МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ   
федеральное государственное автономное образовательное учреждение высшего образования «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

КУРСОВОЙ ПРОЕКТ  
ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ст.преподаватель |  |  |  | Е.О. Шумова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
К КУРСОВОМУ ПРОЕКТУ

Разработка приложения для организации взаимодействия объектов  
при заданных критериях

по дисциплине: ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. |  | Z1432K |  |  |  | Г. М. Штанчаев |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ   
федеральное государственное автономное образовательное учреждение высшего образования «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

**Задание  
на курсовой проект по дисциплине  
«Объектно-ориентированное программирование»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студенту группы |  | Z1432K |  | Г. М. Штанчаев |
|  | № группы |  | инициалы, фамилия |

Тема «Разработка приложения для организации взаимодействия объектов  
при заданных критериях»

Исходные данные: Программа для обеспечения работы кинотеатра

Проект должен содержать:

* анализ предметной области
* разработку классов
* разработку тестового приложения
* оформление пояснительной записки по результатам выполнения проекта
* создание презентации к проекту

Срок сдачи законченного проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ст.преп. Е.О.Шумова

Дата выдачи задания 01.09.2023 г.

# Содержание

[Содержание 3](#_Toc150693461)

[Введение 4](#_Toc150693462)

[1. Постановка задачи 5](#_Toc150693463)

[1.1. Анализ предметной области 5](#_Toc150693464)

[1.2. Формулировка функциональных требований 5](#_Toc150693465)

[2. Проектирование классов 7](#_Toc150693466)

[2.1. Классы сущностей 7](#_Toc150693467)

[2.2. Управляющие классы 18](#_Toc150693468)

[2.3. Используемые паттерны проектирования 20](#_Toc150693469)

[2.4. Диаграмма классов 21](#_Toc150693470)

# Введение

Основной задачей проекта является реализация приложения для обеспечения работы кинотеатра. Программа должна иметь интерфейс пользователя, позволяющий оператору кинотеатра управлять фильмами, сеансами, кинозалами, режиссерами, актерами, клиентами, продавать билеты и анализировать популярность фильмов. Также должна присутствовать возможность сохранения и загрузки всех данных из базы данных.

В цели проекта можно добавить изучение взаимодействия классов в объектно- ориентированном программировании и изучение паттернов проектирования. Основная часть пояснительной̆ записки содержит все необходимые разделы для описания процесса создания приложения.

В разделе «Постановка задачи» анализируется предметная область и формируется техническое.

В разделе «Проектирование классов» отображается процесс создания основной структуры программы, создание классов и их связей.

В разделе «Разработка приложения» в первую очередь подробно описывается пользовательский интерфейс, далее следует процесс разработки реализации классов и тестового приложения.

Раздел «Тестирование» содержит в себе демонстрацию работы описанного приложения в виде рисунков, содержащих скриншоты.

Раздел «Приложение» хранит полный код всех созданных программ.

# 1. Постановка задачи

## 1.1. Анализ предметной области

Предметной областью курсового проекта является программа для обеспечения работы кинотеатра. В данной предметной области необходимо разработать программу, позволяющую управлять фильмами и сеансами, продавать билеты, вести учет клиентов и статистику посещений сеансов. Программа должна иметь графический интерфейс и возможность хранить данные в базе.

Используемые сущности:

* Человек
  + Режиссер
  + Актер
  + Клиент
* Фильм
* Зал
* Билет
* Сеанс
* Кресло (в зале)
* Дата
* Время
* Статистика
* Сохранение и загрузка (для работы с БД)

## 1.2. Формулировка функциональных требований

Программа должна иметь графический интерфейс, предоставляющий оператору кинотеатра следующий функционал:

* Добавлять фильмы
* Добавлять сеансы
* Добавлять актеров и режиссеров
* Добавлять клиентов и управлять ценовой политикой для постоянных клиентов (выдавать скидки)
* Распределять места на сеансе
* Продавать и печатать билеты
* Смотреть статистику по фильмам и сеансам
* Отслеживать популярность фильмов определенных режиссеров
* Сохранять данные в базу и загружать из нее

Технические требования:

* Реализация на языке C++ с помощью среды Qt
* Графический интерфейс реализованный на фреймворке Qt

# 2. Проектирование классов

## 2.1. Классы сущностей

2.1.1. Класс «Hall»

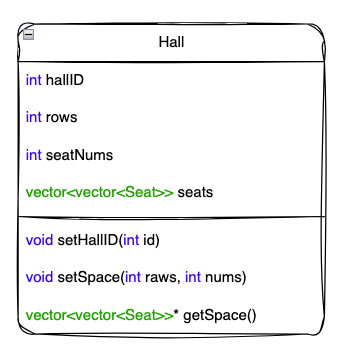


Рисунок 1. Класс "Hall"

Класс «Hall» описывает кинозал.

Описание полей:

* hallID – идентификатор (или номер) зала
* rows – количество рядов в зале
* seatNums – количество кресел в одном ряду
* seats – двумерный массив кресел

Описание методов:

* setHallID – сеттер для id
* setSpace – задает количество рядов и сидений в ряду
* getSpace() – возвращает двумерный массив кресел

2.1.2. Класс «Session»

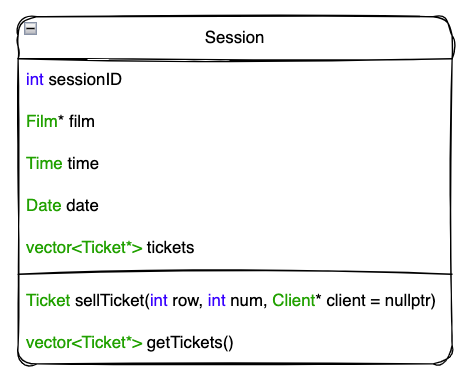


Рисунок 2. Класс "Session"

Класс «Session» описывает сущность сеанса. Этот класс наследуется от класса «Hall».

Описание полей:

* sessionID – идентификатор сеанса
* film – указать на фильм
* time – время начала сеанса
* date – дата проведения сеанса
* tickets – массив билетов на сеанс

Описание методов:

* sellTicket() – продажа билета (бронирует место на сеансе и прикрепляет билет к клиенту, если он зарегистрирован в базе)
* getTickets() – возвращает массив указателей на все билеты сеанса

2.1.3. Класс «Film»

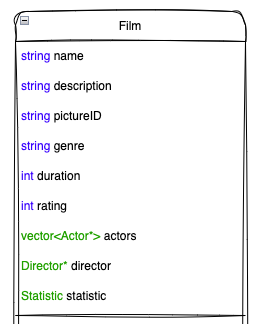
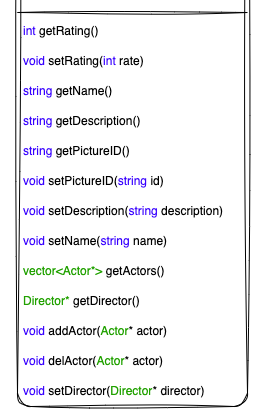
 

Рисунок 3. Класс "Film"

Класс «Film» описывает сущность фильма.

Описание полей:

* name – название фильма
* description – описание фильма
* pictureID – идентификатор изображения в хранилище
* genre – жанр
* duration – длительность в минутах
* rating – рейтинг фильма
* actors – указатели на актерев, играющих в фильме
* director – указатель на режисера фильма
* statistic – статистика по фильму

Описание методов:

* getRating() – выдает рейтинг фильма
* setRating() – задать рейтинг фильму
* getName() – получить название фильма
* setName() – задать название
* getPictureID() – получить идентификатор изображения
* setPictureID() – задать идентификатор
* getDescription() – получить описание
* setDescription() – задать описание
* setDirector() – задать режиссера фильма
* getDirector() – получить режиссера
* getActors() – возвращает массив указателей на актеров, играющих в фильме
* addActor() – добавляет актера
* dellActor() – удаляет актера из фильма

2.1.4. Класс «Seat»

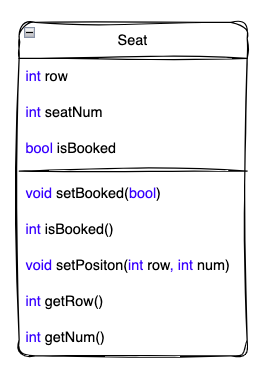


Рисунок 4. Класс "Seat"

Класс «Seat» описывает сущность кресла в кинозале.

Описание полей:

* row – ряд
* seatNum – номер кресла
* isBooked – занято ли место

Описание методов:

* setBooked() – задать занято или свободно
* isBooked() – возвращает true если занято, иначе - false
* setPosition() – задать ряд и номер кресла
* getRow() – получить ряд
* getNum() – получить номер места

2.1.5. Класс «Ticket»



Рисунок 5. Класс "Ticket"

Класс «Ticket» описывает сущность билета.

Описание полей:

* cost – стоимость билета
* session – указатель на сеанс
* seat – указатель на место (на сеансе)

Описание методов:

* setCost() – задать стоимость
* getCost() – получить стоимость
* setSession() – задать сеанс
* getSession() – получить сеанс
* setSeat() – задать кресло
* getSeat() – поучить кресло

2.1.6. Класс «Statistic»

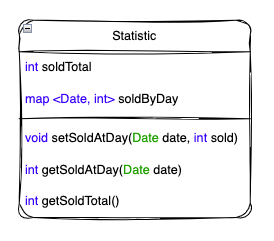


Рисунок 6. Класс "Statistic"

Класс «Statistic» описывает статистику.

Описание полей:

* soldTotal – продано всего
* soldByDay – продано по дням. Словарь, в котором ключ – дата, а значение – количество проданных билетов в эту дату

Описание методов:

* setSoldAtDay() – добавляет значение в словарь. Также увеличивает soldTotal
* getSoldAtDay() – получить количество проданных билетов в определенный день
* getSoldTotal() – получить количество проданных билетов за все время

2.1.7. Класс «Date»



Рисунок 7. Класс "Date"

Класс «Date» описывает дату.

Описание полей:

* day – день
* month – месяц
* year – год

Описание методов:

* setDate() – задать дату
* getDay() – получить день
* getMonth() – получить месяц
* getYear() – получить год
* getDate() - получить всю дату строкой в формате «ДД.ММ.ГГГГ»

2.1.7. Класс «Time»

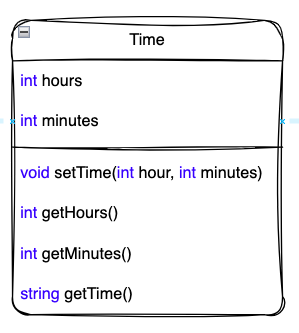


Рисунок 8. Класс "Time"

Класс «Time» описывает время.

Описание полей:

* hours – часы
* minutes – минуты

Описание методов:

* setTime() – задать время
* getHours() – получить часы
* getMinutes() – получить минуты
* getTime() – получить время строкой в формате «ЧЧ:ММ»

2.1.8. Класс «Client»

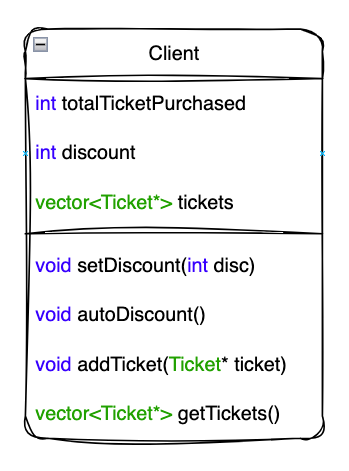


Рисунок 9. Класс "Client"

Класс «Client» описывает сущность клиента. Наследуется от класса «Human»

Описание полей:

* totalTicketPurchased – количество купленных билетов
* discount – скидка в процентах
* tickets – массив указателей на купленные билеты

Описание методов:

* setDiscount() – задать скидку
* autoDiscount() – рассчитать скидку автоматически (по количеству купленных билетов)
* addTicket() – добавить билет в список купленных данным клиентом
* getTickets() – выдает все билеты клиента

2.1.9. Класс «Actor»

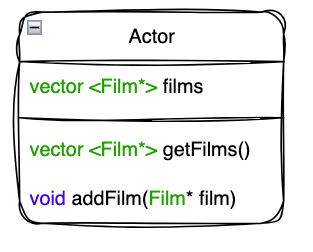


Рисунок 10. Класс "Actor"

Класс «Actor» описывает сущность актера. Наследуется от класса «Human»

Описание полей:

* films – массив указателей на фильмы, в которых играет актер

Описание методов:

* addFilm() – добавить фильм в список
* getFilms() – выдает список всех фильмов с этим актером

2.1.10. Класс «Director»

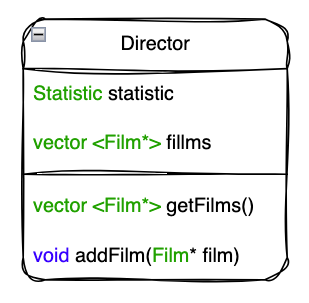


Рисунок 11. Класс "Director"

Класс «Director» описывает сущность режиссера. Наследуется от класса «Human»

Описание полей:

* films – массив указателей на фильмы, снятые этим режиссером
* statistic – статистика по всем фильмам этого режиссера

Описание методов:

* addFilm() – добавить фильм в список
* getFilms() – выдает список всех фильмов этого режиссера

2.1.11. Класс «Human»

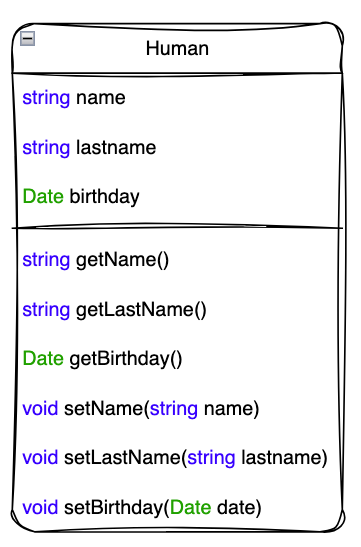


Рисунок 12. Класс "Human"

Класс «Human» описывает сущность человека.

Описание полей:

* name – имя
* lastname – фамилия
* birthday – дата рождения

Описание методов:

* getName() – получить имя
* getLastname() – получить фамилию
* getBirthday() – получить дату рождения
* setName() – задать имя
* setLastname() – задать фамилию
* setBirthday() – задать день рождения

## 2.2. Управляющие классы

2.2.1. Класс «SaveAndLoad»

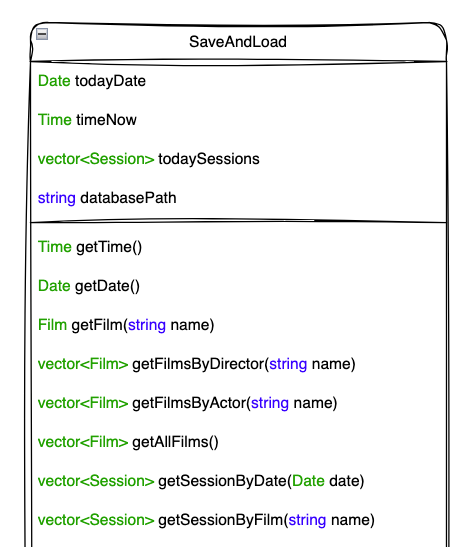


Рисунок 13. Класс "SaveAndLoad" (1)

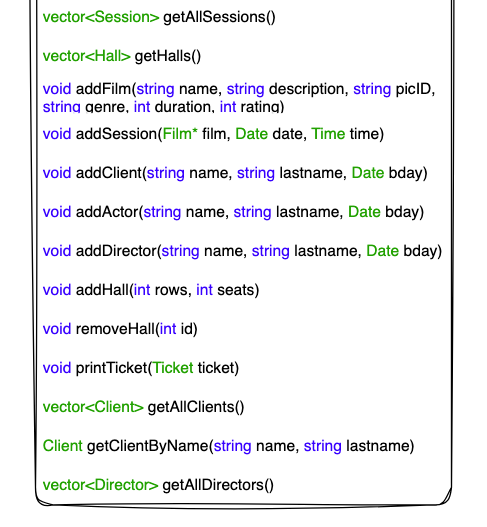


Рисунок 14. Класс "SaveAndLoad" (2)

Класс «SaveAndLoad» является управляющим классом, реализующим удобный API для работы с базой(ами) данных и хранилищем.

Описание полей:

* todayDate – сегодняшняя дата
* timeNow – время сейчас
* todaySessions – массив сегодняшних сеансов
* databasePath – путь к базе данных

Описание методов:

* getTime() – получить время (сейчас)
* getDate() – получить сегодняшнюю дату
* getFilm() – получить фильм по названию
* getFilmsByDirector() – получить фильмы по режиссеру
* getFilmsByActor() – получить фильмы с участием актера
* getAllFilms() – получить все фильмы
* getSessionByDate() – получить сеансы по дате
* getSessionsByFilm() – получить сеансы по фильму
* getAllSessions() – получить список всех сеансов
* getHalls() – получить
* addFilm() – добавить фильм
* addSession() – добавить сеанс
* addClient() – добавить клиента
* addActor() – добавить актера
* addDirector() – добавить режиссера
* addHall() – добавить зал
* removeHall() – удалить зал
* printTicket() – распечатать билет (для выдачи клиенту)
* getAllClients() – получить список всех клиентов
* getClientByName() – получить клиента по имени
* getAllDirectors() – получить список всех режиссеров

## 2.3. Используемые паттерны проектирования

2.3.1. Singleton

Реализован в классе «SaveAndLoad». Данный класс в программе глобален и должен существовать в одном экземпляре, поэтому паттерн «одиночка» хорошо подходит. Цель использования данного паттерна в том, чтобы реализовать интерфейс сохранения и загрузки данных, абстрагируя логику работы программы, от работы с базой данных. Такой подход позволит масштабировать программу. Например, в будущем, при расширении сети кинотеатров, можно вынести базу в облако чтобы все кинотеатры работали с одной БД. А в программе нам нужно будет поправить только реализацию методов данного класса.

## 2.4. Диаграмма классов

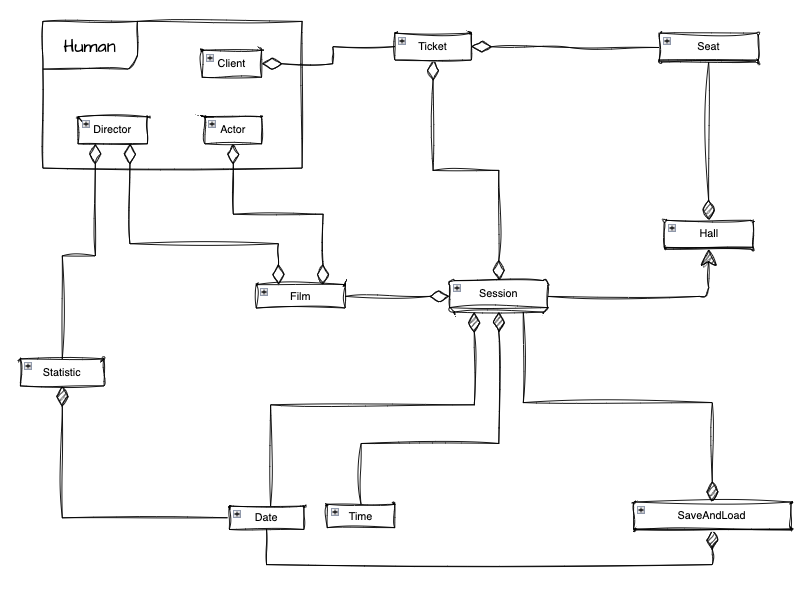


Рисунок 15. Диаграмма классов

# 3. Разработка приложения

## 3.1. Разработка интерфейса приложения

3.1.1. Главное меню программы

Главное меню программы содержит кнопки (QPushButton):

* Сеансы сегодня
* Фильмы сегодня
* Сеансы
* Фильмы
* Клиенты
* Актеры
* Режиссеры
* Кинозалы

Нажатие на каждую кнопку открывает интерфейс управления соответствующим разделом.

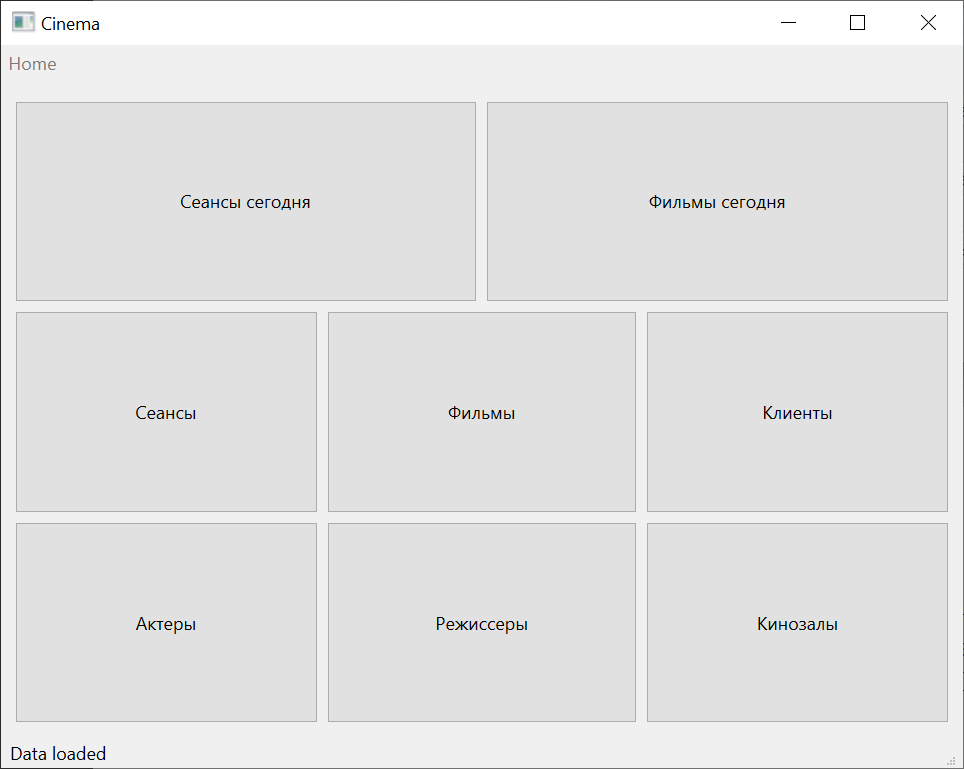


Рисунок . Главное меню программы

3.1.2. Сеансы