МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра информационных систем

ОТЧЕТ

по курсовой работе

по дисциплине «Объектно-ориентированное программирование»

Тема: GUI приложение «Лифт»

Студенты гр. 3372 _____ Беляев К. В.
Лазарев Ф. Н.

Преподаватель _____ Егоров С.С.

Санкт-Петербург
2024

СОДЕРЖАНИЕ

1. ПОСТАНОВКА ЗАДАЧИ

1.1. Описание предметной области *Вариант 1*

В каждом из M (2) подъездов N-этажного (9) дома установлен лифт. Лифт имеет грузоподъемность К (4) человек. Все управление лифтом осуществляет пользователь (кроме перемещения между этажами в режиме движения).

Заданы следующие правила:

- лифт может находиться в состояниях «останов на этаже» и «движение вверх/вниз»,
- в состоянии «останов на этаже» дверь лифта открыта,
- в состоянии «останов на этаже» в лифт могут заходить или выходить пассажиры,
- на первом и последних этажах выходить должны все, приехавшие в лифте, независимо от входящих пассажиров,
- число пассажиров в лифте не должно превышать заданного предела,
- при нажатии кнопки вызова лифта на этаже необходимо указывать направление требуемого движения и число пассажиров в нем нуждающихся. На промежуточных этажах могут находиться пассажиры и вверх и вниз,
- лифт, двигающийся по направлению вызова, останавливается на этаже только в случае наличия свободных мест. При его остановке допустимое число пассажиров входят,
- пустой лифт начинает движение только при его вызове,
- лифт с пассажирами начинает движение при нажатии в нем кнопки движения, до этого должны быть выбраны этажи назначения для всех пассажиров (или группы), находящихся в лифте. Этот список очищается по мере освобождения лифта,
- время перемещения между этажами 1 сек. (промежуточное движение отображать не обязательно).

1.2. Синтаксический анализ задания

Таблица синтаксического анализа

Лексема	Элемент объектной модели
Лифт	Класс
Этаж	Класс
Пассажир	Класс
Дом	-
Подъезд	-
Грузоподъемность	Характеристика лифта
Содержит	Метод этажа
Вызывается	Метод лифта
Останавливается	Метод лифта

“Главное окно”, “Окно управления”, “Окно отображения” - классы, неявно выявленные в синтаксическом разборе.

2. ДИАГРАММЫ ОБЪЕКТНОЙ МОДЕЛИ

2.1. Модель «сущность-связь»

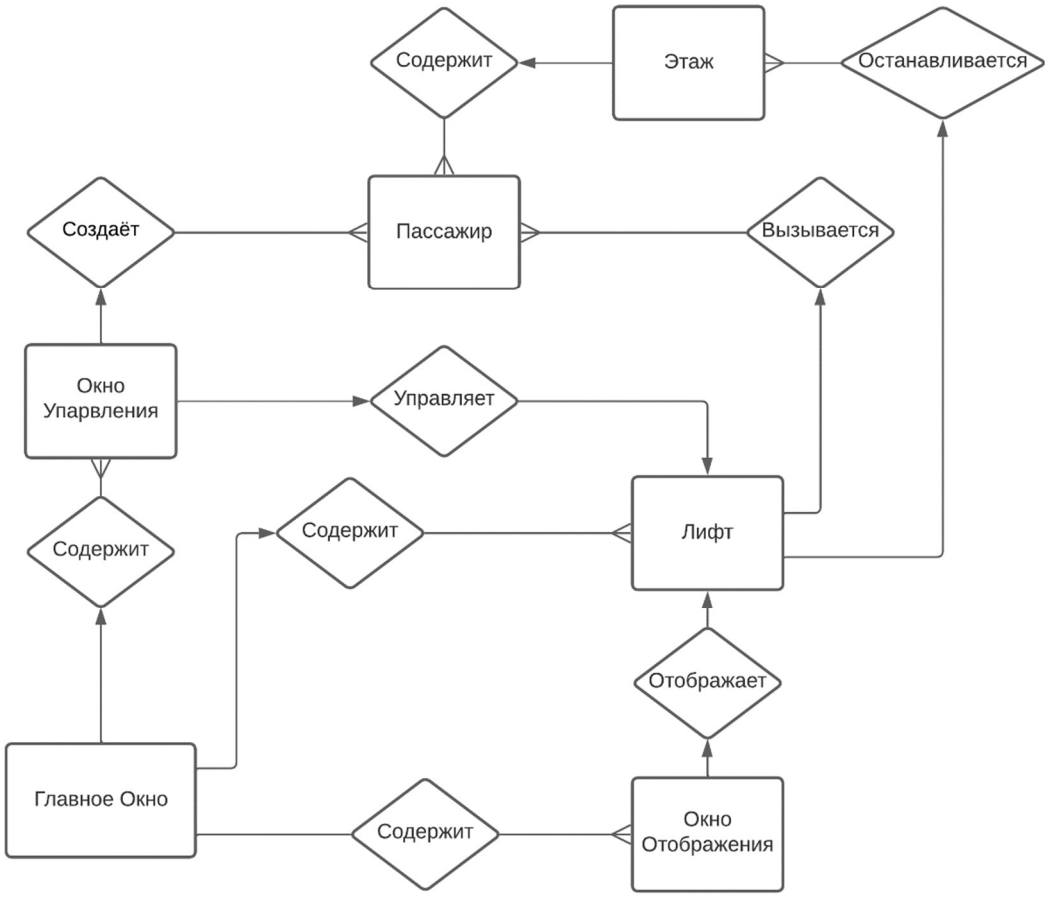


Рисунок 1 – Модель «сущность-связь»

2.2. Диаграмма классов

Диаграмма классов соответствует представленной на рисунке 1 модели «сущность-связь».

Таблица соответствия.

Предметная область	Модель “сущность-связь”	Диаграмма классов
Лифт	Лифт	Elevator
Этаж	Этаж	Floor

Пассажир	Пассажир	Passanger

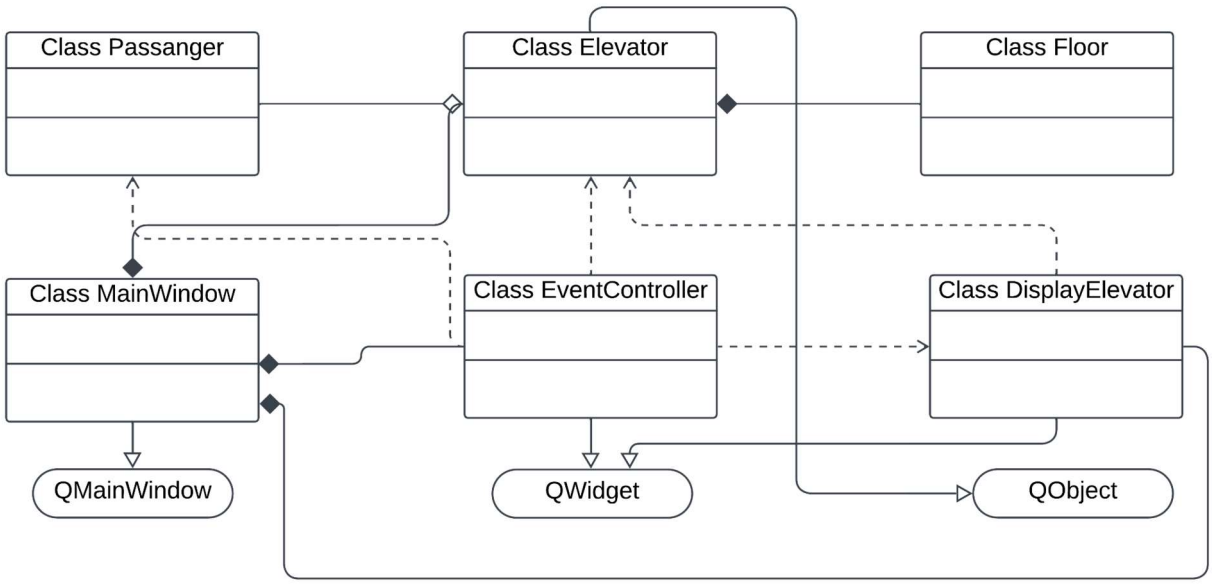


Рисунок 2 – Диаграмма классов

Описание классов и логика наследования классов, представленных на рисунке 2, отражены в таблице 1.

Таблица 1 - Описания классов и наследования

Наименование	Идентификатор	Назначение
Главное окно	MainWindow	Главное интерфейсное окно, объединяющее в себе другие
	Наследуется от:	QMainWindow
	Наследники:	-
Окно отображения	DisplayElevator	Отображает состояние лифта в каждый момент времени, список пассажиров на этажах
	Наследуется от:	QWidget
	Наследники:	-
Окно управления	EventController	Окно управления позволяет выставлять на выбранный этаж пассажиров. Пассажирам можно задать направление движения
	Наследуется от:	QWidget
	Наследники:	-
Лифт	Elevator	Лифт. Перемещает пассажиров между этажами
	Наследуется от:	QObject
	Наследники:	-
Этаж	Floor	Содержит в себе информацию о количестве пассажиров и их направлении для определенного этажа
	Наследуется от:	-

	Наследники:	-
Пассажир	Passanger	Пассажир, может перемещаться по этажам используя лифт
	Наследуется от:	-
	Наследники:	-

Атрибуты классов, представленных на рисунке 2, отражены в таблице 2.

Таблица 2 - Атрибуты классов

Наименование	Идентификатор	Назначение
Класс: MainWindow		
Графическая форма	Ui	Отображение графической формы окна.
Окно отображения для лифта 1	elevatorDisplay1	Отображение лифта 1.
Окно отображения для лифта 2	elevatorDisplay2	Отображения лифта 2.
Окно управления для лифта 1	controller1	Управление лифтом 1.
Окно управления для лифта 2	controller2	Управление лифтом 2.
Поток лифта 1	thread1	Поток для лифта 1.
Поток лифта 2	thread2	Поток для лифта 2.
Кнопка запуска программы	start	Кнопка запуска лифтов.
Разметка страницы 1	layout	Удобное размещение.
Разметка страницы 2	elevators	Удобное размещение.
Разметка страницы 3	controllers	Удобное размещение.
Лифт 1	elevator1	Лифт 1. Перемещает пассажиров.
Лифт 2	elevator2	Лифт 2. Перемещает пассажиров.
Класс: DisplayElevator		
Указатель на лифт	elevator	Позволяет отправлять информацию лифту и получать информацию от лифта.
Разметка страницы 1	layout	Позволяет передавать информацию окну отображения.
Таблица	display	Позволяет выставить количество пассажиров, едущих наверх.
Класс: EventController		
Указатель на лифт.	elevator	Позволяет отправлять информацию лифту и получать информацию от лифта.
Указатель на окно отображения.	display	Позволяет передавать информацию окну отображения.
Спинбокс пассажиров вверх.	upSpinBox	Позволяет выставить количество пассажиров, едущих наверх.
Спинбокс пассажиров вниз.	downSpinBox	Позволяет выставить количество пассажиров, едущих вниз.
Спинбокс этажей.	floorSpinBox	Позволяет выставить этаж, на котором необходимо задать пассажиров.
Текст спинбокса 1.	upLab	Текст для интерфейса.
Текст спинбоксов 2.	downLab	Текст для интерфейса.
Текст спинбоксов 3.	floorLab	Текст для интерфейса.

Кнопка подверждения	confirmButton	Подтверждает создание пассажиров.
Разметка страницы.	layout	Позволяет размещать элементы в окне.
Класс: Elevator		
Состояние лифта.	status	Вверх/вниз/остановка.
Текущий этаж.	currentFloor	Текущий этаж лифта.
Пассажиры в лифте.	currentPas	Вектор пассажиров, находящихся внутри лифта.
Этажи.	floors	Вектор этажей, содержащей информацию.
Очередь.	queue	Вектор, содержащий в себе пассажиров на каждом этаже.
Класс: Floor		
Номер этажа.	number	Номер этажа.
Количество пассажиров вверх.	countUp	Количество пассажиров на этаже, которые поедут вверх.
Количество пассажиров вниз.	countDown	Количество пассажиров на этаже, которые поедут вниз.
Класс: Passanger		
Текущий этаж	currentFloor	Этаж, на котором появляется пассажир.
Этаж назначения	destinationFloor	Этаж, на который пассажир отправится на лифте.

Методы классов, представленных на рисунке 2, отражены в таблице 3.

Таблица 3 - Методы классов

Идентификатор	Наименование
Класс: MainWindow	
MainWindow	Конструктор, создает главное окно приложения и задает в нем различные элементы интерфейса. Происходит инициализация двух объектов displayElevator и двух объектов eventController. Также создаются объекты виджетов и устанавливаются связи между элементами интерфейса.
elevatorsConnect	Запускает движение лифтов в отдельных потоках. Сначала вызывает метод moveToThread() для каждого лифта, чтобы переместить их в отдельные потоки. Затем устанавливает соединения между сигналом started() каждого потока и соответствующим слотом объекта лифта, чтобы запустить цикл работы лифта в каждом потоке. Запускает каждый поток с помощью метода start().
stop1	Прекращает работу потока первого лифта.
stop2	Прекращает работу потока второго лифта.
~MainWindow	Деструктор класса, удаляет графический интерфейс.
Класс: Passanger	
Passanger	Конструктор класса Passanger, который инициализирует нового пассажира значениями переданных аргументов (текущий этаж и выбранное направление движения) и вызывает метод для

	генерации случайного этажа, на который пассажир хочет переместиться.
randFloor	Метод использует генератор случайных чисел, чтобы сгенерировать этаж назначения для пассажира в заданном диапазоне.
getDestFloor	Возвращает значение поля destinationFloor – этаж, на который пассажир хочет переместиться.
Класс: Floor	
Floor	Конструктор, инициализирует новый объект этажа и устанавливает номер его этажа по переданному значению.
getCountUp	Получает количество пассажиров на этаже, желающих двигаться вверх.
getCountDown	Получает количество пассажиров на этаже, желающих двигаться вниз.
setCountUp	Устанавливает количество пассажиров на этаже, желающих двигаться вверх.
setCountDown	Устанавливает количество пассажиров на этаже, желающих двигаться вниз.
Класс: eventController	
eventController	Конструктор класса, инициализирует новый объект окна eventController и его элементы интерфейса. Принимает два аргумента: указатель на лифт и указатель на объект окна displayElevator. Виджеты добавляются в различные ячейки QGridLayout, устанавливаются соответствующие значения минимальных и максимальных значений QSpinBox, устанавливается разметка.
extremeFloors	Обновляет максимальные значения QSpinBox для выбора количества пассажиров на этаже, которые могут ехать вверх или вниз, в зависимости от этажа. Принимает номер этажа и при помощи оператора switch устанавливает максимальное значение вверх в 0, чтобы пассажиры не могли двигаться вверх с 9 этажа, или максимальное значение вниз в 0, чтобы пассажиры не могли двигаться вниз с 1 этажа. Для промежуточных этажей устанавливаются максимальные значения вверх и вниз в 6 (можно выбирать до 6 пассажиров вверх и 6 пассажиров вниз).
generatePassangers	Внутри метода извлекаются значения количества пассажиров, которые должны направляться вверх и вниз, а также текущий этаж из QSpinBox. Для каждого значения upCount и downCount метод создает новый объект пассажира с текущим этажом и желаемым направлением. Затем добавляет созданного пассажира в соответствующую очередь на текущем этаже лифта. Обновляет отображение количества пассажиров на текущем этаже в объекте displayElevator, и значения счетчиков пассажиров на текущем этаже в объекте Floor.
Класс: Elevator	
Elevator	Конструктор класса, инициализирует новый объект лифта и его поля. Внутри метода создаются объекты каждого этажа в здании и добавляются в вектор. Таким образом, лифт получает вектор этажей и очередь для каждого этажа.
getCurPas	Возвращает количество пассажиров, находящихся внутри лифта в данный момент.

takePassangers	Внутри метода проверяется, есть ли еще места в лифте, чтобы загрузить новых пассажиров, и есть ли пассажиры в очереди на текущем этаже. Если лифт полон или на этаже нет пассажиров, метод завершается. Если проверка пройдена, загружает пассажиров внутрь лифта из текущего этажа.
mDestFloor	Внутри метода сначала проверяется, есть ли пассажиры внутри лифта. Затем определяется текущее направление движения лифта и находится максимальное/минимальное значение целевого этажа среди пассажиров, находящихся внутри лифта. Если лифт движется вверх, метод находит максимальное значение целевого этажа пассажиров, иначе метод находит минимальное значение. Возвращается значение целевого этажа.
releasePassangers	Выпускает пассажиров, которые достигли своего целевого этажа и находятся внутри лифта. Если лифт пуст, метод завершается без действий.
isWaiting	Проверяет, ожидает ли лифт новых пассажиров на текущем этаже или на каком-либо другом этаже, и возвращает соответствующее логическое значение.
chooseDestFloor	Определяет, куда должен отправиться лифт, исходя из наличия пассажиров в очередях на этажах здания и текущего положения лифта.
loop	Запускает бесконечный цикл и управляет движением лифта, забирает и выпускает пассажиров на каждом этаже, обновляет позицию лифта и отправляет сигналы для обновления отображения.
Класс: displayElevator	
displayElevator	Конструктор класса создает интерфейс, который отображает информацию о состоянии лифта и ожидающих пассажирах на каждом этаже здания в виде таблицы QTableWidget.
redraw	Обновляет соответствующую ячейку таблицы, отображающую информацию о состоянии лифта и ожидающих пассажирах на этажах здания, в зависимости от текущего этажа и направления движения лифта.
release	Обновляет соответствующую ячейку таблицы, отображающую информацию о состоянии лифта и ожидающих пассажирах на этажах здания, после высвобождения всех пассажиров на текущем этаже. В ячейке отображается количество пассажиров внутри лифта и направление его движения. Чтобы обозначить текущий этаж лифта, для его ячейки устанавливается фоновый цвет.
take	Обновляет соответствующую ячейку таблицы, отображающую количество пассажиров, ожидающих лифт на этаже. В ячейке отображается количество пассажиров, ожидающих лифт в направлениях вверх и вниз.

Отношения между классами, представленными на рисунке 2, отражены в таблице 4.

Таблица 4 - Отношения между классами

Класс с исх. стрелкой	Класс с вх. стрелкой	Вид отношения
MainWindow	QMainWindow	Обобщение
EventController	MainWindow	Композиция
DisplayElevator	MainWindow	Композиция
Elevator	MainWindow	Композиция
Floor	Elevator	Композиция
DisplayElevator	Elevator	Зависимость

2.3. Перечень библиотечных конструкторских классов, использованных для построения диаграммы классов

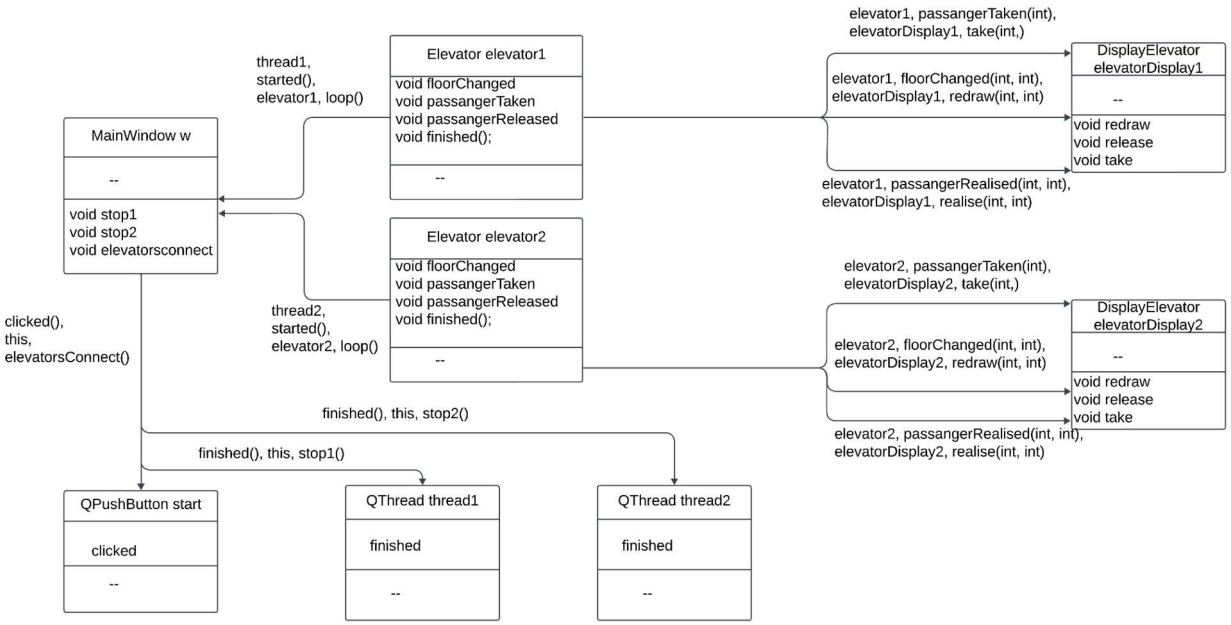
QWidget – добавляет механизмы создания GUI интерфейса.

QObject – добавляет механизмы управления сигналами и слотами.

QMainWindow – класс, представляющий окно приложения в Qt.

2.4. Схема соединений «сигнал-слот»

Схема соединений «сигнал-слот» представлена на рисунке 3. Каждый класс представлен в виде блока, где: 1 – имя, 2 – сигналы, 3 – слоты. Каждое соединение подписано соответствующей командой «connect».



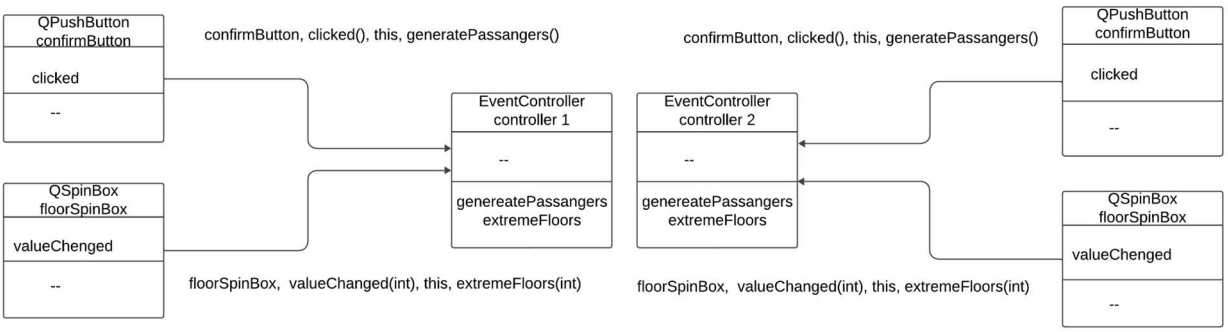


Рисунок 3 – Схема соединений «сигнал-слот»

3. ПОДСИСТЕМА «ИНТЕРФЕЙС»

3.1. Графическое представление интерфейсных окон

3.1.1. Основное окно

Главное окно приложения реализуется классом MainWindow. В нем создаются экземпляры лифтов, окон отображения и управления.

Также оно содержит кнопку запуска программы.

3.1.2. Окно параметров и отображения состояния объектов ПрО

Данное окно реализовано классом DisplayElevator, оно получает информацию от лифта и окна управления и отображает ее в виджете-таблице (QtableWidget).

3.1.3. Окно управления событиями ПрО

Окно управления событиями реализуется классом EventController, окно позволяет указать в спинбоксах этаж, на котором требуется сгенерировать пассажиров, количество пассажиров, едущих вверх и количество пассажиров едущих вниз. Кнопкой «ОК» можно занести эти данные в программу, они отобразятся на окне отображения.

3.1.4. Заголовочные файлы (h-файлы) интерфейсных классов

Основное окно — mainwindow.h, класс MainWindow:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "eventController.h"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
```

```
{
    Q_OBJECT
    DisplayElevator *elevatorDisplay1, *elevatorDisplay2;
    EventController *controller1, *controller2;

    QThread thread1, thread2;

    QPushButton start;
    QVBoxLayout Layout;
    QHBoxLayout elevators, controllers;

    Elevator elevator1, elevator2;

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

public slots:
    void stop1();
    void stop2();
    void elevatorsConnect();

private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

Окно отображения — displayElevator.h, класс DisplayElevator:

```
#ifndef DISPLAYELEVATOR_H
#define DISPLAYELEVATOR_H

#include <QTableWidget>
#include <QHeaderView>
#include "elevator.h"

class DisplayElevator : public QWidget
{
    Q_OBJECT

    Elevator *elevator;
    QHBoxLayout layout;
public:
    QTableWidget display;
    DisplayElevator(Elevator *elev);

public slots:
    void redraw(int floor, int direction);
    void release(int floor, int direction);
    void take(int floor);
};
```

Окно отображения — eventController.h, класс EventController:

```
#ifndef EVENTCONTROLLER_H
#define EVENTCONTROLLER_H

#include <algorithm>
#include "displayElevator.h"

class EventController : public QWidget
{
    Q_OBJECT
    Elevator *elevator;
    DisplayElevator *display;

    QSpinBox upSpinBox;
    QSpinBox downSpinBox;
    QSpinBox floorSpinBox;
    QLabel floorLab, upLab, downLab;
    QGridLayout layout;
    QPushButton confirmButton;
public:
    EventController(Elevator *elevator, DisplayElevator
*display);


public slots:
    void generatePassangers();
    void extremeFloors(int i);

};

#endif // EVENTCONTROLLER_H
```

4. ПОДСИСТЕМА «МОДЕЛЬ»

4.1. Перечень событий, изменяющих состояние модели ПрО

Следующие события могут повлиять на состояние модели ПрО:

1) Вставка чисел в спинбоксы и нажатие кнопки «ОК» — в программу отправятся введенные данные, появятся на окне отображения и будут обрабатываться лифтом.

2) Нажатие кнопки «Запуск» — запускает движение лифтов, которые будут двигаться до тех пор, пока не закончатся пассажиры на этажах.

4.2. Заголовочные файлы (h-файлы) классов модели ПрО

Класс лифта, описывающий поведение лифта, файл elevator.h:

```
#ifndef ELEVATOR_H
#define ELEVATOR_H

#include <QDebug>

#include <QObject>
#include <QWidget>
#include <QVBoxLayout>
#include <QHBoxLayout>
#include <QGridLayout>
#include <QSpinBox>
#include <QPushButton>
#include <QLabel>
#include <QThread>
#include <QVector>
#include "floor.h"
#include "passanger.h"


#define UP 1
#define IDLE 0
#define DOWN -1


class Elevator : public QObject
{
    Q_OBJECT
    int status = IDLE;
    int currentFloor = 0;
    QVector<Passanger> currentPas;
```

```
    void takePassangers();
    void releasePassangers();
    int mDestFloor();
    bool isWaiting();
    int chooseDestFloor();
    QThread thread1;

public:
    QVector<Floor> floors;
    QVector<QVector<Passanger>> queue;

    Elevator();
    int getCurPas();
    int getCurFlr();

public slots:
    void loop();


signals:
    void floorChanged(int floor, int dir);
    void passangerTaken(int floor);
    void passangerReleased(int floor, int dir);
    void finished();
};

#endif // ELEVATOR_H
```

Класс этажа, файл floor.h:

```
#ifndef FLOOR_H
#define FLOOR_H


class Floor
{
    int number;
    int countUp = 0;
    int countDown = 0;

public:
    Floor();
    Floor(int i);
    void setCountUp(int c);
    void setCountDown(int c);
    int getCountUp();
    int getCountDown();
};

#endif // FLOOR_H
```

Класс пассажира, файл passanger.h:

```
#ifndef PASSANGER_H
#define PASSANGER_H
#include <random>
enum direction{
    LAST,
    FIRST
};
class Passanger
{
    int currentFloor;
    int destinationFloor;
    int randFloor();
public:
    direction where;
    Passanger();
    Passanger(int c, direction w);
    int getDestFloor();
};
#endif // PASSANGER_H
```

5. СКРИНШОТЫ КОНТРОЛЬНОГО ПРИМЕРА РАБОТЫ РЕАЛИЗОВАННОГО ПРИЛОЖЕНИЯ

При запуске приложения открывается окно, представлено на рисунке 1. В данном окне можно выставлять пассажиров на нужные этажи тремя спин-боксами(Рисунок 2), после чего нажать кнопку “Запуск” и запустить лифты (Рисунок 3).

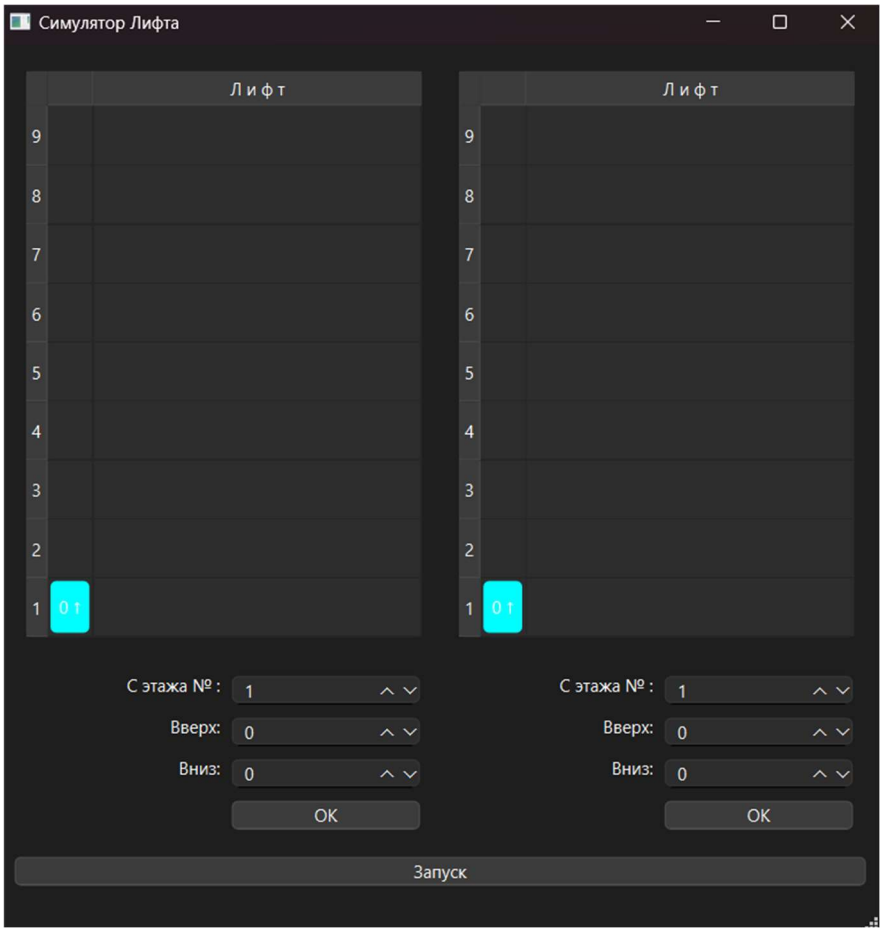


Рисунок 4 – Окно запуска

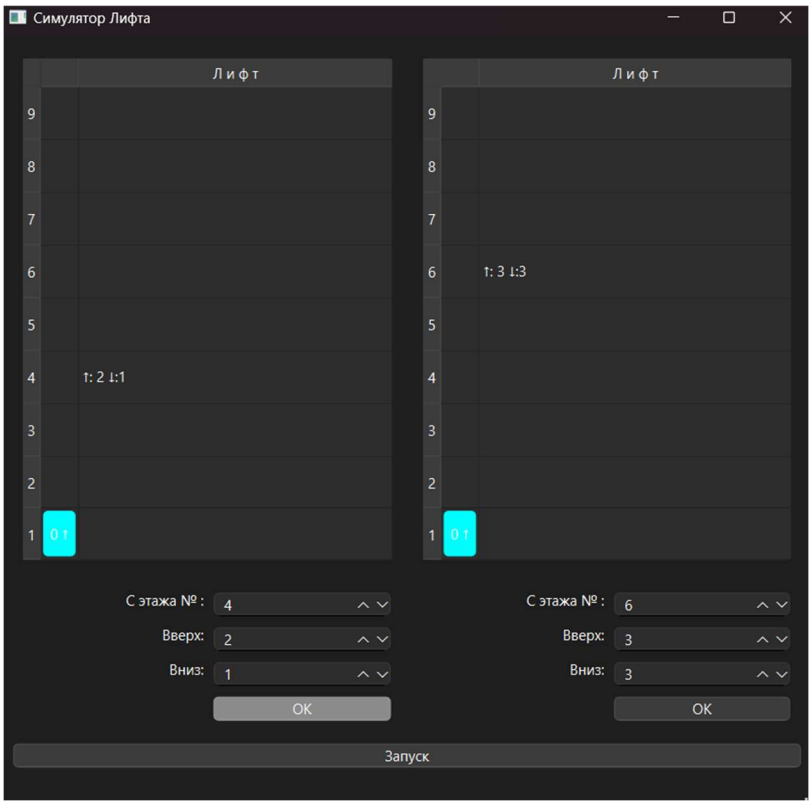


Рисунок 5 –Количество пассажиров

Рисунок 6 – Запуск лифтов

После того как все пассажиры доедут до своих этажей лифты остановятся (Рисунок 7).

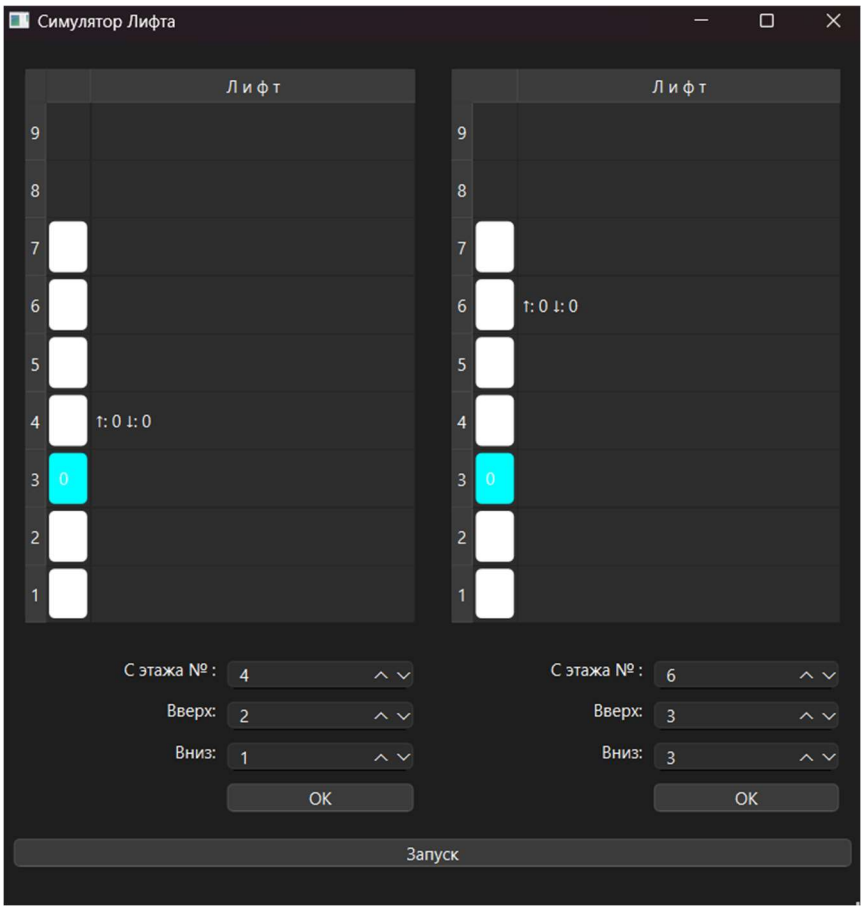


Рисунок 7 – Лифты остановились

6. ВЫВОДЫ ПО КУРСОВОЙ РАБОТЕ

В результате выполнения курсовой работы на языке C++ реализовано приложение – GUI приложение «Симулятор Лифта». Для реализации использовались конструкторские классы Qt.

Был проведен анализ предметной и синтаксической областей, выделены объекты и связи между ними, была построена модель «сущность-связь».

Была построена диаграмма классов в соответствии с моделью «сущность-связь» и написанной программой. К диаграмме классов были представлены таблицы с описаниями, наследованием, атрибутами и методами, таблица отношений и таблица соответствия.

Была построена схема соединений «сигнал-слот».

Реализованное приложение представляет из себя объектно-ориентированную программу, состоящую из модулей “Интерфейс” и “Модель”.

Модуль Интерфейс состоит из 3-х интерфейсных окон:

1) MainWindow – основное окно
2) DisplayElevator – окно отображения
3) EventController – окно управления событиями

Модуль “Модель” состоит из класса, описывающего логику симулятора, класса описывающего пассажира и класса описывающего этажи.

Работы приложения была протестирована на контрольном примере и представлена в разделе 5 отчета.