МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информационные технологии»

Лабораторная работа №1

**«Классы и объекты, инкапсуляция и наследование»**

по дисциплине

Информационные технологии и программирование

Выполнил: студент гр. БЭИ2202

Кулешов А. С.

Вариант 16

Проверил: Халабия Р.Ф.

Москва, 2024 г

1. Цели и задачи

Овладение навыками проектирования простейших классов и создания объектов класса; освоение принципов инкапсуляции и наследования.

Реализовать 4 класса: грузовик, автомобиль, легковое авто, транспорт. Добавить конструкторы, методы и атрибуты

1. Схема иерархии классов

Схема иерархии классов представлена на рисунке 1

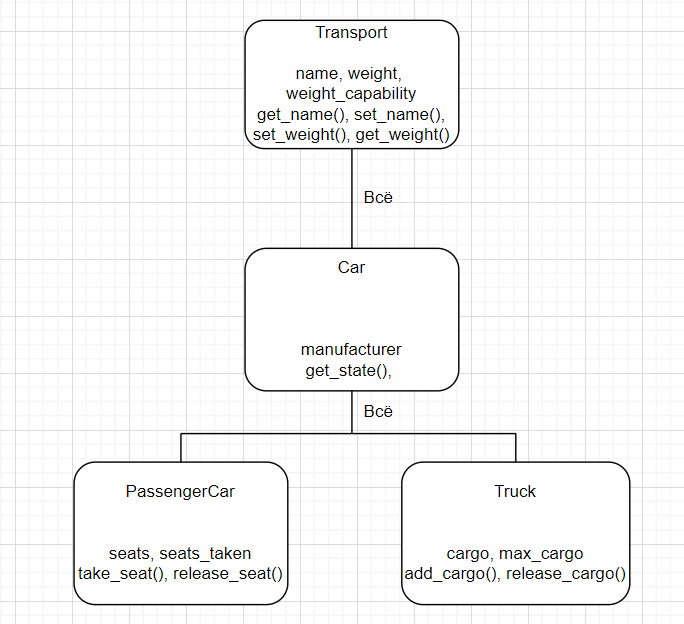


Рисунок 1 – Схема иерархии классов

1. Листинг программы

#include <iostream>

**using** **namespace** std;

/\*

Грузовик, автомобиль, лёгковое авто, транспорт

\*/

**class** **Transport**{

**protected:**

string name;

**double** weight;

**double** weight\_capability;

**public:**

Transport(string name\_, **double** weight\_, **double** weight\_capability\_ ){

name = name\_;

weight = weight\_;

weight\_capability = weight\_capability\_;

}

string get\_name(){

**return** name;

}

**void** set\_name(string new\_name){

name = new\_name;

}

**void** set\_weight(**double** new\_weight){

weight = new\_weight;

}

**double** get\_weight(){

**return** weight;

}

};

**class** **Car** : **public** Transport{

**protected:**

string manufacturer;

**public:**

Car(string name\_, **double** weight\_, **double** weight\_capability\_, string manu\_) : Transport(name\_, weight\_, weight\_capability\_){

manufacturer = manu\_;

}

string get\_state(){

**double** ratio = weight\_capability / weight;

**if** (ratio < **1**) **return** "broken";

**if** (ratio < **1.5**) **return** "used";

**return** "new";

}

};

**class** **PassengerCar** : **public** Car{

**int** seats;

**int** seats\_taken = **0**;

**public:**

PassengerCar(string name\_, **double** weight\_, **double** weight\_capability\_, string manu\_, **int** seats\_) : Car(name\_, weight\_, weight\_capability\_, manu\_){

seats = seats\_;

}

**void** take\_seat(){

**if** (seats\_taken >= seats){

cout << "All seats were already taken" << endl;

**return**;

}

cout << "Added passenger " << seats\_taken << " -> " << ++seats\_taken << endl;

}

**void** release\_seat(){

**if** (seats\_taken <= **0**){

cout << "There are no passengers" << endl;

**return**;

}

cout << "Removed passenger " << seats\_taken << " -> " << --seats\_taken << endl;

}

};

**class** **Truck** : **public** Car{

**double** cargo = **0**, max\_cargo;

**public:**

Truck(string name\_, **double** weight\_, **double** weight\_capability\_, string manu\_, **double** max\_cargo\_) : Car(name\_, weight\_, weight\_capability\_, manu\_){

max\_cargo = max\_cargo\_;

}

**void** add\_cargo(**double** extra){

**if** (cargo + extra > max\_cargo){

cout << "Not enough space for new cargo" << endl;

**return**;

}

cout << "Added cargo " << cargo << " -> " << (cargo += extra) << endl;

}

**void** release\_cargo(**double** release){

**if** (cargo - release < **0**){

cout << "There is not enough cargo to be released" << endl;

**return**;

}

cout << "Removed cargo " << cargo << " -> " << (cargo -= release) << endl;

}

};

**int** **main**()

{

Truck JohnTruck("teslatruck", **5000**, **10000**, "tesla\_foundation", **2500**);

JohnTruck.add\_cargo(**1000000**);

JohnTruck.add\_cargo(**1000**);

JohnTruck.release\_cargo(**1000000**);

JohnTruck.release\_cargo(**100**);

PassengerCar MariaCar("teslacar", **1500**, **2500**, "tesla\_foundation", **4**);

MariaCar.release\_seat();

MariaCar.take\_seat();

MariaCar.take\_seat();

MariaCar.take\_seat();

MariaCar.take\_seat();

MariaCar.take\_seat();

MariaCar.take\_seat();

MariaCar.release\_seat();

MariaCar.release\_seat();

MariaCar.release\_seat();

MariaCar.release\_seat();

MariaCar.release\_seat();

MariaCar.release\_seat();

MariaCar.release\_seat();

MariaCar.release\_seat();

MariaCar.release\_seat();

/\*

? JT name

+ MC

e - exit

\*/

**while**(**1**){

**char** c;

cin >> c;

**if** (c == 'e'){

**break**;

}

**if** (c == '?'){

string name;

cin >> name;

Car\* ts;

**if** (name != "MC" && name != "JT"){

cout << "No such vehicle found" << endl;

**continue**;

}

**if** (name == "JT")

ts = &JohnTruck;

**if** (name == "MC")

ts = &MariaCar;

string val;

cin >> val;

**if** (val == "name")

cout << ts->get\_name() << endl;

**if** (val == "weight")

cout << ts->get\_weight();

}

**if** (c == '+'){

string name;

cin >> name;

Car\* ts;

**if** (name != "MC" && name != "JT"){

cout << "No such vehicle found" << endl;

**continue**;

}

**if** (name == "JT")

ts = &JohnTruck;

**if** (name == "MC")

ts = &MariaCar;

string val;

cin >> val;

**if** (val == "name"){

string var;

cin >> var;

ts->set\_name(var);

}

**if** (val == "weight"){

**double** var;

cin >> var;

ts->set\_weight(var);

}

}

}

**return** **0**;

}

1. Схемы алгоритмов

Схемы алгоритмов методов представлены на рисунках 2-5

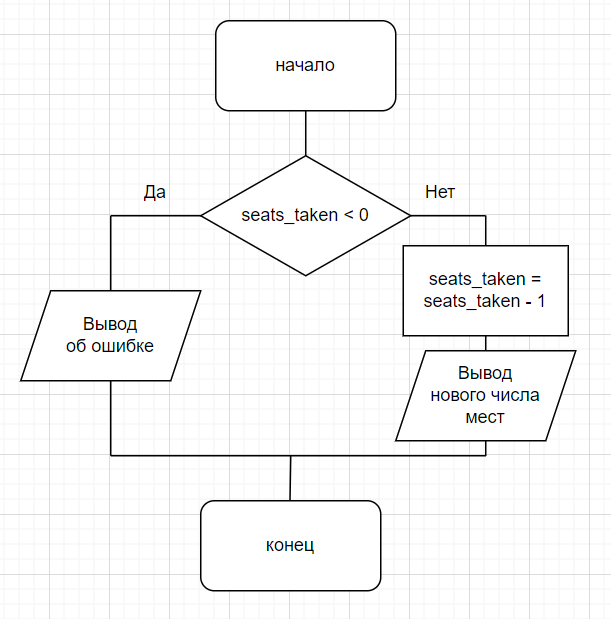


Рисунок 2 – Схема алгоритма метода release\_seat

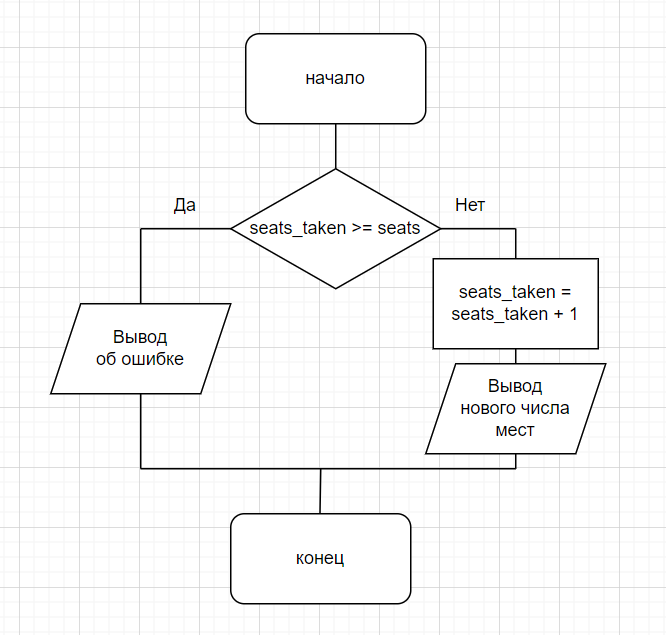


Рисунок 3 – Схема алгоритма метода take\_seat

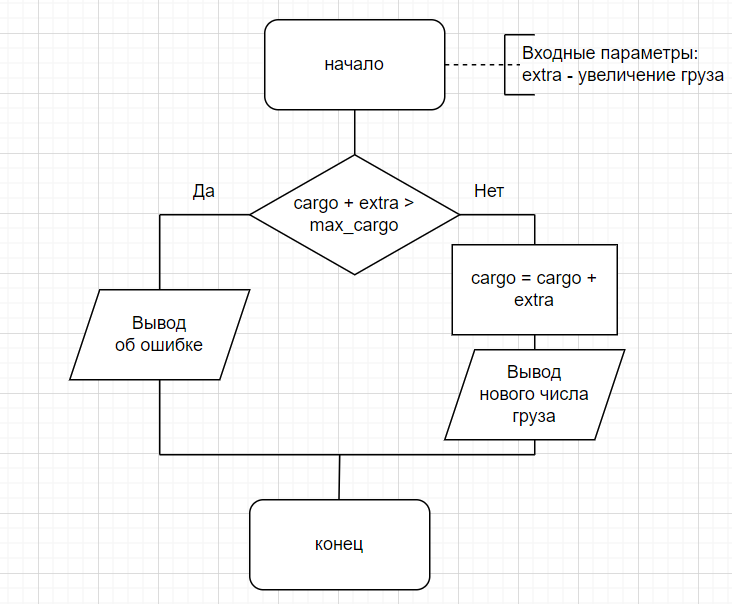


Рисунок 4 – Схема алгоритма метода add\_cargo

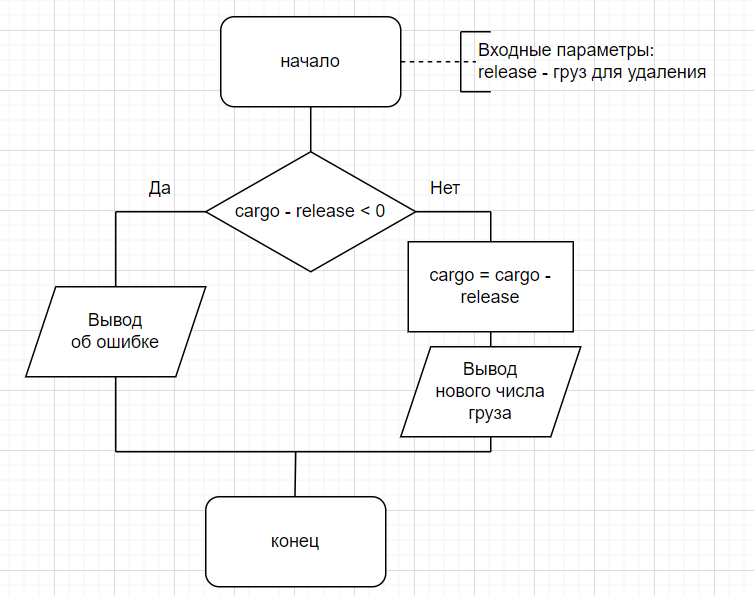


Рисунок 5 – Схема алгоритма метода release\_cargo

1. Контрольный тест. Результаты программы

Результат запуска программы на контрольных данных можно увидеть на рисунке 6.

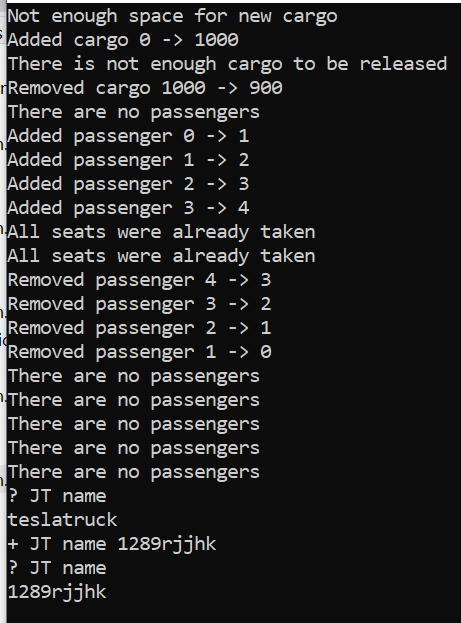


Рисунок 6 – результат запуска программы

1. Выводы по работе

В результате выполнения лабораторной работы мы смогли реализовать 4 класса, их методы, конструкторы, а также их иерархию наследования.