EOOP Preliminary Project

"CAR RENTAL COMPANY"

Tymon Żarski | 22.04.2020

Table of Contest

1. “The story” – Description of the project

Brief Introduction

The main goal of this project is to develop an C++ application which is going to store all necessary data for the Car Renal Company. Program will by equipped with a simple user interface allowing the company manager to have a look (and export it to “.xlsx” file) on all important statistics to keep hand on plus all the time, menage employees, check cars rental status and whole needed history. In addition application will be storing data in the “.txt” file which could be saved and loaded at any point.

Concept of user interface

The user interface fill work like a tree. First of all user will choose the operation (for example change car technical review) then will search for a list member (usage of find functions) to at the end insert new data (after correct validation)

Limits and restrictions

The biggest problem of the application is users action, so to reduce that factor manipulating rental history will be forbidden and removing will be protected by typing and sentence to confirm action. Other type of error is invalid data type or file type to solve those issues there will be special set of methods to validate input data for example remove extra spaces or VIN pattern to protect user from inserting senseless data. Last but not lest we need to take care of memory allocating to prevent all segmentation faults and core dumps for prevent that application will be based on memory map (that may change an new uploaded version will be at GitHub repository https://github.com/tymzar/EOOP-Car-Rental-Company)

Class explanation

CarRentalCompany - The core of the application (contains rest of the classes), it stores all the data. It handless every operation on performed on lists and calculations/inserts of the data.

Employee – alone branch if the structure created to keep all workers and eye on their work (notifying if car fueling is needed).

Customer – a class which holds all the data needed to be a customer and all his history with the company.

Car – holds all data about the car, every car is unique (differentiated by VIN number) also contains important information about the rental process for example who last rented it or its current status

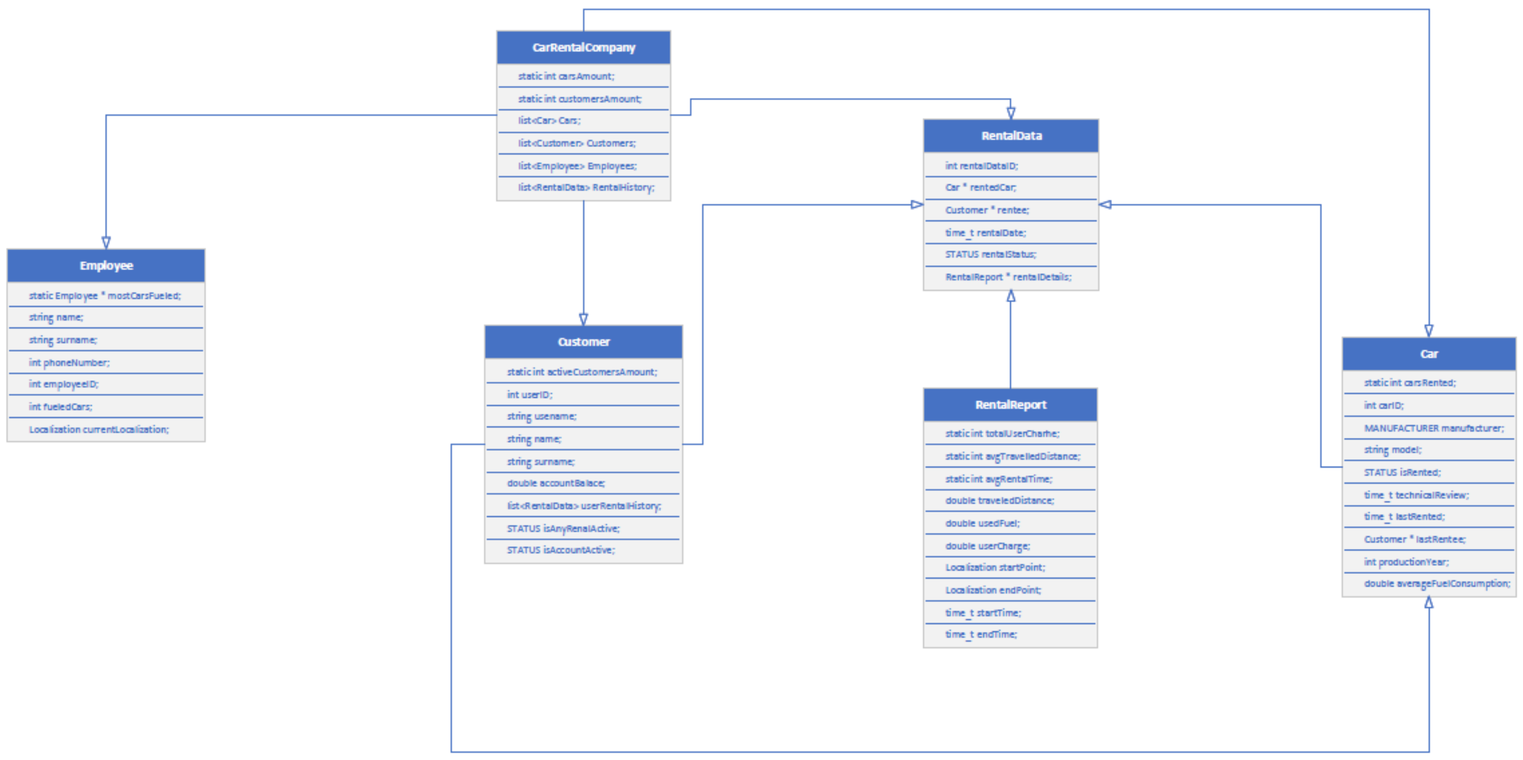
RentalData – head of the report rental data it hold most important information about rental.

RentalReport – biggest data holder in the project. Contains all details about the rental which are used to create statistics helping develop company in wright direction.

2. Memory map

Pdf format of the map can be found in the GitHub repository in docs directory

https://github.com/tymzar/EOOP-Car-Rental-Company



3. Declaration of the classes – preliminary c++ code

All the C++ classes can by found in the GitHub repository
https://github.com/tymzar/EOOP-Car-Rental-Company

Class CarRentalCompany

Members	
int carsAmount;	Number of cars available in the fleet
int customersAmount;	Number of customers in the DataBase
list<Car> Cars;	List of all cars
list<Customer> Customers;	List of all customers
list<Employee> Employees;	List of all employees
list<RentalData> RentalHistory;	List of ever rental
Public Methods	
CarRentalCompany([…]);	Constructor and Deconstructor
~CarRentalCompany();	
void addCar(Car& car);	Addition of unique car to the list
void addCustomer(Customer& customer);	Addition of unique customer to the list
void addEmployee(Employee& employee);	Addition of unique employee to the list
void addRentalData(RentalData& rentalData);	Addition of unique rentalData to the list
void removeCar(int VINnumber);	Removal of car by VINnumber
void removeCustomer(int customerDBID);	Removal of customer by DataBaseID
void removeEmployee(int employeeDBID);	Removal of employee by DataBaseID
Car* getCar(int carDBID);	Method returns (goes to) wanted car
Customer* getCustomer(int customerDBID);	Method returns (goes to) wanted customer
Employee* getEmployee(int employeeDBID);	Method returns (goes to) wanted employee
RentalData* getRentalData(int rentalDataDBID);	Method returns (goes to) wanted rental data
void updateCar(int carDBID);	Method allowing user to go through all car data and update it.
void updateCustomer(int customerDBID);	Method allowing customer to go through all car data and update it.
void updateEmployee(int employeeDBID);	Method allowing employee to go through all car data and update it.
void updateRentalData(int rentalDataDBID);	Method allowing rentalData to go through all car data and update it.

void addData(DATA_TYPE type);	Method (middle-man) passing options as data specified as **DATA_TYPE type** to be added
void removeData(DATA_TYPE type, int memberDBID);	Method (middle-man) passing options as data specified as **DATA_TYPE type** to be removed
void updateData(DATA_TYPE type);	Method (middle-man) passing options as data specified as **DATA_TYPE type** to be updated
void saveData(string path);	Method saving all data to the file
void loadData(DATA_TYPE type, string path);	Method loading data form file (all or just one class)
void getData(DATA_TYPE type, int memberDBID);	Method that returns searched list member to run action
void exportStatisticsToXLSX();	Method creating xlsx file with all company statistics
void getStatistics();	Method printing all statistics
int returnNumberOfCars();	Returns number of all cars available in the fleet
int returnNumberOfCustomers();	Number of customers in the DataBase
void outData(ostream& out);	Return all data to ofstream variable
void printData(DATA_TYPE type);	Method (middle-man) passing selected **DATA_TYPE type to** ofstream variable
Operators	
<<	Return all data to ofstream variable

Class Car

Members	
static int carsRented;	Number of cars rented
int carID;	Unique Car ID
MANUFACTURER manufacturer;	Manufacturer of the car
string model;	Car model
string VINnumber;	Car VIN number
STATUS isRented;	Car rental statis
time_t technicalReview;	Next car technical review
time_t lastRented;	Last time car was rented
Customer * lastRentee;	Last car rentee
int productionYear;	Car production year
double averageFuelConsumption;	Car average fuel consumption
Public Methods	
Car([…]);	Constructor and Deconstructor
~Car();	
int getCarID();	Returns car ID
MANUFACTURER getManufacturer();	Returns car manufacturer
string getModel();	Returns car model
STATUS getIsRented();	Returns car rental status
time_t getTechnicalReview();	Returns car next technical review
time_t getLastRented();	Returns car last time car was rented
Customer * getLastRentee();	Returns car last rentee
int getProductionYear();	Returns car production year
double getAverageFuelConsumption();	Returns car average fuel consumption
double getCarRange();	Returns car car range
void updateCarID(int x);	Updates car ID
void updateManufacturer(MANUFACTURER x);	Updates car manufacturer
void updateModel(string x);	Updates car model
void updateTechnicalReview(time_t x);	Updates car rental status
void updateLastRented(time_t x);	Updates car next technical review
void updateLastRentee(Customer x);	Updates car last time car was rented
void updateProductionYear(int x);	Updates car last rentee
void updateAverageFuelConsumption(double x);	Updates car production year
void toggleIsRented();	Inverts car rental status
void printData();	Return all data to ofstream variable
Operators	
<<	Return all data to ofstream variable

Class Customer

Members	
static int activeCustomersAmount;	All coustomers with active accuonts
int userID;	Customer DataBase ID
string usename;	Customer username
string name;	Customer name
string surname;	Customer surname
double accountBalace;	Customer account balance
list<RentalData> userRentalHistory;	Customer renal history
STATUS isAnyRenalActive;	Customer rental status
Public Methods	
Customer();	Constructor and Deconstructor
~Customer();	
int getUserID();	Returns customer ID
string getUsername();	Returns customer username
string getName();	Returns customer name
string getSurname();	Returns customer surname
double getAccountBalace();	Returns customer account balance
list<RentalData> getUserRentalHistory();	Returns customer rental history
STATUS getIsAnyRenalActive();	Returns customer rental status
STATUS getIsAccountActive();	Returns customer account status
void updateUserID(int x);	Updates customer ID
void updateUsername(string x);	Updates customer Username
void updateName(string x);	Updates customer Name
void updateSurname(string x);	Updates customer Surname
void updateAccountBalace(double x);	Updates customer Account Balance
void updateUserRentalHistory(RentalData& x);	Updates customer rental history
void toggleIsAnyRenalActive();	Inverts customer rental status
void toggleIsAccountActive();	Inverts customer account status
void addUserRentalHistory(RentalData* rentalData);	Addition to customer rental history
void printAllCustomerData();	Method prints all customer data (only last record from history)
void printUserRentalHistory();	Method prints customer history.
void printData(OUT_CUSTOMER type);	Method (middle-man) passing selected **OUT_CUSTOMER type** to ofstream variable
void outData(ostream& out);	Prints all car data
Operators	
<<	Return all data to ofstream variable

Class RentalData

Members	
int rentalDataID;	Data of rental record
Car * rentedCar;	Pointer to rented car
Customer * rentee;	Pointer to rentee (customer)
STATUS rentalStatus;	Customer surname
RentalReport * rentalDetails;	Customer account balance
Public Methods	
RentalData();	Constructor and Deconstructor
~RentalData();	
Car * getRentedCar();	Returns pointer to rented car
Customer * getRentee();	Returns pointer to rentee (customer)
STATUS getRentalStatus();	Returns rental status
RentalReport * getRentalDetails();	Returns rental record
void updateRentedCar(Car * x);	Updates pointer to rented car
void updateRentee(Customer* x);	Updates pointer to rentee (customer)
void updateRentalDetails(RentalReport *);	Updates rental record
void toggleRentalStatus();	Inverts rental status
void notifyNearestEmployee();	Notifies nearest employee to fuel the car (triggered after rental end)
void notifyCustomer();	Notifies nearest customer about insufficient balance to rent or after renting a car
Operators	
<<	Return all data to ofstream variable

Class RentalReport

Members	
static int totalUserCharge;	Full company profit
static int avgTravelledDistance;	Average rental distance distance
static int avgRentalTime;	Average rental time
double travelledDistance;	Traveled distance
double userCharge;	Customer renal charge
double usedFuel;	Used fuel during the rental
Localization startPoint;	Car localization at the start of the rental
Localization endPoint;	Car localization at the end of the rental
time_t startTime;	Rental start time
time_t endTime;	Rental end time
Public Methods	
RentalReport();	Constructor and Deconstructor
~RentalReport();	
double getTravelledDistance();	Returns travelled distance during rental
double getUsedFuel();	Returns used fuel during rental
double getUserCharge();	Returns customer rental charge
Localization getStartPoint();	Returns car rental start point
Localization getEndPoint();	Returns car rental start end
time_t getStartTime();	Returns rental start time
time_t getEndTime();	Returns rental end time
void updateTravelledDistance(double x);	Updates travelled distance during rental
void updateUsedFuel(double x);	Updates used fuel during rental
void updateUserCharge(double x);	Updates customer rental charge
void updateStartPoint(Localization x);	Updates rental start time
void updateEndPoint(Localization x);	Updates rental end time
void updateStartTime(time_t x);	Updates rental start time
void updateEndTime(time_t x);	Updates rental end time
void avgTravelledDistance(time_t x);	Updates average rental distance
Operators	
<<	Return all data to ofstream variable

Employee

Members	
static Employee * mostCarsFueled;	Pointer to employee with most cards filed
string name;	Employee name
string surname;	Employee surname
int phoneNumber;	Employee phone number
int employeeID;	Employee DataBaseID
int fueledCars;	Employee all cars fueled
Localization currentLocalization;	Employee current localization
Public Methods	
Employee();	Constructor and Deconstructor
~Employee();	
string getName();	Returns employee name
string getSurname();	Returns employee surname
int getPhoneNumber();	Returns employee phone number
int getEmployeeID();	Returns employee DataBaseID
int getFueledCars();	Returns employee all cars fueled
Localization getCurrentLocalization();	Returns employee current localization
void updateName(string x);	Updates employee name
void updateSurname(string x);	Updates employee surname
void updatePhoneNumber(int x);	Updates employee phone number
void updateEmployeeID(int x);	Updates employee DataBaseID
void updateFueledCars(int x);	Updates employee all cars fueled
void updateCurrentLocalization(Localization x);	Updates employee current localization
Operators	
<<	Return all data to ofstream variable

Utilities.hpp

Enum	
MANUFACTURER	AUDI, BMW, SKODA, TOYOTA, FORD,HONDA, VOLKSWAGEN
STATUS	UNACTIVE, ACTIVE
DATA_TYPE	ALL, CAR, CUSTOMER, EMPLOYEE, RENTAL_HISTORY
OUT_CUSTOMER	ALL, USER_RENTAL_HISTORY
int employeeID;	Employee DataBaseID
int fueledCars;	Employee all cars fueled
Struct	
Localization	double longitude;double latitude;

4. Testing

A. AIM OF THE TESTS

The biggest aim of the test is to present cases of incorrect an correct usage of the code, and get along with the structure of the project.

B. EXAMPLES

Below in the table I will show few examples of the test which will be later on implemented. (It's only a small portion of the tests, because concept in every single one is similar)

Classes	Operation	Data	Correct Result	Result
CarRentalCompany.hpp Employee.hpp	Add new employee	Employee already exists	Adding employee to list	Emit error ang do to the main menu
		New Employee		Add employee to list
CarRentalCompany.hpp Employee.hpp Car.hpp Customer.hpp RentalData.hpp RentalReport.hpp	Load data from file	Wrong file type	Importing all the data	Display correct error and ask again for file path
		Wrong file structure		
		File do not exists		
		Correct file		Import all data and show main menu
CarRentalCompany.hpp Customer.hpp RentalData.hpp RentalReport.hpp	Add new rental	Inserting wrong input type	Adding new rental	Display error and ask for correct input type
		Correct rental data		Add data to list and go to main menu
CarRentalCompany.hpp Car.hpp Customer.hpp RentalData.hpp RentalReport.hpp	Toggle rental status	Rental status is INACTIVE	Error every rental can by ACTIVE only once	Error displayed rental has ended and cannot be resumed
		Rental status in ACTIVE	Change status to INACTIVE	Successfully change status and go bac to main menu