rent/return car**" in main menu.



Then specify the reservation by entering customer's phone number



Now choose action for this reservation do you want to rent a car or customer returned it. Or you can delete reservation.