# EOOP - preliminary project

### Date:29.04.2022 Semester: Spring 2022

### Author and Group:!!!!!!!!!!!!!!

### Subject (Keyword):Rent a Car

## **Description of the project**

### Overview of the project

### Rent a car service have cars which client can rent, Clients data and schedule for each car. Each people can come to service, fill out a form, choose the car and rant on dates which they want. Service making sure that one people can’t take more then one car in day and one car can’t rent more than one people in day.

### Class and data structures overview

### There are following classes: RentCarService, Car, Client, ScheduleForCar, ScheduleForClient and Schedule.

### RentCarService contain unlimited number of Cars and Clients which you access with passport number and method Menu in which service manager can call method for add, delete, update, rent, return and show cars, add new client, and cancel rent schedule, this method ask date for example choose car, enter data for new car or client, in which period client want rent car or which schedule delete and check data for right.

### Car contain data about car and set of SchelduleForCar and methods to initializate new car, show this car, update mileage, price and deposit, add, delete and show schedule. These methods call method from RentCarService and they do not require data from manager, besides update methods.

### Client contain data about client and set of SchelduleForClient and methods to initializate new Client, show this Client, add, delete and show schedule. These methods calls method from RentCarService and they do not require data from manager.

### Schedule contain data from which days to which rent car. It has initialization and operator < for set.

### ScheduleForCar this is derived class from Schedule and contain pointer to client and method to show and initializate.

### ScheduleForClient this is derived class from Schedule and contain pointer to car and method to show and initializate.

### Restrictions, limits, assumptions

### R1. One people can’t take more than one car in day

### R2. One car can’t rent more than one people in day

## **Case study (a memory map)**

**At the end of Preliminary Project.**

## **Declaration of the classes**

class RentCarService

{

private:

    vector<Car> cars;            // Vector of cars

    map<string, Client> clients; // Map of clients with their passport number as key

    void addCar();

// Add a car to the vector with parameters from the manager

    void addClient();

// Add client to the map with parameters from the manager

    void rentCar();

// This function call Car\* chooseCar() and Client\* chooseClient(), ask date period from manager and if can rent the car to the client using function Car::addSchedule() and Client::addSchedule()

    void deleteRentSchedule();

// Delete rent schedule from the car and client call Client\* choose Client using function Client::deleteSchedule()

    void showCars();

// Show cars all cars in the vector using function Car::showCar()

    void returnCar();

// Return car call Client\* chooseClient() and call Client::deleteSchedule()

    void updateCar();

// Update car call Car\* chooseCar() and ask parameters from the manager and call Car::updateMileage(), Car::updatePrice() or Car::updateDeposit()

    void deleteCar();            // Delete car call Car\* chooseCar()

    Car \*chooseCar();

// Choose car from the vector using function Car::showCar() and ask for the number of the car from the manager

    Client \*chooseClient();

// Choose client: ask for the passpot number of the client and return the pointer to the client

public:

    RentCarService(); // Constructor

    void enterMenu();

// Enter menu ask manager to enter the option by number and call the function

};

class Car

{

private:

    string brand;                 // Brand

    string model;                 // Model

    string color;                 // Color

    string VIN;                   // VIN

    int year;                     // Year

    int mileage;                  // Mileage

    float price\_per\_day;          // Price per day

    int deposit;                  // Deposit

    set<ScheduleForCar> schedule; // Schedule for car

public:

    Car(string brand, string model, string color, string VIN, int year, int mileage, int price\_per\_day, int deposit); // Constructor

    Car();                                                                                                            // Constructor

    bool addSchedule(ScheduleForCar schedule);

// Add schedule to the set schedule if the car is available for this schedule and return true if the car is available for this schedule

    void deleteSchedule(ScheduleForCar schedule);    // Delete schedule

    void showSchedule();          // Show schedule

    void showCar();                                 // Show car

    void updateMileage();

// Update mileage ask manager to enter the new mileage which larger then before and update the mileage

    void updatePrice();

// Update price ask manager to enter the new price and update the price

    void updateDeposit();

// Update deposit ask manager to enter the new deposit and update the deposit

};

class Client

{

private:

    string name;                     // Name

    string surname;                  // Surname

    string date\_of\_birth;            // Date of birth

    string passport\_number;          // Passport number

    string passport\_expiration\_date; // Passport expiration date

    string phone\_number;             // Phone number

    string email;                    // Email

    string address;                  // Address

    set<ScheduleForClient> schedule; // Schedule for customer

public:

    Client(string name, string surname, string date\_of\_birth, string passport\_number, string passport\_expiration\_date, string phone\_number, string email, string address); // Constructor

    Client();       // Constructor

    bool addSchedule(ScheduleForClient schedule);

// Add schedule to the set schedule if the car is available for this schedule and return true if the car is available for this schedule

    void deleteSchedule();

// Delete schedule: call showSchedule() and ask for the number of the schedule from the manager, call Car::deleteSchedule(ScheduleForClient schedule) and delete the schedule from the set schedule

    void showSchedule(); // Show schedule

    void showClient();   // Show client

};

class Schedule

{

private:

    int day\_from;   // Day from

    int month\_from; // Month from

    int year\_from;  // Year from

    int day\_to;     // Day to

    int month\_to;   // Month to

    int year\_to;    // Year to

public:

    bool operator<(Schedule &other);       // Operator <

    Schedule(int day\_from, int month\_from, int year\_from, int day\_to, int month\_to, int year\_to); // Constructor

    Schedule(); // Constructor

};

class ScheduleForCar : public Schedule

{

private:

    Client \*client; // Client

public:

    ScheduleForCar(int day\_from, int month\_from, int year\_from, int day\_to, int month\_to, int year\_to, Client \*client); // Constructor

    ScheduleForCar();                   // Constructor

    void showScheduleForCar();          // Show schedule for car

};

class ScheduleForClient : public Schedule

{

private:

    Car \*car; // Car

public:

    ScheduleForClient(int day\_from, int month\_from, int year\_from, int day\_to, int month\_to, int year\_to, Car \*car); // Constructor

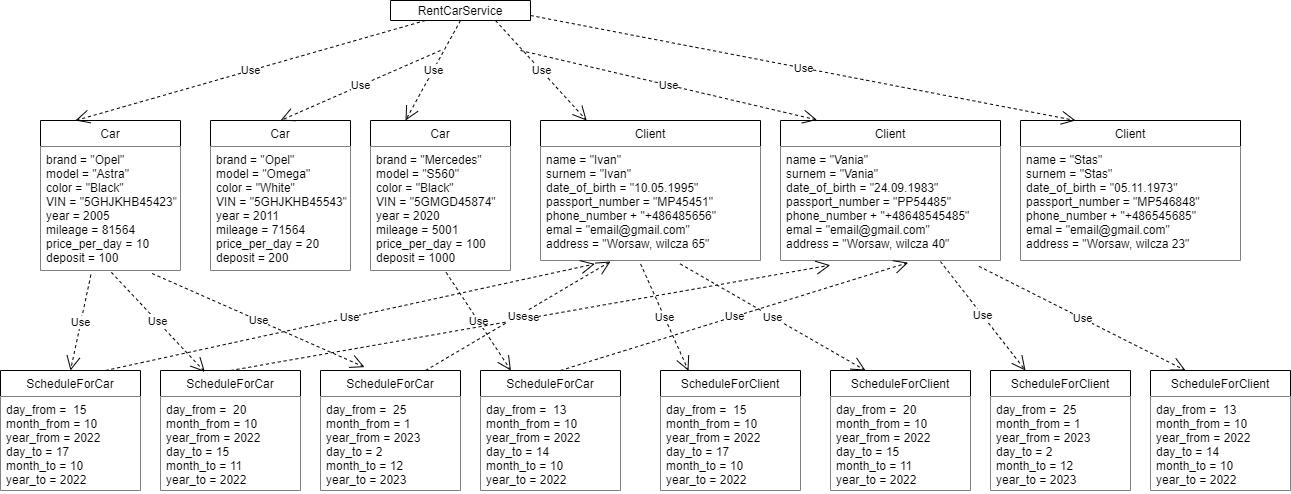
    ScheduleForClient();         // Constructor

    void showScheduleForCustomer(); // Show schedule for customer

};

## **Functional test cases**

* + - 1. **Try to enter number negative or higher thin numbers in menu in RentCarService::enterMenu(), RentCarService::Car \*chooseCar(), RentCarService:: Client \*chooseClient() and Client::deleteSchedule().**
      2. **Try to choose schedule which note available for car(R2) or client(R1) in method RentCarService::rentCar().**
      3. **Try to enter mileage lower than before in method Car::updateMileage().**
      4. **Try to enter incorrect with negative number or larger than can be in day, month and year date\_of\_birth in RentCarService::addClient().**
      5. **Try to enter year more than 2022 in RentCarService::addCar();**

****