

University College of Northern Denmark   
Computer Science AP Degree

Gabrielius Burba, Dora Grgurev, Patryk Iciak, Tristan Schalk

2nd Semester Project

RI-CARS Fleet Management System

University College of Northern Denmark

Computer Sciences Academy Profession Degree Programme

**Class:**DMAJ0918 – Project group 3

**Title:**2nd Semester Project RI-CARS Fleet Management System

## Abstract

The paper evaluates the planning, creation and implementation of car tracking system and drivers (employees) management software. The client company is displeased of the lack of a tool which could make processes more efficient. The presented solution includes car tracking system and employee management based on database for data storage. The text describes, in detail, the process of outlining to creating the software in terms of business planning, system development and programming. The development method used was the unified process, an iterative and use-case based approach. The results of these are a more efficient and agile working environment. This software is a good framework for future upgrades.

**Project participants:**Gabrielius BurbaDora Grgurev Patryk IciakTristan Schalk

**Supervisor:**Henrik Jens Vollesen

**External Collaborator:**Ryszard Iciak(owner of RI-CARS Ryszard Iciak)

**Submission date:**04-06-2018

**Revision date:**24-06-2018

**Number of characters:**43359

Contents

[Abstract 2](file:///C:\Users\Administrator\Desktop\End%20Of%20First%20Year\Actual%20report%20-%20Group%203.docx#_Toc10451968)

[Table of Figures 5](#_Toc10451969)

[Introduction 7](#_Toc10451970)

[Subject 7](#_Toc10451971)

[Problem Area 7](#_Toc10451972)

[Problem Statement 8](#_Toc10451973)

[Method 8](#_Toc10451974)

[Organisational Form, Structure and Culture 9](#_Toc10451975)

[Form 9](#_Toc10451976)

[Structure 9](#_Toc10451977)

[Culture 10](#_Toc10451978)

[Mission & Vision, and Strategy 11](#_Toc10451979)

[Mission 11](#_Toc10451980)

[Vision 11](#_Toc10451981)

[Business Strategy 11](#_Toc10451982)

[Stakeholder Analysis 12](#_Toc10451983)

[SWOT Analysis 14](#_Toc10451984)

[Porter’s 5 forces 16](#_Toc10451985)

[Business case 17](#_Toc10451986)

[Summary 17](#_Toc10451987)

[Available options 17](#_Toc10451988)

[Cost-Benefit Analysis 18](#_Toc10451989)

[Impacts and risks 18](#_Toc10451990)

[Conclusion and recommendation 19](#_Toc10451991)

[Methodology 20](#_Toc10451992)

[Decision 20](#_Toc10451993)

[Activity Diagram 21](#_Toc10451994)

[System sequence diagrams 23](#_Toc10451995)

[Domain Model 24](#_Toc10451996)

[Relational model 25](#_Toc10451997)

[Interaction diagram 26](#_Toc10451998)

[Use case diagram 27](#_Toc10451999)

[Design Class Diagram 28](#_Toc10452000)

[Fully dressed use case 30](#_Toc10452001)

[Test Cases 32](#_Toc10452002)

[Programming 34](#_Toc10452003)

[Database & SQL Scripts 34](#_Toc10452004)

[Coding 36](#_Toc10452005)

[Conclusion 48](#_Toc10452006)

[SVN Repository 50](#_Toc10452007)

[Group Contract 51](#_Toc10452008)

[Reference List 52](#_Toc10452009)

## Table of Figures

[Model 1. Create Driver Activity Diagram 21](#_Toc10449915)

[Model 2. Update Driver Activity Diagram 22](#_Toc10449916)

[Model 3. Relational Model 25](#_Toc10449917)

[Model 4. Design Class Diagram 28](#_Toc10449918)

[Table 1. Stakeholder's interests 13](#_Toc10449919)

[Table 2. Cost-Benefit Analysis 18](#_Toc10449920)

[Table 3. Create Cars Fully Dressed Use Case 33](#_Toc10449921)

[Table 4. Test Case Scenarios 34](#_Toc10449922)

[Figure 1. Organizational structure 9](#_Toc10449923)

[Figure 2. Stakeholder analysis 12](file:///C:\Users\Administrator\Desktop\End%20Of%20First%20Year\Actual%20report%20-%20Group%203.docx#_Toc10449924)

[Figure 3. Relations between stakeholders 12](file:///C:\Users\Administrator\Desktop\End%20Of%20First%20Year\Actual%20report%20-%20Group%203.docx#_Toc10449925)

[Figure 4. SWOT Analysis 14](#_Toc10449926)

[Figure 5. Unified Process 20](file:///C:\Users\Administrator\Desktop\End%20Of%20First%20Year\Actual%20report%20-%20Group%203.docx#_Toc10449927)

[Figure 6. Create Driver SSD 23](file:///C:\Users\Administrator\Desktop\End%20Of%20First%20Year\Actual%20report%20-%20Group%203.docx#_Toc10449928)

[Figure 7. Update Driver SSD 23](file:///C:\Users\Administrator\Desktop\End%20Of%20First%20Year\Actual%20report%20-%20Group%203.docx#_Toc10449929)

[Figure 8. Domain Model 24](#_Toc10449930)

[Figure 9. Interaction Diagram 26](file:///C:\Users\Administrator\Desktop\End%20Of%20First%20Year\Actual%20report%20-%20Group%203.docx#_Toc10449931)

[Figure 10. Use Case Diagram 27](file:///C:\Users\Administrator\Desktop\End%20Of%20First%20Year\Actual%20report%20-%20Group%203.docx#_Toc10449932)

[Figure 11. Create Car Table 30](file:///C:\Users\Administrator\Desktop\End%20Of%20First%20Year\Actual%20report%20-%20Group%203.docx#_Toc10449933)

[Figure 12. Create Driver Table 30](#_Toc10449934)

[Figure 13. Create Earning Table 31](#_Toc10449935)

[Figure 14. Create CarSwap Table 31](#_Toc10449936)

[Code 1. App 36](file:///C:\Users\Administrator\Desktop\End%20Of%20First%20Year\Actual%20report%20-%20Group%203.docx#_Toc10449937)

[Code 2. DbConnection 37](file:///C:\Users\Administrator\Desktop\End%20Of%20First%20Year\Actual%20report%20-%20Group%203.docx#_Toc10449938)

[Code 3. APP #2 38](file:///C:\Users\Administrator\Desktop\End%20Of%20First%20Year\Actual%20report%20-%20Group%203.docx#_Toc10449939)

[Code 4. Fleet Management System Interface 38](file:///C:\Users\Administrator\Desktop\End%20Of%20First%20Year\Actual%20report%20-%20Group%203.docx#_Toc10449940)

[Code 5. Earning GUI 39](file:///C:\Users\Administrator\Desktop\End%20Of%20First%20Year\Actual%20report%20-%20Group%203.docx#_Toc10449941)

[Code 6. drawTable 39](file:///C:\Users\Administrator\Desktop\End%20Of%20First%20Year\Actual%20report%20-%20Group%203.docx#_Toc10449942)

[Code 7. toArray 40](file:///C:\Users\Administrator\Desktop\End%20Of%20First%20Year\Actual%20report%20-%20Group%203.docx#_Toc10449943)

[Code 8. Filters 40](#_Toc10449944)

[Code 9. RowFilter 40](#_Toc10449945)

[Code 10. FleetManagementSystem Filter 41](#_Toc10449946)

[Code 11. clearFiltersButton 41](#_Toc10449947)

[Code 12. exportTable 42](#_Toc10449948)

[Code 13. exportTable Interface 42](#_Toc10449949)

[Code 14. Calendar 43](#_Toc10449950)

[Code 15. JDatePicker 43](#_Toc10449951)

[Code 16. Maven 44](#_Toc10449952)

[Code 17. Test 44](#_Toc10449953)

[Code 18. WageCalculation Interface 45](#_Toc10449954)

[Code 19. taxationType 45](#_Toc10449955)

[Code 20. isPaid 46](#_Toc10449956)

[Code 21. createDriver 46](#_Toc10449957)

[Code 22. earningDaoInterface 46](#_Toc10449958)

[Code 23. earningDaoInterface Implementation 47](#_Toc10449959)

## Introduction

### Subject

The subject for this project is middleman “RI-CARS Ryszard Iciak”, a small sole proprietorship owned by the afore mentioned person, Ryszard Iciak. Sole proprietorship is one of the simplest types of businesses available to anyone and everyone willing to invest their own time and resources into. The concept needs only a few minor necessities in order to become approved to be set in motion. In the easiest way one could also describe it as a single entity being personally responsible for both the establishment’s assets and debts. One of those bold individuals is Ryszard who works as an intermediary between the Uber company, which provides him with orders, and the drivers whom he provides his own cars.

This company uses a ‘*brokerage’* business model. Often described as a “business which connects *sellers and buyers* / *businesses and customers* together” it incorporates many variations and versions of the operations alike (*B2B/B2C/C2C*[[1]](#footnote-1)). As the self-explanatory term *broker* portrays¸ this particular business model also arranges transactions between two (or more) parties in order to receive a commission when the deal is complete.

The young, flourishing business is currently in high demand of a system which could ease all the issues of manual work that the executive deals with on a regular basis. The needed structure of the system should be composed out of *database tables* which would include a compressed synopsis (of managing the working field) in order to make all of the data easily accessible and managable.

### Problem Area

In the modern day and age, business owners face many difficulties whilst managing their field of work. Whether it be a plain transaction or advanced scheduling there will always be unavoidable issues that people cannot always solve without the help of pragmatic gadgets. An every-day action of planning car-swap schedules or calculating the employees’ wages could take up to a few hours or even days to accomplish. Additionally, determining the wage costs due to tax laws in certain countries where the business is run is not an easy task to take on by yourself.

That is the case of Ryszard Iciak, the owner of RI-CARS, who is requesting a brand new, easy-to-use system to assist him in dealing with daily complications. His primary concerns are timetables consisted of his driver’s names, the cars and the amount they have driven them at a particular time period, and car-swapping between the drivers themselves. There is also an important factor affecting the wages in the country that he runs the business[[2]](#footnote-2) in, and that is [*the taxation itself*](https://en.wikipedia.org/wiki/Taxation_in_Poland). It is difficult to calculate wages for each driver because the way the taxation works is very different depending on factors such as *if an employee is a student* or *if he/she has a job on the side* etc.

### Problem Statement

* How can we simplify the process of planning, storing, accessing and analyzing the data about cars, drivers and their payments, while considering fast and efficient services?
* How can we implement a business logic which would be able to plan schedules for the company?

### Method

To collect the data/information about the problem - we interviewed the owner of RI-CARS. He explained the daily duties in the company along with the issues he was facing. After the interview, we deduced the problematic features that we will need to work on as well as what tools and theories we should use in order to achieve mutual agreement.

To figure out the solution for the problem statements we will be using Unified Process, Eclipse for coding, MS-SQL for the databases, SVN as version control system and UML for specifying, visualizing, constructing, and documenting the artifacts of software systems.

Organizational Form, Structure and Culture  
  
In this section we will examine the organizational form, structure and culture of RI-CARS Ryszard Iciak.

The small firm contains characteristics of an independent, uncomplicated, easy-to-set-up business with a nominal charge, which leads to the verdict that all those attributes are main components of a sole proprietorship.

### Form

As being a small and independent company, RI-CARS qualifies for being a sole proprietorship. In the business, owners can establish a sole proprietorship instantly, easily and inexpensively. Sole proprietorships carry little, if any, ongoing formalities. With it being such a business, it may come with some disadvantages - owners are subject to unlimited personal liability for the debts, losses and liabilities of the business, owners cannot raise capital by selling an interest in the business, sole proprietorships rarely survive the death or incapacity of their owners and so do not retain value.

### Structure

The organizational structure is an important part of any kind of business; from small one-man businesses to corporations as big as McDonald’s and the like. The term describes the action of planning and performing tasks which were previously defined in coherence with a person’s business partner/s. The organizational structure for RI-CARS is being kept smooth and straightforward:

* The owner of the company manages everything company related, including drivers, wages, car maintenance
* The drivers, who are the only other members of the business, have access to the clients
* Clients themselves are able to access the company’s services through the official Uber app

#### 

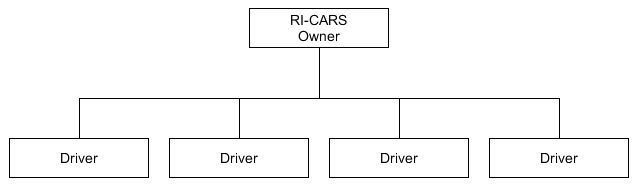


Figure 1. Organizational structure

#### Managing the growth of the company

As the lean entrepreneurial organization grows, it might change into one of the other types of organizations as it responds to an increased need to efficiently manage complexity and segmentation of operations. As an entrepreneurial organization grows, swift decision-making might become a challenge for the owner-operator or small management team. Here, external demands might overwhelm rapidly growing business processes and procedures that do not have enough management oversight.

### Culture

This section will briefly go into the organizational culture. Considering that the Uber company, the main contractor of RI-CARS, is well-known all over the world, many of its customers come from various countries. For Uber drivers in Poland it is very common to meet customers from all around the world especially the Scandinavian region and Germany. It is required from drivers to be extroverted and open-minded in order to work with a lot of different nationalities and their cultures. Among other things, the Uber company is an international giant which can have other habits and cultures brought into the business.

## Mission & Vision, and Strategy

A company’s values and goals are mirrored in the mission and vision they put out in the world. In RI-CARS Ryszard Iciak’s case, the mission is very simple since the company is a very small entrepreneurship. The subsequent mission and vision were created by the company’s founder and CEO, Ryszard Iciak.

### Mission

Our mission statement is represented by Ryszard Iciak:

*“To provide the most efficient transport services*

*by ensuring to people*

*who are willing to be drivers*

*a high quality, low-maintenance environment. “*

### Vision

Our vision statement is represented by Ryszard Iciak:

*“Our vision is to make it affordable*

*for every customer*

*to use the possibly cheapest transport services.”*

### Business Strategy

Setting the goal for the company is one of the most important aspects of a business. It is the company’s way to set and engrave its development and success.

Undoubtedly, the most important aspect is to keep competitive prices on the Polish market and still stay profitable. To accomplish this, the company tracks changes of the passenger transportation market, keeping in mind other competitors. All of them offer different collaborate conditions, take different commision rates, set different prices of services and different frequency of orders.

Therefore, the company is looking for the solution which would simplify and automate the process of management and thus, evade the unnecessary time consumption needed for the manual tasks.

Finally, in the future, despite of the increasing number of cars and drivers, the company’s expansion is not expected. RI-CARS is a small private business and it is not going to turn into a huge enterprise anytime soon. It’s rather a way of making income for the owner in an unconventional way, therefore, RI-CARS is expected to keep as simple of an organizational structure as possible.

## Stakeholder Analysis

By analyzing stakeholders of the company, we find it to be simplistic, but fulfilling. It does not have many stakeholders because it is a sole proprietorship company, meaning it is owned by one person and managed by them alone.

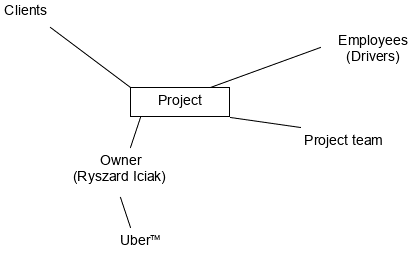


Figure 2. Stakeholder analysis

#### Relations between the stakeholders

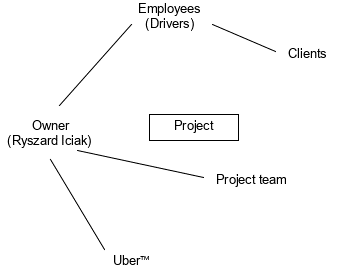
Once again, we can see that the relation is simple. The owner of the company communicates between the project team, the Uber and the employees who then communicate with the customers. Notably, we could say that clients also communicate with Uber, but in fact they only use their application.

Figure 3. Relations between stakeholders

Table 1. Stakeholder's interests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Stakeholder** | **Their goals** | **Past reactions** | **Behavior expected** | **Positive or negative to changes** | **Likely reaction** |
| **Owner** | Manage the company, be a middleman between Uber and their service in the city | Manual labor with managing the various aspects within the company | Improved and faster gathering of information (i.e. employee work hours, car management, wages, etc.) | Positive | Overall positive and better performance in managing the company. |
| **Employees** | Driving the customers to the destinations they provide | With the owner's manual management of the taxes, payments come slower and less accurate wages | - | Positive | More organized and better job schedule, overall better performance in the work |
| **Clients** | Use the provided company’s service | - | Achieve the best experience with the service | Briefly positive | - |
| **Uber™** | Receive the percentage of the customers payment. | - | Unnoticed | Neutral | - |

With this table it is made clear of just how interested the stakeholders would be in the newly built system.  
Obviously the most affected stakeholder would be the owner.

Employees would benefit marginally as well,  
with the most important changes for them being the automatic system which counts wages, taxes and the like, lowering the time wasted (by doing it manually) and improving the accuracy of the action performed.

The clients would, most likely, notice no change at all.

Uber™ also would not notice a change since the money they receive from the service is calculated by the official Uber™ app.

## SWOT Analysis

|  |  |
| --- | --- |
| **Strengths**   * Meeting all requirements of a transport market required by the law * A small fleet can easily be disposed of even in case of failure of the company * Low-maintenance cars of the same model cause even lower expenses * Consent to hire people with less taxed contracts | **Weaknesses**   * Income is closely linked with factors which the company cannot affect * Employees may easily become discouraged on worse periods of time since their earnings depend on turnovers |
| **Opportunities**   * Changes of transport law cut off dishonest competitors off the market * People changing their habits into ordering transport services with phone application | **Threats**   * Conventional taxi subjects trying to affect the government in order to outlaw modern transport companies * A quickly growing number of competitors |

Figure 4. SWOT Analysis

#### *Strengths*

Most of the strengths, which the company has, comes from its straightforward structure. Being easily managable makes it simple to control and allows the owner to make decisions without having serious negative consequences. Having just started the company also makes the cheap decisions reflect better.

#### *Weaknesses*

Negative part about the business is that it is largely affected by the outside factors. With the income being solely based on the number of clients the company gets during the day, it generally influences the drivers aswell. During worse periods it might leave drivers with little to no work and it might affect their view towards their work which would leave them discouraged. This would also affect the performance and the satisfaction of the customers.

#### *Opportunities*

Great opportunities await the business if they keep up with satisfying their customers. As technology keeps thriving, more people tend to use more convenient means of transport and ordering a ride by an app is one of the easiest and cheapest ways to do so. It would also cut off a lot of other services that rely on phone lines to get customers, and if they do not adapt to it, they would be out of the competition.

#### *Threats*

With the constant competition comes many inconveniences. Evergrowing Uber service creates more rivalry towards RI-CARS. Moreover, the modern transport companies have become an opponent for conventional organisations which try to eliminate new comptetitors. Main way they attempt to do so is by convincing the government to make laws against modern transport solutions.

## Porter’s 5 forces

#### Competition in the industry

The passenger transportation market is very crowded. There is a lot of competitors with same or different business model. Uber has hundreds of partners such as RI-CARS, which apparently are comptetitors for each other. Moreover, there is plenty of conventional taxi companies which do not want to give up the market and act offensively trying to affect the government to delegalize companies whose drivers do not meet all requirements of taxi industry. Also there are other means of transport which a potential customer may choose instead of ordering own ride. Public transport (busses, trains, rentals) is also competitive for RI-CARS and similar companies.

#### Potential of new entrants into an industry

It is easily accessible to enter the market of passenger transportation. Every person who has a car can start a business and become whether uber partner or other kind of active participant on this market. Nowadays it is also easy to retrieve a car even if you do not have cash for purchasing it – loans are very cheap and banks are willing to lease cars for new businesses.

#### Power of suppliers

Uber is the main supplier of services for RI-CARS. By using Ubers platform RI-CARS company gains much needed exposure for the business to grow. On that contractor relies almost the entire business, since they are a major provider of orders. The other notable suppliers are car services such as car repair shops, gas stations and car dealers. These are also important significant contractors but they are easily replaceable, therefore any specific one of them is essential for RI-CARS.

#### Power of customers

With the current age of communication systems and technologies comes great responsibilty. By easily accessing the internet you get lots of choices regarding services you want to take or use. For RI-CARS it would mean a great deal but would also makes the company more perceived and critiqued. If, by any chance, the customer is happy or dissatissfied with the cost or quality of the service, they are free to leave a brief review about the driver in the official Uber app. The lower rating it reaches, the fewer customers would want to use the service, making the company bankrupt. Also all customers are allowed to report faulty service to the customer office.

#### Threat of substitute products

Customers expect to receive cheap and efficient services. In case they are not satisfied with the service, they can easily change to other means of transport. Examples of possible substitutes are busses, local train connections, car rentals, bikes (last both are getting more popular in rentals per minute) or even walking instead of using any kind of vehicle. Alternatives are easily available, and therefore it is so important to keep customers satisfied.

## Business case

This scope contains the business case of RI-CARS Ryszard Iciak for which the implementation of a new IT-system is needed. The section defines problems which are required to be taken into consideration. The company does not have software which would automatize recurrent processes such as calculating wages and planning schedules for drivers. Moreover, there is a need of a database which would store information about employees, cars and events.

### Summary

Currently there is no system present in RI-CARS. Every schedule, payment and action is made manually. It takes a long time to perform them sufficiently. Therefore, we were requested to create a new IT-system.

Main necessary features that the system must include are:   
1. It can store data about vehicles, drivers and turnover made by them.   
2. It can calculate taxes and payments for drivers relying on their turnover.  
3. It can track car swaps and plan them with taking location into account.

If the implementation of the system was successful, RI-CARS would have more efficient environment and would be more agile when handling daily situations.

### Available options

There are a few available options that are considered:

**1. Making a simple, extendable system with basic implementations**

This option assumes creation the basic business logic including database accessing, automatized calculation of wages and planning car swaps. It requires a user to enter all information manually. The created software would be easily extendable.

**2. Implementing the necessary features using Uber API and Google API.**

This option assumes implementing both API shared by the Uber company and API shared by the Google company. With these, the user would not be obligated to enter all the data manually. The data of drivers and their earnings would be synchronized with Uber database and the Google API would be used for assessing distances between drivers in order to plan efficient schedules.

**3. Creating a basic IT-software and implementing foreign APIs if there is time left.**

It assumes creating a software according to the plan from the 1st point and implementing features mentioned in the 2nd point only if programmers can implement them within the deadline. It is the most agile approach but does not specify the exact result of the development process.

#### Choice

We have decided to choose the option **number 3**. It is the most flexible option which will allow us to focus on features which we decide to be the most relevant. In case we consider any implementation too complex, it allows us to resign on it and focus our effort on more significant parts of the project.

### Cost-Benefit Analysis

### 

Before committing the project, the assessment of profitability should be undertaken. Since RI-CARS does not pay for the software, which is about to be created, and the software does not affect costs (referred as money) of the company, the simple table comparing not nominal costs and benefits is sufficient.

Table 2. Cost-Benefit Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| **Cost** | | **Benefit** | |
| *Tangible* | *Intangible* | *Tangible* | *Intangible* |
| Software maintenance (Money) | Need to change habits (Mental) | Less need of relying on human factor, saving of time which would be spent on performing recurrent duties (Time) | Satisfaction of well-managed business which is better prepared for a possible growth (Mental) |
|  |  |  | Easier overview of all data collected (Efficiency) |

Since the future application is planned to be a desktop application, there are no significant costs which can affect the decision of undertaking the project. The only cost of the owner of the company is changing their habits into using a new software which would improve recurrent processes. Benefits of new software solution include becoming independent from the human factor which might appear unreliable, easier access to all data gathered in an application and, as a result, better preparation for a possible growth.

### Impacts and risks

The expected impact of the project for the company is positive. New IT solution means less time spent on recurrent tasks and it increases the reliability of results. The new system would relieve the owner of calculating wages, planning schedules and managing cars.

The project does not seem to have any risks for RI-CARS. An unsuccessful implementation of created system would not affect company results negatively.

### Conclusion and recommendation

The most relevant way of committing the project is by choosing the **3rd option** which would make the project workflow more agile and efficient. The scale of benefits in comparison to scale of costs is overwhelming. This would be the most flexible way for running the project as it provides less restrictions and allows to carry out various ways of development.

## Methodology

This section of the report focuses on the methodology part. Undoubtedly, one of the most important aspects of the project is choosing the method which you are going to use. It determines the flow and the structure of the project and is very important in creating systems.

### Decision

With various options available on choosing the development method for company RI-CARS, the most suitable one, in this case, appears to be the **Unified Process** methodology.   
  
What is it?

*“Unified Process (UP) - The****Unified Software Development Process****or****Unified Process****is an*[*iterative and incremental*](https://en.wikipedia.org/wiki/Iterative_and_incremental_development)[*software development process*](https://en.wikipedia.org/wiki/Software_development_process)*framework. The phases of the process are divided by iterations. Each iteration results in an increment, which is a release of the system that contains added or improved functionality compared with the previous release.*

*It insists that the architecture sits as the main point of the project’s team’s effort. One of the most important deliverables of the process is the executable architecture baseline which is created during the Elaboration phase.*

*The Unified Process requires the project team to focus on addressing the most critical risks early in the project life cycle. The deliverables of each iteration, especially in the Elaboration phase, must be selected in order to ensure that the greatest risks are addressed first.”*

Figure 5. Unified Process



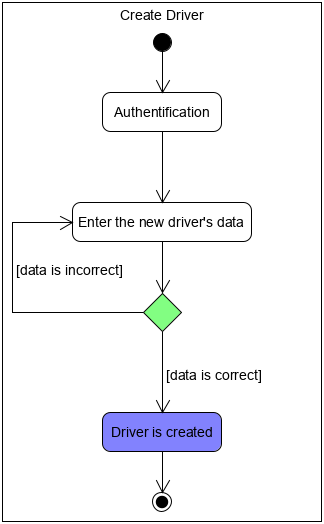
*From Wikipedia,* [*Unified Process*](https://en.wikipedia.org/wiki/Unified_Process)

## Activity Diagram

In this section, the Activity Diagram is described. Create Driver and Update Driver are used as examples of how the system is going to behave when engaged by the owner.

#### Create driver

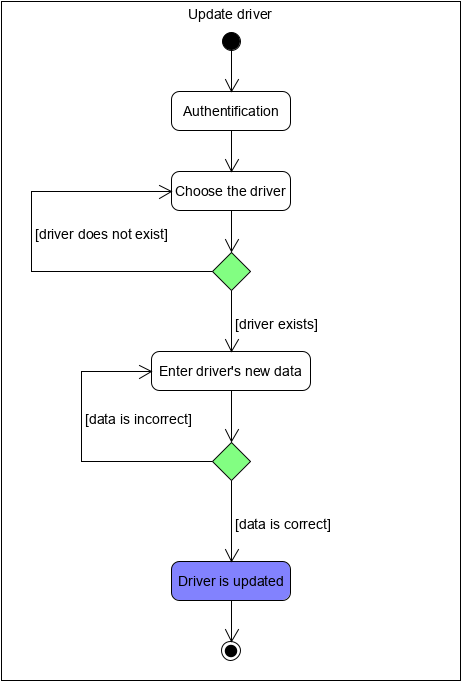
Model 1. Create Driver Activity Diagram



The Create Driver Activity Diagram paints a clear picture of a driver being created. It firstly needs to go through the authentification process of entering data and then checking if it fulfilles all the prerequisites of the wanted field. If the action ends in the result being correct, the Driver will be created. Otherwise the user is granted a chance to try the whole process again.

#### Update driver

Model 2. Update Driver Activity Diagram



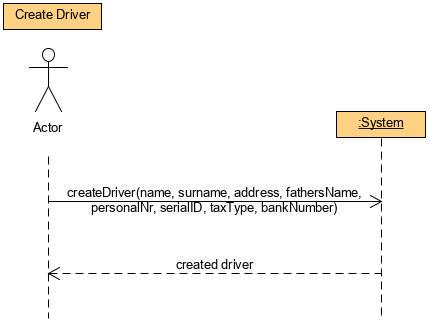
The Update Driver Activity Diagram presents the option of choosing a driver that you would like to update, and if the input is incorrect, the process is repeated until it meets the requirements. Once the driver is found, the user can update the driver’s credentials, and after the successful entry, the system checks for the compatibility and thus the driver’s information is updated.

## System sequence diagrams

System sequence diagrams are a crucial addition to the project. They tell about the way the system works and reacts to the commands and requests send from the client side.

**Create driver:**

Figure 6. Create Driver SSD



Create driver shows how it would work if the user chose the create driver option. It simply requests the user to type in the details specified by the system and if it is successful, the driver is created.

**Update driver:**

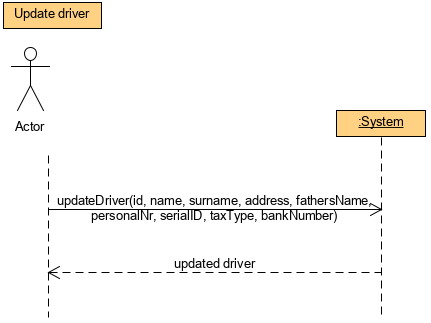


Figure 7. Update Driver SSD

Updating the driver does not differentiate marginally from Create order. The only change is the mandatory ID entering sequence, which indicated what driver should be accessed for the update. If the update is successful, the driver’s information is changed.

## Domain Model

The application is going to serve the following entities: ***Driver***, ***Car***, ***Earning*** and ***CarSwap***. Information about them will be stored in a database and during the application execution the desired ones will be transformed into managable objects with accessible variables. All of objects will be given a unique key. Relations between the entities are presented in the following chart:

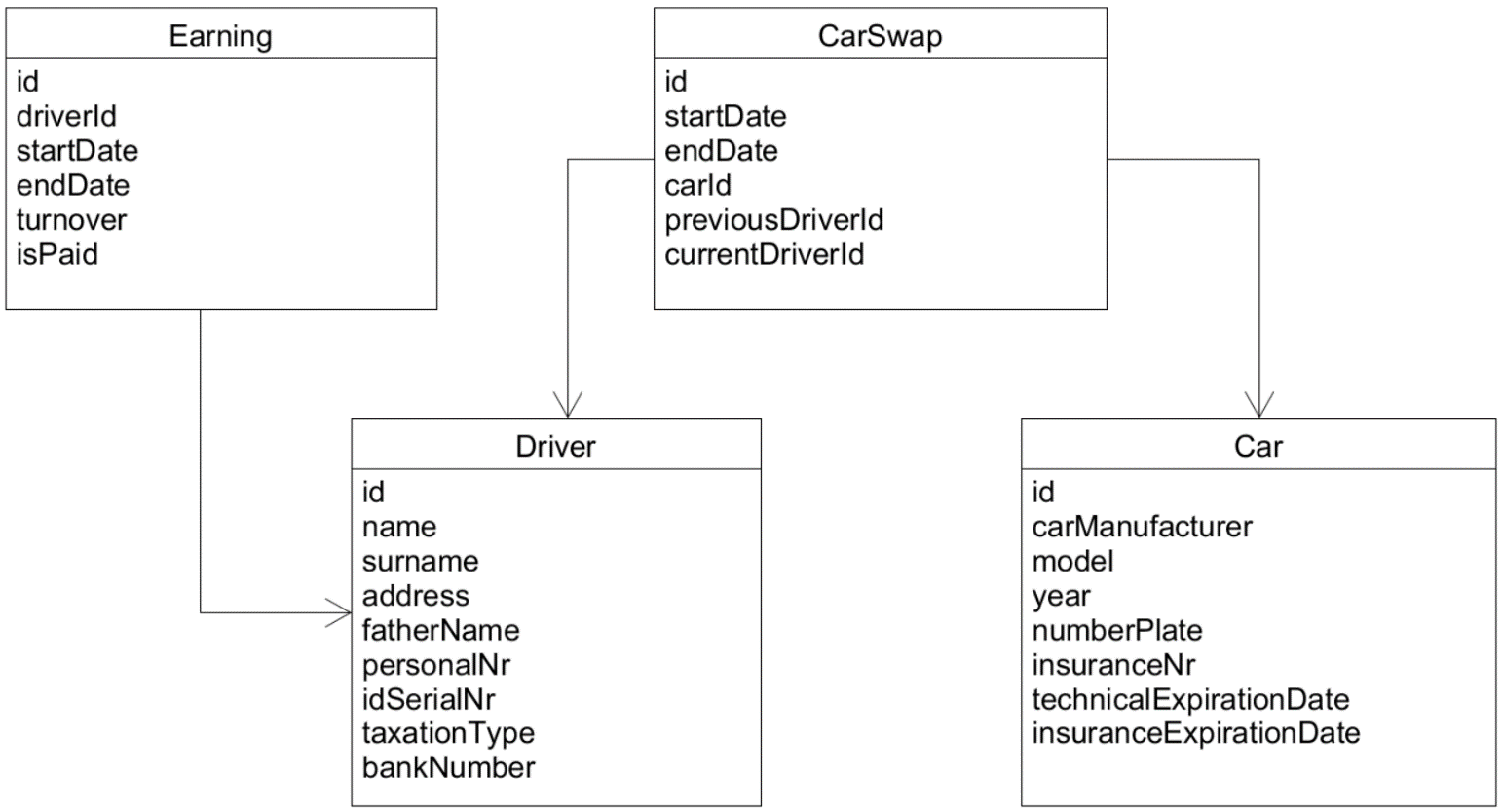


Figure 8. Domain Model

A ***Driver*** individual will have assigned values which are meaningful for the owner of RI-CARS company: *name, surname, address, father's name, personal number, serial number of ID, information about taxation and bank account number*.

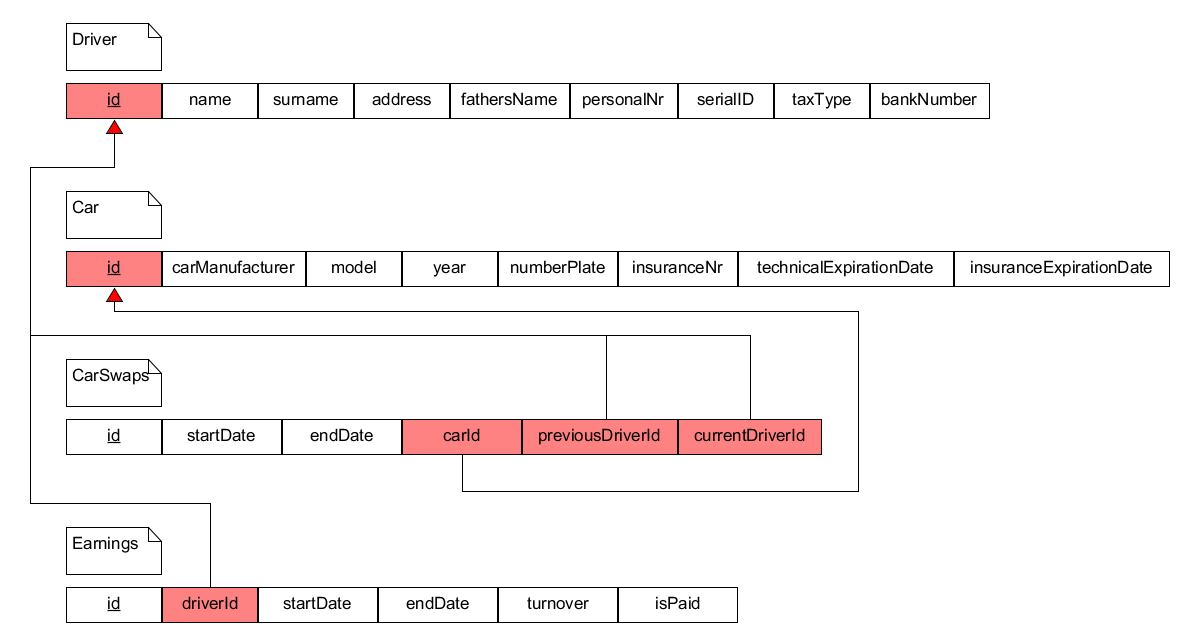
A ***Car*** entity will contain information which are needed for identification purposes: *car manufacturer and model, year of production and number plate*. In order to simplify handling cars, the *date of technical examination expiration date* and *insurance expiration date* will also be stored.

An ***Earning*** object will represent the revenue achieved for company on a certain period of time by an individual driver. It will store data about *the relevant* ***driver****, period's beginning date, period's ending date, the achieved revenue and information if the driver has been paid for the period*.

A ***CarSwap*** entity will store the information about its *beginning and ending date, the relevant* ***car*** *and both previous and current* ***driver*** *of a swap*.

## Relational model

The next image illustrates the Relational Model of the system. It describes how the tables are supposed to store the information and how the primary keys are going to interact with foreign keys.



Model 3. Relational Model

The relational model is an important part of creating a database-based system. This shows how each class will communicate with other classes. Most notable, the ids of ***Driver*** and ***Car*** classes are referenced in two other class tables. This relational model we created has been based by our previously made Solution for the Problem Statement.

## Interaction diagram

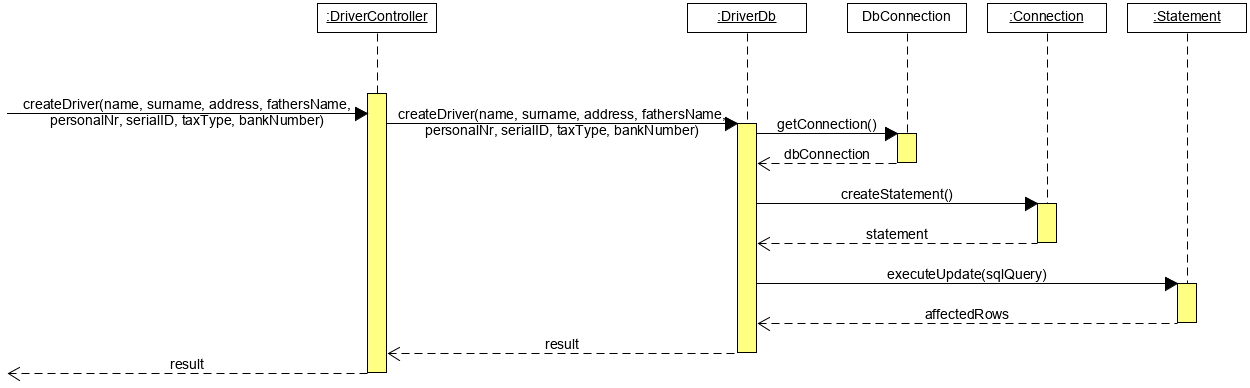
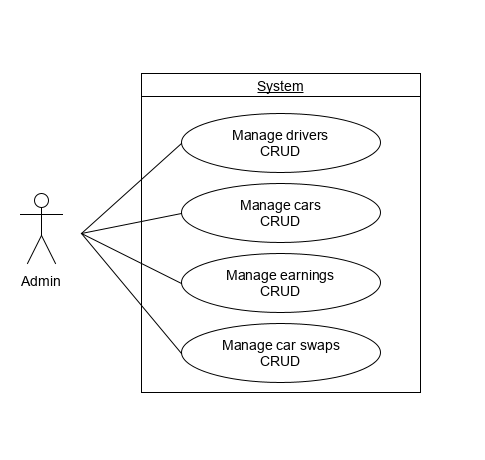


Figure 9. Interaction Diagram

When the controller is requested to create a new driver, it asks the instance of Driver **Data Access Object (DAO)**, which is DriverDb in this case, for inserting the relevant data into a database. In order to do it, the DAO asks the static ***DbConnection*** class for the database connection instance. Furthermore, DAO uses Connection object to retrieve the statement object which is responsible for executing SQL query to database. When committing this specific update method, it retrieves the number of affected rows. Basing on a given integer database class returns the result of operation to the controller layer.

## Use case diagram



In this iteration the use case diagram is described. Each one of the management bubbles explain one classes **CRUD** (Create, Read, Update, Delete).

**Create** – usually creates an entry to the database which then gets managed by other CRUDs.

**Read** – retrieves relevant entries according to the choices made by the admin.

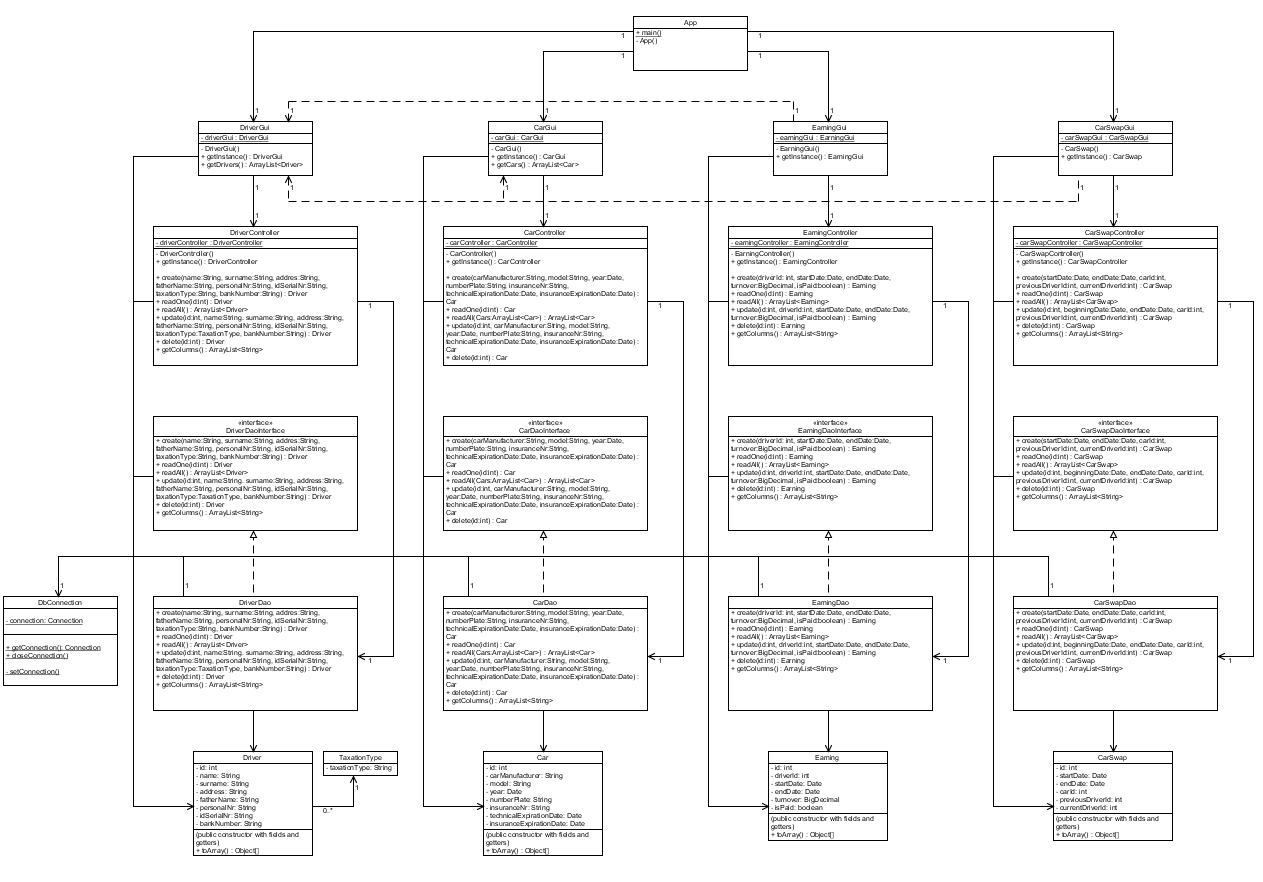
**Update** – changes the relevant information of the entry.

**Delete** – removes the entire entry in the database.

Figure 10. Use Case Diagram

## Design Class Diagram

The application starts at ***App*** class which uses the following GUI classes: ***DriverGui***, ***CarGui***, ***EarningGui***, ***CarSwapGui***. If needed, GUI classes communicate with each other. CarSwapGui can retrieve information from DriverGui and CarGui, EarningGui can retrieve information from DriverGui. All of GUI classes have a private constructor and a static field of themselves. It makes them follow the **singleton** pattern. GUI classes represent entities which are managed by the application. Every single one of those GUI classes has a connection to the relevant controller which also follows **singleton** pattern.



Model 4. Design Class Diagram

The full Design Class Diagram can be found in [Appendix A](#_Design_Class_Diagram).

Controllers do not communicate with each other directly. They take information from GUI layer and pass them to database layer. The database layer consists of **Data Access Object (DAO)** classes which implement relevant interfaces. All DAO classes use the singleton instance of DbConnection class which has two public methods: ***getConnection()*** to establish the connection if needed and ***closeConnection()*** to close the connection (when closing the application). When the connection is retrieved, DAO classes send queries to the database and, based on the received answer, they construct corresponding objects and pass them to controllers as returns. Model classes (***Driver***, ***Car***, ***Earning***, ***CarSwap***) are known by all classes from a column, but especially by **DAO**s which construct new objects that are inserted into the database. The ***Driver*** class has a field of **enumerated** type ***TaxationType***, which describes the state of a driver, what is required to correctly calculate its wages.

## Fully dressed use case

**Create cars:**

|  |  |  |
| --- | --- | --- |
| **Use case** | **Create cars** | |
| **Actors** | Admin | |
| **Frequency** | Rarely | |
| **Preconditions** | A system is in place | |
| **Postconditions** | A car has been added to the database | |
| **Main Success Scenario (flow of events)** | **Actor (action)** | **System (response)** |
| 1. An admin opens the desktop application. | 2. The system starts and opens the desktop application. |
| 3. An admin enters the correct log-in credentials and logs in. | 4. The system recognizes the account that was put in and redirects to the main page. |
| 5. An admin selects the car tab. | 6. The system opens the car tab. |
| 7. An admin clicks on ‘add car’. | 8. The system opens the ‘add car’ tab. |
| 9. An admin types in the credentials. |  |
| 10. An admin confirms the credentials as a new car. | 11. The system checks if all mandatory information has been put in. Then it adds the car to the database. |
| **Alternative flow** | **3a. An admin enters the wrong log-in credentials and tries to log in.** | |
| 3b. An admin enters the wrong log-in credentials and tries to log in. | 4b. The system is unable to locate the credentials that were used for the log-in and displays the following message: *ERROR: PLEASE FILL IN THE CORRECT USERNAME AND PASSWORD* |
| **4b. The system is unable to connect to and retrieve user credentials from the database.** | |
| 3a. An admin enters the correct log-in credentials and logs in. | 4b. The system is unable to establish a connection with the database and therefore does not recognize the log-in credentials that have been entered. The system displays the following message: *ERROR: UNABLE TO CONNECT TO THE DATABASE* |
| **9b. An admin fails to enter all the mandatory information.** | |
| 9b. An admin types in the car credentials but leaves some options open. |  |
| 10b. An admin confirms the credentials as a new car. | 11b. The system checks if all mandatory information has been filled in. It has not, and some options are left open. The system displays the following message: *PLEASE ENTER ALL INFORMATION*, before returning the admin to the ‘new car’ tab. |

Table 3. Create Cars Fully Dressed Use Case

## Test Cases

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Open desktop app** | **Login information** | **Car tab** | **Necessary car information** | **Confirmation of car** | **Expected outcome** |
| **TC1** | **Successful  input** | v | v | v | v | v | A new car has successfully been added to the database |
| **TC2** | **Wrong credentials** | v | i | N/A | N/A | N/A | N/A |
| **TC3** | **No connection** | v | **i** | N/A | N/A | N/A | *ERROR: UNABLE TO CONNECT TO THE DATABASE* |
| **TC4** | **Insufficient information** | v | v | v | i | N/A | *ERROR: PLEASE ENTER ALL CAR DETAILS* |

Table 4. Test Case Scenarios

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Open desktop app** | **Login information** | **Car tab** | **Necessary car information** | **Confirmation of car** | **Expected outcome** |
| **TC1** | **Successful input** | Yes | NONE | Yes | id: 12367  carManufacturer: Volkswagen model: Edicola  productionYear: 2007 numberPlate: 1C1AK-FTW  insuranceNumber: 88558 technicalExpirationDate: 09-08-2020 insuranceExpirationDate: 09-12-2020 | Yes | A new car has successfully been added to the database |
| **TC2** | **Wrong credentials** | N/A | NONE | N/A | N/A | N/A | N/A |
| **TC3** | **No connection** | Yes | NONE | N/A | N/A | N/A | *ERROR: UNABLE TO CONNECT TO THE DATABASE* |
| **TC4** | **Insufficient information** | Yes | NONE | Yes | id: 12367  carManufacturer:  model: Edicola  productionYear: 2007 numberPlate:  insuranceNumber: 88558 technicalExpirationDate: insuranceExpirationDate: 09-12-2020 | N/A | *ERROR: PLEASE ENTER ALL CAR DETAILS* |

## Programming

### Database & SQL Scripts

This part of the report explains the use of scripts to form a database for RI-CARS. These scripts create various tables for the program to use and store information in. The further screenshots include the code that is needed to form a certain table with various constraints and data types.

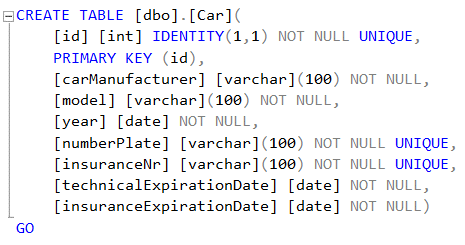
The database is needed for the application to correctly operate. The following scripts generate proper tables in a database. All **varchars** are limited to 100 characters because of SQL **constraint**'s limits. Data type used for a data storage is **date**. All values are marked as ***NOT NULL*** since they are needed for legal purposes.

Figure 11. Create Car Table

The given example is for creating a “Car” database table. The first row creates it and gives it the name. The following rows are for assigning names to the columns and giving them respective datatypes. Most notably, the first row named “id” is assigned for every newly created row throughout the app and adds +1 for each other created row. It identifies the row as, in this case - a car.

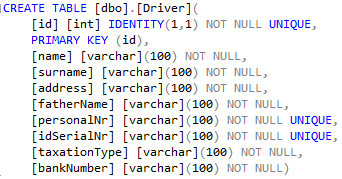


Figure 12. Create Driver Table

The primary key of the ***Driver*** table is a unique integer ***id***, which is always assigned automatically by the database. The personal number (e.g. PESEL in Poland) and ID serial number are unique for each driver therefore they need to be of special value.

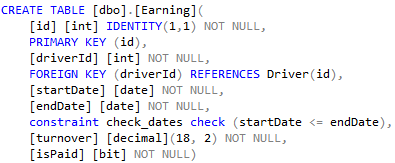


Figure 13. Create Earning Table

The primary key of the ***Earning*** table is a unique integer ***id***, which is assigned automatically by the database. Each earning has its driver which is also an entity in the database therefore integer ***driverId*** is a foreign key which references column ***id*** from ***Driver*** table. The start date of the introduced period cannot be older than its end date therefore there is a contraint which checks if ***startDate*** is earlier than the ***endDate***.

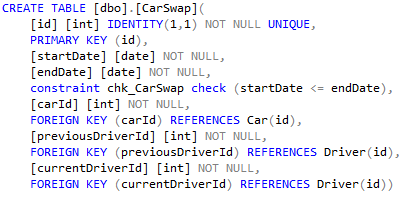
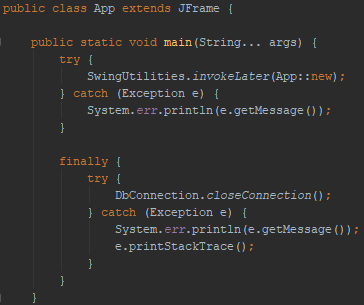


Figure 14. Create CarSwap Table

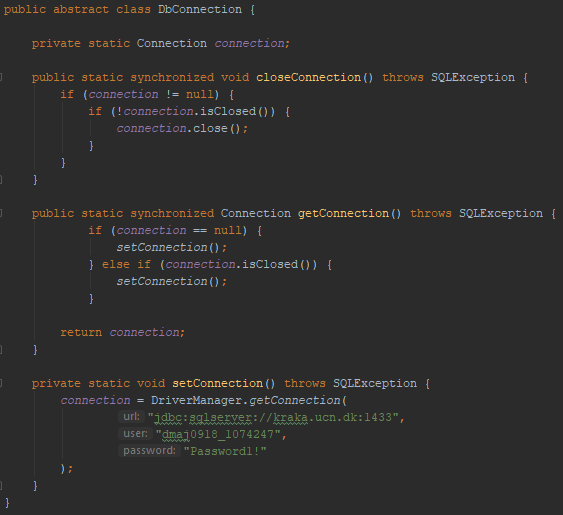
The primary key of the ***CarSwap*** table is a unique integer ***id***, which is assigned automatically by database. Each car swap has its current and previous driver which are also entities in the database therefore integers ***currentDriverId*** and ***previousDriverId*** are foreign keys which reference column ***id*** from ***Driver*** table. Each car swap concerns a car which is also an entity in the database therefore the ***carId*** integer is a foreign key which references column ***id*** from ***Car*** table. The start date of the introduced period cannot be older than its end date therefore there is a contraint which checks if ***startDate*** is earlier than ***endDate***.

### Coding

#### Main class

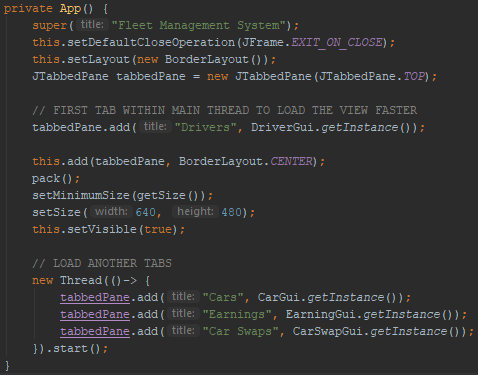
Main class of the project has been named ***App*** and it is placed in the ***gui*** package. It contains the ***public static void main(String… args)*** method which attempts to run GUI thread with ***SwingUtilities.invokeLater()*** method. As parameter it takes a reference to the constructor of ***App*** class.

Code 1. App

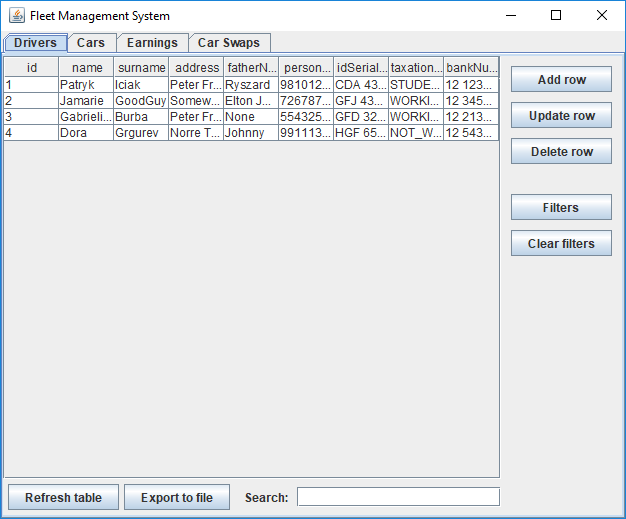


Code 2. DbConnection

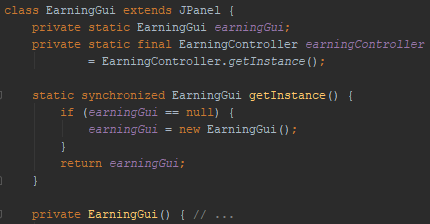
The constructor of ***App*** class is set as private – in order to prevent unwanted invocation (for example outside a proper Swing thread). If it fails, it prints a relevant message in the console and then ***finally*** closes a connection (if one exists) by invoking the ***DbConnection.closeConnection()*** method.



Code 3. APP #2

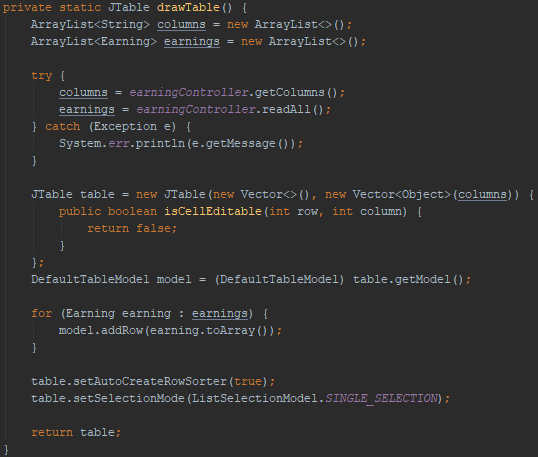
The ***App*** constructor sets basic settings of a frame and loads all other GUI classes. The very first tab named ***Drivers*** is loaded within the same thread. After that, the frame is set as visible and after that the application loads other tabs within a new Thread - in order to faster run the application with a view of the first tab. More tabs are loaded when the frame has become visible.

Code 4. Fleet Management System Interface

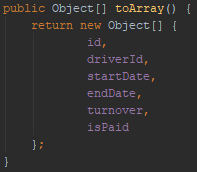


Code 5. Earning GUI

Every tab (every GUI class) is created as **a singleton**. Therefore it has defined a **private constructor**, which is used only by ***static synchronized getInstance()*** method. When the instance is requested, the relevant one is created or returned if it already exists.

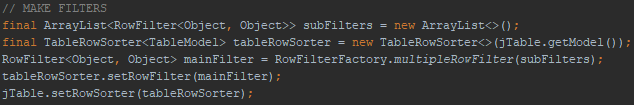
Every GUI has its ***private static drawTable()*** method which returns a relevant ***JTable*** object. The method ***isCellEditable()*** is overridden in order to prevent users from editing data in a created table.

Code 6. drawTable

To make it easier to represent data in a table, every model class has ***toArray()*** method which returns an array ***Object[]*** that stores all its fields.

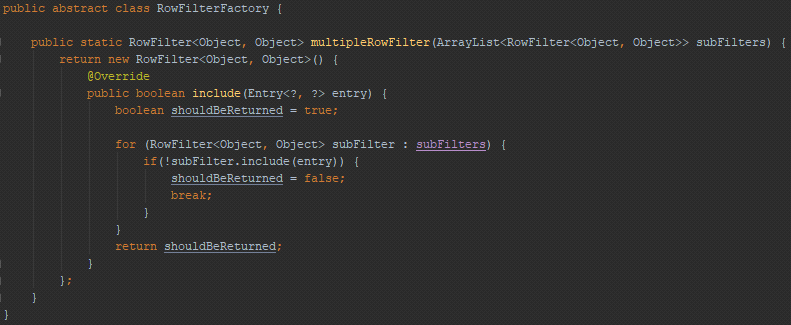
Code 7. toArray

When the table has been created and displayed, a user is presented with the option to filter its data.



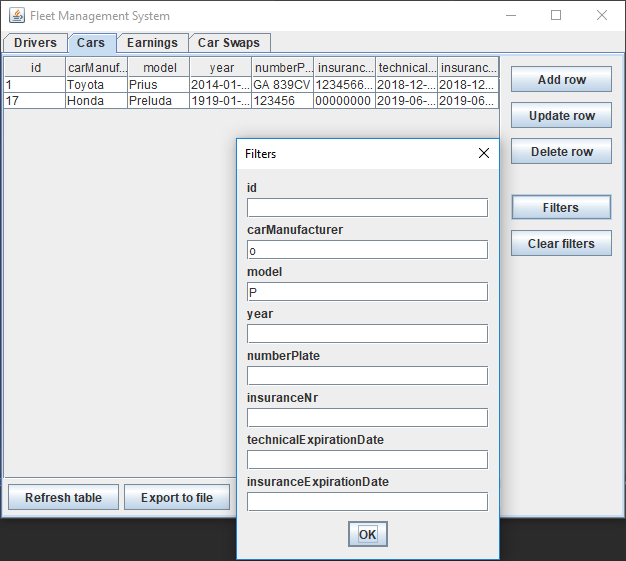
Code 8. Filters

To make it possible, a ***RowFilter*** is needed. To allow filtering by multiple columns, a proper filter is required. Therefore, an abstract class ***RowFilterFactory*** has been created.



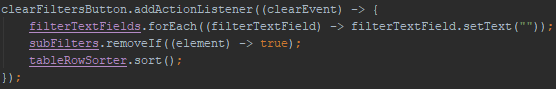
Code 9. RowFilter

***RowFilterFactory*** class creates and returns an object of class ***RowFilter***, whose method ***include()*** (condition check) is overridden. The static method ***RowFilterFactory.multipleRowFilter()*** takes the filters’ list as a parameter. Returned filter checks if all of the filters’ requirements are fulfilled and returns a relevant **boolean** value (***shouldBeReturned***).



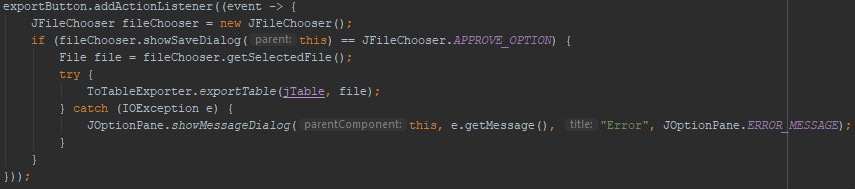
Code 10. FleetManagementSystem Filter

In order to clear all filters, a relevant button has been created. Its action listener cleans up values of proper ***JTextField*** objects and also cleans up the filters’ list and updates the table view.



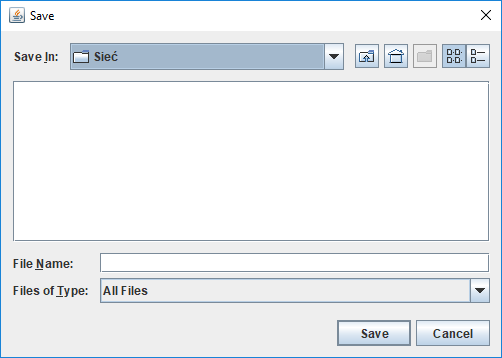
Code 11. clearFiltersButton

To allow a user to export a viewed table, a ***JTextButton*** has been created.



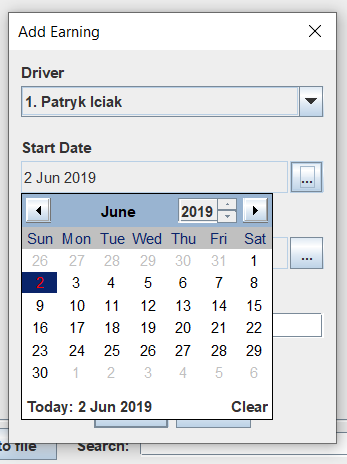
Code 12. exportTable

It has been given an action listener which exports a table by using ***ToTableExporter.exportTable()*** method. This is a static method which has been created to simplify the code inside GUI classes.



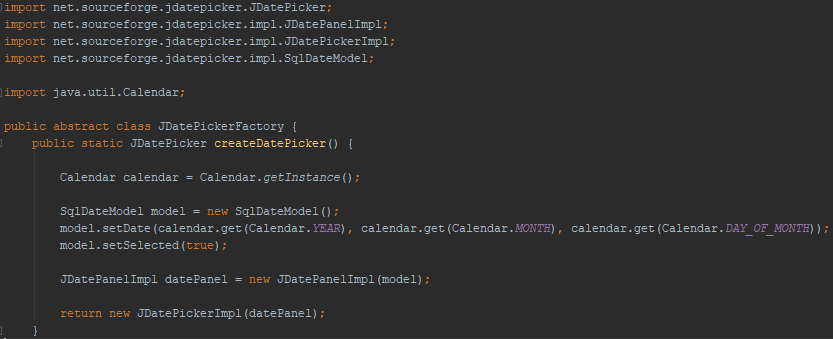
Code 13. exportTable Interface

GUI should be able to let a user enter the data in a convenient way. To enter a date, a relevant component (Calendar) has been implemented.



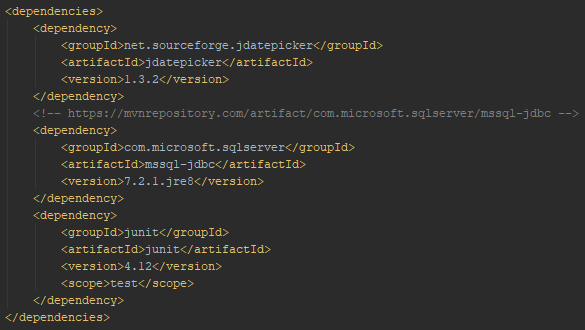
Code 14. Calendar

Inside ***JDatePickerFactory.createDatePicker()*** the object of class ***JDatePicker*** is created.



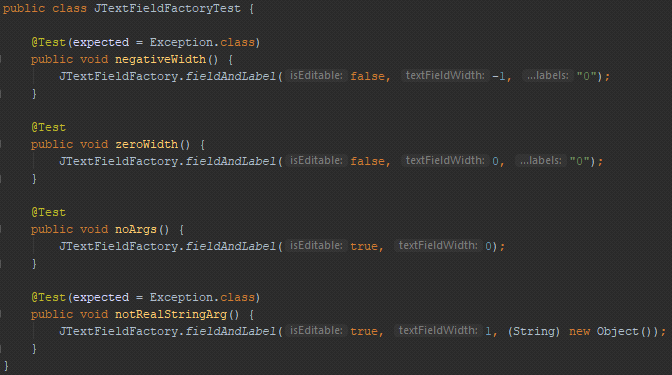
Code 15. JDatePicker

***JDatePicker*** object originates from an external library. Since external libraries are used in the project, a **Maven** build automation tool is used. To import proper libraries the following lines are added to ***pom.xml*** file.



Code 16. Maven

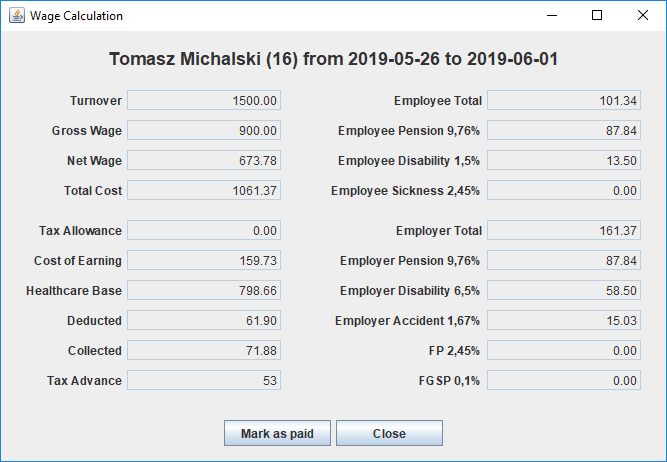
The project also imports a relevant SQL Driver and JUnit library for unit testing.



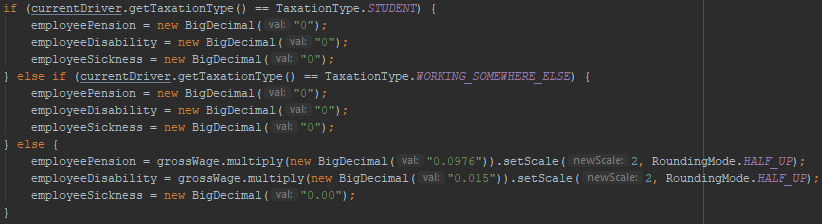
Code 17. Test

The given example shows testing of JTextFieldFactory class, which returns a **HashMap** of text fields and their labels. The very first test expects an exception to be thrown since a text field cannot have a negative width. ***zeroWidth()*** and ***noArgs()*** tests expect a proper creation of an object (which the most likely would have never been created in real life world situation).

The desired calculator of wages has been created



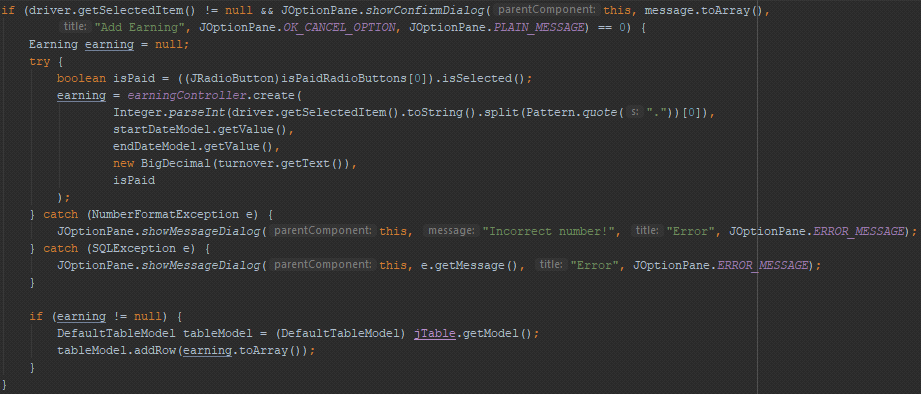
Code 18. WageCalculation Interface



Code 19. taxationType

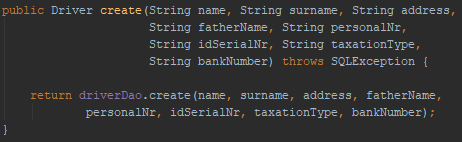
It computes wages, taxes and required contributions basing on drivers' turnover and taxation type. The used variable type is **BigDecimal** which ensures appropriate calculation precision.

#### Data Access Objects, Controllers and GUI communication



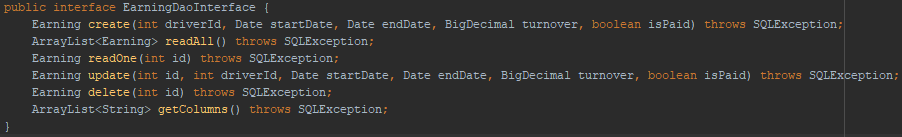
Code 20. isPaid

When inserting a new row to the database, some variables have to be parsed to a proper format or type. If any error has occured, the relevant message is printed out on a screen in a message dialog. Finally, if the operation is successful, a table view is updated.



Code 21. createDriver

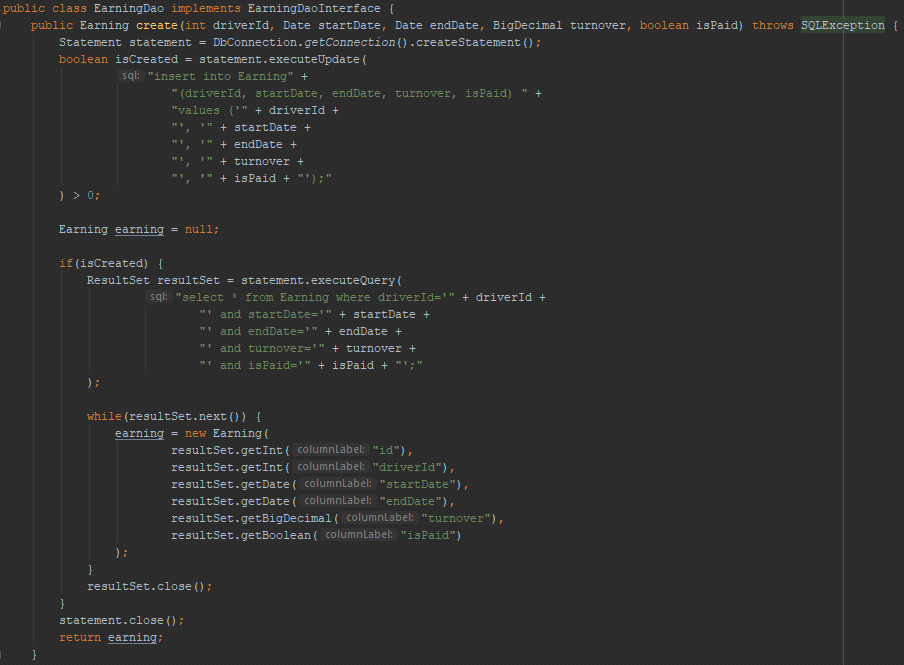
GUI classes invoke controllers' methods. Controllers pass parameteres and execute indicated methods of a data access object.



Code 22. earningDaoInterface

All data access objects implement the relevant interface.

Data Access Objects send queries to a database and return a relevant object (or a ***null***value).



Code 23. earningDaoInterface Implementation

## Conclusion

To evaluate the general performance of the group project, we will primarily assess the way problem statements were handled, how well the group motivation and task-solving went and whether or not we fulfilled all of our requirements.

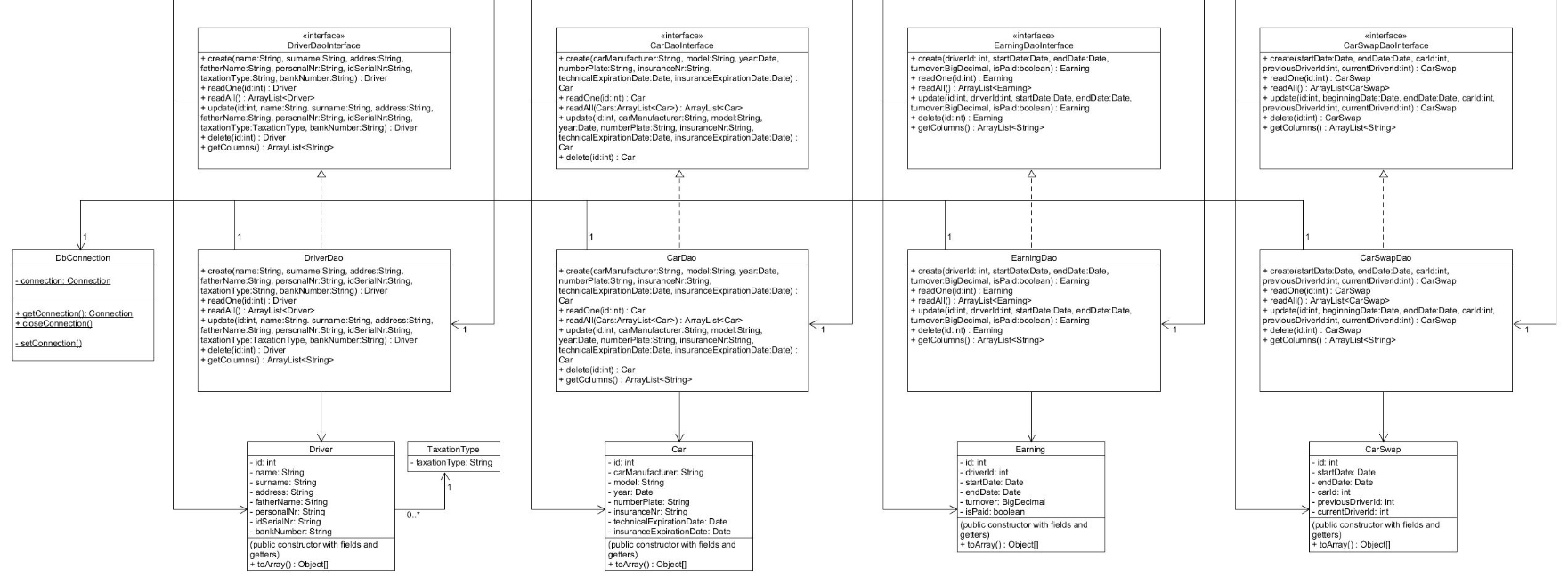
#### Problem Statement Solving

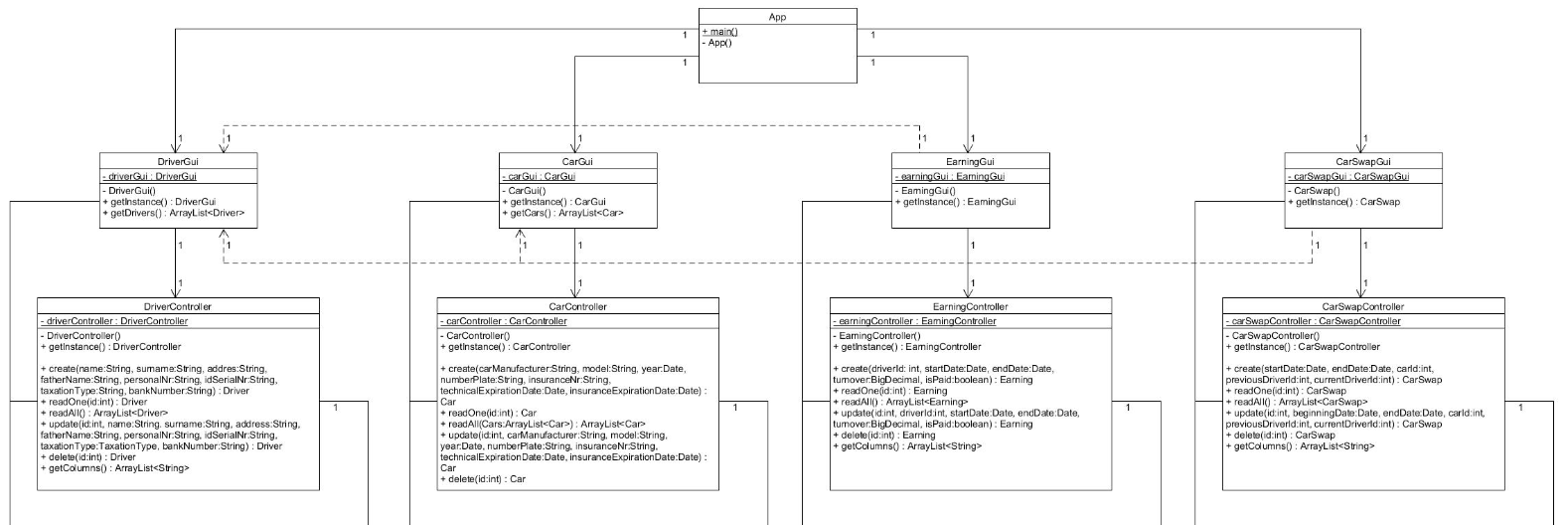
The demands of the tasks displayed before us got solved without any major issues,which is a better outcome than we could have predicted, especially on the programming parts.

It is safe to assume that the RI-CARS owner can perform his everyday tasks much more smoothly and quickly while also having easier access to past transaction entries/documentation all of which is possible just from using our system. Ryszard can also schedule car swaps while using the built-in database access throughout our system.

#### Learning Outcomes

With our continuous work on the project, we managed to gather valuable information which could assist us with similar tasks in the future.Design Class Diagram - Appendix A





## SVN Repository

In this repository You can find our entire application:

URL: [*https://kraka.ucn.dk/svn/dmaj0918\_2Sem\_3*](https://kraka.ucn.dk/svn/dmaj0918_2Sem_3)

Password: *IsAllowed*

## Group Contract

This contract will point out the responsibilities of the group members as well as their work ethics.

Reviewed by group 3's very own members; Patryk Iciak, Gabrielius Burba, Dora Grgurev & Tristan Schalk, they hereby agree to the following terms:

Time and place of meeting:

* Whenever a group meeting is scheduled, every member needs to be present, no excuses.
* Usual meeting time is 10 am at Peter Freuchensvej 23. The meetings last until the group members decide that it's enough for the day.
* Meetings are not scheduled on a daily basis, it all depends on the remainder of work left before the hand-in date.

Responsibilities:

* Members are responsible for completing their duties even outside of scheduled meetings
* If a member requires assistance, a meeting can always be scheduled to assist him/her

Team roles:

* Everyone is on the same managerial level.

Team ambitions:

* It is expected from the members to stay motivated throughout the entire project and to push others forward if they're down.
* The team will strive to do the best they can for this and future projects

## Reference List

<https://www.entrepreneur.com/encyclopedia/sole-proprietorship>

<https://en.wikipedia.org/wiki/Business-to-employee>

<https://en.wikipedia.org/wiki/Broker>

<https://en.wikipedia.org/wiki/Taxation_in_Poland>

<https://smallbusiness.chron.com/definition-entrepreneurial-organization-19019.html>

1. [Business model types](https://en.wikipedia.org/wiki/Business-to-employee) [↑](#footnote-ref-1)
2. Poland [↑](#footnote-ref-2)