**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ**  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**  
**(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №5

Дисциплина: Обьектно-ориентированное программирование

по теме «Классы, виды отношений. Наследование»

Выполнил: ст. группы ВТ-22  
Фаракшин Никита Русланович

Проверил: Буханов Д.Г.

**Белгород 2019Цель работы**: Получение теоретических знаний в области разработки классов, получение практических навыков реализаций классов и отношений между ними.

**Вариант задания 4**

Задание 1.

Выполнить построение объектной модели следующей предметной области:

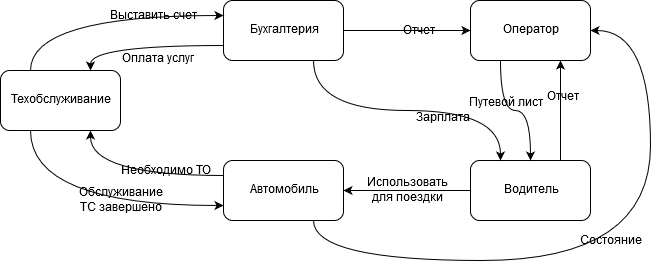
«Программа учета парка автотранспорта» и построить диаграмму классов.

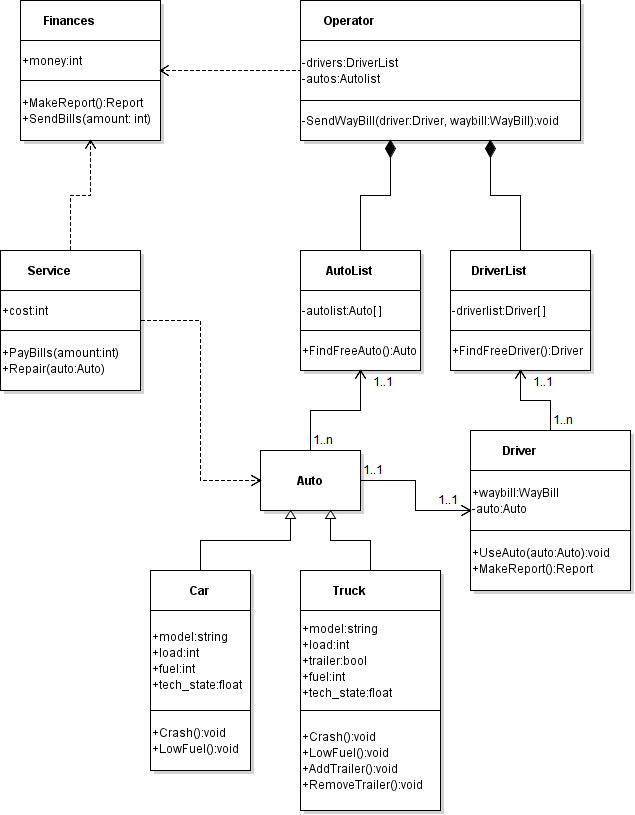
Задание 2.

1. Создать абстрактный класс Trans с методами, позволяющими вывести на экран информацию о транспортном средстве, а также определить грузоподъёмность транспортного средства.

2. Создать производные классы: Легковая\_машина (марка, номер, скорость, грузоподъёмность), Мотоцикл (марка, номер, скорость, грузоподъёмность, наличие коляски, при этом если коляска отсутствует, то грузоподъемность равна 0), Грузовик (марка, номер, скорость, грузоподъёмность, наличие прицепа, при этом если есть прицеп, то грузоподъёмность увеличивается в два раза) со своими методами вывода информации на экран, и определения грузоподъёмности.

3. Создать базу (массив) из n машин, вывести полную информацию из базы на экран, а также организовать поиск машин, удовлетворяющих требованиям грузоподъемности.





#include <iostream>

#include <string>

#include <list>

class Trans{

protected:

std::string model;

std::string reg\_no;

int speed;

int load;

public:

Trans(std::string, std::string, int, int);

virtual void print\_info();

virtual int get\_load\_capacity() = 0;

};

class Car : public Trans{

public:

Car(std::string, std::string, int, int);

void print\_info();

int get\_load\_capacity();

};

class Motorcycle : public Trans{

private:

bool has\_sidecar;

public:

Motorcycle(std::string, std::string, int, int, bool);

void print\_info();

int get\_load\_capacity();

};

class Truck : public Trans{

private:

bool has\_trailer;

public:

Truck(std::string, std::string, int, int, bool);

void print\_info();

int get\_load\_capacity();

};

Trans::Trans(std::string model, std::string reg\_no, int speed, int load) :

model(model),

reg\_no(reg\_no),

speed(speed),

load(load){};

void Trans::print\_info(){

std::cout << "Марка: " << model << std::endl;

std::cout << "Номер: " << reg\_no << std::endl;

std::cout << "Максимальная скорость" << speed << std::endl;

std::cout << "Грузоподъемность: " << this->get\_load\_capacity() << std::endl;

}

Car::Car(std::string model, std::string reg\_no, int speed, int load):

Trans(model, reg\_no, speed, load){};

void Car::print\_info(){

std::cout << "Легковая машина" << std::endl;;

Trans::print\_info();

}

int Car::get\_load\_capacity(){

return load;

}

Motorcycle::Motorcycle(std::string model, std::string reg\_no, int speed, int load, bool has\_sidecar):

Trans(model, reg\_no, speed, load),

has\_sidecar(has\_sidecar){};

void Motorcycle::print\_info(){

std::cout << "Мотоцикл" << std::endl;

Trans::print\_info();

std::cout << (has\_sidecar ? "Имеется коляска" : "Без коляски") << std::endl;

}

int Motorcycle::get\_load\_capacity(){

return has\_sidecar ? load : 0;

}

Truck::Truck(std::string model, std::string reg\_no, int speed, int load, bool has\_trailer):

Trans(model, reg\_no, speed, load),

has\_trailer(has\_trailer){};

void Truck::print\_info(){

std::cout << "Грузовая машина" << std::endl;

}

int Truck::get\_load\_capacity(){

return has\_trailer ? load \* 2 : load;

}

class TransDB{

private:

Trans \*\*vehicles;

int size;

public:

TransDB(int);

~TransDB();

void input();

void print();

TransDB \*filter(int);

};

TransDB::TransDB(int n) : size(n){

vehicles = new Trans \*[size];

};

void TransDB::input(){

// Внесение машин в базу данных

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

int choice, tmp, load, speed;

std::string model, reg\_no;

std::cout << "Какой вид транспорта вы хотите добавить?" << std::endl <<

"1 - легковое авто, 2 - мотоцикл, 3 - грузовик ";

std::cin >> choice;

std::cout << "Марка? "; std::cin >> model;

std::cout << "Номер? "; std::cin >> reg\_no;

std::cout << "Скорость? "; std::cin >> speed;

std::cout << "Грузоподъемность? "; std::cin >> load;

switch(choice){

case 1:

vehicles[i] = new Car(model, reg\_no, speed, load);

break;

case 2:

std::cout << "Есть коляска? 1 - есть, 0 - нет ";

std::cin >> tmp;

vehicles[i] = new Motorcycle(model, reg\_no, speed, load, (tmp == 1));

break;

case 3:

std::cout << "Есть прицеп? 1 - есть, 0 - нет ";

std::cin >> tmp;

vehicles[i] = new Truck(model, reg\_no, speed, load, (tmp == 1));

break;

}

}

};

TransDB::~TransDB(){

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

delete vehicles[i];

}

delete vehicles;

}

void TransDB::print(){

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

vehicles[i] -> print\_info();

}

}

TransDB \*TransDB::filter(int min\_weight){

std::list <Trans \*> tmp\_lst;

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

if (vehicles[i]->get\_load\_capacity() >= min\_weight){

tmp\_lst.push\_back(vehicles[i]);

}

}

TransDB \*ret\_db = new TransDB(tmp\_lst.size());

Trans \*tmp;

int i = 0;

while (!tmp\_lst.empty()){

tmp = tmp\_lst.back();

ret\_db->vehicles[i] = tmp;

tmp\_lst.pop\_back();

i++;

}

return ret\_db;

}

int main() {

int n;

std::cout << "Количество машин? ";

std::cin >> n;

TransDB \*db = new TransDB(n);

db->input();

db->print();

int min\_weight;

std::cout << "По какому минимальному весу отобрать машины? ";

std::cin >> min\_weight;

TransDB \*newdb = db->filter(10);

std::cout << "Машины, которые соответствуют параметрам: " << std::endl;

newdb->print();

delete db;

delete newdb;

}