JAVA SWING PROJE

RAPOR

ADI: Hala Jabban

STU NO: 2121221350

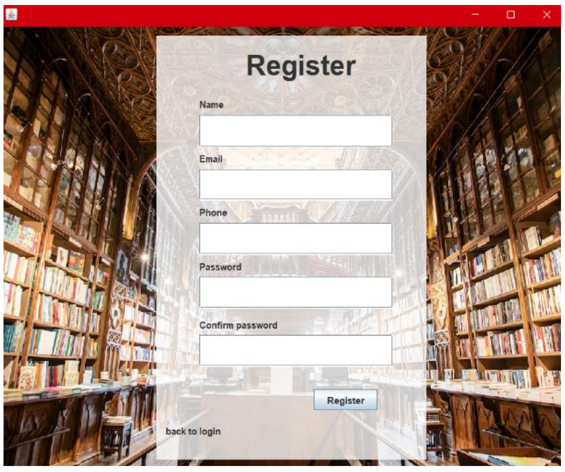
DATA: 21.12.2024 END: 28.12.2024

LIBRARY MANAGEMENT

The Library Management System is a system used to organize and manage library-related operations. It aims to facilitate the management of books, members, and procedures related to borrowing and returning. You have 14 days to return a book; otherwise, late fees will apply. Additionally, you can rent any book you want.

How to design a library Management:

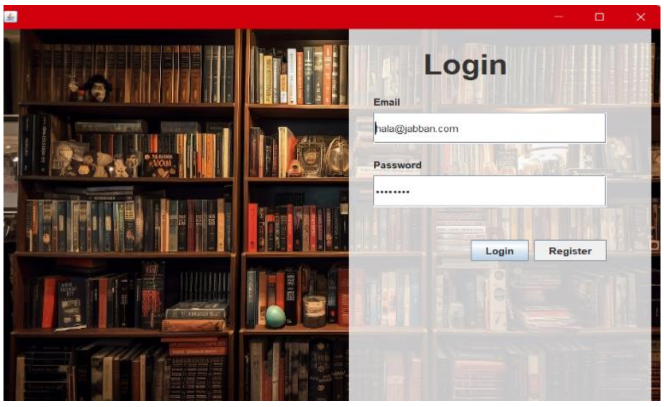
The code handles user registration by validating inputs using **Regex** to ensure the phone number is numeric. It also checks that all fields are required, the password is strong, and the passwords match. If the inputs are valid, a new user object is created and saved, then the fields are cleared, with the option to navigate back to the login page.



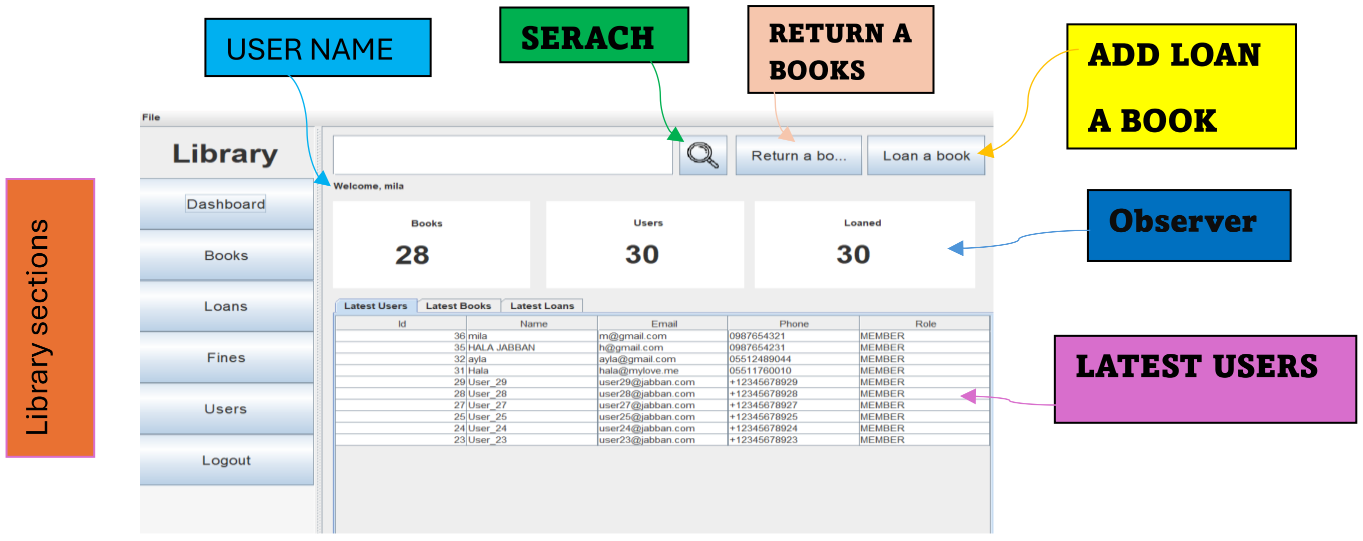
The code represents a login window using Java Swing.

It verifies the entered data when clicking "Login" and opens the dashboard if valid or shows an error if invalid. Clicking "Register" opens the registration window. The background has a transparent image for a better design.

Purpose: To manage the login process and navigate between windows based on user input.



Email: hala@jabban.com
Password: ••••••••



Welcome, mila

Books 28 | Users 30 | Loaned 30

Id	Name	Email	Phone	Role
36	mila	m@gmail.com	0987654321	MEMBER
35	HALA JABBAN	h@gmail.com	0987654231	MEMBER
32	ayla	ayla@gmail.com	05512489044	MEMBER
31	Hala	hala@mylove.me	05511760010	MEMBER
29	User_29	user29@jabban.com	+12345678929	MEMBER
28	User_28	user28@jabban.com	+12345678928	MEMBER
27	User_27	user27@jabban.com	+12345678927	MEMBER
25	User_25	user25@jabban.com	+12345678925	MEMBER
24	User_24	user24@jabban.com	+12345678924	MEMBER
23	User_23	user23@jabban.com	+12345678923	MEMBER

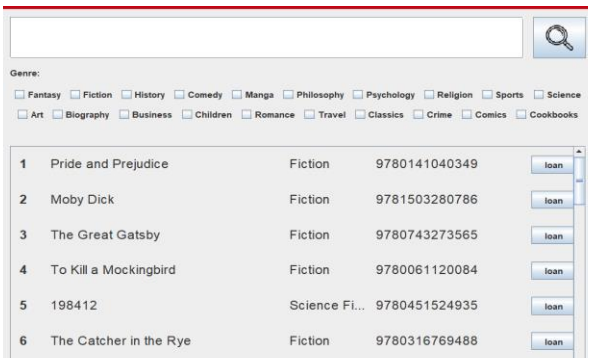
1)The part of the code you sent is from the DashboardFrame class in Java using the **Swing** library to build user interfaces. In the DashboardFrame() constructor, it checks whether the user is logged in using the Helper.isLoggedIn() method. If the result shows that the user is not logged in (or there is no active session), the current window is closed using this.dispose() and the **Login** window (LoginFrame) is opened.

The code represents that uses the Observer design pattern.

Observer (interface): It is an interface that contains the update() method, which must be implemented by the observers. The observer receives the updated value in this method.

2)Search Window (SearchFrame):
A sub-window for searching books, sets the initial text,

and closes without affecting the main application.



1	Pride and Prejudice	Fiction	9780141040349	loan
2	Moby Dick	Fiction	9781503280786	loan
3	The Great Gatsby	Fiction	9780743273565	loan
4	To Kill a Mockingbird	Fiction	9780061120084	loan
5	198412	Science Fi...	9780451524935	loan
6	The Catcher in the Rye	Fiction	9780316769488	loan

Graphical Components:

Text Field Search Button Check Boxes Results Container

Swing Usage:

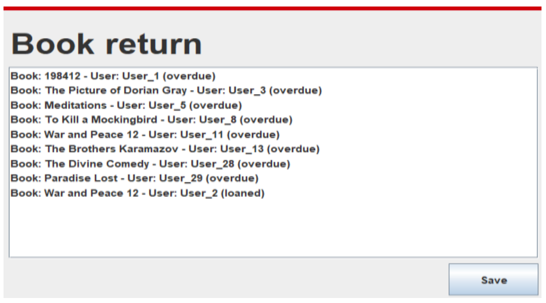
JTextField: For text input.

JButton: For the search button.

JCheckBox: For genres.

JPanel: For layout

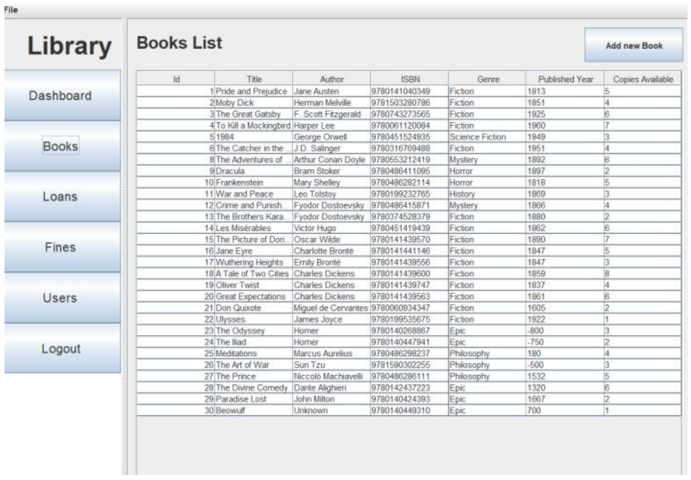
3) **BOOK RETURN**



Book return

Book: 198412 - User: User_1 (overdue)
Book: The Picture of Dorian Gray - User: User_3 (overdue)
Book: Meditations - User: User_5 (overdue)
Book: To Kill a Mockingbird - User: User_8 (overdue)
Book: War and Peace 12 - User: User_11 (overdue)
Book: The Brothers Karamazov - User: User_13 (overdue)
Book: The Divine Comedy - User: User_28 (overdue)
Book: Paradise Lost - User: User_29 (overdue)
Book: War and Peace 12 - User: User_2 (loaned)

1)The code represents a panel that displays a list of books using Java Swing. It includes:



Books List

Id	Title	Author	ISBN	Genre	Published Year	Copies Available
1	Pride and Prejudice	Jane Austen	9780141040349	Fiction	1813	5
2	Moby Dick	Herman Melville	9781503280786	Fiction	1851	4
3	The Great Gatsby	F. Scott Fitzgerald	9780743273565	Fiction	1925	6
4	To Kill a Mockingbird	Harper Lee	9780061120084	Fiction	1960	7
5	1984	George Orwell	9780451524935	Science Fiction	1949	3
6	The Catcher in the ...	J.D. Salinger	9780316769488	Fiction	1951	4
8	The Adventures of ...	Arthur Conan Doyle	9780553212419	Mystery	1892	6
9	Dracula	Bram Stoker	9780486411095	Horror	1897	2
10	Frankenstein	Mary Shelley	9780486282114	Horror	1818	5
11	War and Peace	Leo Tolstoy	9780199232765	History	1869	3
12	Crime and Punish...	Fyodor Dostoevsky	9780486415871	Mystery	1866	4
13	The Brothers Kara...	Fyodor Dostoevsky	9780374528379	Fiction	1880	2
14	Les Misérables	Victor Hugo	9780451419439	Fiction	1862	6
15	The Picture of Dori...	Oscar Wilde	9780141439570	Fiction	1890	7
16	Jane Eyre	Charlotte Brontë	9780141441146	Fiction	1847	5
17	Wuthering Heights	Emily Brontë	9780141439556	Fiction	1847	3
18	A Tale of Two Cities	Charles Dickens	9780141439600	Fiction	1859	8
19	Oliver Twist	Charles Dickens	9780141439747	Fiction	1837	4
20	Great Expectations	Charles Dickens	9780141439563	Fiction	1861	6
21	Don Quixote	Miguel de Cervantes	9780060934347	Fiction	1605	2
22	Ulysses	James Joyce	9780199535675	Fiction	1922	1
23	The Odyssey	Homer	9780140268867	Epic	-800	3
24	The Iliad	Homer	9780140447941	Epic	-750	2
25	Meditations	Marcus Aurelius	9780486298237	Philosophy	180	4
26	The Art of War	Sun Tzu	9781590302255	Philosophy	-500	3
27	The Prince	Niccolò Machiavelli	9780486286111	Philosophy	1532	5
28	The Divine Comedy	Dante Alighieri	9780142437223	Epic	1320	6
29	Paradise Lost	John Milton	9780140424393	Epic	1667	2
30	Beowulf	Unknown	9780140449310	Epic	700	1

JTable: Displays the list of books.

JComboBox: Allows editing book categories.

Context Menu: Provides an option to delete a book when right-clicking on a row.

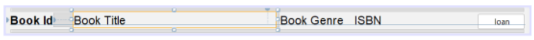
Event Handling: Opens a new window to add a book when the "Add New Book" button is clicked.

The panel updates the book list using the **FrameObserver** pattern.

2) SearchListItem displays book details such as the ID, title, genre, and ISBN.

When the "loan" button is clicked, a new window **NewLoanFrame** opens to start the book loan process.
The components are arranged using **GroupLayout** to display them horizontally and neatly.

Components used in the code:



JLabel: idLabel: Displays the book ID.

bookTitleLabel: Displays the book title.

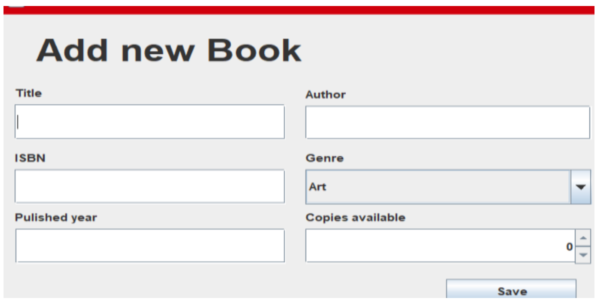
bookGenreLabel: Displays the book genre.

bookGenreLabel1: Displays the ISBN number.

JButton: loanButton: Button to initiate the loan process.

GroupLayout: Used to arrange the components horizontally within the interface.

3) Components used in the frame:



JLabel: Used to display text labels (e.g., "Title", "Author", "ISBN", "Genre", "Published year", "Copies available").

JTextField: titleInput, authorInput, isbnInput, publishedYearInput: For entering book details such as title, author, ISBN, and publication year.

JComboBox: genreCombo: For selecting the book genre from a list of options.

JSpinner:copiesInput: For entering the number of available copies.

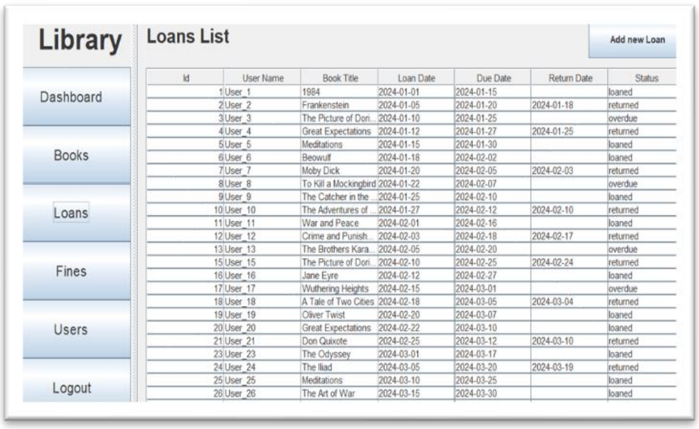
JButton:saveButton: A button to save the new book and add it to the system.

JOptionPane: For displaying messages to the user such as "Please fill all fields" or "Book added successfully".

GroupLayout: For arranging the components neatly in the interface.

FrameObserver: Used for the Observer pattern to notify observers when a new book is added

The code represents a loan management panel using Java Swing. It includes:



Loans List

Id	User Name	Book Title	Loan Date	Due Date	Return Date	Status
1	User_1	1984	2024-01-01	2024-01-15		loaned
2	User_2	Frankenstein	2024-01-05	2024-01-20	2024-01-18	returned
3	User_3	The Picture of Dori...	2024-01-10	2024-01-25		overdue
4	User_4	Great Expectations	2024-01-12	2024-01-27	2024-01-25	returned
5	User_5	Meditations	2024-01-15	2024-01-30		loaned
6	User_6	Beowulf	2024-01-18	2024-02-02		loaned
7	User_7	Moby Dick	2024-01-20	2024-02-05	2024-02-03	returned
8	User_8	To Kill a Mockingbird	2024-01-22	2024-02-07		overdue
9	User_9	The Catcher in the ...	2024-01-25	2024-02-10		loaned
10	User_10	The Adventures of ...	2024-01-27	2024-02-12	2024-02-10	returned
11	User_11	War and Peace	2024-02-01	2024-02-16		loaned
12	User_12	Crime and Punish...	2024-02-03	2024-02-18	2024-02-17	returned
13	User_13	The Brothers Kara...	2024-02-05	2024-02-20		overdue
15	User_15	The Picture of Dori...	2024-02-10	2024-02-25	2024-02-24	returned
16	User_16	Jane Eyre	2024-02-12	2024-02-27		loaned
17	User_17	Wuthering Heights	2024-02-15	2024-03-01		overdue
18	User_18	A Tale of Two Cities	2024-02-18	2024-03-05	2024-03-04	returned
19	User_19	Oliver Twist	2024-02-20	2024-03-07		loaned
20	User_20	Great Expectations	2024-02-22	2024-03-10		loaned
21	User_21	Don Quixote	2024-02-25	2024-03-12	2024-03-10	returned
23	User_23	The Odyssey	2024-03-01	2024-03-17		loaned
24	User_24	The Iliad	2024-03-05	2024-03-20	2024-03-19	returned
25	User_25	Meditations	2024-03-10	2024-03-25		loaned
26	User_26	The Art of War	2024-03-15	2024-03-30		loaned

JTable: To display the list of loans.

JComboBox: To select the loan status such as "loaned", "returned", "overdue".

Context menu: Provides an option to delete a loan when right-clicking on a row.

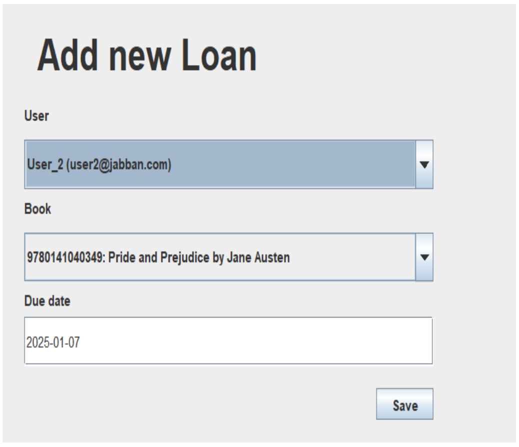
Event handling: When the "Add new loan" button is clicked, a new window for adding a loan opens.

FrameObserver: The loan list is updated when any modification occurs.

The main function is to **display and manage loans**, updating the list as needed.

The code creates a window for adding a new loan in the

application using Java Swing. When the window is opened:



Add new Loan

User

User_2 (user2@jabban.com)

Book

9780141040349: Pride and Prejudice by Jane Austen

Due date

2025-01-07

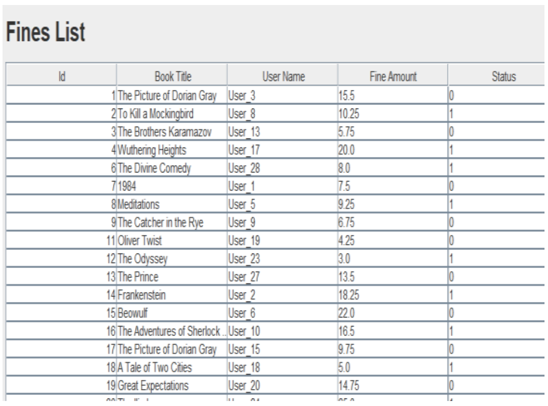
1. Users with the role "member" and available books are loaded.
2. A default due date is set to 14 days from the current date.
3. When the "Save" button is clicked:

It checks if all fields are filled.

A loan is created and saved to the database.

Observers are notified of the update.

4. The window is closed after the loan is saved.



Fines List

Id	Book Title	User Name	Fine Amount	Status
1	The Picture of Dorian Gray	User_3	15.5	0
2	To Kill a Mockingbird	User_8	10.25	1
3	The Brothers Karamazov	User_13	5.75	0
4	Wuthering Heights	User_17	20.0	1
6	The Divine Comedy	User_28	8.0	1
7	1984	User_1	7.5	0
8	Meditations	User_5	9.25	1
9	The Catcher in the Rye	User_9	6.75	0
11	Oliver Twist	User_19	4.25	0
12	The Odyssey	User_23	3.0	1
13	The Prince	User_27	13.5	0
14	Frankenstein	User_2	18.25	1
15	Beowulf	User_6	22.0	0
16	The Adventures of Sherlock ...	User_10	16.5	1
17	The Picture of Dorian Gray	User_15	9.75	0
18	A Tale of Two Cities	User_18	5.0	1
19	Great Expectations	User_20	14.75	0

JTable: To display data in a table containing fine information.

JPopupMenu: To create a context menu that appears when the right mouse button is clicked on the table.

JMenuItem: A menu item in the context menu, used here for deleting a fine.

JOptionPane: To display confirmation messages (such as confirming the deletion of a fine) or error messages.

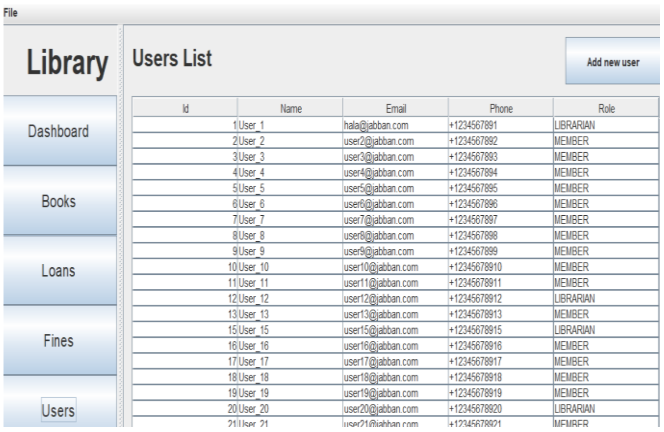
MouseAdapter: To listen for mouse events, especially to detect clicks for opening the context menu.

FrameObserver: An interface implemented in the code, allowing the fine data to be updated when changes occur.

List: To store the fines loaded from the database.

FineTableModel: The data model used to bind data to the JTable.

Swing Layouts: Layout management using GroupLayout and JScrollPane for displaying tables in an organized manner.



Users List

Id	Name	Email	Phone	Role
1	User_1	hala@jabban.com	+1234567891	LIBRARIAN
2	User_2	user2@jabban.com	+1234567892	MEMBER
3	User_3	user3@jabban.com	+1234567893	MEMBER
4	User_4	user4@jabban.com	+1234567894	MEMBER
5	User_5	user5@jabban.com	+1234567895	MEMBER
6	User_6	user6@jabban.com	+1234567896	MEMBER
7	User_7	user7@jabban.com	+1234567897	MEMBER
8	User_8	user8@jabban.com	+1234567898	MEMBER
9	User_9	user9@jabban.com	+1234567899	MEMBER
10	User_10	user10@jabban.com	+12345678910	MEMBER
11	User_11	user11@jabban.com	+12345678911	MEMBER
12	User_12	user12@jabban.com	+12345678912	LIBRARIAN
13	User_13	user13@jabban.com	+12345678913	MEMBER
15	User_15	user15@jabban.com	+12345678915	LIBRARIAN
16	User_16	user16@jabban.com	+12345678916	MEMBER
17	User_17	user17@jabban.com	+12345678917	MEMBER
18	User_18	user18@jabban.com	+12345678918	MEMBER
19	User_19	user19@jabban.com	+12345678919	MEMBER
20	User_20	user20@jabban.com	+12345678920	LIBRARIAN
21	User_21	user21@jabban.com	+12345678921	MEMBER

In the code, the following elements are used:

JTable: To display the list of users in a table.

JComboBox: To edit the user role (librarian or member) in the "Role" column of the table.

JScrollPane: To provide scrolling when displaying the table.

JButton: A button to add a new user.

DefaultCellEditor: To edit the "Role" column using a JComboBox.

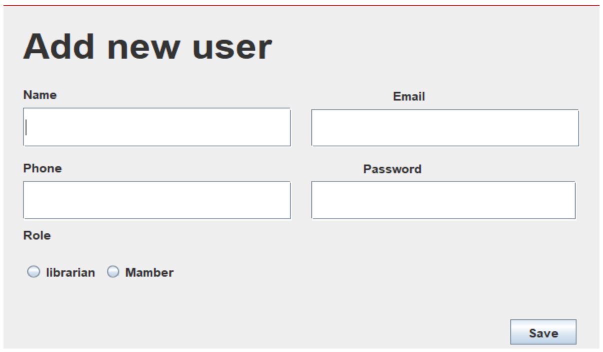
Event Handling: To open a new window to add a user when the button is clicked.

Observer Pattern: To observe updates from the "NewUserFrame" window and notify the UsersPanel.

UserTableModel: To bind the data to the table and update it automatically.

These components are used to create an interactive interface for managing users.

Explanation of the Code:



The NewUserFrame class is a GUI form that allows the creation of a new user.

Components used:

JTextField:

nameInput, emailInput, userNameInput, passwordInput: Text fields for entering user details such as name, email, phone, and password.

JRadioButton: **librarianRadio, mamberRadio**: Radio buttons to select the user's role (librarian or member).

But tonGroup: **roleRadioGroup**: Groups the radio buttons so that only one role can be selected at a time.

JButton: **saveButton**: Button to save the new user and register them.

JOptionPane: For displaying error or success messages (e.g., "All fields are required", "User registered successfully").

Observer Pattern: **FrameObserver**: This is used to notify other parts of the system when a new user is registered.

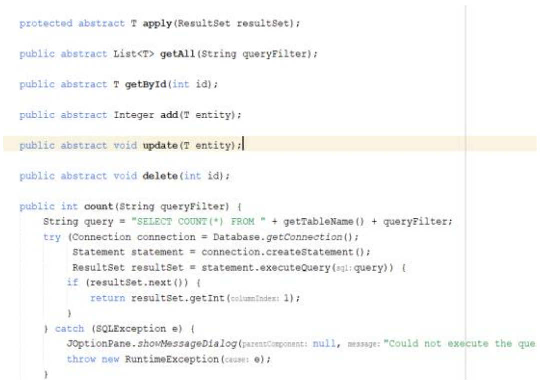
Functionality:

- The form collects user information (name, email, phone, password, and role).
- It performs validation on the inputs (e.g., checking if fields are empty, validating the password format, and ensuring the phone number is numeric).
- After validation, it creates a new User object, saves it, and notifies observers of the new user.
- The form is then closed after successful registration.

In the provided code, several components and techniques are used to manage interaction with the database:

1. JDBC (Java Database Connectivity)**:

- Connection Establishes a connection to the database using `Database.getConnection()`. Statement and PreparedStatement: Used for executing SQL queries.ResultSet: Captures the results from SQL queries.



```
protected abstract T apply(ResultSet resultSet);

public abstract List<T> getAll(String queryFilter);

public abstract T getById(int id);

public abstract Integer add(T entity);

public abstract void update(T entity);

public abstract void delete(int id);

public int count(String queryFilter) {
    String query = "SELECT COUNT(*) FROM " + getTableName() + queryFilter;
    try (Connection connection = Database.getConnection();
         Statement statement = connection.createStatement();
         ResultSet resultSet = statement.executeQuery(sql: query)) {
        if (resultSet.next()) {
            return resultSet.getInt(columnIndex: 1);
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "Could not execute the que
        throw new RuntimeException(cause: e);
    }
```

2. Queries: SELECT is used to retrieve data.

INSERT, UPDATE, and DELETE are used to modify data in the database.

3. Function<ResultSet, T>: This technique passes a function to map data retrieved from the database and convert it into an object of type `T`.

4. Error Handling:JOptionPane is used to display error messages to the user.

5. Transaction Handling: connection.setAutoCommit(false) is used to disable auto-commit and execute multiple queries within a single transaction. If any query fails, changes are rolled back using connection.rollback()

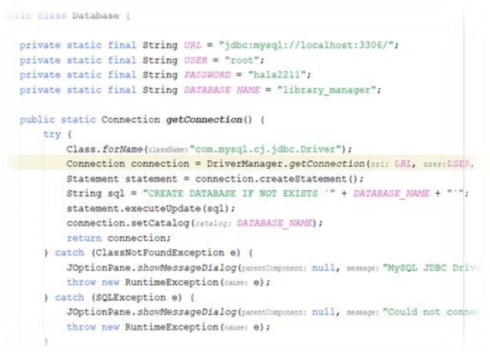
6. Prepared Statements: PreparedStatementis used to execute queries that require input data, helping to protect against SQL injection attacks.

These components ensure secure and efficient communication with the database.

The Database class manages the connection to a MySQL database and handles basic operations:

1. **getConnection()**: Establishes and selects the database connection.
2. **closeConnection()**: Closes the database connection.
3. **instantiateDatabase()**: Initializes the database setup.
4. **createDatabase()**: Reads and executes an SQL script to create tables.
5. **executeSqlScript()**: Executes an SQL script in the database.

It handles driver loading, connection management, and executing necessary setup scripts for the database.



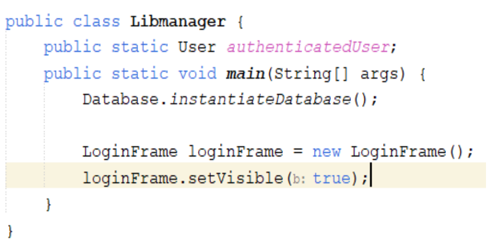
```
lic class Database {

    private static final String URL = "jdbc:mysql://localhost:3306/";
    private static final String USER = "root";
    private static final String PASSWORD = "hala2211";
    private static final String DATABASE_NAME = "library_manager";

    public static Connection getConnection() {
        try {
            Class.forName(className:"com.mysql.cj.jdbc.Driver");
            Connection connection = DriverManager.getConnection(url: URL, user:USER,
            Statement statement = connection.createStatement();
            String sql = "CREATE DATABASE IF NOT EXISTS `" + DATABASE_NAME + "`";
            statement.executeUpdate(sql);
            connection.setCatalog(catalog: DATABASE_NAME);
            return connection;
        } catch (ClassNotFoundException e) {
            JOptionPane.showMessageDialog(parentComponent: null, message: "MySQL JDBC Driv
            throw new RuntimeException(cause: e);
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(parentComponent: null, message: "Could not conne
            throw new RuntimeException(cause: e);
        }
```

The code does the following:

Imports: Imports **LoginFrame** for the login interface. Imports **User** to represent the user. Imports **Database** to set up the database.



```
public class Libmanager {
    public static User authenticatedUser;
    public static void main(String[] args) {
        Database.instantiateDatabase();

        LoginFrame loginFrame = new LoginFrame();
        loginFrame.setVisible(b: true);
    }
}
```

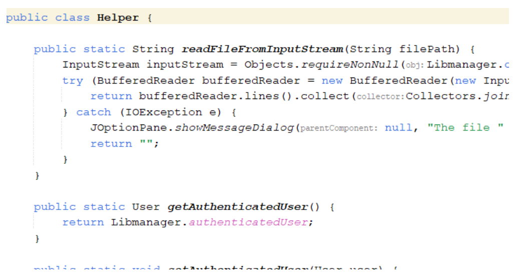
Libmanager Class: **authenticatedUser**: Holds the authenticated user.

main Method:Initializes the database using **Database.instantiateDatabase()**.

Displays the **LoginFrame** for user login.This is the main program for the library management application.

The Helper class contains utility methods for:

1. **readFileFromInputStream**: Reads the content of a file from a given path.
2. **getAuthenticatedUser and setAuthenticatedUser**: Get and set the authenticated user.
3. **isLoggedIn**: Checks if a user is logged in.
4. **validatePassword**: Validates a password using a regular expression.



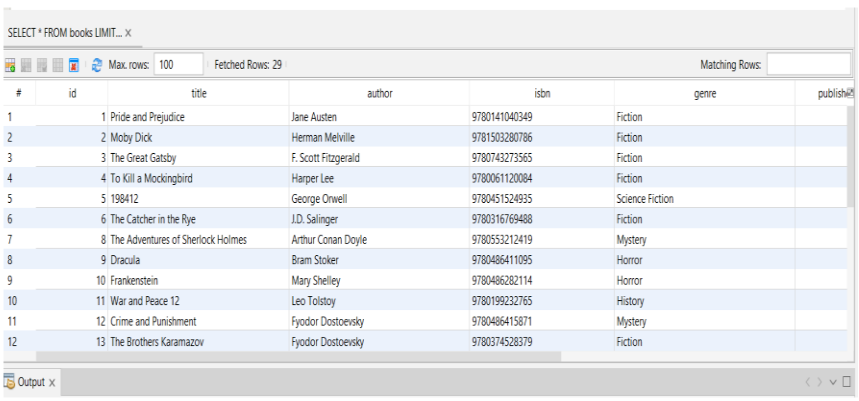
```
public class Helper {

    public static String readFileFromInputStream(String filePath) {
        InputStream inputStream = Objects.requireNonNull(obj: Libmanager.c
        try (BufferedReader bufferedReader = new BufferedReader(new Inpu
            return bufferedReader.lines().collect(collector:Collectors.joi
        } catch (IOException e) {
            JOptionPane.showMessageDialog(parentComponent: null, "The file "
            return "";
        }
    }

    public static User getAuthenticatedUser() {
        return Libmanager.authenticatedUser;
    }
```

5. **getCurrentDate**: Returns the current date and time.
6. **isNumeric**: Checks if a string contains only numeric characters.
7. **DateFromString and DateToString**: Converts between a string and a Date object.
8. **getFutureDate**: Returns a date that is a specified number of days from the current date.

REASULT DATABASE:

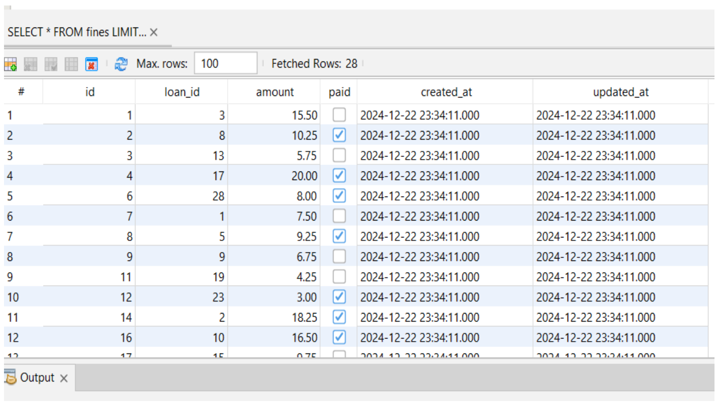


SELECT * FROM books LIMIT... ×

Max. rows: 100 | Fetched Rows: 29 | Matching Rows:

#	id	title	author	isbn	genre	publish
1	1	Pride and Prejudice	Jane Austen	9780141040349	Fiction	
2	2	Moby Dick	Herman Melville	9781503280786	Fiction	
3	3	The Great Gatsby	F. Scott Fitzgerald	9780743273565	Fiction	
4	4	To Kill a Mockingbird	Harper Lee	9780061120084	Fiction	
5	5	198412	George Orwell	9780451524935	Science Fiction	
6	6	The Catcher in the Rye	J.D. Salinger	9780316769488	Fiction	
7	8	The Adventures of Sherlock Holmes	Arthur Conan Doyle	9780553212419	Mystery	
8	9	Dracula	Bram Stoker	9780486411095	Horror	
9	10	Frankenstein	Mary Shelley	9780486282114	Horror	
10	11	War and Peace 12	Leo Tolstoy	9780199232765	History	
11	12	Crime and Punishment	Fyodor Dostoevsky	9780486415871	Mystery	
12	13	The Brothers Karamazov	Fyodor Dostoevsky	9780374528379	Fiction	

Output ×

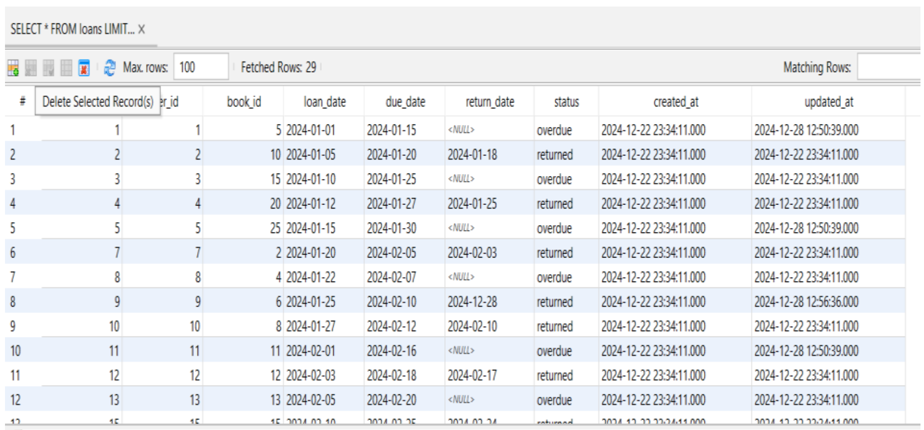


SELECT * FROM fines LIMIT... ×

Max. rows: 100 | Fetched Rows: 28

#	id	loan_id	amount	paid	created_at	updated_at
1	1	3	15.50	☐	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
2	2	8	10.25	✓	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
3	3	13	5.75	☐	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
4	4	17	20.00	✓	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
5	6	28	8.00	✓	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
6	7	1	7.50	☐	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
7	8	5	9.25	✓	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
8	9	9	6.75	☐	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
9	11	19	4.25	☐	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
10	12	23	3.00	✓	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
11	14	2	18.25	✓	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
12	16	10	16.50	✓	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000

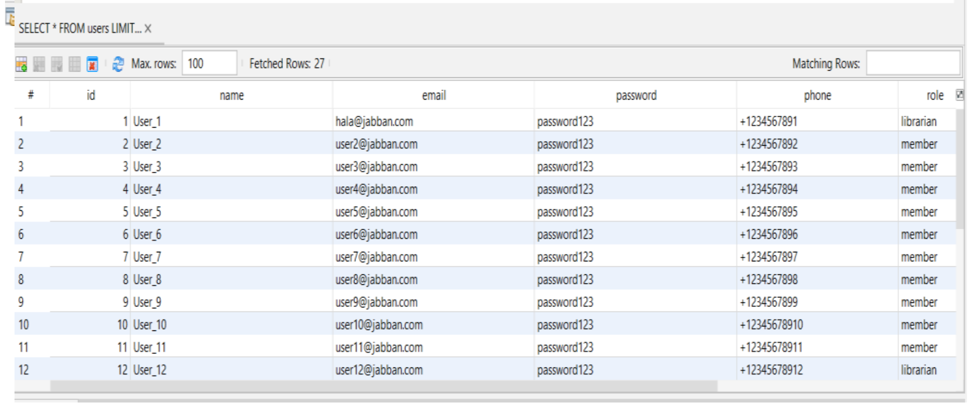
Output ×



SELECT * FROM loans LIMIT... ×

Max. rows: 100 | Fetched Rows: 29 | Matching Rows:

#	Delete Selected Record(s)	er_id	book_id	loan_date	due_date	return_date	status	created_at	updated_at
1	1	1	5	2024-01-01	2024-01-15	<NULL>	overdue	2024-12-22 23:34:11.000	2024-12-28 12:50:39.000
2	2	2	10	2024-01-05	2024-01-20	2024-01-18	returned	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
3	3	3	15	2024-01-10	2024-01-25	<NULL>	overdue	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
4	4	4	20	2024-01-12	2024-01-27	2024-01-25	returned	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
5	5	5	25	2024-01-15	2024-01-30	<NULL>	overdue	2024-12-22 23:34:11.000	2024-12-28 12:50:39.000
6	7	7	2	2024-01-20	2024-02-05	2024-02-03	returned	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
7	8	8	4	2024-01-22	2024-02-07	<NULL>	overdue	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
8	9	9	6	2024-01-25	2024-02-10	2024-12-28	returned	2024-12-22 23:34:11.000	2024-12-28 12:56:36.000
9	10	10	8	2024-01-27	2024-02-12	2024-02-10	returned	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
10	11	11	11	2024-02-01	2024-02-16	<NULL>	overdue	2024-12-22 23:34:11.000	2024-12-28 12:50:39.000
11	12	12	12	2024-02-03	2024-02-18	2024-02-17	returned	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000
12	13	13	13	2024-02-05	2024-02-20	<NULL>	overdue	2024-12-22 23:34:11.000	2024-12-22 23:34:11.000



SELECT * FROM users LIMIT... ×

Max. rows: 100 | Fetched Rows: 27 | Matching Rows:

#	id	name	email	password	phone	role
1	1	User_1	hala@jabban.com	password123	+1234567891	librarian
2	2	User_2	user2@jabban.com	password123	+1234567892	member
3	3	User_3	user3@jabban.com	password123	+1234567893	member
4	4	User_4	user4@jabban.com	password123	+1234567894	member
5	5	User_5	user5@jabban.com	password123	+1234567895	member
6	6	User_6	user6@jabban.com	password123	+1234567896	member
7	7	User_7	user7@jabban.com	password123	+1234567897	member
8	8	User_8	user8@jabban.com	password123	+1234567898	member
9	9	User_9	user9@jabban.com	password123	+1234567899	member
10	10	User_10	user10@jabban.com	password123	+12345678910	member
11	11	User_11	user11@jabban.com	password123	+12345678911	member
12	12	User_12	user12@jabban.com	password123	+12345678912	librarian

✓ **If you want to see my prograimg,**

I have put a short video you can watch

short video about library.mp4