Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт Информационных Технологий и Анализа Данных

**Название работы** – “Разработка приложения с оконным графическим пользовательским интерфейсом”

Отчет по лабораторной работе “Лабораторная работа №6”

по дисциплине Объектно-ориентированное программирование

Вариант 24

Выполнил

Студент, номер группы ИСМб-19-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.Д. Солопов

Дата: \_\_\_\_\_\_\_\_\_\_

Принял

Должность \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.В. Маланова

Дата: \_\_\_\_\_\_\_\_\_\_

Иркутск 2021 г.

Содержание

[1 Постановка задач 3](#_Toc66770201)

[2 Проектирование графического интерфейса 4](#_Toc66770202)

[3 Проектирование базы данных 13](#_Toc66770203)

[4 Проектирование классов 16](#_Toc66770204)

[5 Таблица тестов 18](#_Toc66770205)

[6 Результаты тестов 20](#_Toc66770206)

[7 Листинг исходного кода 25](#_Toc66770207)

[8 Заключение 73](#_Toc66770208)

1 Постановка задач

**Цель работы:**

1. Получение общего представления об объектном подходе к организации доступа к базе данных (БД), выполнению запросов и обработке результатов.

2. Получение практического опыта организации работы с БД средствами платформы java.

**Задача:**

Предметная область: Библиотека

Имеются книги (регистрационный номер, название, количество страниц, год издания, раздел - учебник, художественная общественно-политическая и т.д. ) и читатели (ФИО, домашний адрес, паспортные данные).

Необходимо регистрировать дату, когда какой - либо читатель берет или возвращает книгу.

Выходные документы:

1. список читателей, которые держат на руках книги более месяца, с перечислением названия книги и даты выдачи, упорядоченный по датам выдачи, с указанием количества книг, которые должен сдать каждый читатель;

2. для заданного читателя выдать список прочитанных им книг, сортируя по датам получения.

**Содержание задания:**

Для выполнения задания необходимо разработать java-приложение с графическим пользовательским интерфейсом позволяющее просматривать и редактировать информацию из базы данных. Работу предваряет организация и наполнение базы данных (БД) в соответствии с предметной областью из индивидуального задания.

Графический пользовательский интерфейс должен предоставлять возможности работы с записями основных таблиц БД. Состав используемых компонентов определяется самостоятельно, но для их размещения должны быть предусмотрены соответствующие планировщики раскладки компонентов в основных контейнерах.

Классы, реализующие графический пользовательский интерфейс, должны содержать только обращения к методам классов для выполнения заданий.

2 Проектирование графического интерфейса

Интерфейс программы спроектирован с помощью библиотеки JavaFX. Управление программой разбито на три окна, связанных с таблицами. В окне “Регистрация” имеется интерфейс для решения задач, согласно поставленным задачам.

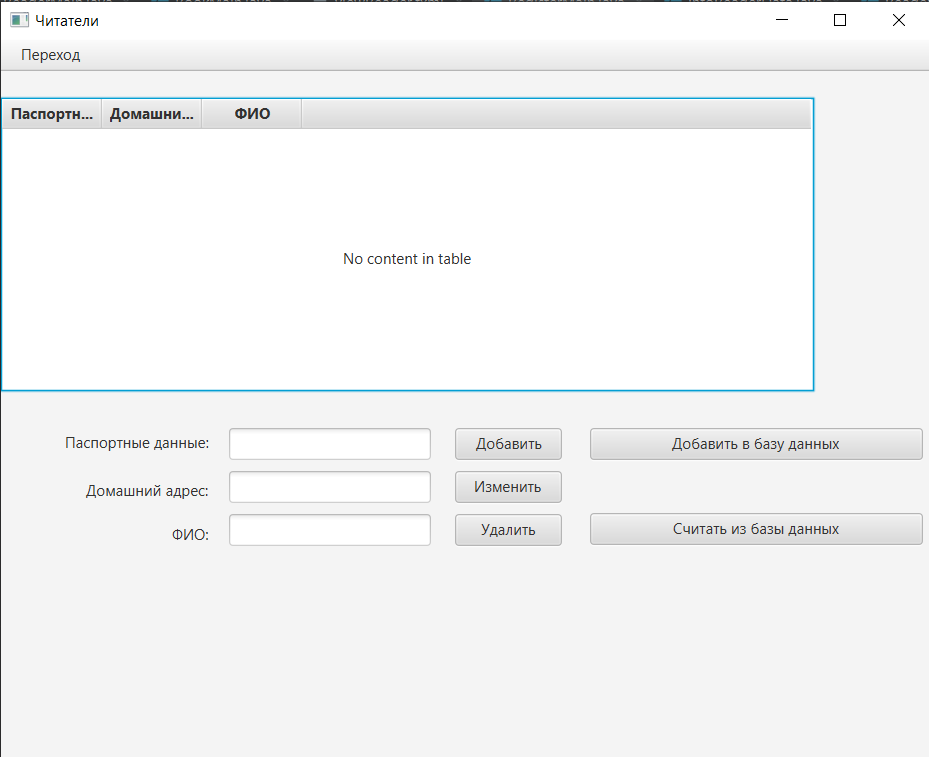


Рисунок 1 – Окно “Читатели”, которое позволяет взаимодействовать со списком читателей

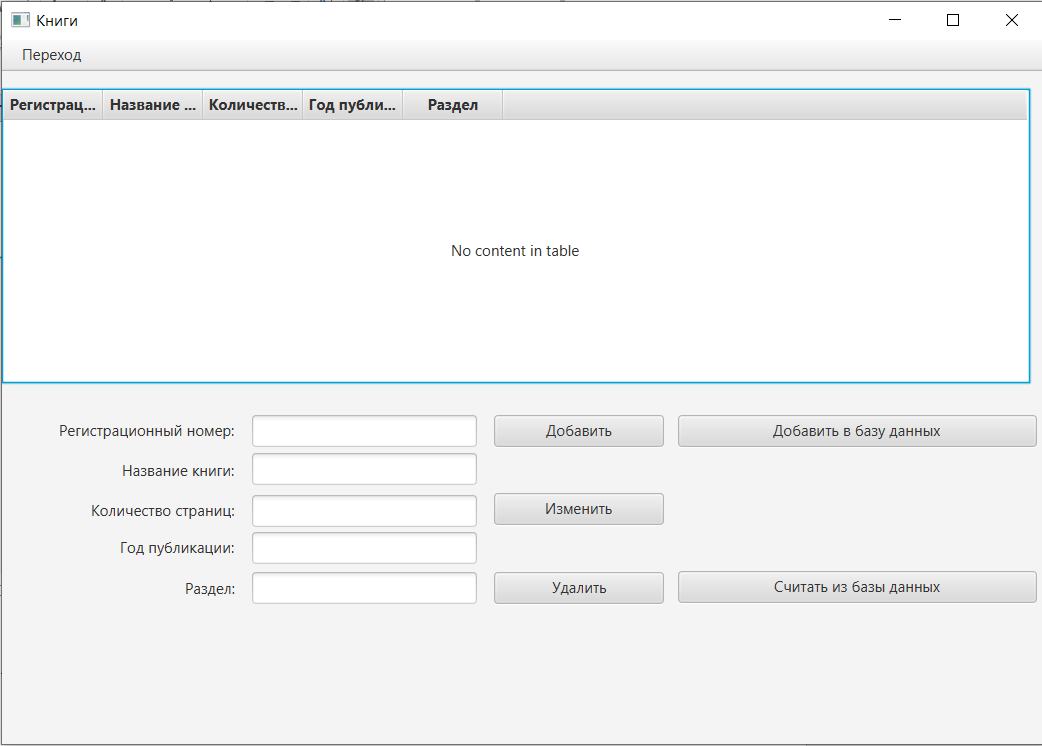


Рисунок 2 – Окно “Книги”, которое позволяет взаимодействовать со списком доступных книг

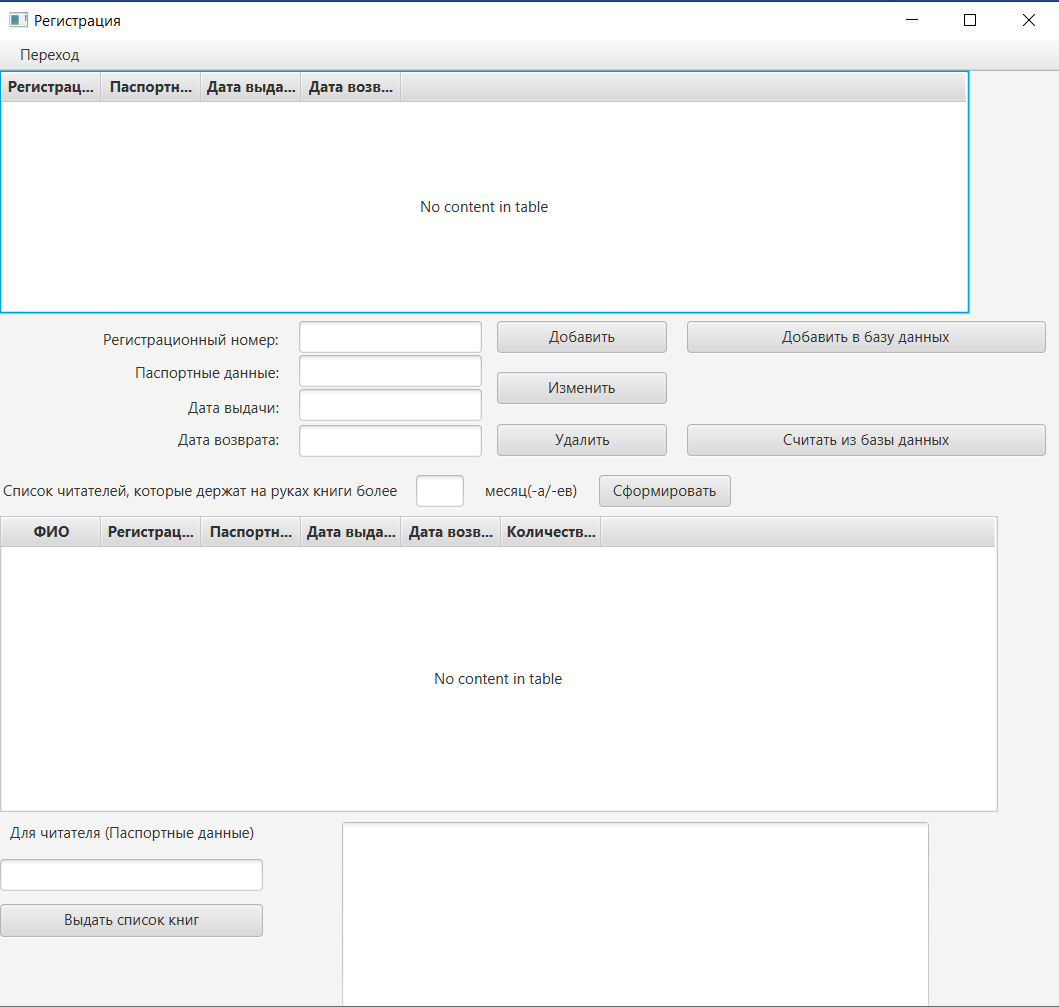


Рисунок 3 – Окно “Регистрация”, которое позволяет взаимодействовать со списком регистрации и решать определённые задачи

Переход между тремя окнами осуществляется с помощью вложенного меню “Переход”, нажав на который пользователь увидит допустимый список переходов. После нажатия на определённый список, текущее окно закрывается и открывается окно, выбранное пользователем.

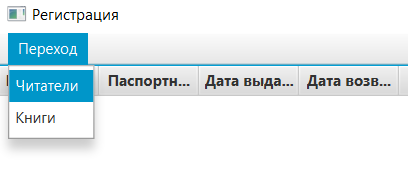


Рисунок 4 – Возможные переходы для окна “Регистрация”

Алгоритм добавления записей в таблицу, удаления и изменения их в каждом окне одинаков, по этой причине будет разобрано взаимодействие с таблицей в окне “Читатели”.

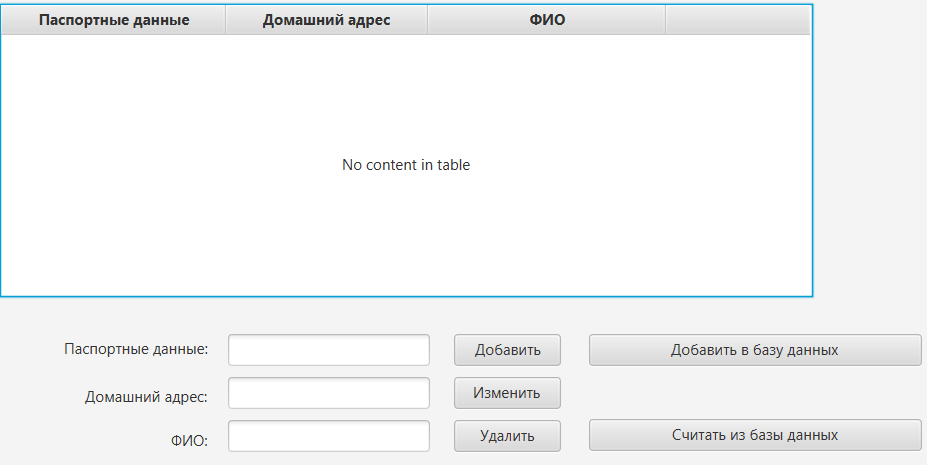


Рисунок 5 – Интерфейс взаимодействия с таблицей и базой данных MySQL

Для добавления записи о читателе, необходимо ввести его паспортные данные, домашний адрес и ФИО в текстовые блоки (TextField), расположенные напротив соответствующих им по смыслу текстовых полей (Label). После внесение данных в текстовые блоки можно добавить данную запись в таблицу, нажав на кнопку “Добавить”.

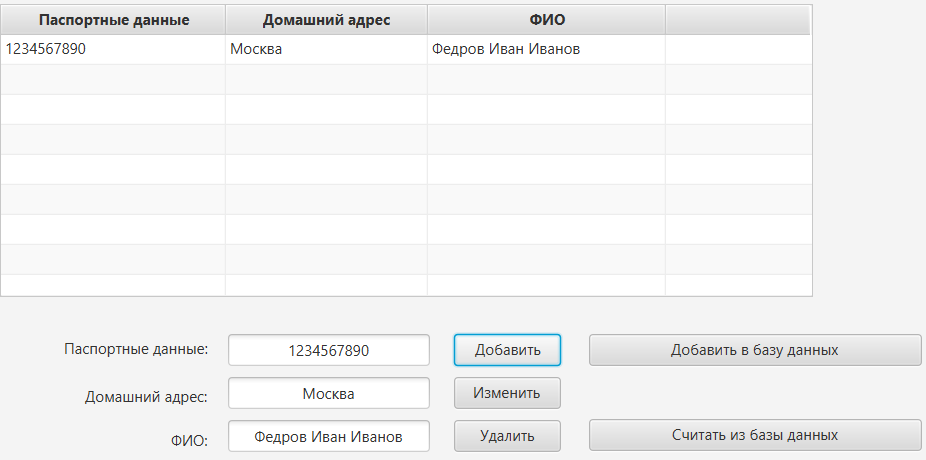


Рисунок 6 – Добавление записи в таблицу

Любую запись в таблице можно изменить. Для этого нужно направить курсор мыши на любую строку таблицы, нажать на неё (для выделения), ввести данные в текстовые блоки и нажать на кнопку изменить.

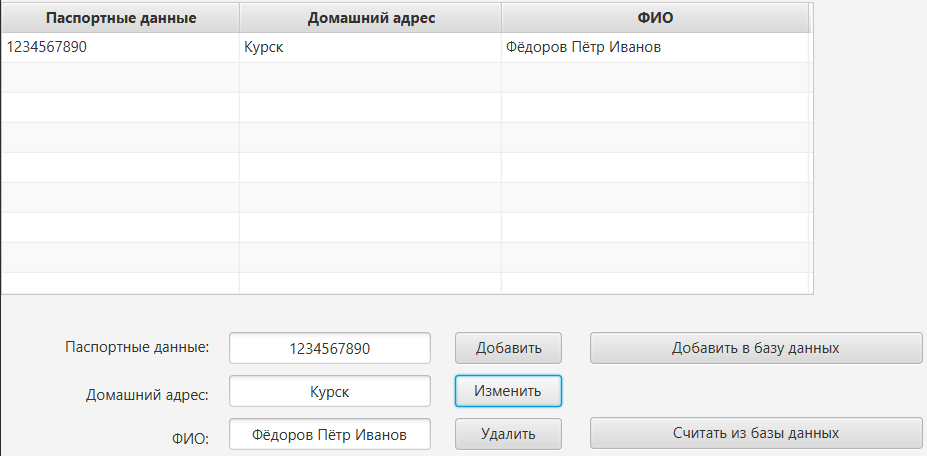


Рисунок 7 – Изменение записи в таблице (сравните с рис. 6)

Также, любую запись в таблице можно удалить. Для этого достаточно щелкнуть левой кнопкой мыши на строку, которую необходимо удалить (для её выделения), и нажать на кнопку “Удалить”.

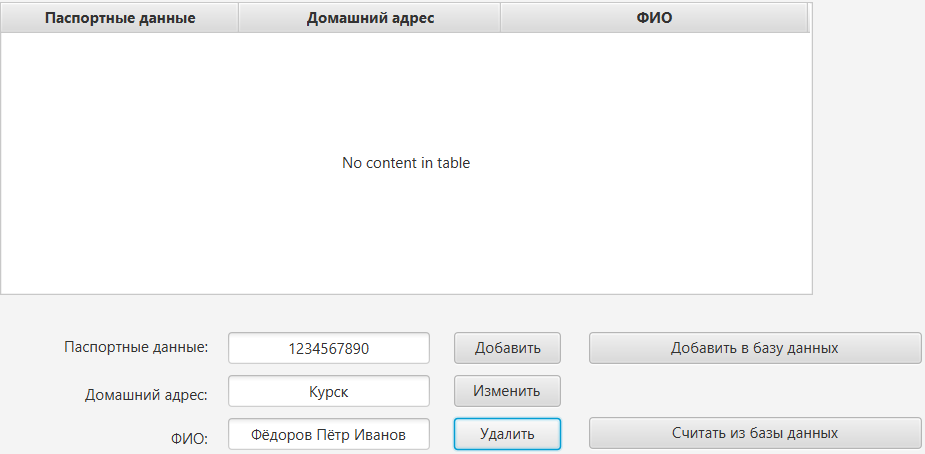


Рисунок 8 – Удаление строки в таблице

Для того, чтобы записать данные таблицы в соответствующую ей таблицу MySQL, необходимо нажать на кнопку “Добавить в базу данных”.

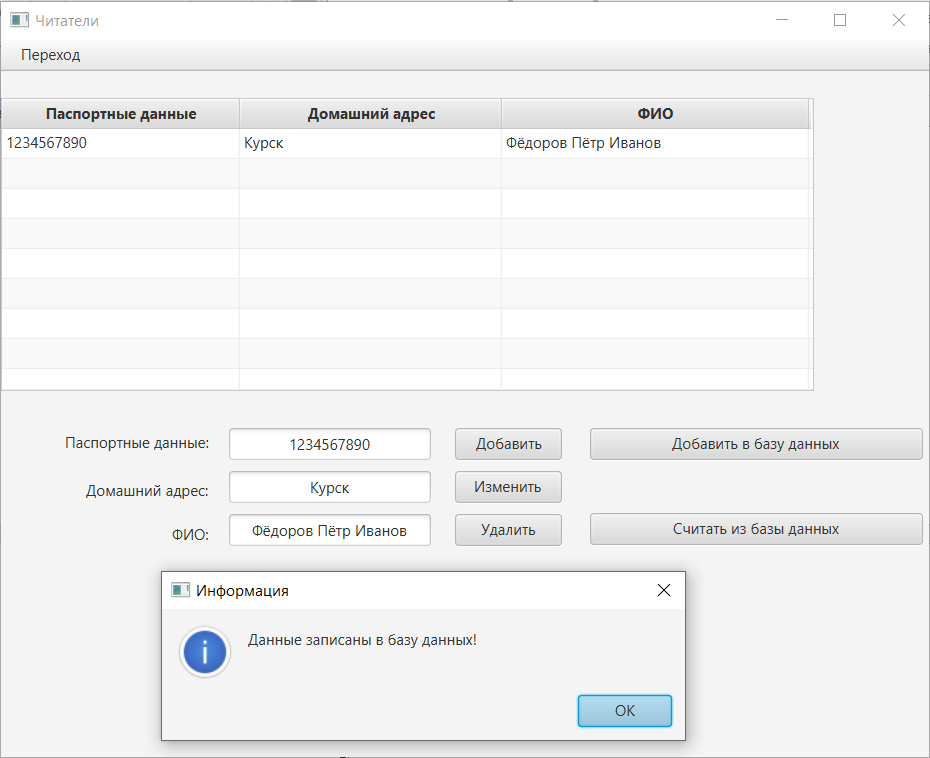


Рисунок 9 – Добавление данных таблицы в базу данных MySQL

Для заполнения пустой таблицы данными из соответствующей ей таблицы базы данных MySQL, необходимо нажать на кнопку “Считать из базы данных”.

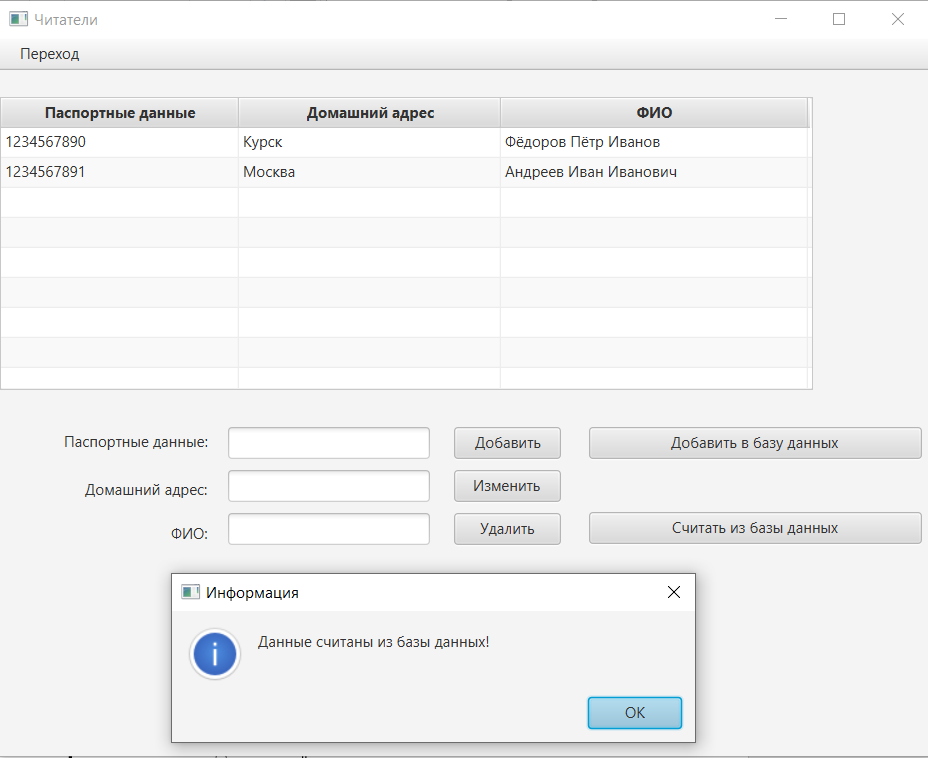


Рисунок 10 – Считывание данных из базы данных

Для решения практических задач, поставленных в данной лабораторной работе, необходимо предварительное наполнение базы данных.

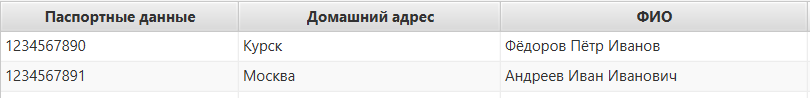


Рисунок 11 – Наполненная таблица “Читатели”

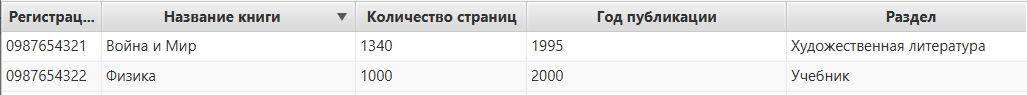


Рисунок 12 – Наполненная таблица “Книги”

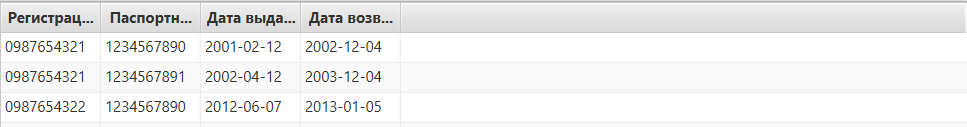


Рисунок 13 – Наполненная таблица “Регистрация”

Интерфейс для решения практических задач расположен в окне “Регистрация”.

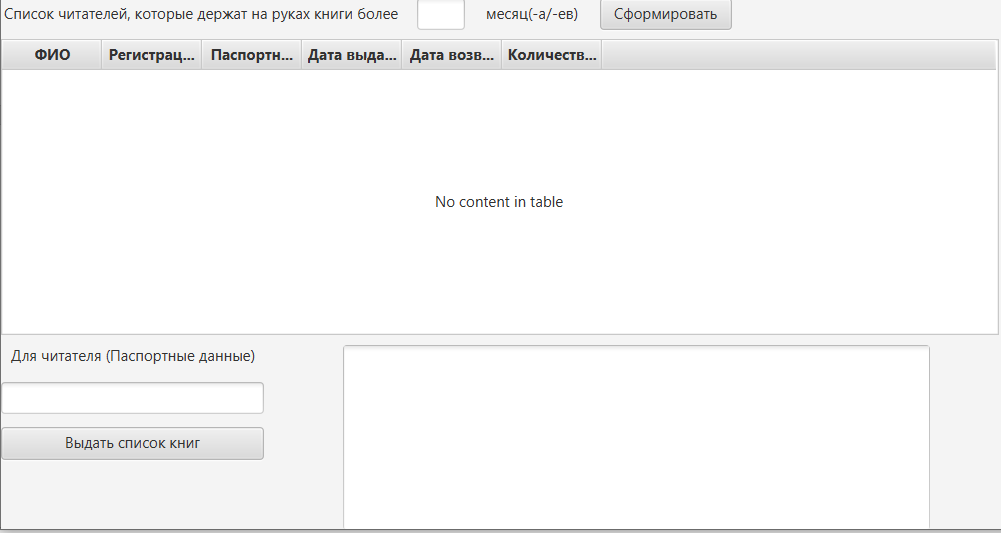


Рисунок 14 – Интерфейс для решения практических задач

При решении практических задач программа руководствуется фактическим наполнением базы данных MySQL, то есть при решении данные из БД считываются.

Для формирования списка читателей, которые держат на руках книги более n месяцев, необходимо количество месяцев ввести в текстовое поле, а затем нажать на кнопку “Сформировать”.

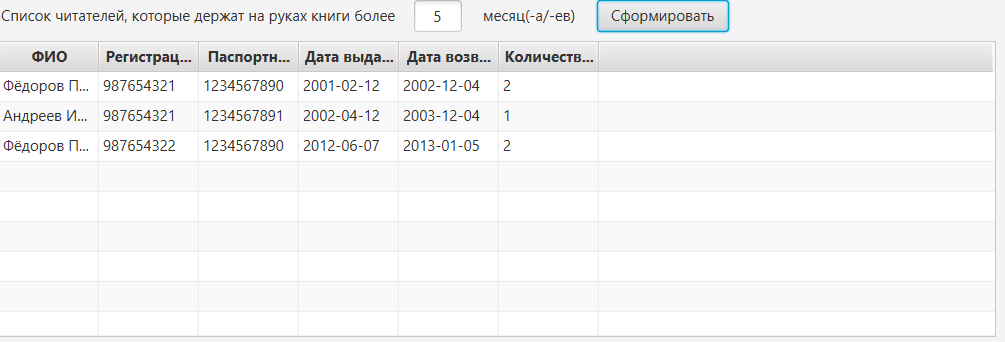


Рисунок 15 – Пример решения задания №1

Для информирования о том, какие книги выданны определённому читателю необходимо прописать в текстовое поле паспортные данные читателя, а затем нажать на клавишу “Выдать список книг”.

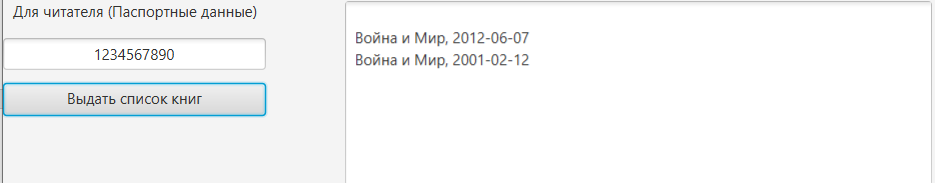


Рисунок 16 – Пример решения задания №2

3 Проектирование базы данных

Степень связи N:M. Необходимо регистрировать дату получения книги читателем и дату возврата книги обратно. Данные о книгах содержаться в отдельном экземпляре объекта, как и данные о читателях. Экземпляры объектов “Читатель” и “Книга” не связаны внешними ключами указывающими друг на друга и хранят информацию не зависимо. Экземпляр объекта “Книга” не зависит от экземпляра объекта “Читатель”. Между ними в связи существует посредник – экземпляр объекта “Регистрация записи”, который должен содержать внешние не уникальные ключи на экземпляры объектов, поскольку в таблице. “Регистрация записи” может регистрироваться ситуация когда один и тот же читатель с одними и теми же паспортными данным получает и возвращает одну и ту же книгу несколько раз с одними и теми же регистрационными данными.

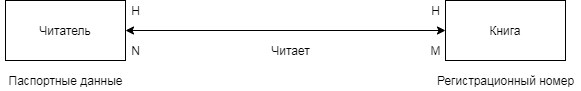


Рисунок 17 – ER-Диаграмма

Если степень связи M:N, то необходимо построение трех отношений: по одному на каждый объект (первичные ключи отношений совпадают с ключами объектов) и связующего отношения, первичным ключом которого будет комбинация ключей объектов.

Для ER-диаграммы получаем по 3 отношения, добавляя связующее звено (экземпляр объекта “Регистрация записи”) первичным ключом, которого является комбинация ключей объектов “Читатель” и “Книга”.

Читатель (Паспортные данные)

Книга (Регистрационный номер, Паспортные данные)

Регистрация записи (Паспортные данные, Регистрационный номер)

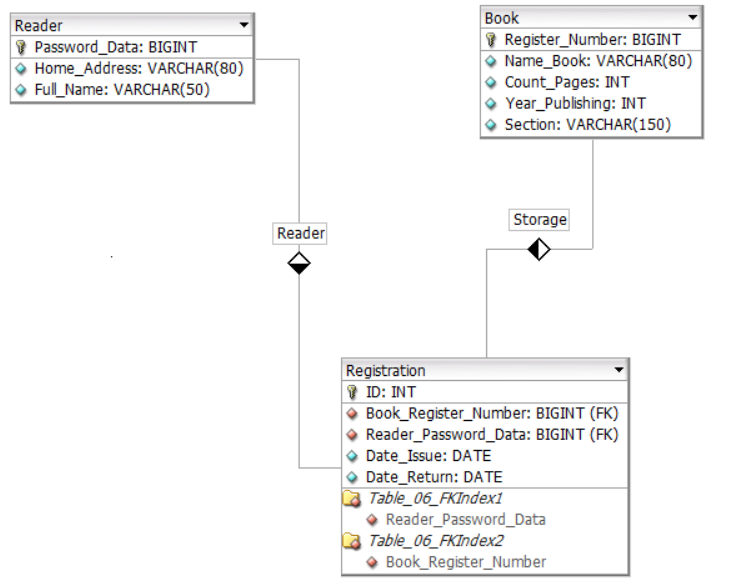


Рисунок 18 – Логическая модель базы данных

Для практической реализации базы данных за основу взята база данных MySQL и язык, для взаимодействия с базой данных – SQL.

SQL-Скрипт создания таблицы для хранения книг:

CREATE TABLE Book (

Register\_Number BIGINT PRIMARY KEY NOT NULL,

Name\_Book NVARCHAR(80) NULL,

Count\_Pages INT NULL,

Year\_Publishing INT NULL,

Section NVARCHAR(150) NULL

);

SQL-Скрипт создания таблицы для хранения читателей:

CREATE TABLE Reader (

Password\_Data BIGINT PRIMARY KEY NOT NULL,

Home\_Address NVARCHAR(80) NULL,

Full\_Name NVARCHAR(50) NULL,

);

SQL-Скрипт создания таблицы регистрации:

CREATE TABLE Register(

ID INT PRIMARY KEY AUTO\_INCREMENT NOT NULL,

Register\_Number BIGINT NULL,

Password\_Data BIGINT NULL,

Date\_Issue DATE NULL,

Date\_Return DATE NULL,

FOREIGN KEY (Register\_Number) REFERENCES Book (Register\_Number),

FOREIGN KEY (Password\_Data) REFERENCES Reader (Password\_Data));

4 Проектирование классов

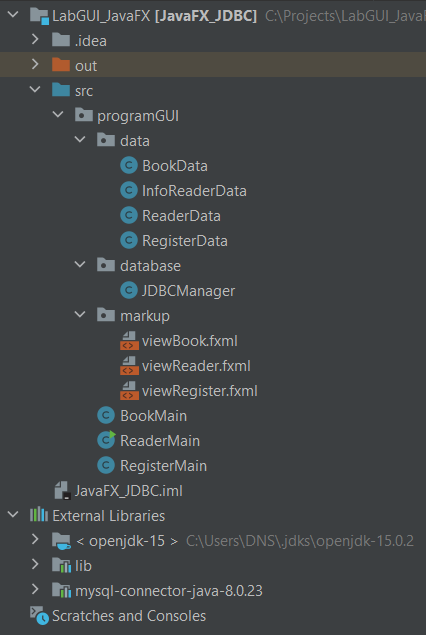


Рисунок 19 – Структура проекта

В проекте присутствуют 8 пользовательских классов, а также 4 файла с расширением fxml, для разметки пользовательского интерфейса.

Классы BookData, InfoReaderData, ReaderData, RegisterData используются как единицы данных для объектов типа TableView.

Класс JDBCManager предоставляет интерфейс для взаимодействия с базой данных MySQL. Например, создание таблиц, чтение из таблиц и запись в таблицы.

Класс BookMain реализует логику взаимодействия пользователя с таблицей, хранящей данные о книгах.

Класс ReaderMain реализует логику взаимодействия пользователя с таблицей, хранящей данные о читателях.

Класс RegisterMain реализует логику взаимодействия пользователя с таблицей, которая регистрирует выдачу и возврат книги из библиотеки.

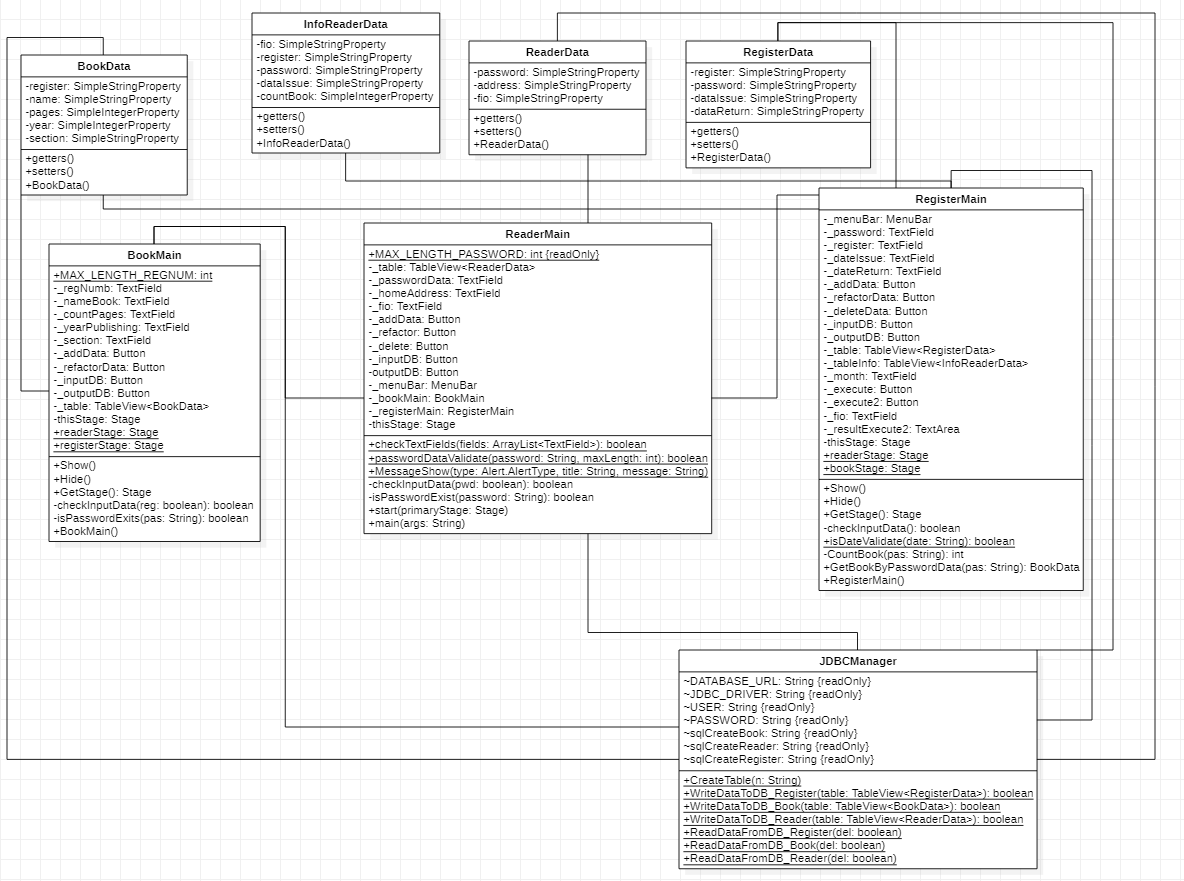


Рисунок 20 – UML-Диаграмма классов

5 Таблица тестов

Таблица 1 – Таблица тестов

|  |  |  |  |
| --- | --- | --- | --- |
| **N теста** | **Назначение теста** | **Входные данные для теста** | **Выходные данные для теста** |
| 1 | Ввод данных в таблицу читателей | Паспортные данные: 1234567881  Домашний адрес: Москва  ФИО: Иванов Иван Иванович | (см. рис. 21) |
| 2 | Удаление данных из таблицы читателей | Нажатие на кнопку “Удалить” при выделении строки | (см. рис. 22) |
| 3 | Изменение данных в таблице читателей | Паспортные данные: 1887654321  Домашний адрес: Ростов  ФИО: Иванов Иван Андреевич | (см. рис. 23) |
| 4 | Добавление в базу данных читателей | - | (см. рис. 24) |
| 5 | Чтение из базы данных читателей | - | (см. рис. 25) |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | Ввод данных в таблицу читателей (ошибка: не все поля введены) | Паспортные данные: 1234554321  Домашний адрес: Москва | (см. рис. 26) |
| 7 | Ввод данных в таблицу читателей (ошибка: данные с такими паспортными данными уже есть) | Паспортные данные: 1234567890  Домашний адрес: Курск  ФИО: Фёдоров Пётр Иванов | (см. рис. 27) |
| 8 | Ввод данных в таблицу регистрации (ошибка: дата выдачи не может быть позже даты возврата) |  | (см. рис. 28) |
| 9 | Формирование списка читателей, держащие на руках книги более n месяцев | n = 9 | (см. рис. 29) |
| 10 | Вывод списка книг для читателя | Паспортные данные: 1234567891 | (см. рис. 30) |

6 Результаты тестов

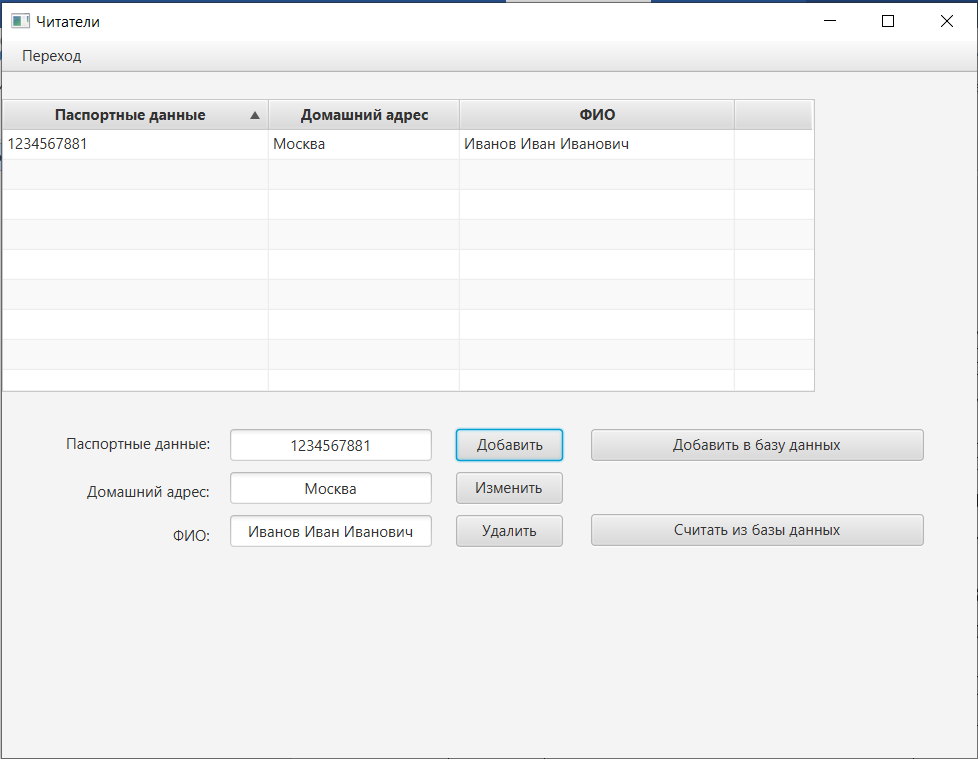


Рисунок 21 – Результат теста №1 из таблицы тестов №1

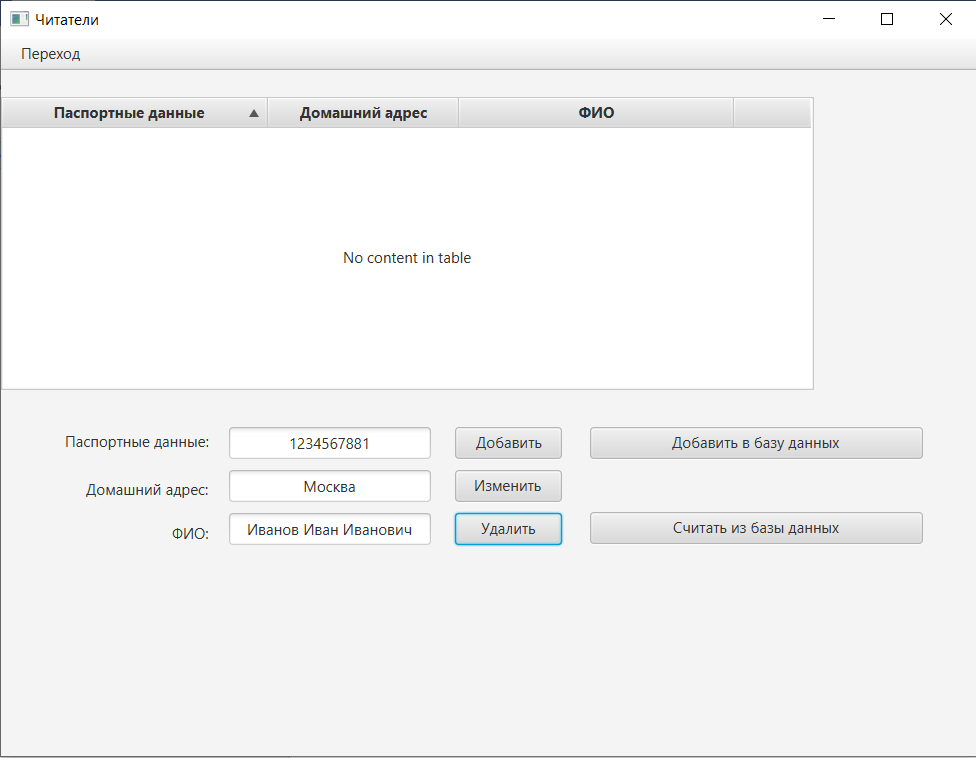


Рисунок 22 – Результат теста №2 из таблицы тестов №1

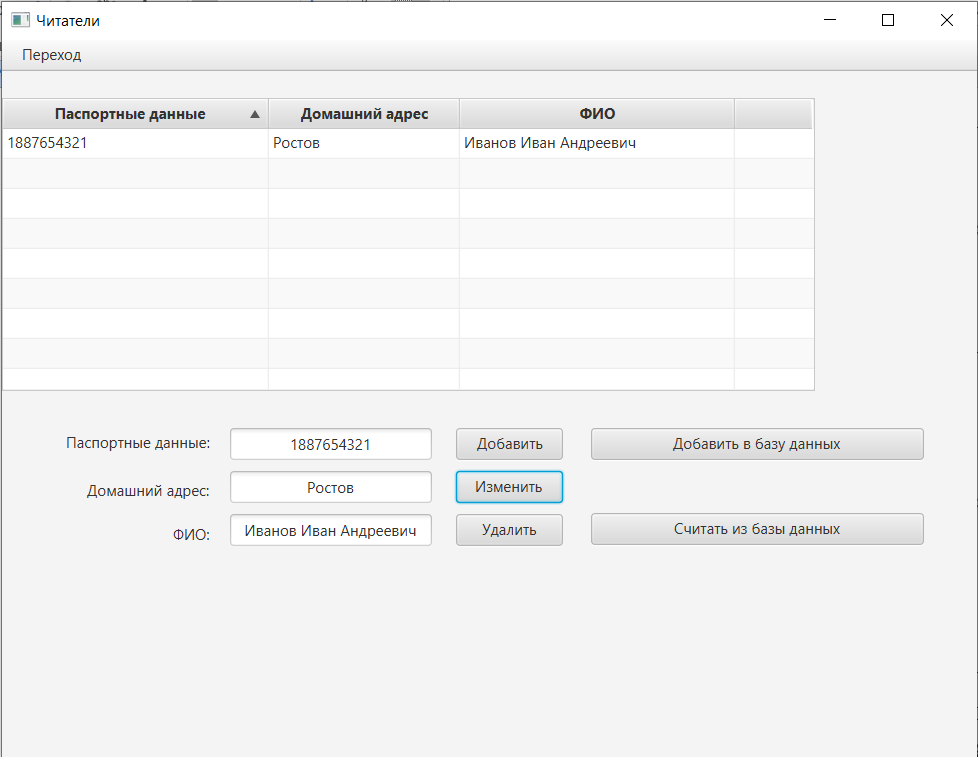


Рисунок 23 – Результат теста №3 из таблицы тестов №1 (сравните с рис. 21)

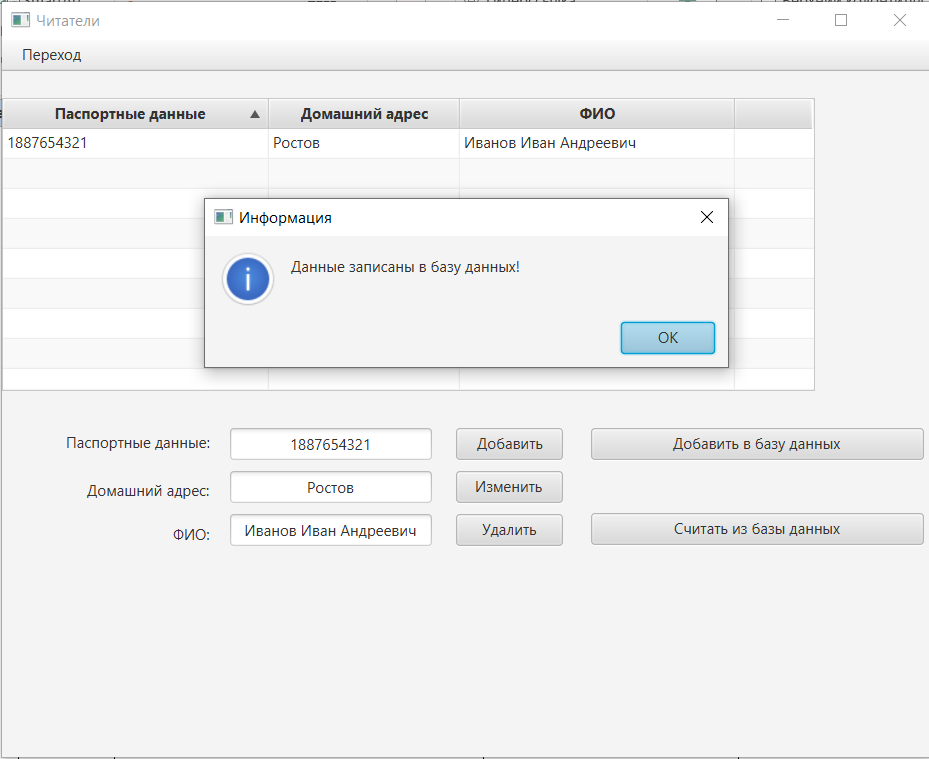


Рисунок 24 – Результат теста №4 из таблицы тестов №1

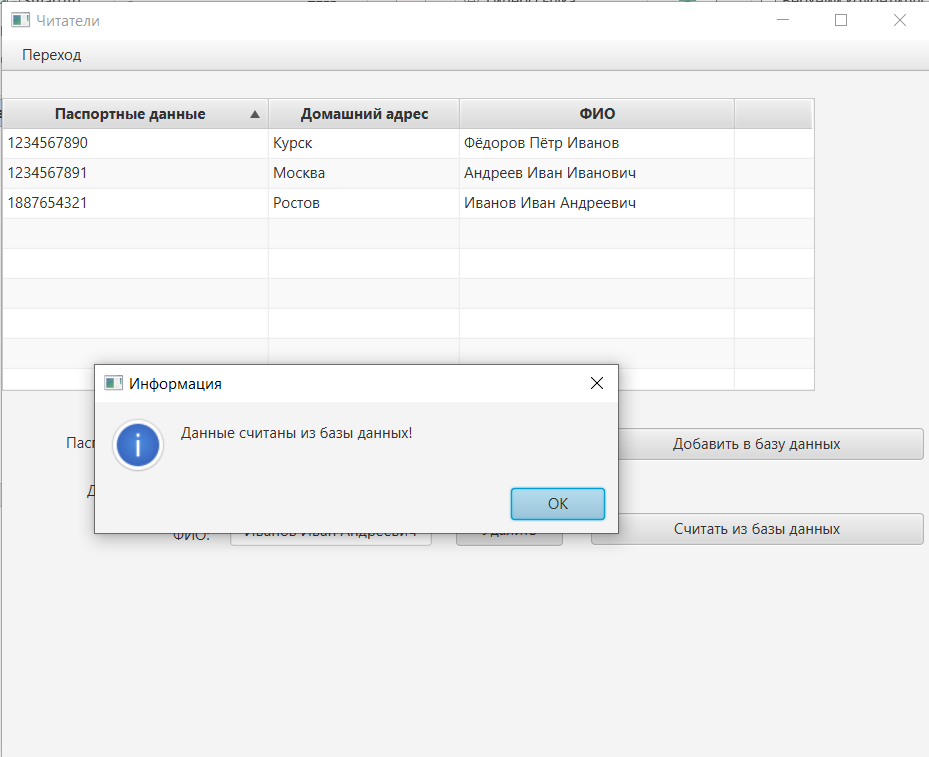


Рисунок 25 – Результат теста №5 из таблицы тестов №1

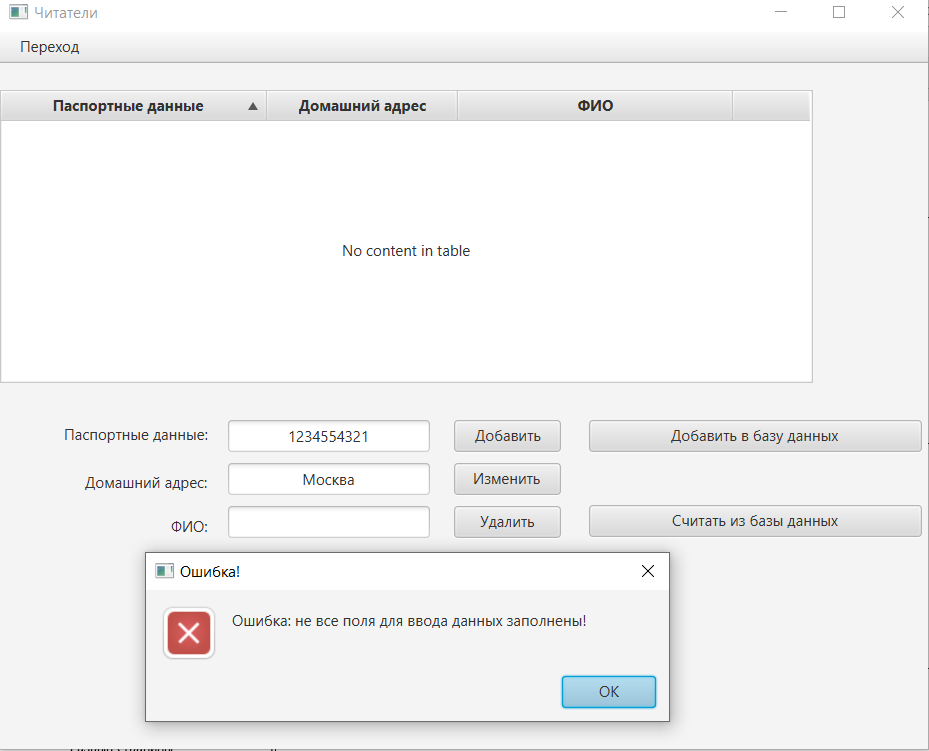


Рисунок 26 – Результат теста №6 из таблицы тестов №1

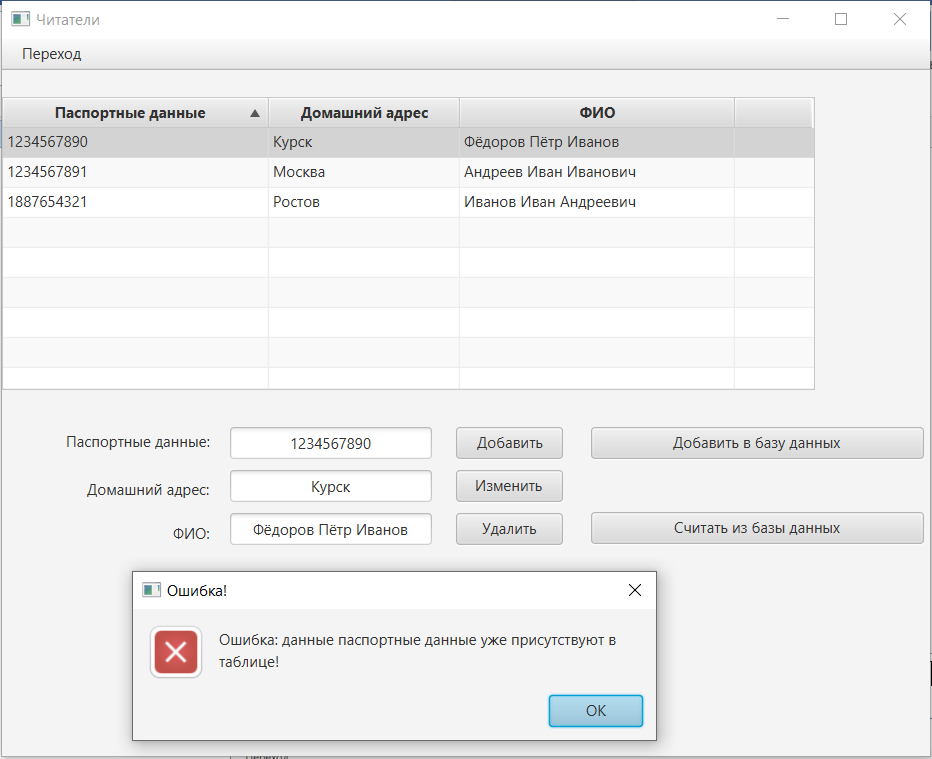


Рисунок 27 – Результат теста №7 из таблицы тестов №1

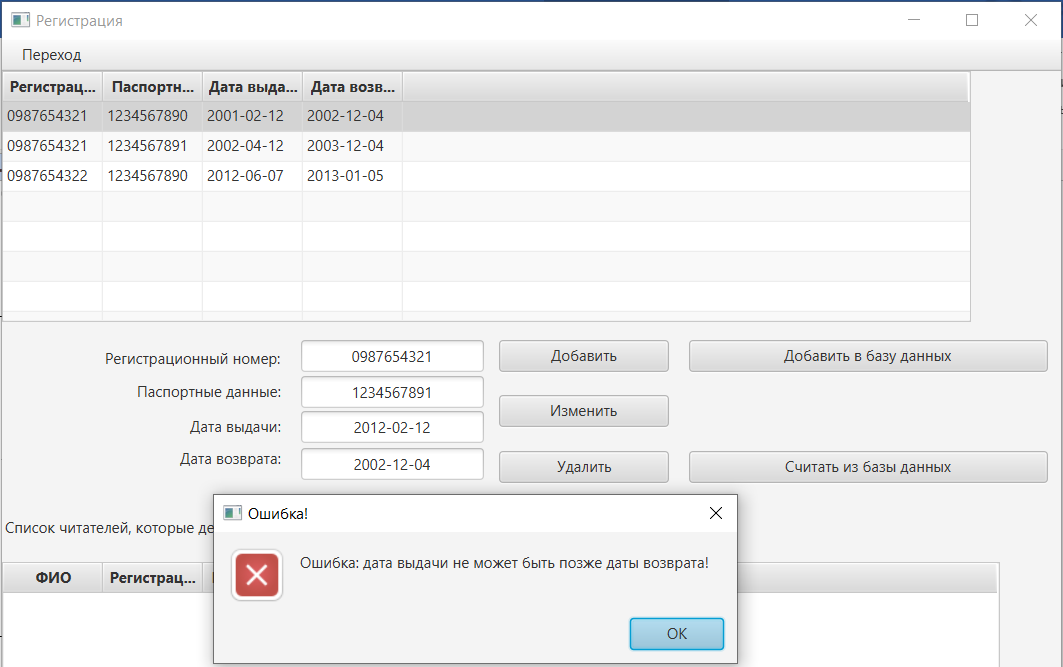


Рисунок 28 – Результат теста №8 из таблицы тестов №1

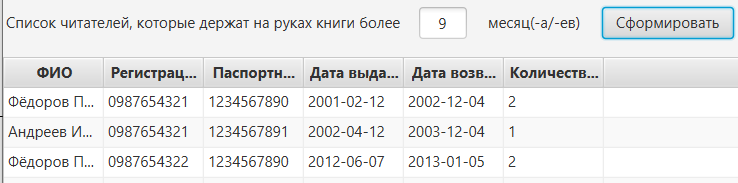


Рисунок 29 – Результат теста №9 из таблицы тестов №1

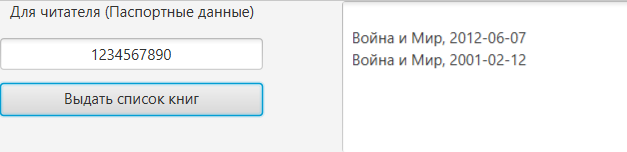


Рисунок 30 – Результат теста №10 из таблицы тестов №1

7 Листинг исходного кода

**Класс JDBCManager:**

package programGUI.database;

import javafx.scene.control.TableView;

import programGUI.BookMain;

import programGUI.ReaderMain;

import programGUI.RegisterMain;

import programGUI.data.BookData;

import programGUI.data.ReaderData;

import programGUI.data.RegisterData;

import java.sql.\*;

public class JDBCManager {

static final String DATABASE\_URL = "jdbc:mysql://localhost/datajava";

static final String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";

static final String USER = "root";

static final String PASSWORD = "10874<\_A\_+\_!MyS!@@$!^-%7@!pASsW@$!+-\*-@";

static final String sqlCreateBook = "CREATE TABLE Book (" +

"Register\_Number BIGINT PRIMARY KEY NOT NULL," +

"Name\_Book NVARCHAR(80) NULL," +

"Count\_Pages INT NULL," +

"Year\_Publishing INT NULL," +

"Section NVARCHAR(150) NULL" +

");";

static final String sqlCreateReader = "CREATE TABLE Reader (" +

"Password\_Data BIGINT PRIMARY KEY NOT NULL," +

"Home\_Address NVARCHAR(80) NULL," +

"Full\_Name NVARCHAR(50) NULL" +

");";

static final String sqlCreateRegister = "CREATE TABLE Register(" +

"ID INT PRIMARY KEY AUTO\_INCREMENT NOT NULL," +

"Register\_Number BIGINT NULL," +

"Password\_Data BIGINT NULL," +

"Date\_Issue DATE NULL," +

"Date\_Return DATE NULL," +

"FOREIGN KEY (Register\_Number) REFERENCES Book (Register\_Number)," +

"FOREIGN KEY (Password\_Data) REFERENCES Reader (Password\_Data));";

public static void CreateTable(String nameTable) throws ClassNotFoundException, SQLException {

Class.forName(JDBC\_DRIVER);

String sqlCommand = "";

if(nameTable.equals("Reader")){

sqlCommand = sqlCreateReader;

}else if(nameTable.equals("Book")){

sqlCommand = sqlCreateBook;

}else if(nameTable.equals("Register")){

sqlCommand = sqlCreateRegister;

}else{

return;

}

Statement stat = null;

Connection con = null;

try{

con = DriverManager.getConnection(DATABASE\_URL, USER, PASSWORD);

stat = con.createStatement();

stat.executeUpdate(sqlCommand);

} catch (Exception e) { } finally{

if(stat != null){

stat.close();

}

if(con != null){

con.close();

}

}

}

public static boolean WriteDataToDB\_Register(TableView<RegisterData> table) throws ClassNotFoundException {

if((table == null) || (table.getItems().size() == 0)){

return false;

}

Class.forName(JDBC\_DRIVER);

try{

try(Connection con = DriverManager.getConnection(DATABASE\_URL, USER, PASSWORD)){

for(int i = 0; i < table.getItems().size(); i++){

Statement stat = null;

try{

stat = con.createStatement();

stat.executeUpdate("INSERT INTO Register(Register\_Number, Password\_Data, Date\_Issue" +

", Date\_Return) VALUES ("

+ table.getItems().get(i).getRegister() + ", "

+ table.getItems().get(i).getPassword() + ", "

+ "DATE \'" + table.getItems().get(i).getDateIssue() + "\', "

+ "DATE \'" + table.getItems().get(i).getDateReturn() + "\'"

+ ");");

}catch (Exception e){}

finally {

if(stat != null)

stat.close();

}

}

} catch (SQLException e) {

return false;

}

}catch (Exception e){

return false;

}

return true;

}

public static TableView ReadDataFromDB\_Register(boolean del) throws ClassNotFoundException, SQLException {

Class.forName(JDBC\_DRIVER);

TableView<RegisterData> table = new TableView<>();

Connection con = null;

Statement stat = null;

ResultSet resultSet = null;

try{

con = DriverManager.getConnection(DATABASE\_URL, USER, PASSWORD);

stat = con.createStatement();

resultSet = stat.executeQuery("SELECT \* FROM Register;");

while(resultSet.next()){

String str = String.valueOf(resultSet.getLong(2)),

str1 = String.valueOf(resultSet.getLong(3));

while(str.length() != BookMain.MAX\_LENGTH\_REGNUM)

str = "0" + str;

while(str1.length() != ReaderMain.MAX\_LENGTH\_PASSWORD)

str1 = "0" + str1;

table.getItems().add(new RegisterData(

str,

str1,

resultSet.getDate(4).toString(),

resultSet.getDate(5).toString())

);

}

if(del){

Statement delStat = con.createStatement();

delStat.executeUpdate("TRUNCATE TABLE Register;");

delStat.close();

}

} catch (SQLException e) {

e.printStackTrace();

return null;

}

finally {

if(con != null)

con.close();

if(stat != null)

stat.close();

if(resultSet != null)

resultSet.close();

}

return table;

}

public static boolean WriteDataToDB\_Book(TableView<BookData> table) throws ClassNotFoundException {

if((table == null) || (table.getItems().size() == 0)){

return false;

}

Class.forName(JDBC\_DRIVER);

try{

try(Connection con = DriverManager.getConnection(DATABASE\_URL, USER, PASSWORD)){

for(int i = 0; i < table.getItems().size(); i++){

Statement stat = null;

try{

stat = con.createStatement();

stat.executeUpdate("INSERT INTO Book(Register\_Number, Name\_Book, Count\_Pages" +

", Year\_Publishing, Section) VALUES ("

+ table.getItems().get(i).getRegister() + ", \'"

+ table.getItems().get(i).getName() + "\', "

+ table.getItems().get(i).getPages() + ","

+ table.getItems().get(i).getYear() + ", \'"

+ table.getItems().get(i).getSection() + "\'" +

");");

}catch (Exception e){}

finally {

if(stat != null)

stat.close();

}

}

} catch (SQLException e) {

return false;

}

}catch (Exception e){

return false;

}

return true;

}

public static boolean WriteDataToDB\_Reader(TableView<ReaderData> table) throws ClassNotFoundException {

if((table == null) || (table.getItems().size() == 0)){

return false;

}

Class.forName(JDBC\_DRIVER);

try{

try(Connection con = DriverManager.getConnection(DATABASE\_URL, USER, PASSWORD)){

for(int i = 0; i < table.getItems().size(); i++){

Statement stat = null;

try{

stat = con.createStatement();

stat.executeUpdate("INSERT INTO Reader(Password\_Data, Home\_Address, Full\_Name) VALUES ("

+ table.getItems().get(i).getPassword() + ", \'"

+ table.getItems().get(i).getAddress() + "\', \'"

+ table.getItems().get(i).getFio() + "\');");

}catch (Exception e){}

finally {

if(stat != null)

stat.close();

}

}

} catch (SQLException e) {

return false;

}

}catch (Exception e){

return false;

}

return true;

}

public static TableView ReadDataFromDB\_Book(boolean del) throws SQLException, ClassNotFoundException {

Class.forName(JDBC\_DRIVER);

TableView<BookData> table = new TableView<>();

Connection con = null;

Statement stat = null;

ResultSet resultSet = null;

try{

con = DriverManager.getConnection(DATABASE\_URL, USER, PASSWORD);

stat = con.createStatement();

resultSet = stat.executeQuery("SELECT \* FROM Book;");

while(resultSet.next()){

String str = String.valueOf(resultSet.getLong(1));

while(str.length() != BookMain.MAX\_LENGTH\_REGNUM)

str = "0" + str;

table.getItems().add(new BookData(

str,

resultSet.getString(2),

resultSet.getInt(3),

resultSet.getInt(4),

resultSet.getString(5))

);

}

if(del){

Statement delStat = con.createStatement();

delStat.executeUpdate("TRUNCATE TABLE Book;");

delStat.close();

}

} catch (SQLException e) {

e.printStackTrace();

return null;

}

finally {

if(con != null)

con.close();

if(stat != null)

stat.close();

if(resultSet != null)

resultSet.close();

}

return table;

}

public static TableView ReadDataFromDB\_Reader(boolean del) throws SQLException, ClassNotFoundException {

Class.forName(JDBC\_DRIVER);

TableView<ReaderData> table = new TableView<>();

Connection con = null;

Statement stat = null;

ResultSet resultSet = null;

try{

con = DriverManager.getConnection(DATABASE\_URL, USER, PASSWORD);

stat = con.createStatement();

resultSet = stat.executeQuery("SELECT \* FROM Reader;");

while(resultSet.next()){

String str =String.valueOf(resultSet.getLong(1));

while(str.length() != ReaderMain.MAX\_LENGTH\_PASSWORD)

str = "0" + str;

table.getItems().add(new ReaderData(

str,

resultSet.getString(2),

resultSet.getString(3)

));

}

if(del){

Statement delStat = con.createStatement();

delStat.executeUpdate("TRUNCATE TABLE Reader;");

delStat.close();

}

} catch (SQLException e) {

e.printStackTrace();

return null;

}

finally {

if(con != null)

con.close();

if(stat != null)

stat.close();

if(resultSet != null)

resultSet.close();

}

return table;

}

}

**Класс BookMain:**

package programGUI;

import javafx.beans.value.ChangeListener;

import javafx.beans.value.ObservableValue;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.input.MouseButton;

import javafx.scene.input.MouseEvent;

import javafx.stage.Stage;

import programGUI.data.BookData;

import programGUI.data.ReaderData;

import programGUI.data.RegisterData;

import programGUI.database.JDBCManager;

import java.io.IOException;

import java.sql.SQLException;

import java.util.ArrayList;

public class BookMain {

public static final int MAX\_LENGTH\_REGNUM = 10;

private MenuBar \_menuBar = null;

private TextField \_regNumber = null;

private TextField \_nameBook = null;

private TextField \_countPages = null;

private TextField \_yearPublishing = null;

private TextField \_section = null;

private Button \_addData = null;

private Button \_refactorData = null;

private Button \_deleteData = null;

private Button \_inputDB = null;

private Button \_outputDB = null;

private TableView<BookData> \_table = null;

private Stage thisStage = null;

public static Stage readerStage = null;

public static Stage registerStage = null;

public void Show(){

if(this.thisStage == null)

return;

this.thisStage.show();

}

public void Hide(){

if(this.thisStage == null)

return;

this.thisStage.hide();

}

public Stage GetStage(){

return this.thisStage;

}

private boolean checkInputData(boolean reg){

ArrayList<TextField> fields = new ArrayList<TextField>();

fields.add(\_regNumber);

fields.add(\_nameBook);

fields.add(\_countPages);

fields.add(\_yearPublishing);

fields.add(\_section);

if(!ReaderMain.checkTextFields(fields)){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: не все поля для ввода данных заполнены!");

return false;

}else if(!ReaderMain.passwordDataValidate(\_regNumber.getText().toString(), MAX\_LENGTH\_REGNUM)){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: не корректно введены регистрационные данные!");

return false;

}else if((reg) && (isPasswordExist(fields.get(0).getText().toString()))){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: данные регистрационные данные уже присутствуют в таблице!");

return false;

}

return true;

}

private boolean isPasswordExist(String password){

if((\_table == null) || (\_table.getItems().size() == 0))

return false;

for(int i = 0; i < \_table.getItems().size(); i++){

if(\_table.getItems().get(i).getRegister().equals(password))

return true;

}

return false;

}

public BookMain() throws IOException, SQLException, ClassNotFoundException {

JDBCManager.CreateTable("Book");

Parent root = FXMLLoader.load(getClass().getResource("markup/viewBook.fxml"));

Scene scene = new Scene(root);

thisStage = new Stage();

thisStage.setScene(scene);

thisStage.setTitle("Книги");

\_menuBar = (MenuBar) scene.lookup("#\_menuBar");

\_addData = (Button) scene.lookup("#\_btnAdd");

\_refactorData = (Button) scene.lookup("#\_btnRefactor");

\_deleteData = (Button) scene.lookup("#\_btnDelete");

\_regNumber = (TextField) scene.lookup("#\_txtRegNumber");

\_nameBook = (TextField) scene.lookup("#\_txtNameBook");

\_yearPublishing = (TextField) scene.lookup("#\_txtYearPublishing");

\_countPages = (TextField) scene.lookup("#\_txtCountPages");

\_section = (TextField) scene.lookup("#\_txtSection");

\_inputDB = (Button) scene.lookup("#\_btnInputDB");

\_outputDB = (Button) scene.lookup("#\_btnOutputDB");

\_table = (TableView) scene.lookup("#\_table");

\_table.getColumns().clear();

\_menuBar.getMenus().clear();

Menu menu = new Menu("Переход");

MenuItem reader = new MenuItem("Читатели");

MenuItem register = new MenuItem("Регистрация");

reader.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

thisStage.hide();

readerStage.show();

}

});

register.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

thisStage.hide();

registerStage.show();

}

});

menu.getItems().add(reader);

menu.getItems().add(register);

\_menuBar.getMenus().add(0, menu);

TableColumn<BookData, String> attrib = new TableColumn<BookData, String>("Регистрационный номер");

attrib.setCellValueFactory(new PropertyValueFactory<BookData, String>("register"));

\_table.getColumns().add(attrib);

attrib = new TableColumn<BookData, String>("Название книги");

attrib.setCellValueFactory(new PropertyValueFactory<BookData, String>("name"));

\_table.getColumns().add(attrib);

TableColumn<BookData, Integer> attrib1 = new TableColumn<BookData, Integer>("Количество страниц");

attrib1.setCellValueFactory(new PropertyValueFactory<BookData, Integer>("pages"));

\_table.getColumns().add(attrib1);

attrib1 = new TableColumn<BookData, Integer>("Год публикации");

attrib1.setCellValueFactory(new PropertyValueFactory<BookData, Integer>("year"));

\_table.getColumns().add(attrib1);

attrib = new TableColumn<BookData, String>("Раздел");

attrib.setCellValueFactory(new PropertyValueFactory<BookData, String>("section"));

\_table.getColumns().add(attrib);

\_regNumber.textProperty().addListener(new ChangeListener<String>() {

@Override

public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

if((oldValue.length() == MAX\_LENGTH\_REGNUM)

&& (newValue.length() > MAX\_LENGTH\_REGNUM)){

\_regNumber.setText(oldValue);

return;

}

if ((!newValue.matches("\\d\*")) ) {

\_regNumber.setText(newValue.replaceAll("[^\\d]", ""));

}

}

});

\_yearPublishing.textProperty().addListener(new ChangeListener<String>() {

@Override

public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

if ((!newValue.matches("\\d\*")) ) {

\_yearPublishing.setText(newValue.replaceAll("[^\\d]", ""));

}

}

});

\_countPages.textProperty().addListener(new ChangeListener<String>() {

@Override

public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

if ((!newValue.matches("\\d\*")) ) {

\_countPages.setText(newValue.replaceAll("[^\\d]", ""));

}

}

});

\_addData.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if(!checkInputData(true))

return;

\_table.getItems().add(new BookData(\_regNumber.getText().toString(),

\_nameBook.getText().toString(),

Integer.valueOf(\_countPages.getText().toString()),

Integer.valueOf(\_yearPublishing.getText().toString()),

\_section.getText().toString()));

}

});

\_deleteData.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

try{

int row = \_table.getSelectionModel().getSelectedIndex();

\_table.getItems().remove(row);

}catch (Exception e){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: строка таблицы для удаления не выделена!");

}

}

});

\_refactorData.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if(!checkInputData(false))

return;

try{

int row = \_table.getSelectionModel().getSelectedIndex();

\_table.getItems().set(row, new BookData(\_regNumber.getText().toString(),

\_nameBook.getText().toString(),

Integer.valueOf(\_countPages.getText().toString()),

Integer.valueOf(\_yearPublishing.getText().toString()),

\_section.getText().toString()));

}catch (Exception e){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: строка таблицы для изменения не выделена!");

}

}

});

\_inputDB.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if((\_table == null) || (\_table.getItems().size() == 0)){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: нет данных для записи!");

return;

}

try {

JDBCManager.WriteDataToDB\_Book(\_table);

ReaderMain.MessageShow(Alert.AlertType.INFORMATION, "Информация", "Данные записаны в базу данных!");

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

}

});

\_outputDB.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

try {

TableView<BookData> book = JDBCManager.ReadDataFromDB\_Book(false);

\_table.getItems().clear();

for(int i = 0; i < book.getItems().size(); i++)

\_table.getItems().add(book.getItems().get(i));

} catch (Exception e) {}

ReaderMain.MessageShow(Alert.AlertType.INFORMATION, "Информация", "Данные считаны из базы данных!");

}

});

\_table.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if((event.getButton() == MouseButton.SECONDARY) && (\_table.getItems().size() > 0)){

int row = \_table.getSelectionModel().getSelectedIndex();

BookData data = \_table.getItems().get(row);

\_regNumber.setText(data.getRegister());

\_nameBook.setText(data.getName());

\_countPages.setText(String.valueOf(data.getPages()));

\_yearPublishing.setText(String.valueOf(data.getYear()));

\_section.setText(data.getSection());

}

}

});

}

}

**Класс ReaderMain:**

package programGUI;

import javafx.application.Application;

import javafx.beans.value.ChangeListener;

import javafx.beans.value.ObservableValue;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXMLLoader;

import javafx.scene.AccessibleRole;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.input.MouseButton;

import javafx.scene.input.MouseEvent;

import javafx.stage.Stage;

import programGUI.data.BookData;

import programGUI.data.ReaderData;

import programGUI.data.RegisterData;

import programGUI.database.JDBCManager;

import java.awt.print.Book;

import java.io.IOException;

import java.util.ArrayList;

public class ReaderMain extends Application {

public static final int MAX\_LENGTH\_PASSWORD = 10;

private TableView<ReaderData> \_table = null;

private TextField \_passwordData = null;

private TextField \_homeAddress = null;

private TextField \_fio = null;

private Button \_addData = null;

private Button \_refactor = null;

private Button \_delete = null;

private Button \_inputDB = null;

private Button \_outputDB = null;

private MenuBar \_menuBar = null;

private BookMain \_bookMain = null;

private RegisterMain \_registerMain = null;

private Stage thisStage = null;

public static boolean checkTextFields(ArrayList<TextField> fields){

if(fields.size() == 0)

return false;

for(int i = 0; i < fields.size(); i++)

if(fields.get(i).getText().toString().length() == 0)

return false;

return true;

}

public static boolean passwordDataValidate(String password, int maxLength){

if(password.length() != maxLength)

return false;

for(int i = 0; i < password.length(); i++)

if(!Character.isDigit(password.charAt(i)))

return false;

return true;

}

public static void MessageShow(Alert.AlertType type, String title, String message){

Alert alert = new Alert(type);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(message);

alert.show();

}

private boolean checkInputData(boolean pwd){

ArrayList<TextField> fields = new ArrayList<TextField>();

fields.add(\_passwordData);

fields.add(\_homeAddress);

fields.add(\_fio);

if(!checkTextFields(fields)){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: не все поля для ввода данных заполнены!");

return false;

}else if(!passwordDataValidate(\_passwordData.getText().toString(), MAX\_LENGTH\_PASSWORD)){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: не корректно введены паспортные данные!");

return false;

}else if((pwd) && (isPasswordExist(fields.get(0).getText().toString()))){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: данные паспортные данные уже присутствуют в таблице!");

return false;

}

return true;

}

private boolean isPasswordExist(String password){

if((\_table == null) || (\_table.getItems().size() == 0))

return false;

for(int i = 0; i < \_table.getItems().size(); i++){

if(\_table.getItems().get(i).getPassword().equals(password))

return true;

}

return false;

}

@Override

public void start(Stage primaryStage) throws Exception{

JDBCManager.CreateTable("Reader");

Parent root = FXMLLoader.load(getClass().getResource("markup/viewReader.fxml"));

Scene scene = new Scene(root);

primaryStage.setScene(scene);

primaryStage.setTitle("Читатели");

//Определение взаимосвязей

this.thisStage = primaryStage;

\_bookMain = new BookMain();

\_registerMain = new RegisterMain();

BookMain.readerStage = thisStage;

BookMain.registerStage = \_registerMain.GetStage();

RegisterMain.readerStage = thisStage;

RegisterMain.bookStage = \_bookMain.GetStage();

\_passwordData = (TextField) scene.lookup("#\_txtPasswordData");

\_homeAddress = (TextField) scene.lookup("#\_txtHomeAddress");

\_fio = (TextField) scene.lookup("#\_txtFio");

\_addData = (Button) scene.lookup("#\_btnAddData");

\_refactor = (Button) scene.lookup("#\_btnRefactor");

\_delete = (Button) scene.lookup("#\_btnDelete");

\_inputDB = (Button) scene.lookup("#\_btnInputDB");

\_outputDB = (Button) scene.lookup("#\_btnOutputDB");

\_menuBar = (MenuBar) scene.lookup("#\_menuBar");

\_table = (TableView) scene.lookup("#\_tableReader");

\_table.getColumns().clear();

\_menuBar.getMenus().clear();

Menu menu = (new Menu("Переход"));

MenuItem item = new MenuItem("Книги");

MenuItem reg = new MenuItem("Регистрация");

item.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

thisStage.hide();

\_bookMain.Show();

}

});

reg.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

thisStage.hide();

\_registerMain.Show();

}

});

menu.getItems().add(item);

menu.getItems().add(reg);

\_menuBar.getMenus().add(0, menu);

TableColumn<ReaderData, String> attrib = new TableColumn<ReaderData, String>("Паспортные данные");

attrib.setCellValueFactory(new PropertyValueFactory<ReaderData, String>("password"));

\_table.getColumns().add(attrib);

attrib = new TableColumn<ReaderData, String>("Домашний адрес");

attrib.setCellValueFactory(new PropertyValueFactory<ReaderData, String>("address"));

\_table.getColumns().add(attrib);

attrib = new TableColumn<ReaderData, String>("ФИО");

attrib.setCellValueFactory(new PropertyValueFactory<ReaderData, String>("fio"));

\_table.getColumns().add(attrib);

\_passwordData.textProperty().addListener(new ChangeListener<String>() {

@Override

public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

if((oldValue.length() == MAX\_LENGTH\_PASSWORD)

&& (newValue.length() > MAX\_LENGTH\_PASSWORD)){

\_passwordData.setText(oldValue);

return;

}

if ((!newValue.matches("\\d\*")) ) {

\_passwordData.setText(newValue.replaceAll("[^\\d]", ""));

}

}

});

\_delete.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

try{

int row = \_table.getSelectionModel().getSelectedIndex();

\_table.getItems().remove(row);

}catch (Exception e){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: строка таблицы для удаления не выделена!");

}

}

});

\_refactor.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if(!checkInputData(false))

return;

try{

int row = \_table.getSelectionModel().getSelectedIndex();

\_table.getItems().set(row, new ReaderData(

\_passwordData.getText().toString(),

\_homeAddress.getText().toString(),

\_fio.getText().toString()));

}catch (Exception e){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: строка таблицы для изменения не выделена!");

}

}

});

\_addData.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if(!checkInputData(true))

return;

\_table.getItems().add(new ReaderData(\_passwordData.getText().toString(),

\_homeAddress.getText().toString(),

\_fio.getText().toString()));

}

});

\_inputDB.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if((\_table == null) || (\_table.getItems().size() == 0)){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: нет данных для записи!");

return;

}

try {

JDBCManager.WriteDataToDB\_Reader(\_table);

MessageShow(Alert.AlertType.INFORMATION, "Информация", "Данные записаны в базу данных!");

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

}

});

\_outputDB.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

try {

TableView<ReaderData> reader = JDBCManager.ReadDataFromDB\_Reader(false);

\_table.getItems().clear();

for(int i = 0; i < reader.getItems().size(); i++)

\_table.getItems().add(reader.getItems().get(i));

} catch (Exception e) {}

MessageShow(Alert.AlertType.INFORMATION, "Информация", "Данные считаны из базы данных!");

}

});

\_table.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if((event.getButton() == MouseButton.SECONDARY) && (\_table.getItems().size() > 0)){

int row = \_table.getSelectionModel().getSelectedIndex();

ReaderData data = \_table.getItems().get(row);

\_passwordData.setText(data.getPassword());

\_homeAddress.setText(data.getAddress());

\_fio.setText(data.getFio());

}

}

});

primaryStage.setScene(scene);

primaryStage.show();

}

public static void main(String[] args) {

launch(args);

}

}

**Класс RegisterMain:**

package programGUI;

import com.sun.javafx.image.IntPixelGetter;

import javafx.beans.value.ChangeListener;

import javafx.beans.value.ObservableValue;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.input.MouseButton;

import javafx.scene.input.MouseEvent;

import javafx.stage.Stage;

import javafx.util.converter.DateTimeStringConverter;

import org.w3c.dom.Text;

import programGUI.data.BookData;

import programGUI.data.InfoReaderData;

import programGUI.data.ReaderData;

import programGUI.data.RegisterData;

import programGUI.database.JDBCManager;

import java.io.IOException;

import java.sql.SQLException;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Comparator;

import java.util.Date;

public class RegisterMain {

private MenuBar \_menuBar = null;

private TextField \_password = null;

private TextField \_register = null;

private TextField \_dateIssue = null;

private TextField \_dateReturn = null;

private Button \_addData = null;

private Button \_refactorData = null;

private Button \_deleteData = null;

private Button \_inputDB = null;

private Button \_outputDB = null;

private TableView<RegisterData> \_table = null;

private TableView<InfoReaderData> \_tableInfo = null;

private TextField \_month = null;

private Button \_execute = null;

private Button \_execute2 = null;

private TextField \_fio = null;

private TextArea \_resultExecute2 = null;

private Stage thisStage = null;

public static Stage readerStage = null;

public static Stage bookStage = null;

public void Show(){

if(this.thisStage == null)

return;

this.thisStage.show();

}

public void Hide(){

if(this.thisStage == null)

return;

this.thisStage.hide();

}

public Stage GetStage(){

return this.thisStage;

}

private boolean checkInputData() throws ParseException {

ArrayList<TextField> fields = new ArrayList<TextField>();

fields.add(\_register);

fields.add(\_password);

fields.add(\_dateIssue);

fields.add(\_dateReturn);

if(!ReaderMain.checkTextFields(fields)){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: не все поля для ввода данных заполнены!");

return false;

}else if(!ReaderMain.passwordDataValidate(\_register.getText().toString(), BookMain.MAX\_LENGTH\_REGNUM)){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: не корректно введены регистрационные данные!");

return false;

}else if(!ReaderMain.passwordDataValidate(\_password.getText().toString(), ReaderMain.MAX\_LENGTH\_PASSWORD)){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: не корректно введены паспортные данные!");

return false;

}else if((!isDateValidate(\_dateIssue.getText().toString())) ||

(!isDateValidate(\_dateReturn.getText().toString()))){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: дата должна быть в формате ГГГГ-ММ-ДД!");

return false;

}

DateFormat format = new SimpleDateFormat("yyyy-mm-dd");

if(format.parse(\_dateIssue.getText().toString()).after(

format.parse(\_dateReturn.getText().toString())

)){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: дата выдачи не может быть позже даты возврата!");

return false;

}

return true;

}

public static boolean isDateValidate(String date){

if((date == null) || (date.length() == 0))

return false;

int count = 0;

for(int i = 0; i < date.length(); i++){

if(date.charAt(i) == '-'){

count++;

}else if(!Character.isDigit(date.charAt(i))){

return false;

}

}

if(count != 2)

return false;

String[] dateSplit = date.split("\\-");

if((dateSplit.length != 3) || (dateSplit[0].length() != 4)

|| (dateSplit[1].length() != 2) || (dateSplit[2].length() != 2))

return false;

int value = Integer.valueOf(dateSplit[2]);

if((value <= 0) || (value > 31))

return false;

value = Integer.valueOf(dateSplit[1]);

if((value <= 0) || (value > 12))

return false;

return true;

}

private int CountBook(String passwordData) throws SQLException, ClassNotFoundException {

TableView<RegisterData> registerData = JDBCManager.ReadDataFromDB\_Register(false);

if((registerData == null) || (registerData.getItems().size() <= 0))

return 0;

int count = 0;

for(int i = 0; i < registerData.getItems().size(); i++)

if(registerData.getItems().get(i).getPassword().equals(passwordData))

count++;

return count;

}

private BookData GetBookByPasswordData(String passwordData) throws SQLException, ClassNotFoundException {

TableView<BookData> books = JDBCManager.ReadDataFromDB\_Book(false);

TableView<RegisterData> registers = JDBCManager.ReadDataFromDB\_Register(false);

if((books == null) || (registers == null)

|| (books.getItems().size() == 0) || (registers.getItems().size() == 0)){

return null;

}

for(int i = 0; i < registers.getItems().size(); i++){

if(registers.getItems().get(i).getPassword().equals(passwordData)){

for(int j = 0; j < books.getItems().size(); j++){

if(registers.getItems().get(i).getRegister().equals(

books.getItems().get(j).getRegister()

)){

return books.getItems().get(j);

}

}

}

}

return null;

}

public RegisterMain() throws IOException, SQLException, ClassNotFoundException {

JDBCManager.CreateTable("Register");

Parent root = FXMLLoader.load(getClass().getResource("markup/viewRegister.fxml"));

Scene scene = new Scene(root);

thisStage = new Stage();

thisStage.setScene(scene);

thisStage.setTitle("Регистрация");

\_menuBar = (MenuBar) scene.lookup("#\_menuBar");

\_addData = (Button) scene.lookup("#\_btnAdd");

\_refactorData = (Button) scene.lookup("#\_btnRefactor");

\_deleteData = (Button) scene.lookup("#\_btnDelete");

\_register = (TextField) scene.lookup("#\_txtRegNumber");

\_inputDB = (Button) scene.lookup("#\_btnInputDB");

\_outputDB = (Button) scene.lookup("#\_btnOutputDB");

\_password = (TextField) scene.lookup("#\_txtPassword");

\_dateIssue = (TextField) scene.lookup("#\_txtDateIssue");

\_dateReturn = (TextField) scene.lookup("#\_txtDateReturn");

\_table = (TableView) scene.lookup("#\_table");

\_tableInfo = (TableView) scene.lookup("#\_table2");

\_execute = (Button) scene.lookup("#\_btnExecute1");

\_month = (TextField) scene.lookup("#\_txtMonth");

\_execute2 = (Button) scene.lookup("#\_btnExecute2");

\_fio = (TextField) scene.lookup("#\_txtFio");

\_resultExecute2 = (TextArea) scene.lookup("#\_txtBooks");

\_table.getColumns().clear();

\_tableInfo.getColumns().clear();

\_menuBar.getMenus().clear();

Menu menu = new Menu("Переход");

MenuItem reader = new MenuItem("Читатели");

MenuItem book = new MenuItem("Книги");

reader.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

thisStage.hide();

readerStage.show();

}

});

book.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

thisStage.hide();

bookStage.show();

}

});

menu.getItems().add(reader);

menu.getItems().add(book);

\_menuBar.getMenus().add(0, menu);

TableColumn<RegisterData, String> attrib = new TableColumn<RegisterData, String>("Регистрационный номер");

attrib.setCellValueFactory(new PropertyValueFactory<RegisterData, String>("register"));

\_table.getColumns().add(attrib);

attrib = new TableColumn<RegisterData, String>("Паспортные данные");

attrib.setCellValueFactory(new PropertyValueFactory<RegisterData, String>("password"));

\_table.getColumns().add(attrib);

attrib = new TableColumn<RegisterData, String>("Дата выдачи");

attrib.setCellValueFactory(new PropertyValueFactory<RegisterData, String>("dateIssue"));

\_table.getColumns().add(attrib);

attrib = new TableColumn<RegisterData, String>("Дата возврата");

attrib.setCellValueFactory(new PropertyValueFactory<RegisterData, String>("dateReturn"));

\_table.getColumns().add(attrib);

TableColumn<InfoReaderData, String> attribInfo = new TableColumn<InfoReaderData, String>("ФИО");

attribInfo.setCellValueFactory(new PropertyValueFactory<InfoReaderData, String>("fio"));

\_tableInfo.getColumns().add(attribInfo);

attribInfo = new TableColumn<InfoReaderData, String>("Регистрационный номер");

attribInfo.setCellValueFactory(new PropertyValueFactory<InfoReaderData, String>("register"));

\_tableInfo.getColumns().add(attribInfo);

attribInfo = new TableColumn<InfoReaderData, String>("Паспортные данные");

attribInfo.setCellValueFactory(new PropertyValueFactory<InfoReaderData, String>("password"));

\_tableInfo.getColumns().add(attribInfo);

attribInfo = new TableColumn<InfoReaderData, String>("Дата выдачи");

attribInfo.setCellValueFactory(new PropertyValueFactory<InfoReaderData, String>("dateIssue"));

\_tableInfo.getColumns().add(attribInfo);

attribInfo = new TableColumn<InfoReaderData, String>("Дата возврата");

attribInfo.setCellValueFactory(new PropertyValueFactory<InfoReaderData, String>("dateReturn"));

\_tableInfo.getColumns().add(attribInfo);

TableColumn<InfoReaderData, Integer> attribInfoCount = new TableColumn<>("Количество взятых книг");

attribInfoCount.setCellValueFactory(new PropertyValueFactory<InfoReaderData, Integer>("countBook"));

\_tableInfo.getColumns().add(attribInfoCount);

\_month.textProperty().addListener(new ChangeListener<String>() {

@Override

public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

if ((!newValue.matches("\\d\*")) ) {

\_month.setText(newValue.replaceAll("[^\\d]", ""));

}

}

});

\_fio.textProperty().addListener(new ChangeListener<String>() {

@Override

public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

if((oldValue.length() == ReaderMain.MAX\_LENGTH\_PASSWORD)

&& (newValue.length() > ReaderMain.MAX\_LENGTH\_PASSWORD)){

\_fio.setText(oldValue);

return;

}

if ((!newValue.matches("\\d\*")) ) {

\_fio.setText(newValue.replaceAll("[^\\d]", ""));

}

}

});

\_execute.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

\_tableInfo.getItems().clear();

TableView<RegisterData> registerData = null;

try {

registerData = JDBCManager.ReadDataFromDB\_Register(false);

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException throwables) {

throwables.printStackTrace();

}

TableView<ReaderData> readerData = null;

try {

readerData = JDBCManager.ReadDataFromDB\_Reader(false);

} catch (SQLException throwables) {

throwables.printStackTrace();

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

if(readerData.getItems().size() <= 0){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: нет зарегистрированных данных о взятии и возврате книг!");

return;

}

for(int i = 0; i < registerData.getItems().size(); i++){

if(((Integer.valueOf(registerData.getItems().get(i).getDateReturn().split("\\-")[1])

- Integer.valueOf(registerData.getItems().get(i).getDateIssue().split("\\-")[1])) >

Integer.valueOf(\_month.getText())) ||

((Integer.valueOf(registerData.getItems().get(i).getDateReturn().split("\\-")[0])

- Integer.valueOf(registerData.getItems().get(i).getDateIssue().split("\\-")[0])) >

(Integer.valueOf(\_month.getText()) / 12))){

int index = (-1);

for(int j = 0; j < readerData.getItems().size(); j++){

if(readerData.getItems().get(j).getPassword().equals(

registerData.getItems().get(i).getPassword()

)){

index = j;

break;

}

}

if(index >= 0){

try {

\_tableInfo.getItems().add(new InfoReaderData(

readerData.getItems().get(index).getFio(),

registerData.getItems().get(i).getRegister(),

registerData.getItems().get(i).getPassword(),

registerData.getItems().get(i).getDateIssue(),

registerData.getItems().get(i).getDateReturn(),

CountBook(registerData.getItems().get(i).getPassword())

));

} catch (SQLException throwables) {

throwables.printStackTrace();

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

}

}

}

}

});

\_register.textProperty().addListener(new ChangeListener<String>() {

@Override

public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

if((oldValue.length() == BookMain.MAX\_LENGTH\_REGNUM)

&& (newValue.length() > BookMain.MAX\_LENGTH\_REGNUM)){

\_register.setText(oldValue);

return;

}

if ((!newValue.matches("\\d\*")) ) {

\_register.setText(newValue.replaceAll("[^\\d]", ""));

}

}

});

\_password.textProperty().addListener(new ChangeListener<String>() {

@Override

public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

if((oldValue.length() == ReaderMain.MAX\_LENGTH\_PASSWORD)

&& (newValue.length() > ReaderMain.MAX\_LENGTH\_PASSWORD)){

\_password.setText(oldValue);

return;

}

if ((!newValue.matches("\\d\*")) ) {

\_password.setText(newValue.replaceAll("[^\\d]", ""));

}

}

});

\_addData.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

try {

if(!checkInputData())

return;

} catch (ParseException e) {

e.printStackTrace();

}

\_table.getItems().add(new RegisterData(\_register.getText().toString(),

\_password.getText().toString(),

\_dateIssue.getText().toString(),

\_dateReturn.getText().toString()));

}

});

\_deleteData.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

try{

int row = \_table.getSelectionModel().getSelectedIndex();

\_table.getItems().remove(row);

}catch (Exception e){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: строка таблицы для удаления не выделена!");

}

}

});

\_table.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if((event.getButton() == MouseButton.SECONDARY) && (\_table.getItems().size() > 0)){

int row = \_table.getSelectionModel().getSelectedIndex();

RegisterData data = \_table.getItems().get(row);

\_password.setText(data.getPassword());

\_register.setText(data.getRegister());

\_dateIssue.setText(data.getDateIssue());

\_dateReturn.setText(data.getDateReturn());

}

}

});

\_refactorData.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

try {

if(!checkInputData())

return;

} catch (ParseException e) {

e.printStackTrace();

}

try{

int row = \_table.getSelectionModel().getSelectedIndex();

\_table.getItems().set(row, new RegisterData(\_register.getText().toString(),

\_password.getText().toString(),

\_dateIssue.getText().toString(),

\_dateReturn.getText().toString()));

}catch (Exception e){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: строка таблицы для изменения не выделена!");

}

}

});

\_inputDB.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if((\_table == null) || (\_table.getItems().size() == 0)){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: нет данных для записи!");

return;

}

try {

JDBCManager.WriteDataToDB\_Register(\_table);

ReaderMain.MessageShow(Alert.AlertType.INFORMATION, "Информация", "Данные записаны в базу данных!");

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

\_table.getItems().clear();

}

});

\_outputDB.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

try {

TableView<RegisterData> book = JDBCManager.ReadDataFromDB\_Register(true);

\_table.getItems().clear();

for(int i = 0; i < book.getItems().size(); i++)

\_table.getItems().add(book.getItems().get(i));

} catch (Exception e) {}

ReaderMain.MessageShow(Alert.AlertType.INFORMATION, "Информация", "Данные считаны из базы данных!");

}

});

\_execute2.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

\_resultExecute2.clear();

if(\_fio.getText().toString().length() != ReaderMain.MAX\_LENGTH\_PASSWORD){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Некорректные паспортные данные читателя!");

return;

}

TableView<RegisterData> registers = null;

try {

registers = JDBCManager.ReadDataFromDB\_Register(false);

} catch (SQLException throwables) {

throwables.printStackTrace();

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

if(registers.getItems().size() <= 0){

ReaderMain.MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Нет данных о регистрации книг за читателями!");

return;

}

ArrayList<RegisterData> data = new ArrayList<>();

for(int i = 0; i < registers.getItems().size(); i++){

if(registers.getItems().get(i).getPassword().equals(\_fio.getText().toString())){

data.add(registers.getItems().get(i));

}

}

data.sort(new Comparator<RegisterData>() {

@Override

public int compare(RegisterData o1, RegisterData o2) {

DateFormat format = new SimpleDateFormat("yyyy-mm-dd");

try {

if(format.parse(o1.getDateIssue()).after(

format.parse(o1.getDateReturn().toString())

)){

return 1;

}else if(format.parse(o1.getDateIssue()).before(

format.parse(o1.getDateReturn().toString())

)){

return (-1);

}else{

return 0;

}

} catch (ParseException e) {

return 0;

}

}

});

for(int i = 0; i < data.size(); i++){

try {

BookData bookData = GetBookByPasswordData(data.get(i).getPassword());

if(bookData != null){

\_resultExecute2.setText(\_resultExecute2.getText().toString() + "\n" +

bookData.getName() + ", " + data.get(i).getDateIssue());

}

} catch (SQLException throwables) {

throwables.printStackTrace();

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

}

}

});

}

}

**Класс BookData:**

package programGUI.data;

import javafx.beans.property.SimpleIntegerProperty;

import javafx.beans.property.SimpleStringProperty;

public class BookData {

private SimpleStringProperty register;

private SimpleStringProperty name;

private SimpleIntegerProperty pages;

private SimpleIntegerProperty year;

private SimpleStringProperty section;

public BookData(String reg, String name, int pages, int year, String section){

this.register = new SimpleStringProperty(reg);

this.name = new SimpleStringProperty(name);

this.pages = new SimpleIntegerProperty(pages);

this.year = new SimpleIntegerProperty(year);

this.section = new SimpleStringProperty(section);

}

public String getRegister() {

return register.get();

}

public SimpleStringProperty registerProperty() {

return register;

}

public String getName() {

return name.get();

}

public SimpleStringProperty nameProperty() {

return name;

}

public int getPages() {

return pages.get();

}

public SimpleIntegerProperty pagesProperty() {

return pages;

}

public int getYear() {

return year.get();

}

public SimpleIntegerProperty yearProperty() {

return year;

}

public String getSection() {

return section.get();

}

public SimpleStringProperty sectionProperty() {

return section;

}

public void setRegister(String register) {

this.register.set(register);

}

public void setName(String name) {

this.name.set(name);

}

public void setPages(int pages) {

this.pages.set(pages);

}

public void setYear(int year) {

this.year.set(year);

}

public void setSection(String section) {

this.section.set(section);

}

}

**Класс InfoReaderData:**

package programGUI.data;

import javafx.beans.property.SimpleIntegerProperty;

import javafx.beans.property.SimpleStringProperty;

public class InfoReaderData {

private SimpleStringProperty fio;

private SimpleStringProperty register;

private SimpleStringProperty password;

private SimpleStringProperty dateIssue;

private SimpleIntegerProperty countBook;

public InfoReaderData(String fio, String reg, String pas, String dateIssue, String dateReturn, int count){

this.fio = new SimpleStringProperty(fio);

this.register = new SimpleStringProperty(reg);

this.password = new SimpleStringProperty(pas);

this.dateIssue = new SimpleStringProperty(dateIssue);

this.dateReturn = new SimpleStringProperty(dateReturn);

this.countBook = new SimpleIntegerProperty(count);

}

public void setFio(String fio) {

this.fio.set(fio);

}

public void setRegister(String register) {

this.register.set(register);

}

public void setPassword(String password) {

this.password.set(password);

}

public void setDateIssue(String dateIssue) {

this.dateIssue.set(dateIssue);

}

public void setDateReturn(String dateReturn) {

this.dateReturn.set(dateReturn);

}

public void setCountBook(int countBook) {

this.countBook.set(countBook);

}

private SimpleStringProperty dateReturn;

public String getFio() {

return fio.get();

}

public SimpleStringProperty fioProperty() {

return fio;

}

public String getRegister() {

return register.get();

}

public SimpleStringProperty registerProperty() {

return register;

}

public String getPassword() {

return password.get();

}

public SimpleStringProperty passwordProperty() {

return password;

}

public String getDateIssue() {

return dateIssue.get();

}

public SimpleStringProperty dateIssueProperty() {

return dateIssue;

}

public String getDateReturn() {

return dateReturn.get();

}

public SimpleStringProperty dateReturnProperty() {

return dateReturn;

}

public int getCountBook() {

return countBook.get();

}

public SimpleIntegerProperty countBookProperty() {

return countBook;

}

}

**Класс ReaderData:**

package programGUI.data;

import javafx.beans.property.SimpleStringProperty;

public class ReaderData {

private SimpleStringProperty password;

private SimpleStringProperty address;

private SimpleStringProperty fio;

public ReaderData(String pas, String home, String fname){

this.password = new SimpleStringProperty(pas);

this.address = new SimpleStringProperty(home);

this.fio = new SimpleStringProperty(fname);

}

public void setPassword(String password) {

this.password.set(password);

}

public void setAddress(String address) {

this.address.set(address);

}

public void setFio(String fio) {

this.fio.set(fio);

}

public String getPassword() {

return password.get();

}

public SimpleStringProperty passwordProperty() {

return password;

}

public String getAddress() {

return address.get();

}

public SimpleStringProperty addressProperty() {

return address;

}

public String getFio() {

return fio.get();

}

public SimpleStringProperty fioProperty() {

return fio;

}

}

**Класс RegisterData:**

package programGUI.data;

import javafx.beans.property.SimpleStringProperty;

public class RegisterData {

private SimpleStringProperty register;

private SimpleStringProperty password;

private SimpleStringProperty dateIssue;

private SimpleStringProperty dateReturn;

public RegisterData(String reg, String pas, String dateIssue, String dateReturn){

this.register = new SimpleStringProperty(reg);

this.password = new SimpleStringProperty(pas);

this.dateIssue = new SimpleStringProperty(dateIssue);

this.dateReturn = new SimpleStringProperty(dateReturn);

}

public void setRegister(String register) {

this.register.set(register);

}

public void setPassword(String password) {

this.password.set(password);

}

public void setDateIssue(String dateIssue) {

this.dateIssue.set(dateIssue);

}

public void setDateReturn(String dateReturn) {

this.dateReturn.set(dateReturn);

}

public String getRegister() {

return register.get();

}

public SimpleStringProperty registerProperty() {

return register;

}

public String getPassword() {

return password.get();

}

public SimpleStringProperty passwordProperty() {

return password;

}

public String getDateIssue() {

return dateIssue.get();

}

public SimpleStringProperty dateIssueProperty() {

return dateIssue;

}

public String getDateReturn() {

return dateReturn.get();

}

public SimpleStringProperty dateReturnProperty() {

return dateReturn;

}

}

**Файл разметки viewBook:**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.Menu?>

<?import javafx.scene.control.MenuBar?>

<?import javafx.scene.control.MenuItem?>

<?import javafx.scene.control.TableColumn?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.HBox?>

<?import javafx.scene.layout.StackPane?>

<?import javafx.scene.layout.VBox?>

<StackPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="564.0" prefWidth="832.0" xmlns="http://javafx.com/javafx/15.0.1" xmlns:fx="http://javafx.com/fxml/1">

<children>

<VBox accessibleRole="BUTTON" prefHeight="394.0" prefWidth="600.0">

<children>

<MenuBar id="\_menuBar">

<menus>

<Menu mnemonicParsing="false" text="File">

<items>

<MenuItem mnemonicParsing="false" text="Close" />

</items>

</Menu>

<Menu mnemonicParsing="false" text="Edit">

<items>

<MenuItem mnemonicParsing="false" text="Delete" />

</items>

</Menu>

<Menu mnemonicParsing="false" text="Help">

<items>

<MenuItem mnemonicParsing="false" text="About" />

</items>

</Menu>

</menus>

</MenuBar>

<VBox prefHeight="14.0" prefWidth="600.0" />

<HBox prefHeight="235.0" prefWidth="686.0">

<children>

<TableView id="\_table" prefHeight="235.0" prefWidth="822.0">

<columns>

<TableColumn prefWidth="75.0" text="C1" />

<TableColumn prefWidth="75.0" text="C2" />

</columns>

</TableView>

</children>

</HBox>

<VBox prefHeight="25.0" prefWidth="600.0" />

<HBox prefHeight="150.0" prefWidth="686.0">

<children>

<VBox prefHeight="170.0" prefWidth="186.0">

<children>

<VBox prefHeight="4.0" prefWidth="186.0" />

<Label alignment="CENTER\_RIGHT" prefHeight="18.0" prefWidth="200.0" text="Регистрационный номер:" />

<VBox prefHeight="13.0" prefWidth="186.0" />

<Label alignment="CENTER\_RIGHT" prefHeight="18.0" prefWidth="242.0" text="Название книги:" />

<VBox prefHeight="13.0" prefWidth="186.0" />

<Label alignment="CENTER\_RIGHT" prefHeight="18.0" prefWidth="223.0" text="Количество страниц:" />

<VBox prefHeight="11.0" prefWidth="186.0">

<children>

<VBox prefHeight="200.0" prefWidth="100.0" />

</children>

</VBox>

<Label alignment="CENTER\_RIGHT" prefHeight="18.0" prefWidth="225.0" text="Год публикации:" />

<VBox prefHeight="14.0" prefWidth="186.0" />

<Label alignment="CENTER\_RIGHT" prefHeight="18.0" prefWidth="239.0" text="Раздел:" />

</children>

</VBox>

<VBox prefHeight="150.0" prefWidth="13.0" />

<VBox prefHeight="150.0" prefWidth="180.0">

<children>

<TextField id="\_txtRegNumber" alignment="CENTER" prefHeight="14.0" prefWidth="180.0" />

<VBox prefHeight="5.0" prefWidth="180.0" />

<TextField id="\_txtNameBook" alignment="CENTER" />

<VBox prefHeight="8.0" prefWidth="180.0" />

<TextField id="\_txtCountPages" alignment="CENTER" />

<VBox prefHeight="4.0" prefWidth="180.0" />

<TextField id="\_txtYearPublishing" alignment="CENTER" />

<VBox prefHeight="6.0" prefWidth="180.0" />

<TextField id="\_txtSection" alignment="CENTER" />

</children>

</VBox>

<VBox prefHeight="150.0" prefWidth="13.0" />

<VBox prefHeight="150.0" prefWidth="136.0">

<children>

<Button id="\_btnAdd" mnemonicParsing="false" prefHeight="26.0" prefWidth="165.0" text="Добавить" />

<VBox prefHeight="114.0" prefWidth="136.0" />

<Button id="\_btnRefactor" mnemonicParsing="false" prefHeight="26.0" prefWidth="158.0" text="Изменить" />

<VBox prefHeight="115.0" prefWidth="136.0" />

<Button id="\_btnDelete" mnemonicParsing="false" prefHeight="26.0" prefWidth="196.0" text="Удалить" />

</children>

</VBox>

<VBox prefHeight="150.0" prefWidth="11.0" />

<VBox prefHeight="150.0" prefWidth="287.0">

<children>

<Button id="\_btnInputDB" mnemonicParsing="false" prefHeight="26.0" prefWidth="310.0" text="Добавить в базу данных" />

<VBox prefHeight="126.0" prefWidth="287.0" />

<Button id="\_btnOutputDB" mnemonicParsing="false" prefHeight="26.0" prefWidth="314.0" text="Считать из базы данных" />

</children>

</VBox>

</children>

</HBox>

</children>

</VBox>

</children>

</StackPane>

**Файл разметки viewReader:**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.Menu?>

<?import javafx.scene.control.MenuBar?>

<?import javafx.scene.control.MenuItem?>

<?import javafx.scene.control.TableColumn?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.HBox?>

<?import javafx.scene.layout.StackPane?>

<?import javafx.scene.layout.VBox?>

<StackPane fx:id="root" xmlns="http://javafx.com/javafx/15.0.1" xmlns:fx="http://javafx.com/fxml/1">

<children>

<VBox prefHeight="574.0" prefWidth="742.0">

<children>

<MenuBar id="\_menuBar" accessibleRole="BUTTON">

<menus>

<Menu mnemonicParsing="false" text="File">

<items>

<MenuItem mnemonicParsing="false" text="Close" />

</items>

</Menu>

<Menu mnemonicParsing="false" text="Edit">

<items>

<MenuItem mnemonicParsing="false" text="Delete" />

</items>

</Menu>

<Menu mnemonicParsing="false" text="Help">

<items>

<MenuItem mnemonicParsing="false" text="About" />

</items>

</Menu>

</menus>

</MenuBar>

<VBox prefHeight="21.0" prefWidth="742.0" />

<HBox prefHeight="234.0" prefWidth="742.0">

<children>

<TableView id="\_tableReader" prefHeight="148.0" prefWidth="650.0">

<columns>

<TableColumn prefWidth="75.0" text="C1" />

<TableColumn prefWidth="85.5999755859375" text="C2" />

</columns>

</TableView>

</children>

</HBox>

<VBox prefHeight="29.0" prefWidth="742.0" />

<HBox prefHeight="93.0" prefWidth="742.0">

<children>

<VBox prefHeight="174.0" prefWidth="166.0">

<children>

<Label alignment="CENTER\_RIGHT" contentDisplay="CENTER" prefHeight="26.0" prefWidth="166.0" text="Паспортные данные:" />

<VBox prefHeight="16.0" prefWidth="166.0" />

<Label alignment="CENTER\_RIGHT" contentDisplay="CENTER" prefHeight="25.0" prefWidth="166.0" text="Домашний адрес:" />

<VBox prefHeight="16.0" prefWidth="166.0" />

<Label alignment="CENTER\_RIGHT" contentDisplay="CENTER" prefHeight="18.0" prefWidth="193.0" text="ФИО:" />

</children>

</VBox>

<HBox prefHeight="94.0" prefWidth="16.0" />

<VBox prefHeight="94.0" prefWidth="161.0">

<children>

<TextField id="\_txtPasswordData" alignment="CENTER" />

<VBox prefHeight="17.0" prefWidth="162.0" />

<TextField id="\_txtHomeAddress" alignment="CENTER" />

<VBox prefHeight="17.0" prefWidth="162.0" />

<TextField id="\_txtFio" alignment="CENTER" />

</children>

</VBox>

<HBox prefHeight="82.0" prefWidth="19.0" />

<VBox alignment="TOP\_RIGHT" prefHeight="94.0" prefWidth="85.0">

<children>

<Button id="\_btnAddData" mnemonicParsing="false" prefHeight="20.0" prefWidth="86.0" text="Добавить" />

<VBox prefHeight="12.0" prefWidth="86.0" />

<Button id="\_btnRefactor" mnemonicParsing="false" prefHeight="26.0" prefWidth="98.0" text="Изменить" />

<VBox prefHeight="12.0" prefWidth="86.0" />

<Button id="\_btnDelete" mnemonicParsing="false" prefHeight="26.0" prefWidth="105.0" text="Удалить" />

</children>

</VBox>

<HBox prefHeight="94.0" prefWidth="22.0" />

<VBox prefHeight="94.0" prefWidth="266.0">

<children>

<Button id="\_btnInputDB" mnemonicParsing="false" prefHeight="26.0" prefWidth="272.0" text="Добавить в базу данных" />

<VBox prefHeight="54.0" prefWidth="266.0" />

<Button id="\_btnOutputDB" mnemonicParsing="false" prefHeight="26.0" prefWidth="276.0" text="Считать из базы данных" />

</children>

</VBox>

</children>

</HBox>

</children>

</VBox>

</children>

</StackPane>

**Файл разметки viewRegister:**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.Menu?>

<?import javafx.scene.control.MenuBar?>

<?import javafx.scene.control.MenuItem?>

<?import javafx.scene.control.TableColumn?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.control.TextArea?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.HBox?>

<?import javafx.scene.layout.StackPane?>

<?import javafx.scene.layout.VBox?>

<StackPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="832.0" prefWidth="847.0" xmlns="http://javafx.com/javafx/15.0.1" xmlns:fx="http://javafx.com/fxml/1">

<children>

<VBox prefHeight="547.0" prefWidth="655.0">

<children>

<MenuBar id="\_menuBar">

<menus>

<Menu mnemonicParsing="false" text="File">

<items>

<MenuItem mnemonicParsing="false" text="Close" />

</items>

</Menu>

<Menu mnemonicParsing="false" text="Edit">

<items>

<MenuItem mnemonicParsing="false" text="Delete" />

</items>

</Menu>

<Menu mnemonicParsing="false" text="Help">

<items>

<MenuItem mnemonicParsing="false" text="About" />

</items>

</Menu>

</menus>

</MenuBar>

<HBox prefHeight="201.0" prefWidth="672.0">

<children>

<TableView id="\_table" prefHeight="202.0" prefWidth="775.0">

<columns>

<TableColumn prefWidth="75.0" text="C1" />

<TableColumn prefWidth="75.0" text="C2" />

</columns>

</TableView>

</children>

</HBox>

<VBox prefHeight="14.0" prefWidth="672.0" />

<HBox prefHeight="114.0" prefWidth="672.0">

<children>

<VBox prefHeight="112.0" prefWidth="223.0">

<children>

<VBox prefHeight="6.0" prefWidth="223.0" />

<Label alignment="CENTER\_RIGHT" prefHeight="18.0" prefWidth="259.0" text="Регистрационный номер:" />

<VBox prefHeight="8.0" prefWidth="223.0" />

<Label alignment="CENTER\_RIGHT" prefHeight="18.0" prefWidth="296.0" text="Паспортные данные:" />

<VBox prefHeight="9.0" prefWidth="223.0" />

<Label alignment="CENTER\_RIGHT" prefHeight="18.0" prefWidth="286.0" text="Дата выдачи:" />

<VBox prefHeight="7.0" prefWidth="223.0" />

<Label alignment="CENTER\_RIGHT" prefHeight="18.0" prefWidth="273.0" text="Дата возврата:" />

</children>

</VBox>

<VBox prefHeight="114.0" prefWidth="16.0" />

<VBox prefHeight="114.0" prefWidth="146.0">

<children>

<VBox prefHeight="0.0" prefWidth="146.0" />

<TextField id="\_txtRegNumber" alignment="CENTER" />

<VBox prefHeight="3.0" prefWidth="146.0" />

<TextField id="\_txtPassword" alignment="CENTER" />

<VBox prefHeight="2.0" prefWidth="146.0" />

<TextField id="\_txtDateIssue" alignment="CENTER" />

<VBox prefHeight="4.0" prefWidth="146.0" />

<TextField id="\_txtDateReturn" alignment="CENTER" />

</children>

</VBox>

<VBox prefHeight="114.0" prefWidth="12.0" />

<VBox prefHeight="150.0" prefWidth="136.0">

<children>

<Button id="\_btnAdd" mnemonicParsing="false" prefHeight="26.0" prefWidth="165.0" text="Добавить" />

<VBox prefHeight="114.0" prefWidth="136.0" />

<Button id="\_btnRefactor" mnemonicParsing="false" prefHeight="26.0" prefWidth="158.0" text="Изменить" />

<VBox prefHeight="115.0" prefWidth="136.0" />

<Button id="\_btnDelete" mnemonicParsing="false" prefHeight="26.0" prefWidth="196.0" text="Удалить" />

</children>

</VBox>

<VBox prefHeight="114.0" prefWidth="16.0" />

<VBox prefHeight="150.0" prefWidth="287.0">

<children>

<Button id="\_btnInputDB" mnemonicParsing="false" prefHeight="26.0" prefWidth="310.0" text="Добавить в базу данных" />

<VBox prefHeight="126.0" prefWidth="287.0" />

<Button id="\_btnOutputDB" mnemonicParsing="false" prefHeight="26.0" prefWidth="314.0" text="Считать из базы данных" />

</children>

</VBox>

</children>

</HBox>

<VBox prefHeight="23.0" prefWidth="838.0" />

<HBox prefHeight="22.0" prefWidth="838.0">

<children>

<VBox prefHeight="55.0" prefWidth="323.0">

<children>

<Label alignment="CENTER" prefHeight="26.0" prefWidth="356.0" text="Список читателей, которые держат на руках книги более " />

</children>

</VBox>

<VBox prefHeight="55.0" prefWidth="9.0" />

<TextField id="\_txtMonth" alignment="CENTER" prefHeight="26.0" prefWidth="38.0" />

<VBox prefHeight="26.0" prefWidth="10.0" />

<Label alignment="CENTER" prefHeight="26.0" prefWidth="86.0" text="месяц(-а/-ев)" />

<HBox prefHeight="26.0" prefWidth="11.0" />

<Button id="\_btnExecute1" mnemonicParsing="false" prefHeight="26.0" prefWidth="105.0" text="Сформировать" />

</children>

</HBox>

<VBox prefHeight="14.0" prefWidth="838.0" />

<HBox prefHeight="244.0" prefWidth="847.0">

<children>

<TableView id="\_table2" prefHeight="214.0" prefWidth="798.0">

<columns>

<TableColumn prefWidth="75.0" text="C1" />

<TableColumn prefWidth="75.0" text="C2" />

</columns>

</TableView>

</children>

</HBox>

<VBox prefHeight="15.0" prefWidth="672.0" />

<HBox prefHeight="155.0" prefWidth="847.0">

<children>

<VBox prefHeight="155.0" prefWidth="210.0">

<children>

<Label alignment="CENTER" prefHeight="18.0" prefWidth="208.0" text=" Для читателя (Паспортные данные)" />

<VBox prefHeight="11.0" prefWidth="158.0" />

<TextField id="\_txtFio" alignment="CENTER" />

<VBox prefHeight="10.0" prefWidth="158.0" />

<Button id="\_btnExecute2" mnemonicParsing="false" prefHeight="26.0" prefWidth="214.0" text="Выдать список книг" />

</children>

</VBox>

<VBox prefHeight="155.0" prefWidth="63.0" />

<TextArea id="\_txtBooks" prefHeight="155.0" prefWidth="469.0" />

</children>

</HBox>

</children>

</VBox>

</children>

</StackPane>

8 Заключение

Были получены общие представления об объектном подходе к организации доступа к базе данных (БД), выполнению запросов и обработке результатов. Был получен практический опыт организации работы с БД средствами платформы java.

Было разработано java-приложение с графическим пользовательским инетфейсом позволяющее просматривать и редактировать информацию из базы данных. Была организована и наполнена база данных в соответствии с предметной областью “Библиотека”.