Piekary Śląskie, 24.01.2019

Semester: III
Group: 3
Section: III

# COMPUTER PROGRAMMING LABORATORY

Author: Karolina Wylężek
E-mail: karowyl642@student.polsl.pl
Tutor: Anna Gorawska

## 1. Task topic

Create car rental application using elements of object oriented programming in C++ language.

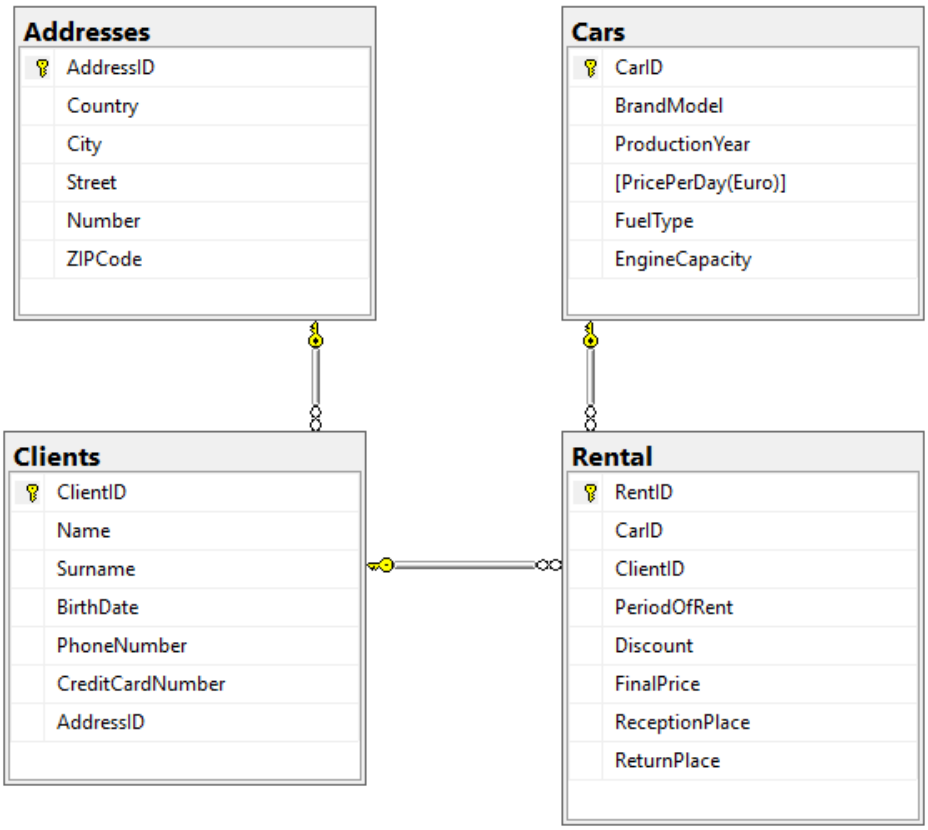
## 2. Project analysis

In order to fulfill the requirement of the project I decided to create an application for car rental. I wanted to create a program that will be easy to use for the client and, in the same time, practical for the company. To achieve that I had to think about a few aspects.

The first one was how to store the data. In the beginning I wanted to do it using the database. I've even created one using MS SQL (pic.1). Unfortunately connection of the database with program written in C++ programming language turned out too difficult and time-consuming for now, but I think it can be good development for future functioning of the application. In that case I decided to store the data in lists. I decided to choose that kind of data structure, because it lets us add the new element very quickly, what is useful when we have to register the new client or add a new car to our offer. The disadvantage of the list is the fact that to get access to a single element it is necessary to use iterator, because lists don't let us do that using indexes.

The other aspect that I had to think about was how to make the application easy to modify. The first thing to do that, as I mentioned before, was choice of the lists to store the data. Secondly I have created a class template to let the company change the currency in which they want to show prices in their application (used for example when the company want to enter the market of different countries). I also thought that it would be useful for company to be able to check the history of rentals, so in method responsible for rent I implemented ofstream responsible for recording all transactions in text file.

The last thing that I had to consider was how to make the application easy in use for the client. The procedure of renting a car starts with simple login. Users give their name and surname and the application checks whether the person already has "an account". The program does that using loops, if statements and list's iterator. If it does not find the person, it creates the new list element and the client can proceed. Then the available cars appear on the screen and the user rent a car by simply writing its registration number. Here the program can throw 2 exceptions: when the user owes the company already too much money or when the user gives wrong registration number.

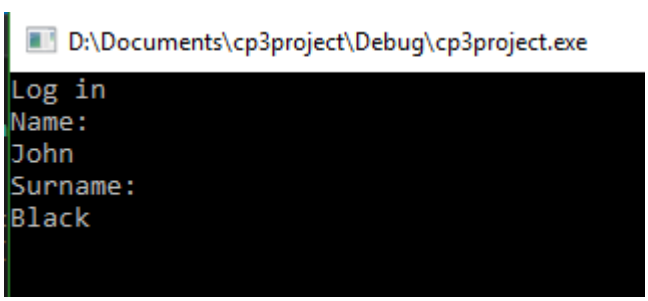


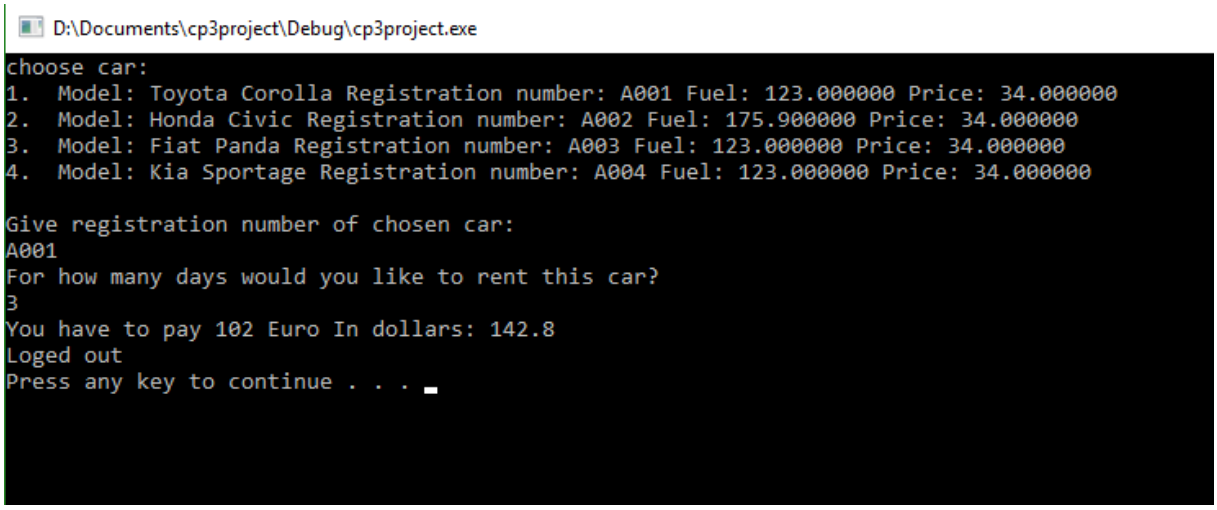

*Pic. 1- Database*

## 3. External specification

In order to rent a car you should make following steps:

1. Write your first name (e.g. John)
2. Write your last name (e.g. Black)
3. Give the registration number of chosen car (e.g. A001)
4. If you owe the company too much money, the rental will not be possible (the application will inform you about it)
5. If you give wrong registration number, the application will ask you to do it again.
6. Write for how many days would you like to rent the car (e.g. 2)
7. On the screen just appeared the price of your rental. Enjoy the pleasure of driving best cars in the city ☺






## *4. Internal specification*

The program is divided into the cpp and header files, that contains classes and main function. In order to properly describe the program I will describe separately the most important parts of classes and main function:

1. **Currency class**

The currency class is the base class for two other classes: Euro and PLN. Here we can observe the example of polymorphism. This class contains virtual function, that lets convert the price from every other currency to dollars. Derived classes have specific implementation of this function.

2. **Car class**

This class contains all the information that we need to have for every car. It is a template function, because it contains "Price" object and, as we already know, the price can have different types (Euro or PLN).

3. **CarRepo class**

In that class all the cars and information about them are stored in list. The class contained also "FindCar" method that let us find the specific car by its registration number.

4. **Client class**

This class store the information that we need to have about every client.

5. **ClientsRepo**

Here clients and information about them are stored in the list. This class has also FindClient and AddClient methods that let us add new client or find the client that already is "registered" to the application respectively.

6. **Rental class**

This class is responsible for the rental procedure. We can find there few methods. First of them, called RentCar, is responsible for "renting a car" – it changes the availability of chosen by client car to "unavailable", assigns the car to the client and then saves the information about the rental in text file. Here we can also find the implementation of exceptions: one, when the registration number given by user don't match any of available cars number and the second one in case when the client has already too high debt to rent a car.

Method “Rental” is responsible for preparing a place for client that is about to login into the system – it makes the “loggedClient” object NULL and the “Login” method assigns the actual client to “loggedClient”.

Then we have two methods whose function is to write all cars or clients on the screen. In case of “OutputCar” method the program check the availability of every car and shows only the available ones. In “OutputAllClients” all clients are printed with all information that we have about them together with the message whether the client has any car rented or not. This function is in use only for the company employees so right now it is not called in main function, but in case when company would like to check the list of all clients it is easy to add it to version of main for company.

7. **Main**

Here all methods needed to conduct the rental procedure are called. In main there is also calling of a virtual function from “Currency” class and simple interface implementation. Everything is composed using various loops.

8. **typedef specifier**

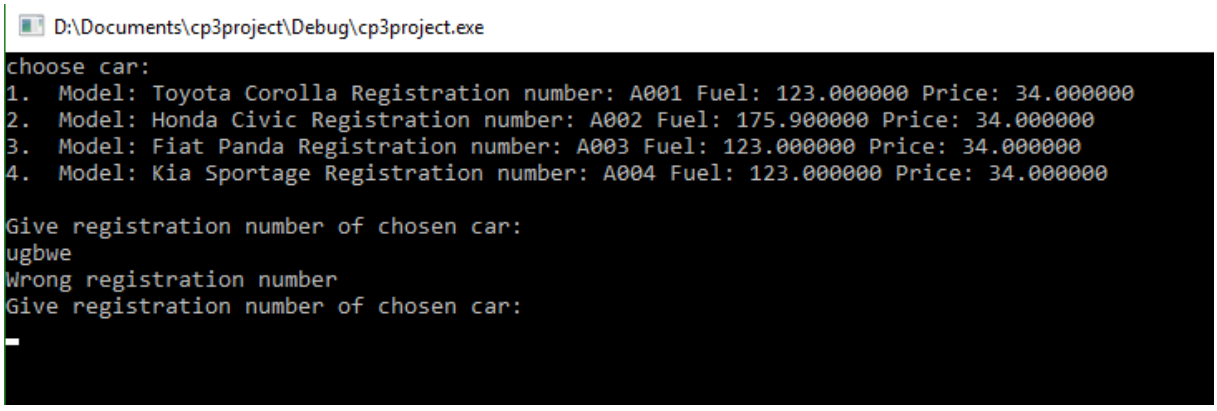
In stdafx.h file we can find typedef specifier. In here we can change the currency that we want to use on our application page.

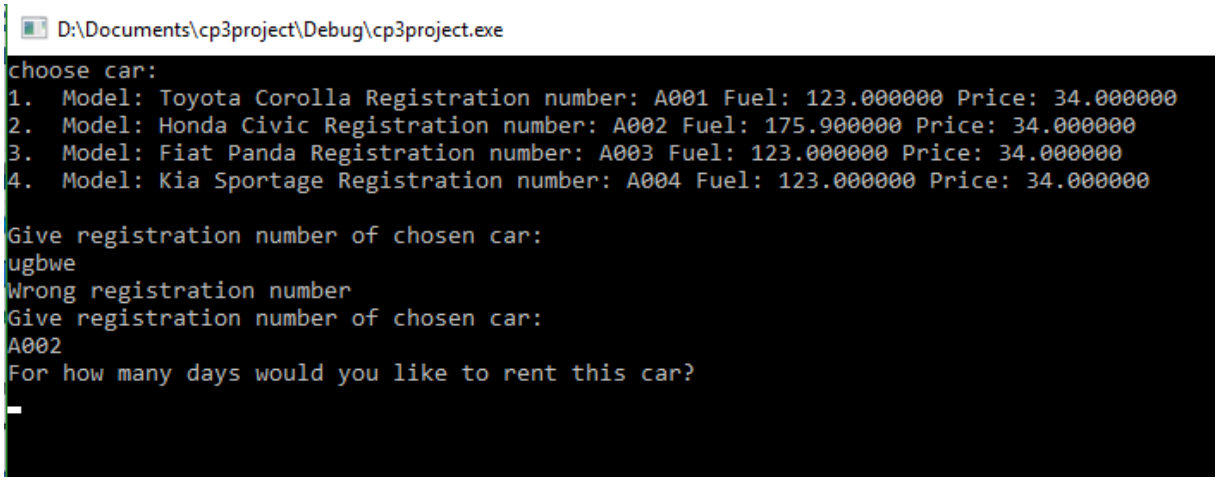
## 5. *Source code*

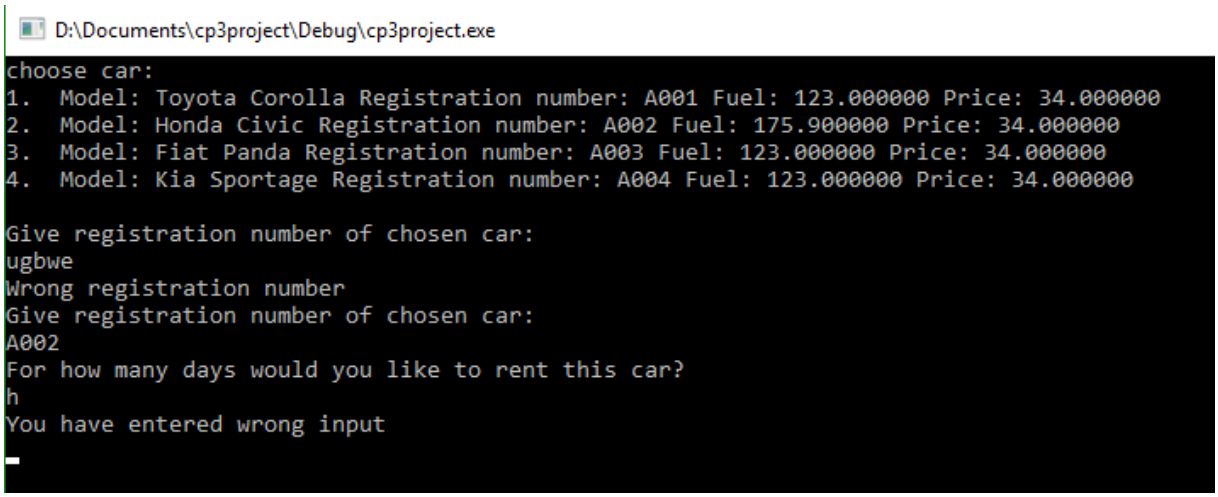
https://github.com/KarolinaWylezek/CarRent.git

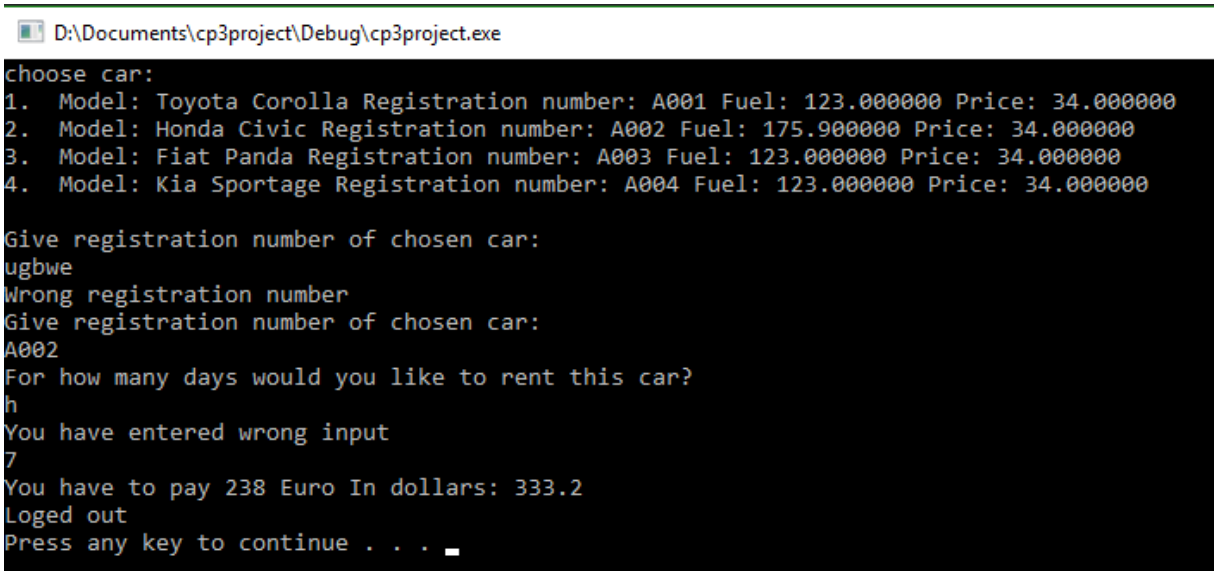
## 6. *Testing*

First I made all procedure, making mistakes to check if everything works good:

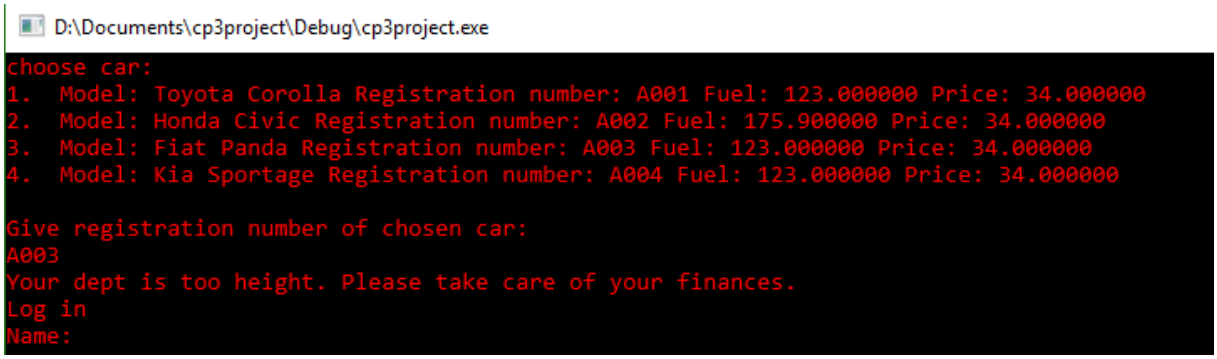





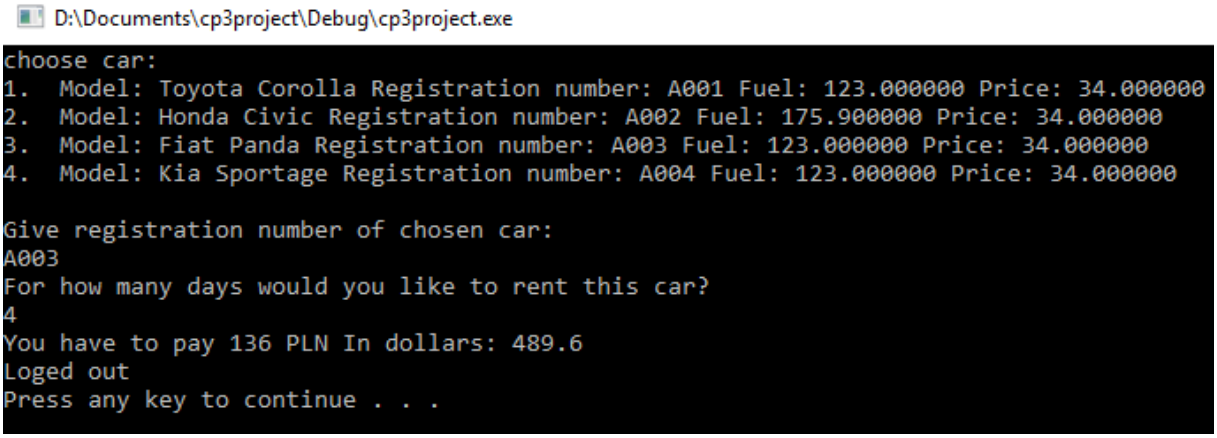



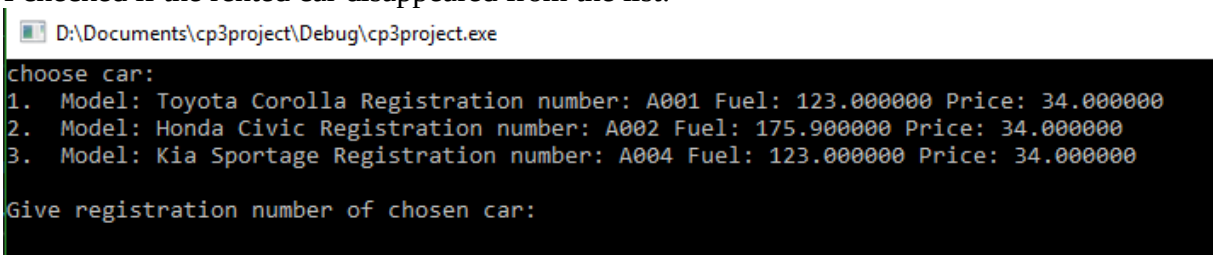

Then I tried to log in as a person with high debt:




And I changed the currency to PLN:




I checked if the rented car disappeared from the list:




In the end I opened the text file to make sure that all rentals have been saved: