LIBRARY MANAGEMENT SYSTEM WITH JAVA

**Technique paper – January 2020**

One author:



Abdybap uulu Damirbek

Ataturk-Alatoo University, new technologies faculty

[See profile in GitHub](https://github.com/Damirbek1997)

**Table of Contents**

INTRODUCTION . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

EXPLANATIONS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

*Dependencies* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

*Execution procedure* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

*Execution procedure* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

*Function* *(Issued Books)* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

*Function (View Books)* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

*Function (Sign Out)* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

ISSUE PART . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

*Function (Issue Book)* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

*Function (Submission Book)* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

ADMIN MENU . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

*Function (Add Book)* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

*Function (View Books)* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

*Function (Issued Books)* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

*Function (Delete Book)* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

*Function (View Users)* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

*Function (Sign Out)* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

DATABASE . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

UML DIAGRAMS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

CODE EXPLANATION . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

REFERENCES . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

1 | Page

**INTRODUCTION**

This assignment is based on developing an LIS (Library Management 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. All dates *(Users, Books, Issued Books)* stores in MySQL Database.

2 | Page

**EXPLANATIONS**

In this documentation, we have explained how to interact successfully with this LIS. We have explained here systematically so that it will surely help users to become more user friendly with it. Below are my explanations:

*Dependencies:*

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

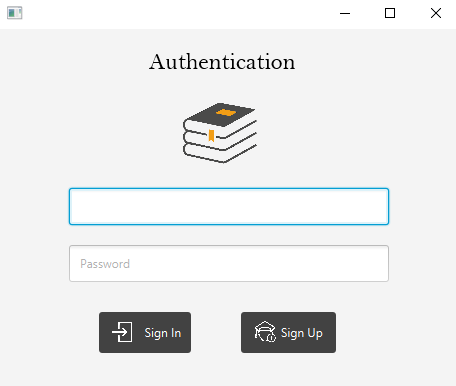
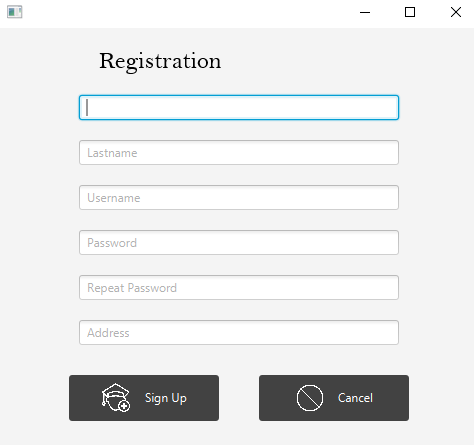
[JDK link](http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html)

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 *Intellij Idea* depending on the windows (32bit/64bit). [Click to download.](https://www.jetbrains.com/ru-ru/idea/download/#section=windows)

3 | Page

*Execution procedure:*

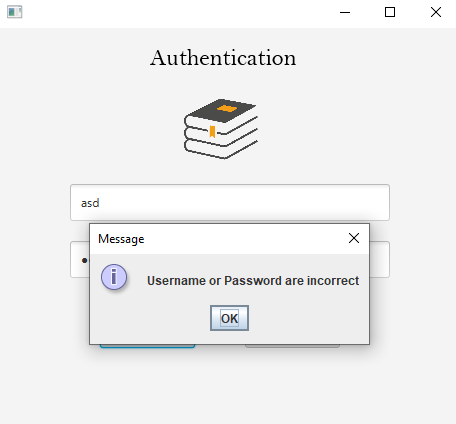
When user executes this program, it will open *login menu*. There is startup GUI (Graphical User Interface) of this program (*Sign In, Sign up*). User has to register in system to continue with work; after all, they can just logged in through *Sign In menu*.



*Picture 1* *Picture 2*

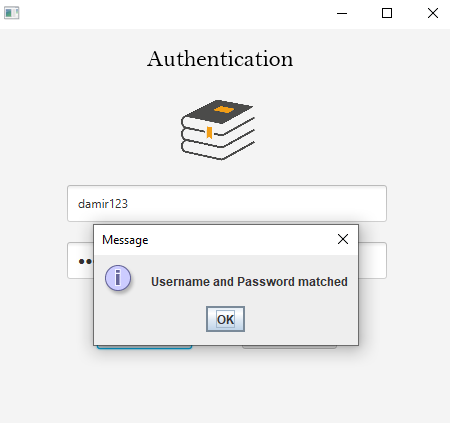
4 | Page

If *Username* or *Password* are wrong, it will pop up *Error* window



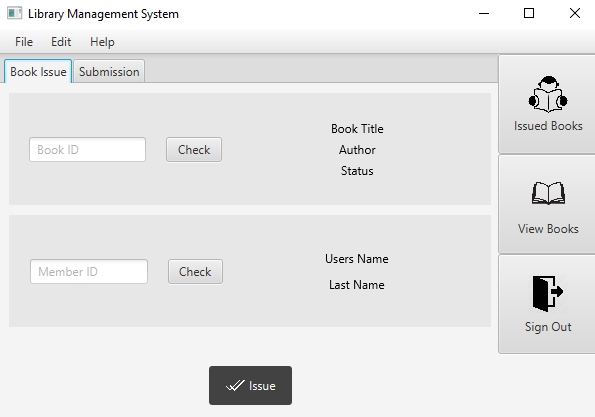
*Picture 3*

If all are correct, it will redirect into *main menu*



*Picture 4*

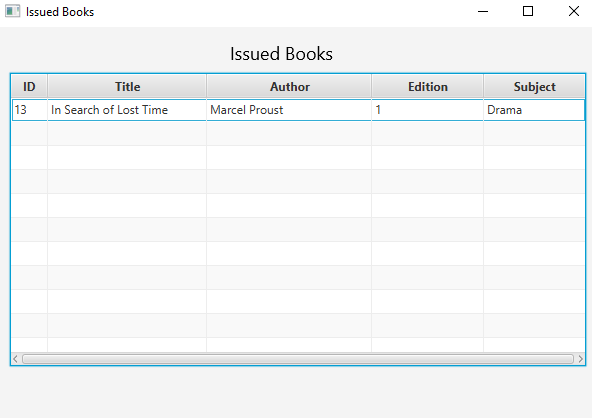
5 | Page



*Picture 5*

*Function (Issued Books):*

When users click *Issued Books* it will redirect into another window with list of books that he has taken, he can view all books that he has issued.

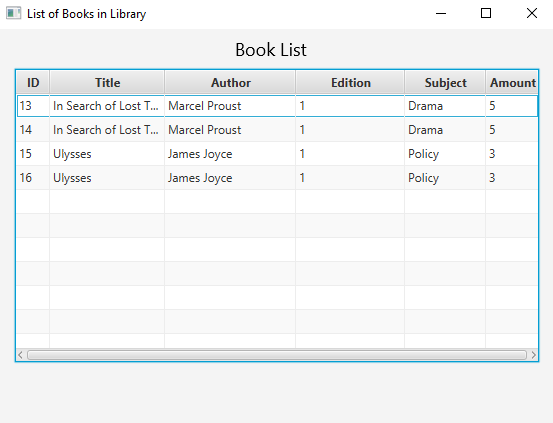


*Picture 6*

6 | Page

*Function (View Books):*

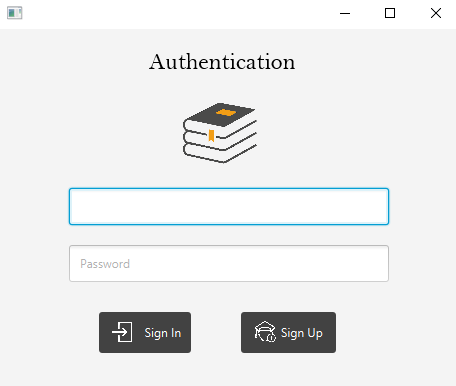
By clicking this button, user available to see *List* all of *Books* that *Library* Contains.



*Picture 7*

*Function (Sign out):*

This button uses to *Sign Out* to *Login Menu*.



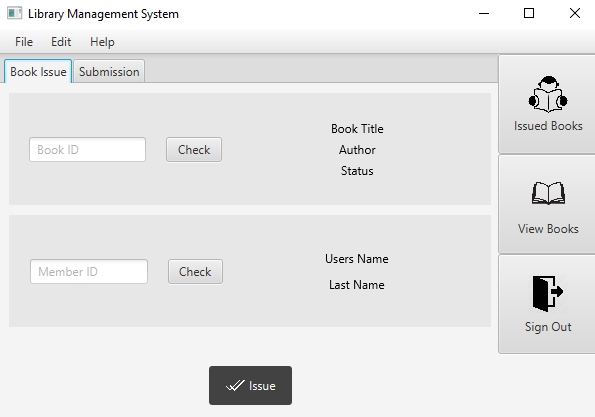
*Picture 8*

7 | Page

ISSUE PART

*Function (Issue Book):*

Firstly, user have to check which books is available in *View Books* and choose which one he want to *Issue*. After he has chosen it, he enters its *ID* into *Book ID Field*. He can also check if that is book, he had chosen by clicking *Button Check*. After *Clicking* in place *Book Title*, *Author*, *Status* would appear dates of book.

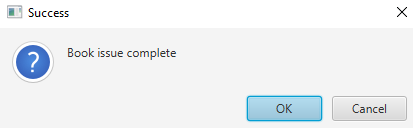
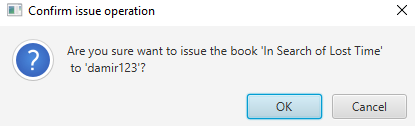
He also need to enter own *User ID* and *(is not necessary)* can check the correctness. As soon as *User* finishes checking the dates, he must press *Button Issue*.

*Picture 9*

8 | Page

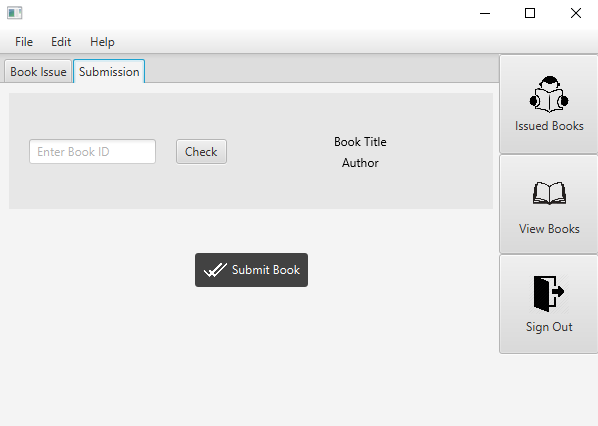
System will open *“Pop Up”* *Success Window* to notify that you have successfully *Issued Book* if it is *OK* and by clicking *Issue Button* it will ask about your choice

“Are you sure want to issue this book?”



*Picture 10* *Picture 11*

*Function (Submission Book)*:

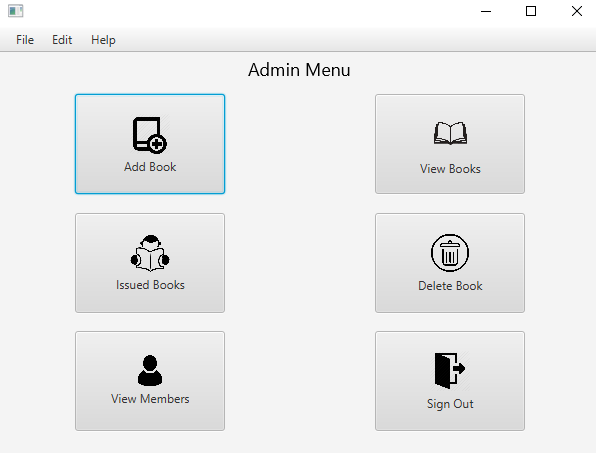
There is same procedure, *User* enters *Book ID (does not necessary)* and checks the dates, if it is correct he presses the *Button Submit Book*. There would *“Pop Up” Window* with *Success* message.

*Picture 12*

9 | Page

**ADMIN MENU**

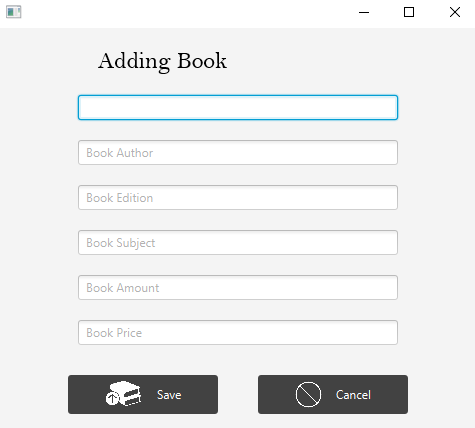
Only *Admin* can view *Admin Menu* and make changes here. There are six *Buttons*: *Add Book*, *View Books*, *Issued Books* (all of *Users*), *Delete Book*, *View Members* and *Sign Out*.



*Picture 13*

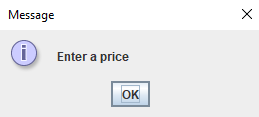
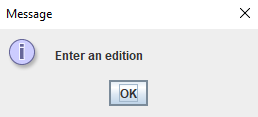
10 | Page

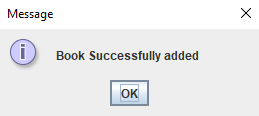
*Function (Add Book):*

By clicking *Button Add Book*, it will redirect into another *Window* where there will be six *Text Fields* (*Book Title*, *Book Author*, *Book Edition*, *Book Subject*, *Book Amount* and *Book Price*). *Admin* has to fill all of these *Fields* and press *Button Save*.

Picture 14

If there is any *Field*, which is not filled, it will *“Pop Up” Error Window*, otherwise *Success Window*.



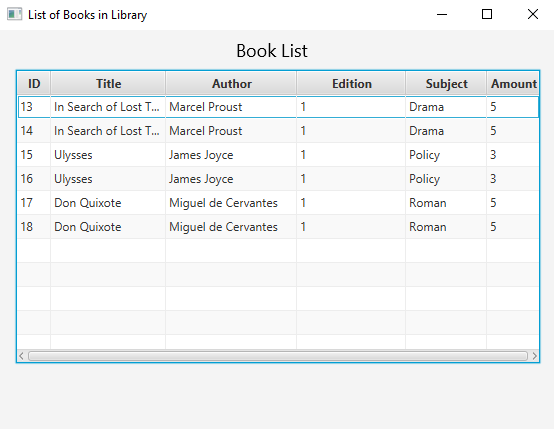
 *Picture 15*  *Picture 16*

*Picture 17*

11 | Page

*Function (View Books):*

By clicking *Button View Books*, it will open a window with *List of Books*. There *Admin* (also *User*) can see all of six field that he has filled before.



*Picture 18*

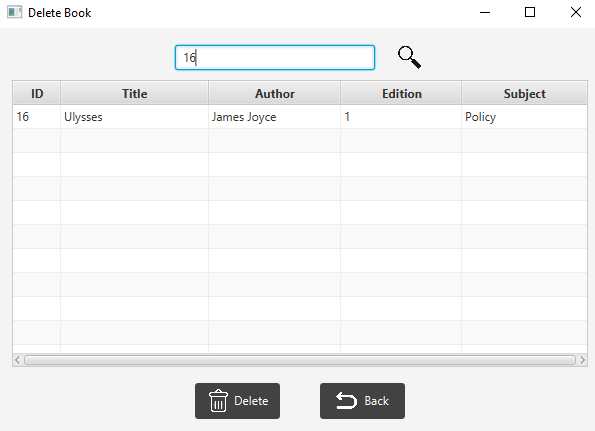
12 | Page

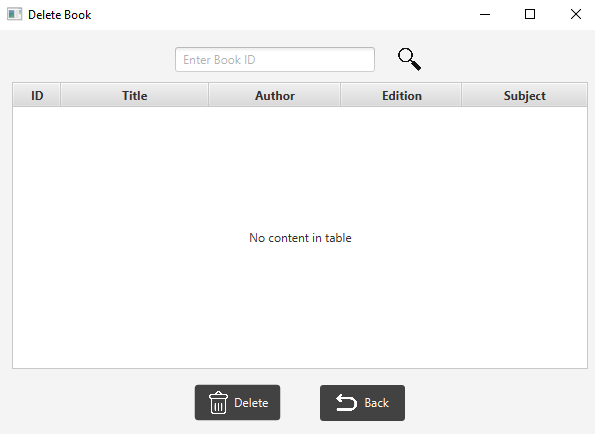
*Function (Issued Books):*

13 | Page

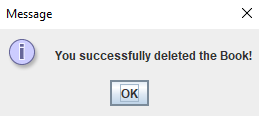
*Function (Delete Book):*

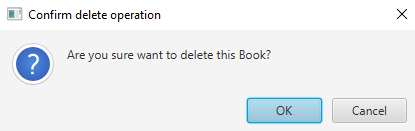
By clicking, *System* opens a window with *Book ID Field* and *Button Search* (to be sure in correctness). After filling *Book ID Field* and pressing *Button Search* it will show all dates corresponding to these *ID* (*Author, Edition, Title and etc.*). As soon, as finish checking *Admin* can easily *Delete Book* by clicking *Delete Button*. Also if *Admin* change mind and do not want to delete *Books*, he press the *Button Back* and goes back to *Admin Menu*.





*Picture 20* *Picture 21*

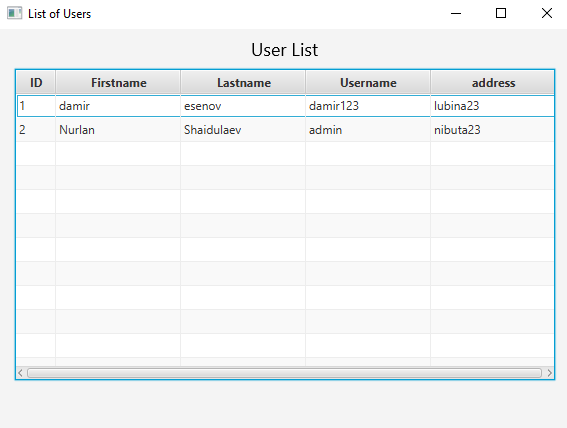
*System* also ask if you are sure want to delete this *Book*. If it is *OK*, it will show *Success Message.*

 *Picture 21* *Picture 22*

14 | Page

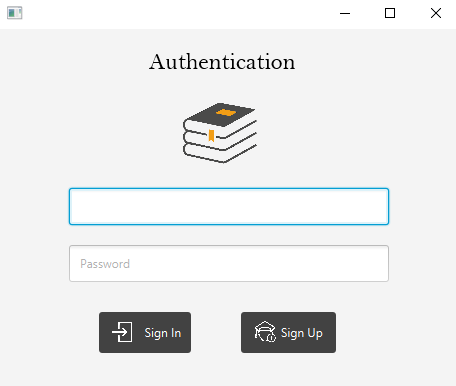
*Function (View Users):*

This window uses only to monitor which *Users* has registered.



*Picture 23*

*Function (Sign Out):*

By clicking this *Button,* it will redirect to *Login Menu*.

*Picture 24*

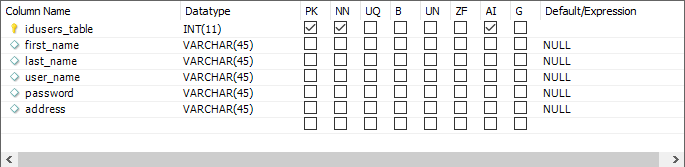
15 | Page

**MYSQL WORKBENCH**

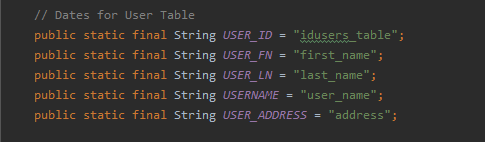
In this program, I have used **MySQL Database** (*link for download*):   
https://cdn.mysql.com//Downloads/MySQLInstaller/mysql-installer-community-8.0.18.0.msi

There are three *Tables* in *Library Database*: *book\_table*, *issued\_books\_table*, *users\_table*.

*Users\_table:*

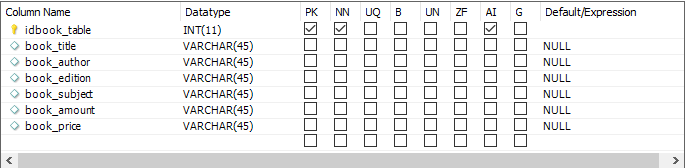


*Picture 25*



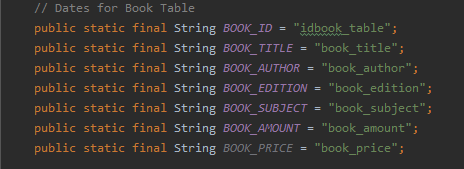
*Picture 26*

*Book\_table:*



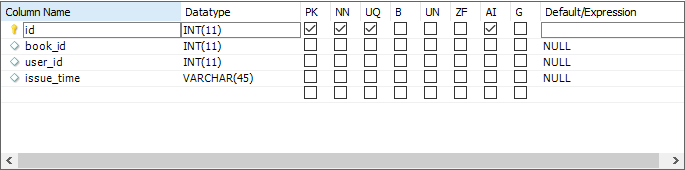
*Picture 27*

16 | Page

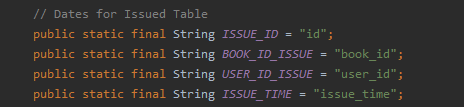


*Picture 28*

*Issued\_books\_table:*



*Picture 29*



*Picture 30*

16 | Page

**UML DIAGRAMS**

17 | Page

18 | Page

**CODE EXPLANATION**

There is explanation of code and what it does. I have explained here systematically so that it will surely help users to become more user friendly with it.

*Controller Folder:*

There are all controllers for all fxml files. They need to able the system work right.

*addBookController.java:*

**add\_book()** – this method responds for taking dates from *Text Fields* and insert it into *DB*.

**cancel\_adding()** – this method stands for cancel *Adding Book*, if *User* & *Admin* change they mind.

*bookListController.java:*

**initialize()** – this method responds for executing code as soon as fxml file opens. It contains two functions: **loadData()** and **showBooks().**

**loadData()** – this method stands for loading *Book* dates from *DB* and storing it into *ArrayList*.

**showBooks(Observable<Issue> list)** – this method has entering *ArrayList* with type of dates *Issue*. It stands for inserting dates from list into *Columns* in *Table*.

*deleteController.java:*

**backHandler()** – this method stands for going back into *Main Menu*.

**deleteHandler()** – this is main method, it deletes *Book* from *DB*.

**findBookAction()** – this method respond for searching *Book* in *DB* by his

*ID* and show it in *Table* for *Admin*.

*issuedController.java*:

**initialize()** – this method responds for executing code as soon as fxml file opens. It contains two functions: **loadData()** and **showBooks().**

**loadData()** – this method stands for loading *Issue* dates from *DB* and storing it into *ArrayList*.

19 | Page

**showBooks(Observable<Issue> list)** – this method has entering *ArrayList* with type of dates *Issue*. It stands for inserting dates from list into *Columns* in *Table*.

*issuedUserController.java:*

**initialize()** – this method responds for executing code as soon as fxml file opens. It contains two functions: **loadData()** and **showBooks().**

**loadData()** – this method stands for loading *Issued* dates from *DB* and storing it into *ArrayList*.

**showBooks(Observable<Issue> list)** – this method has entering *ArrayList* with type of dates *Issue*. It stands for inserting dates from list into *Columns* in *Table*.

*loginController.java:*

**signInHandler()** – this method responds for checking if *Username*

and *Password* matches in *DB*, if it does it will redirect into *Main Menu*.

**signUpHandler()** – this method stands for redirecting into *Sign Up Window*

for registering new *User*.

*mainAdminController.java:*

**addBookHandler()** – this method needs to redirect Admin into *Add Book*

*Window*.

**deleteHandler()** – this method stands for redirecting *Admin* into *Delete*

*Book* *Window*.

**issuedBooksHandler()** – this method responds for redirecting *Admin* into

*List* of *Issued* *Books* by *Users*.

**signOutHandler()** – this method stands for *Signing* *Out* and redirecting into

*Login* *Menu*.

**viewBooksHandler()** – this method responds for redirecting *Admin* into

*View* *Books* *Window*.

**viewMembersHandler()** – this method needs for redirecting *Admin* into *List*

of *Users*.

20 | Page

*mainController.java:*

**checkBookAction()** – this method needs for checking if *User* has chosen

right *Book*, it takes *ID* from *Text* *Field* and make search in *DB*, takes *Book*

*Title*, *Author* and *Status* and shows it to *User*.

**checkSubmitAction()** – this method responds for checking if *User* has

chosen right *Book* for *Submitting*.

**checkUserAction()** – this method needs for checking if *User* filled

correct *ID*.

**clearBookCache()**, **clearUserCache()** and **clearBookSubmitCache()** – needs

for clearing dates that was filling.

**issueNowHandler()** – responds for issuing *Book*.

**issuedBooksHandler()** – this method responds for redirecting *User* into

*List* of *Issued* *Books*.

**signOutHandler()** – this method stands for *Signing* *Out* and redirecting into

*Login* *Menu*.

**submissionAction()** – this method responds for submitting *Book* to *Library*. It

takes *Book* *ID* and make search in *DB* after matching it deletes from that

*Table*.

**viewBooksHandler()** – this method responds for redirecting *Admin* into

*View* *Books* *Window*.

*SceneHandler.java:*

**SceneHandler** – needs for opening new *Window* and closing old one.

*signUpController.java:*

**cancelHandler()** – stands for cancel *Sign* *Up* operation, it will redirect into

*Login* *Menu*.

**signUpHandler()** – responds for adding *User* into *DB*.

21 | Page

*userListController.java:*

**initialize()** – this method responds for executing code as soon as fxml file opens. It contains two functions: **loadData()** and **showUsers().**

**loadData()** – this method stands for loading *Users* dates from *DB* and storing it into *ArrayList*.

**showBooks(Observable<Issue> list)** – this method has entering *ArrayList* with type of dates *Users*. It stands for inserting dates from list into *Columns* in *Table*.

*DatabaseHandler Folder:*

**ConnectionDB.java** – this class needs to establish connection between *IntelIj* and *DB*.

**DatesDB.java** – this class responds for storing dates for three *tables*: *book\_table*, *users\_table*, *issued\_books\_table*.

References:

<https://bytescout.com/blog/20-important-sql-queries.html>

<https://cdn.mysql.com//Downloads/MySQLInstaller/mysql-installer-community-8.0.18.0.msi>

<https://www.w3schools.com/sql/sql_join_left.asp>

<https://dev.mysql.com/downloads/workbench/>

<https://material.io/resources/icons/?style=baseline>

<https://thegreatestbooks.org/>

22 | Page