**МИНИСТЕРСТВО ТРАНСПОРТА И КОММУНИКАЦИЙ РЕСПУБЛИКИ БЕЛАРУСЬ**

**Учреждение образования**

**«Белорусский государственный университет транспорта»**

**Кафедра «Управление эксплуатационной работой и охрана труда**

**Отчёт  
по лабораторным работам**

по дисциплине «Средства и технологии анализа и разработки информационных систем»

Выполнил Проверил

Студент группы ГИ-31 м.т.н., ст.пр.

Долинова В. А. Козлов В. Г.

Гомель 2024

**ЛАБОРАТОРНАЯ РАБОТА №6  
ТЕМА: ПРОЕКТИРОВАНИЕ КЛАССОВ**

**Цель работы:** получить основные понятия и навыки по проектированию и созданию классов с использованием UML.

**Задание 1.**

1. Разработать программу в соответствии с индивидуальным заданием (вариантом);

2. Использовать возможности ООП;

3. Каждый класс должен иметь отражающее смысл название и информативный состав.

Общее для всех вариантов: реализовать механизм сериализации (десериализации) объекта согласно индивидуальному заданию.

Вариант 7. Таксопарк. Определить иерархию легковых автомобилей. Создать таксопарк. Подсчитать стоимость автопарка.

**Листинг.**

#include <iostream>

#include <vector>

#include <fstream>

#include <sstream>

class Car {

public:

virtual int getCost() = 0;

virtual std::string getType() = 0;

};

class Audi : public Car {

public:

int getCost() override {

return 200000;

}

std::string getType() override {

return "Audi";

}

};

class BMV : public Car {

public:

int getCost() override {

return 70000;

}

std::string getType() override {

return "BMV";

}

};

class TaxiPark {

private:

std::vector<Car\*> cars;

public:

void addCar(Car\* car) {

cars.push\_back(car);

}

int getTotalCost() {

int totalCost = 0;

for (Car\* car : cars) {

totalCost += car->getCost();

}

return totalCost;

}

void serialize(const std::string& filename) {

std::ofstream file(filename);

for (Car\* car : cars) {

file << car->getType() << std::endl;

}

file.close();

}

void deserialize(const std::string& filename) {

std::ifstream file(filename);

std::string line;

while (std::getline(file, line)) {

if (line == "Audi") {

cars.push\_back(new Audi());

} else if (line == "BMV") {

cars.push\_back(new BMV());

}

}

file.close();

}

};

int main() {

setlocale(LC\_ALL, "");

TaxiPark taxiPark;

taxiPark.addCar(new Audi());

taxiPark.addCar(new BMV());

std::cout << "Стоимость таксопарка составляет: $" << taxiPark.getTotalCost() << std::endl;

taxiPark.serialize("taxi\_park.txt");

TaxiPark newTaxiPark;

newTaxiPark.deserialize("taxi\_park.txt");

//std::cout << "$" << newTaxiPark.getTotalCost() << std::endl;

return 0;

}

**Результат.**

На рисунке 1 представлена UML диаграмма классов для таксопарка.

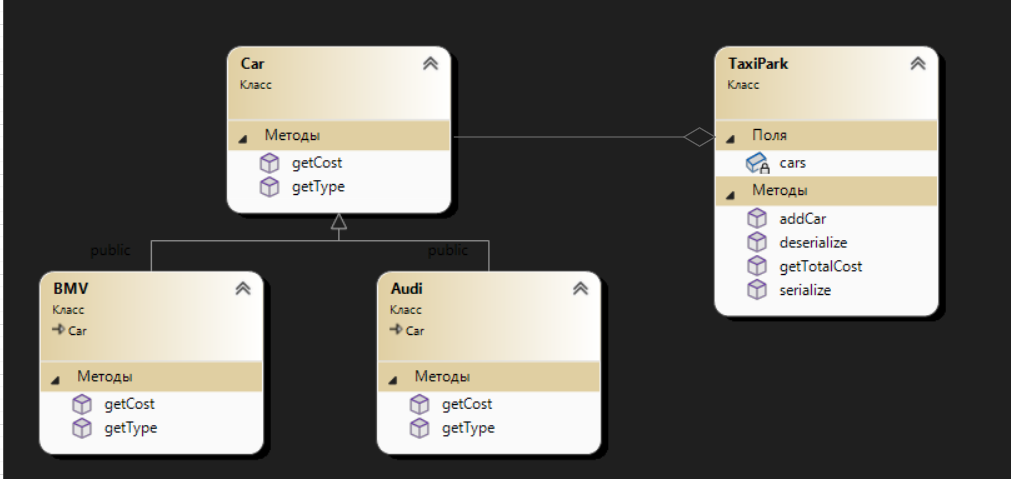


Рисунок 1 – Диаграмма классов для таксопарка

Приведенная на рисунке 1 диаграмма классов для таксопарка описывает 4 класса: TaxiPark, Car, Audi, BMV.

Класс Car является классом-родителем для двух дочерних классов BMV и Audi, таким образом, установив связь наследование. Между классами Car и TaxiPart установлен вид отношений агрегация.

Классы BMV и Audi наследуются от класса Car, состоят из наследованных виртуальных методов getCost() и getType(), которые переопределяются и реализуют свой функционал в дочерних классах: возвращают стоимость каждого из автомобилей, а также их название.

Класс-родитель Audi содержит два виртуальных метода getCost() и getType() для дальнейшего наследования и переопределения дочерними классами.

Класс TaxiPark содержит поле cars, которое является динамическим массивом, а также методы:

- addCar() – позволяет добавлять новые элементы в вектор Car.

- getTotalCost() – подсчитывает стоимость всего автопарка на основе добавленных элементов в массив.

- serialize() – реализует механизм сериализации (процесс сохранения данных объекта во внешнем хранилище), в данном случае – в файл.

- deserialize() – реализует механизм десериализации, в рассматриваемой ситуации – из файла.

**Вывод:** получены основные понятия и навыки по проектированию и созданию классов с использованием UML при помощи среды Visual Studio, а также реализована система классов на языке C++.