

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное образовательное учреждение высшего образования  
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»  
КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ  
И ПРОГРАММНОЙ ИНЖЕНЕРИИ (КАФЕДРА №43)

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ:

Старший преподаватель

должность, уч. степень, звание

подпись, дата

Е. О. Шумова

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №8

«Структурные шаблоны проектирования»

ПО ДИСЦИПЛИНЕ «ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР.

4831

К

подпись, дата

К.А. Корнющенко

инициалы, фамилия

Санкт-Петербург 2020

## Задание:

Изучить принципы построения приложений с графическим интерфейсом, используя библиотеку Qt, применив на практике знания базовых синтаксических конструкций языка C++ и объектно-ориентированного программирования.

Закрепить знания по теме: Структурные шаблоны проектирования.

## Листинг программы:

### Заголовочный файлы

#### apartmentcalc.h

```
#ifndef APARTMENTCALC_H
#define APARTMENTCALC_H

#include <QObject>
#include "estate.h"

class ApartmentCalc : public QObject{
    Q_OBJECT
public:
    explicit ApartmentCalc(QObject *parent = nullptr);
    static int getCost(Estate* value); // статическая функция, которая
    // рассчитывает и возвращает стоимость страхового взноса для жилья типа
    // Апартаменты
signals:
};

#endif // APARTMENTCALC_H
```

#### calculationfacade.h

```
#ifndef CALCULATIONFACADE_H
#define CALCULATIONFACADE_H
#include <QObject>
#include <estate.h>
#include <apartmentcalc.h>
#include <luxuriousapartmentcalc.h>
#include <townhousecalc.h>
#include <cottagecalc.h>

class CalculationFacade : public QObject{
    Q_OBJECT
public:
    explicit CalculationFacade(QObject *parent = nullptr);
    static int getCost(Estate *value);
};

#endif // CALCULATIONFACADE_H
```

#### cottagecalc.h

```
#ifndef COTTAGECALC_H
#define COTTAGECALC_H
```

```

#include <QQuickItem>
#include "estate.h"

class cottagecalc : public QQuickItem{
    Q_OBJECT
public:
    cottagecalc(); //конструктор класса ApartmentCalc
    static int getCost(Estate* value); // статическая функция, которая
    рассчитывает и возвращает стоимость страхового взноса для жилья типа Коттедж
signals:

};

```

```

#endif // COTTAGECALC_H

```

## estate.h

```

#ifndef ESTATE_H
#define ESTATE_H

#include <QObject>

class Estate : public QObject{
    Q_OBJECT
public:
    //перечисление недвижимости в предметной области
    enum EstateType {
        ECONOM,
        LUXURIOUS,
        TOWN_HOUSE,
        COTTAGE
    };
    Estate(); //конструктор по умолчанию
    Estate(int age,int area,int residents,int months,EstateType type,QString
owner); // конструктор с параметрами
    int getAge(); //функция для возврата значение age
    int getArea(); //функция для возврата значение area
    int getResidents(); //функция для возврата значение residents
    int getMonths(); //функция для возврата значение months
    EstateType getType(); //функция для возврата значение type
    QString getOwner(); //функция для возврата значение owner
    explicit Estate(QObject *parent = nullptr);
    EstateType getType() const;
private:
    int age;
    int area;
    int residents;
    int months;
    EstateType type;
    QString owner;
signals:
};

```

```

#endif // ESTATE_H

```

## luxuriousapartmentcalc.h

```

#ifndef LUXURIOUSAPARTMENTCALC_H

```

```

#define LUXURIOUSAPARTMENTCALC_H

#include <QObject>
#include "estate.h"

class luxuriousapartmentcalc : public QObject{
    Q_OBJECT
public:
    explicit luxuriousapartmentcalc(QObject *parent = nullptr);
    static int getCost(Estate* value); // статическая функция, которая
    расчитывает и возвращает стоимость страхового взноса для жилья типа Люкс
signals:

};

#endif // LUXURIOUSAPARTMENTCALC_H

```

## states.h

```

#ifndef STATES_H
#define STATES_H

#include <QObject>
#include <estate.h>

//класс для хранения информации о предыдущих запросах
class States : public QObject{
    Q_OBJECT
public:
    explicit States(QObject *parent = nullptr);
    ~States();
    void undo(); //функция для работы с actualData
    bool hasStates(); //функция для проверки коллекции на наличие элементов
    Estate *getActualData(); //функция возвращает последний элемент коллекции
    void add(Estate *value); //функция добавляет элемент в коллекцию
private:
    QList<Estate *> array; //объявление коллекции, которая может хранить
    элементы класса Estate
    Estate *actualData; // последние данные
};

#endif // STATES_H

```

## townhousecalc.h

```

#ifndef TOWNHOUSECALC_H
#define TOWNHOUSECALC_H

#include <QWidget>
#include <estate.h>

class townhousecalc : public QWidget{
    Q_OBJECT
public:
    explicit townhousecalc(QWidget *parent = nullptr);
    static int getCost(Estate* value); // статическая функция, которая
    расчитывает и возвращает стоимость страхового взноса для жилья типа ТаунХаус
signals:

```

```
};
```

```
#endif // TOWNHOUSECALC_H
```

## widget.h

```
#ifndef WIDGET_H
#define WIDGET_H

#include <QWidget>

QT_BEGIN_NAMESPACE
namespace Ui { class Widget; }
QT_END_NAMESPACE

class Widget : public QWidget{
    Q_OBJECT
public:
    Widget(QWidget *parent = nullptr);
    ~Widget();
private:
    Ui::Widget *ui;
};

#endif // WIDGET_H
```

## Исходный файлы

### apartmentcalc.cpp

```
#include "apartmentcalc.h"

ApartmentCalc::ApartmentCalc(QObject *parent) : QObject(parent) {}
//конструктор класса ApartmentCalc

//расчет стоимости
//30 - кол-во дней в месяце
//320 - коэффициент для данного типа жилья
int ApartmentCalc::getCost(Estate *value){
    return value->getArea() * value->getMonths() * 30 * 320;
}
```

### calculationfacade.cpp

```
#include "calculationfacade.h"

CalculationFacade::CalculationFacade(QObject *parent) : QObject(parent) {}

int CalculationFacade::getCost(Estate *value) {
    int cost;
    switch (value->getType()) {
        case Estate::EstateType::ECONOM:
            cost = ApartmentCalc::getCost(value); //произходит вызов статической
            функции класса ApartmentCalc, если value->getType() == ECONOM
            break;
        case Estate::EstateType::LUXURIOUS:
```

```

        cost = luxuriousapartmentcalc::getCost(value); //произходит вызов
статической функции класса luxuriousapartmentcalc, если value->getType() ==
LUXURIOUS
        break;
    case Estate::EstateType::TOWN_HOUSE:
        cost = townhousecalc::getCost(value); //произходит вызов статической
функции класса townhousecalc, если value->getType() == TOWN_HOUSE
        break;
    case Estate::EstateType::COTTAGE:
        cost = cottagecalc::getCost(value); //произходит вызов статической
функции класса cottagecalc, если value->getType() == COTTAGE
        break;
    default:
        cost = -1;
        break;
    }
    return cost;
}

```

### cottagecalc.cpp

```

#include "cottagecalc.h"

cottagecalc::cottagecalc() {}

//функция для расчета стоимости
//20 - коэффициент для данного типа жилья
int cottagecalc::getCost(Estate *value) {
    return value->getArea() * value->getMonths() * value->getResidents() *
20;
}

```

### estate.cpp

```

#include "estate.h"
#include<iostream>
#include<string>

Estate::Estate(QObject *parent) : QObject(parent) {}

//конструктор по умолчанию
Estate::Estate() {
    this->age = 0;
    this->area = 0;
    this->residents = 0;
    this->months = 0;
    this->type = ECONOM;
    this->owner = "";
}

//конструктор с параметрами
Estate::Estate(int age, int area, int residents, int months, EstateType
type, QString owner) {
    this->age = age;
    this->area = area;
    this->residents = residents;
    this->months = months;
    this->type = type;
    this->owner = owner;
}

//функция возврата возраста

```

```

int Estate::getAge() {
    return age;
}

//функция возврата площади
int Estate::getArea() {
    return area;
}

//функция кол-ва жильцов
int Estate::getResidents() {
    return residents;
}

//функция кол-ва месяцев
int Estate::getMonths() {
    return months;
}

//функция возврата имени
QString Estate::getOwner() {
    return owner;
}

//функция возврата типа жилья
Estate::EstateType Estate::getType() {
    return type;
}

```

### luxuriousapartmentcalc.cpp

```

#include "luxuriousapartmentcalc.h"

luxuriousapartmentcalc::luxuriousapartmentcalc(QObject *parent) :
QObject(parent) {}

//функция для расчета стоимости
//500 - коэффициент для данного типа жилья
int luxuriousapartmentcalc::getCost(Estate *value) {
    return value->getArea() * value->getMonths() * 500;
}

```

### main.cpp

```

#include "widget.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Widget w;
    w.show();
    return a.exec();
}

```

### states.cpp

```

#include "states.h"

```

```

States::States(QObject *parent) : QObject(parent) {
    actualData = nullptr;
}

States::~States() {
    if(actualData){
        delete actualData;
        actualData = nullptr;
    }
    qDeleteAll(array);
    array.clear();
}

//проверка коллекции на пустоту
bool States::hasStates() {
    return !array.isEmpty();
}

//возврат актуального значение
Estate* States::getActualData() {
    return actualData;
}

//добавление данных в коллекцию
void States::add(Estate *value) {
    array.append(value);
}

//обновление актуального значения
void States::undo() {
    if (hasStates()) {
        actualData = NULL;
    }else{
        actualData = array.last();
        array.removeLast();
    }
}

```

## townhousecalc.cpp

```

#include "townhousecalc.h"

townhousecalc::townhousecalc(QWidget *parent) : QWidget(parent){}

//функция расчета стоимости жилья
//420 - коэффициент для данного типа жилья
int townhousecalc::getCost(Estate *value) {
    return value->getArea() * value->getMonths() * 420;
}

```

## widget.cpp

```

#include "widget.h"
#include "ui_widget.h"

Widget::Widget(QWidget *parent) : QWidget(parent), ui(new Ui::Widget) {
    ui->setupUi(this);
}

Widget::~Widget() {
    delete ui;
}

```



}

Скриншоты с результатами:

Корнющенко

Имя:	Кирилл
Возрас:	20
Класс жиль:	Квартира эконом-класса
Число проживающих:	4
Площадь	124
Срок страхования:	6 месяцев

Стоимость страхового взноса: 0

Расчитать      Последний запрос

**Вывод:** в ходе выполнение лабораторной работы научились пользоваться средой разработки Qt и обобщили полученные знания по структурным шаблонам проектирования.