ГУАП

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Старший преподаватель		Е.О. Шумова
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №8
Описание классов и порождение объектов

по курсу: ОБЪЕКТНО ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	4134к		Костяков Н.А.
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2022

Цель работы

Научиться на практике применять паттерны проектирования.

Предметная область - кинопрокат

Сущности

Фильм

id	название	автор	год

Юзер

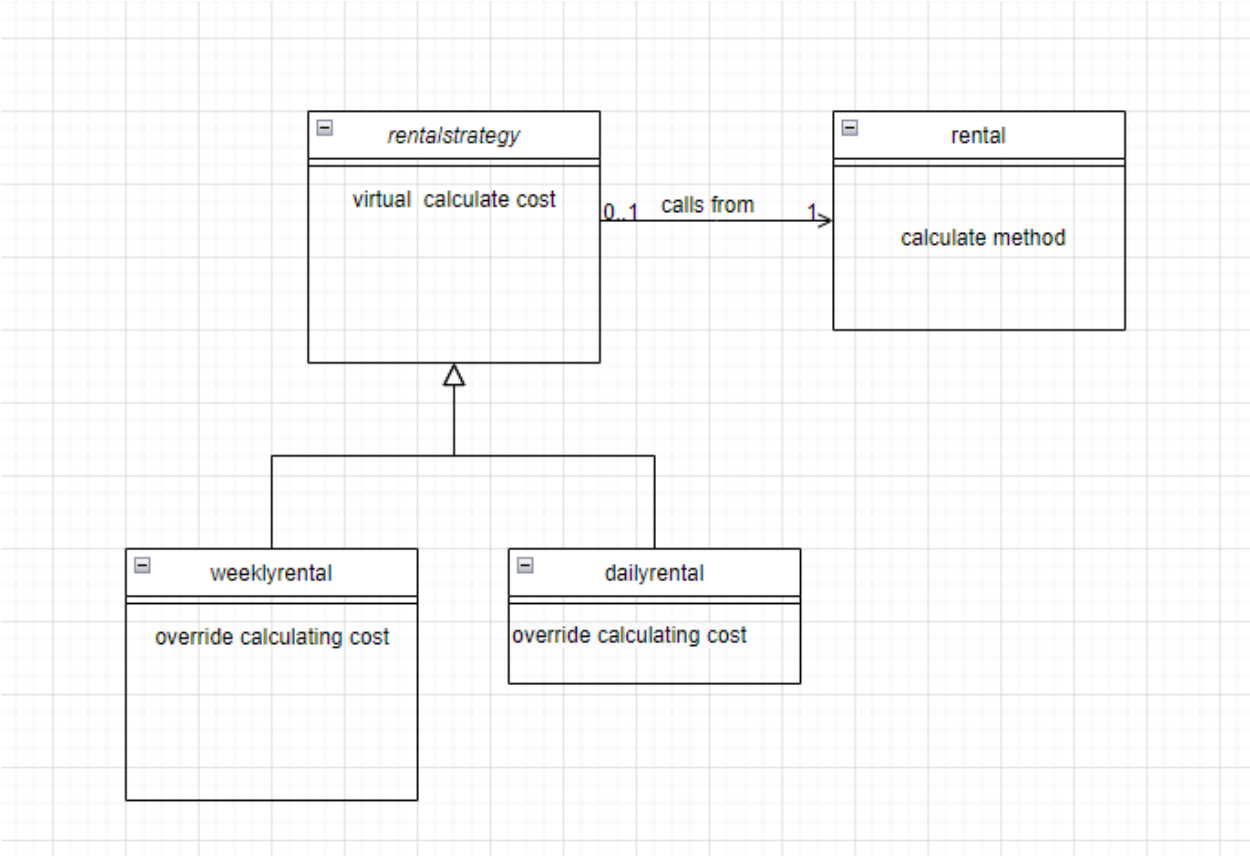
id	ФИО

аренда

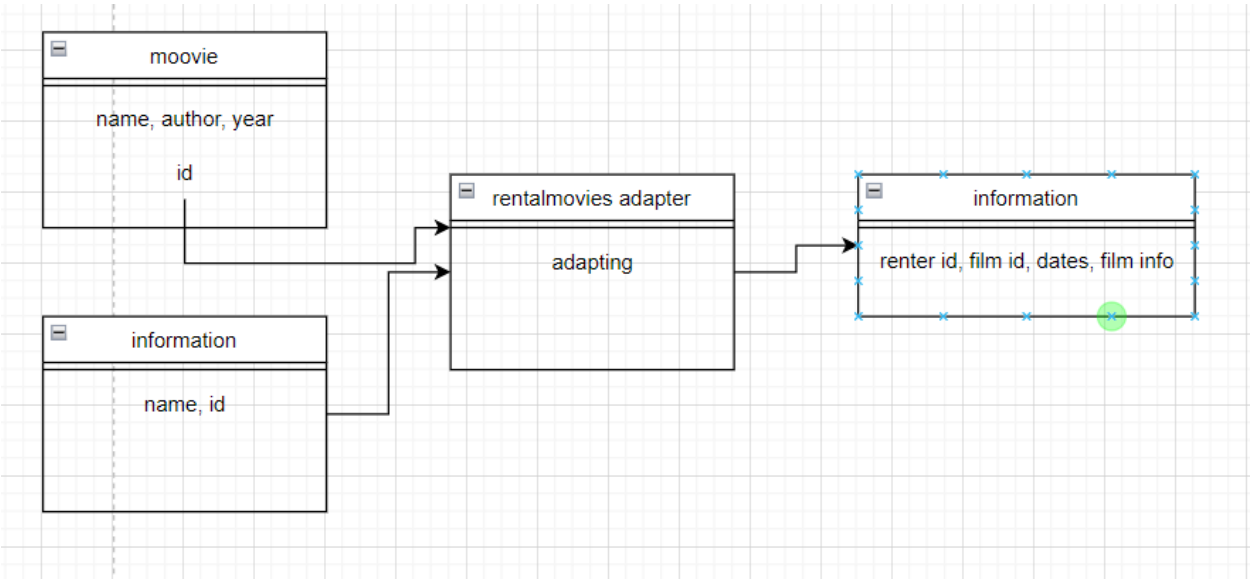
id	userId	filmID	Date_take	Date_return

Диаграммы

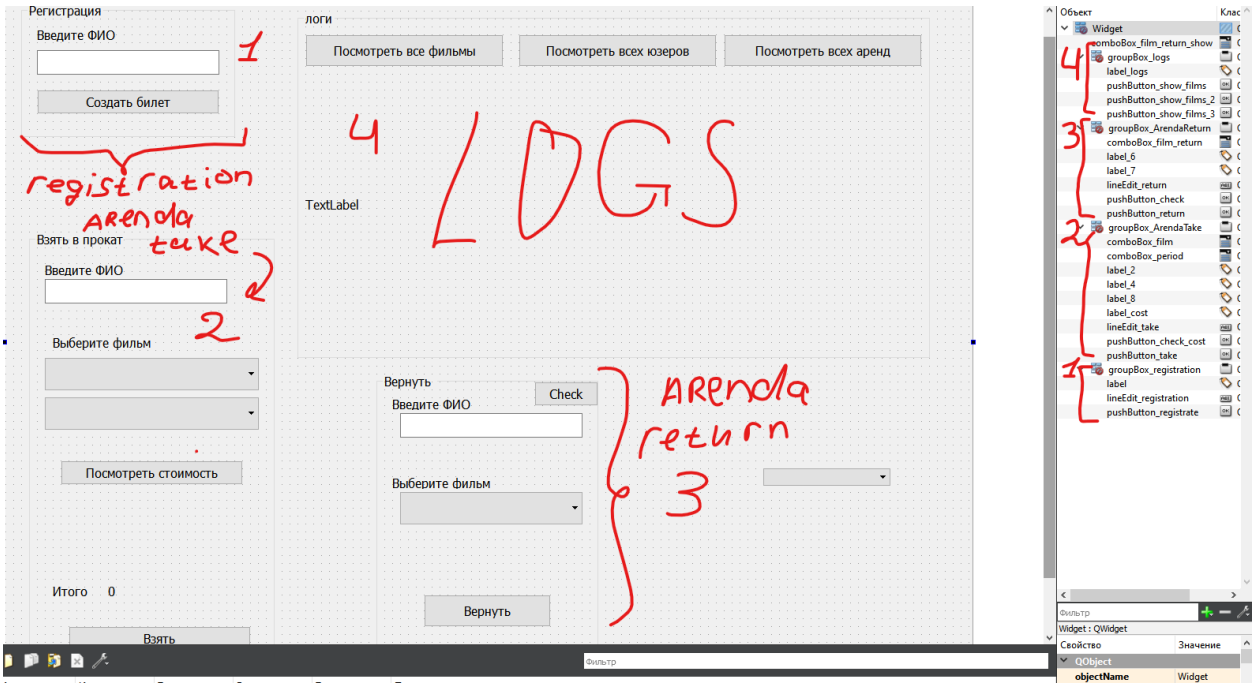
factory



adapter



Общий вид программы



Листинг программы

dailyrentalstrategy.h

```
#ifndef DAILYRENTALSTRATEGY_H
#define DAILYRENTALSTRATEGY_H
#include "rentalstrategy.h"
class DailyRentalStrategy : public IRentalStrategy {
public:
    double CalculateRentalCost(int rentalTime) override {
        return rentalTime * 5.0;
    }
};
#endif // DAILYRENTALSTRATEGY_H
```

dailyrentalstrategyh.cpp

```
#include "dailyrentalstrategyh.h"

dailyrentalstrategyh::dailyrentalstrategyh()
{

}
```

main.cpp

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

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Widget w;
```

```
    w.show();

    return a.exec();
}
```

movies.h

```
#ifndef MOVIES_H
#define MOVIES_H
#include <string>


class movie{
public:
    int number;
    int id, year;
    std::string author, name;
    bool state; //0 - на складе, 1 - выдан

    movie* next= nullptr;
    movie(int num=0, int i=0, int y=0, std::string auth=0, std::string n=0, bool stat
=1){number=num;id=i; year = y; author = auth; name = n; state = stat;}

};

class movies{
private:
    int count;
    movie* head = nullptr;

public:
    void add(int y, std::string auth, std::string n);
    movie* get(int index);
    int get_length(){return count;}

    movies(){
        add(1999, "Стивен Кинг","Зеленая миля");
        add(1994, "Френк Дарабонт","Побег из Шоушенка");
        add(1994, "Роберт Земекис","Форест Гамп");
    }
};

movie* movies::get(int index){
    if(count<index) return nullptr;
    movie* current = head;
    int i = 0;
    while (i<index){
        current=current->next;
        i++;
    }
    return current;
}
```

```
void movies::add(int y, std::string auth, std::string n){
    int number = get_length();
    int id =1+ number*7;
    movie* sub = new movie(number, id, y, auth, n ,1);

    if(head==nullptr){
        head = sub;
        count++;
        return;
    }
    movie* current = head;
    while(current->next!=nullptr){
        current = current->next;
    }
    current->next = sub;
    count++;
    return;
}

#endif // MOVIES_H
```

rental.h

```
#ifndef RENTAL_H
#define RENTAL_H
#include "rentalstrategy.h"
#include "weeklystrategy.h"
#include "dailyrentalstrategy.h"



class Rental {
private:
    IRentalStrategy* _rentalStrategy;

public:
    Rental(IRentalStrategy* rentalStrategy) : _rentalStrategy(rentalStrategy) {}

    double CalculateRentalCost(int rentalTime) {
        return _rentalStrategy->CalculateRentalCost(rentalTime);
    }
};

//int main() {
    //auto dailyRental = new Rental(new DailyRentalStrategy());
    //std::cout << "Daily rental cost for 3 days: " << dailyRental->CalculateRentalCost(3) <<
std::endl;

   // auto weeklyRental = new Rental(new WeeklyRentalStrategy());
   // std::cout << "Weekly rental cost for 2 weeks: " << weeklyRental->CalculateRentalCost(2)
<< std::endl;

   // delete dailyRental;
```

```
  // delete weeklyRental;

  // return 0;
//}
#endif // RENTAL_H
```

rentalmovieadapter.h

```
#ifndef RENTALMOVIEADAPTER_H
#define RENTALMOVIEADAPTER_H
#include "rentalmovies.h"
#include "movies.h"
#include "subscriber.h"

class RentalMoviesAdapter : public RentalMovies {
private:
    movie movieInfo;
    subscriber customerInfo;

public:
    RentalMoviesAdapter(movie movieInfo, subscriber customerInfo) {
        this->movieInfo = movieInfo;
        this->customerInfo = customerInfo;
    }

    void rentMovie(string movieName) override {
            }

    void returnMovie(string movieName) override {
        cout << "The movie " << movieName << " has been returned by " << customerInfo.name
<< "." << endl;
    }
};

#endif // RENTALMOVIEADAPTER_H
```

rentalmovies.h

```
#ifndef ARENDA_H
#define ARENDA_H
#include <string>
#include "movies.h"
#include "subscriber.h"
#include <iostream>
class info{
public:
    int renterId;
    int filmId;
    int OperId;
    std::string date_take;
    std::string date_return;
    std::string film_name;
    info* next = nullptr;
```

```
    info(movie* mov, subscriber* sub){
        renterId = sub->id;
        filmId = mov->id;
        film_name = mov->name;
        date_take = "16.07.23";
        date_return = "17.07.23";
    }
};

class RentalMovies {
private:
    int count;


public:
    info* head= nullptr;
    int get_length(){return count;}
    std::string show();
    void add(info *input);
    info* find(std::string name, int sub_id);
    void return_film(std::string name, subscriber* sub);
    std::string* get_all_from_sub(subscriber* sub);
    info* get(int index);
};

info* RentalMovies::get(int index){

        if(count<index) return nullptr;
        info* current = head;
        int i = 0;
        while (i<index){
            current=current->next;
            i++;
        }
        return current;

}

void RentalMovies::return_film(std::string name, subscriber* sub){
    info* to_remove = find(name, sub->id);
    info* current = head;
    if(current == to_remove){
        head = head->next;
        count--;
        return;

    }

    while(current->next!= to_remove){
        current = current->next;
    }
    current->next=current->next->next;
```

```
    count--;
}

info* RentalMovies::find(std::string name, int sub_id){
    info* current = head;
    while (current!=nullptr){
        if(current->film_name==name && current->renterId == sub_id) return current;
        current= current->next;
    }
    return nullptr;
}

std::string RentalMovies::show(){
    std::string res = "";
    info* current = head;
    while(current!=nullptr){
        res += std::to_string(current->OperId)+": "+" "+current->film_name+", ";
        current = current->next;
    }
    return res;
}

void RentalMovies::add(info* input){
    if(!input) return;
    int number = get_length();
    int id = 1+number*21;
    input->OperId= id;
    info* sub = input;

    if(head==nullptr){
        head = sub;
        count++;
        return;
    }
    info* current = head;
    while(current->next!=nullptr){
        current = current->next;
    }
    current->next = sub;
    count++;
    return;
}
#endif
```

rentalstrategy.h

```
#ifndef RENTALSTRATEGY_H
#define RENTALSTRATEGY_H

class IRentalStrategy {
public:
    virtual double CalculateRentalCost(int rentalTime) = 0;
};
```

```
#endif // RENTALSTRATEGY_H
```

subscriber.h

```
#ifndef SUBSCRIBER_H
#define SUBSCRIBER_H
#include <string>
class subscriber{
public:
    int number;
    int id;
    std::string fio;

    subscriber* next = nullptr;
    subscriber(std::string fio, int number, int id){
        this->fio = fio;
        this->id = id;
        this->number = number;
    }

};

class sub_table{
  private:
    subscriber* head = nullptr;
    int count = 0;

  public:

    int get_length();
    std::string show();
    void registers(std::string name);
    subscriber* find(std::string name);
    subscriber* get(int index);
};

subscriber* sub_table::get(int index){
    subscriber* current = head;
    for(int i = 0; i<index; i++){
        current=current->next;
    }
    return current;
}

std::string sub_table::show(){

    std::string res= "";
    subscriber* current = head;
    while(current!=nullptr){
        res = res + current->fio+", ";
        current = current->next;
```

```
    }
    return res;
}


subscriber* sub_table::find(std::string name){

    if(name == "")return nullptr;
    subscriber* current = head;
    while(current!=nullptr){
        if (current->fio == name){
            return current;
        }
        current = current->next;
    }
    return nullptr;
}

void sub_table::registers(std::string name){

    int number = get_length();
    int id = 1+ number*13;
    subscriber* sub = new subscriber(name, number, id);

    if(head==nullptr){
        head = sub;
        count++;
        return;
    }
    subscriber* current = head;
    while(current->next!=nullptr){
        current = current->next;
    }
    current->next = sub;
    count++;
    return;

}

int sub_table::get_length(){
    return count;
}

#endif // SUBSCRIBER_H
```

weeklystrategy.h

```
#ifndef WEEKLYRENTALSTRATEGY_H
#define WEEKLYRENTALSTRATEGY_H
#include "rentalstrategy.h"
class WeeklyRentalStrategy : public IRentalStrategy {
public:
    double CalculateRentalCost(int rentalTime) override {
```

```
        return rentalTime * 20.0;
    }
};
#endif // WEEKLYRENTALSTRATEGY_H
```

widget.cpp

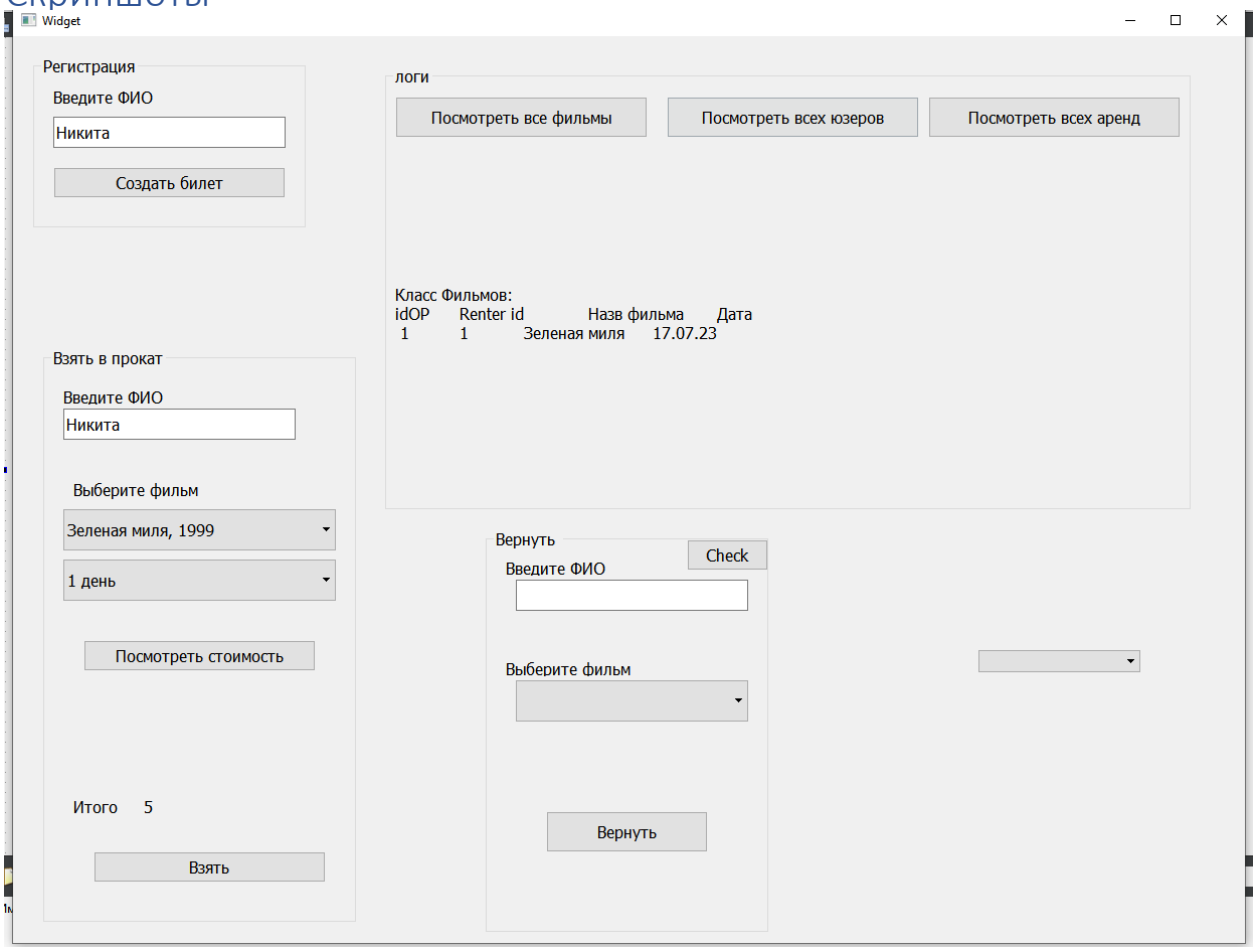
```
#include "widget.h"
#include "ui_widget.h"
#include "subscriber.h"
#include "movies.h"
#include "rental.h"
#include "rentalmovies.h"
sub_table subs;
movies films;
RentalMovies RM;


Widget::Widget(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Widget)
{
    ui->setupUi(this);
    for(int i = 0; i< films.get_length(); i++){
        std::string name = films.get(i)->name + ", " + std::to_string(films.get(i)->year);
        ui->comboBox_film->addItem(QString::fromStdString(name));

    }
    ui->comboBox_period->addItem("1 день");
    ui->comboBox_period->addItem("2 дня");
    ui->comboBox_period->addItem("1 неделя");
}

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

void Widget::on_pushButton_registrate_clicked()
{
    std::string name = ui->lineEdit_registration->text().toStdString();
    subs.registers(name);

}

void Widget::on_pushButton_check_cost_clicked()
{
    double cost;
    auto rent = new Rental(new DailyRentalStrategy());
    int period = ui->comboBox_period->currentIndex();
    switch (period) {
    case 0:
        rent = new Rental(new DailyRentalStrategy());
```

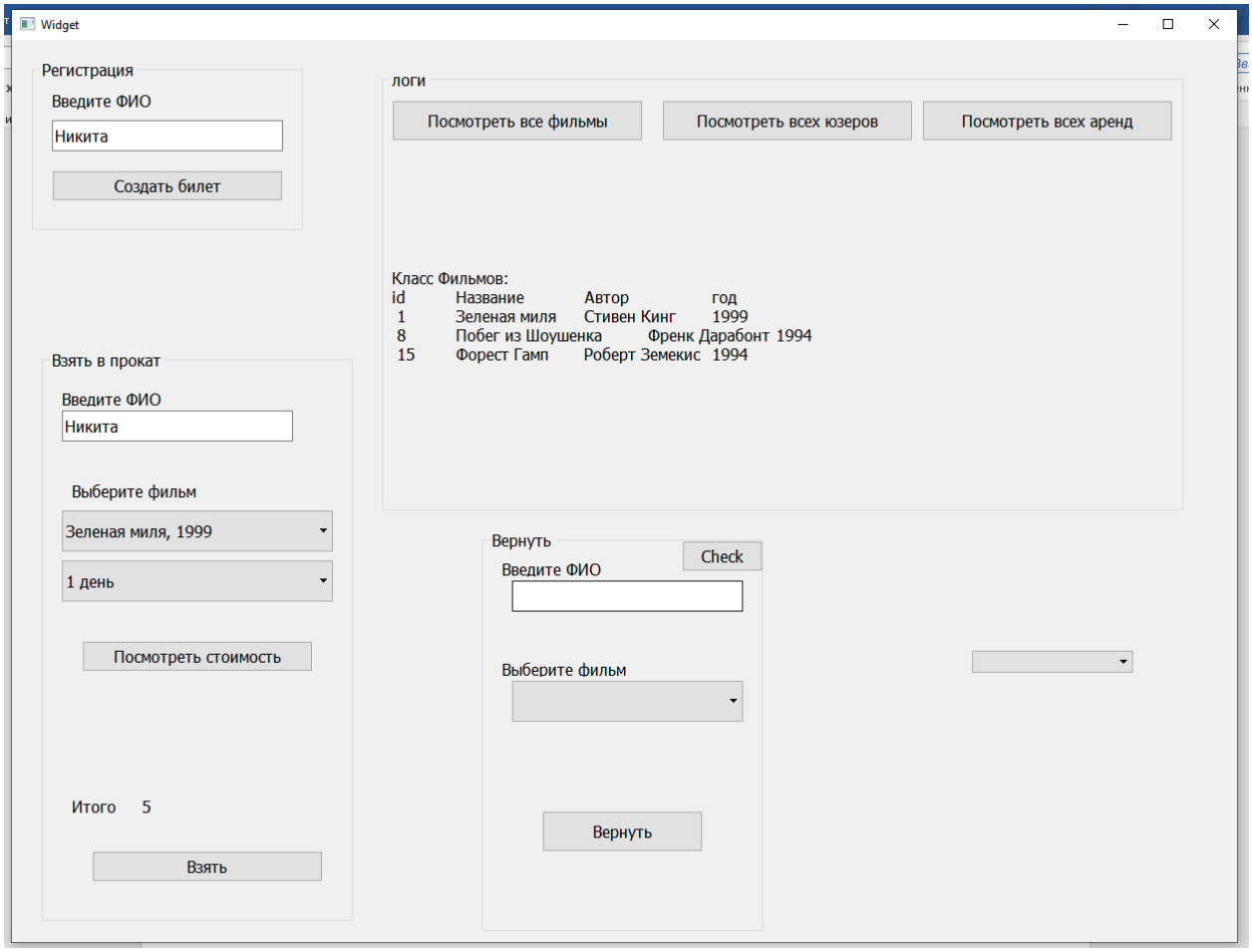
```
        cost = rent->CalculateRentalCost(1);
        break;
    case 1:
         rent = new Rental(new DailyRentalStrategy());
        cost = rent->CalculateRentalCost(2);
        break;
    case 2:
        rent = new Rental(new WeeklyRentalStrategy());
        cost = rent->CalculateRentalCost(1);
        break;
    default:
        break;
    }

    ui->label_cost->setText(QString::number(cost));
}

void Widget::on_pushButton_take_clicked()
{
    std::string name = ui->lineEdit_take->text().toStdString();
    subscriber* sub = subs.find(name);
    movie* mov = films.get(ui->comboBox_film->currentIndex());
    info * input = new info(mov, sub);
    RM.add(input);


   // std::string addition = mov->name;
   // ui->comboBox_film_return->addItem(QString::fromStdString(addition));
}



void Widget::on_pushButton_return_clicked()
{
    std::string name = ui->lineEdit_take->text().toStdString();
    subscriber* sub = subs.find(name);
    std::string mov = ui->comboBox_film_return->currentText().toStdString();


    RM.return_film(mov, sub);
   // ui->comboBox_film_return->removeItem(0);

}

void Widget::on_pushButton_check_clicked()
{
    ui->comboBox_film_return->clear();
    std::string name = ui->lineEdit_return->text().toStdString();
    if(name=="") return;
    subscriber* sub = subs.find(name);
    if (sub==nullptr) return;
```

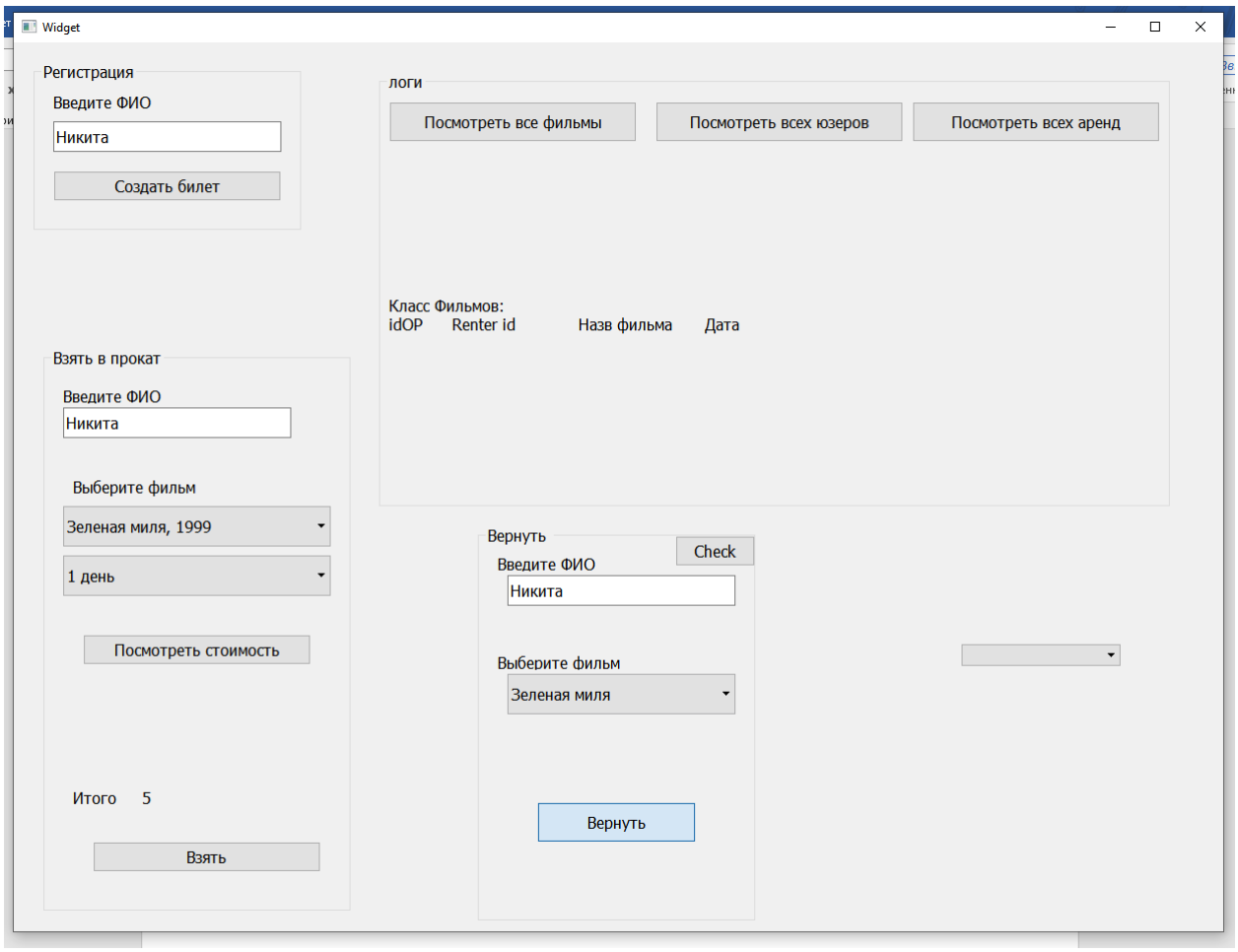
```
    info* current = RM.head;
    std::string res ;
    while (current){
        if(current->renterId == sub->id)
        {res= current->film_name;
        ui->comboBox_film_return->addItem(QString::fromStdString(res));}
        current = current->next;
    }


}

void Widget::on_pushButton_show_films_clicked()
{
    ui->label_logs->clear();
    std::string outp = "Класс Фильмов:\nid\tНазвание\tАвтор\t\tгод";

    for(int i = 0; i<films.get_length();i++){
        outp+="\n "+std::to_string(films.get(i)->id)+"\t"+films.get(i)->name+"\t"+films.get(i)-
>author+"\t"+std::to_string(films.get(i)->year);
    }
    ui->label_logs->setText(QString::fromStdString(outp));
}

void Widget::on_pushButton_show_films_2_clicked()
{
    ui->label_logs->clear();
    std::string outp = "Класс юзеров:\nid\tФИО";

    for(int i = 0; i<subs.get_length();i++){
        outp+="\n "+std::to_string(subs.get(i)->id)+"\t"+subs.get(i)->fio;
    }
    ui->label_logs->setText(QString::fromStdString(outp));
}

void Widget::on_pushButton_show_films_3_clicked()
{
    ui->label_logs->clear();
    std::string outp = "Класс Фильмов:\nidOP\tRenter id\tНазв фильма\tДата";

    for(int i = 0; i<RM.get_length();i++){
        outp+="\n "+std::to_string(RM.get(i)->OperId)+"\t"+std::to_string(RM.get(i)-
>renterId)+"\t"+RM.get(i)->film_name+"\t"+RM.get(i)->date_return;
    }
    ui->label_logs->setText(QString::fromStdString(outp));
}

widget.h
#ifndef WIDGET_H
#define WIDGET_H

#include <QWidget>
```

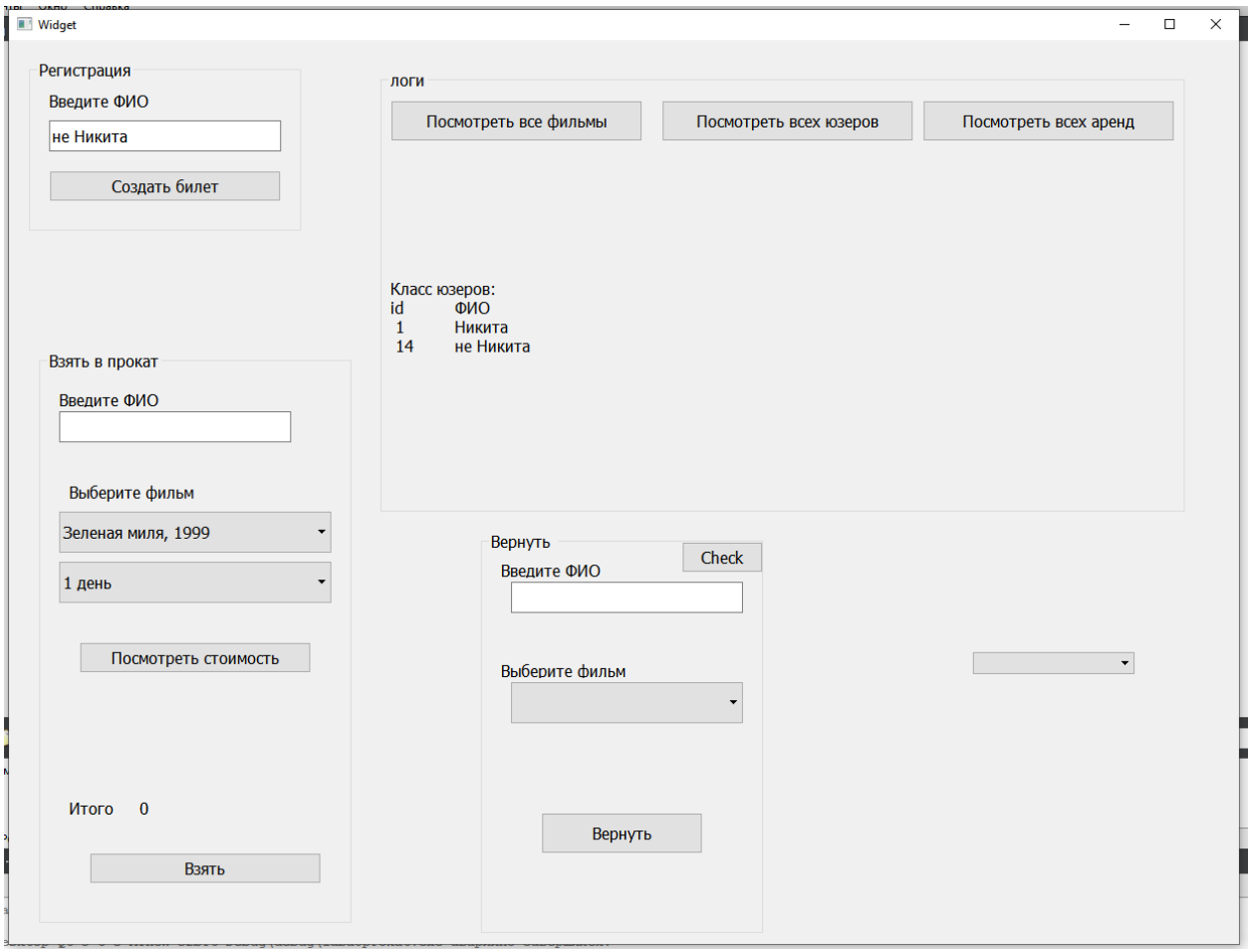
```
#include <string>

namespace Ui {
class Widget;
}

class Widget : public QWidget
{
   Q_OBJECT


public:

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

private slots:
   void on_pushButton_registrate_clicked();

   void on_pushButton_check_cost_clicked();

   void on_pushButton_take_clicked();



   void on_pushButton_return_clicked();

   void on_pushButton_check_clicked();

   void on_pushButton_show_films_clicked();

   void on_pushButton_show_films_2_clicked();

   void on_pushButton_show_films_3_clicked();

private:
   Ui::Widget *ui;
};

#endif // WIDGET_H
```

Скриншоты


Widget
Регистрация
Введите ФИО
Никита
Создать билет
логи
Посмотреть все фильмы
Посмотреть всех юзеров
Посмотреть всех аренд
Класс Фильмов:
idOP Renter id Назв фильма Дата
1 1 Зеленая миля 17.07.23
Взять в прокат
Введите ФИО
Никита
Выберите фильм
Зеленая миля, 1999
1 день
Посмотреть стоимость
Итого 5
Взять
Вернуть
Check
Введите ФИО
Выберите фильм
Вернуть


Widget
Регистрация
Введите ФИО
Никита
Создать билет
логи
Посмотреть все фильмы
Посмотреть всех юзеров
Посмотреть всех аренд
Класс Фильмов:
id Название Автор год
1 Зеленая миля Стивен Кинг 1999
8 Побег из Шоушенка Френк Дарабонт 1994
15 Форест Гамп Роберт Земекис 1994
Взять в прокат
Введите ФИО
Никита
Выберите фильм
Зеленая миля, 1999
1 день
Посмотреть стоимость
Итого 5
Взять
Вернуть
Check
Введите ФИО
Выберите фильм
Вернуть


Widget
Регистрация
Введите ФИО
Никита
Создать билет
логи
Посмотреть все фильмы
Посмотреть всех юзеров
Посмотреть всех аренд
Класс Фильмов:
idOP
Renter id
Назв фильма
Дата
Взять в прокат
Введите ФИО
Никита
Выберите фильм
Зеленая миля, 1999
1 день
Посмотреть стоимость
Итого 5
Взять
Вернуть
Check
Введите ФИО
Никита
Выберите фильм
Зеленая миля
Вернуть



Выводы

Я освоил применение паттернов на практики