

Library Information System

Lito



Ibraimov Marlen com-18

Ala-Too International University

Table of Contents

ACKNOWLEDGEMENT.....	3
INTRODUCTION.....	4
EXPLANATIONS	5
OBJECT ORIENTED EXPLANATION:	11
ASSUMPTION	14
REFERENCE.....	15

ACKNOWLEDGEMENT

First of all, I would like to thank my lecturer Mr. Nurlan Shaidullaev for helping me to acquire some basic knowledge of “Java Programming Language”. At the same time, he gave me the opportunity to learn something new related to our module like constructors, methods, arrays, JFrames etc.

Beside from my lecturer, I like to thank my other classmates for helping to understand the assignment related questions more clearly. They gave their best for completing this report on time. I thank them for their efforts.

INTRODUCTION

This program is based on [*Koha Library System*](#) using “Java Programming Language”. For that we used **JavaFX** in this development so that it will become easy with graphical interface another word with **Scene Builder** program.

Besides, I also studied [MySQL](#) for storing data in Database.

EXPLANATIONS

In this documentation we have given explanations of how to interact successfully with this program “Heart of Math”. We have explained here step by step so that it will surely help users to become more user friendly with it. Below are our explanations:

First Things First: Before execute this program, users need to do some works so that it will run properly into their system. First, they need to make sure their system is having “JDK”. If they don’t have it then they can download from this below link:

<https://www.oracle.com/technetwork/java/javase/downloads/jdk11-downloads-5066655.html>

Depending on their system (Windows 64bit/32bit) they need to download and install. Then they need to add the “JAVA” files to their system “PATH” so that the system can run the program from CMD (Command Prompt). The path will show something like this “C:\Program Files\Java\jdk1.6.0_02.”. Now just add the address besides the current path directory and save it.

The other way they can execute this program in to download the IDE (Integrated Development Environment) on their system. They can download IntelliJ IDEA or Eclipse depending on the windows (32bit/64bit). Below is the link:

IntelliJ IDEA:

<https://www.jetbrains.com/ru-ru/idea/download/>

Eclipse:

<http://www.eclipse.org/downloads/>

We developed this program using “IntelliJ IDEA”.

Execution Procedures:

When user executes this program, it will show this:

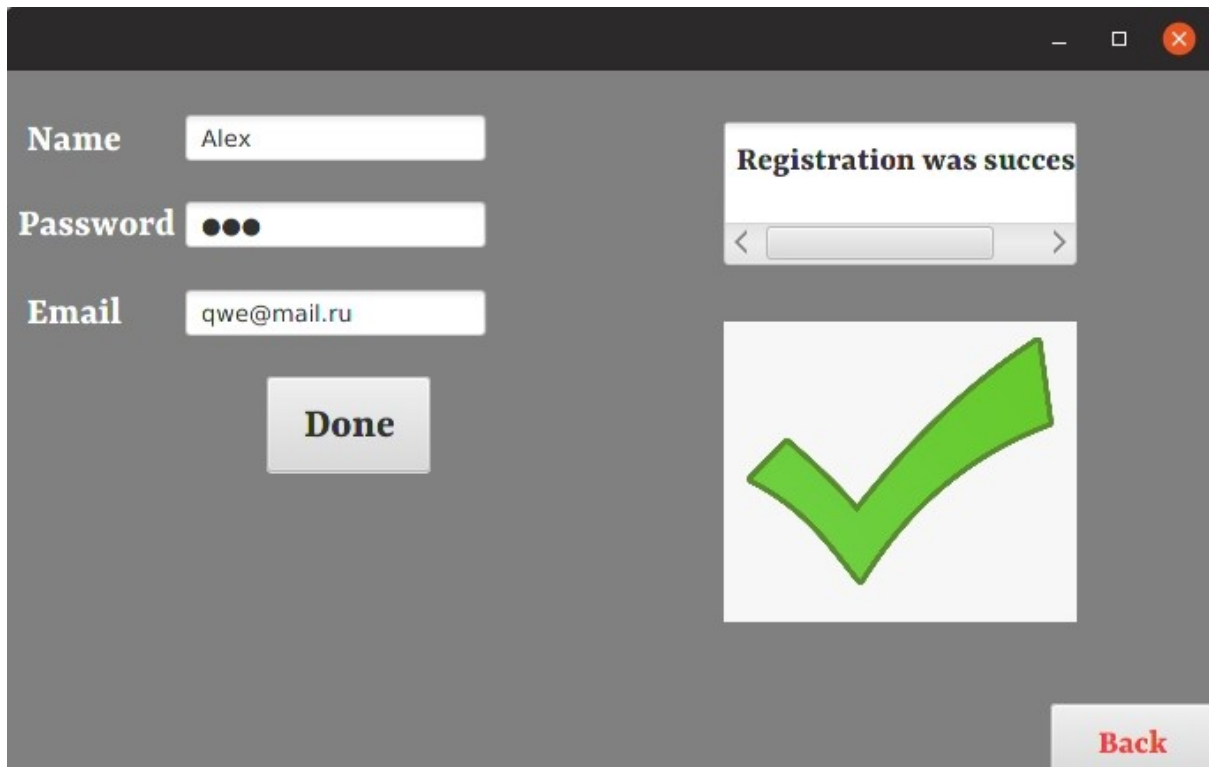


There is my main menu, as you can see there are two options .

After clicking Sign up you will see this:

A screenshot of a web application window showing a sign-up form. The background is a solid gray. On the left, there are three labels: "Name", "Password", and "Email", each followed by a white rectangular input field. To the right of these fields is a larger white rectangular area, possibly for a profile picture or a confirmation code. Below the input fields is a white rectangular button with the text "Done". In the bottom right corner, there is a white rectangular button with the text "Back" in red. The window has standard OS controls (minimize, maximize, close) in the top right corner.

It is my registration form to creating new user that will have opportunity to use my program



The image shows a graphical user interface for a registration form. On the left, there are three labeled input fields: "Name" containing "Alex", "Password" containing three black dots, and "Email" containing "qwe@mail.ru". Below the "Email" field is a button labeled "Done". To the right of these fields is a white rectangular box with the text "Registration was succes" (sic) in bold. Below this text is a large green checkmark. At the bottom right of the interface is a button labeled "Back". The entire form is set against a dark gray background within a window frame.

If the records satisfy requirements(if no user with same name and password) ,record will be saved into database system. In another case it will throw error.



Clicking Log in

Then you will see new window:

A screenshot of a web application window showing a login form. The form has a light gray background. At the top, the text "UserName" is displayed above a text input field containing the name "Mara". Below this, the text "Password" is displayed above a password input field with three black dots. A white button labeled "Log in" is positioned below the password field. In the bottom right corner, there is a small gray button labeled "Back" in red text.

We are logged as user Mara

Hello,I'm LITO

Lito Library

Search

Select language
EN

Quit

Title	Author	Category	Year	Riting
a waki to remember	Nicholas Sparks	Drama	1999	4

There are books inside my system and user can go to library and take book for one week.

Also you can log in as admin:

Hello,I'm LITO

Admin Dashboard

a waki to remember
Dota

Add
Delete

Title

Author

Category

Year

Riting

Back

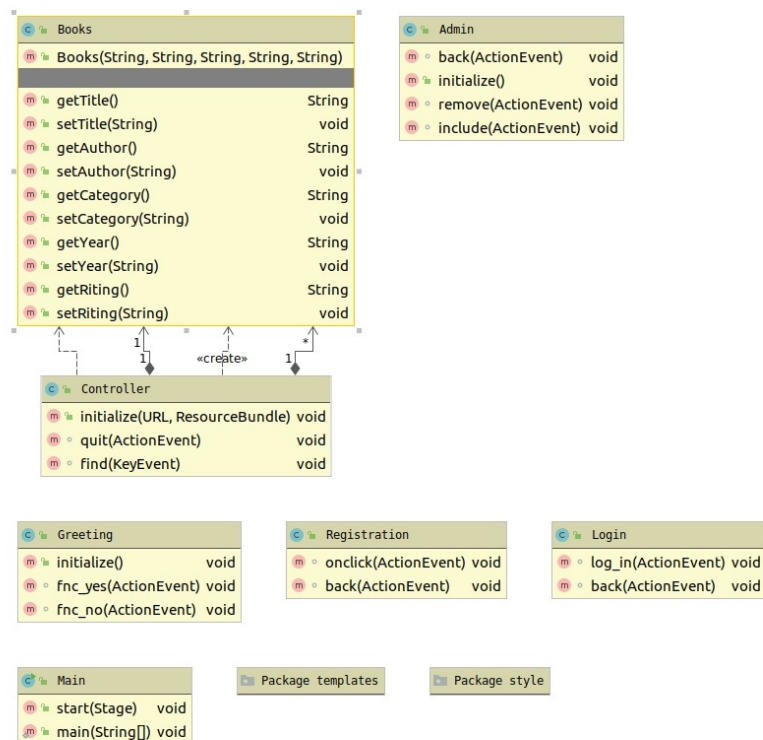
We are logged as Admin user

Admin can add books and delete books from system. It's more useful.

OBJECT ORIENTED EXPLANATION

In object-oriented programming, for example, an object is a self-contained entity that consists of both data and procedures to manipulate the data. In other way, object oriented is the software engineering concept where it is represented using the “OBJECTS”. Below are the objected oriented samples we used in this “Java Programming Language”.

Dependency:



As you can see there are dependencies. I did not used a lot of.

```

public void initialize(URL url, ResourceBundle rb) {
    try {
        Books books;
        Books books1;
        String myUrl = "jdbc:mysql://localhost/lito";
        Connection conn = DriverManager.getConnection(myUrl, user: "username", password: "password");
        String get_request = "SELECT * FROM books";
        Statement statement = conn.createStatement();
        ResultSet rs = statement.executeQuery(get_request);
        choicebox.getItems().add("EN");
        choicebox.getItems().add("RU");
        choicebox.getSelectionModel().selectedIndexProperty().addListener(new ChangeListener<Number>() {
            @Override
            public void changed(ObservableValue<? extends Number> observableValue, Number number, Number number2) {
                String choice = choicebox.getItems().get((Integer) number2);
                if (choice.equals("EN")) {
                    en_content.setVisible(true);
                    ru_content.setVisible(false);
                }
                if (choice.equals("RU")) {
                    en_content.setVisible(false);
                    ru_content.setVisible(true);
                }
            }
        });

        while (rs.next()) {
            books = new Books(rs.getString(2), rs.getString(3), rs.getString(4), rs.getString(5), rs.getString(6));

            oblist.addAll(books);
        }
        get_request = "SELECT * FROM ru_books";
        ResultSet rs_ru = statement.executeQuery(get_request);

        while (rs_ru.next()) {
            books1 = new Books(rs_ru.getString(2), rs_ru.getString(3), rs_ru.getString(4), rs_ru.getString(5), rs_ru.getString(6));
            oblist_2.addAll(books1);
        }
    }
}

```

The initialize method is called after all @FXML then it create Books instances which will be inside ObservableList that we need to implement by FXCollections.observableArrayList();

Then I will put all of that into TableView (JavaFX) to represent.

Also there was used Listener for track behavior of choicebox.

In order to use DB we:

- 1)Establishing a connection
- 2)Create a statement
- 3)Execute the query

```

public class Books {

    String title, author, category, year, riting;

    public Books(String title, String author, String category, String year, String riting) {
        this.title = title;
        this.author = author;
        this.category = category;
        this.year = year;
        this.riting = riting;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) { this.title = title; }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) { this.author = author; }

    public String getCategory() { return category; }

    public void setCategory(String category) { this.category = category; }

    public String getYear() { return year; }

    public void setYear(String year) { this.year = year; }

    public String getRiting() { return riting; }

    public void setRiting(String riting) { this.riting = riting; }
}

```

My Book class with own setter,getter and Book constructor.

ASSUMPTION

I tried to make this app more useful for myself but i will fix some functionality and add flexible opportunities for users not only admins.

For instance people should have a possibility to add own favorites books and order books that not inside in the Library.

Reference

Github: <https://github.com/Marlen1703/qwe.git>

JavaFX: <https://openjfx.io/>

IntelliJ: IDEA: <https://www.jetbrains.com/idea/>