**Технологии конструирования программного обеспечения**

**Отчет по лабораторной работе № 4**

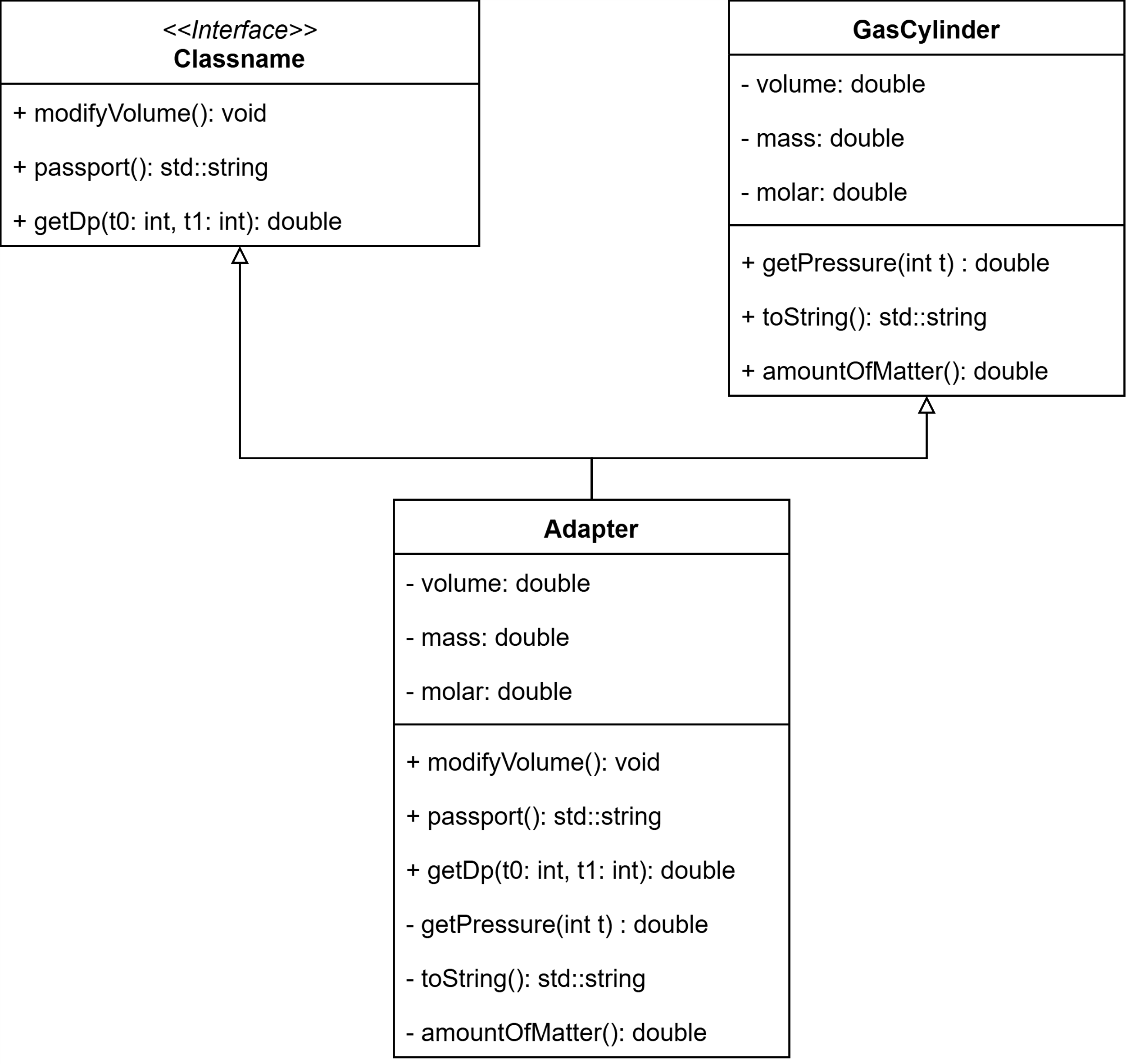
**Группа:** 221-3210 **Студент:** Обухов Алексей Сергеевич

**Задание на лабораторную работу**

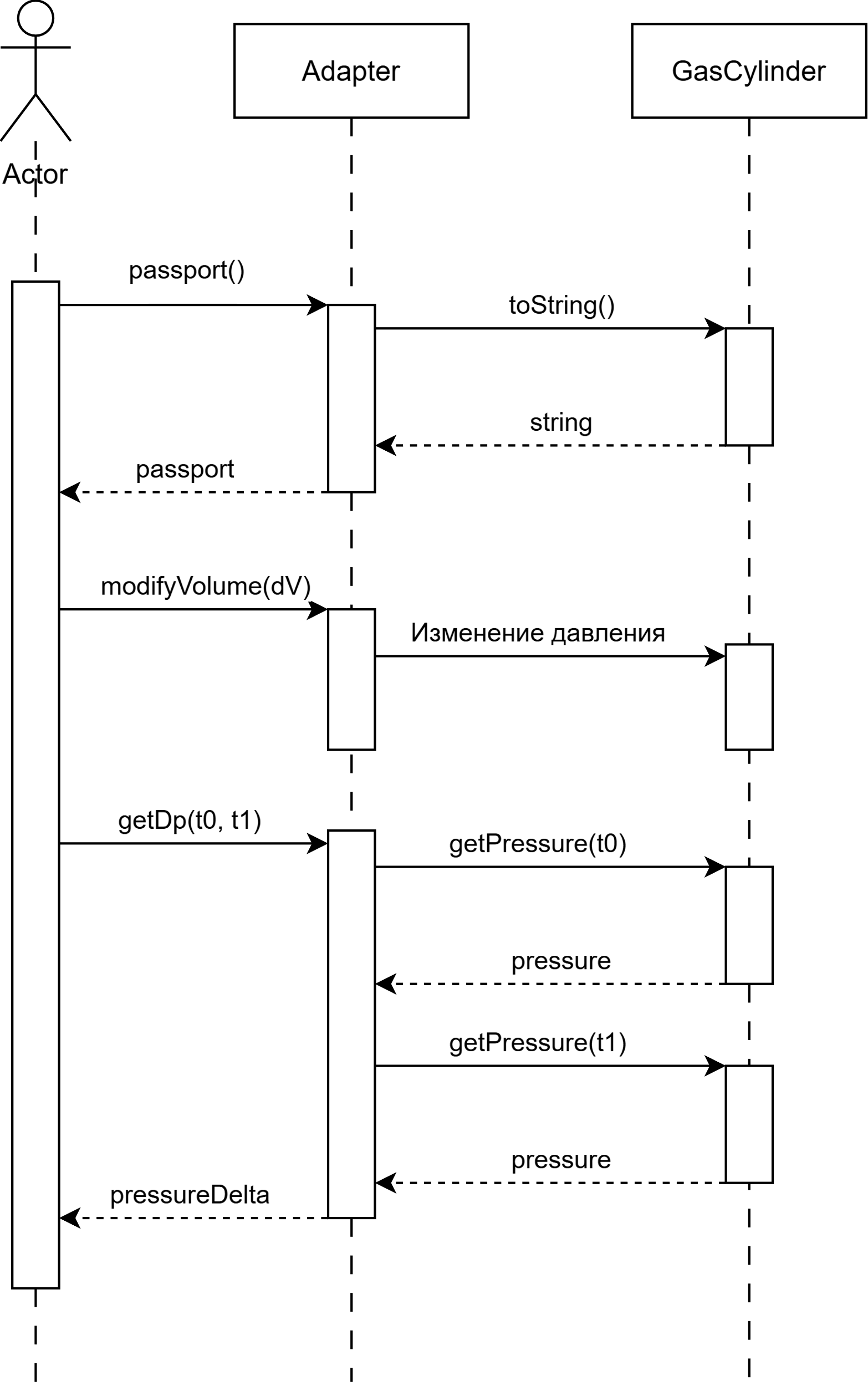
**1. Адаптер класса.**

|  |  |
| --- | --- |
| **Требуемый интерфейс** | **Адаптируемый класс** |
| + 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 | Баллон с газом.  Атрибуты:  • Volume : double – объём баллона, м3;  • 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 |

**Диаграмма классов**

****

**Диаграмма последовательности**

****

**Исходный код программы**

Содержимое «Adapter.h»

#pragma once

#include <string>

// Класс баллона с газом

class GasCylinder {

protected:

double volume; // объем баллона в м^3

double mass; // масса газа в кг

double molar; // молярная масса газа в кг/ммоль

public:

GasCylinder() {

volume = 0.0;

mass = 0.0;

molar = 0.0;

}

GasCylinder(double volume, double mass, double molar) {

this->volume = volume;

this->mass = mass;

this->molar = molar;

}

// Возвращает текущее давление в баллоне

double getPressure(int t) {

// 8.31 - число R

double pressure = (amountOfMatter() \* 8.31 \* t) / volume;

return pressure;

};

// Возвращает кол-во вещества в баллоне

double amountOfMatter() {

double amount = (mass \* 1'000) / molar;

return amount;

};

// Возвращает строку с сведениями о баллоне

std::string toString() {

std::string data = "Сведения о газе, содержащемся в баллоне:\n";

data += "Объем баллона: " + std::to\_string(volume) + " м^3\n";

data += "Масса газа: " + std::to\_string(mass) + " кг\n";

data += "Молярная масса газа: " + std::to\_string(molar) + " г/моль\n";

data += "Количество газа: " + std::to\_string(amountOfMatter()) + " моль\n";

return data;

};

};

// Класс интерфейс

class ITarget {

public:

// Изменяет объем баллона на величину dV

virtual void modifyVolume(double dV) = 0 ;

// Определяет изменение давления при изменении температуры

virtual double getDp(int t0, int t1) = 0;

// Получить сведения о баллоне

virtual std::string passport() = 0;

};

// Класс адаптера

class Adapter : public ITarget, private GasCylinder {

public:

Adapter(double volume, double mass, double molar) {

this->volume = volume;

this->mass = mass;

this->molar = molar;

}

void modifyVolume(double dV) override {

this->volume += dV;

}

double getDp(int t0, int t1) override {

double pressureT0 = getPressure(t0);

double pressureT1 = getPressure(t1);

return pressureT1 - pressureT0;

}

std::string passport() override {

return toString();

}

};

Содержимое «main.cpp»

#include <iostream>

#include "Adapter.h"

int main() {

setlocale(LC\_ALL, "Russian");

ITarget\* target = new Adapter(1.0, 1.0, 32.0);

std::cout << target->passport() << "\n";

target->modifyVolume(2);

std::cout << target->passport() << "\n";

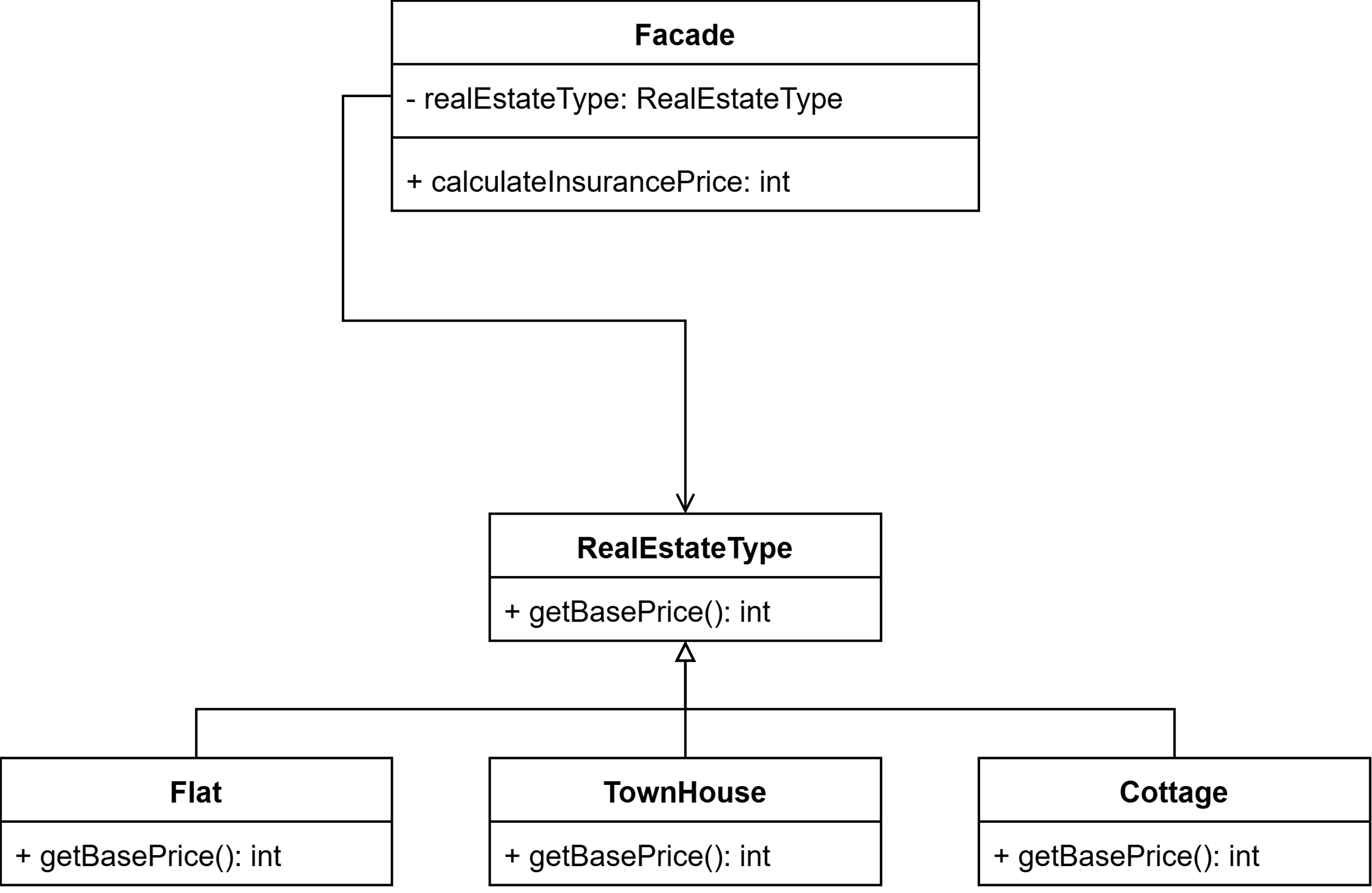
std::cout << "Разница в давлении между 22°C и 30°C: " << target->getDp(22,30) << " Па\n";

return 0;

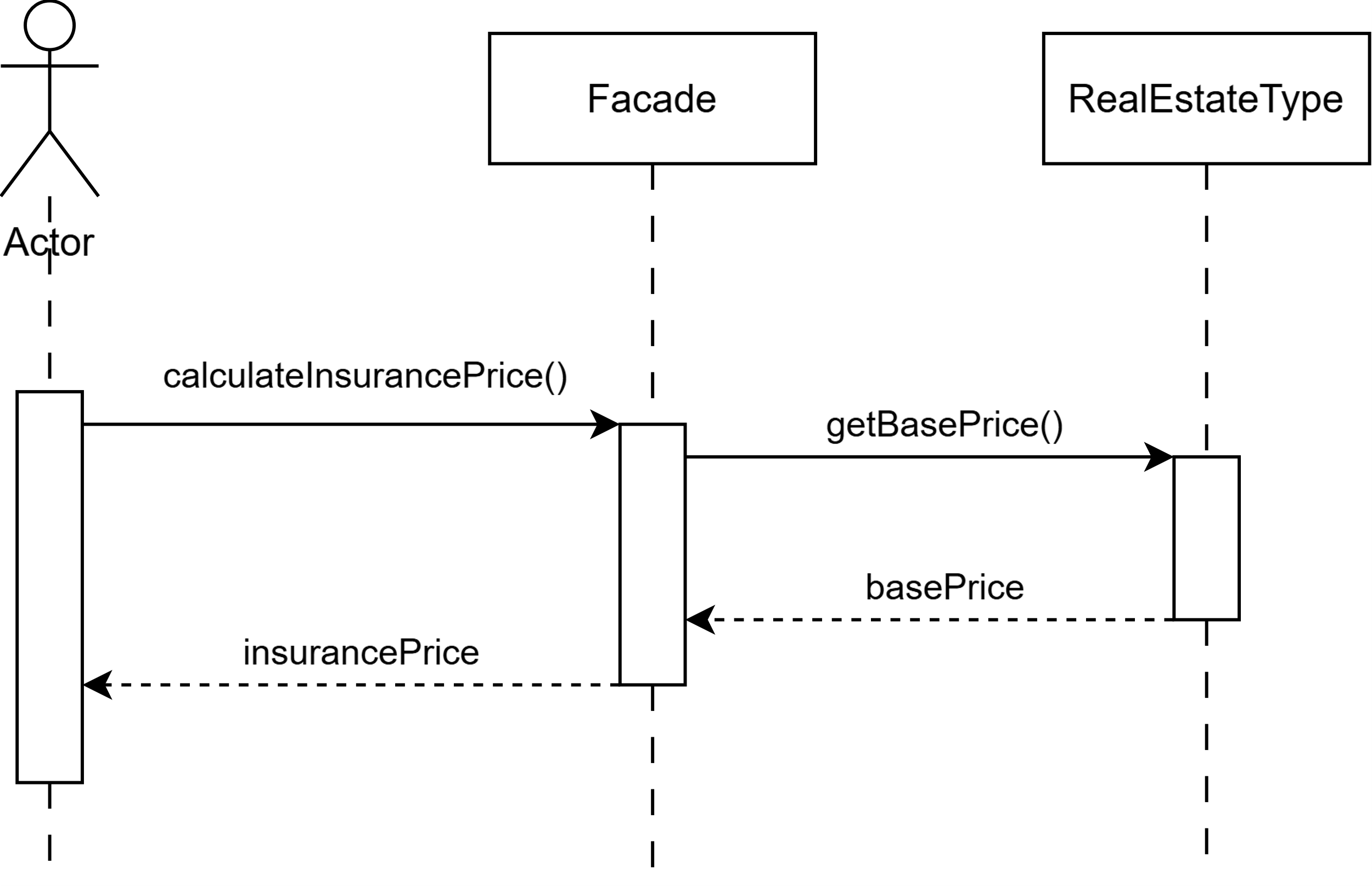
}

**2. Фасад.** Расчёт страхового взноса за недвижимость. Классы (типы недвижимости): квартира, таун-хаус, коттедж. Параметры: срок страхования, жилплощадь (м2), число проживающих, год постройки здания, износ здания (%)

**Диаграмма классов**

****

**Диаграмма последовательности**

****

**Исходный код программы**

Содержимое «Facade.h»

#pragma once

#include <iostream>

namespace subsystem {

// Базовый класс недвижимости

class RealEstateType {

public:

// Базовый платеж

virtual int getBasePrice() = 0;

};

// Класс квартиры

class Flat : public RealEstateType {

int getBasePrice() override {

return 3500;

}

};

// Класс таунхауса

class TownHouse : public RealEstateType {

int getBasePrice() override {

return 2700;

}

};

// Класс коттеджа

class Cottage : public RealEstateType {

int getBasePrice() override {

return 4200;

}

};

}

// Фасад для расчета стоимости страхового взноса

class Facade {

protected:

// Тип недвижимости

subsystem::RealEstateType\* realEstate;

// Площадь недвижимости

int area;

// Износ недвижимости

int wear;

public:

Facade(subsystem::RealEstateType\* realEstate, int area, int wear) {

this->realEstate = realEstate;

this->area = area;

this->wear = wear;

}

int calculateInsurancePrice() {

double wearCoefficient = 1.0 + ((double)wear / 100.0);

int insurancePrice = realEstate->getBasePrice() \* area \* wearCoefficient;

return insurancePrice;

}

};

Содержимое «main.cpp»

#include <iostream>

#include "Facade.h"

int insurancePriceCalculator(subsystem::RealEstateType\* realEstateType, int area, int wear) {

Facade\* facade = new Facade(realEstateType, area, wear);

int price = facade->calculateInsurancePrice();

delete facade;

return price;

}

int main() {

setlocale(LC\_ALL, "Russian");

std::cout << "Стоимость страхового взноса за:\n";

subsystem::RealEstateType\* flat = new subsystem::Flat;

std::cout << "Кватиру: " << insurancePriceCalculator(flat, 60, 10) << "\n";

delete flat;

subsystem::RealEstateType\* townHouse = new subsystem::TownHouse;

std::cout << "Таунхаус: " << insurancePriceCalculator(townHouse, 60, 10) << "\n";

delete townHouse;

subsystem::RealEstateType\* cottage = new subsystem::Cottage;

std::cout << "Коттедж: " << insurancePriceCalculator(cottage, 60, 10) << "\n";

delete cottage;

return 0;

}