**Car Rental System**

*The purpose of this project:*

We want to create an easy usable system for people who want to rent cars. The purpose? Streamlining the way the cars are rented.

*The strucure:*

**System Class**

* company’s name – static member
* **Company’s locations - array**
  + Nume filiala
  + country
  + city
  + the number of cars available
  + mark (extracted from the reviews)
  + review
  + **Customers - array**
    - first name
    - last name
    - email
    - telephone
    - CNP
    - ID card series
    - ID card number
    - Mark for the company branch
    - **Rental Process Class**
      * **RentalStartDate**
        + start day
        + start month
        + start year
        + **Start Time**

hour

minutes

* + - * **RentalEndtDate**
        + end day
        + end month
        + end year
        + **EndTime**

hour

minutes

* + - **Driving license Class**
      * **StartDate**
        + start day
        + start month
        + start year
      * **EndDate**
        + end day
        + end month
        + end year
      * is the person able to drive? (yes - true/no - false)
    - **Customers’s home address**
      * country
      * city
      * postal code
      * street
      * number
    - **Date of birth Class**
      * day
      * month
      * year
  + **Car Class**
    - make
    - class
    - color
    - brake horsepower
    - engine
    - doors
    - seats
    - transmission (automatic/manual)
    - consumption
    - availability (if during a certain period, the car is available for rental)
    - price per day
    - deposit
    - payment in advance
    - type (economy/medium/premium)
    - **Car’s review Class**
      * value considering the price
      * car cleanliness
      * comfort
      * car condition
      * nota overall
  + **Location’s address Class**
    - street
    - number
    - postal code
* **Contact Class**
  + telephone
  + email
  + administrator name

*Conventions:*

1. The name of every member of the class starts with: “m\_” .
2. The name of every pointer starts with: “p\_” , “p<variable\_name> .
3. Code indentation

We will use the indentaion with 4 spaces, without TABs. For Visual Studio, the setup should be done as it follows: Tools -- Options – Text Editor – C/C++. Moreover, we will set: Tab size 4, Ident size 4, [x] Insert spaces.

1. Header files

Every „ .H ” files should start and end with #ifndef / #deﬁne / #endif and a unique macro or #pragma once, including the file name. For instance, for a files called “help\_tools.h”, we will write as is presented below:

#ifndef \_HELP\_TOOLS\_H\_

#define \_HELP\_TOOLS\_H\_

...

#endif // \_HELP\_TOOLS\_H\_

or

#pragma once

...

1. The name of parameters and methods should use Capital Case. For example:

int MyFunction( int FirstParameter );

1. Naming local variables

The naming of local variables should be done using almostCapitalCase. For instance:

int retStatus;

CC\_LIST bufferLength;

Identifiers in a block of code are not allowed to have the same nams as identifiers in “parent” blocks in order not to hide them.

1. Naming global variables

The naming of global variables should be done using almostCapitalCase and the name should pe preceded by „g”. For instance:

int gCcVector;

CC\_LIST gList;

Identifiers in a functions are not allowed to have the same name as global identifiers in order not to hide them.

1. The description of code blocks, the usage and the curly braces positioning.

Every for, while, if, do, switch should be followed by { }, even if there is only an instruction in the block. The block {} must have the same indentation level as the instruction and the contents of the blocks, indented with a level. In the case of “if”, the “else” should be written at the same indentation level with the “if”. For instance:

for (i = 0; i < 100; i++)

{

...

}

if (a > b)

{

while (x < 5)

{

x++;

}

// ...

}

else

{

...

}

1. Signature of local variables names, functions, structures etc.

The parameters, local variables, functions, structures etc. should have descriptive names. The only exception is the case of cycle/index variables.

1. Constants

The usage of constants should be marked as often as possible (constant parameters, constant functions – including the prototypes).

1. Classes’ names

The name of the every class should start with Capital Case.