МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Лабораторная работа

по программированию №7

“Обработка собственных событий”

Студент \_\_\_\_\_\_\_\_\_\_\_\_ Станиславчук С. М.

(подпись, дата)

Группа АС-21-1

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_ Ведищев В. В.

(подпись, дата)

Липецк 2022 г.

Содержание

2. Задание

3. Цель

4. Код программы

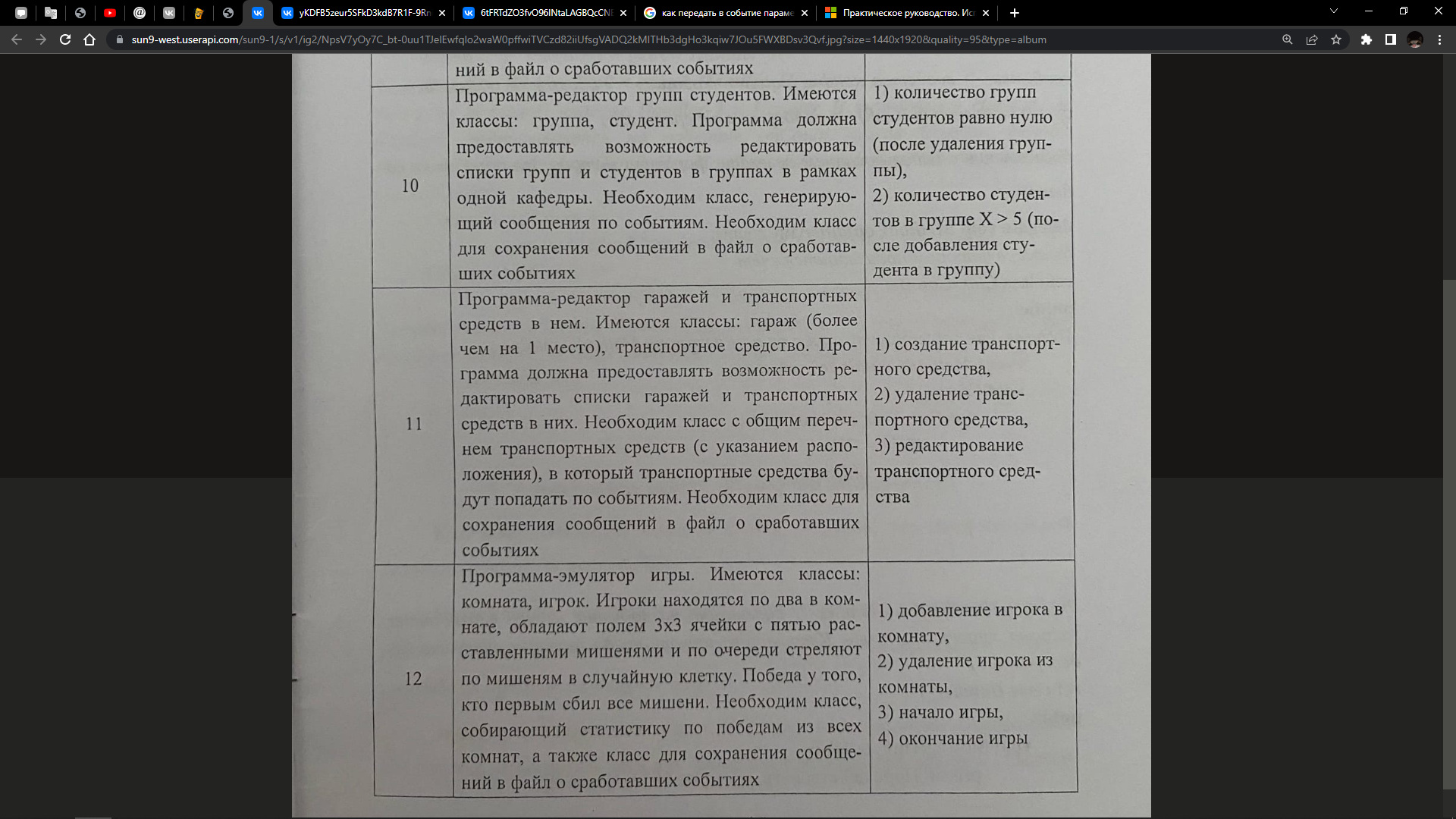
5. Пример выполнения

6. Вывод

2. Задание:

Реализовать на языке C++ программу, требования для которой представлены на изображении, с использованием механизма событий.

Вариант Задание События



3. Цель работы:

Научиться создавать и использовать собственные события для возможности отслеживания другими классами наступления отдельных условий в текущем классе.

4. Код программы:

using namespace System;

using namespace System::IO;

using namespace System::Text;

public delegate void AddNewVehicle();

public delegate void DeleteVehicle(int);

public delegate void EditVehicle(int, String^, String^, int);

public delegate void MethodReport(int, int);

int GaragesSize = 0;

int GarageID = 0;

ref class Vehicle

{

public:

String^ brand;

String^ engine;

int price, id;

void SetVehicles()

{

Console::WriteLine("Input vehicle specifications (brand, engine, price): ");

brand = Convert::ToString(Console::ReadLine());

engine = Convert::ToString(Console::ReadLine());

price = Convert::ToUInt32(Console::ReadLine());

}

};

template <class T>

ref class Garage

{

public:

String^ GarageName;

cli::array < T^ >^ Vehicles;

event AddNewVehicle^ eAddNewVehicle;

event DeleteVehicle^ eDeleteVehicle;

event EditVehicle^ eEditVehicle;

event MethodReport^ eReportM;

int size;

Vehicle v;

Garage()

{

Console::WriteLine("Input garage name: ");

GarageName = Convert::ToString(Console::ReadLine());

size = 0;

Vehicles = gcnew cli::array < T^ >(size);

}

void GetGarage()

{

for (int i = 0; i < size; i++)

{

Console::WriteLine("Garage name\t" + GarageName + "\tVehicle Id:\t" + Vehicles[i]->id + "\tBrand:\t" + Vehicles[i]->brand);

}

}

void AddNewVehicle()

{

size += 1;

cli::array< T^ >^ temp = gcnew cli::array < T^ >(size);

for (int i = 0; i < size; i++)

{

if (i == size - 1)

{

temp[size - 1] = gcnew T;

temp[size - 1]->id = (size - 1);

temp[size - 1]->SetVehicles();

}

else

{

temp[i] = gcnew T;

temp[i] = Vehicles[i];

}

}

delete Vehicles;

Vehicles = gcnew array < T^ >(size);

for (int i = 0; i < size; i++)

{

Vehicles[i] = gcnew T;

Vehicles[i] = temp[i];

}

delete temp;

GaragesSize++;

GarageID++;

eReportM(GarageID, size);

}

void DeleteVehicle(int id)

{

size -= 1;

array< T^ >^ temp = gcnew array < T^ >(size);

for (int i = 0; i < size; i++)

{

if (i < id)

{

temp[i] = gcnew T;

temp[i] = Vehicles[i];

}

else if (i > id)

{

temp[i] = gcnew T;

temp[i] = Vehicles[i];

Vehicles[i]->id -= 1;

}

else if (id + 1 == size)

{

temp[i] = gcnew T;

temp[i] = Vehicles[i + 1];

Vehicles[i + 1]->id -= 1;

}

}

delete Vehicles;

Vehicles = gcnew array < T^ >(size);

for (int i = 0; i < size; i++)

{

Vehicles[i] = gcnew T;

Vehicles[i] = temp[i];

}

delete temp;

GaragesSize--;

eDeleteVehicle(id);

}

void EditVehicle(int \_id, String^ \_brand, String^ \_engine, int \_price)

{

for (int i = 0; i < size; i++)

{

if (\_id == Vehicles[i]->id) {

Vehicles[i]->brand = \_brand;

Vehicles[i]->engine = \_engine;

Vehicles[i]->price = \_price;

}

}

}

};

template <class T>

ref class Events

{

public:

event AddNewVehicle^ eAddNewVehicle;

event DeleteVehicle^ eDeleteVehicle;

event EditVehicle^ eEditVehicle;

event MethodReport^ ReportM;

String^ GarageName;

String^ VehicleName;

array < T^ >^ Vehicles;

int size;

void AddNewVehicle()

{

size += 1;

array< T^ >^ temp = gcnew array < T^ >(size);

for (int i = 0; i < size; i++)

{

if (i == size - 1)

{

temp[size - 1] = gcnew T;

temp[size - 1]->id = (size - 1);

temp[size - 1]->SetVehicles();

}

else

{

temp[i] = gcnew T;

temp[i] = Vehicles[i];

}

}

delete Vehicles;

Vehicles = gcnew cli::array < T^ >(size);

for (int i = 0; i < size; i++)

{

Vehicles[i] = gcnew T;

Vehicles[i] = temp[i];

}

delete temp;

GaragesSize++;

GarageID++;

ReportM(GarageID, size);

}

void DeleteVehicle(int id)

{

size -= 1;

array< T^ >^ temp = gcnew array < T^ >(size);

for (int i = 0; i < size; i++)

{

if (i < id)

{

temp[i] = gcnew T;

temp[i] = Vehicles[i];

}

else if (i > id)

{

temp[i] = gcnew T;

temp[i] = Vehicles[i];

Vehicles[i]->id -= 1;

}

else if (id + 1 == size)

{

temp[i] = gcnew T;

temp[i] = Vehicles[i + 1];

Vehicles[i + 1]->id -= 1;

}

}

delete Vehicles;

Vehicles = gcnew cli::array < T^ >(size);

for (int i = 0; i < size; i++)

{

Vehicles[i] = gcnew T;

Vehicles[i] = temp[i];

}

delete temp;

GaragesSize--;

eDeleteVehicle(id);

ReportM(GarageID, size);

}

void EditVehicle(int \_id, String^ \_brand, String^ \_engine, int \_price)

{

for (int i = 0; i < size; i++)

{

if (\_id == Vehicles[i]->id) {

Vehicles[i]->brand = \_brand;

Vehicles[i]->engine = \_engine;

Vehicles[i]->price = \_price;

}

}

eEditVehicle(\_id, \_brand, \_engine, \_price);

ReportM(GarageID, size);

}

};

ref class Report

{

public:

Report()

{

IO::File::WriteAllText("Report.txt", "Event report\n");

}

void AddNewVehicle(int Garage\_id, int size)

{

String^ Message = "Vehicle in garage# " + Garage\_id + " added. Total in the garage " + size + " vehicles\n";

IO::File::AppendAllText("Report.txt", Message);

}

void DeleteVehicle(int Garage\_id, int size)

{

String^ Message = "Vehicle in garage# " + Garage\_id + "deleted. Total in the garage " + size + " vehicles\n";

IO::File::AppendAllText("Report.txt", Message);

}

void EditVehicle(int Garage\_id, int size)

{

String^ Message = "Vehicle in garage# " + Garage\_id + "edited. \n";

IO::File::AppendAllText("Report.txt", Message);

}

};

int main()

{

int amount, choice;

Console::OutputEncoding = Encoding::UTF8;

Console::WriteLine("Input amount of garages: ");

amount = Convert::ToUInt32(Console::ReadLine());

array<Garage<Vehicle>^>^ GARAGES = gcnew array<Garage<Vehicle>^>(amount);

for (int i = 0; i < amount; i++)

{

GARAGES[i] = gcnew Garage<Vehicle>;

}

Events<Vehicle>^ eMethod = gcnew Events<Vehicle>();

Report^ FILE = gcnew Report();

while (true)

{

Console::Clear();

for (int i = 0; i < amount; i++)

{

GARAGES[i]->GetGarage();

}

Console::WriteLine("Select an option:\n1. Create vehicle\n2. Delete vehicle\n3. Edit vehicle specs\n0. Exit");

choice = Convert::ToUInt32(Console::ReadLine());

switch (choice)

{

case 1:

{

String^ name;

Console::WriteLine("Input name of garage to add a vehicle: ");

name = Convert::ToString(Console::ReadLine());

for (int i = 0; i < amount; i++)

{

if (GARAGES[i]->GarageName == name)

{

GARAGES[i]->eAddNewVehicle += gcnew AddNewVehicle(eMethod, &Events<Vehicle>::AddNewVehicle);

GARAGES[i]->eReportM += gcnew MethodReport(FILE, &Report::AddNewVehicle);

GARAGES[i]->AddNewVehicle();

Console::ReadLine();

}

}

break;

}

case 2:

{

String^ name;

Console::WriteLine("Input name of garage for deleting a vehicle: ");

name = Convert::ToString(Console::ReadLine());

for (int i = 0; i < amount; i++)

{

if (GARAGES[i]->GarageName == name)

{

int id;

Console::WriteLine("Input vehicle id to delete: ");

id = Convert::ToUInt32(Console::ReadLine());

GARAGES[i]->eDeleteVehicle += gcnew DeleteVehicle(eMethod, &Events<Vehicle>::DeleteVehicle);

GARAGES[i]->eReportM += gcnew MethodReport(FILE, &Report::DeleteVehicle);

GARAGES[i]->DeleteVehicle(id);

}

}

break;

}

case 3:

{

String^ name;

Console::WriteLine("Input name of garage for editing a vehicle: ");

name = Convert::ToString(Console::ReadLine());

for (int i = 0; i < amount; i++)

{

if (GARAGES[i]->GarageName == name)

{

int id, price;

String^ brand;

String^ engine;

Console::WriteLine("Input id of vehicle to edit: ");

id = Convert::ToUInt32(Console::ReadLine());

Console::WriteLine("Input new specifications (brand, engine, price): ");

brand = Convert::ToString(Console::ReadLine());

engine = Convert::ToString(Console::ReadLine());

price = Convert::ToUInt32(Console::ReadLine());

GARAGES[i]->eEditVehicle += gcnew EditVehicle(eMethod, &Events<Vehicle>::EditVehicle);

GARAGES[i]->eReportM += gcnew MethodReport(FILE, &Report::EditVehicle);

GARAGES[i]->EditVehicle(id, brand, engine, price);

}

}

break;

}

case 0: {

return 0;

}

default:

Console::WriteLine("Wrong choice!");

Console::Clear();

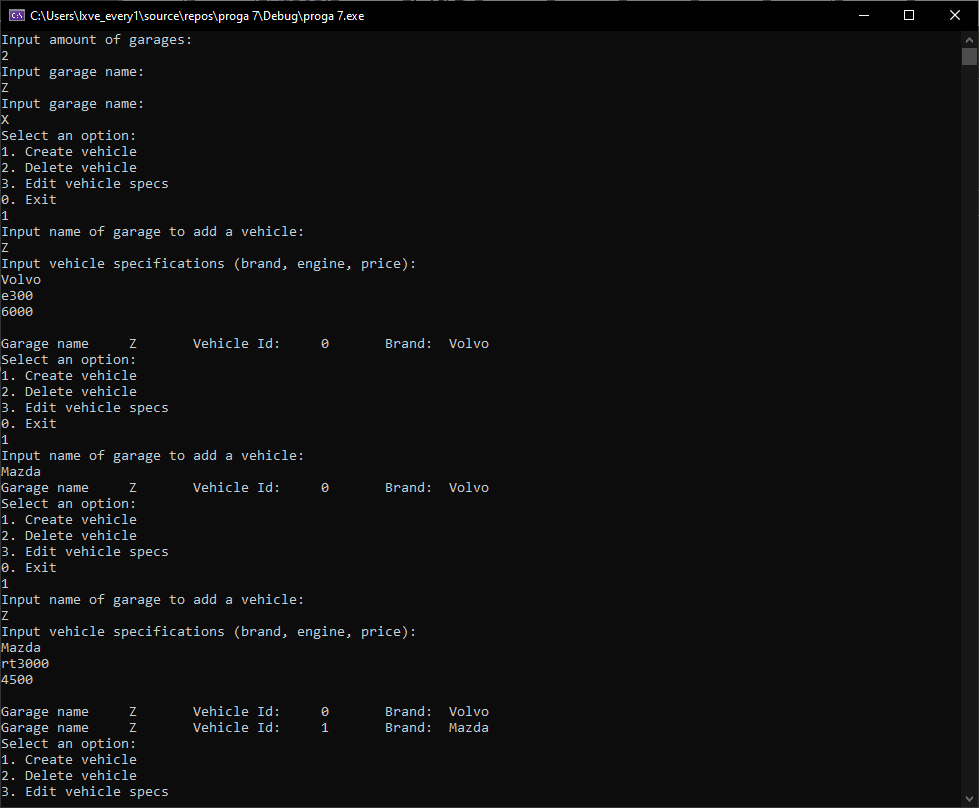
break;

}

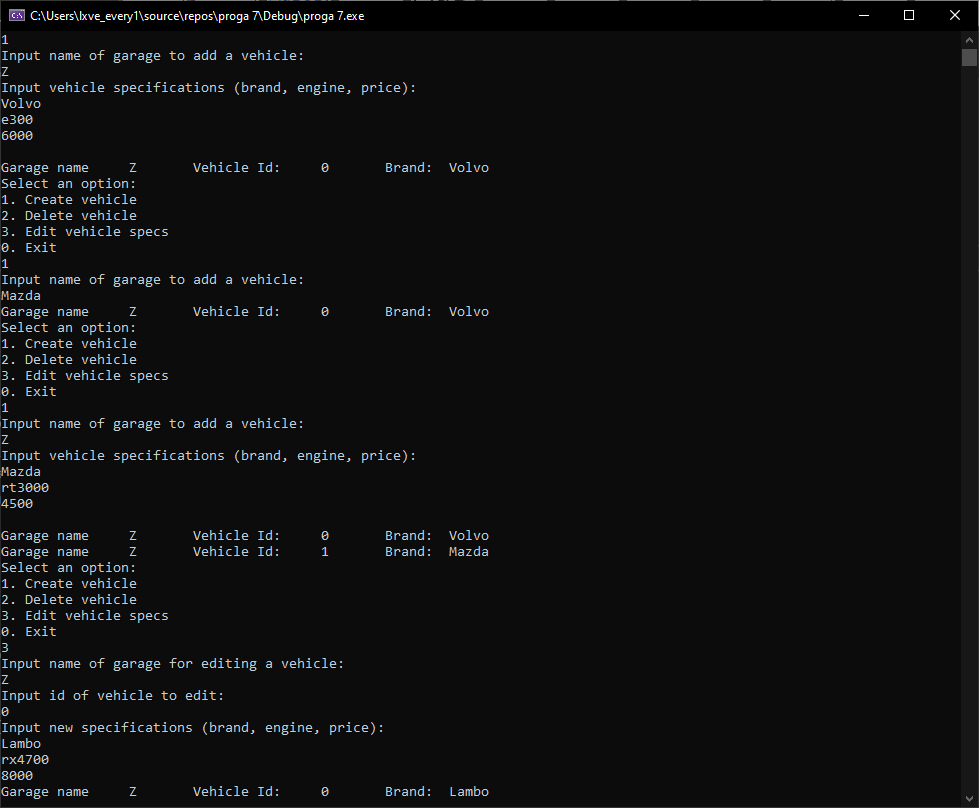
}

}

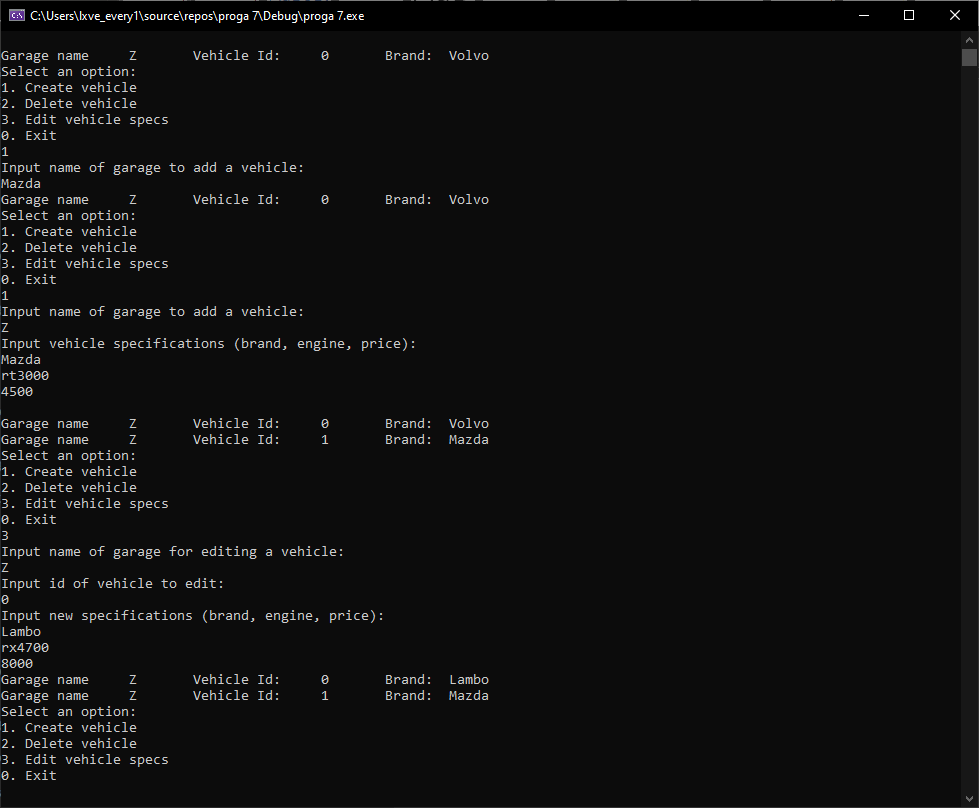
5. Пример выполнения



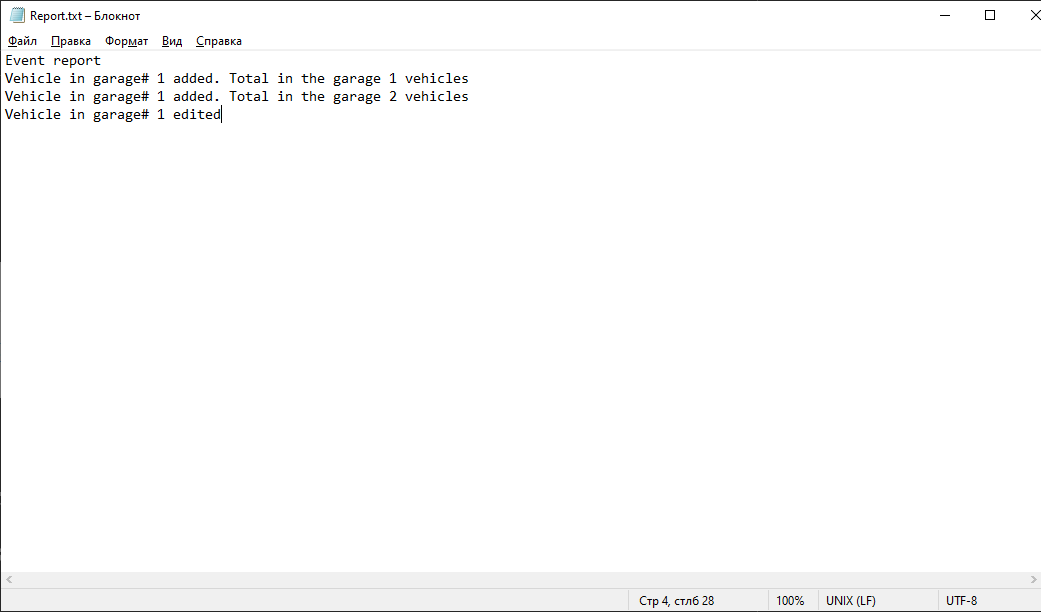
Вводим количество и имена гаражей



Создаем 2 ТС и добавляем их в гараж, после чего редактируем одно из них.



Изменение файла Report.txt:



Вывод:

Научился создавать и использовать собственные события для возможности отслеживания другими классами наступления отдельных условий в текущем классе.