

КАФЕДРА №

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

должность, уч. степень, звание

подпись, дата

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №6

«Структурные и поведенческие шаблоны проектирования»

по курсу: Объектно-ориентированное программирование

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

подпись, дата

инициалы, фамилия

1. Цель работы

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

1.1 Задание

Задание 1. Дополните класс Estate конструктором, заполняющим все его поля и функциями чтения полей. Задание 2. Реализуйте методы undo, hasStates, getActualData, add.

Задание 3. Реализуйте собственные функции расчета стоимости в отдельном классе каждая и организуйте их вызов из класса-фасада. Класс-фасад может быть изменен, но передача данных для расчета должна производиться в виде объекта Estate, как показано в примере выше.

Бизнес-логика приложения:

Расчёт стоимости страховки транспортных средств.

VIN номер транспортного средства:

Поле ввода номера транспортного средства.

Стаж вождения:

Чем выше стаж, тем дешевле конечная стоимость страховки.

Для 1, 2, 3 и 4-летнего опыта вождения конкретные коэффициенты, если стаж от 5 лет включительно, либо отсутствует, то рассчитывается по фиксированной ставке.

1 год стажа - $\text{value} \rightarrow \text{getAge()} * 150$

2 года стажа - $\text{value} \rightarrow \text{getAge()} * 50$

3 года стажа - $\text{value} \rightarrow \text{getAge()} * 30$

4 года стажа - $\text{value} \rightarrow \text{getAge()} * 10$

5 и более лет стажа – 5

Класс транспортного средства:

Класс транспортного средства влияет на коэффициент, который будет учитываться при расчёте стоимости страховки.

Транспортное средство ценой до 10 млн. рублей. — Коэффициент X20

Транспортное средство ценой более 10 млн. рублей включительно. — Коэффициент X70

Спецтехника. — Коэффициент X90

Пилотируемое воздушное судно массой более 115 кг. — Коэффициент X150

Количество людей, вписанных в страховку:

Чем больше людей вписано в страховку, тем больше получится финальная стоимость.

Если вписан 1 человек – в расчётную формулу добавляется 100

От 2 до 5 человек включительно - добавляется 250

Более 5 человек - добавляется 500

Количество лошадиных сил транспортного средства:

В зависимости от того сколько лошадиных сил у транспортного средства, определяется коэффициент, необходимый для достоверного подсчёта

Менее 100 Л/С включительно - $\text{value} \rightarrow \text{getHp()} * 3$ – каждая Л/С ТС умножается на 3

От 100 до 150 Л/С включительно - $\text{value} \rightarrow \text{getHp()} * 4$ – каждая Л/С ТС умножается на 4

Более 150 Л/С - $\text{value} \rightarrow \text{getHp()} * 5$ – каждая Л/С ТС умножается на 5

Срок страхования:

Доступно 3 тарифа: 12, 24 и 36 месяцев. Скидки и акции не предусмотрены.

Соответственно, если итоговая стоимость страхования на 12 месяцев X, то для 24 месяцев будет 2X, для 36 месяцев будет 3X.

Итог:

Кнопка «Рассчитать» рассчитывает финальную стоимость страховки.

Кнопка «Последний запрос» вводит данные предыдущего транспортного средства и считает итоговую цену его страхования.

2. Форма

Owner_label	VIN номер транспортного средства:	ZA9AB01E0PCD39023	Owner
Age_label	Стаж вождения:	6	Age
estateType_label	Класс транспортного средства:	Транспортное средство ценой до 10 млн. рублей.	estateType
residents_label	Количество людей вписанных в страховку:	1	residents
hp_label	Количество лошадиных сил транспортного средства:	110	hp
period_label	Срок страхования:	12 месяцев	period
cost_label	Итог:	0	cost

Рассчитать Последний запрос

btnCalc btnUndo

Были использованы следующие виджеты:

8 – label: age_label, cost_label, estateType_label, hp_label, owner_label, period_label, residents_label, которые относятся к наименованию, пояснительным надписям для полей, выводу цены

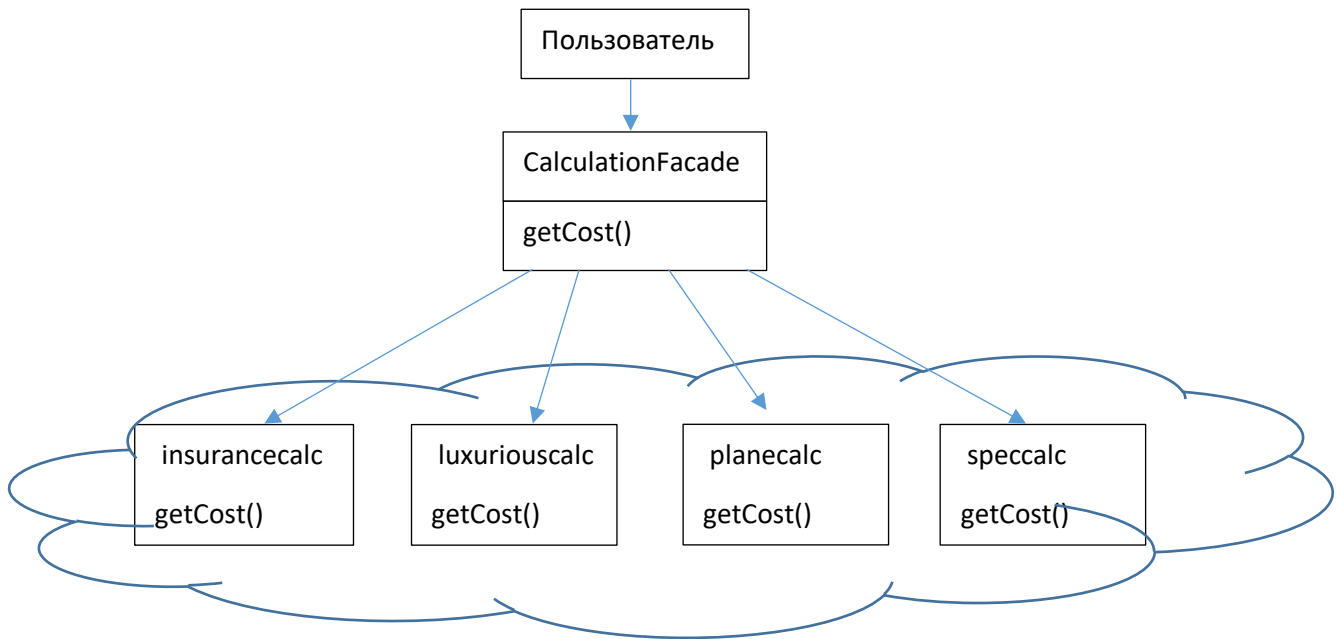
2 – pushButton: btnCalc, btnUndo, которые отвечают за расчёт суммы и отображение значений предыдущего транспортного средства

2 – comboBox: estateType, period: для выбора класса и срока

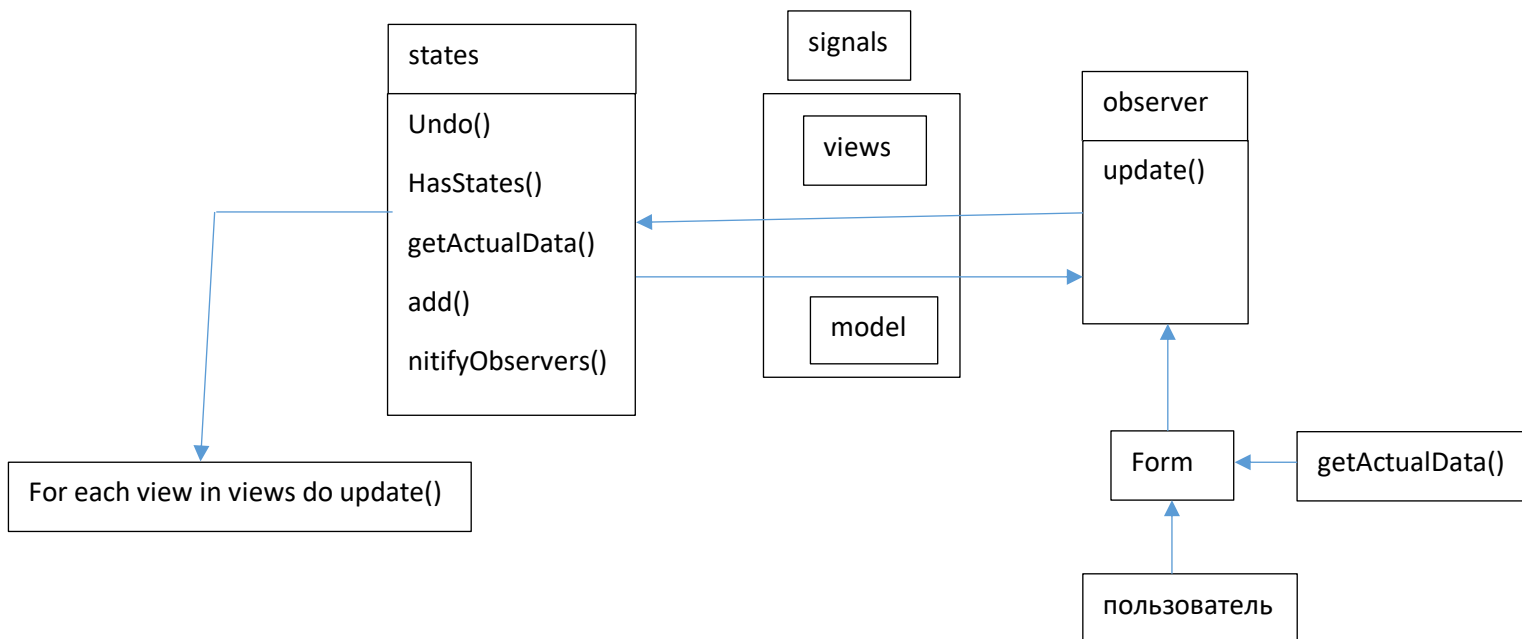
4 – lineEdit: age, hp, owner, residents: ввод индивидуальных значений

3. Диаграммы:

calculationFacade



Observer



4.1 Текст программы (calculationfacade.h)

```
#ifndef CALCULATIONFACADE_H
#define CALCULATIONFACADE_H

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

```

#include <luxuriouscalc.h>
#include <speccalc.h>
#include <planecalc.h>
class CalculationFacade : public QObject
{
    Q_OBJECT
public:
    explicit CalculationFacade(QObject *parent = nullptr);
    static int getCost(Estate *value);
};
#endif // CALCULATIONFACADE_H

текст программы (estate.h)
// Листинг файла estate.h
#ifndef ESTATE_H
#define ESTATE_H
#include <QObject>
class Estate : public QObject
{
    Q_OBJECT
public:
    enum EstateType {
        ECONOM,
        LUXURIOUS,
        SPEC,
        PLANE
    };
    Estate(const int age, const int hp, const int residents,
           const int months, const EstateType type, const QString owner);
    explicit Estate(QObject *parent = nullptr);
    EstateType getType() const;
    int getAge() const;
    int getHp() const;
    int getResidents() const;
    int getMonths() const;
    QString getOwner() const;

```

```

private:
    int age;
    int hp;
    int residents;
    int months;
    EstateType type;
    QString owner;
};

#endif // ESTATE_H

текст программы (insurancecalc.h)
#ifndef INSURANCECALC_H
#define INSURANCECALC_H
#include <QObject>
#include <estate.h>
class insurancecalc : public QObject
{
    Q_OBJECT
public:
    explicit insurancecalc(QObject *parent = nullptr);
    static int getCost(Estate *value);

signals:
};

#endif // INSURANCECALC_H

текст программы (luxuriouscalc.h)
#ifndef LUXURIOUSCALC_H
#define LUXURIOUSCALC_H
#include <QObject>
#include <estate.h>
class luxuriouscalc : public QObject
{
    Q_OBJECT
public:
    explicit luxuriouscalc(QObject *parent = nullptr);
    static int getCost(Estate *value);

```

```

signals:
};

#endif // LUXURIOUSCALC_H

текст программы (planecalc.h)

#ifndef PLANEALC_H
#define PLANEALC_H
#include <QObject>
#include <estate.h>
class planecalc : public QObject
{
    Q_OBJECT
public:
    explicit planecalc(QObject *parent = nullptr);
    static int getCost(Estate *value);
signals:
};

#endif // PLANEALC_H

текст программы (speccalc.h)

#ifndef SPECCALC_H
#define SPECCALC_H
#include <QObject>
#include <estate.h>
class speccalc : public QObject
{
    Q_OBJECT
public:
    explicit speccalc(QObject *parent = nullptr);
    static int getCost(Estate *value);
signals:
};

#endif // SPECCALC_H

текст программы (states.h)

#ifndef STATES_H
#define STATES_H
#include <QObject>

```

```

#include <estate.h>

class States : public QObject
{
    Q_OBJECT
public:
    //для 2 задания
    explicit States(QObject *parent = nullptr);
    ~States();
    void undo();
    bool hasStates();
    Estate *getActualData();
    void add(Estate *value);
    int getSize();
signals:
    void notifyObservers();
private:
    QList<Estate *> array;
    Estate *actualData;
};
#endif

```

текст программы (widget.h)

```

#ifndef WIDGET_H
#define WIDGET_H
#include <QWidget>
#include <states.h>
#include <estate.h>
#include <calculationfacade.h>
#include <myexception.h>
namespace Ui {
class Widget;
}
class Widget : public QWidget
{
    Q_OBJECT
public:

```



```
explicit Widget(QWidget *parent = 0);
```

```
~Widget();
```

```
public slots:
```

```
void update();
```

```
private slots:
```

```
void btnCalcPressed();
```

```
void btnUndoPressed();
```

```
private:
```

```
//для 4 задания
```

```
Estate *processForm();
```

```
void fillForm(Estate *value);
```

```
void showCost(Estate *value);
```

```
private:
```

```
Ui::Widget *ui;
```

```
States info;
```

```
};
```

```
#endif // WIDGET_H
```

Текст программы (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 = insurancecalc::getCost(value);
```

```
            break;
```

```
        }
```

```
        case Estate::EstateType::LUXURIOUS:
```

```
        {
```

```
            cost = luxuriouscalc::getCost(value);
```

```
            break;
```

```

    }
    case Estate::EstateType::SPEC:
    {
        cost = speccalc::getCost(value);
        break;
    }
    case Estate::EstateType::PLANE:
    {
        cost = planecalc::getCost(value);
        break;
    }
    default:
    {
        cost = -1;
        break;
    }
}

return cost;

```

Текст программы (estate.cpp)

//для 1 задания

```
#include "estate.h"
```

```
#include <widget.h>
```

```
Estate::Estate(QObject *parent)
```

```
: QObject(parent)
```

```
{
```

```
    age = 6;
```

```
    hp = 110;
```

```
    residents = 1;
```

```
    months = 12;
```

```
    owner = "ZA9AB01E0PCD39023";
```

```
    type = Estate::ECONOM;
```

```
}
```

```
Estate::Estate(const int age, const int hp, const int residents,
```

```
               const int months, const EstateType type, const QString owner)
```

```

{
    this->age = age;
    this->hp = hp;
    this->residents = residents;
    this->months = months;
    this->owner = owner;
    this->type = type;
}

Estate::EstateType Estate::getType() const
{
    return this->type;
}

int Estate::getAge() const
{
    return this->age;
}

int Estate::getHp() const
{
    return this->hp;
}

int Estate::getResidents() const
{
    return this->residents;
}

int Estate::getMonths() const
{
    return this->months;
}

QString Estate::getOwner() const
{
    return this->owner;
}

```

Текст программы (insurancecalc.cpp) //аналогичны luxurioscalc.cpp/plane.cpp/speccalc.cpp

```
#include "insurancecalc.h"
```

```

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

int insurancecalc::getCost(Estate *value)
{
    if (value->getAge() == 0) //стаж
    {
        if (value->getResidents() == 1) //кол-во людей в страховке
        {
            if (value->getHp() <= 100) //кол-во лс
                return ((250 + 100 + value->getHp()*3) * (value->getMonths()/6))*20;
            if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
                return ((250 + 100 + value->getHp()*4) * (value->getMonths()/6))*20;
            if (value->getHp() > 150) //кол-во лс
                return ((250 + 100 + value->getHp()*5) * (value->getMonths()/6))*20;
        }

        if (value->getResidents() >= 2 && value->getResidents() <= 5) //кол-во людей в страховке
        {
            if (value->getHp() <= 100) //кол-во лс
                return ((250 + 250 + value->getHp()*3) * (value->getMonths()/6))*20;
            if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
                return ((250 + 250 + value->getHp()*4) * (value->getMonths()/6))*20;
            if (value->getHp() > 150) //кол-во лс
                return ((250 + 250 + value->getHp()*5) * (value->getMonths()/6))*20;
        }

        if (value->getResidents() > 5) //кол-во людей в страховке
        {
            if (value->getHp() <= 100) //кол-во лс
                return ((250 + 500 + value->getHp()*3) * (value->getMonths()/6))*20;
            if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
                return ((250 + 500 + value->getHp()*4) * (value->getMonths()/6))*20;
            if (value->getHp() > 150) //кол-во лс

```

```

        return ((250 + 500 + value->getHp()*5) * (value->getMonths()/6))*20;
    }
}

if (value->getAge() <= 1) //стаж
{
    if (value->getResidents() == 1) //кол-во людей в страховке
    {
        if (value->getHp() <= 100) //кол-во лс
            return ((value->getAge()*150 + 100 + value->getHp()*3) * (value->getMonths()/6))*20;
        if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
            return ((value->getAge()*150 + 100 + value->getHp()*4) * (value->getMonths()/6))*20;
        if (value->getHp() > 150) //кол-во лс
            return ((value->getAge()*150 + 100 + value->getHp()*5) * (value->getMonths()/6))*20;
    }

    if (value->getResidents() >= 2 && value->getResidents() <= 5) //кол-во людей в страховке
    {
        if (value->getHp() <= 100) //кол-во лс
            return ((value->getAge()*150 + 250 + value->getHp()*3) * (value->getMonths()/6))*20;
        if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
            return ((value->getAge()*150 + 250 + value->getHp()*4) * (value->getMonths()/6))*20;
        if (value->getHp() > 150) //кол-во лс
            return ((value->getAge()*150 + 250 + value->getHp()*5) * (value->getMonths()/6))*20;
    }

    if (value->getResidents() > 5) //кол-во людей в страховке
    {
        if (value->getHp() <= 100) //кол-во лс
            return ((value->getAge()*150 + 500 + value->getHp()*3) * (value->getMonths()/6))*20;
        if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
            return ((value->getAge()*150 + 500 + value->getHp()*4) * (value->getMonths()/6))*20;
        if (value->getHp() > 150) //кол-во лс
            return ((value->getAge()*150 + 500 + value->getHp()*5) * (value->getMonths()/6))*20;
    }
}

```

```

}

if (value->getAge() == 2) //стаж
{
    if (value->getResidents() == 1) //кол-во людей в страховке
    {
        if (value->getHp() <= 100) //кол-во лс
            return ((value->getAge()*50 + 100 + value->getHp()*3) * (value->getMonths()/6))*20;
        if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
            return ((value->getAge()*50 + 100 + value->getHp()*4) * (value->getMonths()/6))*20;
        if (value->getHp() > 150) //кол-во лс
            return ((value->getAge()*50 + 100 + value->getHp()*5) * (value->getMonths()/6))*20;
    }
    if (value->getResidents() >= 2 && value->getResidents() <= 5) //кол-во людей в страховке
    {
        if (value->getHp() <= 100) //кол-во лс
            return ((value->getAge()*50 + 250 + value->getHp()*3) * (value->getMonths()/6))*20;
        if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
            return ((value->getAge()*50 + 250 + value->getHp()*4) * (value->getMonths()/6))*20;
        if (value->getHp() > 150) //кол-во лс
            return ((value->getAge()*50 + 250 + value->getHp()*5) * (value->getMonths()/6))*20;
    }
    if (value->getResidents() > 5) //кол-во людей в страховке
    {
        if (value->getHp() <= 100) //кол-во лс
            return ((value->getAge()*50 + 500 + value->getHp()*3) * (value->getMonths()/6))*20;
        if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
            return ((value->getAge()*50 + 500 + value->getHp()*4) * (value->getMonths()/6))*20;
        if (value->getHp() > 150) //кол-во лс
            return ((value->getAge()*50 + 500 + value->getHp()*5) * (value->getMonths()/6))*20;
    }
}

if (value->getAge() == 3) //стаж
{

```

```

if (value->getResidents() == 1) //кол-во людей в страховке
{
    if (value->getHp() <= 100) //кол-во лс
        return ((value->getAge()*30 + 100 + value->getHp()*3) * (value->getMonths()/6))*20;
    if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
        return ((value->getAge()*30 + 100 + value->getHp()*4) * (value->getMonths()/6))*20;
    if (value->getHp() > 150) //кол-во лс
        return ((value->getAge()*30 + 100 + value->getHp()*5) * (value->getMonths()/6))*20;
}

if (value->getResidents() >= 2 && value->getResidents() <= 5) //кол-во людей в страховке
{
    if (value->getHp() <= 100) //кол-во лс
        return ((value->getAge()*30 + 250 + value->getHp()*3) * (value->getMonths()/6))*20;
    if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
        return ((value->getAge()*3050 + value->getHp()*4) * (value->getMonths()/6))*20;
    if (value->getHp() > 150) //кол-во лс
        return ((value->getAge()*30 + 250 + value->getHp()*5) * (value->getMonths()/6))*20;
}

if (value->getResidents() > 5) //кол-во людей в страховке
{
    if (value->getHp() <= 100) //кол-во лс
        return ((value->getAge()*30 + 500 + value->getHp()*3) * (value->getMonths()/6))*20;
    if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
        return ((value->getAge()*30 + 500 + value->getHp()*4) * (value->getMonths()/6))*20;
    if (value->getHp() > 150) //кол-во лс
        return ((value->getAge()*30 + 500 + value->getHp()*5) * (value->getMonths()/6))*20;
}

}

if (value->getAge() == 4) //стаж
{
    if (value->getResidents() == 1) //кол-во людей в страховке
    {
        if (value->getHp() <= 100) //кол-во лс

```

```

        return ((value->getAge()*10 + 100 + value->getHp()*3) * (value->getMonths()/6))*20;
    if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
        return ((value->getAge()*10 + 100 + value->getHp()*4) * (value->getMonths()/6))*20;
    if (value->getHp() > 150) //кол-во лс
        return ((value->getAge()*10 + 100 + value->getHp()*5) * (value->getMonths()/6))*20;
    }
    if (value->getResidents() >= 2 && value->getResidents() <= 5) //кол-во людей в страховке
    {
        if (value->getHp() <= 100) //кол-во лс
            return ((value->getAge()*10 + 250 + value->getHp()*3) * (value->getMonths()/6))*20;
        if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
            return ((value->getAge()*10 + 250 + value->getHp()*4) * (value->getMonths()/6))*20;
        if (value->getHp() > 150) //кол-во лс
            return ((value->getAge()*10 + 250 + value->getHp()*5) * (value->getMonths()/6))*20;
        }
    if (value->getResidents() > 5) //кол-во людей в страховке
    {
        if (value->getHp() <= 100) //кол-во лс
            return ((value->getAge()*10 + 500 + value->getHp()*3) * (value->getMonths()/6))*20;
        if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
            return ((value->getAge()*10 + 500 + value->getHp()*4) * (value->getMonths()/6))*20;
        if (value->getHp() > 150) //кол-во лс
            return ((value->getAge()*10 + 500 + value->getHp()*5) * (value->getMonths()/6))*20;
        }
    }
    if (value->getAge() >= 5) //стаж
    {
        if (value->getResidents() == 1) //кол-во людей в страховке
        {
            if (value->getHp() <= 100) //кол-во лс
                return ((5 + 100 + value->getHp()*3) * (value->getMonths()/6))*20;
            if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
                return ((5 + 100 + value->getHp()*4) * (value->getMonths()/6))*20;
            if (value->getHp() > 150) //кол-во лс

```



```

        return ((5 + 100 + value->getHp()*5) * (value->getMonths()/6))*20;
    }
    if (value->getResidents() >= 2 && value->getResidents() <= 5) //кол-во людей в страховке
    {
        if (value->getHp() <= 100) //кол-во лс
            return ((5 + 250 + value->getHp()*3) * (value->getMonths()/6))*20;
        if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
            return ((5 + 250 + value->getHp()*4) * (value->getMonths()/6))*20;
        if (value->getHp() > 150) //кол-во лс
            return ((5 + 250 + value->getHp()*5) * (value->getMonths()/6))*20;
    }
    if (value->getResidents() > 5) //кол-во людей в страховке
    {
        if (value->getHp() <= 100) //кол-во лс
            return ((5 + 500 + value->getHp()*3) * (value->getMonths()/6))*20;
        if (value->getHp() > 100 && value->getHp() <= 150) //кол-во лс
            return ((5 + 500 + value->getHp()*4) * (value->getMonths()/6))*20;
        if (value->getHp() > 150) //кол-во лс
            return ((5 + 500 + value->getHp()*5) * (value->getMonths()/6))*20;
    }
}
}
}

```

Текст программы (main.cpp)

```

#include "widget.h"
#include <QApplication>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Widget w;
    w.show();
    return a.exec();
}

```

Текст программы (states.cpp)

```

//реализуем 2 задание
#include "states.h"

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

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

void States::add(Estate *value)
{
    array.append(value);
}

bool States::hasStates()
{
    return !(States::array.empty());
}

Estate *States::getActualData()
{
    return actualData;
}

void States::undo()
{

```

```

    if (hasStates())
    {
        array.pop_back();
        actualData = array.last();
        emit notifyObservers();
    }
    else
        actualData = nullptr;
}

int States::getSize()
{
    return array.size();
}

```

Текст программы (widget.cpp)

```

//реализуем задание
#include "widget.h"
#include "ui_widget.h"

Widget::Widget(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Widget),
    info(this)
{
    ui->setupUi(this);
    ui->btnUndo->setEnabled(false);
    // регистрация слушателя
    connect(&info, SIGNAL(notifyObservers()), this, SLOT(update()));
    connect(ui->btnCalc, SIGNAL(pressed()), this, SLOT(btnCalcPressed()));
    connect(ui->btnUndo, SIGNAL(pressed()), this, SLOT(btnUndoPressed()));
}

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

// public slots

```

```

void Widget::update()
{
    auto value = info.getActualData();
    if(value != nullptr){
        fillForm(value);
    }

    // update btnUndo state
    ui->btnUndo->setEnabled(info.hasStates());

    // setting value to NULL
    value = nullptr;
}

// private slots
void Widget::btnCalcPressed()
{
    try {
        if (ui->age->text().toInt() == 0 || ui->hp->text().toInt() == 0
            || ui->owner->text() == "" || ui->residents->text().toInt() == 0){
            throw myException("Не все поля формы заполнены.");
        }

        Estate *estate = new Estate(ui->age->text().toInt(),
                                     ui->hp->text().toInt(),
                                     ui->residents->text().toInt(),
                                     (ui->period->currentIndex() + 1) * 6,
                                     (Estate::EstateType)ui->estateType->currentIndex(),
                                     ui->owner->text());

        info.add(estate);
        ui->btnUndo->setEnabled(true);
        showCost(estate);
    }

    catch (myException &ex) {
        QMessageBox err;
        err.setText(ex.what());
        err.setWindowTitle("Внимание!");
        err.exec();
    }
}

```

```

        return;
    }
}

void Widget::btnUndoPressed()
{
    if (info.getSize() > 1){
info.undo();
    }
}

// private
Estate *Widget::processForm()
{
    return new Estate();
}

void Widget::fillForm(Estate *value)
{
    ui->age->setText(QString::number(info.getActualData()->getAge()));
    ui->hp->setText(QString::number(info.getActualData()->getHp()));
    ui->residents->setText(QString::number(info.getActualData()->getResidents()));
    ui->owner->setText(info.getActualData()->getOwner());
    ui->period->setCurrentIndex((info.getActualData()->getMonths() / 6) - 1);
    switch (info.getActualData()->getType())
    {
        case Estate::ECONOM:
        {
            ui->estateType->setCurrentIndex(0);
            break;
        }
        case Estate::LUXURIOUS:
        {
            ui->estateType->setCurrentIndex(1);
            break;
        }
    }
}

```

```

        case Estate::SPEC:
        {
            ui->estateType->setCurrentIndex(2);
            break;
        }
        case Estate::PLANE:
        {
            ui->estateType->setCurrentIndex(3);
            break;
        }
    }

    ui->cost->setText(QString::number(CalculationFacade::getCost(info.getActualData())));

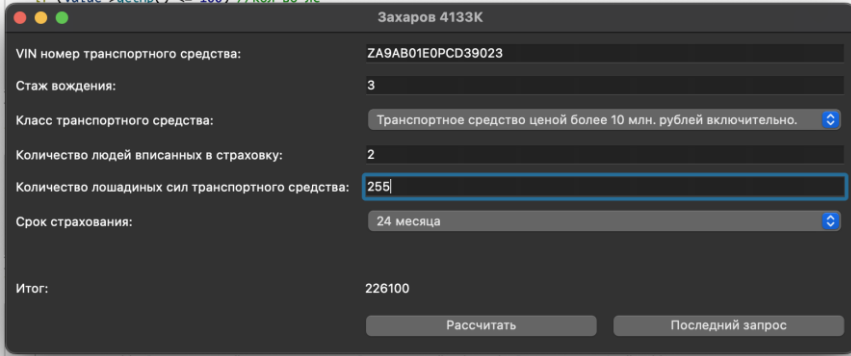
}

void Widget::showCost(Estate *value)
{
    ui->cost->setText(QString::number(CalculationFacade::getCost(value)));
}

```

4. Пример выполнения программы

1 запрос:



VIN номер транспортного средства:	ZA9AB01E0PCD39023
Стаж вождения:	3
Класс транспортного средства:	Транспортное средство ценой более 10 млн. рублей включительно.
Количество людей вписанных в страховку:	2
Количество лошадиных сил транспортного средства:	255
Срок страхования:	24 месяца
Итог:	226100

Рассчитать Последний запрос

2 запрос:

Field	Value
VIN номер транспортного средства:	ZA9AB01E0PCD39023
Стаж вождения:	7
Класс транспортного средства:	Пилотируемое воздушное судно массой более 115 кг.
Количество людей вписанных в страховку:	5
Количество лошадиных сил транспортного средства:	350
Срок страхования:	12 месяцев
Итого:	300750

Использование функции «Последний запрос»

Field	Value
VIN номер транспортного средства:	ZA9AB01E0PCD39023
Стаж вождения:	3
Класс транспортного средства:	Транспортное средство ценой более 10 млн. рублей включительно.
Количество людей вписанных в страховку:	2
Количество лошадиных сил транспортного средства:	255
Срок страхования:	24 месяца
Итого:	226100

5. Анализ результатов и выводы

В результате выполнения лабораторной работы были изучены принципы построения приложений с графическим интерфейсом, используя библиотеку Qt, применив на практике знания базовых синтаксических конструкций языка C++ и объектно-ориентированного программирования.