|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

**КУРСОВАЯ РАБОТА**

по дисциплине «Объектно-ориентрованное программирование»

**Тема курсовой работы** Проектирование и реализация программной системы с использованием объектного подхода

**Студент группы** ИВБО-09-17 Ушакова Анастасия Сергеевна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель курсовой работы** ст. преподаватель Гусев К.В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена к защите «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

Допущен к защите «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

Москва 2018

|  |
| --- |
| C:\Documents and Settings\admin\Рабочий стол\МИРЭА_ЭМБЛЕМА_приказ.JPG |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Московский технологический университет»**  **МИРЭА** |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

Утверждаю

Заведующий кафедрой МОСИТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Головин С.А.

«\_\_\_» \_\_\_\_\_\_\_\_\_ 2018 г.

**ЗАДАНИЕ**

**на выполнение курсовой работы по дисциплине**

«Объектно-ориентрованное программирование»

Студент Ушакова Анастасия Сергеевна Группа ИВБО-09-17

**Тема работы:** Проектирование и реализация программной системы с использованием объектного подхода

**Исходные данные:** задание на курсовую работу,тема №1, вариант №8. 8. Разработать систему Формирование расписания экзаменов группы. Функционал: Сотрудником учебного отдела посредством Интерфейса системы для Группы составляется расписание по нескольким экзаменам, при этом распределяются аудитории так, чтобы не было наложений. Группа имеет уникальный номер, По номеру группы можно получить ее расписание. Расписание экзамена определяется: Номер группы, Дисциплина, дата, время начала экзамена. Формируется запрос на свободную аудиторию на заданное время. Аудитория (Номер аудитории, Признак свободной или занятой аудитории, Дата, Время) Формируется запрос на свободную аудиторию для конкретного экзамена.

**Перечень вопросов, подлежащих разработке, и обязательного графического материала:**

Моделирование объектной системы с использованием обозначений языка моделирования UML

Реализация программной системы средствами языка С++

Отчет по курсовой работе в виде расчетно-пояснительной записки .

**Срок представления к защите курсовой работы:** до «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

**Задание на курсовую работу выдал** \_\_\_\_\_\_\_\_\_\_\_\_ (Гусев К.В.)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

**Задание на курсовую работу получил**  \_\_\_\_\_\_\_\_\_\_\_\_ (Ушакова А.С.)

СОДЕРЖАНИЕ

1.[Реферат 2](#_Toc515219392)

1.1. [Общее число страниц, количество таблиц, рисунков, количество разделов 2](#_Toc515219393)

1.2. [Краткий обзор содержания основных разделов – расчетно-пояснительной записки 2](#_Toc515219394)

2. [Введение 3](#_Toc515219395)

3. [Теоретическое введение 5](#_Toc515219396)

3.1. [Объектно-ориентированное программирование 5](#_Toc515219397)

3.2. [UML 6](#_Toc515219398)

3.3. [Подходы к сбору и анализу требований в ПО 9](#_Toc515219399)

3.4. [Методы создания пользовательского интерфейса 10](#_Toc515219400)

4. [Проектная часть 11](#_Toc515219401)

4.1. [Разработка спецификаций 12](#_Toc515219402)

4.2. [Список выявленных понятий 13](#_Toc515219403)

4.3. [Список понятий, претендующих на класс 13](#_Toc515219404)

4.4. [Концептуальная модель предметной области 14](#_Toc515219405)

4.5. [Проектирование программного обеспечения 14](#_Toc515219406)

5. [Эксперементальная часть 17](#_Toc515219407)

6. [Тестирование 21](#_Toc515219408)

7. [Заключение 27](#_Toc515219409)

8. [Список литературы 28](#_Toc515219410)

9. [Приложение 29](#_Toc515219411)

# РЕФЕРАТ

Курсовая работа содержит 3 раздела, 41 страницу, одну таблицу, 21 рисунок.

## Краткий обзор содержания основных разделов – расчетно-пояснительной записки

В данной курсовой работе рассматривается поэтапное проектирование и реализация программной системы с использованием стандартных библиотек объектного-ориентированного языка С++, а также библиотек Qt.

В теоретической части раскрыты понятия объектно-ориентированного программирования, языка UML, рассмотрены подходы к сбору и анализу требований к ПО, рассмотрены методы создания интерфейса.

В проектной части содержится разработка спецификаций, список выявленных понятий, концептуальная модель предметной области, проектирование ПО, а также некоторые диаграммы.

Экспериментальная часть содержит описание реализации каждого класса с пояснениями и описанием работы программы.

В разделе «Тестирование» проведены некоторые тесты, дабы проверить корректность работы программы.

Исходный программный код, разделенный на модули приведен в приложении.

# ВВЕДЕНИЕ

Основная цель представленной курсовой работы состоит в создании GUI приложения с использованием объектных возможностей языка программирования С++. Помимо того, данная работа преследует такие цели, как:

* Обучение элементам технологии создания программного продукта;
* Применение теоретических знаний и получение практических навыков по проектированию и реализации программной системы с применением объектно - ориентированного подхода;
* Знакомство со стандартом оформления документации на программное обеспечение.

Приложения с графическим интерфейсом обеспечивают произвольный доступ ко всем видимым экранным объектам, которыми человек может непосредственно манипулировать.

GUI приложения весьма актуальны в наше время, ведь они упрощают работу с документацией, помогают эффективнее решить поставленную задачу, автоматизируют некоторые действия человека (например, поиск информации), а наличие графического интерфейса, элементы которого реализованы на основе метафор, отражающих их назначение и свойства, позволяет неподготовленному пользователю быстро разобраться в работе программы.

В ходе курсовой работы решаются следующие задачи:

1. Объектный анализ решаемой задачи и разработка диаграмм, характеризующих соответствующие классы и объекты системы, выделенные в ходе анализа.
2. Разработка спецификаций программного обеспечения. Анализ требований к функциональности, указанной в варианте.
3. Построение концептуальной модели предметной области и диаграммы классов и отношений между ними.
4. Проектирование программного обеспечения:
5. Уточнение отношений между классами;
6. Определение атрибутов и методов класса;
7. Составление текстовых спецификаций интерфейса классов;
8. Проектирование методов класса;
9. Создание диаграммы объектов с указанием воздействия объектов друг на друга и передачей сообщения;
10. Реализация программы на основе всех проектных решений;
11. Реализация каждого класса, представленного на диаграмме классов;
12. Обработка исключительных ситуаций, используя аппарат исключений «try-catch»;
13. Разработка графического интерфейса пользователя;
14. Выполнение тестирование реализованной программной системы;
15. Составление отчета по курсовой работе.

# ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

## Объектно-ориентированное программирование

Объектно-ориентированное моделирование и проектирование – это подход к решению задач с использованием моделей, основанных на понятиях реального мира. Фундаментальным элементом является объект, объединяющий структуру данных с поведением. В отличие от процедурного программирования, в котором акцент делается на алгоритмах, в объектно-ориентированном - на первое место поставлены данные о свойствах и поведении объектов реального мира.

Существует несколько основных принципов объектно-ориентированного программирования. А именно:

* Инкапсуляция;
* Наследорвание;
* Полиморфоизм.

Инкапсуляция означает, что объекты с одинаковыми атрибутами и поведением объединяются в классы (расширенная концепция структур данных), то есть происходит связь данных и функций, которые ими и манипулируют.

Наследование – это наличие общих атрибутов и операций классов, образующих иерархию. Оно дает возможность создавать абсолютно новые классы путем абстрагирования общих атрибутов и шаблонов поведения. Существуют понятия базового и дочернего классов. Дочерний класс наследует все свойства и методы базового, добавляя к ним собственные. Наследование обеспечивает большое количество преимуществ в плане проектирования, сокращая количество повторений в программах.

Полиморфизм означает множественность форм, то есть одна и та же операция может в разных классах вести себя по-разному. К примеру, есть связка ключей, но к данному замку подойдет только один, поэтому человек перебирает всевозможные варианты, пока не найдет подходящий. Похожее поведение присуще и компилятору. При вызове метода класса он сначала ищет его в классе, если ничего не находит, то совершается переход в базовый класс и продолжается поиск.

## UML

Язык UML ‒ это графический язык моделирования общего назначения, предназначенный для спецификации, визуализации, проектирования и документирования всех артефактов, создаваемых при разработке программных систем.

Одно из основных предназначений UML – создание универсального и удобного средства, которое позволило бы снизить до минимума расхождения в толковании спецификации (декларативного описание того, как нечто устроено или работает). Некоторые модели UML могут быть представлены в виде картинок, что обеспечивает наглядность их визуализации и простоту восприятие человеком.

UML создан не только для описания абстрактного образа приложения, но и для манипулирования частями этих приложений. То есть этот язык позволяет создавать модели программных систем, по которым может производиться, к примеру, автоматическая генерация программного кода.

Модели UML уже сами по себе являются документацией, ведь их можно хранить и использовать в форме документов в формате XMI (внешний формат данных, основанный на языке XML, предназначенный для перевода моделей в последовательность битов и обмена ими). Также, по требованиям стандарта, на любом UML графике можно добавить текстовые заметки с пояснениями.

В ходе разработки какого-либо программного продукта, чаще всего используются UML диаграммы. Они делятся на две группы: общие и специальные;

Общие диаграммы могут быть использованы, не зависимо от предмета моделирования. Они также подразделяются на несколько видов.

1) Диаграмма вариантов использования. Изображаются варианты использования и действующие лица, между которыми устанавливаются ассоциация, различные типы зависимости, а также обобщение.

2) Диаграмма классов. Изображаются классы и отношения между ними.

3) Диаграмма автомата. Выделяются состояния и описываются переходы между ними.

4) Диаграмма деятельности – один из способов описания поведения системы. По своему виду напоминает уже знакомую нам блок-схему.

5) Диаграмма последовательности. Отображается последовательность пересылок сообщений между взаимодействующими объектами. На диаграмме связи между экземплярами классов, объектов и действующих лиц располагаются сверху вниз, с учетом времени отправки, т.е. самое позднее сообщение будет изображено внизу.

6) Диаграмма коммуникации. Аналогично диаграмме последовательности изображает последовательность передачи сообщений, но они уже изображаются другими графическими компонентами, а также акцент делается не на времени, а на структуре связей.

7) Диаграмма компонентов. Показывает связи между модулями системы.

8) Диаграмма размещения. В отличие от других диаграмм, показывает, как компоненты системы физически размещены на вычислительных ресурсах во время выполнения.

Специальные диаграммы являются дополнением или частным случаем какой-либо общей диаграммы. И также подразделяются на виды:

1) Диаграмма объектов – экземпляр диаграммы классов

2)Диаграмма внутренней структуры. Используется для более подробного представления взаимодействия классов.

3) Обзорная диаграмма взаимодействия – разновидность диаграммы деятельности с расширенным синтаксисом.

4)Диаграмма синхронизации – особая форма диаграммы последовательности, где внимание акцентируется на изменении состояний и временной синхронизации компонентов.

5)Диаграмма пакетов. Позволяет управлять сложностью самой модели.

На основе ранее рассмотренных диаграмм создаются UML модели. На разных этапах разработки, используются разные разновидности:

* Этап анализа и уточнения спецификаций – модель использования;
* Этап логического и физического проектирования – логическая модель, модель реализации, модель процессов;
* Этап реализации – модель развертывания.

Модель использования определяет требуемое поведение ПС в виде сценариев работы, показанных на диаграмме вариантов использования, и концептуальной модели ПО.

Концептуальная модель показывает семантику предметной области в виде совокупности понятий, их атрибутов и связей между ними.

Логическая модель отражает ключевые абстракции ПО в виде диаграмм классов, последовательностей системы, деятельностей, последовательностей действий сценария.

Модель реализации преобразует программные модули в другой вид, к примеру, в программный код.

Модель процессов отображает упрощает представление о внутренней работе системы, показывая взаимодействие отдельных вычислительных процессов в мультипроцессорных системах.

Модель развертывания также, как и модель процессов, создана для понимания внутреннего устройства системы и отображает размещение компонентов на её устройствах.

## Подходы к сбору и анализу требований в ПО

На самом первом этапе разработки продукта главное - понять, что требует заказчик от системы. Дабы избежать нежелательных рисков, разработчики используют различные способы сбора, документирования, согласования и модификации требований к ПО. Очень важно выбрать правильный подход, ведь «ошибки, допущенные на стадии сбора требований, составляют от 40 до 60% всех дефектов проекта» - пишут К.Виггерс и Д.Битти в своей книге «Разработка требований к программному обеспечению».

Рассмотрим некоторые приемы сбора и анализа требований:

1. Сбор информации.
2. Определение процесса формулирования требований;
3. Определение образа и границ проекта;
4. Определение классов пользователей;
5. Выделение из пользователей ярого сторонника продукта;
6. Создание фокус-группы;
7. Определение назначения продукта;
8. Определение системных событий и реакции на них;
9. Проведение совместных семинаров, упрощающих сбор требований;
10. Наблюдение за пользователями на рабочих местах;
11. Изучение отчётов о проблемах;
12. Многократное использование требований.
13. Анализ.
14. Разработка контекстной диаграммы;
15. Создание прототипов;
16. Анализ осуществимости;
17. Расставление приоритетов для требований;
18. Моделирование требований;
19. Создание словаря терминов;
20. Разделение требований по подсистемам;

## Методы создания пользовательского интерфейса

Для обеспечения взаимодействия пользователя с программой необходим интерфейс. Существует два способа его организации: командная строка и графический интерфейс (GUI).

Командная строка представляет собой диалог между человеком и системой в явной форме. На каждую введенную команду пользователь тут же получает ответ.

Графический интерфейс открывает пользователю произвольный доступ к любому из объектов, изображенных на экране с помощью клавиатуры, мышки или других устройств.

Наиболее широкое распространение получил именно графический интерфейс, ведь он нагляден, прост для понимания неопытным пользователем и не требует особой подготовки для работы с ним. GUI скрывает прямой диалог между компьютерной системой и человеком, для которого при клике на кнопку команды выполняются будто бы чисто механически.

# ПРОЕКТНАЯ ЧАСТЬ

Разработать систему:

Формирование расписания экзаменов группы.

Функционал:

* Составление расписания пользователем посредством интерфейса системы;
* Получение расписания по номеру группы;
* Поиск свободных аудиторий по заданному промежутку времени.

В соответствии с поставленными целями, было принято решение использовать в качестве среды разработки Qt Creator 4.6.1 (Рисунок 1). Этот выбор обусловлен тем, что Qt имеет все необходимое для быстрого и экономического проектирования, разработки, тестирования, развертывания и поддержки программного обеспечения для любого проекта.

Qt Creator — кроссплатформенная свободная интегрируемая среда для разработки на С/С++ и QML. Он позволяет запускать написанное разработчиком программное обеспечение в большинстве современных операционных систем, благодаря тому, что он компилирует программу для каждой системы без изменения исходного кода. Еще одно из достоинств Qt – это то, что он включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Таким образом, можно сосредоточиться на создании лучших пользовательских впечатлений, вместо того, чтобы кодировать то, что уже было закодировано ранее.

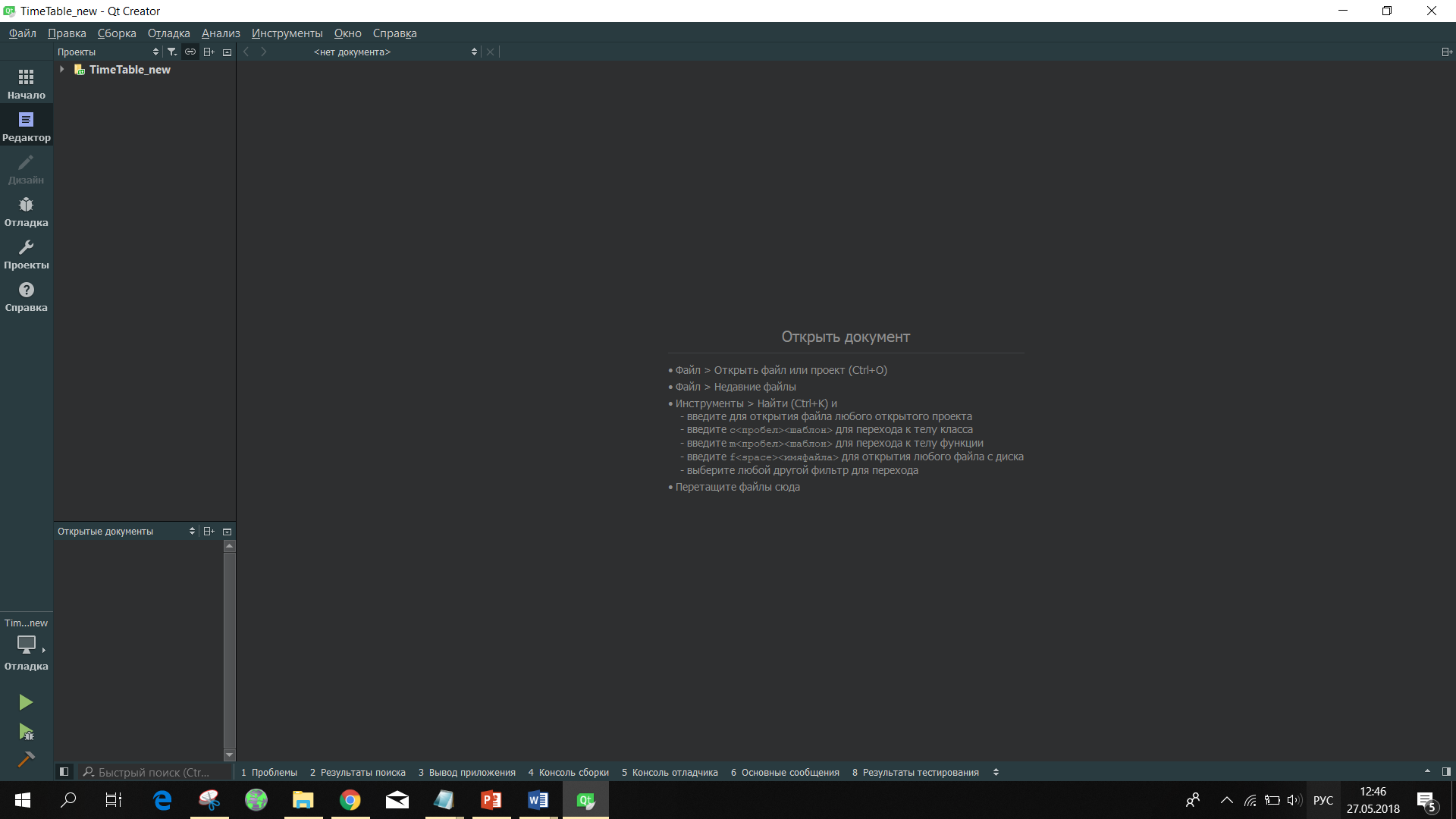


Рисунок 1 - Главное окно среды разработки Qt

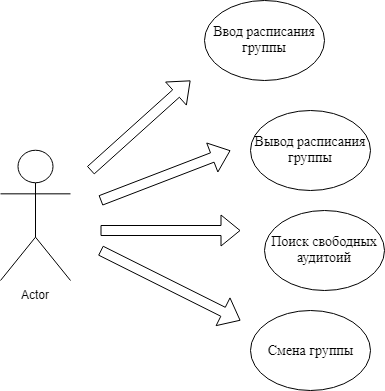
Одной из главных причин моего выбора послужило наличие новой концепции ведения меж-объектных коммуникаций – «сигналы и слоты» - полностью заменяющая не вполне надежную модель обратных вызовов.

Кроме всего прочего, Qt имеет замечательную документацию. С помощью Qt Assistant можно всегда легко и просто найти нужную информацию.

## Разработка спецификаций

Предметной областью задания является база данных «расписание экзаменов», которая содержит поля данных:

* День недели;
* Число;
* Название дисциплины;
* Имя преподавателя;
* Время;
* Аудитория;

Рисунок 2 - Диаграмма вариантов использования

## Список выявленных понятий

1. Каталог Список групп (системный каталог);
2. Каталог Список аудиторий (системный каталог);
3. Каталог Информация об экзаменах;
4. Время проведения экзамена;

## Список понятий, претендующих на класс

В ходе разработки были выявлены классы-кандидаты:

1. Класс MainWindow – объекты создаются при запуске программы;
2. Класс Menu – объекты создаются при правильном вводе группы в начальном окне;
3. Класс Classes – объекты создаются при выводе списка свободных аудиторий;

## Концептуальная модель предметной области

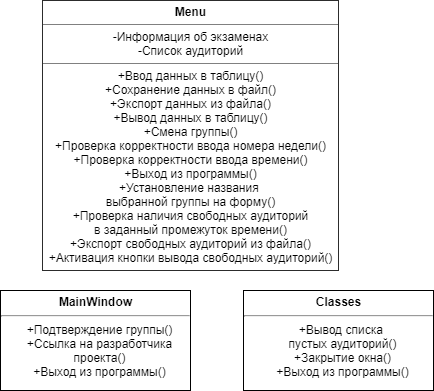


Рисунок 3 - Диаграмма классов

## Проектирование программного обеспечения

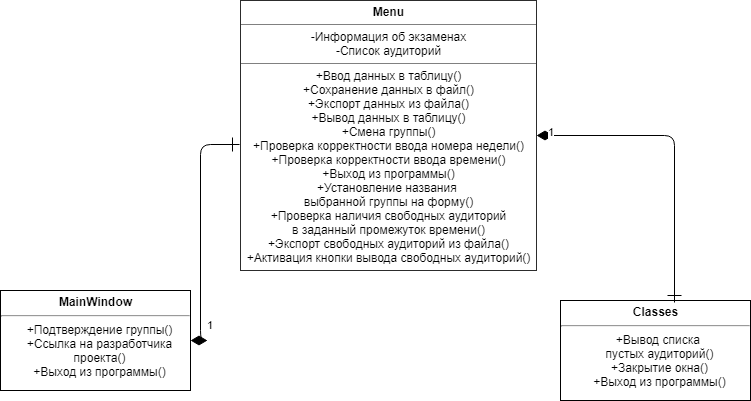


Рисунок 4 - Диаграмма отношений классов

Рассмотрим каждый класс с его собственными атрибутами и методами более подробно.

1. Класс MainWindow.

Методы:

* void link() – активирует ссылку на страницу разработчика;
* void onClick() – проверяет корректность введенной пользователем группы;
* void onExit() – завершает работу программы;

1. Класс Menu.

Поля:

* QString Base[35][6] – хранит информацию об экзаменах;
* QList <QString> Audit – хранит список свободных аудиторий.

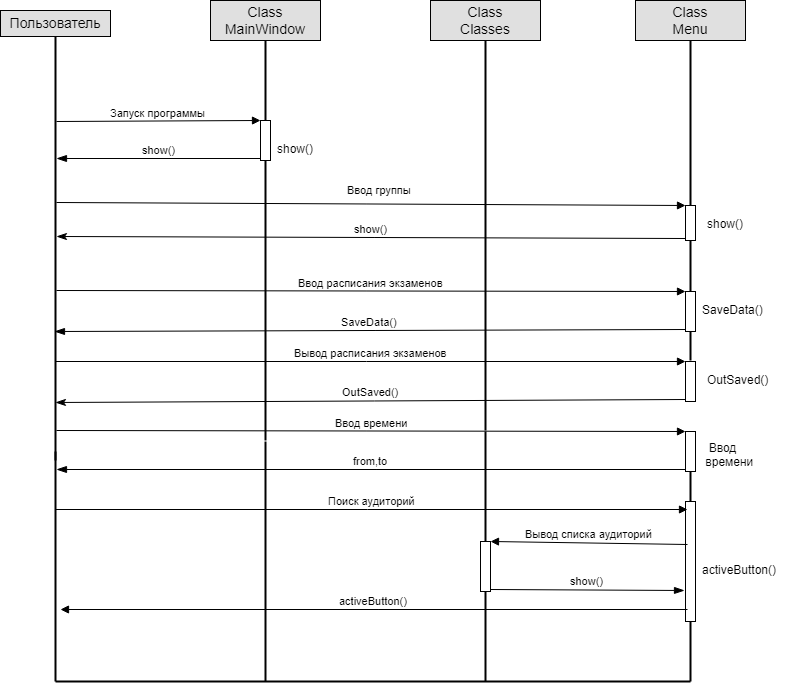
Методы:

* void setHeadline(QString text) – передаёт строку с названием группы на форму menu.ui;
* void InputList() – производит запись данных об экзаменах, введенных непосредственно пользователем в таблицу, в массив;
* void SaveData() – сохраняет данные из массива в файл;
* void OutputList() – считывает последние сохранённые данные из соответсвующего файла в массив;
* void changer() – закрывает окно расписания экзаменов и возвращает пользователя на окно ввода группы;
* void activeButton() – по нажатию кнопки вызывает метод void OutClass();
* void OutClass() – выводит список свободных аудиторий;
* void timeget() – ищет свободные аудитории в заданный промежуток времни;
* int Tcheck(int from,int to) – проверяет корректность ввода времени;
* int selectweek() – проверяет корректность ввода номера недели;
* void on\_exit\_2\_triggered() – завершает работу приложения.

1. Класс Classes.

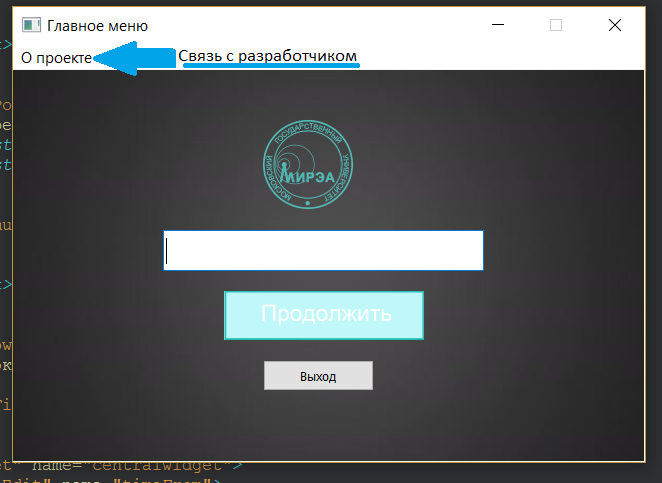
Методы:

* void ShowClass(QString txt) – выводит список свободных аудиторий в форму
* void closed() – закрывает данное окно и возвращает пользователя в предыдущее;
* void Exit() – завершает работы приложения.

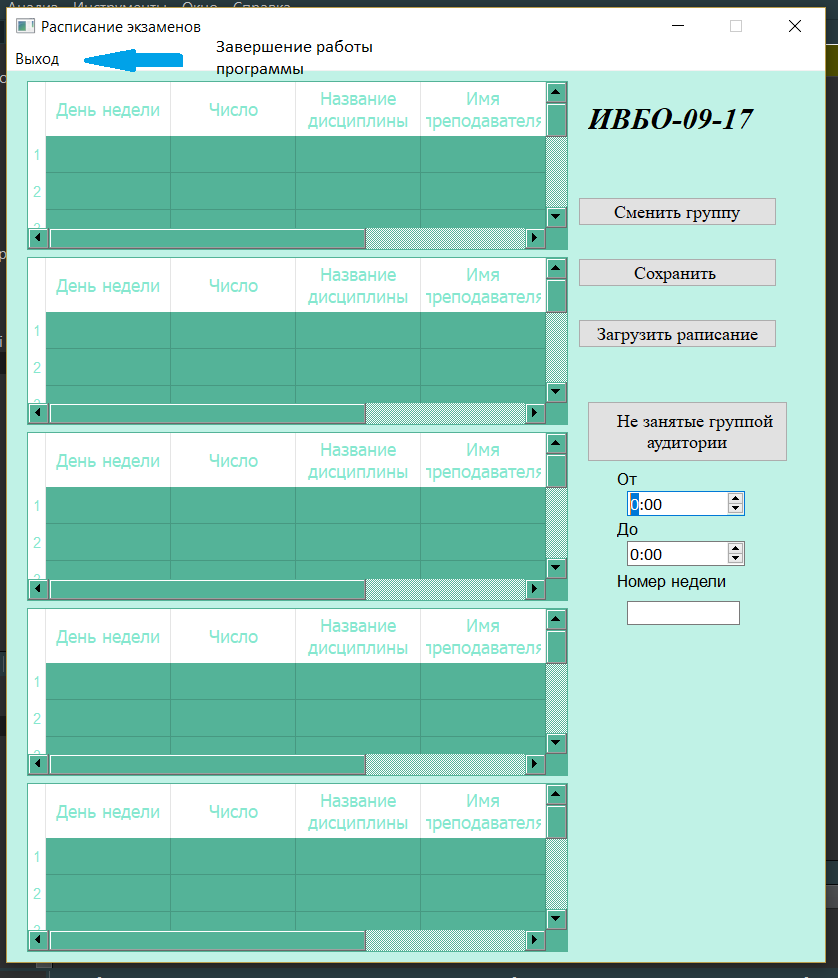
Рисунок 5. Диаграмма сценариев

# ЭКСПЕРЕМЕНТАЛЬНАЯ ЧАСТЬ

При запуске программы открывается окно с полем для ввода группы (Рисунок 6). Пользователь вводит название, которое считывается и сравнивается с заранее введёнными в файл данными. Если введенная группа существует, то открывается второе окно с расписанием экзаменов (Рисунок 7), в противном случае выводится информация об ошибке.

Рисунок 6 – Стартовое окно

Во втором окне пользователь видит ряд функций: ввод расписания экзаменов, вывод последнего сохранения, поиск свободных аудиторий в конкретный момент времени и переключение группы.

Рисунок 7 – Окно с расписанием экзаменов

1. Ввод расписания.

Для начала, пользователь должен заполнить таблицу, записав в ячейки информацию об экзаменах: день недели, число, название дисциплины, имя преподавателя, время и аудитория проведения. Затем, чтобы сохранить введенные данные, необходимо нажать на кнопку «Сохранить». После нажатия информация из таблицы считывается в двумерный строковый массив, в котором хранятся данные, пока открыто данное окно, а затем, из него построчно, с разделительным знаком “/”, записываются в файл с названием “Group <название группы> .txt”, и открывается окошко с информацией об успешном сохранении. Если файл не найден, программа выдаёт ошибку.

1. Вывод расписания.

Чтобы вывести последнее сохранение, пользователь должен нажать на кнопку «Загрузить расписание». Она активирует процесс, при котором данные считываются из файла “Group <название группы> .txt” в двумерный строковый массив, из которого они выводятся в таблицу.

1. Поиск аудиторий.

Если пользователь желает найти аудитории, не занятые данной группой, он должен ввести в соответствующие поля промежуток времени и номер недели (Рисунок 8). При вводе данных, происходит проверка. Если они введены неверно, программа выдаёт ошибку, иначе – происходит приведение переменных, в которые были считаны данные, к целочисленному типу. Далее переменные, содержащая промежуток времени сравнивается со значением, также приведенным к целочисленному типу, ячейки столбца двумерного строкового массива, содержащего время проведения экзаменов. Если промежуток, полученный из ячейки массива, покрывает тот, который ввёл пользователь, то из ячейки соответствующего столбца массива записывается в переменную типа QList. После этого считываются данные из файла “Class.txt” и сравниваются с информацией, содержащейся в переменной типа QList. Если данные не совпали, то информация из файла записывается в новую строковую переменную с разделителем “\n”. Далее через объект класса “Classes” вывзваем окно со списком аудиторий (Рисунок 9).

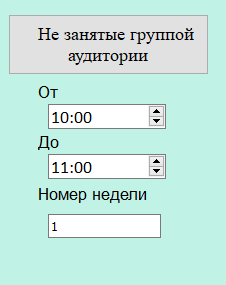
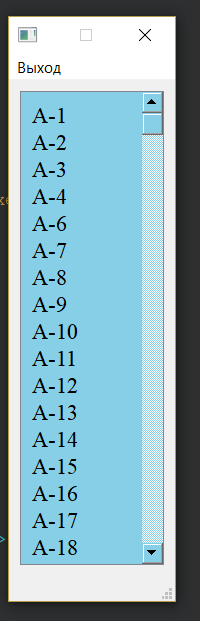


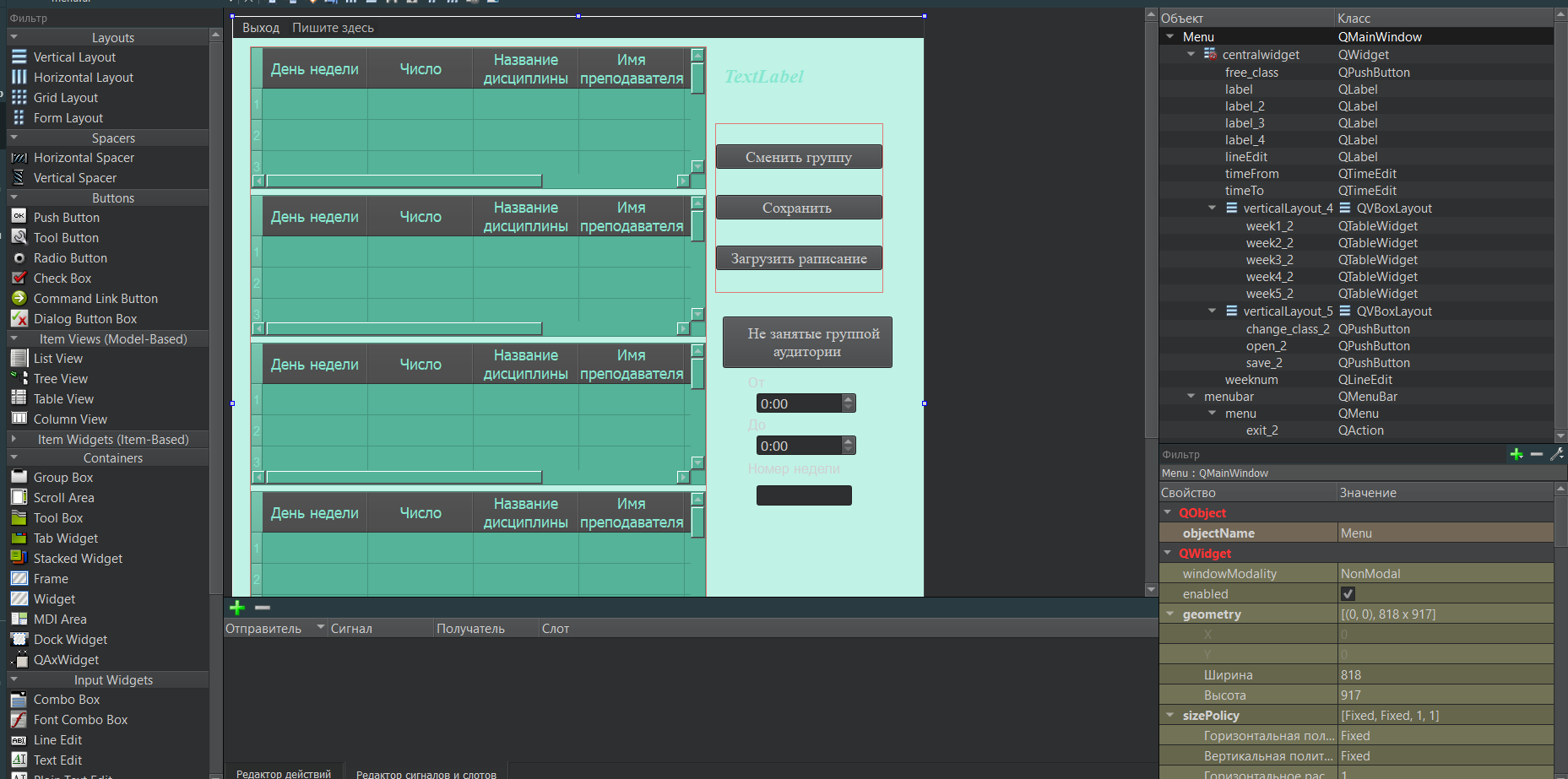
Рисунок 8 – Поля ввода времени и номера недели

Рисунок 9 – Окно со списком аудиторий

1. Смена группы.

Для того, чтобы начать работу с новой группой, пользователь должен нажать на кнопку «Сменить группу». При этом окно с расписанием экзаменов закрывается, и программа возвращает стартовое окно.

Для создания пользовательского интерфейса были использованы инструменты Qt Designer (Рисунок 10).

Рисунок 10 – Окно разработки дизайна

# ТЕСТИРОВАНИЕ

Тест 1: Нажимаем на вкладку «О проекте», затем на «Разработчики», далее на ссылку.

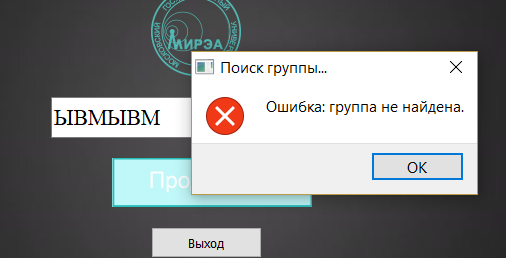
Результат: Открылось окно браузера и совершился переход по ссылке.

Тест 2: Нажимаем на кнопку «Выход».

Результат: Приложение закрылось.

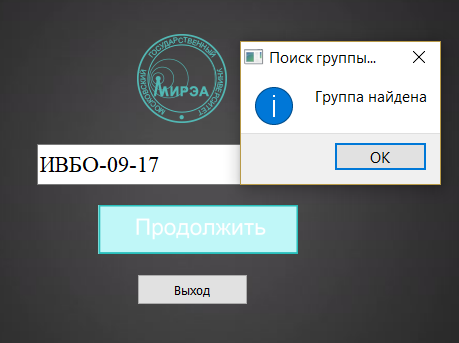
Тест 3. Вариант 1: Вводим группу неверно.

Результат: Вывод ошибки. (Рисунок 11)

Рисунок 11 – Ошибка ввода группы

Вариант 2: Вводим существующую группу.

Результат: Вывод информации о том, что группа нашлась и переход на окно расписания. (Рисунок 12)

Рисунок 12 – Группа найдена

Вариант 3: Нажмем кнопку «Продолжить», оставив поле для ввода группы.

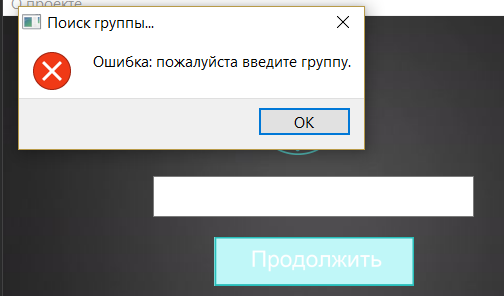
Результат: Ошибка (Рисунок 13)

Рисунок 13-Ошибка.Группа не введена

Тест 4: Сохраним расписание

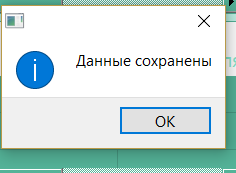
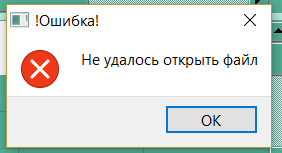
Результат: Вывод информации о том, что данные сохранены. (Рисунок 14)

Рисунок 14 - Сохранение

Тест 5. Вариант 1: Загрузим расписание группы, предварительно ничего не сохранив.

Результат: Ошибка поиска файла. (Рисунок 15)

Рисунок 15-Файл не найден

Вариант 2: Ведем данные в таблицу, сохраним, затем откроем окно с расписанием экзаменов заново и нажмем кнопку «Загрузить расписание».

Результат: Вывод в таблицу информации об экзаменах. (Рисунок 16)

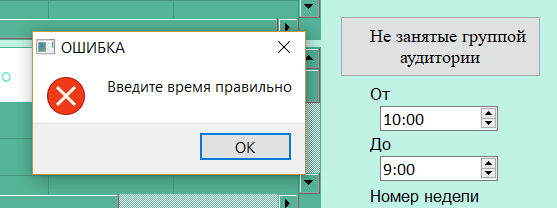
Рисунок 16 – Вывод расписания

Тест 6: Переключим группу, нажав кнопку «Сменить группу».

Результат: Окно с расписанием закрылось, открылось стартовое окно.

Тест 7. Вариант 1: Введем время неверно. И нажмем на кнопку «Не занятые группой аудитории».

Результат: Вывод ошибки. (Рисунок 17)

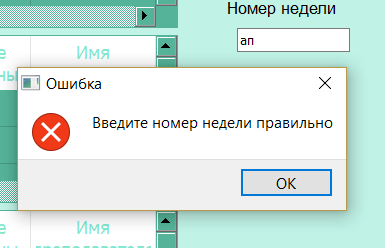
Рисунок 17 – Ошибка ввода времени

Вариант 2: Введем номер недели неверно. И нажмем на кнопку «Не занятые группой аудитории».

Результат: Ошибка о том, что номер недели введен неверно. (Рисунок 18)

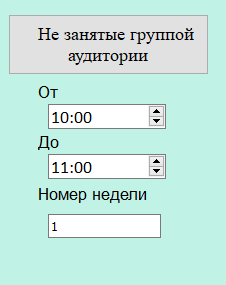
Вариант 3: Введем в поле «Номер недели 6 или 0». И нажмем на кнопку «Не занятые группой аудитории».

Результат: Ошибка о том, что номер недели введен неверно. (Рисунок 18)

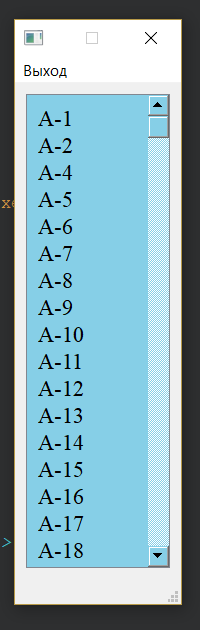
Рисунок 18 – Ошибка ввода номера недели

Тест 8: Введем корректно время и номер недели. И нажмем на кнопку «Не занятые группой аудитории».

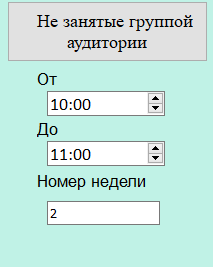
Вариант 1: Введём номер недели 1. И промежуток времени от 10:00 до 11:00. (Рисунок 19)

Рисунок 19 – Ввод данных для первой недели

Результат: Вывод всех аудиторий, кроме А-3. (Рисунок 20)

Рисунок 20 – Вывод списка свободных аудиторий первой недели

Вариант 2: Введём номер недели 2. И промежуток времени от 10:00 до 11:00. (Рисунок 21)

Рисунок 21 - Ввод данных для второй недели

Результат: Вывод всех аудиторий.

Таблица 1 – Тестирование реализованной программной системы.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Вариант 1 | Вариант 2 | Вариант 3 |
| Тест 1 | + |  |  |
| Тест 2 | + |  |  |
| Тест 3 | + | + | + |
| Тест 4 | + |  |  |
| Тест 5 | + | + |  |
| Тест 6 | + |  |  |
| Тест 7 | + | + | + |
| Тест 8 | + | + |  |

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была достигнута основная цель: разработано GUI приложение с использованием языка программирования С++. В ходе разработки были спроектированы модули, реализующие функции, необходимые для работы с объектами класса “Menu” и классов “MainWindow” и “Classes”.

# СПИСОК ЛИТЕРАТУРЫ

1) К. Виггерс, Д. Битти Разработка требований к программному обеспечению. 3-е изд, дополненное / Пер. с англ. –М.: Издательство «Русская редакция»; СПб.: БХВ-Петербург, 2014. – 736 стр.: ил.

2) Денис Иванов, Федор Новиков Моделирование на UML: [Электронный ресурс]. 2013-2018. URL: <http://book.uml3.ru/about> (Дата обращения: 26.05.2018)

3) Максим Отставнов, Вадим Виниченко, Андрей Добровольский, Кирилл Маслинский Открытые системы и свободные программы: [Электронный ресурс]. 2018. URL: <http://heap.altlinux.org/modules/scheme_article_1.kirill.next/index.html> (Дата обращения: 26.05.2018)

4) Шлее М. Qt 5.3. Профессиональное программирование на С++. – СПб.: БХВ-Петербург, 2015. – 928 с.: ил.

5) Вайсфельд М. Объектно-ориентированное мышление. — СПб.: Питер, 2014. — 304 с.: ил.

6) Дж.Рамбо, М. Блаха UML 2.0 Объектно-ориентированное моделирование и разработка. 2-е изд. – СПб.: Питер, 2007. – 544 с.: ил.

7) Скворцова Л.А. Методическое пособие. Объектная декомпозиция с использованием языка моделирования UML. 2018.

8) Документация Qt [Электронные ресурс] URL: <http://doc.qt.io/> (Дата обращения: 26.05.2018)

# ПРИЛОЖЕНИЕ

mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

*//#include* *<QString>*

*namespace* Ui {

*class* MainWindow;

}

*class* MainWindow : *public* QMainWindow

{

Q\_OBJECT

*public*:

*explicit* MainWindow(QWidget \*parent = 0);

bool flag=0;

~***MainWindow***();

*private*:

Ui::MainWindow \*ui;

void **link**(); *//активация* *ссылки* *на* *страницу* *разработчика*

void **onClick**(); *//проверка* *группы*

void **onExit**(); *//выход* *из* *программы*

};

#endif *//* *MAINWINDOW\_H*

mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "menu.h"

#include <QFile>

#include <QTextStream>

#include <QMessageBox>

*//#include* *<QDebug>*

#include <QUrl>

#include <QDesktopServices>

MainWindow::**MainWindow**(QWidget \*parent) :

QMainWindow(parent),

ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

*//связь* *сигналов* *и* *слотов*

connect(ui->actionVK\_2, &QAction::triggered, *this*, &MainWindow::link);

connect(ui->pushButton, &QPushButton::clicked, *this*, &MainWindow::onClick);

connect(ui->exitButton, &QPushButton::clicked, *this*, &MainWindow::onExit);

}

MainWindow::~***MainWindow***()

{

*delete* ui;

}

void MainWindow::**onClick**()*//проверка* *группы*

{

*try*{

QFile file("C:/Users/Lenovo/Documents/1)My kursach/TimeTable\_new/GroupEnter.txt");

QString group=""; *//переменная* *для* *стчитывания* *строка* *из* *файла*

*if*(ui->lineEdit->text() != "")*//проверка* *ввода* *данных*

{

*if*((file.exists())&&(file.*open*(QIODevice::*ReadOnly*)))

{

QTextStream stream(&file);

*while* (!stream.atEnd()) *//считываем* *данные* *до* *конца* *файла*

{

group = stream.readLine(); *//считываем* *по* *строке*

*if*(group==ui->lineEdit->text()+";") *//ищем* *введенную* *группу* *в* *файле*

{

QMessageBox::information(*this*,"Поиск группы...", "Группа найдена");

flag=1;

Menu \*menu = *new* Menu;

menu->show();*//выводим* *окно* *с* *расписанием*

menu->setHeadline(ui->lineEdit->text());*//передаем* *форме* *с* *расписанием* *название* *группы*

close();

}

}

*if*(!flag){

QMessageBox::critical(*this*,"Поиск группы...", "Ошибка: группа не найдена.");

}

file.*close*(); *//закрываем* *файл*

}*else*{

*throw* 1;

}

}*else*

{

QMessageBox::critical(*this*,"Поиск группы...", "Ошибка: пожалуйста введите группу.");

}

}

*catch*(int i)

{

QMessageBox::critical(*this*,"!Ошибка!","Не удалось открыть файл GroupEnter.txt");

}

}

void MainWindow::**link**()

{

*try*{

*if*(!QDesktopServices::openUrl(QUrl("https://vk.com/id442156396"))) *//если* *ссылка* *не* *активна*

{

*throw* 911;

}

}

*catch*(int i){

QMessageBox::critical(*this*, "!Ошибка!" + QString::number(i), "Ссылка неактивна.");

}

}

void MainWindow::**onExit**()

{

exit(0); *//выход*

}

menu.h

#ifndef MENU\_H

#define MENU\_H

#include <QMainWindow>

#include <QString>

#include <QStandardItemModel>

#include <QTableWidget>

#include <QList>

#include <QMessageBox>

#include <QTextStream>

#include <QString>

#include <QTextEdit>

#include <QTableWidget>

#include <QFile>

#include <QLabel>

*namespace* Ui {

*class* Menu;

}

*class* Menu : *public* QMainWindow

{

Q\_OBJECT

*public*:

*explicit* Menu(QWidget \*parent = 0);

~***Menu***();

QString Base[35][6]; *//массив* *для* *хранения* *информации* *об* *экзаменах*

QList <QString> Audit; *//лист* *для* *хранения* *свободных* *аудиторий*

void **setHeadline**(QString text); *//получение* *названия* *группы* *из* *предыдущей* *формы*

void **timeget**(); *//проверка* *наличия* *свободных* *аудиторий* *в* *заданный* *промежуток* *времени* *и* *запись* *их* *в* *лист*

void **OutClass**(); *//вывод* *свободных* *аудиторий*

*private*:

void **InputList**(); *//запись* *данных,* *введенных* *пользователем,* *в* *массив*

void **SaveData**(); *//запись* *данных* *из* *массива* *в* *файл*

void **OutputList**(); *//считывание* *последних* *сохранённых* *данных* *из* *файла* *в* *массив*

void **OutSaved**(); *//вывод* *данных* *из* *массива* *в* *таблицу* *на* *форме*

void **changer**(); *//смена* *группы*

void **activeButton**(); *//вывод* *свобобных* *аудиторий*

int **selectweek**(); *//проверка* *корректности* *ввода* *номера* *недели*

int **Tcheck**(int from,int to); *//проверка* *корректности* *ввода* *времени*

void **on\_exit\_2\_triggered**(); *//выход* *из* *приложения*

Ui::Menu \*ui;

};

#endif *//* *MENU\_H*

menu.cpp

#include "menu.h"

#include "ui\_menu.h"

#include "mainwindow.h"

#include "classes.h"

Menu::**Menu**(QWidget \*parent) :

QMainWindow(parent),

ui(*new* Ui::Menu)

{

ui->setupUi(*this*);

connect(ui->change\_class\_2, &QPushButton::clicked, *this*, &Menu::changer);

connect(ui->save\_2, &QPushButton::clicked, *this*, &Menu::SaveData);

connect(ui->open\_2, &QPushButton::clicked, *this*, &Menu::OutSaved);

connect(ui->exit\_2, &QAction::triggered, *this*, &Menu::on\_exit\_2\_triggered);

connect(ui->free\_class, &QPushButton::clicked, *this*, &Menu::activeButton);

}

Menu::~***Menu***()

{

*delete* ui;

}

void Menu::**setHeadline**(QString text)

{

ui->lineEdit->setText(text); *//запись* *названия* *группы* *на* *форму*

}

void Menu::**InputList**()

{

*//1ая* *неделя*

*for*(int i=0;i<ui->week1\_2->rowCount();i++) *//пробег* *ячейкам* *таблицы*

{

*for*(int j=0;j<ui->week1\_2->columnCount();j++)

{

*if* (ui->week1\_2->item(i,j)==0) *//если* *ячейка* *пуста,* *записываем* *в* *массив* *"?"*

{

Base[i][i]='?';

}*else*

Base[i][j]=ui->week1\_2->item(i,j)->text(); *//в* *ином* *случае* *записывает* *значение*

}

}

*//2ая* *неделя*

*for*(int i=0;i<ui->week2\_2->rowCount();i++)

{

*for*(int j=0;j<ui->week2\_2->columnCount();j++)

{

*if* (ui->week2\_2->item(i,j)==0)

{

Base[i+7][i]='?';

}*else*

Base[i+7][j]=ui->week2\_2->item(i,j)->text();

}

}

*//3я* *неделя*

*for*(int i=0;i<ui->week3\_2->rowCount();i++)

{

*for*(int j=0;j<ui->week3\_2->columnCount();j++)

*if* (ui->week3\_2->item(i,j)==0)

{

Base[i+14][i]='?';

}*else*

{

Base[i+14][j]=ui->week3\_2->item(i,j)->text();

}

}

*//4ая* *неделя*

*for*(int i=0;i<ui->week1\_2->rowCount();i++)

{

*for*(int j=0;j<ui->week4\_2->columnCount();j++)

{

*if* (ui->week4\_2->item(i,j)==0)

{

Base[i+21][i]='?';

}*else*

Base[i+21][j]=ui->week4\_2->item(i,j)->text();

}

}

*//5ая* *неделя*

*for*(int i=0;i<ui->week5\_2->rowCount();i++)

{

*for*(int j=0;j<ui->week5\_2->columnCount();j++)

{

*if* (ui->week5\_2->item(i,j)==0)

{

Base[i+28][i]='?';

}*else*

Base[i+28][j]=ui->week5\_2->item(i,j)->text();

}

}

}

void Menu::**SaveData**()

{

*try*{

*this*->Menu::InputList();

QFile new\_file("C:/Users/Lenovo/Documents/1)My kursach/TimeTable\_new/Group"+ui->lineEdit->text()+".txt");

*//удаляем* *старые* *данные,* *если* *они* *есть*

*if*(new\_file.*size*() != 0)

{

*if* (new\_file.*open*(QIODevice::*WriteOnly* | QIODevice::*Text*))

new\_file.*close*();

}

*if*(new\_file.*open*(QIODevice::*WriteOnly*|QIODevice::*Append*))

{

*//записываем* *новые*

QTextStream out(&new\_file);

*//записываем* *каждую* *строку* *из* *массива* *в* *файл* *построчно,разделяя* *данные* *из* *яеек* *символом* *"/"*

*for*(int i=0;i<35;i++)

{

*for*(int j=0;j<6;j++)

{

out<<Base[i][j]<<'/';

}

out<<"\r\n";

}

new\_file.*close*();

}*else* *throw* 112;

QMessageBox::information(*this*," ","Данные сохранены"); *//информируем* *пользователя* *об* *успешной* *записи*

}

*catch*(int i)

{

QMessageBox::critical(*this*,"!Ошибка!","Не удалось открыть файл для записи");

}

}

void Menu::**OutSaved**()

{

*this*->Menu::OutputList();

*for*(int i =0; i<7;i++)

{

*for*(int j=0;j<6;j++)

{

*//1-ая* *таблица*

QTableWidgetItem \*item1 = *new* QTableWidgetItem();

item1->setText(Base[i][j]);

ui->week1\_2->setItem(i,j,item1);*//записываем* *данные* *в* *таблицу*

*//2-ая* *таблица*

QTableWidgetItem \*item2 = *new* QTableWidgetItem();

item2->setText(Base[i+7][j]);

ui->week2\_2->setItem(i,j,item2);

*//3-ая* *таблица*

QTableWidgetItem \*item3 = *new* QTableWidgetItem();

item3->setText(Base[i+14][j]);

ui->week3\_2->setItem(i,j,item3);

*//4-ая* *таблица*

QTableWidgetItem \*item4 = *new* QTableWidgetItem();

item4->setText(Base[i+21][j]);

ui->week4\_2->setItem(i,j,item4);

*//5-ая* *таблица*

QTableWidgetItem \*item5 = *new* QTableWidgetItem();

item5->setText(Base[i+28][j]);

ui->week5\_2->setItem(i,j,item5);

}

}

}

void Menu::**OutputList**()

{

*try*{

QFile myfile("C:/Users/Lenovo/Documents/1)My kursach/TimeTable\_new/Group"+ui->lineEdit->text()+".txt");

QStringList help; *//лист* *для* *записи* *данных* *без* *разделительного* *символа*

QString str; *//переменная* *для* *записи* *строки* *из* *файла*

int j=0, index=0,i=0;

*if*(myfile.*size*() != 0) *//если* *файл* *не* *пуст*

{

*if*((myfile.exists())&&(myfile.*open*(QIODevice::*ReadOnly*)))

{

QTextStream stream(&myfile);

*while* (!stream.atEnd()&&(i<35)) *//считываем* *данные* *построчно* *доконца* *файла*

{

str = stream.readLine();

help = str.split("/"); *//убираем* *разделительные* *символ*

*while*(j<6&&index<7) *//пробегаемся* *по* *столбцам* *массива* *и* *элементам* *листа*

{

*if*(help[index]=="?") *//если* *встретился* *"?",* *записываем* *в* *массив* *пробел*

{

Base[i][j]=" ";

}*else*

{

Base[i][j]=help[index]; *//в* *ином* *случае* *записываем* *данные*

}

j++;index++;

}j=0;i++; help.clear();index=0;

}

}

}*else* *throw* 2;

}

*catch*(int i)

{

QMessageBox::critical(*this*,"!Ошибка!","Не удалось открыть файл");

}

}

int Menu::**Tcheck**(int from,int to)

{

*if*(from<to) *//если* *начальное* *время* *меньше* *конечного*

*return* 1;

*else*

*return* 0;

}

void Menu::**timeget**()

{

Audit.clear();

QTime time=ui->timeFrom->time(); *//считываем* *с* *формы* *введенное* *пользователем* *время*

int timef;

timef=time.hour()\*60+time.minute(); *//переводим* *его* *целочисленный* *тип* *данных*

time=ui->timeTo->time();

int timeto;

timeto=time.hour()\*60+time.minute();

*if*(Tcheck(timef,timeto)) *//если* *время* *введено* *правильно*

{

*if* (selectweek()>0 && selectweek()<6) *//если* *номер* *недели* *введен* *верно*

{

*for*(int i=0;i<7;i++) *//пробегаемся* *по* *строкам* *массива*

{

*//1ая* *неделя*

*if*(selectweek()==1){

*if* ((Base[i][4] !=0)&&(Base[i][4]!=" ")) *//если* *ячейка* *массива* *содержит* *значение*

{

*//переводим* *время* *проведения* *экзамена* *в* *целочисленный* *тип* *данных*

int Tfrom=(Base[i][4].mid(0,2).toInt(0,10))\*60 + Base[i][4].mid(3,2).toInt(0,10);

int Tto=(Base[i][4].mid(6,2).toInt(0,10))\*60+ Base[i][4].mid(9,2).toInt(0,10);

*//проверяем,* *свободна* *ли* *аудитория* *в* *заданное* *пользователем* *время*

*if*((Tcheck(timef,Tfrom)&&!Tcheck(timeto,Tfrom))||(Tcheck(timef,Tto)&&Tcheck(timeto,Tto)))

{

Audit.push\_back(Base[i][5]); *//если* *нет,* *то* *записываем* *её* *в* *лист*

}

}

*//2ая* *неделя*

}*else* *if*(selectweek()==2)

{

*if* ((Base[i+7][4] !=0)&&(Base[i+7][4]!=" "))

{

int Tfrom=(Base[i+7][4].mid(0,2).toInt(0,10))\*60 + Base[i+7][4].mid(3,2).toInt(0,10);

int Tto=(Base[i+7][4].mid(6,2).toInt(0,10))\*60+ Base[i+7][4].mid(9,2).toInt(0,10);

*if*((Tcheck(timef,Tfrom)&&!Tcheck(timeto,Tfrom))||(Tcheck(timef,Tto)&&Tcheck(timeto,Tto)))

{

Audit.push\_back(Base[i+7][5]);

}

}

*//3я* *неделя*

}*else* *if* (selectweek()==3)

{

*if* ((Base[i+14][4] !=0)&&(Base[i+14][4]!=" "))

{

int Tfrom=(Base[i+14][4].mid(0,2).toInt(0,10))\*60 + Base[i+14][4].mid(3,2).toInt(0,10);

int Tto=(Base[i+14][4].mid(6,2).toInt(0,10))\*60+ Base[i+14][4].mid(9,2).toInt(0,10);

*if*((Tcheck(timef,Tfrom)&&!Tcheck(timeto,Tfrom))||(Tcheck(timef,Tto)&&Tcheck(timeto,Tto)))

{

Audit.push\_back(Base[i+14][5]);

}

}

*//4ая* *неделя*

}*else* *if* (selectweek()==4)

{

*if* ((Base[i+21][4] !=0)&&(Base[i+21][4]!=" "))

{

int Tfrom=(Base[i+21][4].mid(0,2).toInt(0,10))\*60 + Base[i+21][4].mid(3,2).toInt(0,10);

int Tto=(Base[i+21][4].mid(6,2).toInt(0,10))\*60+ Base[i+21][4].mid(9,2).toInt(0,10);

*if*((Tcheck(timef,Tfrom)&&!Tcheck(timeto,Tfrom))||(Tcheck(timef,Tto)&&Tcheck(timeto,Tto)))

{

Audit.push\_back(Base[i+21][5]);

}

}

*//5ая* *неделя*

}*else* *if* (selectweek()==5)

{

*if* ((Base[i+28][4] !=0)&&(Base[i+28][4]!=" "))

{

int Tfrom=(Base[i+28][4].mid(0,2).toInt(0,10))\*60 + Base[i+28][4].mid(3,2).toInt(0,10);

int Tto=(Base[i+28][4].mid(6,2).toInt(0,10))\*60+ Base[i+28][4].mid(9,2).toInt(0,10);

*if*((Tcheck(timef,Tfrom)&&!Tcheck(timeto,Tfrom))||(Tcheck(timef,Tto)&&Tcheck(timeto,Tto)))

{

Audit.push\_back(Base[i+28][5]);

}

}

}

}

}*else* *return*;

}*else* QMessageBox::critical(*this*,"ОШИБКА","Введите время правильно");

}

void Menu::**OutClass**()

{

*try*{

*this*->Menu::timeget();

QFile file1("C:/Users/Lenovo/Documents/1)My kursach/TimeTable\_new/Class.txt");

QString line=0; *//переменная* *для* *строки,* *содержащей* *список* *свободных* *аудиторий*

QString cl; *//переменная* *для* *полученной* *строки* *из* *файла*

int i=0;

*if*((file1.exists())&&(file1.*open*(QIODevice::*ReadOnly*)))

{

QTextStream stream1(&file1);

*//считываем* *данные* *построчно* *до* *конца* *файла*

*while* (!stream1.atEnd())

{

cl=stream1.readLine();

*if*(!Audit.empty()) *//если* *список* *занятых* *аудиторий* *не* *пуст*

{

*while*(i<Audit.size())

{

*if*(cl.mid(2,2)!=Audit[i].mid(2,2)) *//если* *аудитория* *из* *списка,* *совпадает* *с* *занятой*

{

line+=cl+"\n"; *//записываем* *в* *строку* *номаер* *аудитории* *через* *ввод*

}i++;

}i=0;

}*else* {*throw* 411;

*return*;}

} file1.*close*();

*//через* *объект* *класса* *вызываем* *окно* *со* *списком* *аудиторий*

Classes \*obj=*new* Classes;

obj->ShowClass(line);

obj->show();

} *else* *throw* 404;

}

*catch*(int i){

*if* (i==404)

QMessageBox::critical(*this*,"!Ошибка!","Файл не найден");

*else* *if* (i==411)

QMessageBox::critical(*this*,"!Ошибка!","Введите данные правильно");

}

}

void Menu::**activeButton**()

{

*this*->Menu::OutClass();

}

void Menu::**on\_exit\_2\_triggered**()

{

close();

}

void Menu::**changer**()

{

*//подготавливаем* *массив* *для* *записи* *новых* *данных*

*for*(int i=0;i<35;i++)

{

*for*(int j=0;j<6;j++)

{

Base[i][j]="";

}

}

*//* *открываем* *стартовое* *окно*

MainWindow \*new\_groupe = *new* MainWindow();

new\_groupe->show();

close();

}

int Menu::**selectweek**()

{

QString a=ui->weeknum->text(); *//считываем* *введенный* *пользователем* *текст*

*//если* *номер* *недели* *введен* *правильно,* *возвращаем* *его,* *в* *противном* *случае* *выдаём* *ошибку*

*if* ((ui->weeknum->text()!="1")&&(ui->weeknum->text()!="2")&&(ui->weeknum->text()!="3")&&(ui->weeknum->text()!="4")&&(ui->weeknum->text()!="5"))

{

QMessageBox::critical(*this*,"Ошибка","Введите номер недели правильно");*return* 0;

}*else* *return* a.toInt(0,10);

}

classes.h

#ifndef CLASSES\_H

#define CLASSES\_H

#include <QMainWindow>

#include <QString>

*namespace* Ui {

*class* Classes;

}

*class* Classes : *public* QMainWindow

{

Q\_OBJECT

*public*:

*explicit* Classes(QWidget \*parent = 0);

~***Classes***();

void **ShowClass**(QString txt);

*private*:

void **closed**();*//закрываем* *окно*

void **Exit**();

Ui::Classes \*ui;

};

#endif *//* *CLASSES\_H*

classes.cpp

#include "classes.h"

#include "ui\_classes.h"

#include "menu.h"

Classes::**Classes**(QWidget \*parent) :

QMainWindow(parent),

ui(*new* Ui::Classes)

{

ui->setupUi(*this*);

connect(ui->action, &QAction::triggered, *this*, &Classes::closed);

connect(ui->action\_2, &QAction::triggered, *this*, &Classes::Exit);

}

Classes::~***Classes***()

{

*delete* ui;

}

void Classes::**ShowClass**(QString txt)

{

ui->Auditories->setText(txt);

}

void Classes::**closed**()*//закрываем* *окно*

{

close();

}

void Classes::**Exit**()*//закрываем* *окно*

{

exit(0);

}

main.cpp

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

*return* a.exec(); }