Министерство науки и высшего образования РФ

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа «Компьютерных технологий и информационных систем»

ОТЧЕТ

по дисциплине «Практикум по программированию»

**Лабораторная работа № 3**

**Выполнил:**

Cтудент гр. 5130902/30002 С. С. Фибих

**Проверил**

Ст. преподаватель А.М. Журавская

Санкт-Петербург

2024 г.

## **Задание**

Вариант 4

В данной работе необходимо использовать один из шаблонов проектирования, например, описанный выше **Strategy,** реализовать наследование, полиморфизм и парадигму интерфейсных классов. В работе для обработки ошибок должны быть реализованы исключения (часть проверок можно сделать классическим способом).

Сначала необходимо создать диаграмму классов, а потом уже начинать реализацию.

Предметная область: **Аэропорт.** Касса аэропорта имеет список тарифов на различные направления. Тариф содержит название направления и стоимость перевозки. На некоторые направления предоставляется фиксированная скидка. В классе «аэропорт» реализовать метод добавления нового тарифа и метод поиска направления с максимальной стоимостью.

## **Код программы**

**Часть 1.**

#include <iostream>

#include <string>

#include <vector>

#include <sstream>

using namespace std;

float inputFloat(float& f);

bool isFloat(const std::string& input);

class ITariff {

protected:

float cost;

string TName;

public:

ITariff(float cost, string TName) : cost(cost), TName(TName) { cout << "the class has been CREATED" << endl; }

virtual ~ITariff() { cout << "the class has been DESTROYED" << endl; }

virtual float GetCost() const = 0;

virtual string GetName() { return TName; }

};

class DiscontTariff : public ITariff {

private:

float discont;

public:

DiscontTariff(float cost, string TName, float discont) : ITariff(cost, TName), discont(discont) {}

float GetCost() const override { return (cost - discont); };

};

class ClassicTariff : public ITariff {

public:

ClassicTariff(float cost, string TName) : ITariff(cost, TName) {};

float GetCost() const override { return cost; }

};

class Airport {

private:

vector<ITariff\*> tariffs;

public:

~Airport() {

for (ITariff\* tarif : tariffs) delete tarif;

cout << "Airport was deleted" << endl;

}

void addTarif(ITariff\* tarif) {

tariffs.push\_back(tarif);

}

void ShowTariff() {

for (const auto& tarif : tariffs) {

cout << tarif->GetCost() << " " << tarif->GetName() << endl;

}

}

void GetMaxTariff() {

if (tariffs.empty()) {

cout << "Нет тарифов" << endl;

return;

}

ITariff\* max\_cost = tariffs[0];

for (const auto& tarif : tariffs) {

if (tarif->GetCost() > max\_cost->GetCost()) max\_cost = tarif;

}

cout << "Стоимость билета - " << max\_cost->GetCost() << " ; Название - " << max\_cost->GetName() << endl;

}

};

int main()

{

system("chcp 1251");

system("cls");

Airport Y;

string test, wntd; float Tcost, Tdiscont; int error;

//список комманд - 1-2. добавить тариф со скидкой, 3. вывести все тарифы, 4. вывести тариф с максимальной стоимостью.

do {

cout << "Список комманд:\n1 - Добавить тариф с фиксированной скидкой\n2 - Добавить тариф без фиксированной скидкой\n3 - Вывести список тарифов\n4 - Вывести тариф с максимальной стоимостью\n5 - Завершить работу\n";

cout << "Введите комманду: ";

cin >> wntd;

cout << "\n";

if (!(wntd == "1" || wntd == "2" || wntd == "3" || wntd == "4" || wntd == "5")) wntd = "0";

if (wntd == "0") {

cout << "Неправильная команда\n";

}

else if (wntd == "1") {

try {

cout << "Введите стоимость билета" << endl;

inputFloat(Tcost);

cout << "Введите фиксированную скидку" << endl;

inputFloat(Tdiscont);

cout << "Введите название направления" << endl;

cin >> test;

cin.get();

}

catch (int code) {

if (code == -1) cout << "Значение не входит в диапазон [0; 100000]" << endl;

else if (code == -2) cout << "Значение некорректно" << endl;

continue;

}

if (Tcost < Tdiscont) {

cout << "Скидка не может быть больше стоимости" << endl;

continue;

}

Y.addTarif(new DiscontTariff(Tcost, test, Tdiscont));

}

else if (wntd == "2") {

try {

cout << "Введите стоимость билета" << endl;

inputFloat(Tcost);

cout << "Введите название направления" << endl;

cin >> test;

cin.get();

}

catch (int code) {

if (code == -1) cout << "Значение не входит в диапазон [0; 100000]" << endl;

else if (code == -2) cout << "Значение некорректно" << endl;

continue;

}

Y.addTarif(new ClassicTariff(Tcost, test));

}

else if (wntd == "3") {

Y.ShowTariff(); cout << "\n";

}

else if (wntd == "4") {

Y.GetMaxTariff(); cout << "\n";

}

} while (wntd != "5");

printf("\n\n\n\tСПАСИБО ЗА ВНИМАНИЕ\n\n\n");

return 0;

}

float inputFloat(float& f) {

string temp;

while (true) {

cin >> temp;

if (isFloat(temp)) {

//std::cout << "Введено корректное число с плавающей точкой.\n";

if (stof(temp) < 0 || stof(temp) > 100000) {

throw (-1);

continue;

}

break;

}

else {

//std::cout << "Введено некорректное значение.\n";

cin.get();

throw (-2);

break;

}

}

return f = stof(temp);

}

bool isFloat(const std::string& input) {

std::istringstream stream(input);

float f;

// Проверяем, что поток успешно конвертирует строку в float

if (!(stream >> f)) {

return false; // Если не удалось, возвращаем false

}

// Проверяем, что в потоке не осталось лишних символов

return stream.eof();

}

**Пример работы программы**

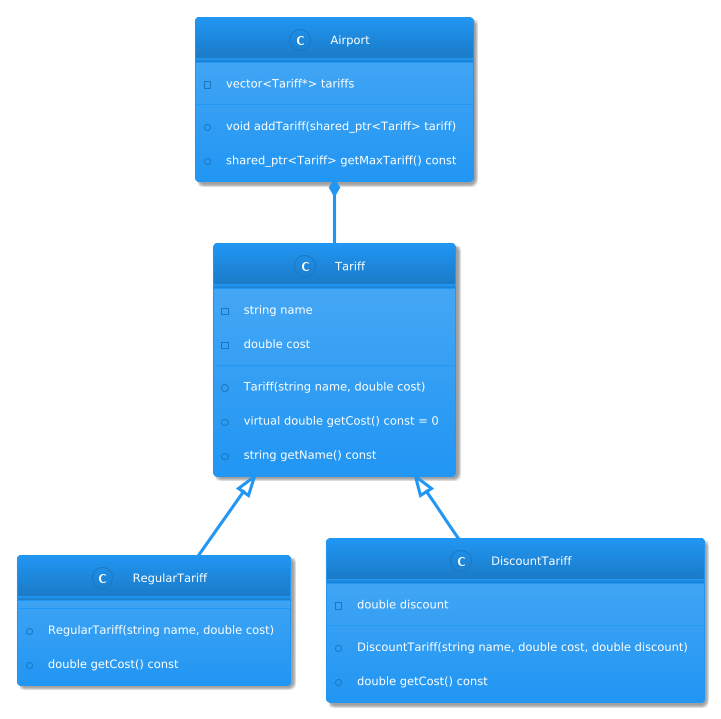
****

Рис. 1: Диаграмма классов

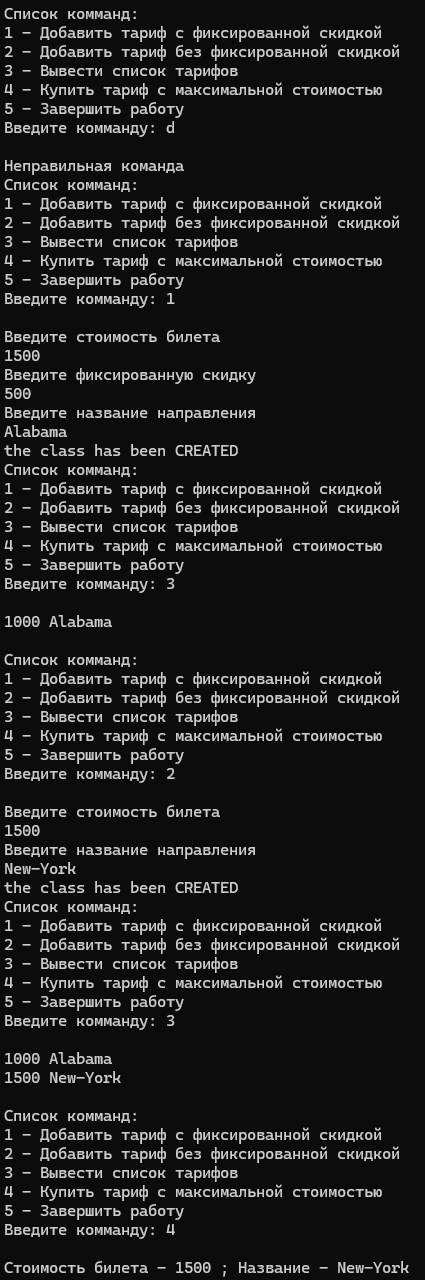


Рис. 2: Демонстрация работы

## **Вывод**

В ходе выполнения этой работы, я научился реализовывать шаблон проектирования классов Strategy, наследование, полиморфизм, а также парадигму интерфейсных классов.