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

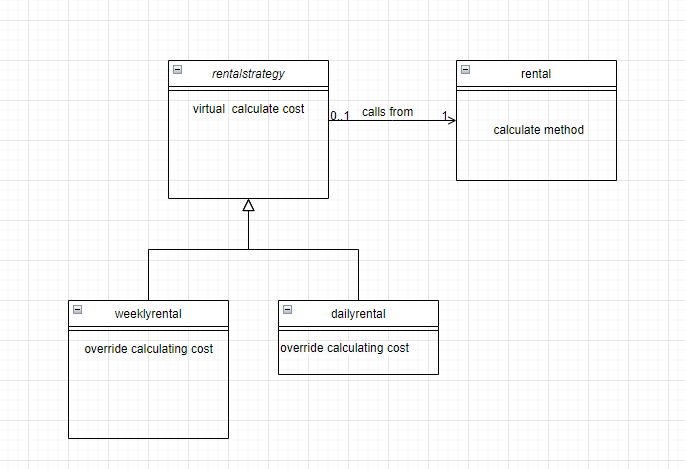
|  |  |
| --- | --- |
| id | ФИО |

аренда

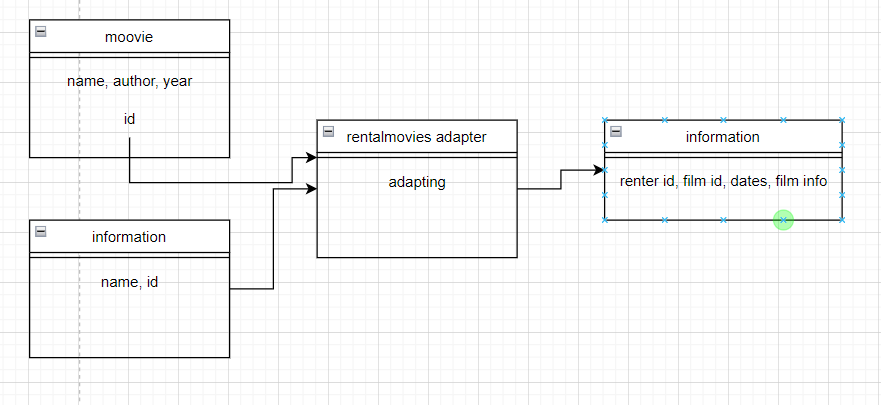
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | userId | filmID | Date\_take | Date\_return |

# Диаграммы

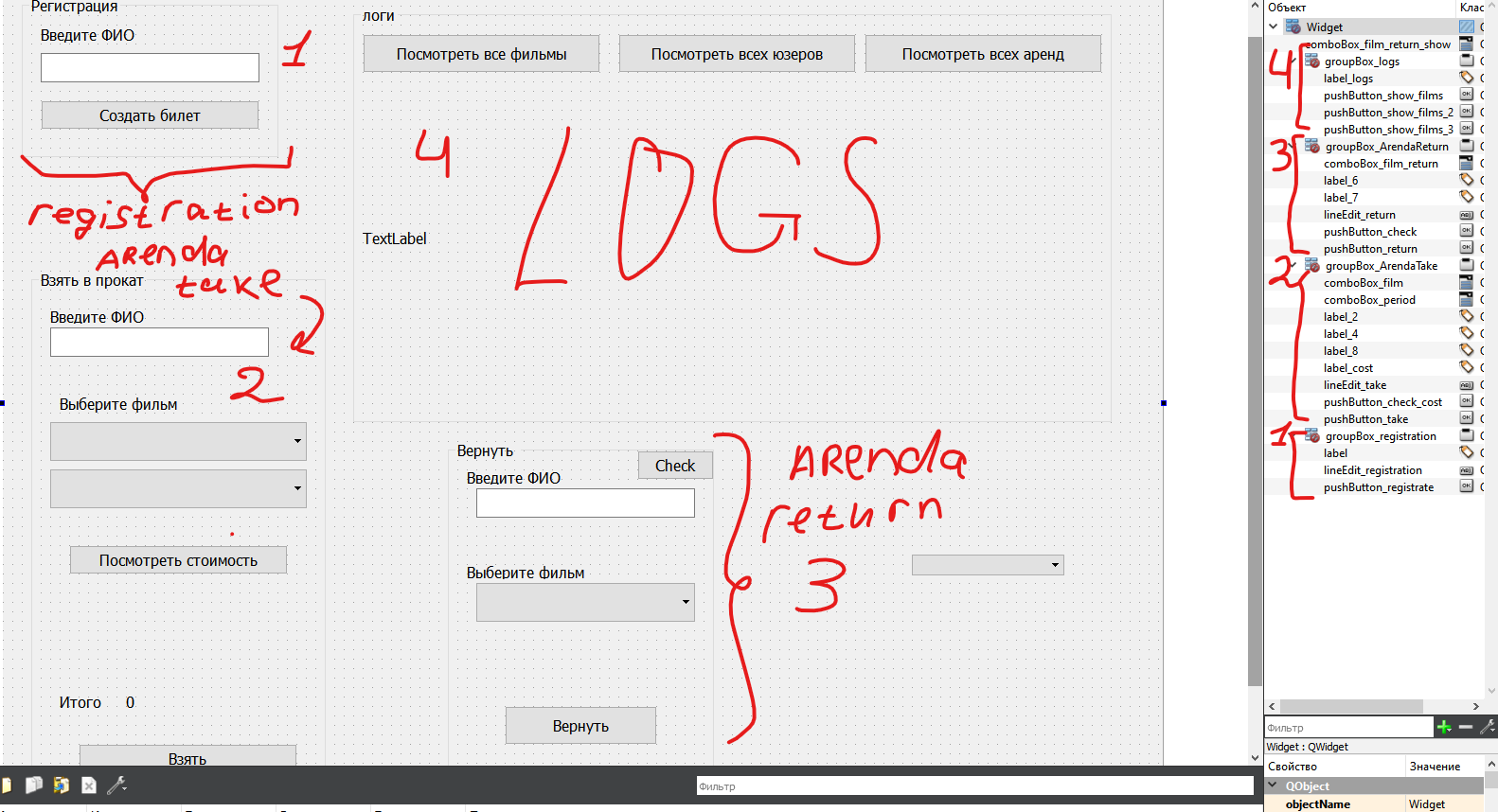
**factory**



**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:**\n**id**\t**Название**\t**Автор**\t\t**год";

**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:**\n**id**\t**ФИО";

**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:**\n**idOP**\t**Renter id**\t**Назв фильма**\t**Дата";

**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();

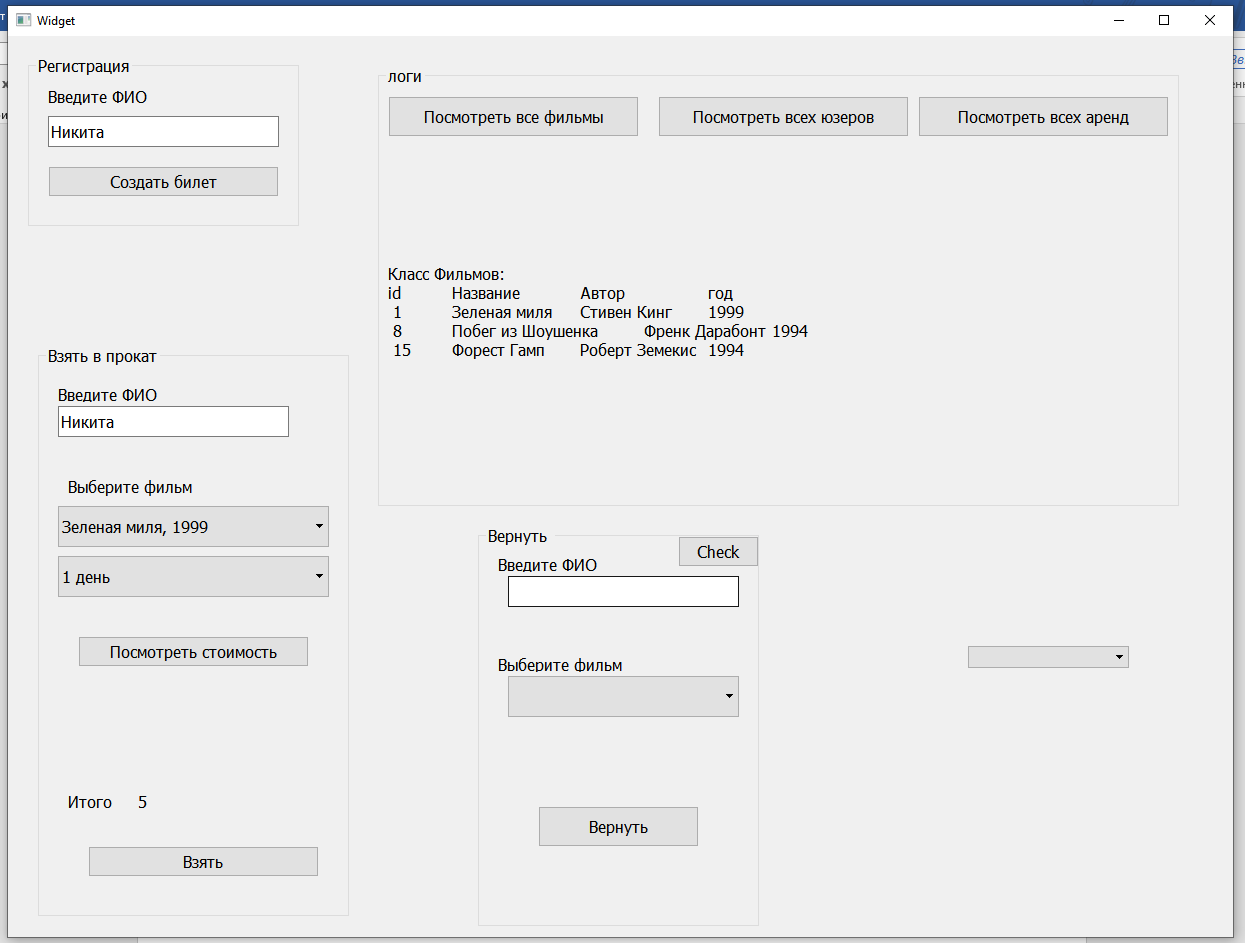
**private**:

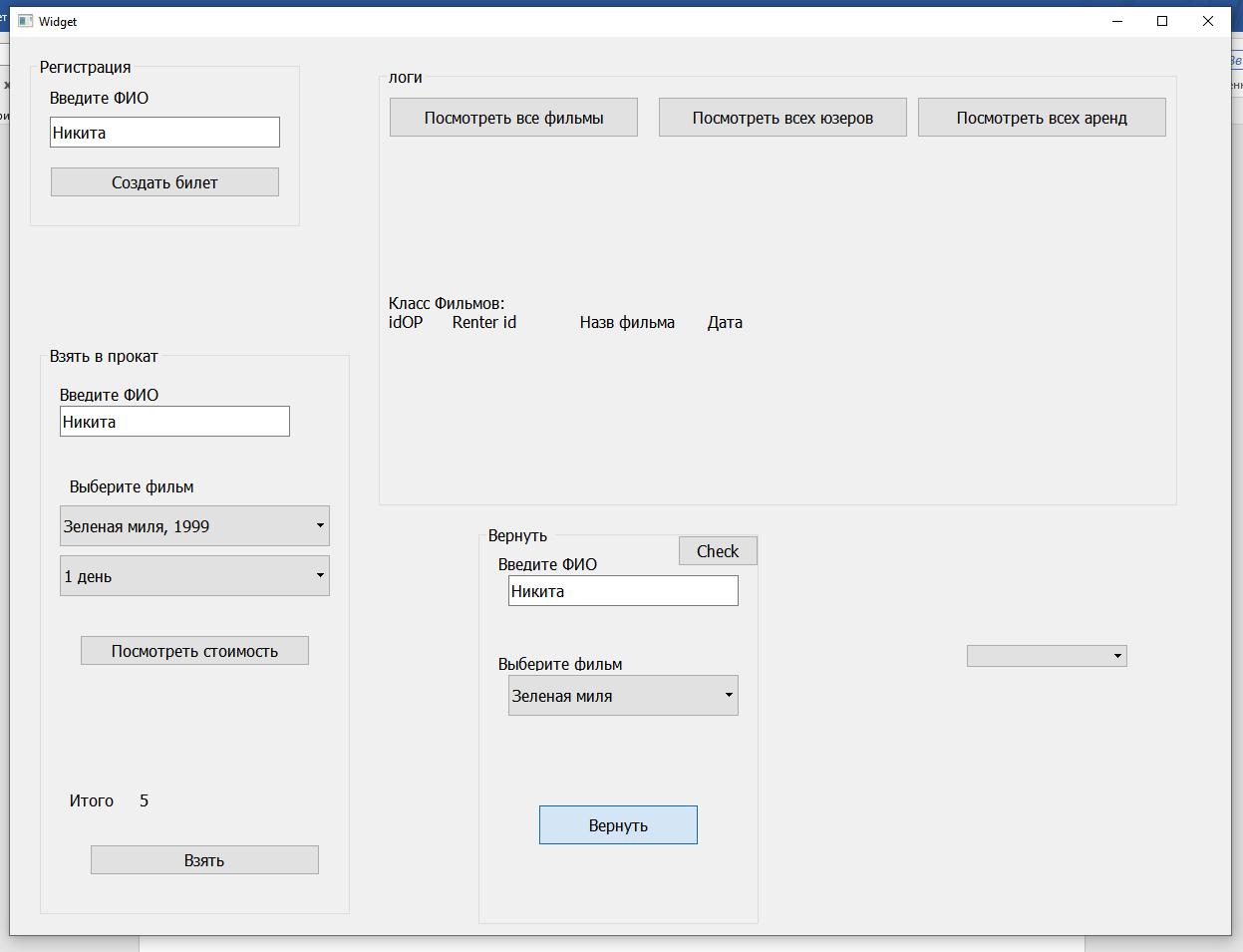
Ui::Widget \*ui;

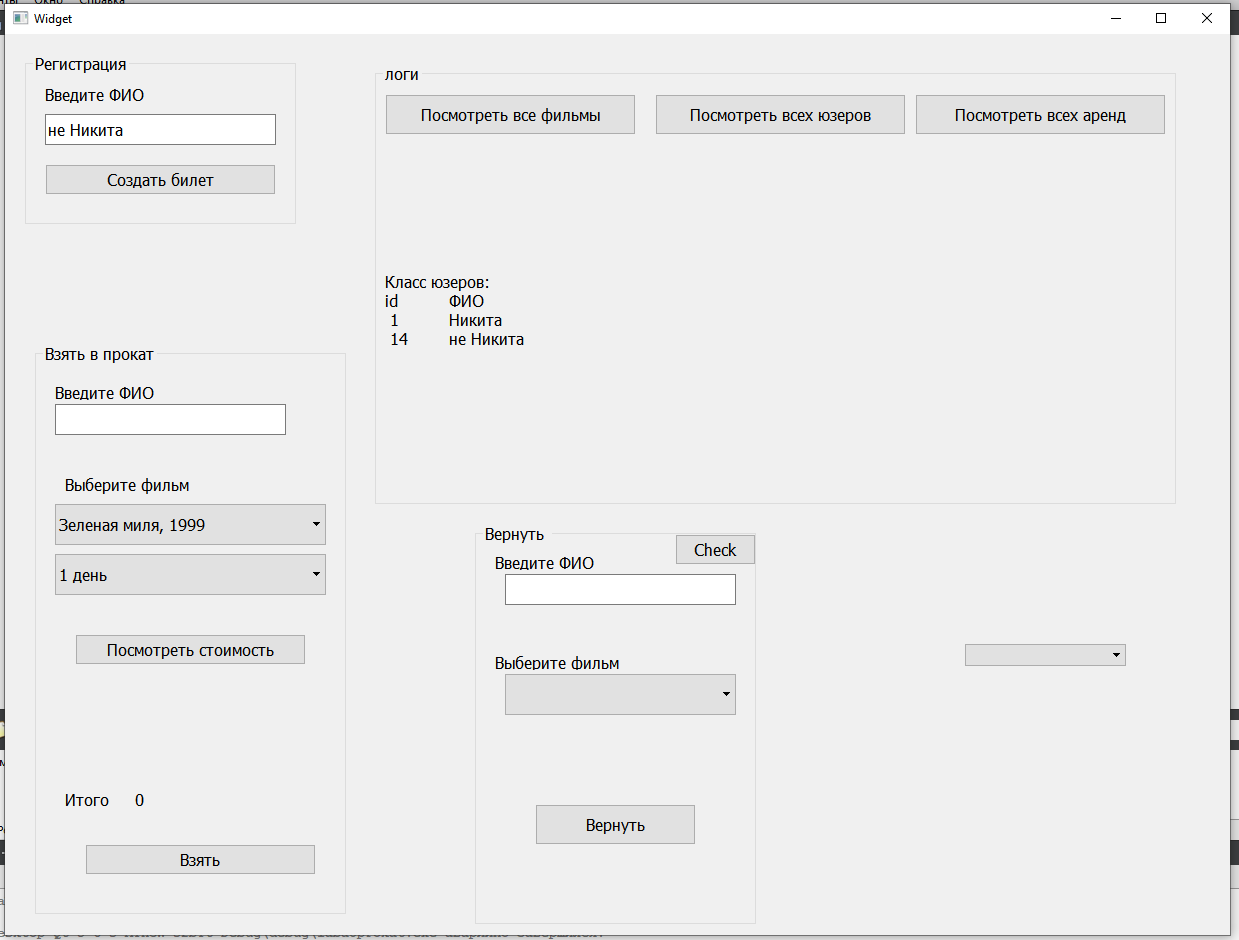
};

#endif *// WIDGET\_H*

# Скриншоты







# Выводы

Я освоил применение паттернов на практики