Технологии конструирования программного обеспечения

Отчет по лабораторной работе № 04

Группа: 221-329 **Студент:** Минчаков Аркадий Сергеевич

Задание на лабораторную работу:

Для заданного варианта задания разработать UML-диаграмму классов и диаграмму последовательности. Разработать консольное приложение (C++, C#). Допускается вводить дополнительные понятия предметной области. В программе предусмотреть тестирование функциональности созданных объектов классов.

1. Изучить пример проектирования программной системы с использованием паттерна Адаптер [Турчин-Архитектура ИС.pdf [Электронный ресурс], с. 117–123].
2. Изучить пример проектирования программной системы с использованием паттерна Фасад [Турчин-Архитектура ИС.pdf [Электронный ресурс], с. 123–133].
3. Разработать библиотеку классов, которая содержит указанные классы (таблица 1), задействованные в паттерне Адаптер. Для адаптера объектов атрибуты класса должны быть реализованы как автоматические свойства, а для паттерна Адаптер классов – как защищённые поля.
4. Разработать библиотеку классов, которая должна содержать классфасад и заданный набор классов (таблица 2). В фасаде необходимо задать ссылки на другие классы библиотеки.
5. Добавить в решение консольное приложение, которое для реализованных паттернов играет роль клиента. Продемонстрировать работу в консольном приложении работу шаблонов проектирования.
6. Для заданных вариантов разработать UML-диаграммы классов и диаграммы последовательности.

Номер в списке группы - 15, вариант адаптера - 1, вариант фасада - 3

Номер варианта	Тип адаптера	Требуемый интерфейс	Адаптируемый класс
1	объектов	`+ CalculateDp(T0 : int, dT : int): double` – определить изменение давления при заданной начальной температуре `T0` и изменении температуры `dT`; `+ ModifMass(dm: double):void` – изменить массу газа в баллоне на величину `dm`; `+ GetData() : string` – возвращает строку с данными об объекте	Баллон с газом. *Атрибуты:* • `Volume : double` – объём баллона, м^3; • `Mass : double` – масса газа, кг; • `Molar : double` – молярная масса газа, кг/моль. *Операции:* `+ GetPressure(T : int) : double` – определить давление в баллоне при заданной температуре газа `T`; `+ AmountOfMatter() : double` – определить количество вещества; `+ ToString() : string` – возвращает строку с данными об объекте
2	классов	`+ ModifVolume(dV : double) : void` – изменить объём баллона на величину `dV`; `+ GetDp(T0 : int, T1 : int) : double` – определить изменение давления при изменении температуры с `T0` до `T1`; `+ Passport() : string` – возвращает строку с данными об объекте	

Figure 1: Таблица 1

Номер варианта	Данные для разработки приложения на основе шаблона Фасад
1	*Расчёт страхового взноса за недвижимость*. Классы (типы недвижимости): квартира, таун-хаус, коттедж. Параметры: срок страхования, жилплощадь (м^2), число проживающих, год постройки здания, износ здания (%)
2	*Расчёт ежедневной нормы потребления килокалорий*. Классы (Тип телосложения): Астеник, Нормостеник, Гиперстеник. Параметры: Рост, Вес, Возраст, Пол Группа физической активности (низкая, средняя и высокая активность)
3	*Расчёт стоимости туристической путевки*. Классы (виды путевок): пляжный отдых, экскурсия, горные лыжи. Параметры: длительность, страна, гостиница (число звезд), рацион питания (двухразовый, трехразовый, всё включено)

Figure 2: Таблица 2

UML-диаграммы классов:

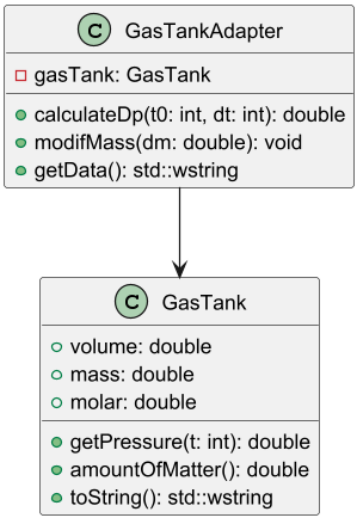

Figure 3: UML-диаграмма классов для шаблона Adapter

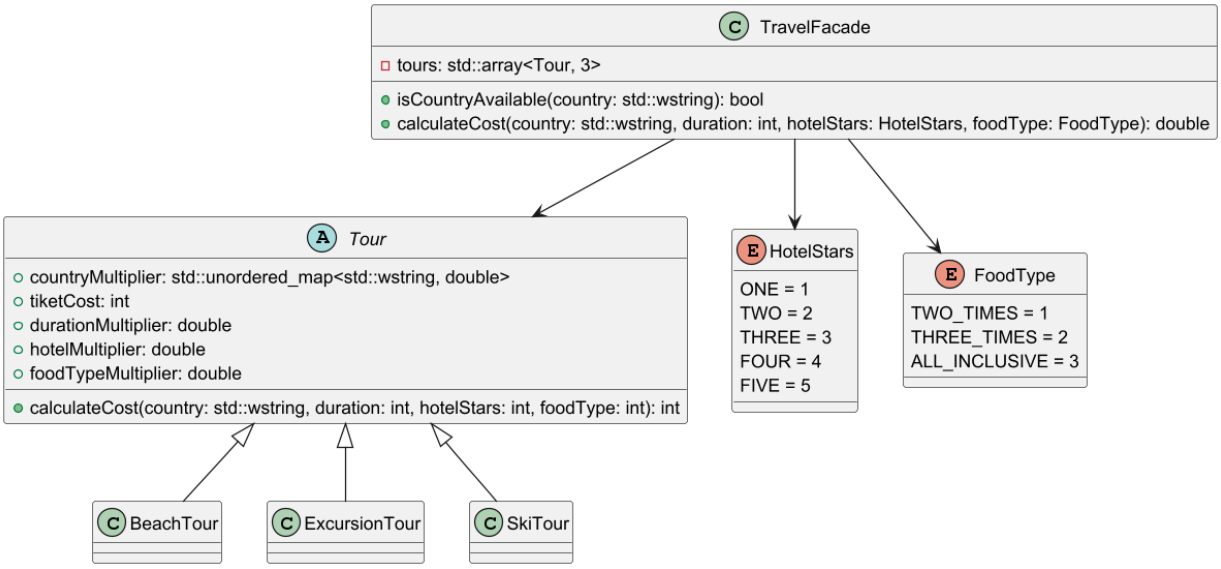

Figure 4: UML-диаграмма классов для шаблона Facade

Диаграммы последовательности:

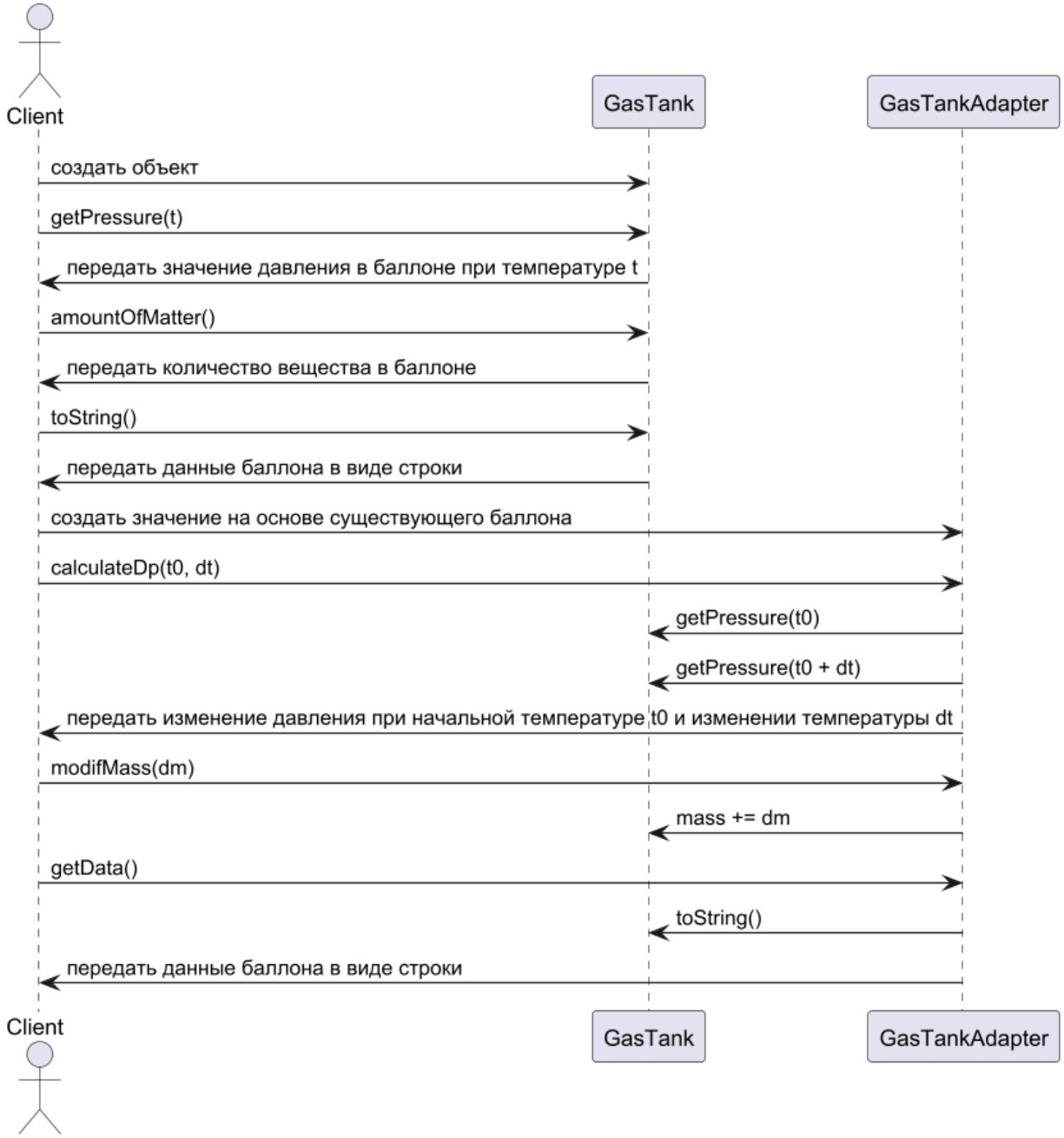

Figure 5: Диаграмма последовательности для шаблона Adapter

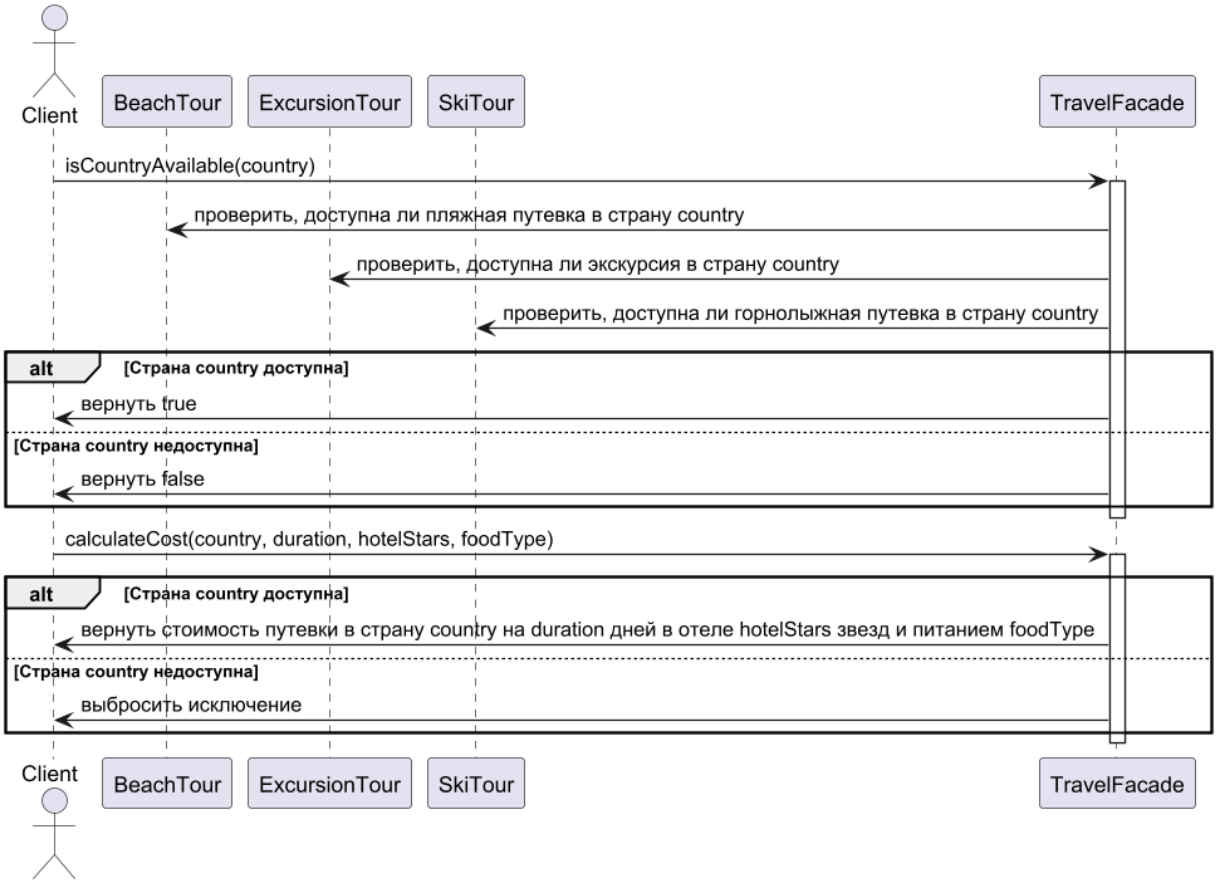

Figure 6: Диаграмма последовательности для шаблона Facade

Исходный код программы:

adapter/main.cpp

```cpp
#include <string>
#include <iostream>
#include <cassert>

class GasTank {
private:
    const double R = 8.31;

public:
    double volume;

    double mass;

    double molar;

    GasTank(double volume, double mass, double molar) : volume(volume),
        mass(mass), molar(molar) {
        std::wcout << L"Создан новый баллон с газом" << std::endl;
    }

    double getPressure(int t) {
        // p V = v R T => p = v R T / V
        return amountOfMatter() * R * t / volume;
    }

    double amountOfMatter() {
        // p V = v R T = m R T / M => v = m / M
        return mass / molar;
    }

    std::wstring toString() {
        return L"Баллон с газом {"
               "Объем сосуда - " + std::to_wstring(volume) + L", "
               "Масса газа - " + std::to_wstring(mass) + L", "
               "Молярная масса газа - " + std::to_wstring(molar) + L"}";
    }
};

class GasTankAdapter {
private:
    GasTank gasTank;

public:
    GasTankAdapter(GasTank gasTank) : gasTank(gasTank) {
        std::wcout << L"Создан новый адаптер для газового баллона" << std::endl;
    }

    double calculateDp(int t0, int dt) {
        // dp(t0, dt) = p - p0 = p(t0 + dt) - p(t0)
        return gasTank.getPressure(t0 + dt) - gasTank.getPressure(t0);
    }

    void modifMass(double dm) {
        gasTank.mass += dm;
    }

    std::wstring getData() {
        return gasTank.toString();
    }
};

```

```cpp
int main() {
    setlocale(LC_ALL, "Russian");

    GasTank gasTank(3.12, 50, 0.002016);

    std::wcout << gasTank.toString() << std::endl;
    std::wcout << L"Количество вещества в баллоне: " << gasTank.amountOfMatter() <<
        std::endl;

    GasTankAdapter gasTankAdapter(gasTank);

    std::wcout << L"Изменение давления при t0 = 0 и dt = 1: " <<
        gasTankAdapter.calculateDp(0, 1) << std::endl;

    gasTankAdapter.modifMass(40);

    std::wcout << L"Баллон изменен: " << gasTankAdapter.getData() << std::endl;
    std::wcout << L"Изменение давления при t0 = 0 и dt = 1: " <<
        gasTankAdapter.calculateDp(0, 1) << std::endl;
    std::wcout << L"Изменение давления при t0 = 10 и dt = 2: " <<
        gasTankAdapter.calculateDp(10, 2) << std::endl;

    return 0;
}
```

facade/TravelFacade.h

```cpp
#ifndef SOFTWARE_DESIGN_TECHNOLOGIES_TRAVELFACADE_H
#define SOFTWARE_DESIGN_TECHNOLOGIES_TRAVELFACADE_H


#include <iostream>
#include <string>
#include <unordered_map>
#include <array>

namespace Travel {
    class Tour {
    public:
        std::unordered_map<std::wstring, double> countryMultiplier;
        const int ticketCost;
        const double durationMultiplier;
        const double hotelMultiplier;
        const double foodTypeMultiplier;

        Tour(
            std::unordered_map<std::wstring, double> countryMultiplier,
            const int ticketCost,
            const double durationMultiplier,
            const double hotelMultiplier,
            const double foodTypeMultiplier
        );

        int calculateCost(std::wstring country, int duration, int hotelStars,
            int foodType);
    };

    class BeachTour : public Tour {
    public:
        BeachTour();
    };

    class ExcursionTour : public Tour {
    public:
```

```
        ExcursionTour();
    };

    class SkiTour : public Tour {
    public:
        SkiTour();
    };
}

struct CountryNotAvailableException {
    const wchar_t *message;
};

class TravelFacade {
private:
    std::array<Travel::Tour, 3> tours;
public:
    TravelFacade();

    enum HotelStars {
        ONE = 1,
        TWO = 2,
        THREE = 3,
        FOUR = 4,
        FIVE = 5
    };

    enum FoodType {
        TWO_TIMES = 1,
        THREE_TIMES = 2,
        ALL_INCLUSIVE = 3
    };

    bool isCountryAvailable(std::wstring country);

    double calculateCost(std::wstring country, int duration, HotelStars
        hotelStars, FoodType foodType);
};


#endif //SOFTWARE_DESIGN_TECHNOLOGIES_TRAVELFACADE_H
```

facade/main.cpp

```
#include <iostream>
#include <iomanip>

#include "TravelFacade.h"

int main() {
    setlocale(LC_ALL, "Russian");

    TravelFacade facade;

    if (facade.isCountryAvailable(L"ОАЭ")) {
        std::wcout
                << L"Путевка на пляжный курорт в ОАЭ на 7 дней в 5-звездочный отель с
                    двухразовым питанием стоит: "
                << std::setprecision(2) << std::fixed
                << facade.calculateCost(L"ОАЭ", 7, TravelFacade::FIVE,
                    TravelFacade::TWO_TIMES) << std::endl;
    }

    if (facade.isCountryAvailable(L"Непал")) {
```

```
        std::wcout
                << L"Путевка на экскурсию в Непал на 5 дней в 3-звездочный отель с трехразовым питанием стоит: "
                << std::setprecision(2) << std::fixed
                << facade.calculateCost(L"Непал", 5, TravelFacade::THREE, TravelFacade::THREE_TIMES) << std::endl;
    }

    if (facade.isCountryAvailable(L"Россия")) {
        std::wcout
                << L"Путевка на горнолыжный курорт в Россию на 14 дней в 4-звездочный отель с питанием типа все "
                   "включено стоит: " << std::setprecision(2) << std::fixed
                << facade.calculateCost(L"Россия", 14, TravelFacade::FOUR, TravelFacade::ALL_INCLUSIVE) << std::endl;
    }

    std::wcout << L"Доступность страны 'Люксембург': " << (facade.isCountryAvailable(L"Люксембург") ? "true" : "false") << std::endl;
    std::wcout << L"Попытка посчитать стоимость путевки в Люксембург: " << std::endl;
    try {
        facade.calculateCost(L"Люксембург", 0, TravelFacade::ONE, TravelFacade::TWO_TIMES);
    } catch (CountryNotAvailableException &e) {
        std::wcout << e.message << std::endl;
    }

    return 0;
}
```

facade/TravelFacade.cpp

```
#include "TravelFacade.h"

Travel::Tour::Tour(
    std::unordered_map<std::wstring, double> countryMultiplier,
    const int ticketCost,
    const double durationMultiplier,
    const double hotelMultiplier,
    const double foodTypeMultiplier
) : countryMultiplier(countryMultiplier),
    ticketCost(ticketCost),
    durationMultiplier(durationMultiplier),
    hotelMultiplier(hotelMultiplier),
    foodTypeMultiplier(foodTypeMultiplier) {}

int Travel::Tour::calculateCost(std::wstring country, int duration, int hotelStars, int foodType) {
    if (!countryMultiplier.contains(country)) {
        return -1;
    }
    return countryMultiplier[country] * (
        ticketCost + durationMultiplier * duration *
                     (0.5 + hotelMultiplier * hotelStars) *
                     (0.7 + foodTypeMultiplier * foodType)
    );
}

Travel::BeachTour::BeachTour() : Tour({{L"Турция", 1.0}, {L"Индонезия", 1.3}, {L"ОАЭ", 2}}, 40000, 5050, 0.1, 0.05) {}

Travel::ExcursionTour::ExcursionTour() : Tour({{L"Непал", 0.9}, {L"Америка", 1.3}, {L"Африка", 0.6}}, 30000, 3500, 0.08, 0.03) {}

```

```
30 Travel::SkiTour::SkiTour() : Tour({{L"Швейцария", 1.0}, {L"Россия", 0.7},
       {L"Австрия", 0.9}}, 50000, 8000, 0.11, 0.1) {}
31
32 TravelFacade::TravelFacade() : tours({Travel::BeachTour(),
       Travel::ExcursionTour(), Travel::SkiTour()}) {}
33
34 bool TravelFacade::isCountryAvailable(std::wstring country) {
35     for (auto &tour : tours) {
36         if (tour.countryMultiplier.contains(country)) {
37             return true;
38         }
39     }
40     return false;
41 }
42
43 double TravelFacade::calculateCost(
44     std::wstring country,
45     int duration,
46     TravelFacade::HotelStars hotelStars,
47     TravelFacade::FoodType foodType
48 ) {
49     for (auto &tour : tours) {
50         if (tour.countryMultiplier.contains(country)) {
51             return tour.calculateCost(country, duration, hotelStars, foodType);
52         }
53     }
54
55     throw CountryNotAvailableException{L"Страна из путевки недоступна для поездки."};
56 }
```