

Alatoo INFORMATION SYSTEM

Made by:
Kanybekov Azatbek
Zhusuev Timur
for CS-201:OOP in Ala-Too University

27.12.2021

Table of Contents

ACKNOWLEDGEMENT

1

INTRODUCTION

2

EXPLANATIONS	3
First Things First:	3
Execution Procedures:	3
Function (Catalogue):	6
Function (Search Books):	6
Function (Current Books):	7
Function (Add Book):	8
Function (Members):	8
OBJECT ORIENTED EXPLANATION	9
Object Oriented Samples:	9
Sample 1:	9
Sample 2:	10
Sample 3:	11
ASSUMPTION	13
REFERENCE	14

ACKNOWLEDGEMENT

First of all, I would like to thank my lecturer R.R Isaev for helping me to acquire some basic knowledge of “**Java Programming Language**” and “**Object Oriented Programming**”. At the same time, he gave me the opportunity to learn something new related to our module like constructors, methods, arrays, JavaFX 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 assignment is based on developing an AIS (Alatoo Information System) using “**Java Programming Language**”. For that I used GUI (Graphical User Interface) in this development so that it will become more users friendly to interact.

Besides, I also added text files for user's records that are directly linked with this program. It is so called a heart of this program where all the functions depend on it.

EXPLANATIONS

In this documentation I gave explanations of how to interact successfully with this AIS (Alatoo Information System). I have explained here step by step so that it will surely help users to become more user friendly with it. Below are my 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/jdk13-downloads-5672538.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 (x86) \Java\jre1.8.0_25\bin;**”. 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 ECLIPSE, NETBEANS or INTELLIJ IDEA depending on the windows (32bit/64bit). Below is the link:

NETBEANS:

<https://netbeans.org/downloads/> **ECLIPSE:**

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

IDEA:

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

I developed this program using “**INTELLIJ IDEA**”.

Execution Procedures:

When users execute program, they will see the startup GUI (Graphical User Interface) of this program (login screen):

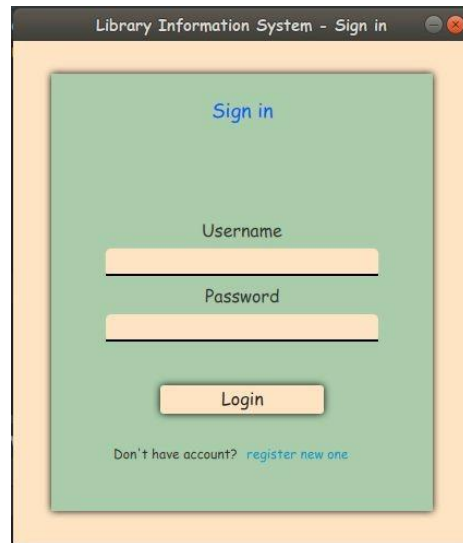


FIGURE 1: LOGIN SCREEN

User needs to enter username and password and click button “**Login**” for further proceed. If they don’t have account yet, they can click link “**register new one**” to enter registration screen:

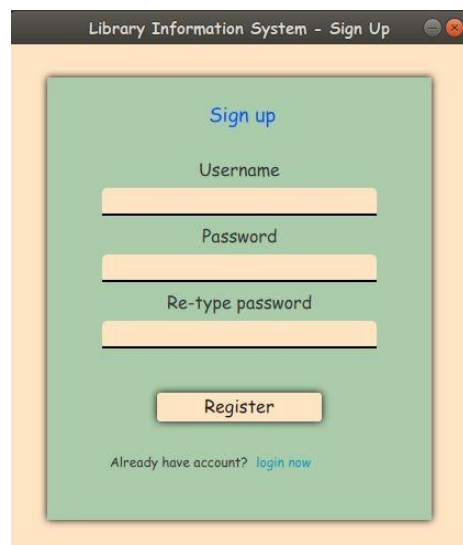


FIGURE 2: REGISTER SCREEN

If user enters invalid username or password system will show the warning message:



FIGURE 3: ERROR

If user leaves one of field empty, system will show warning message:

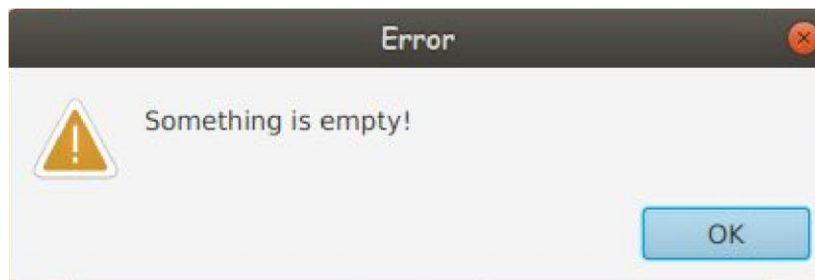


FIGURE 4: ERROR

If user enters valid username and password, system will show main menu depending on type of account (member or admin):

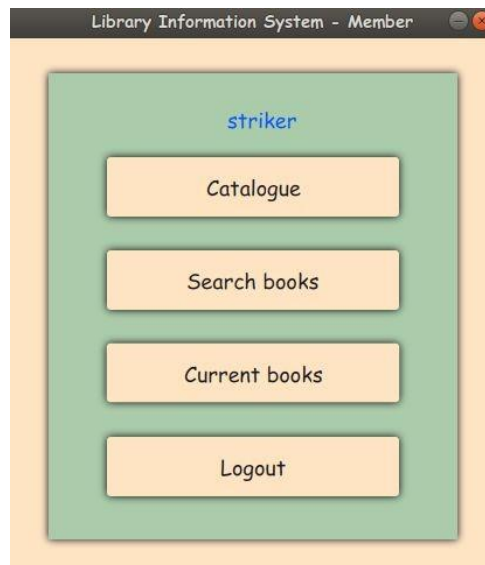


FIGURE 5: MAIN MENU - MEMBER

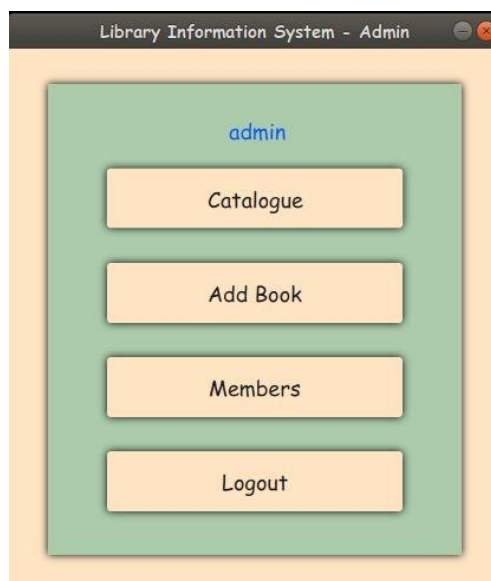


FIGURE 6: MAIN MENU - ADMIN

Function (Catalogue):

When user clicks button “Catalogue”, system will ask for type of book:

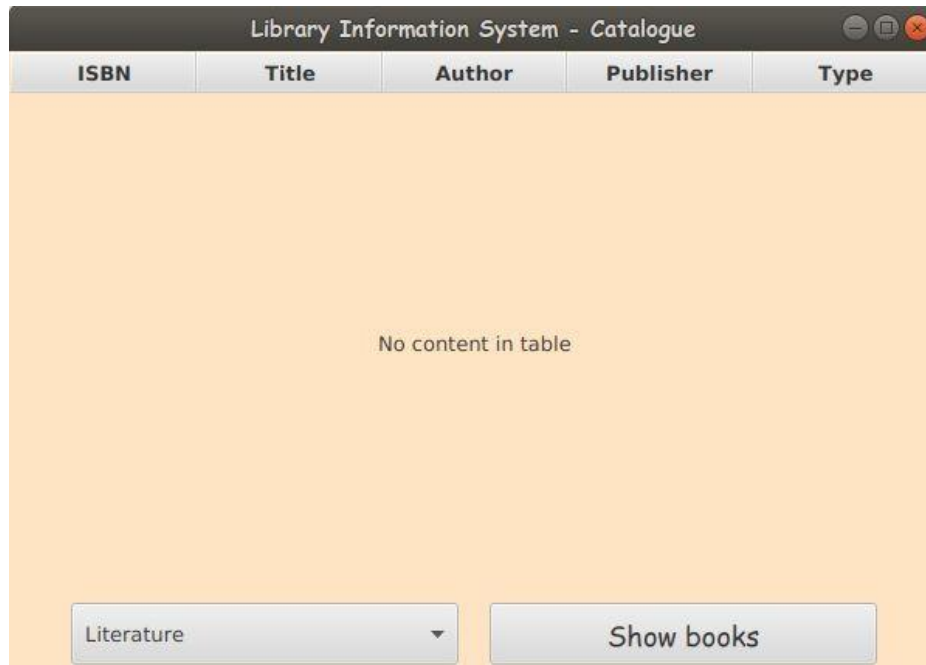


FIGURE 7: CATALOGUE WITHOUT BOOKS

When user chooses type and clicks “Show books”, system will show all books of that type:

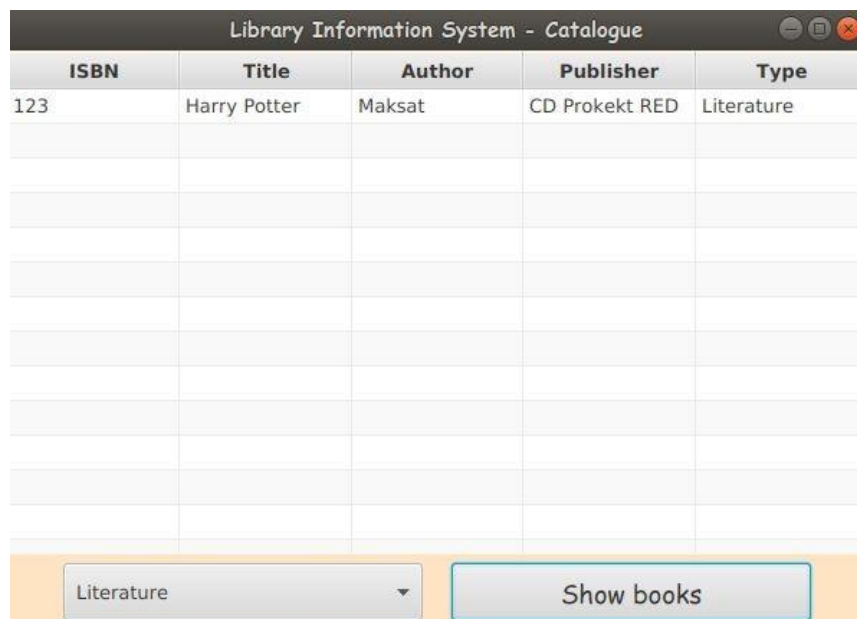


FIGURE 8: CATALOGUE WITH BOOKS

Function (Search Books):

When user clicks button “Search Books”, system will ask to type name of book:

ISBN	Title	Author	Publisher	Type
No content in table				

Name

Search

Take it Cancel

FIGURE 9: SEARCH BOOKS

When user types name of book and clicks button “Search”, system will show all books with name which user typed, and user can take this book:

ISBN	Title	Author	Publisher	Type
123	Harry Potter	Maksat	CD Prokekt RED	Literature

Name

Harry Potter

Search

Take it Cancel

FIGURE 10: TAKE BOOK

Function (Current Books):

When user clicks button “Current Books”, system will show all books belonging to user, and user can remove them:

Username	Password	Type
admin	admin	admin
Maksat	123	member
striker	eredivise	member

Convert Cancel

FIGURE 13: MEMBERS

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 I used in this “**Java Programming Language**”:

Object Oriented Samples:

Sample 1:

For each frame (FXML file) Java Controller class where it declares buttons, labels and textFields for user’s interaction:

```

import ...

public class LoginController implements Initializable {
    @FXML
    private TextField username;
    @FXML
    private PasswordField password;

    DatabaseHandler handler;

    public void goToRegisterPage(ActionEvent event) throws IOException {
        Parent root = FXMLLoader.load(getClass().getResource( "name: ../register/register.fxml"));
        Scene scene = new Scene(root);
        Stage appStage = (Stage) ((Node) event.getSource()).getScene().getWindow();
        appStage.setTitle("Library Information System - Sign Up");
        appStage.setScene(scene);
        appStage.show();
        appStage.setResizable(false);
    }

    public void tryLogin(ActionEvent event) throws IOException{
        if (password.getText().isEmpty() || username.getText().isEmpty()){
            Alert alert = new Alert(Alert.AlertType.WARNING);
            alert.setHeaderText(null);
            alert.setTitle("Error");
            alert.setContentText("Something is empty!");
            alert.showAndWait();
            return;
        }
    }
}

```

Sample 2:

This is Book class where I contain details about book like ISBN, title, author. When application starts console will show information for test:

```

package sample.classes;

public class Book {

    private final String isbn;
    private final String title;
    private final String author;
    private final String publisher;
    private final String type;

    public Book(String isbn, String title, String author, String publisher, String type) {
        this.isbn = isbn;
        this.title = title;
        this.author = author;
        this.publisher = publisher;
        this.type = type;
    }

    public String getISBN() { return isbn; }

    public String getTitle() { return title; }

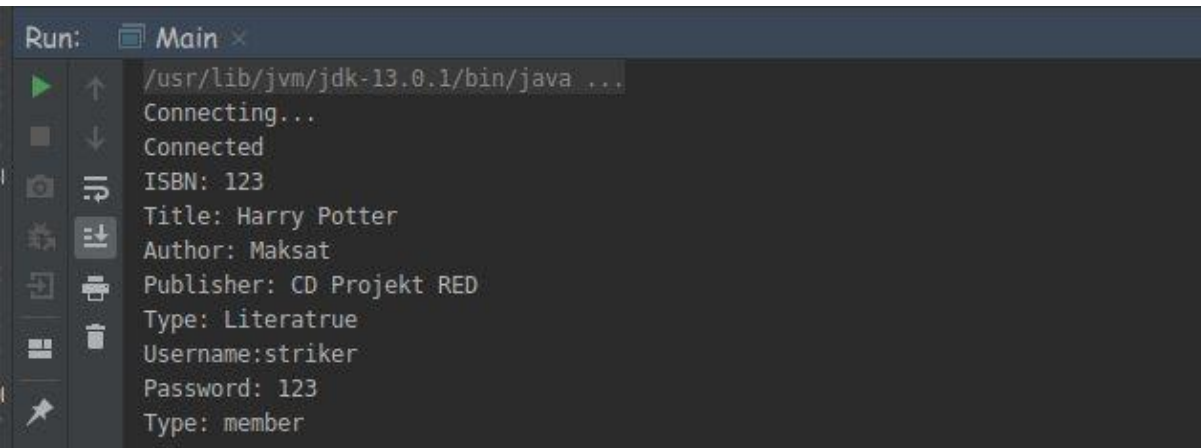
    public String getAuthor() { return author; }

    public String getPublisher() { return publisher; }

    public String getType() { return type; }

    public void printDetails(){
        System.out.println("ISBN: " + isbn);
    }
}

```



```

Run: Main x
/usr/lib/jvm/jdk-13.0.1/bin/java ...
Connecting...
Connected
ISBN: 123
Title: Harry Potter
Author: Maksat
Publisher: CD Projekt RED
Type: Literatrue
Username:striker
Password: 123
Type: member

```

Sample 3:

This is DatabaseHandler class which controls connection between database and program:

```

public class DatabaseHandler {
    private static DatabaseHandler handler = null;

    private static final String DB_URL = "jdbc:mysql://localhost:3306/test";
    private static final String USERNAME = "root";
    private static final String PASSWORD = "eredivise";
    private static Connection conn = null;
    private static Statement stmt = null;

    private DatabaseHandler() { createConnection(); }

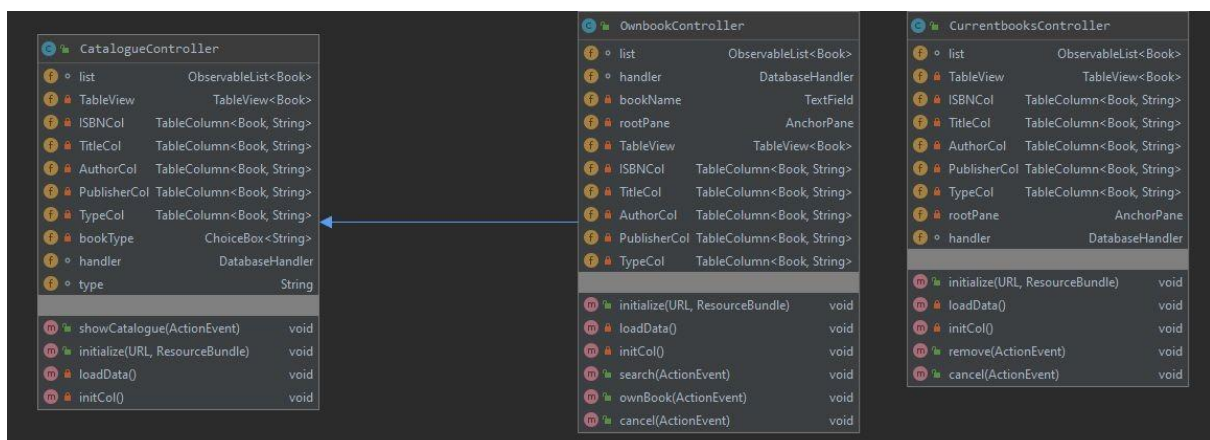
    public static DatabaseHandler getInstance(){
        if (handler == null){
            handler = new DatabaseHandler();
        }
        return handler;
    }

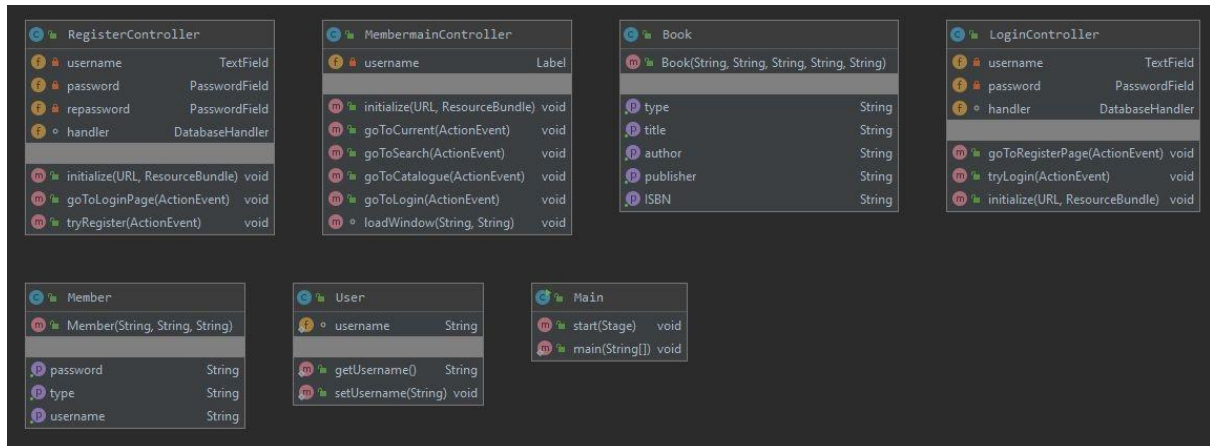
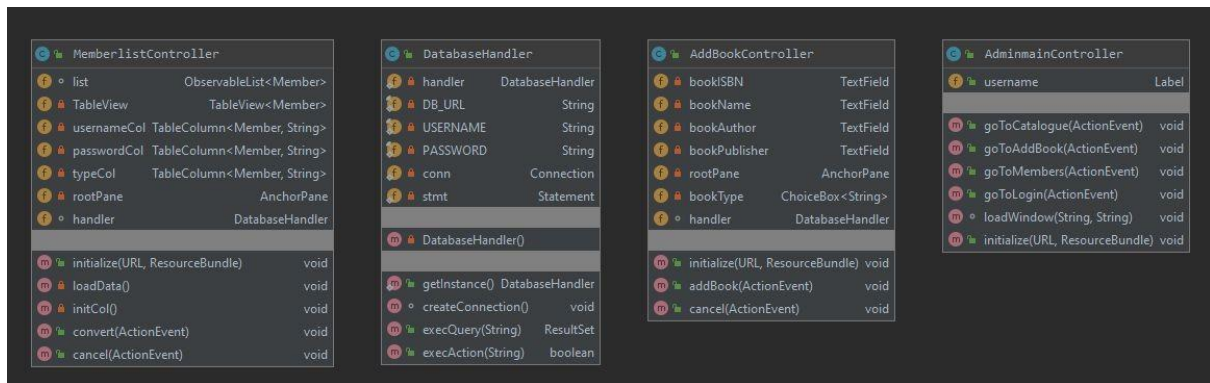
    void createConnection(){
        System.out.println("Connecting...");
        try {
            conn = DriverManager.getConnection(DB_URL, USERNAME, PASSWORD);
            System.out.println("Connected");
        } catch (Exception e){
            System.out.println(e.getMessage() + "-----CreateConnection-----");
        }
    }

    public ResultSet execQuery(String query){
        ResultSet result;
    }
}

```

CLASS DIAGRAMS





ASSUMPTION

Although I tried to develop this program more user-friendly, it has some limitations i have given in details:

To take book user first of all needs to check books from catalogue, find needed book, remember it's name and then go to search function and type it's name.

I did not add any button for deleting registered users from this program. If admin want to delete user, he or she needs to open database and manually delete user's raw from "users" table.

REFERENCE

OPENJAVAFX, openjfx.io

available at: <https://openjfx.io/javadoc/13/>

ORACLE, docs.oracle.com available

at:

https://docs.oracle.com/javafx/2/get_started/jfxpub-get_started.htm