Informatics Practices Project File

Mohammad Muzammil Khan

XII-B1

Index

Index

Certificate

This is to certify that this project has been made by me, **Mohammad Muzammil Khan** of class **XII-B1**, under the guidance of my Informatics Practices teacher **Ms. Sheetal** and has been completed successfully.

Mohammad Muzammil Khan

Acknowledgment

I would like to express my special **thanks** to my teacher **Ms. Sheetal** as well as our principal **Ms. Parul Tyagi** who gave me the golden opportunity to do this wonderful project on the topic of **Database connectivity with Java programming Language and MySql**, which also helped me in doing a lot of research and I came to know so many new things. I am really thankful to them.

Secondly I would also like to thank my **parents and friends** who helped me a lot in finalizing this project within the limited time frame.

Requirements

- Java IDE NetBeans
- MySQL
- Knowledge of MySQL (Create Database, Create Table, Insert, Select, Update, Delete & Drop etc.).
- Basic Knowledge of Designing Java UI Frame.

Project Description

This project is a **Library Management Software**, named **Libri**. It allows a user to Login into their account and have control over their library like:

- Add, Remove and Update Books
- View available books and issued books
- Add, Remove and Update Members
- View Members
- Issue Books to members
- Etc.

Project Structure

Libri has a Login page ("**Login.java**") which shows up as soon as the application is launched. After entering the correct details, a new frame ("**MainFrame.java**") is loaded where user can use **Libri** as they like.

MainFrame has all of the controls, including, add/remove/update books/members and also shows all issued/available books and issue books to members within **two tabs** (Books and Members).

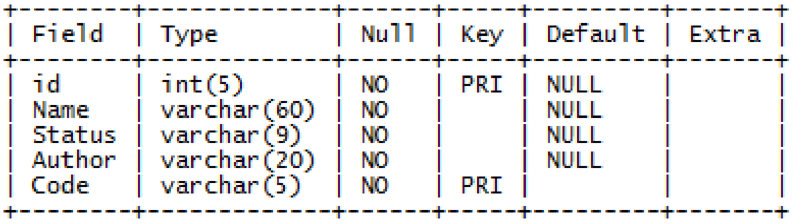
Each button on **MainFrame** opens another frame which is dedicated to the function described on button. **For Example**; if user clicks "**Add Book**" button, a new frame will load to add book to the data base.

Once the user is done with a frame, opened via **MainFrame**, they can click either "**Close**" button or regular "**x**" button, it **disposes** the frame without closing the application as the close operation has been set to dispose on close instead of exit on close.

Database Structure

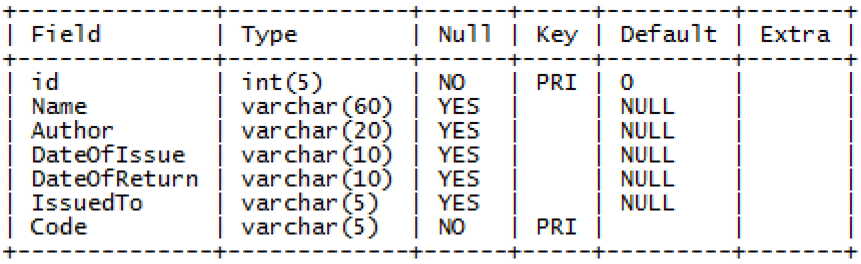
Description: Database is **automatically created** on a successful login (see Project Source Code Part Two). Database name is **libri** and it has three tables, i.e., **books**, **issuedBooks**, **members**.

Description of table “**books**”



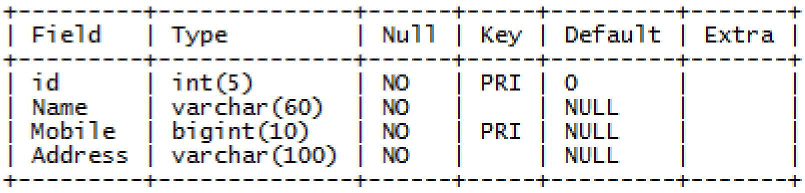
```
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| id     | int(5)      | NO   | PRI | NULL    |       |
| Name   | varchar(60) | NO   |     | NULL    |       |
| Status | varchar(9)  | NO   |     | NULL    |       |
| Author | varchar(20) | NO   |     | NULL    |       |
| Code   | varchar(5)  | NO   | PRI |         |       |
+--------+-------------+------+-----+---------+-------+
```

Description of table “**issuedBooks**”



```
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| id           | int(5)      | NO   | PRI | 0       |       |
| Name         | varchar(60) | YES  |     | NULL    |       |
| Author       | varchar(20) | YES  |     | NULL    |       |
| DateOfIssue  | varchar(10) | YES  |     | NULL    |       |
| DateOfReturn | varchar(10) | YES  |     | NULL    |       |
| IssuedTo     | varchar(5)  | YES  |     | NULL    |       |
| Code         | varchar(5)  | NO   | PRI |         |       |
+--------------+-------------+------+-----+---------+-------+
```

Description of table “**members**”



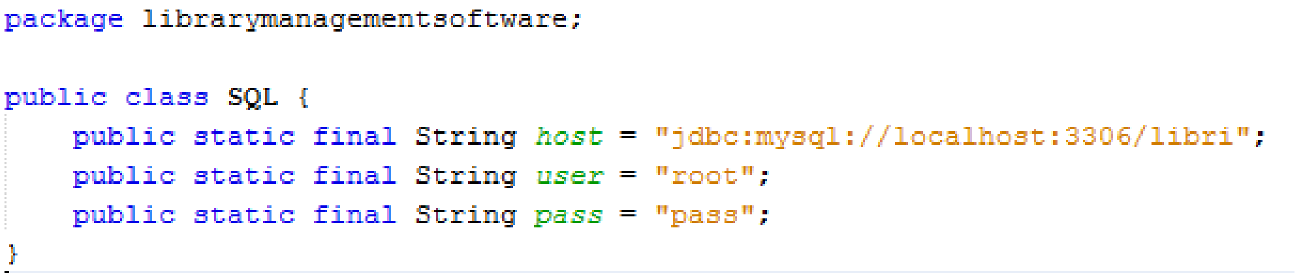
```
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| id      | int(5)       | NO   | PRI | 0       |       |
| Name    | varchar(60)  | NO   |     | NULL    |       |
| Mobile  | bigint(10)   | NO   | PRI | NULL    |       |
| Address | varchar(100) | NO   |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
```

Project

Source Code

(Part One – SQL.java)

Description: This is a custom **class** that provides **SQL** user data to the whole application, so if there is a change, we just have to change it in **SQL class** instead of changing it everywhere.



```
package librarymanagementsoftware;

public class SQL {
    public static final String host = "jdbc:mysql://localhost:3306/libri";
    public static final String user = "root";
    public static final String pass = "pass";
}
```

Project

Source Code

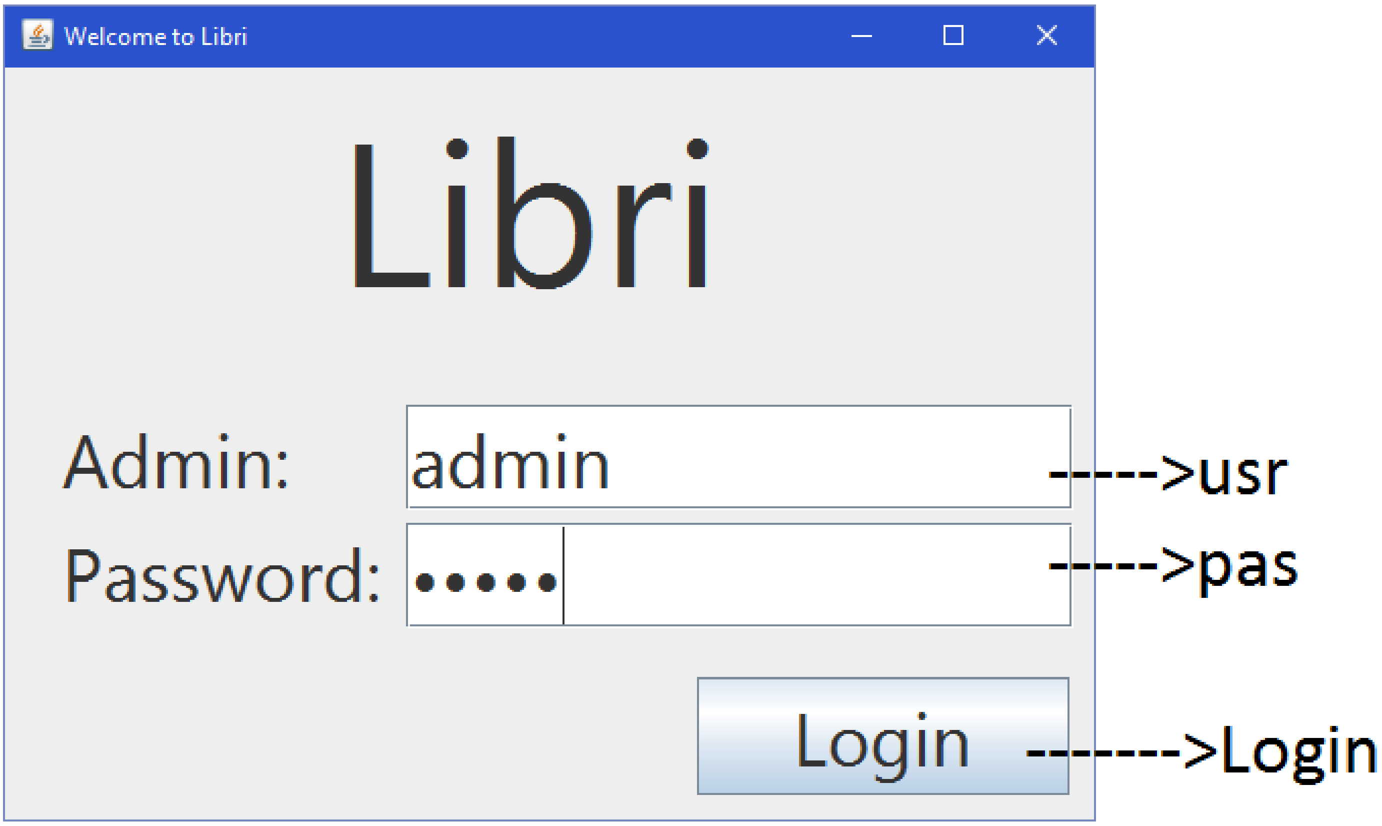
(Part Two – Login)

Description: This is a **JFrame** which first, takes user input as **username and password**, then, checks if they are correct or not. If the credentials are correct, it creates a database **if not** existed earlier and then add 3 tables to it (**books, issuedBooks, members**) and it also adds dummy data for testing which can be removed later and if all is set, **MainFrame.java** opens else a Message Box is shown saying that user name or password is incorrect.

Import Statements:

```
import java.sql.*;
import javax.swing.*;
```

Login frame design

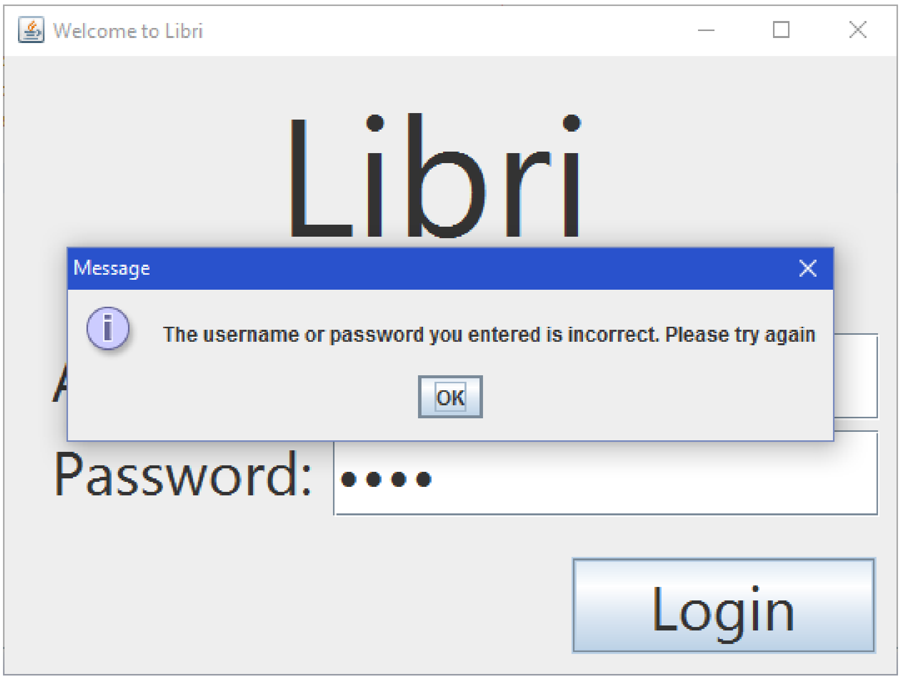


Welcome to Libri

Libri

Admin: admin

Password: •••••



Welcome to Libri

Libri

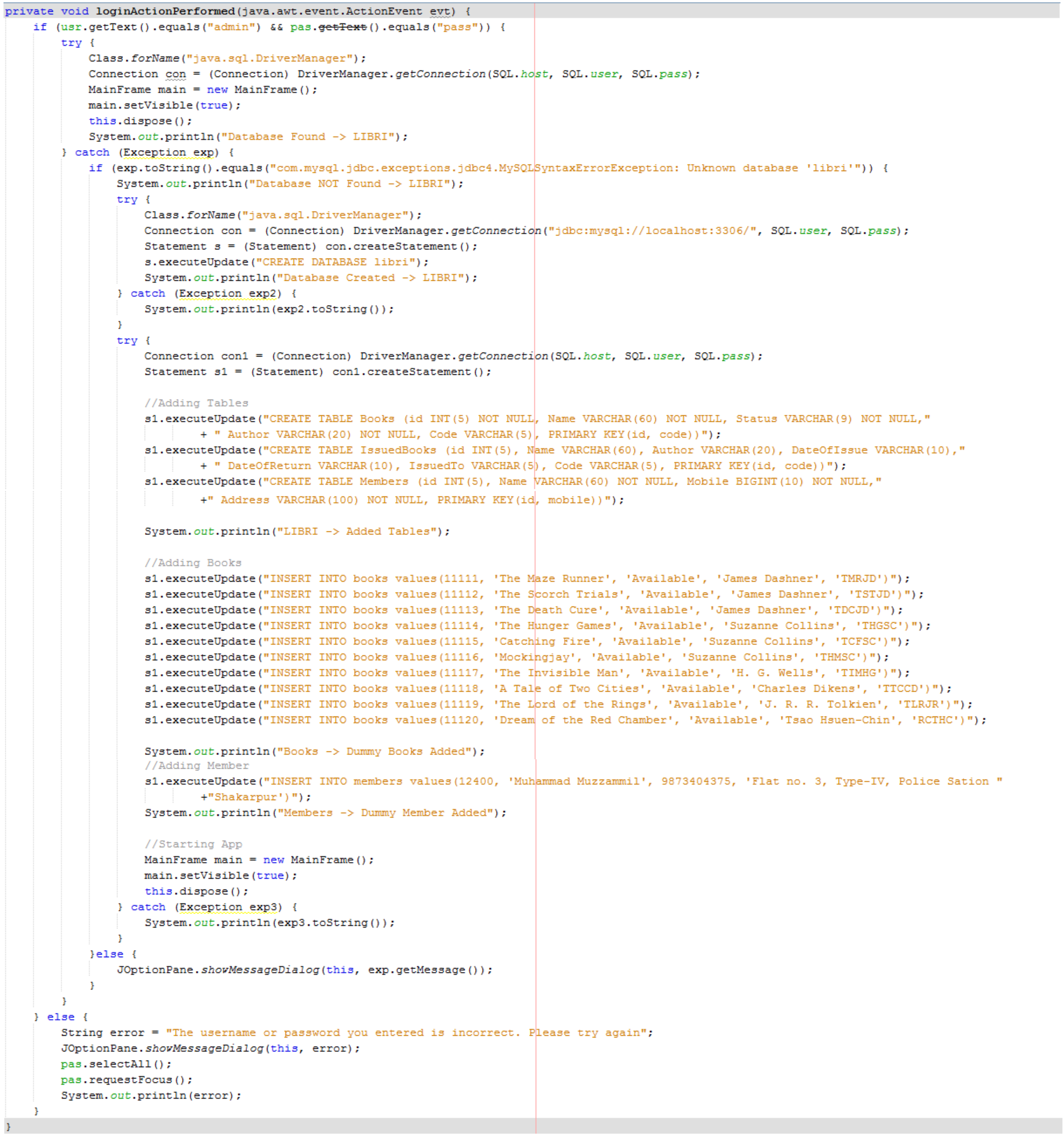
Message

The username or password you entered is incorrect. Please try again

Password: ••••

Libri showing an error when username or password entered are incorrect. Password field is focused after this message.

On click event of **Login**



```
private void loginActionPerformed(java.awt.event.ActionEvent evt) {
    if (usr.getText().equals("admin") && pas.~~getText~~().equals("pass")) {
        try {
            Class.forName("java.sql.DriverManager");
            Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
            MainFrame main = new MainFrame();
            main.setVisible(true);
            this.dispose();
            System.out.println("Database Found -> LIBRI");
        } catch (Exception exp) {
            if (exp.toString().equals("com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException: Unknown database 'libri'")) {
                System.out.println("Database NOT Found -> LIBRI");
                try {
                    Class.forName("java.sql.DriverManager");
                    Connection con = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/", SQL.user, SQL.pass);
                    Statement s = (Statement) con.createStatement();
                    s.executeUpdate("CREATE DATABASE libri");
                    System.out.println("Database Created -> LIBRI");
                } catch (Exception exp2) {
                    System.out.println(exp2.toString());
                }
                try {
                    Connection con1 = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
                    Statement s1 = (Statement) con1.createStatement();

                    //Adding Tables
                    s1.executeUpdate("CREATE TABLE Books (id INT(5) NOT NULL, Name VARCHAR(60) NOT NULL, Status VARCHAR(9) NOT NULL,"
                            + " Author VARCHAR(20) NOT NULL, Code VARCHAR(5), PRIMARY KEY(id, code))");
                    s1.executeUpdate("CREATE TABLE IssuedBooks (id INT(5), Name VARCHAR(60), Author VARCHAR(20), DateOfIssue VARCHAR(10),"
                            + " DateOfReturn VARCHAR(10), IssuedTo VARCHAR(5), Code VARCHAR(5), PRIMARY KEY(id, code))");
                    s1.executeUpdate("CREATE TABLE Members (id INT(5), Name VARCHAR(60) NOT NULL, Mobile BIGINT(10) NOT NULL,"
                            +" Address VARCHAR(100) NOT NULL, PRIMARY KEY(id, mobile))");

                    System.out.println("LIBRI -> Added Tables");

                    //Adding Books
                    s1.executeUpdate("INSERT INTO books values(11111, 'The Maze Runner', 'Available', 'James Dashner', 'TMRJD')");
                    s1.executeUpdate("INSERT INTO books values(11112, 'The Scorch Trials', 'Available', 'James Dashner', 'TSTJD')");
                    s1.executeUpdate("INSERT INTO books values(11113, 'The Death Cure', 'Available', 'James Dashner', 'TDCJD')");
                    s1.executeUpdate("INSERT INTO books values(11114, 'The Hunger Games', 'Available', 'Suzanne Collins', 'THGSC')");
                    s1.executeUpdate("INSERT INTO books values(11115, 'Catching Fire', 'Available', 'Suzanne Collins', 'TCFSC')");
                    s1.executeUpdate("INSERT INTO books values(11116, 'Mockingjay', 'Available', 'Suzanne Collins', 'THMSC')");
                    s1.executeUpdate("INSERT INTO books values(11117, 'The Invisible Man', 'Available', 'H. G. Wells', 'TIMHG')");
                    s1.executeUpdate("INSERT INTO books values(11118, 'A Tale of Two Cities', 'Available', 'Charles Dikens', 'TTCCD')");
                    s1.executeUpdate("INSERT INTO books values(11119, 'The Lord of the Rings', 'Available', 'J. R. R. Tolkien', 'TLRJR')");
                    s1.executeUpdate("INSERT INTO books values(11120, 'Dream of the Red Chamber', 'Available', 'Tsao Hsuen-Chin', 'RCTHC')");

                    System.out.println("Books -> Dummy Books Added");
                    //Adding Member
                    s1.executeUpdate("INSERT INTO members values(12400, 'Muhammad Muzzammil', 9873404375, 'Flat no. 3, Type-IV, Police Sation "
                            +"Shakarpur')");
                    System.out.println("Members -> Dummy Member Added");

                    //Starting App
                    MainFrame main = new MainFrame();
                    main.setVisible(true);
                    this.dispose();
                } catch (Exception exp3) {
                    System.out.println(exp3.toString());
                }
            }else {
                JOptionPane.showMessageDialog(this, exp.getMessage());
            }
        }
    } else {
        String error = "The username or password you entered is incorrect. Please try again";
        JOptionPane.showMessageDialog(this, error);
        pas.selectAll();
        pas.requestFocus();
        System.out.println(error);
    }
}
```

Project

Source Code

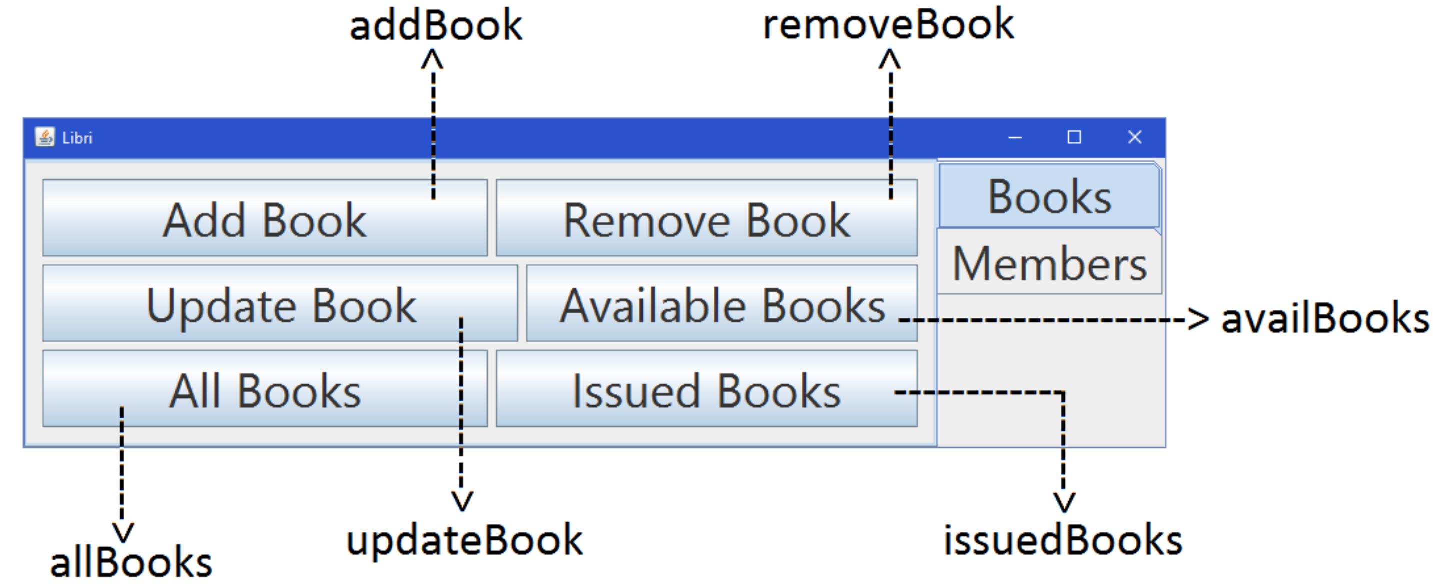
(Part Three – Books)

Description: There are **two tabs (JTabbedPane)** which separates **Books** and **Member** functions on frame.

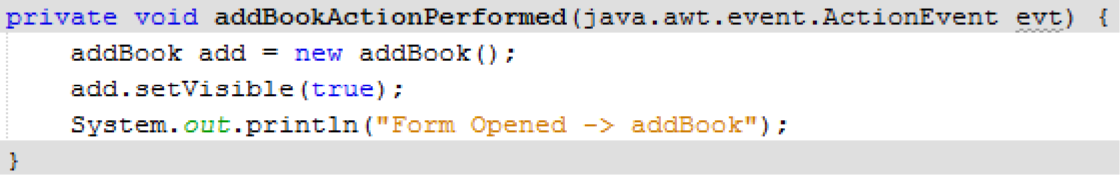
In **Books tab** there are **six frames** which it redirects to by clicking on buttons which are:

1. Add Book	->	**addBook**.java
2. Remove Book	->	**removeBook**.java
3. Update Book	->	**updateBook**.java
4. Available Books	->	**availableBooks**.java
5. All Books	->	**allBooks**.java
6. Issued Books	->	**issuedBooks**.java

Books tab design

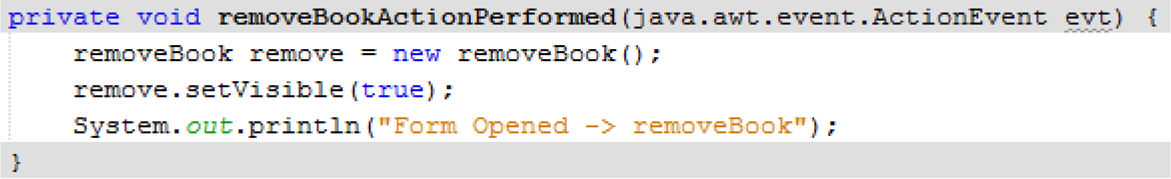


On click event of addBook



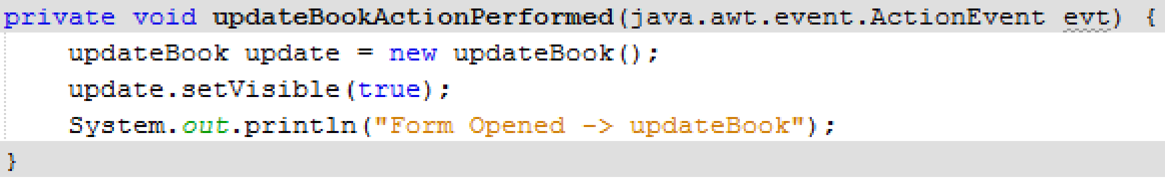
```
private void addBookActionPerformed(java.awt.event.ActionEvent evt) {
    addBook add = new addBook();
    add.setVisible(true);
    System.out.println("Form Opened -> addBook");
}
```

On click event of removeBook



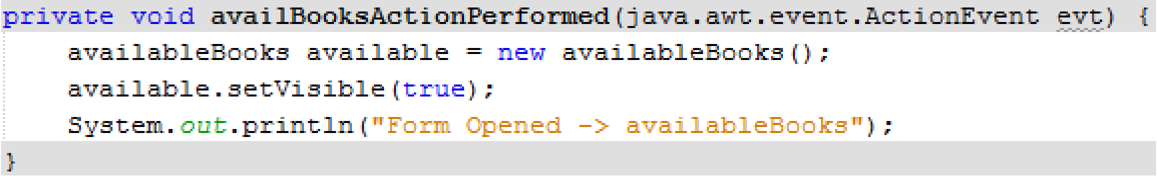
```
private void removeBookActionPerformed(java.awt.event.ActionEvent evt) {
    removeBook remove = new removeBook();
    remove.setVisible(true);
    System.out.println("Form Opened -> removeBook");
}
```

On click event of updateBook



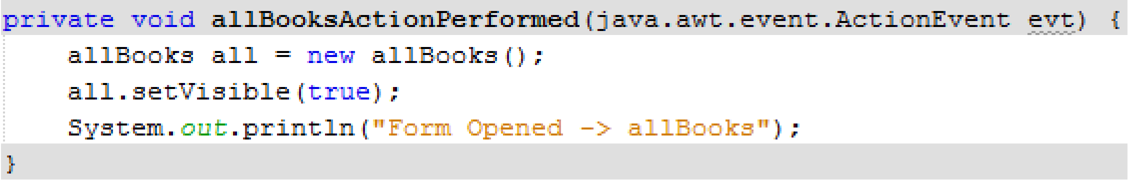
```
private void updateBookActionPerformed(java.awt.event.ActionEvent evt) {
    updateBook update = new updateBook();
    update.setVisible(true);
    System.out.println("Form Opened -> updateBook");
}
```

On click event of **availBooks**



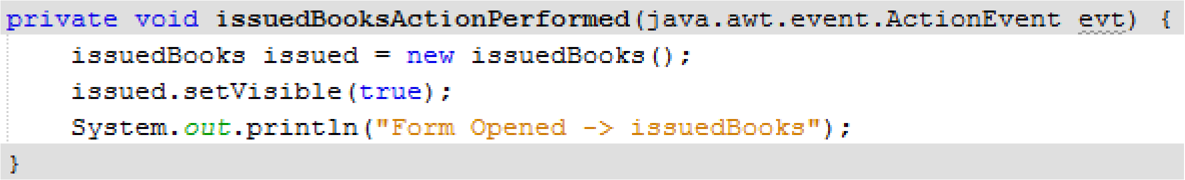
```
private void availBooksActionPerformed(java.awt.event.ActionEvent evt) {
    availableBooks available = new availableBooks();
    available.setVisible(true);
    System.out.println("Form Opened -> availableBooks");
}
```

On click event of **allBooks**



```
private void allBooksActionPerformed(java.awt.event.ActionEvent evt) {
    allBooks all = new allBooks();
    all.setVisible(true);
    System.out.println("Form Opened -> allBooks");
}
```

On click event of **issuedBooks**



```
private void issuedBooksActionPerformed(java.awt.event.ActionEvent evt) {
    issuedBooks issued = new issuedBooks();
    issued.setVisible(true);
    System.out.println("Form Opened -> issuedBooks");
}
```

Add Book Frame

Description: This frame is used to **add books** in database (“libri”). When it is loaded, **a random number generator** generates a random number and assigns it to **id** JTextField, which is **uneditable**, to be book’s id in database.

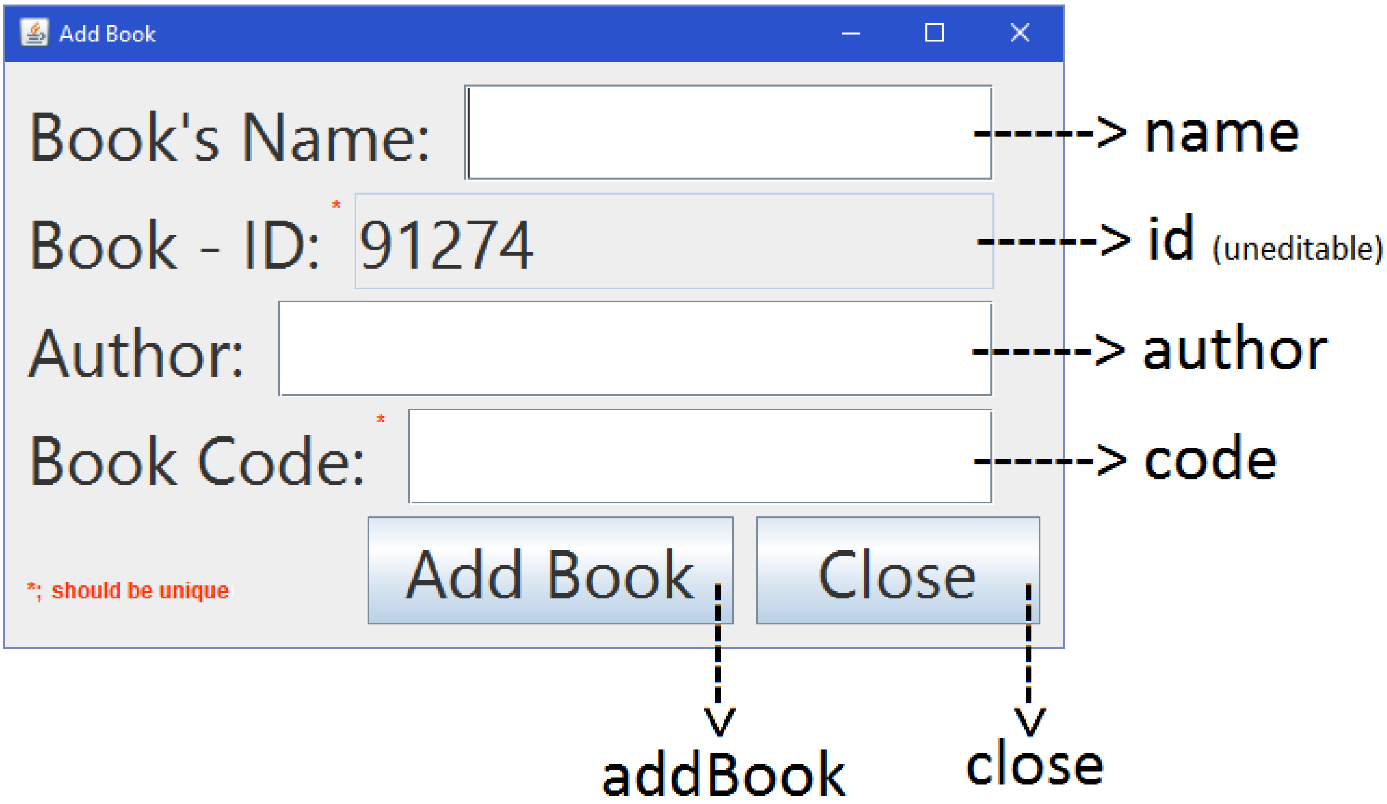
Import Statements:

```
import java.sql.*;
import java.util.Random;
import javax.swing.*;
```

Global Variable:

```
int bookID;
```

addBook frame design



Book's Name:

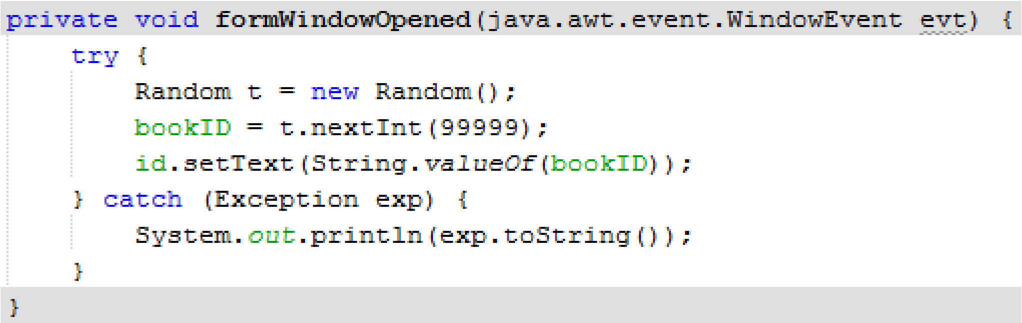
Book - ID: * 91274

Author:

Book Code: *

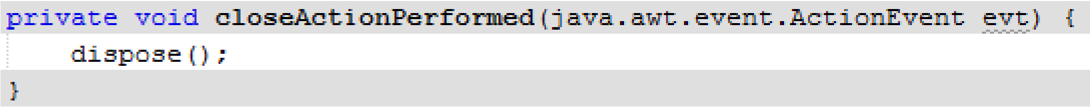
*; should be unique

On frame load event



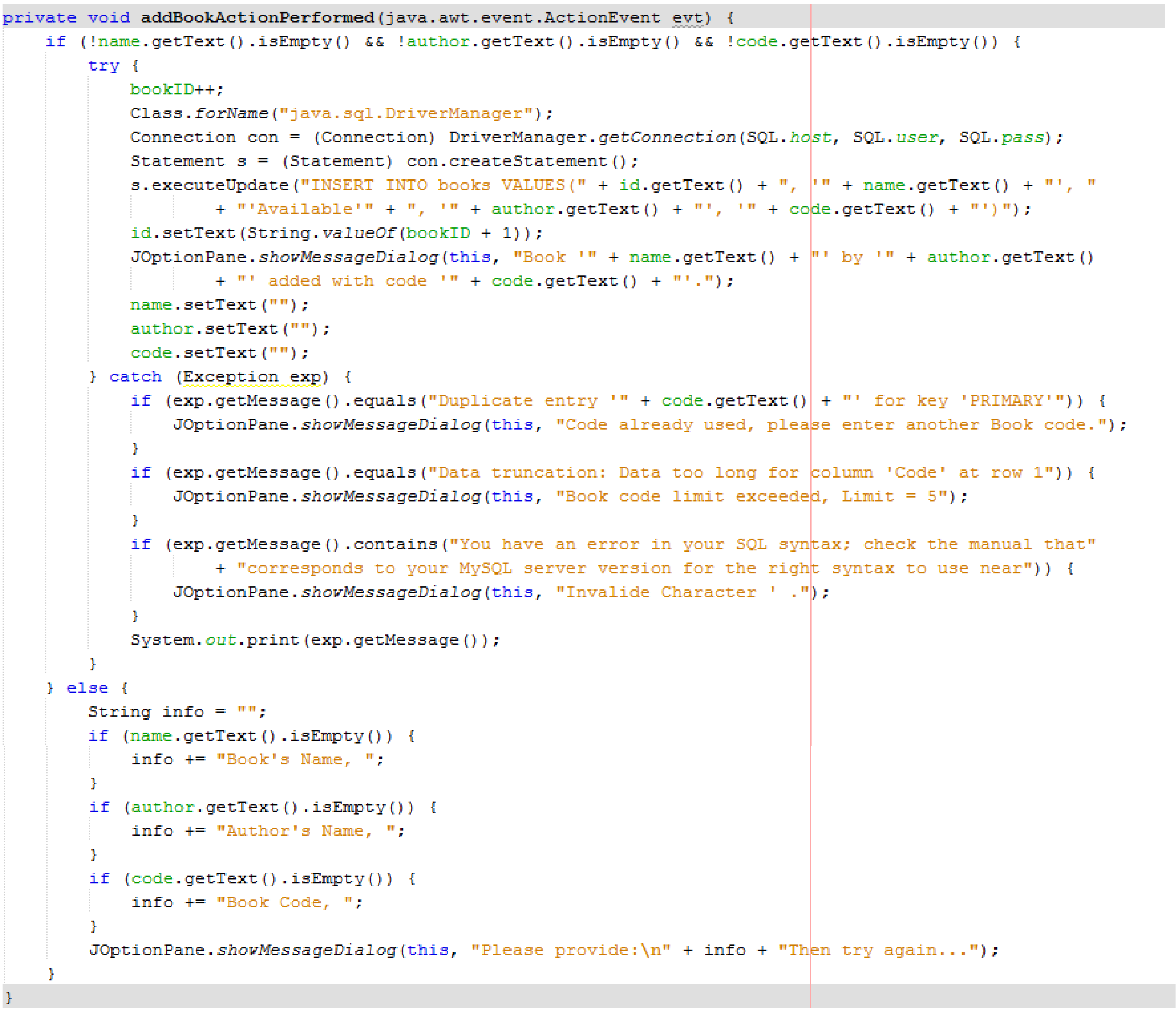
```
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    try {
        Random t = new Random();
        bookID = t.nextInt(99999);
        id.setText(String.valueOf(bookID));
    } catch (Exception exp) {
        System.out.println(exp.toString());
    }
}
```

On click event of close

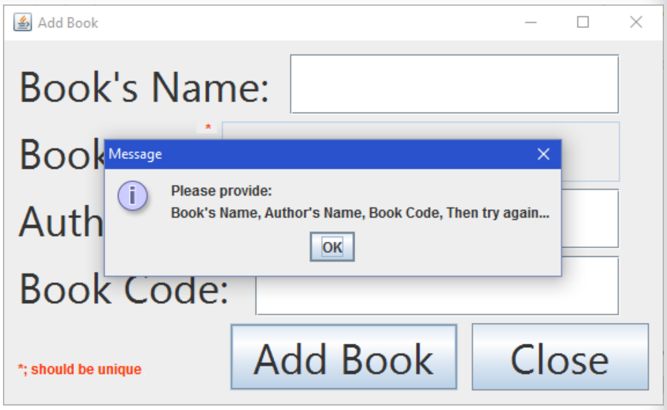


```
private void closeActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```

On click event of **addBook**



```
private void addBookActionPerformed(java.awt.event.ActionEvent evt) {
    if (!name.getText().isEmpty() && !author.getText().isEmpty() && !code.getText().isEmpty()) {
        try {
            bookID++;
            Class.forName("java.sql.DriverManager");
            Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
            Statement s = (Statement) con.createStatement();
            s.executeUpdate("INSERT INTO books VALUES(" + id.getText() + ", '" + name.getText() + "', "
                    + "'Available'" + ", '" + author.getText() + "', '" + code.getText() + "')");
            id.setText(String.valueOf(bookID + 1));
            JOptionPane.showMessageDialog(this, "Book '" + name.getText() + "' by '" + author.getText()
                    + "' added with code '" + code.getText() + "'.");
            name.setText("");
            author.setText("");
            code.setText("");
        } catch (Exception exp) {
            if (exp.getMessage().equals("Duplicate entry '" + code.getText() + "' for key 'PRIMARY'")) {
                JOptionPane.showMessageDialog(this, "Code already used, please enter another Book code.");
            }
            if (exp.getMessage().equals("Data truncation: Data too long for column 'Code' at row 1")) {
                JOptionPane.showMessageDialog(this, "Book code limit exceeded, Limit = 5");
            }
            if (exp.getMessage().contains("You have an error in your SQL syntax; check the manual that"
                    + "corresponds to your MySQL server version for the right syntax to use near")) {
                JOptionPane.showMessageDialog(this, "Invalide Character ' .");
            }
            System.out.print(exp.getMessage());
        }
    } else {
        String info = "";
        if (name.getText().isEmpty()) {
            info += "Book's Name, ";
        }
        if (author.getText().isEmpty()) {
            info += "Author's Name, ";
        }
        if (code.getText().isEmpty()) {
            info += "Book Code, ";
        }
        JOptionPane.showMessageDialog(this, "Please provide:\n" + info + "Then try again...");
    }
}
```



Please provide:
Book's Name, Author's Name, Book Code, Then try again...

Libri showing an error when name, author and book code is not provided.

Remove Book Frame

Description: This frame is used to **remove books** in database ("libri"). User just have to provide the book id and password to remove a book. If the book is issued, libri will also remove it from **issued books**.

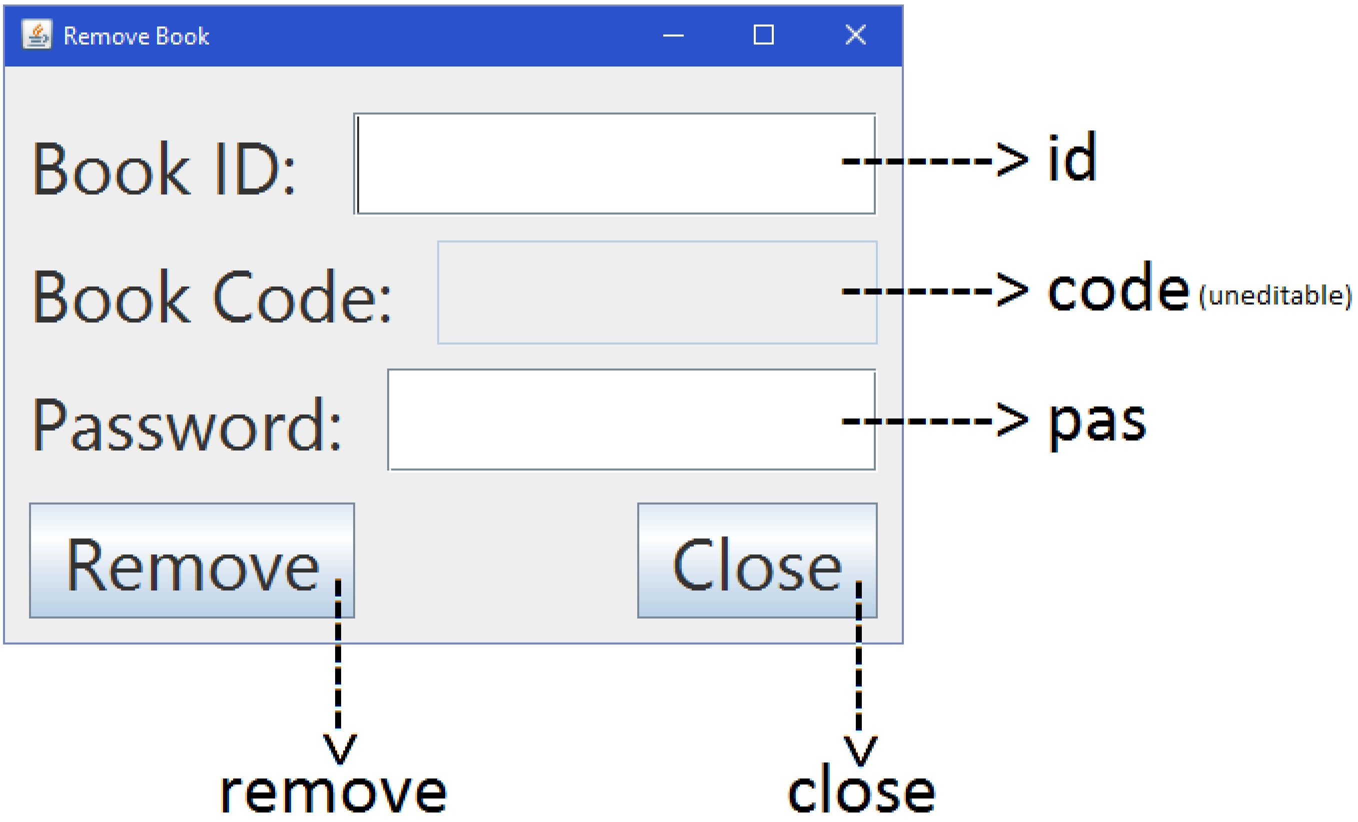
Import Statements:

```
import java.sql.*;

import javax.swing.*;
```

Global Variables:

```
boolean available, issued;
```

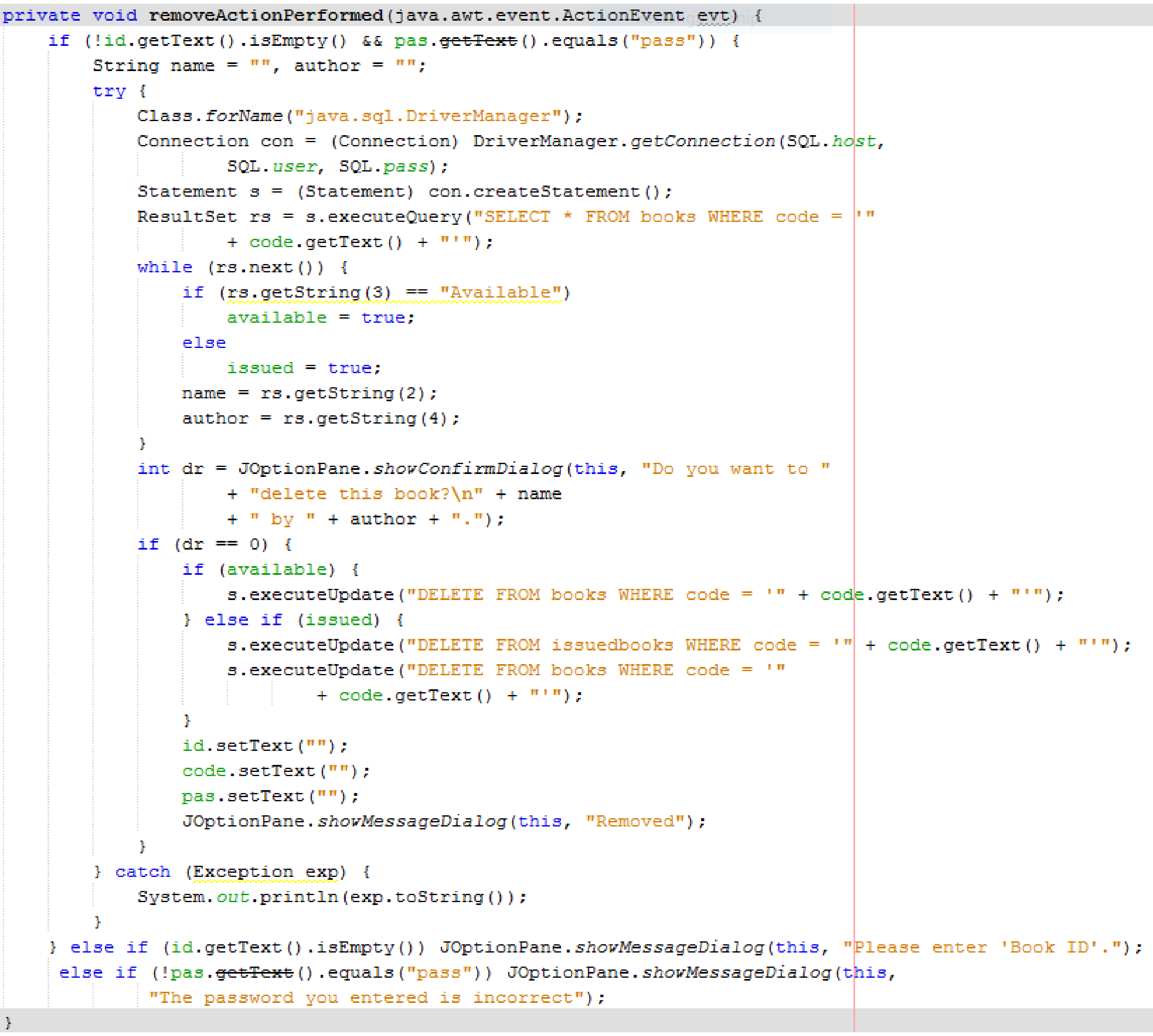
removeBook frame design



On click event of **close**

```
private void closeActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```

On click event of **remove**



```
private void removeActionPerformed(java.awt.event.ActionEvent evt) {
    if (!id.getText().isEmpty() && pas.~~getText~~().equals("pass")) {
        String name = "", author = "";
        try {
            Class.forName("java.sql.DriverManager");
            Connection con = (Connection) DriverManager.getConnection(SQL.host,
                    SQL.user, SQL.pass);
            Statement s = (Statement) con.createStatement();
            ResultSet rs = s.executeQuery("SELECT * FROM books WHERE code = '"
                    + code.getText() + "'");
            while (rs.next()) {
                if (rs.getString(3) == "Available")
                    available = true;
                else
                    issued = true;
                name = rs.getString(2);
                author = rs.getString(4);
            }
            int dr = JOptionPane.showConfirmDialog(this, "Do you want to "
                    + "delete this book?\n" + name
                    + " by " + author + ".");
            if (dr == 0) {
                if (available) {
                    s.executeUpdate("DELETE FROM books WHERE code = '" + code.getText() + "'");
                } else if (issued) {
                    s.executeUpdate("DELETE FROM issuedbooks WHERE code = '" + code.getText() + "'");
                    s.executeUpdate("DELETE FROM books WHERE code = '"
                            + code.getText() + "'");
                }
                id.setText("");
                code.setText("");
                pas.setText("");
                JOptionPane.showMessageDialog(this, "Removed");
            }
        } catch (Exception exp) {
            System.out.println(exp.toString());
        }
    } else if (id.getText().isEmpty()) JOptionPane.showMessageDialog(this, "Please enter 'Book ID'.");
     else if (!pas.~~getText~~().equals("pass")) JOptionPane.showMessageDialog(this,
             "The password you entered is incorrect");
}
```

Update Book Frame

Description: This frame is used to **update books** in database ("libri"). User just has to provide the book **id** and **name**, **author name**, and **book code** will appear and then user can **edit** them and then **save** them.

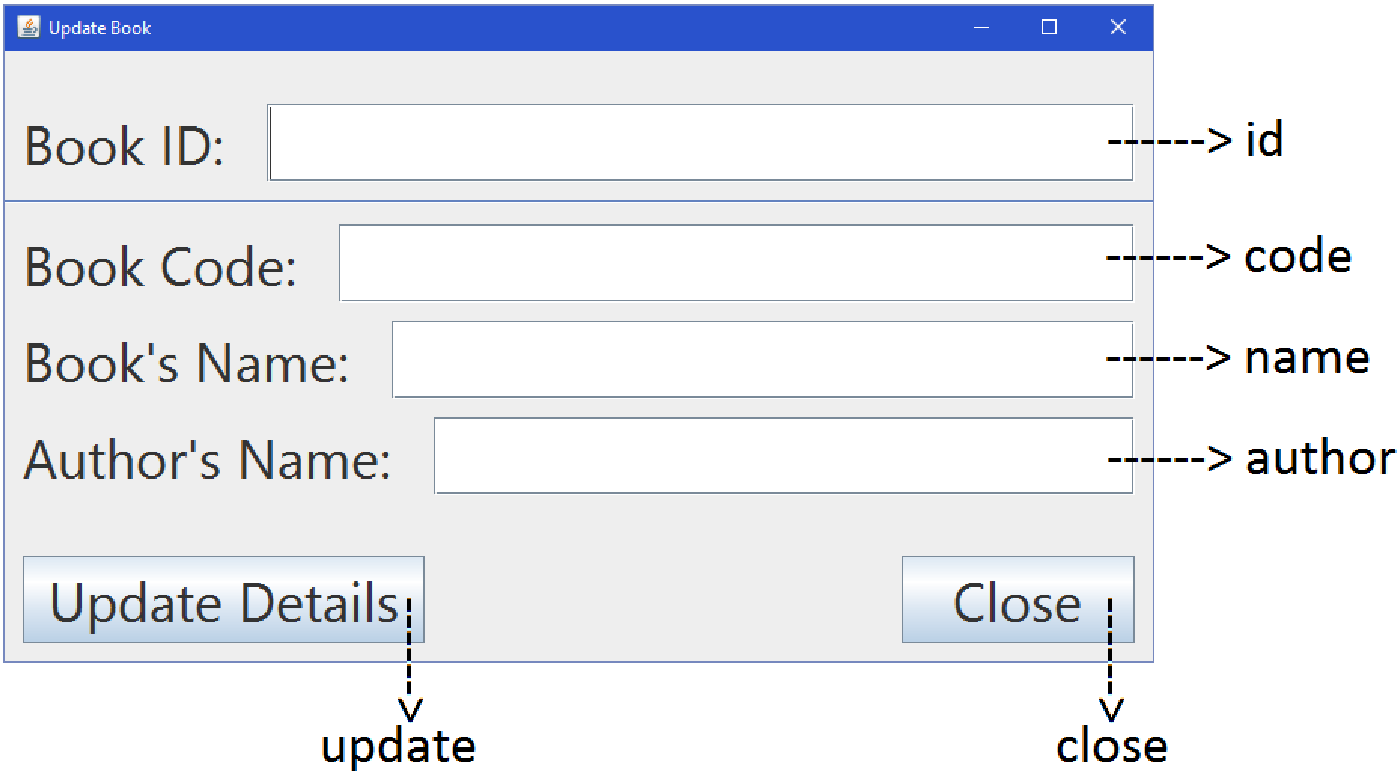
Import Statements:

```
import java.sql.*;

import javax.swing.*;
```

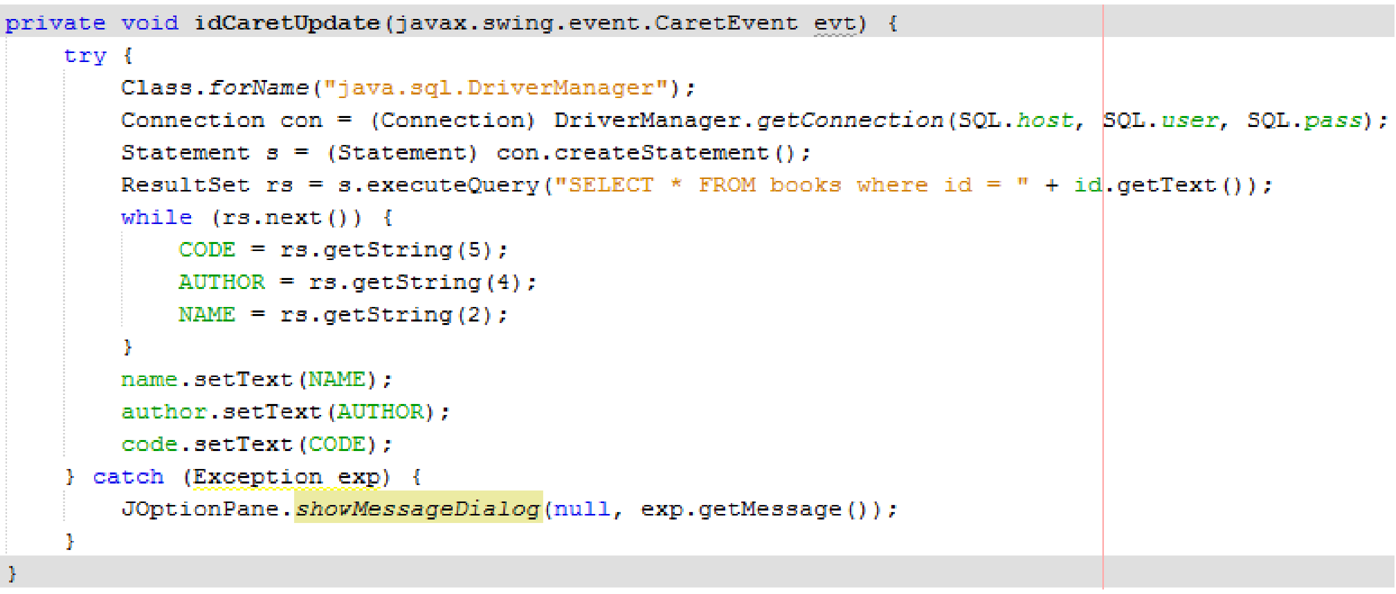
Global Variables:

```
String NAME, AUTHOR, CODE;
```

updateBook frame design

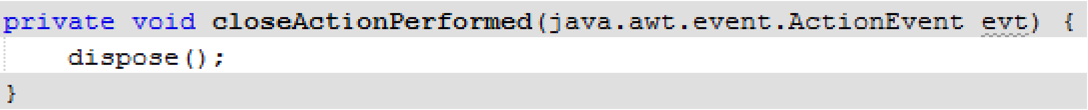


On caret update event of **id**



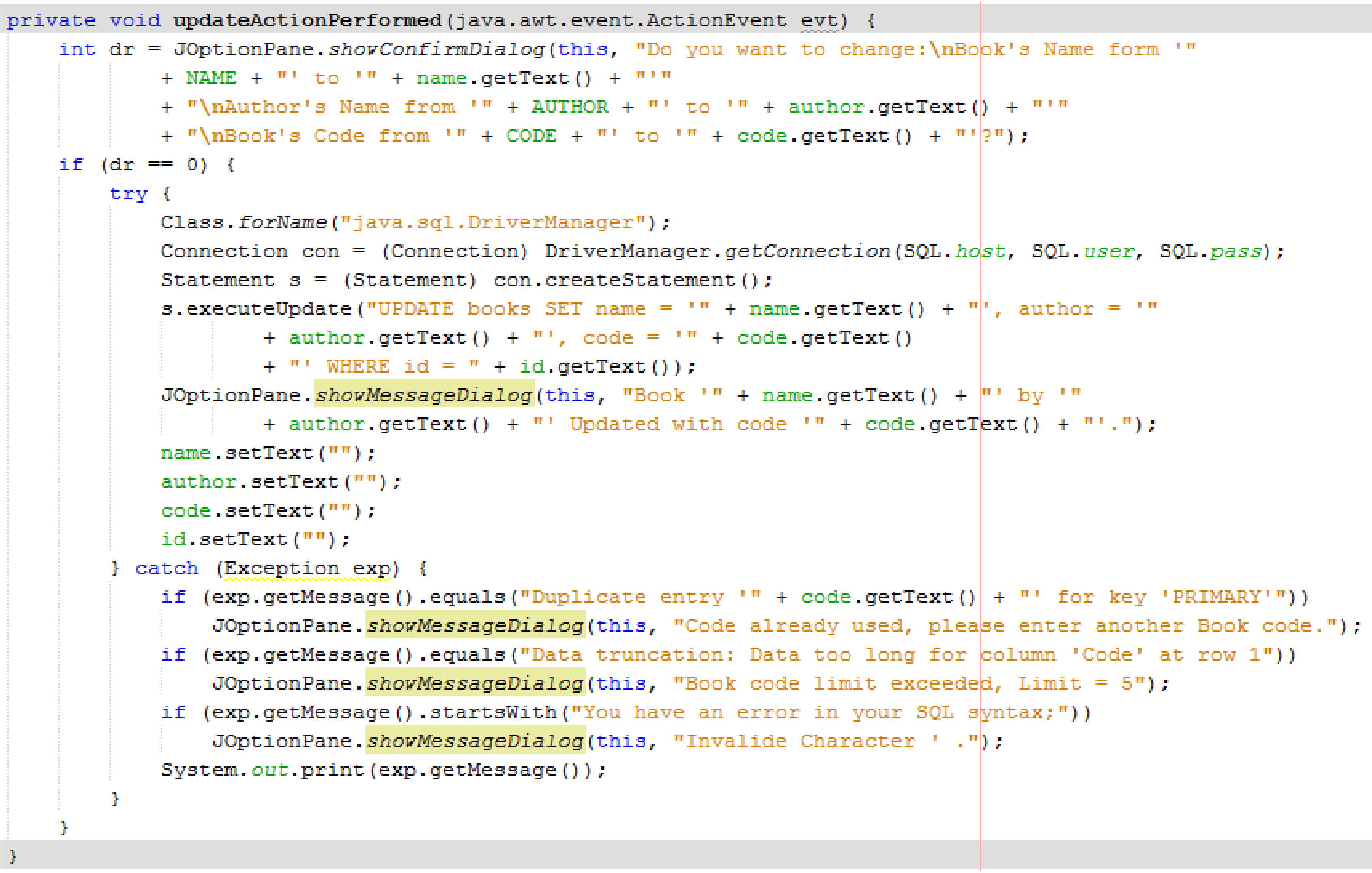
```
private void idCaretUpdate(javax.swing.event.CaretEvent evt) {
    try {
        Class.forName("java.sql.DriverManager");
        Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
        Statement s = (Statement) con.createStatement();
        ResultSet rs = s.executeQuery("SELECT * FROM books where id = " + id.getText());
        while (rs.next()) {
            CODE = rs.getString(5);
            AUTHOR = rs.getString(4);
            NAME = rs.getString(2);
        }
        name.setText(NAME);
        author.setText(AUTHOR);
        code.setText(CODE);
    } catch (Exception exp) {
        JOptionPane.showMessageDialog(null, exp.getMessage());
    }
}
```

On click event of **close**



```
private void closeActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```

On click event of **update**



```
private void updateActionPerformed(java.awt.event.ActionEvent evt) {
    int dr = JOptionPane.showConfirmDialog(this, "Do you want to change:\nBook's Name form '"
            + NAME + "' to '" + name.getText() + "'"
            + "\nAuthor's Name from '" + AUTHOR + "' to '" + author.getText() + "'"
            + "\nBook's Code from '" + CODE + "' to '" + code.getText() + "'?");
    if (dr == 0) {
        try {
            Class.forName("java.sql.DriverManager");
            Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
            Statement s = (Statement) con.createStatement();
            s.executeUpdate("UPDATE books SET name = '" + name.getText() + "', author = '"
                    + author.getText() + "', code = '" + code.getText()
                    + "' WHERE id = " + id.getText());
            JOptionPane.showMessageDialog(this, "Book '" + name.getText() + "' by '"
                    + author.getText() + "' Updated with code '" + code.getText() + "'.");
            name.setText("");
            author.setText("");
            code.setText("");
            id.setText("");
        } catch (Exception exp) {
            if (exp.getMessage().equals("Duplicate entry '" + code.getText() + "' for key 'PRIMARY'"))
                JOptionPane.showMessageDialog(this, "Code already used, please enter another Book code.");
            if (exp.getMessage().equals("Data truncation: Data too long for column 'Code' at row 1"))
                JOptionPane.showMessageDialog(this, "Book code limit exceeded, Limit = 5");
            if (exp.getMessage().startsWith("You have an error in your SQL syntax;"))
                JOptionPane.showMessageDialog(this, "Invalide Character ' .");
            System.out.print(exp.getMessage());
        }
    }
}
```

Available Books Frame

Description: This frame is used to **show all books** currently marked as **available**, not issued to anyone, in database ("libri").

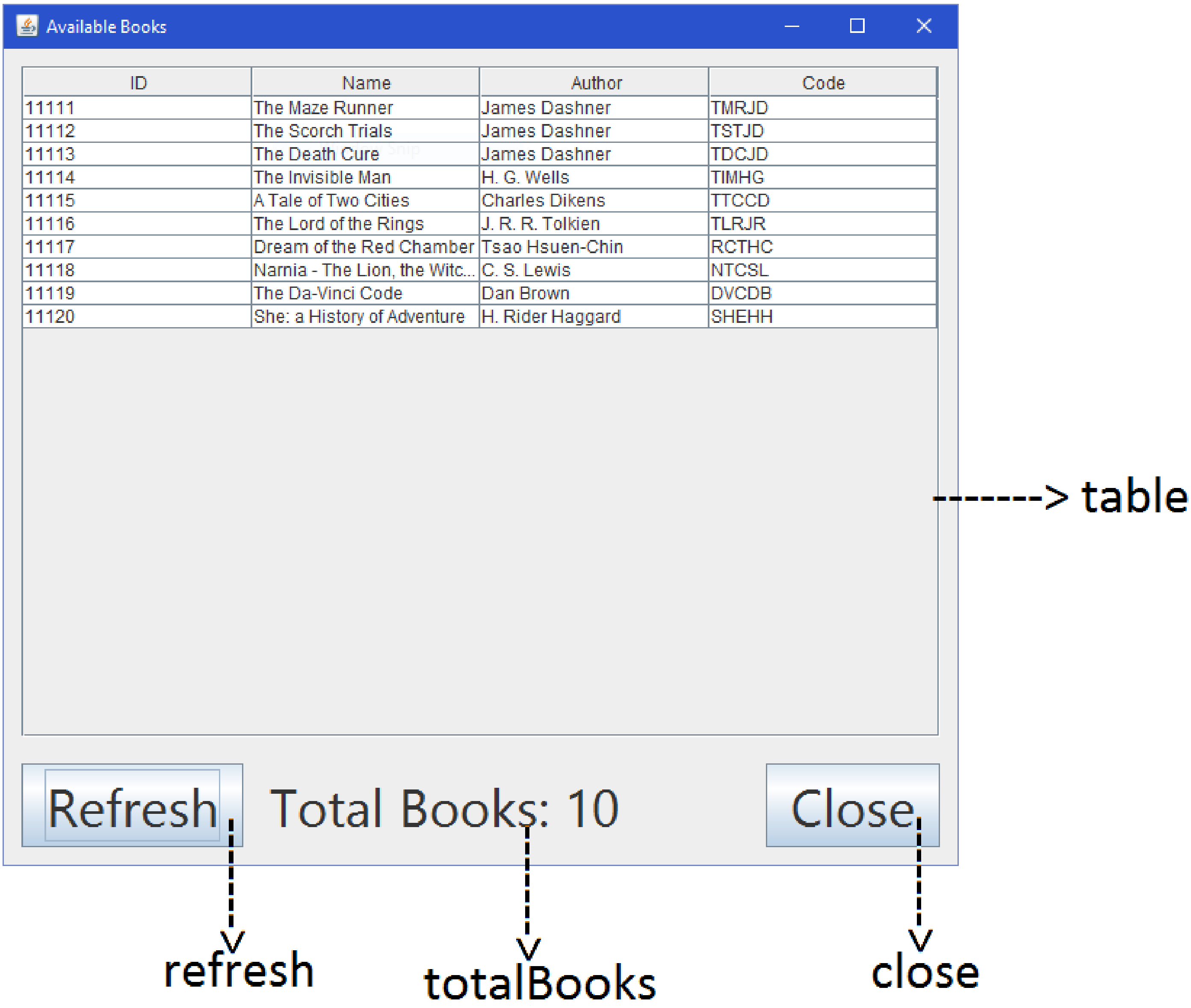
Import Statements:

```
import java.sql.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
```

Custom Method:

```
public void LoadAvailableBooks()
```

availableBooks frame design



ID	Name	Author	Code
11111	The Maze Runner	James Dashner	TMRJD
11112	The Scorch Trials	James Dashner	TSTJD
11113	The Death Cure	James Dashner	TDCJD
11114	The Invisible Man	H. G. Wells	TIMHG
11115	A Tale of Two Cities	Charles Dikens	TTCCD
11116	The Lord of the Rings	J. R. R. Tolkien	TLRJR
11117	Dream of the Red Chamber	Tsao Hsuen-Chin	RCTHC
11118	Narnia - The Lion, the Witc...	C. S. Lewis	NTCSL
11119	The Da-Vinci Code	Dan Brown	DVCDB
11120	She: a History of Adventure	H. Rider Haggard	SHEHH

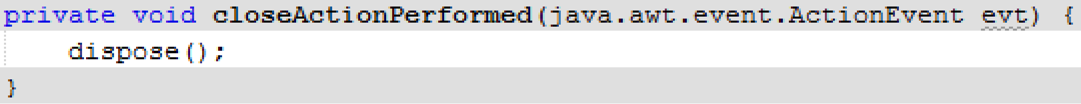
On frame load **event**

```
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    LoadAvailableBooks();
}
```

On click event of **refresh**

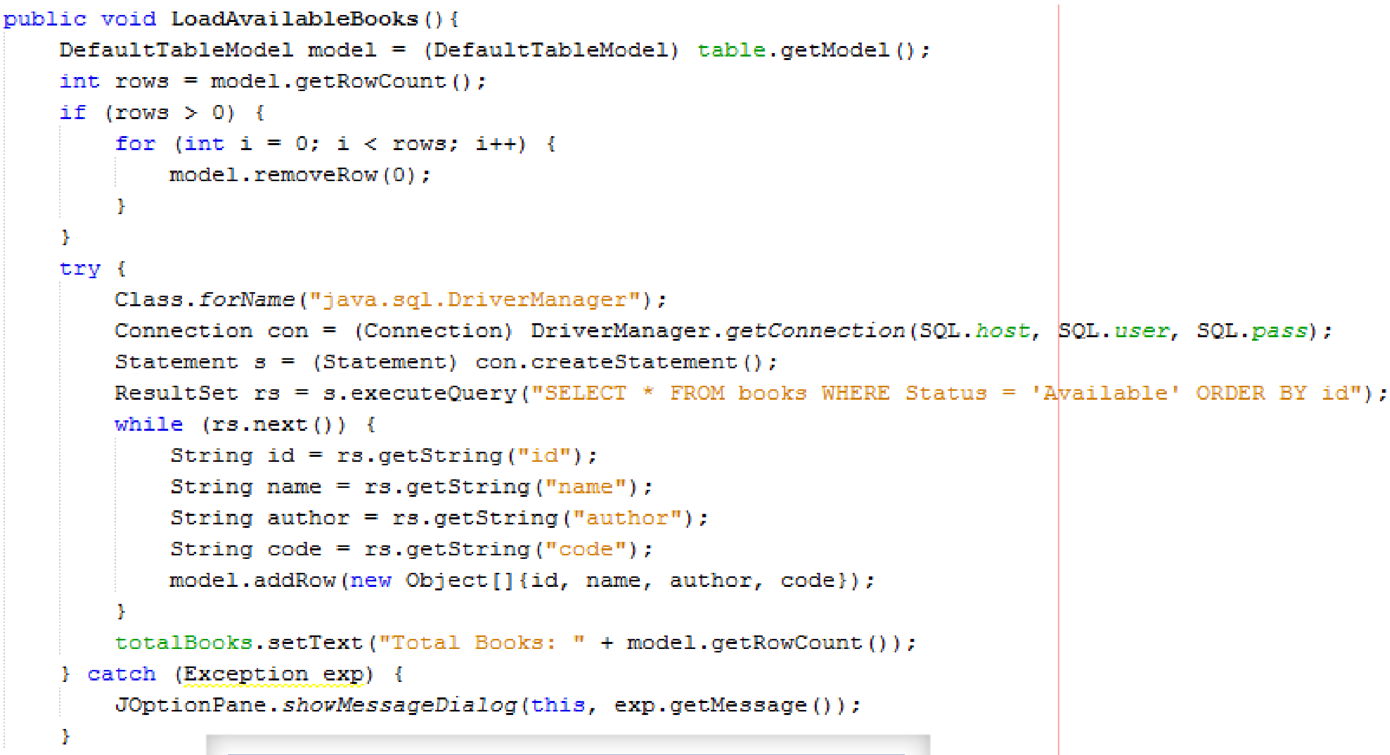
```
private void refreshActionPerformed(java.awt.event.ActionEvent evt) {
    LoadAvailableBooks();
}
```

On click event of **close**

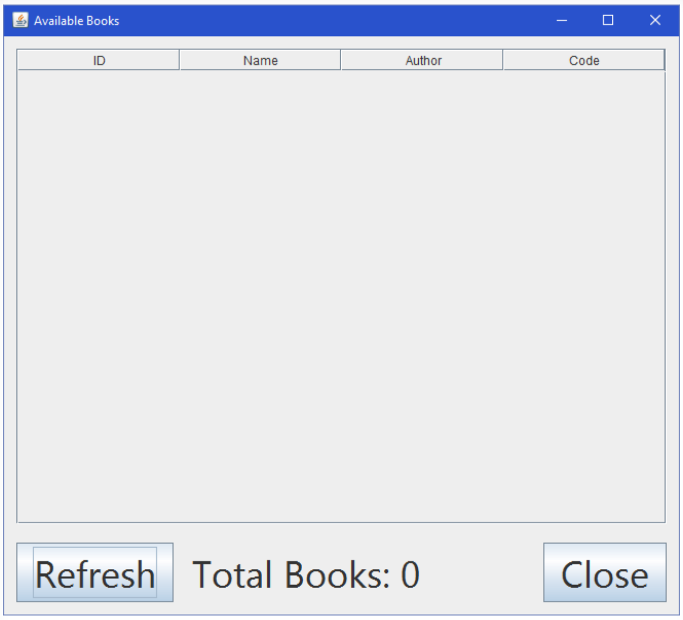


```
private void closeActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```

LoadAvailableBooks() method



```
public void LoadAvailableBooks(){
    DefaultTableModel model = (DefaultTableModel) table.getModel();
    int rows = model.getRowCount();
    if (rows > 0) {
        for (int i = 0; i < rows; i++) {
            model.removeRow(0);
        }
    }
    try {
        Class.forName("java.sql.DriverManager");
        Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
        Statement s = (Statement) con.createStatement();
        ResultSet rs = s.executeQuery("SELECT * FROM books WHERE Status = 'Available' ORDER BY id");
        while (rs.next()) {
            String id = rs.getString("id");
            String name = rs.getString("name");
            String author = rs.getString("author");
            String code = rs.getString("code");
            model.addRow(new Object[]{id, name, author, code});
        }
        totalBooks.setText("Total Books: " + model.getRowCount());
    } catch (Exception exp) {
        JOptionPane.showMessageDialog(this, exp.getMessage());
    }
}
```



ID	Name	Author	Code

Total Books: 0

Libri showing 0 books as there are no books currently available.

All Books Frame

Description: This frame is used to **show all books**, currently marked as **issued** **or** **available**, to anyone in database ("libri").

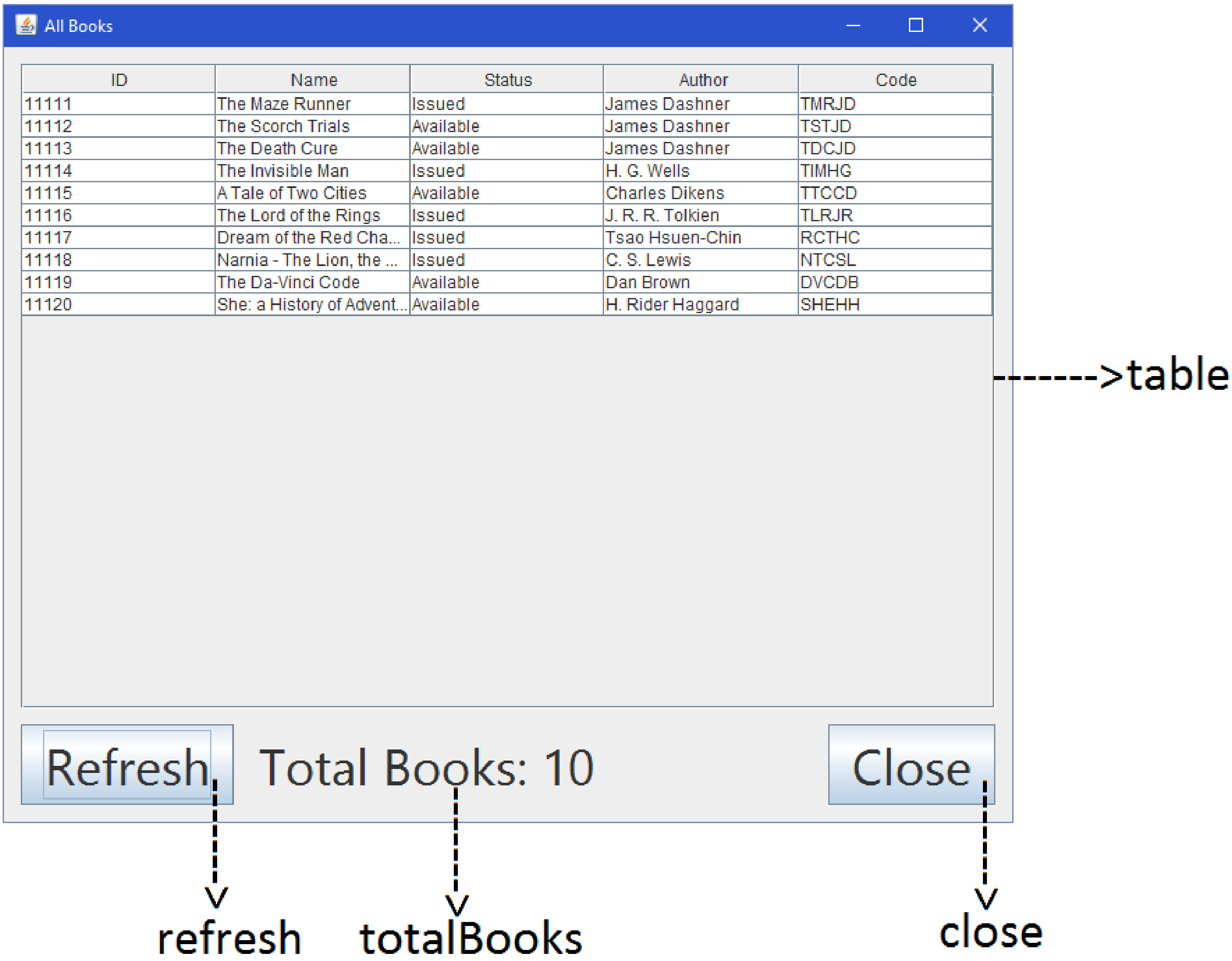
Import Statements:

```
import java.sql.*;

import javax.swing.*;

import javax.swing.table.DefaultTableModel;
```

Custom Method:

```
public void LoadBooks()
```

allBooks frame design



ID	Name	Status	Author	Code
11111	The Maze Runner	Issued	James Dashner	TMRJD
11112	The Scorch Trials	Available	James Dashner	TSTJD
11113	The Death Cure	Available	James Dashner	TDCJD
11114	The Invisible Man	Issued	H. G. Wells	TIMHG
11115	A Tale of Two Cities	Available	Charles Dikens	TTCCD
11116	The Lord of the Rings	Issued	J. R. R. Tolkien	TLRJR
11117	Dream of the Red Cha...	Issued	Tsao Hsuen-Chin	RCTHC
11118	Narnia - The Lion, the ...	Issued	C. S. Lewis	NTCSL
11119	The Da-Vinci Code	Available	Dan Brown	DVCDB
11120	She: a History of Advent...	Available	H. Rider Haggard	SHEHH

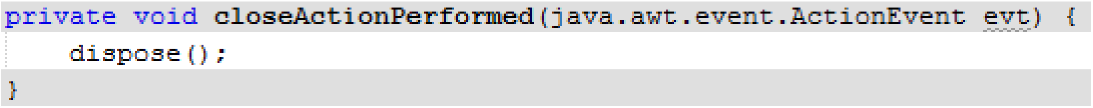
On frame load event

```
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    LoadBooks();
}
```

On click event of refresh

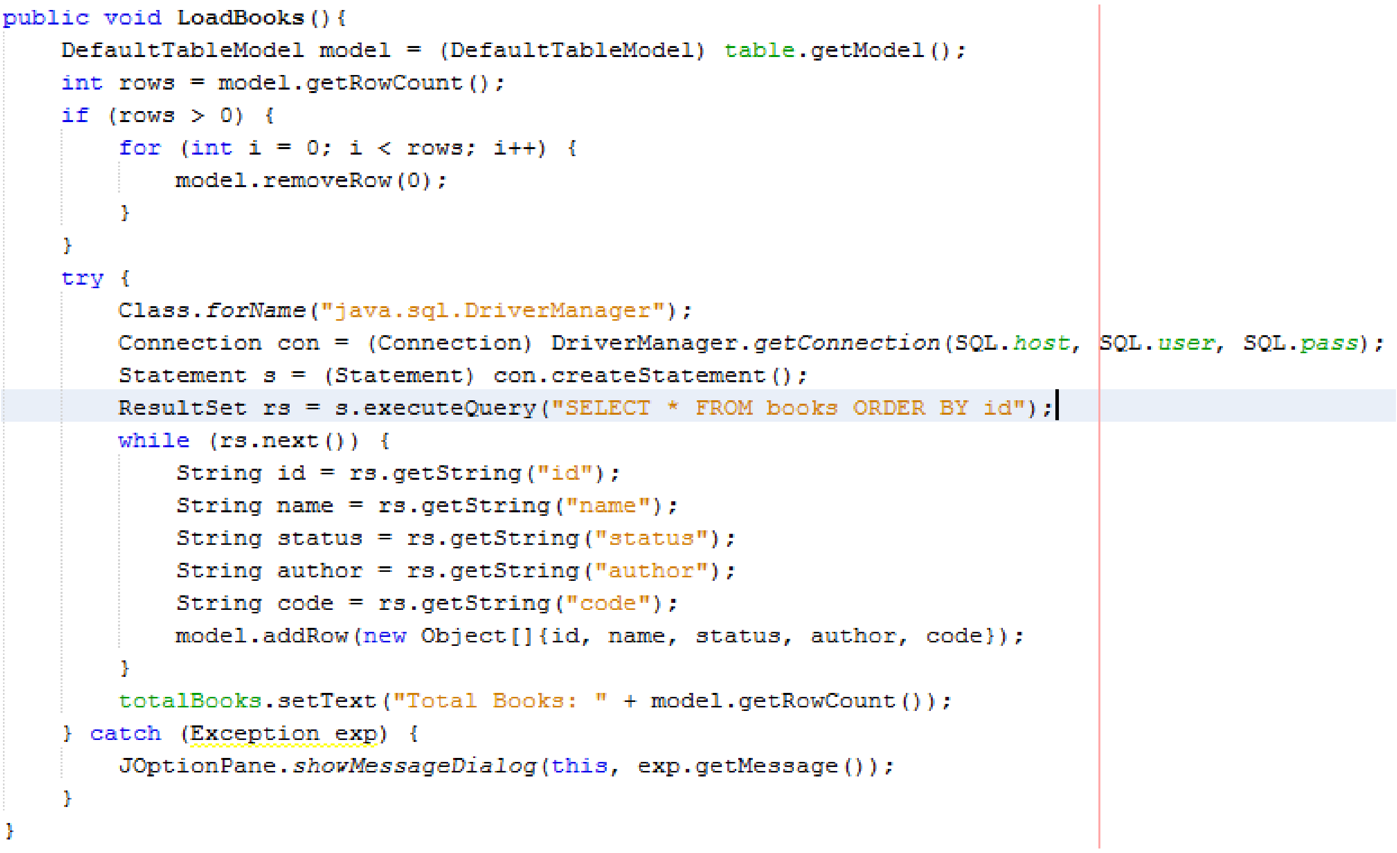
```
private void refreshActionPerformed(java.awt.event.ActionEvent evt) {
    LoadBooks();
}
```

On click event of **close**



```
private void closeActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```

LoadBooks() method



```
public void LoadBooks(){
    DefaultTableModel model = (DefaultTableModel) table.getModel();
    int rows = model.getRowCount();
    if (rows > 0) {
        for (int i = 0; i < rows; i++) {
            model.removeRow(0);
        }
    }
    try {
        Class.forName("java.sql.DriverManager");
        Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
        Statement s = (Statement) con.createStatement();
        ResultSet rs = s.executeQuery("SELECT * FROM books ORDER BY id");
        while (rs.next()) {
            String id = rs.getString("id");
            String name = rs.getString("name");
            String status = rs.getString("status");
            String author = rs.getString("author");
            String code = rs.getString("code");
            model.addRow(new Object[]{id, name, status, author, code});
        }
        totalBooks.setText("Total Books: " + model.getRowCount());
    } catch (Exception exp) {
        JOptionPane.showMessageDialog(this, exp.getMessage());
    }
}
```

Issued Books Frame

Description: This frame is used to **show issued books** which are currently marked as issued. It allows us to **remove and change** the book status of book as available, and, if we want, **re-issue** them to members of library in database ("libri").

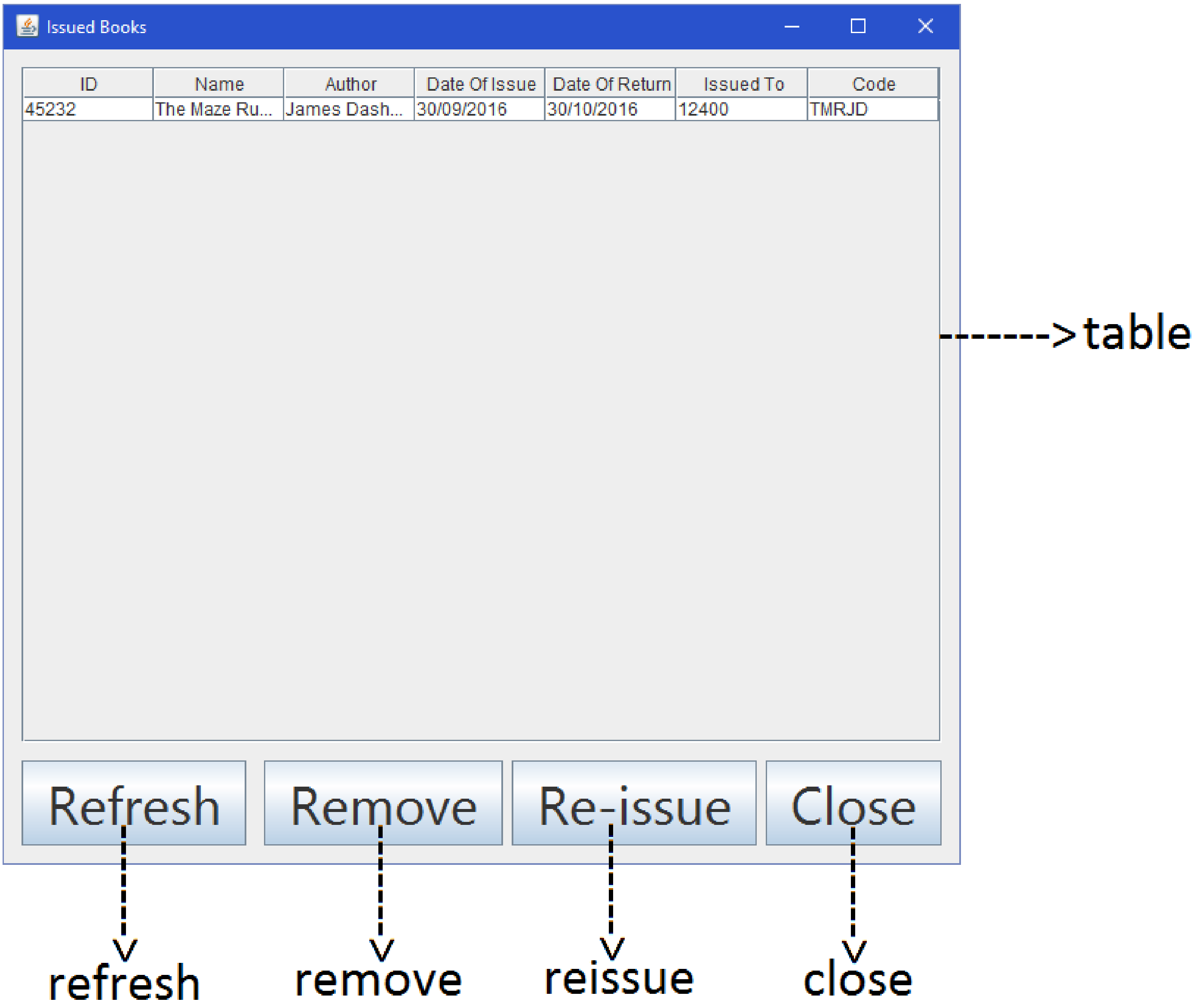
Import Statements:

```
import java.sql.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
```

Custom Method:

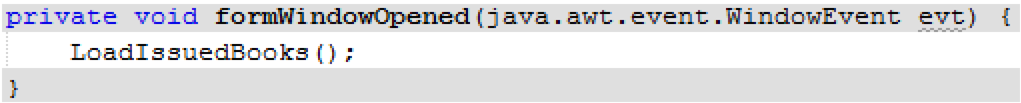
```
public void LoadIssuedBooks()
```

issuedBooks frame design



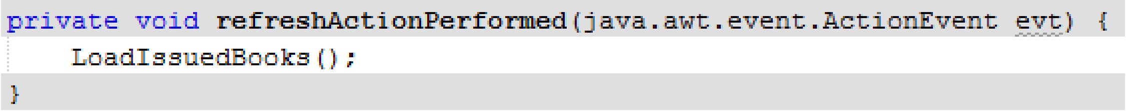
ID	Name	Author	Date Of Issue	Date Of Return	Issued To	Code
45232	The Maze Ru...	James Dash...	30/09/2016	30/10/2016	12400	TMRJD

issuedBooks -> On frame load event



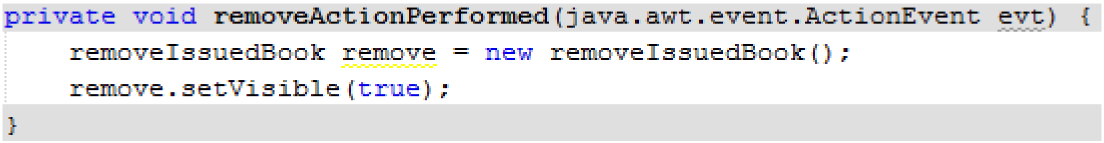
```
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    LoadIssuedBooks();
}
```

issuedBooks -> On click event of refresh



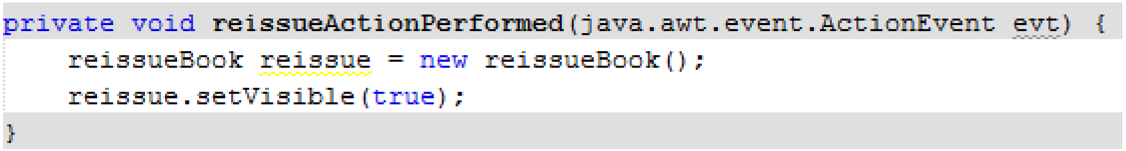
```
private void refreshActionPerformed(java.awt.event.ActionEvent evt) {
    LoadIssuedBooks();
}
```

issuedBooks -> On click event of **remove**



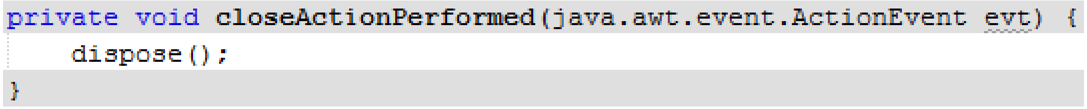
```
private void removeActionPerformed(java.awt.event.ActionEvent evt) {
    removeIssuedBook remove = new removeIssuedBook();
    remove.setVisible(true);
}
```

issuedBooks -> On click event of **reissue**



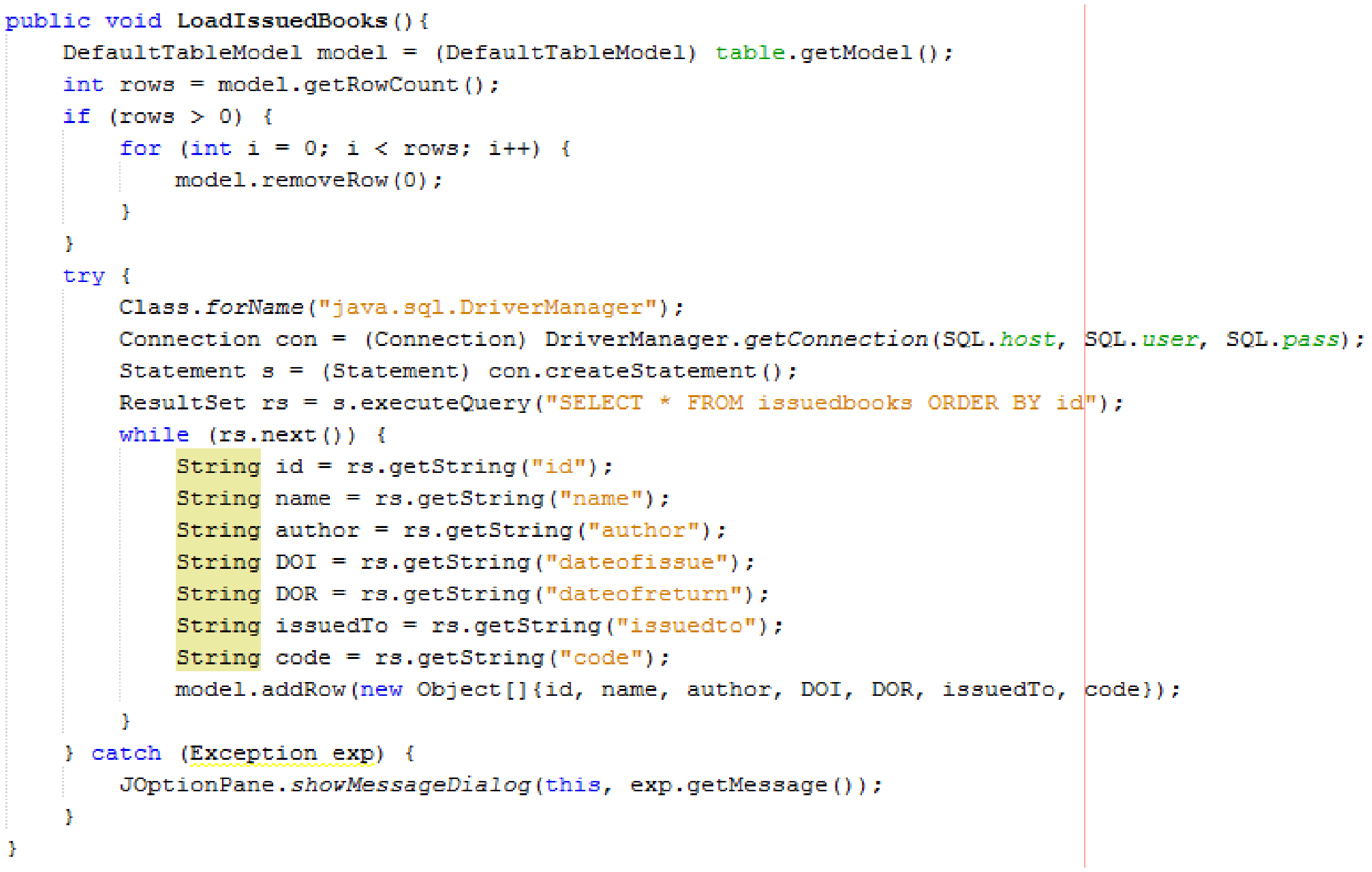
```
private void reissueActionPerformed(java.awt.event.ActionEvent evt) {
    reissueBook reissue = new reissueBook();
    reissue.setVisible(true);
}
```

issuedBooks -> On click event of **close**



```
private void closeActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```

issuedBooks -> **LoadIssuedBooks()** Method



```
public void LoadIssuedBooks(){
    DefaultTableModel model = (DefaultTableModel) table.getModel();
    int rows = model.getRowCount();
    if (rows > 0) {
        for (int i = 0; i < rows; i++) {
            model.removeRow(0);
        }
    }
    try {
        Class.forName("java.sql.DriverManager");
        Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
        Statement s = (Statement) con.createStatement();
        ResultSet rs = s.executeQuery("SELECT * FROM issuedbooks ORDER BY id");
        while (rs.next()) {
            String id = rs.getString("id");
            String name = rs.getString("name");
            String author = rs.getString("author");
            String DOI = rs.getString("dateofissue");
            String DOR = rs.getString("dateofreturn");
            String issuedTo = rs.getString("issuedto");
            String code = rs.getString("code");
            model.addRow(new Object[]{id, name, author, DOI, DOR, issuedTo, code});
        }
    } catch (Exception exp) {
        JOptionPane.showMessageDialog(this, exp.getMessage());
    }
}
```

Issued Books Frame

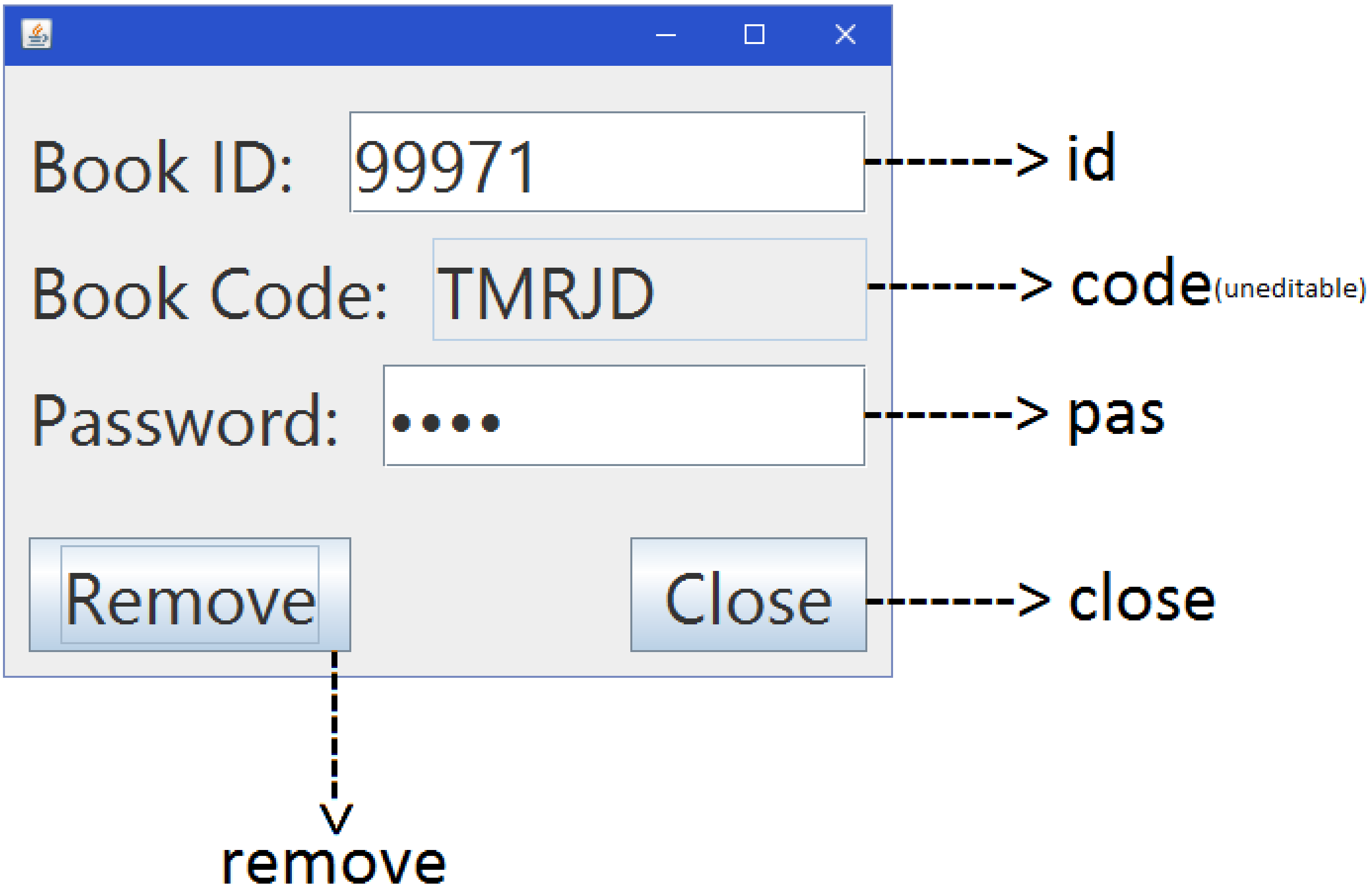
Child Frame – **Remove Issued Book**

Description: This frame is used to **remove issued books** from database ("libri") and set them as available. User just have to provide the book id and password to remove a book.

Import Statements:

```
import java.sql.*;
import javax.swing.*;
```

removeIssuedBook frame design

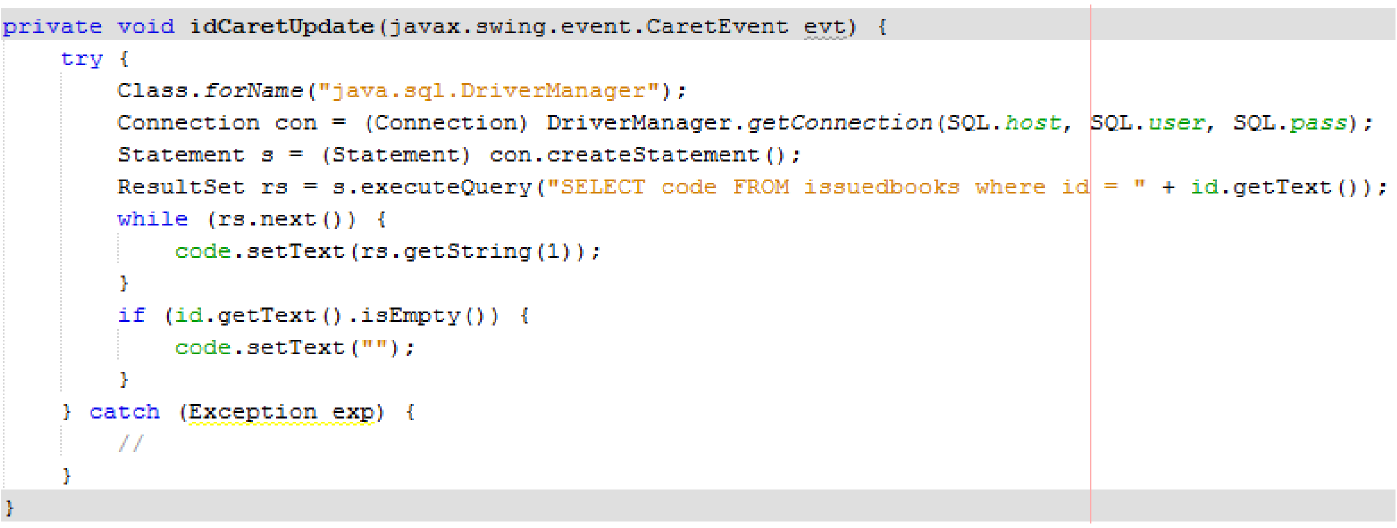


Book ID: 99971

Book Code: TMRJD

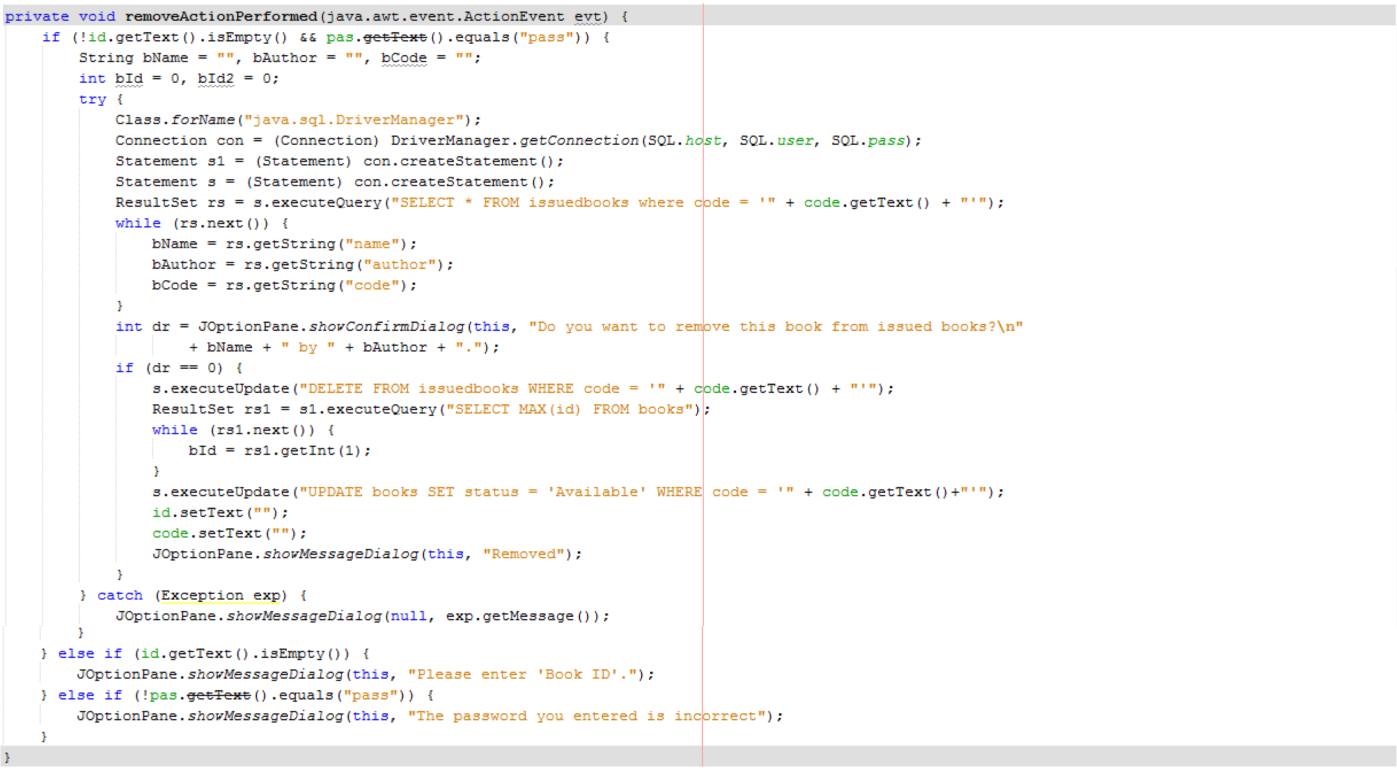
Password: ••••

removeIssuedBook -> On caret update event of id



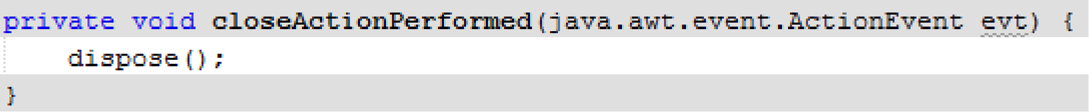
```
private void idCaretUpdate(javax.swing.event.CaretEvent evt) {
    try {
        Class.forName("java.sql.DriverManager");
        Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
        Statement s = (Statement) con.createStatement();
        ResultSet rs = s.executeQuery("SELECT code FROM issuedbooks where id = " + id.getText());
        while (rs.next()) {
            code.setText(rs.getString(1));
        }
        if (id.getText().isEmpty()) {
            code.setText("");
        }
    } catch (Exception exp) {
        //
    }
}
```

removeIssuedBook -> On click event of **remove**

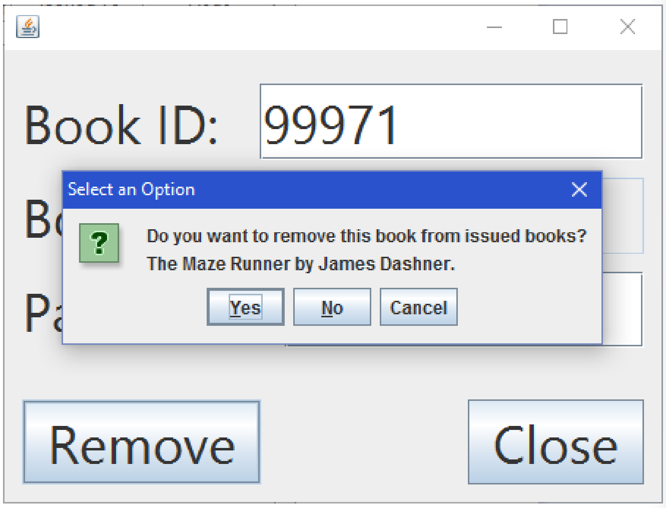


```
private void removeActionPerformed(java.awt.event.ActionEvent evt) {
    if (!id.getText().isEmpty() && pas.~~getText~~().equals("pass")) {
        String bName = "", bAuthor = "", bCode = "";
        int bId = 0, bId2 = 0;
        try {
            Class.forName("java.sql.DriverManager");
            Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
            Statement s1 = (Statement) con.createStatement();
            Statement s = (Statement) con.createStatement();
            ResultSet rs = s.executeQuery("SELECT * FROM issuedbooks where code = '" + code.getText() + "'");
            while (rs.next()) {
                bName = rs.getString("name");
                bAuthor = rs.getString("author");
                bCode = rs.getString("code");
            }
            int dr = JOptionPane.showConfirmDialog(this, "Do you want to remove this book from issued books?\n"
                    + bName + " by " + bAuthor + ".");
            if (dr == 0) {
                s.executeUpdate("DELETE FROM issuedbooks WHERE code = '" + code.getText() + "'");
                ResultSet rs1 = s1.executeQuery("SELECT MAX(id) FROM books");
                while (rs1.next()) {
                    bId = rs1.getInt(1);
                }
                s.executeUpdate("UPDATE books SET status = 'Available' WHERE code = '" + code.getText()+"'");
                id.setText("");
                code.setText("");
                JOptionPane.showMessageDialog(this, "Removed");
            }
        } catch (Exception exp) {
            JOptionPane.showMessageDialog(null, exp.getMessage());
        }
    } else if (id.getText().isEmpty()) {
        JOptionPane.showMessageDialog(this, "Please enter 'Book ID'.");
    } else if (!pas.~~getText~~().equals("pass")) {
        JOptionPane.showMessageDialog(this, "The password you entered is incorrect");
    }
}
```

removeIssuedBook -> On click event of **close**



```
private void closeActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```



Book ID: 99971

Select an Option

Do you want to remove this book from issued books?
The Maze Runner by James Dashner.

On click event of remove, libri confirms the delete.

Issued Books Frame

Child Frame – **Re-issue Book**

Description: This frame is used to **change issued book's period of issue** from database ("libri"). User just have to provide the issued book's id to re-issue it.

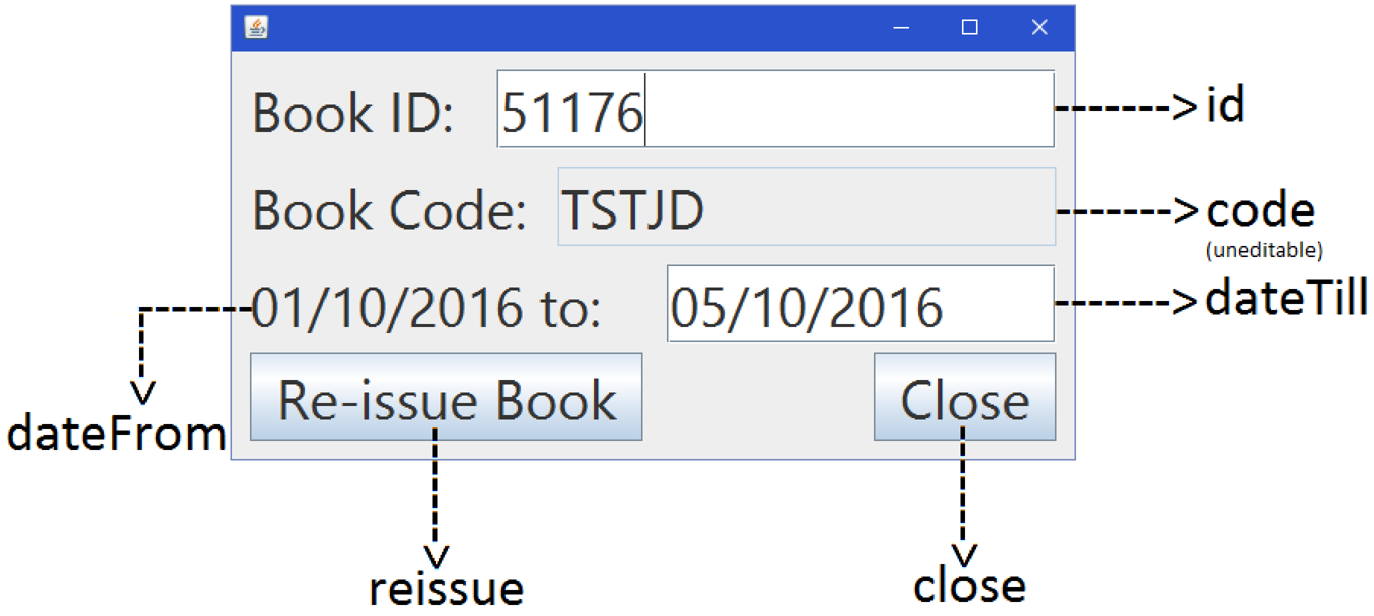
Import Statements:

```
import java.sql.*;
import javax.swing.*;
```

Global Variables:

```
String oldDate;
```

reissueBook frame design

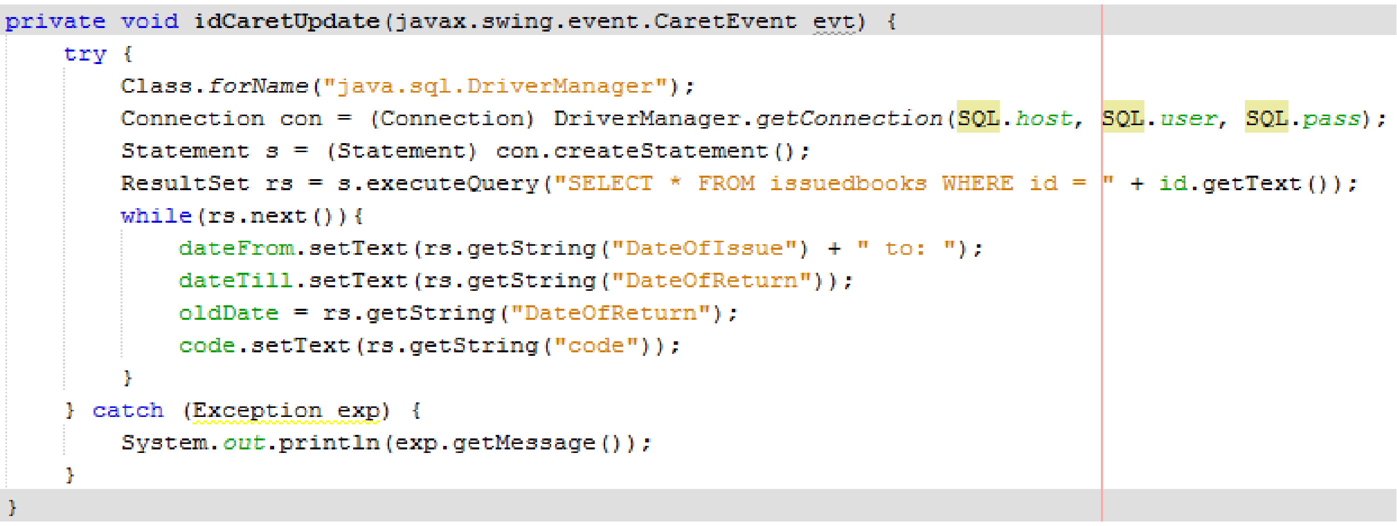


Book ID: 51176

Book Code: TSTJD

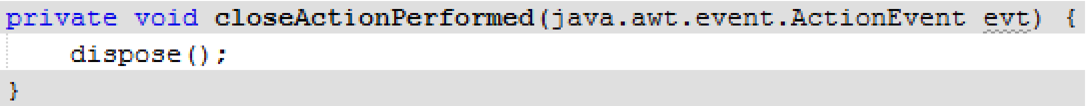
01/10/2016 to: 05/10/2016

reissueBook -> On caret update event of id



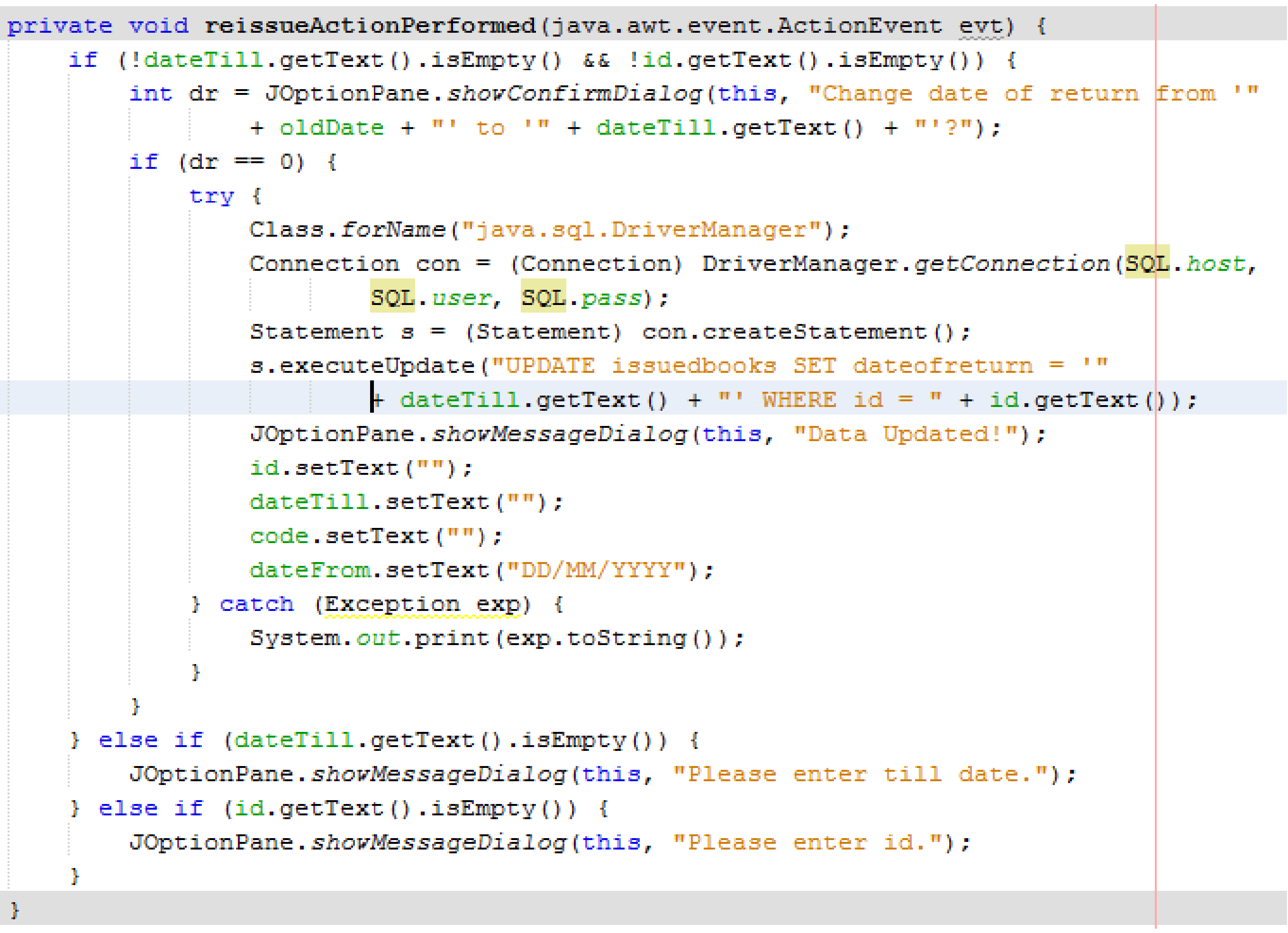
```
private void idCaretUpdate(javax.swing.event.CaretEvent evt) {
    try {
        Class.forName("java.sql.DriverManager");
        Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
        Statement s = (Statement) con.createStatement();
        ResultSet rs = s.executeQuery("SELECT * FROM issuedbooks WHERE id = " + id.getText());
        while(rs.next()){
            dateFrom.setText(rs.getString("DateOfIssue") + " to: ");
            dateTill.setText(rs.getString("DateOfReturn"));
            oldDate = rs.getString("DateOfReturn");
            code.setText(rs.getString("code"));
        }
    } catch (Exception exp) {
        System.out.println(exp.getMessage());
    }
}
```

reissueBook -> On click event of close

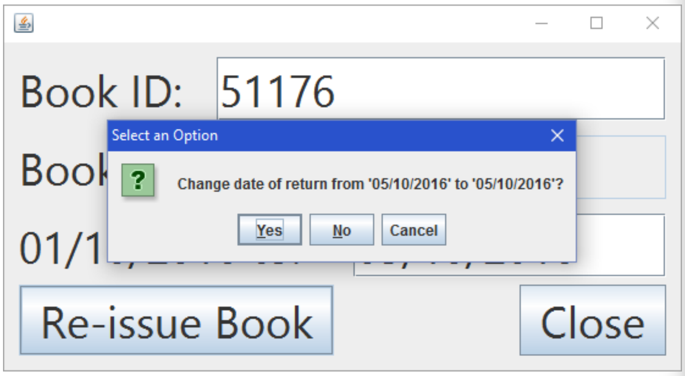


```
private void closeActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```

reissueBook -> On click event of reissue



```
private void reissueActionPerformed(java.awt.event.ActionEvent evt) {
    if (!dateTill.getText().isEmpty() && !id.getText().isEmpty()) {
        int dr = JOptionPane.showConfirmDialog(this, "Change date of return from '"
                + oldDate + "' to '" + dateTill.getText() + "'?");
        if (dr == 0) {
            try {
                Class.forName("java.sql.DriverManager");
                Connection con = (Connection) DriverManager.getConnection(SQL.host,
                        SQL.user, SQL.pass);
                Statement s = (Statement) con.createStatement();
                s.executeUpdate("UPDATE issuedbooks SET dateofreturn = '"
                        + dateTill.getText() + "' WHERE id = " + id.getText());
                JOptionPane.showMessageDialog(this, "Data Updated!");
                id.setText("");
                dateTill.setText("");
                code.setText("");
                dateFrom.setText("DD/MM/YYYY");
            } catch (Exception exp) {
                System.out.print(exp.toString());
            }
        }
    } else if (dateTill.getText().isEmpty()) {
        JOptionPane.showMessageDialog(this, "Please enter till date.");
    } else if (id.getText().isEmpty()) {
        JOptionPane.showMessageDialog(this, "Please enter id.");
    }
}
```



Book ID: 51176

Select an Option

Change date of return from '05/10/2016' to '05/10/2016'?

On click event of reissue, libri confirms the change.

Project

Source Code

(Part Four – Members)

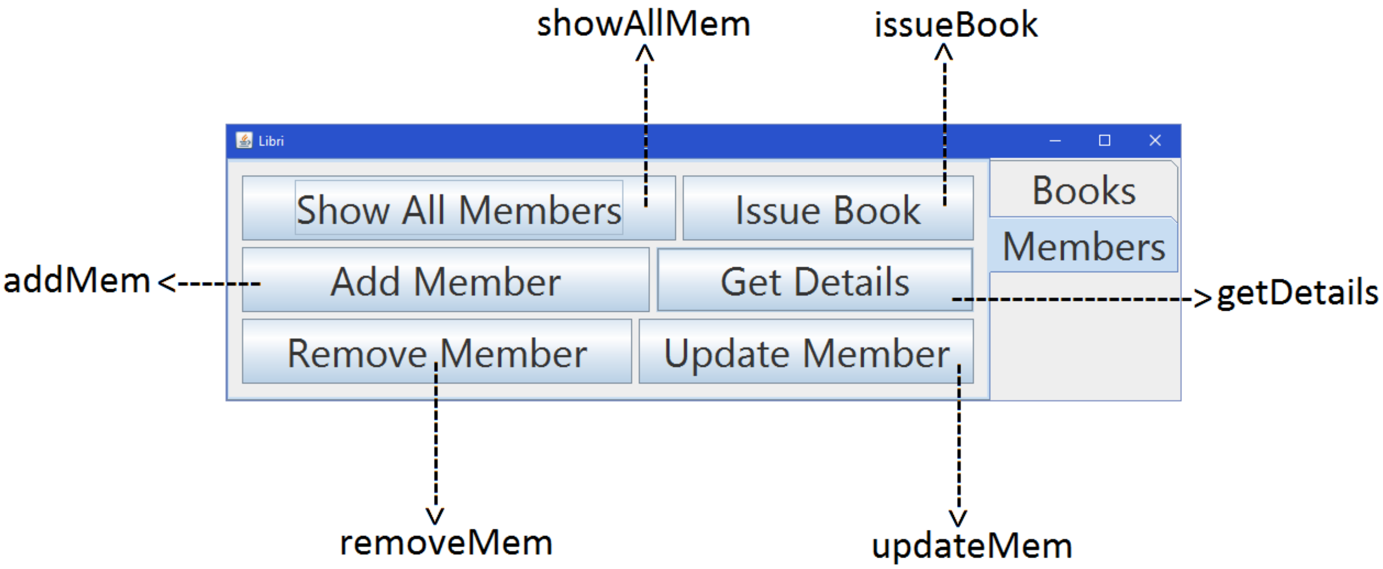
Description: There are **two tabs (JTabbedPane)** which separates **Books** and **Member** functions on frame.

In **Members tab** there are **six** frames which it can redirect to by clicking on buttons which are:

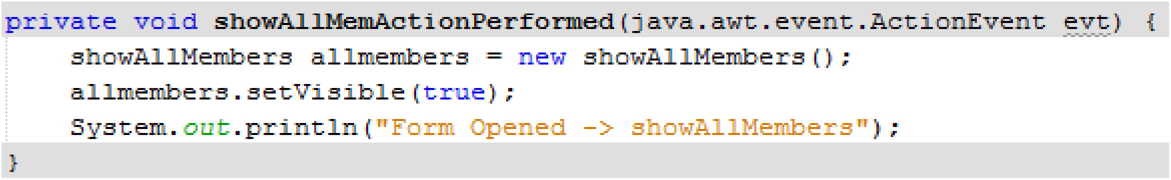
1. Show All Members	->	**showAllMembers**.java
2. Issue Book	->	**issueBook**.java
3. Add Member	->	**addMember**.java
4. Get Details	->	**getDetails**.java
5. Remove Member	->	**removeMember**.java

6. Update Member -> **updateMember**.java

Members tab design

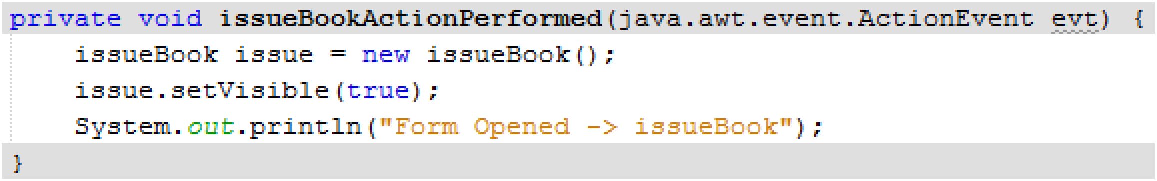


On click event of **showAllMem**



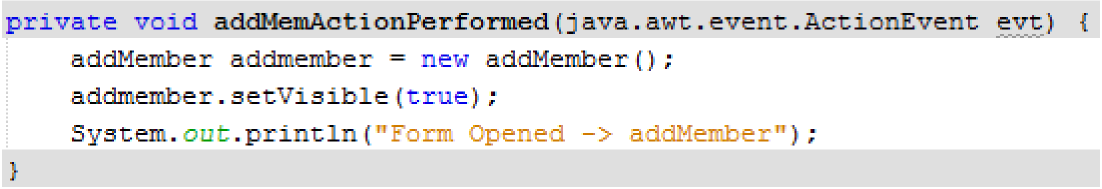
```
private void showAllMemActionPerformed(java.awt.event.ActionEvent evt) {
    showAllMembers allmembers = new showAllMembers();
    allmembers.setVisible(true);
    System.out.println("Form Opened -> showAllMembers");
}
```

On click event of **issueBook**



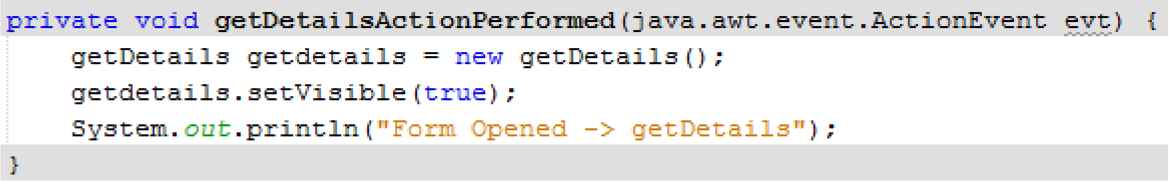
```
private void issueBookActionPerformed(java.awt.event.ActionEvent evt) {
    issueBook issue = new issueBook();
    issue.setVisible(true);
    System.out.println("Form Opened -> issueBook");
}
```

On click event of **addMem**



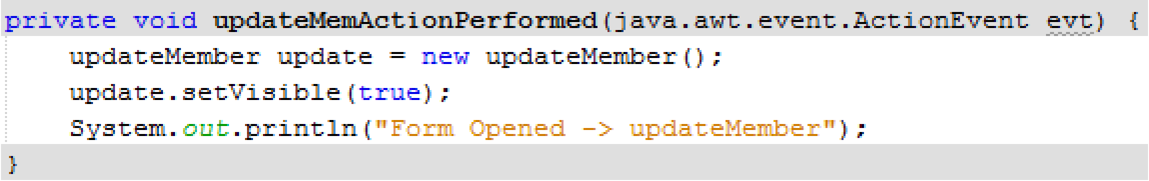
```
private void addMemActionPerformed(java.awt.event.ActionEvent evt) {
    addMember addmember = new addMember();
    addmember.setVisible(true);
    System.out.println("Form Opened -> addMember");
}
```

On click event of **getDetails**



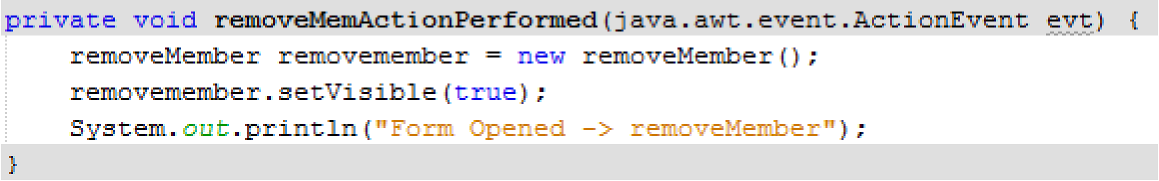
```
private void getDetailsActionPerformed(java.awt.event.ActionEvent evt) {
    getDetails getdetails = new getDetails();
    getdetails.setVisible(true);
    System.out.println("Form Opened -> getDetails");
}
```

On click event of **removeMem**



```
private void updateMemActionPerformed(java.awt.event.ActionEvent evt) {
    updateMember update = new updateMember();
    update.setVisible(true);
    System.out.println("Form Opened -> updateMember");
}
```

On click event of **updateMem**



```
private void removeMemActionPerformed(java.awt.event.ActionEvent evt) {
    removeMember removemember = new removeMember();
    removemember.setVisible(true);
    System.out.println("Form Opened -> removeMember");
}
```

Show All Members Frame

Description: This frame is used to **show all members** in database (“libri”).

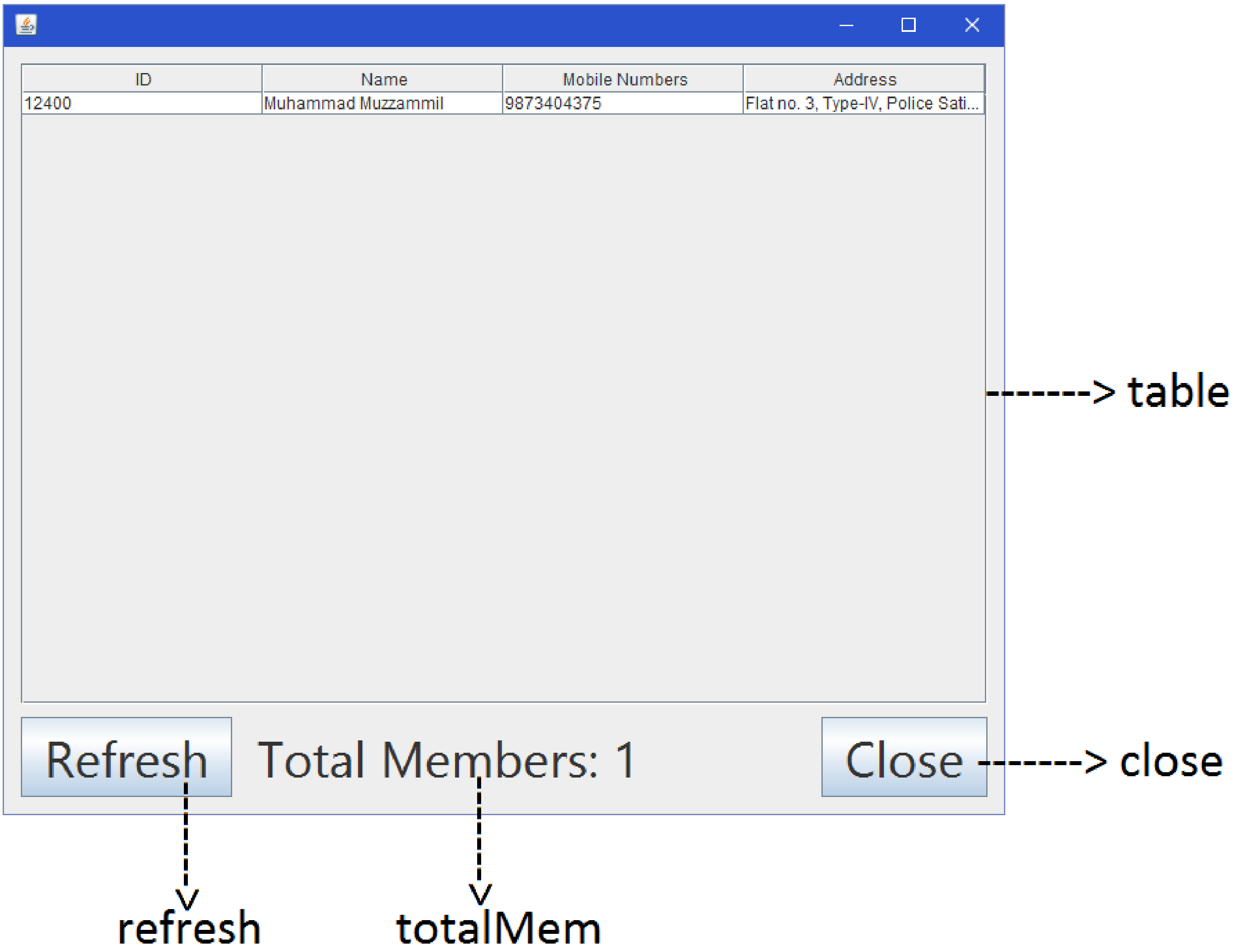
Import Statements:

```
import java.sql.*;

import javax.swing.*;

import javax.swing.table.DefaultTableModel;
```

Custom Method:

```
public void LoadMembers()
```

showAllMembers frame design



ID	Name	Mobile Numbers	Address
12400	Muhammad Muzzammil	9873404375	Flat no. 3, Type-IV, Police Sati...

On frame load event

```
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    LoadMembers();
}
```

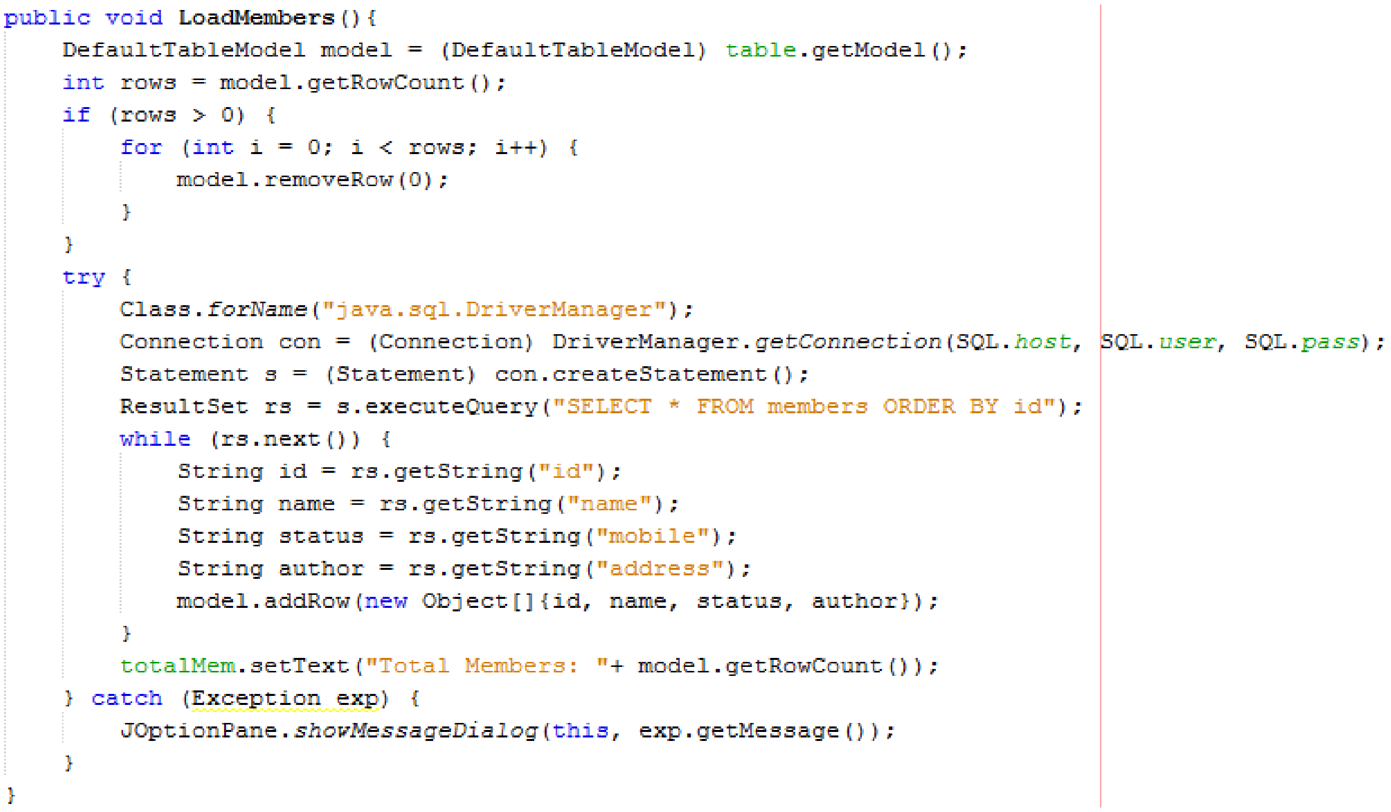
On click event of refresh

```
private void refreshActionPerformed(java.awt.event.ActionEvent evt) {
    LoadMembers();
}
```

On click event of close

```
private void closeActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```

LoadMembers() method



```
public void LoadMembers(){
    DefaultTableModel model = (DefaultTableModel) table.getModel();
    int rows = model.getRowCount();
    if (rows > 0) {
        for (int i = 0; i < rows; i++) {
            model.removeRow(0);
        }
    }
    try {
        Class.forName("java.sql.DriverManager");
        Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
        Statement s = (Statement) con.createStatement();
        ResultSet rs = s.executeQuery("SELECT * FROM members ORDER BY id");
        while (rs.next()) {
            String id = rs.getString("id");
            String name = rs.getString("name");
            String status = rs.getString("mobile");
            String author = rs.getString("address");
            model.addRow(new Object[]{id, name, status, author});
        }
        totalMem.setText("Total Members: "+ model.getRowCount());
    } catch (Exception exp) {
        JOptionPane.showMessageDialog(this, exp.getMessage());
    }
}
```

Issue Book Frame

Description: This frame is used to **issue books** to the members. User have to provide book id, member id and return date to do so…

Import Statements:

```
import java.sql.*;
import javax.swing.*;
import java.util.Date;
import java.util.Random;
import java.text.*;
```

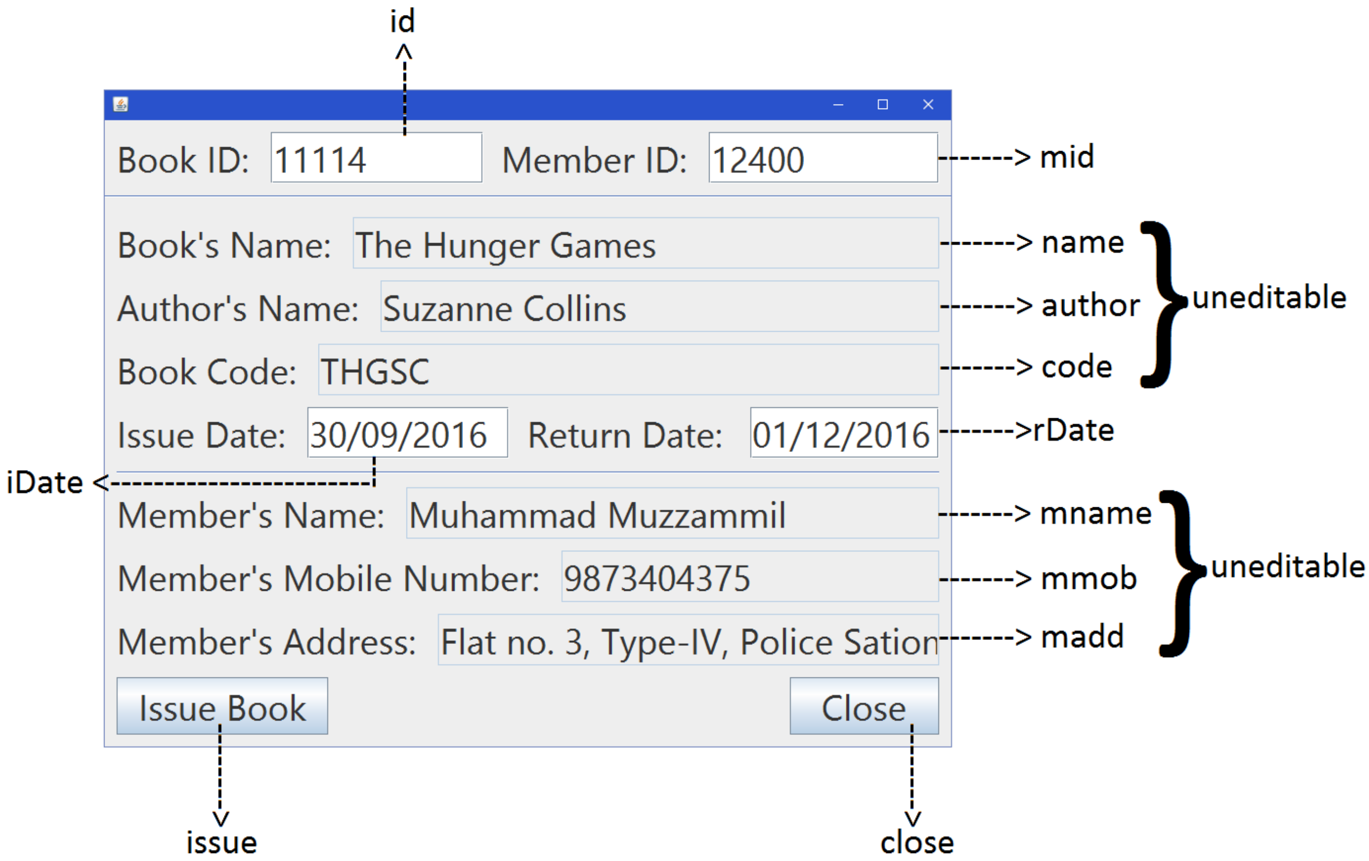
Global Variables:

```
String status;

int iID;
```

Custom Method:

```
public boolean bookIssued()

public boolean memberIssued()
```

issueBook frame design



Book ID: 11114 Member ID: 12400

Book's Name: The Hunger Games

Author's Name: Suzanne Collins

Book Code: THGSC

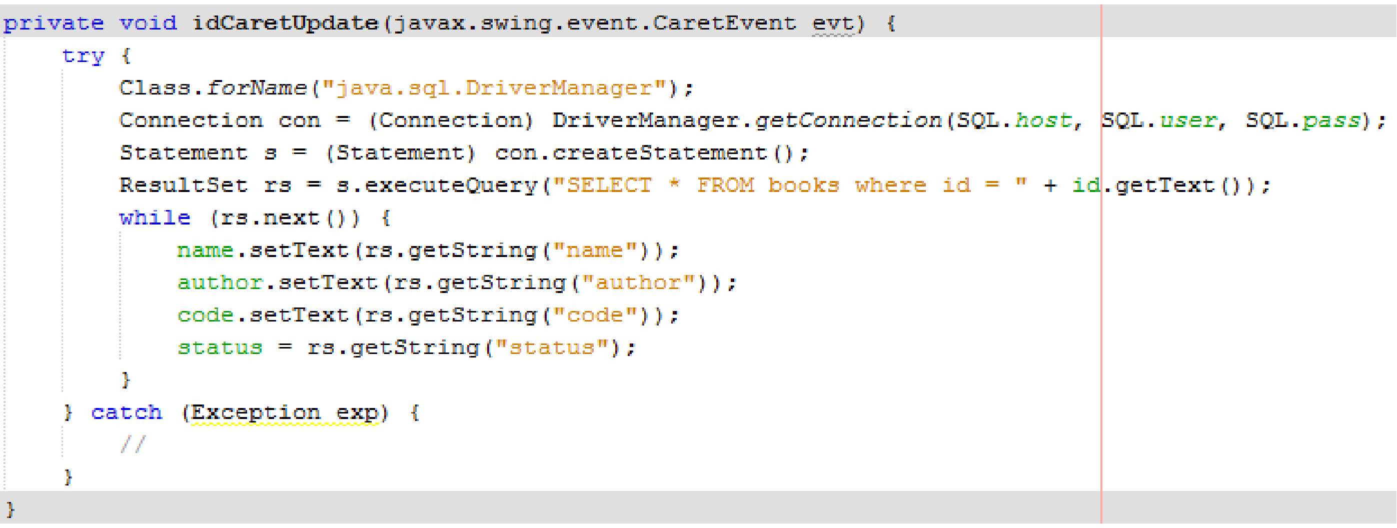
Issue Date: 30/09/2016 Return Date: 01/12/2016

Member's Name: Muhammad Muzzammil

Member's Mobile Number: 9873404375

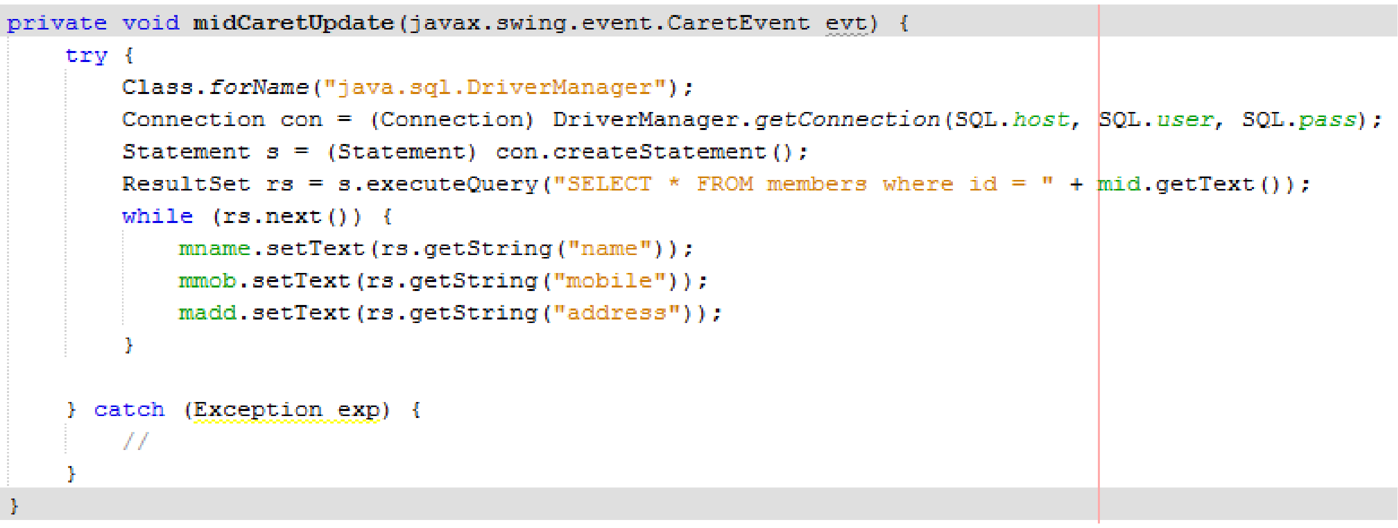
Member's Address: Flat no. 3, Type-IV, Police Sation

On caret update event of **id**



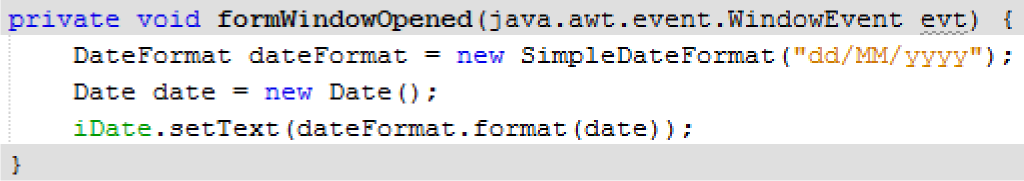
```
private void idCaretUpdate(javax.swing.event.CaretEvent evt) {
    try {
        Class.forName("java.sql.DriverManager");
        Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
        Statement s = (Statement) con.createStatement();
        ResultSet rs = s.executeQuery("SELECT * FROM books where id = " + id.getText());
        while (rs.next()) {
            name.setText(rs.getString("name"));
            author.setText(rs.getString("author"));
            code.setText(rs.getString("code"));
            status = rs.getString("status");
        }
    } catch (Exception exp) {
        //
    }
}
```

On caret update event of **mid**



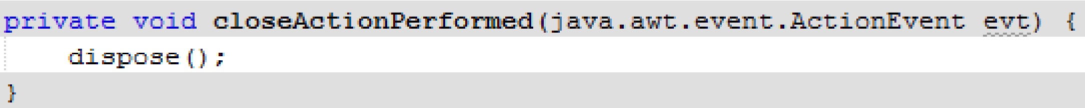
```
private void midCaretUpdate(javax.swing.event.CaretEvent evt) {
    try {
        Class.forName("java.sql.DriverManager");
        Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
        Statement s = (Statement) con.createStatement();
        ResultSet rs = s.executeQuery("SELECT * FROM members where id = " + mid.getText());
        while (rs.next()) {
            mname.setText(rs.getString("name"));
            mmob.setText(rs.getString("mobile"));
            madd.setText(rs.getString("address"));
        }

    } catch (Exception exp) {
        //
    }
}
```

On **frame** load event



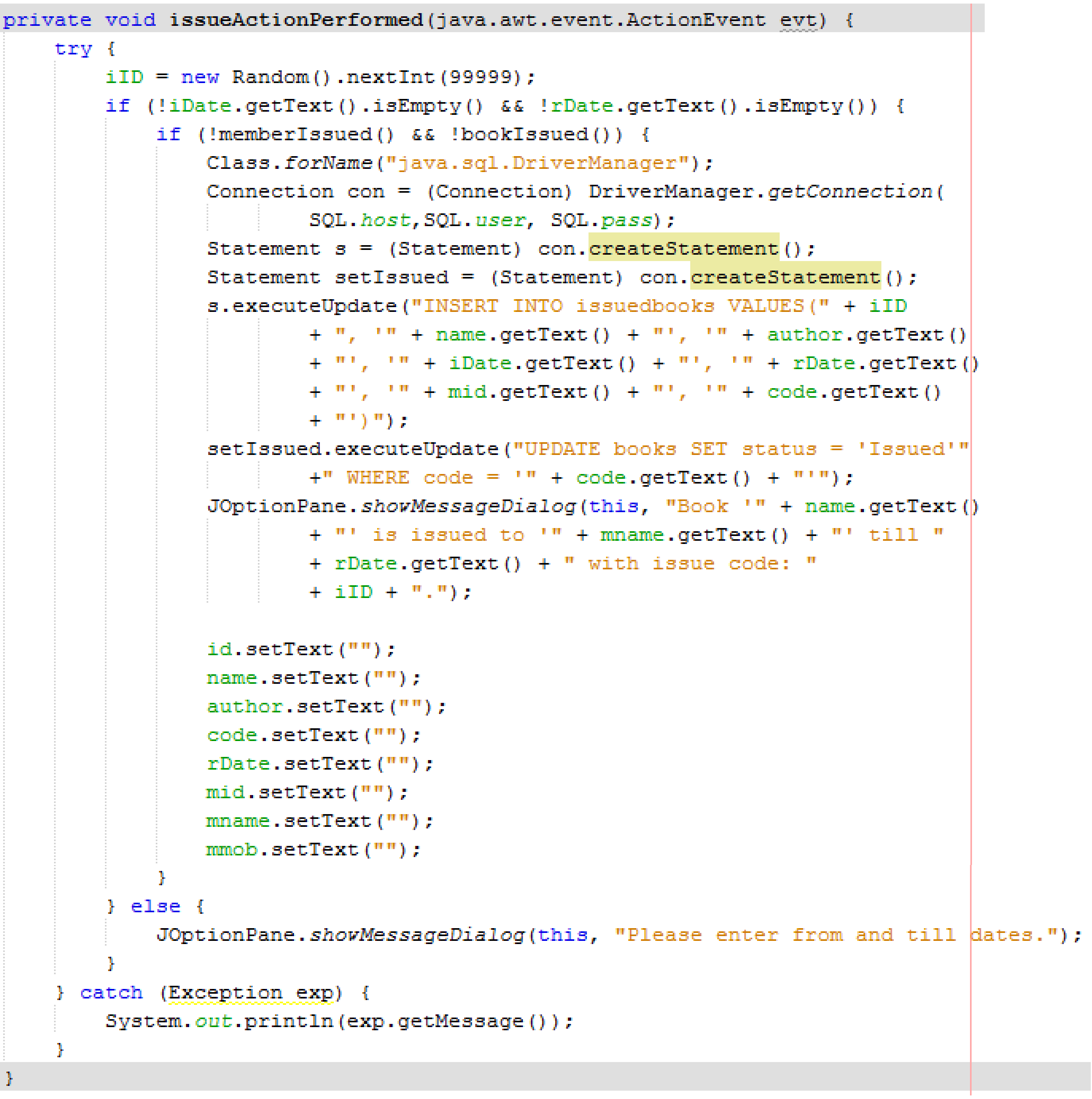
```
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
    Date date = new Date();
    iDate.setText(dateFormat.format(date));
}
```

On click event of **close**



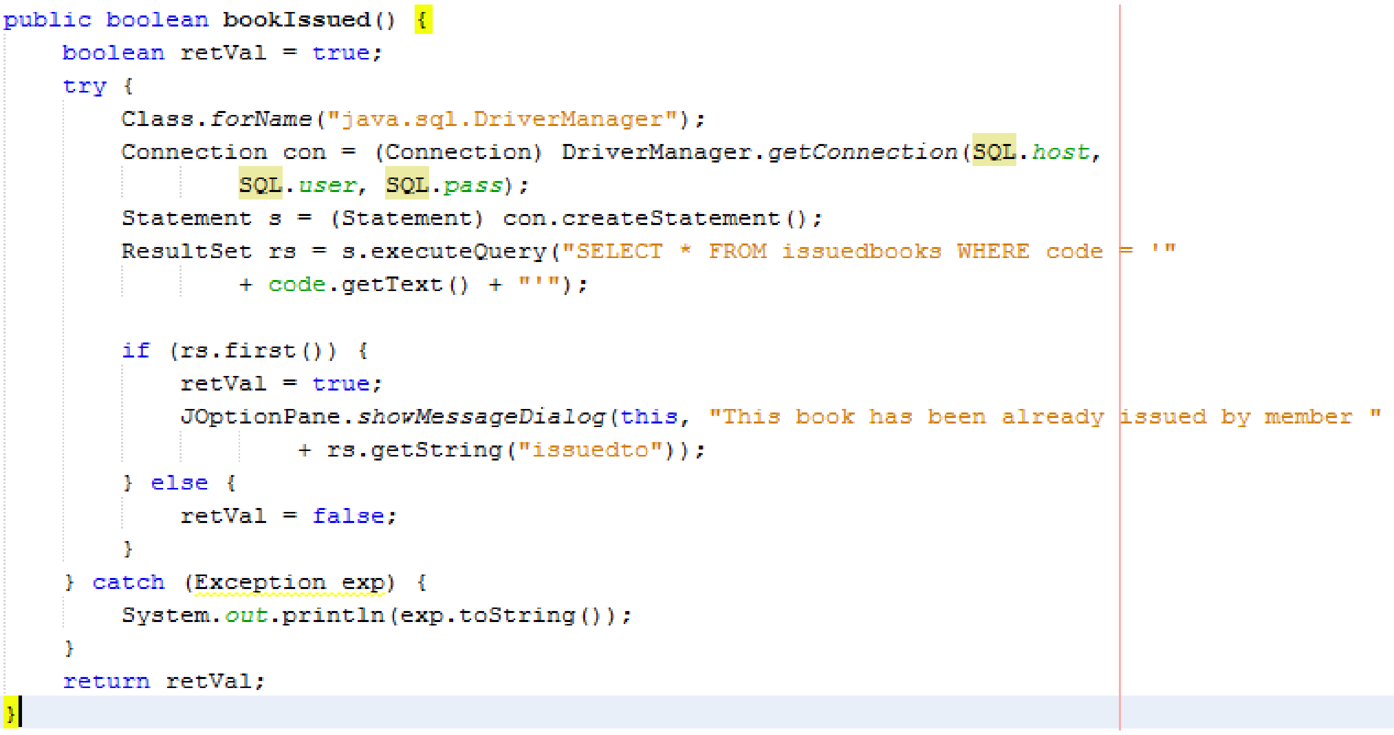
```
private void closeActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```

On click event of **issue**



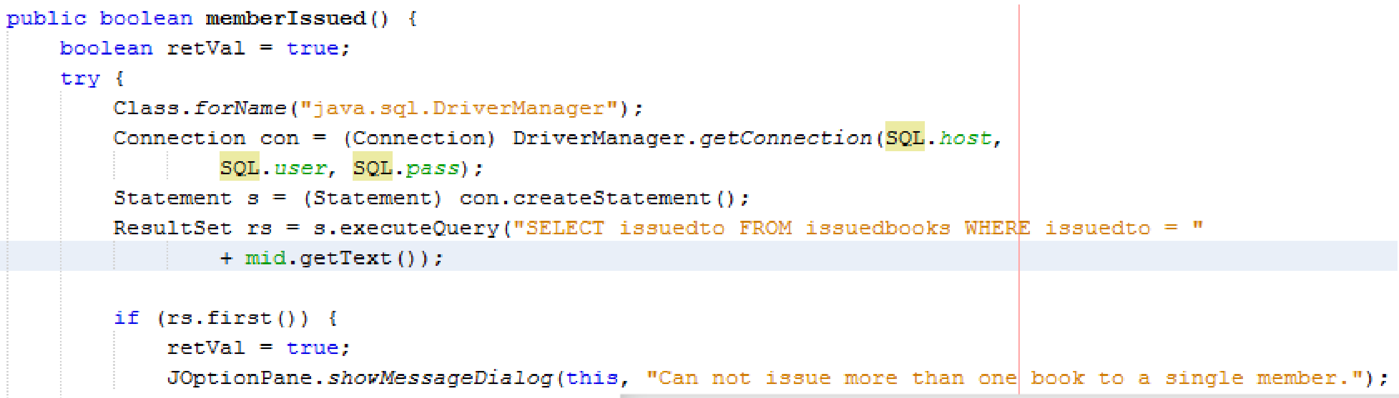
```
private void issueActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        iID = new Random().nextInt(99999);
        if (!iDate.getText().isEmpty() && !rDate.getText().isEmpty()) {
            if (!memberIssued() && !bookIssued()) {
                Class.forName("java.sql.DriverManager");
                Connection con = (Connection) DriverManager.getConnection(
                        SQL.host,SQL.user, SQL.pass);
                Statement s = (Statement) con.createStatement();
                Statement setIssued = (Statement) con.createStatement();
                s.executeUpdate("INSERT INTO issuedbooks VALUES(" + iID
                        + ", '" + name.getText() + "', '" + author.getText()
                        + "', '" + iDate.getText() + "', '" + rDate.getText()
                        + "', '" + mid.getText() + "', '" + code.getText()
                        + "')");
                setIssued.executeUpdate("UPDATE books SET status = 'Issued'"
                        +" WHERE code = '" + code.getText() + "'");
                JOptionPane.showMessageDialog(this, "Book '" + name.getText()
                        + "' is issued to '" + mname.getText() + "' till "
                        + rDate.getText() + " with issue code: "
                        + iID + ".");

                id.setText("");
                name.setText("");
                author.setText("");
                code.setText("");
                rDate.setText("");
                mid.setText("");
                mname.setText("");
                mmob.setText("");
            }
        } else {
            JOptionPane.showMessageDialog(this, "Please enter from and till dates.");
        }
    } catch (Exception exp) {
        System.out.println(exp.getMessage());
    }
}
```

bookIssued() method

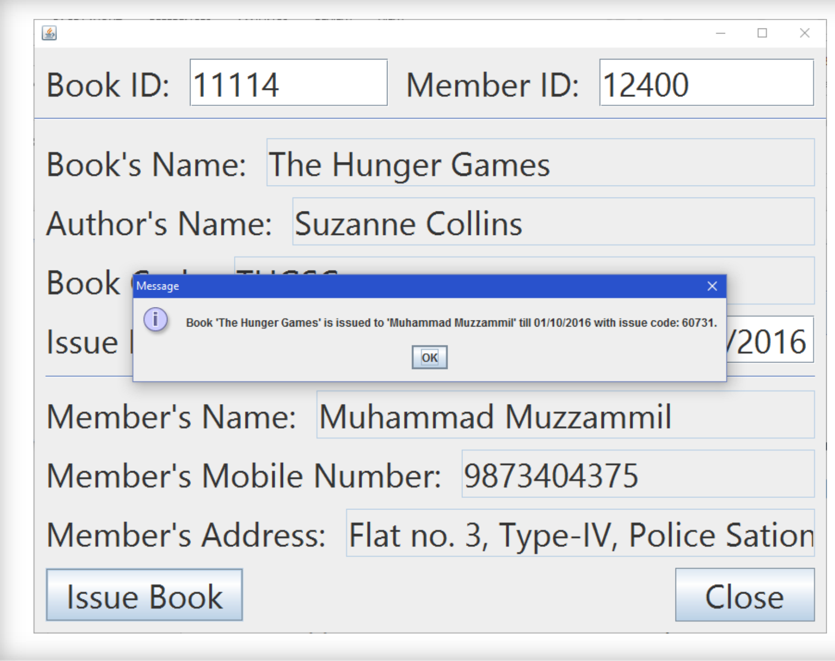


```
public boolean bookIssued() {
    boolean retVal = true;
    try {
        Class.forName("java.sql.DriverManager");
        Connection con = (Connection) DriverManager.getConnection(SQL.host,
                SQL.user, SQL.pass);
        Statement s = (Statement) con.createStatement();
        ResultSet rs = s.executeQuery("SELECT * FROM issuedbooks WHERE code = '"
                + code.getText() + "'");

        if (rs.first()) {
            retVal = true;
            JOptionPane.showMessageDialog(this, "This book has been already issued by member "
                    + rs.getString("issuedto"));
        } else {
            retVal = false;
        }
    } catch (Exception exp) {
        System.out.println(exp.toString());
    }
    return retVal;
}
```

memberIssued() method



```
public boolean memberIssued() {
    boolean retVal = true;
    try {
        Class.forName("java.sql.DriverManager");
        Connection con = (Connection) DriverManager.getConnection(SQL.host,
                SQL.user, SQL.pass);
        Statement s = (Statement) con.createStatement();
        ResultSet rs = s.executeQuery("SELECT issuedto FROM issuedbooks WHERE issuedto = "
                + mid.getText());

        if (rs.first()) {
            retVal = true;
            JOptionPane.showMessageDialog(this, "Can not issue more than one book to a single member.");
        } else {
            retVal = false;
        }
    } catch (Exception exp) {
        System.out.println(exp.toString());
    }
    return retVal;
}
```



Book ID: 11114 Member ID: 12400

Book's Name: The Hunger Games

Author's Name: Suzanne Collins

Book C... TH...

Issue ... /2016

Message

Book 'The Hunger Games' is issued to 'Muhammad Muzzammil' till 01/10/2016 with issue code: 60731.

Member's Name: Muhammad Muzzammil

Member's Mobile Number: 9873404375

Member's Address: Flat no. 3, Type-IV, Police Sation

"Book 'The hunger games' is issued to 'Muhammad Muzzammil' till 01/10/2016 with issue code: 60731."

Message box is shown when issue is clicked

Add Member Frame

Description: This frame is used to **add new members** to database ("libri"). User has to provide name, mobile number, which must be unique, and address of member. **Member ID is randomly generated** and is **unique**.

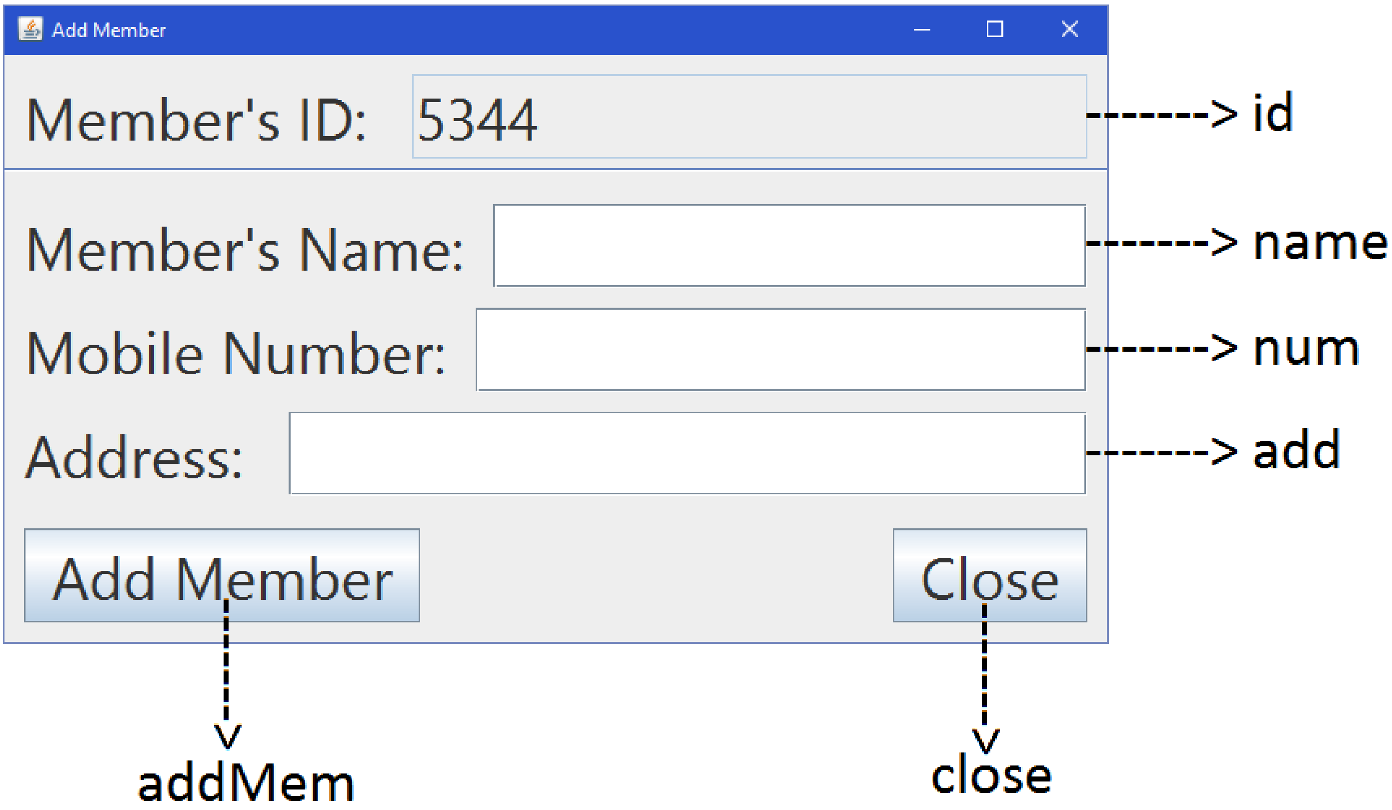
Import Statements:

```
import java.sql.*;
import javax.swing.*;
import java.util.Random;
```

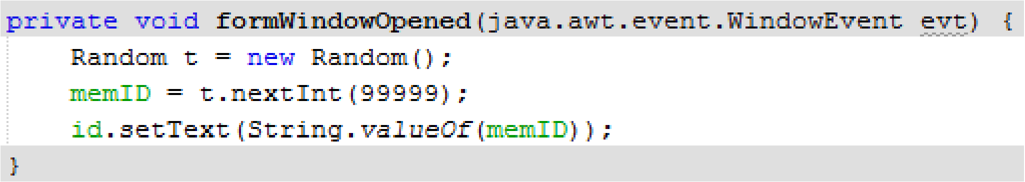
Global Variables:

```
int memID;
```

addMember frame design

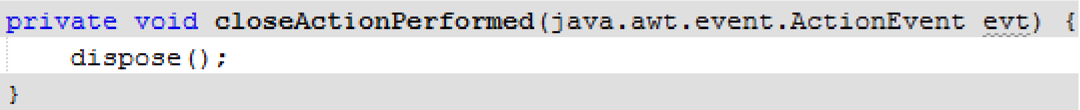


On frame load event



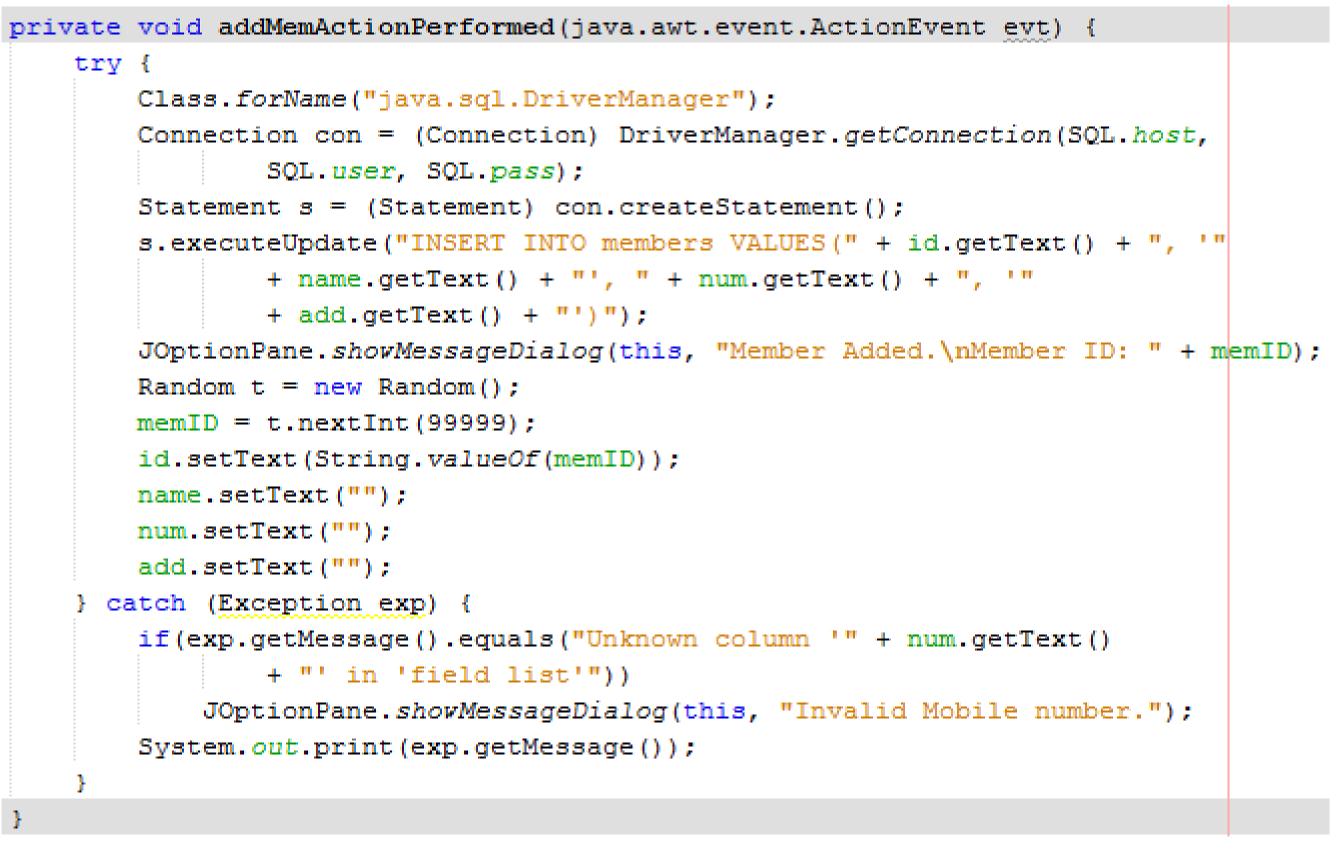
```
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    Random t = new Random();
    memID = t.nextInt(99999);
    id.setText(String.valueOf(memID));
}
```

On click event of close

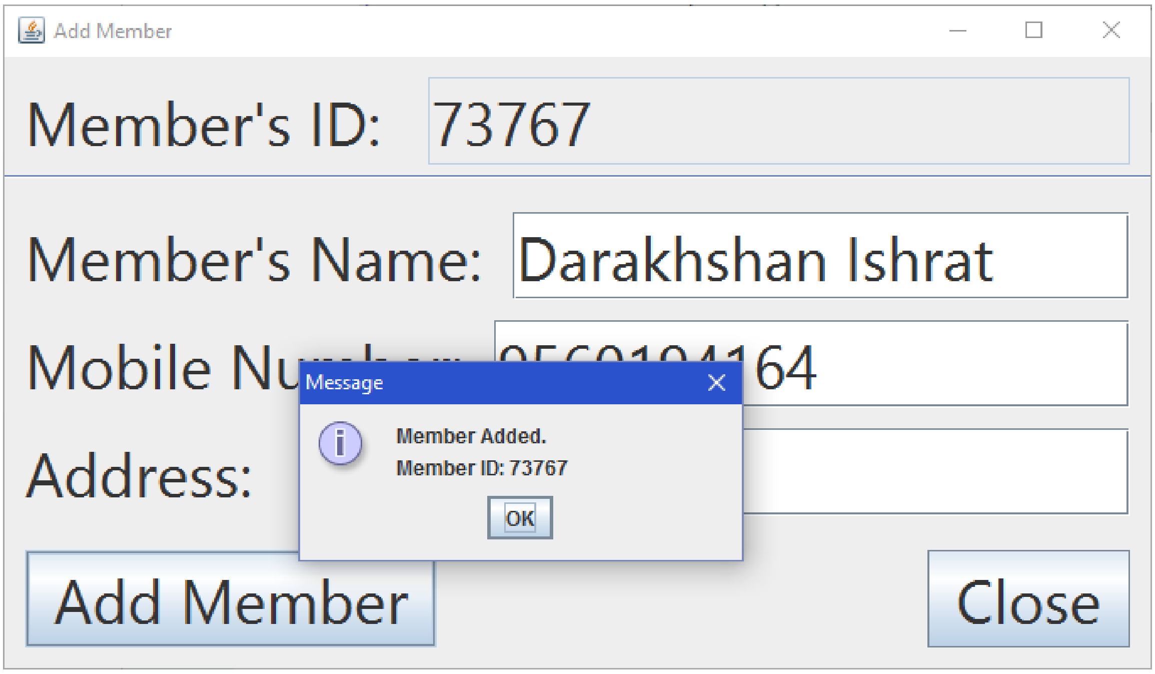


```
private void closeActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```

On click event of **addMem**



```
private void addMemActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Class.forName("java.sql.DriverManager");
        Connection con = (Connection) DriverManager.getConnection(SQL.host,
                SQL.user, SQL.pass);
        Statement s = (Statement) con.createStatement();
        s.executeUpdate("INSERT INTO members VALUES(" + id.getText() + ", '"
                + name.getText() + "', " + num.getText() + ", '"
                + add.getText() + "')");
        JOptionPane.showMessageDialog(this, "Member Added.\nMember ID: " + memID);
        Random t = new Random();
        memID = t.nextInt(99999);
        id.setText(String.valueOf(memID));
        name.setText("");
        num.setText("");
        add.setText("");
    } catch (Exception exp) {
        if(exp.getMessage().equals("Unknown column '" + num.getText()
                + "' in 'field list'"))
            JOptionPane.showMessageDialog(this, "Invalid Mobile number.");
        System.out.print(exp.getMessage());
    }
}
```



Member's ID: 73767

Member's Name: Darakhshan Ishrat

Mobile Nu 64

Address:

Member Added.
Member ID: 73767

"Member Added. Member ID: 73767"

Message box is shown when addMem is clicked

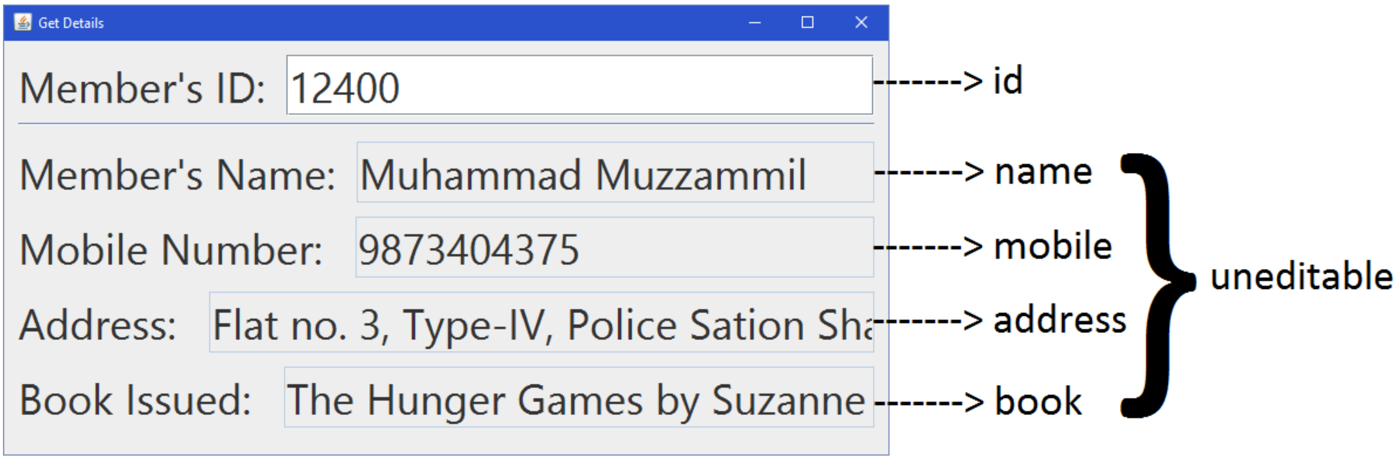
Get Details Frame

Description: This frame is used to **show details of a member** by entering member id. It will also show book which is issued by the member, if there is any.

Import Statements:

```
import java.sql.*;
```

getDetails frame design



Member's ID: 12400

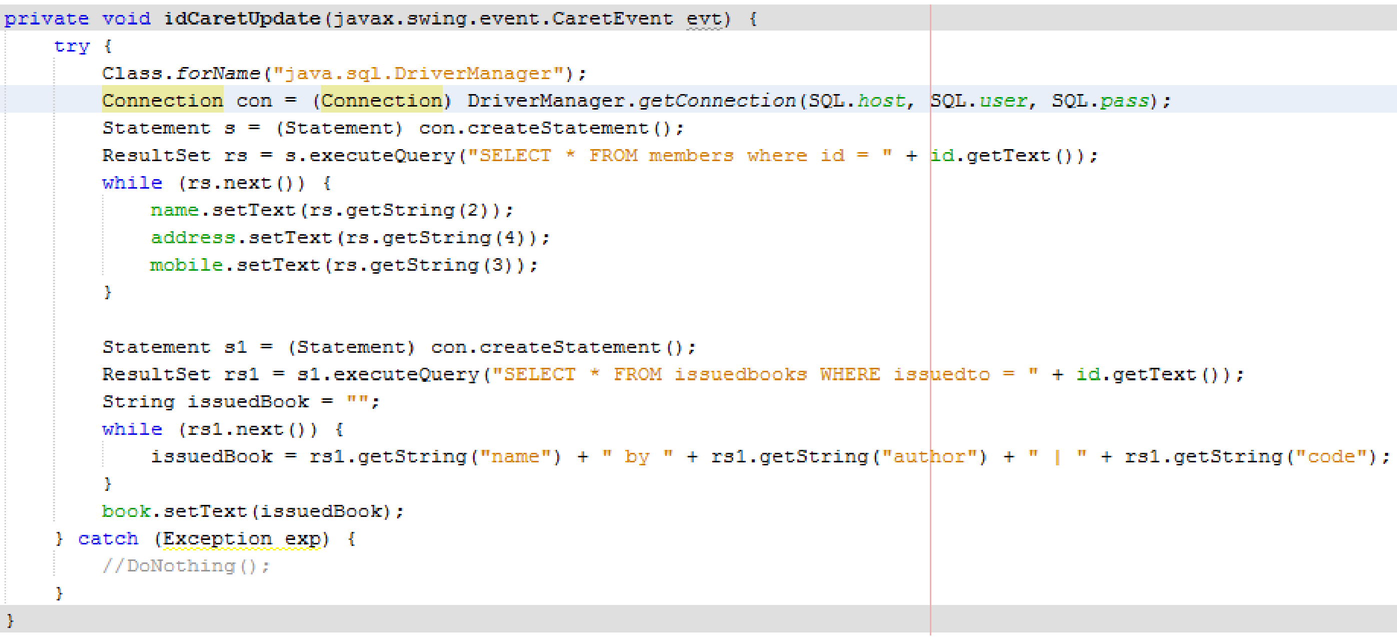
Member's Name: Muhammad Muzzammil

Mobile Number: 9873404375

Address: Flat no. 3, Type-IV, Police Sation Sha

Book Issued: The Hunger Games by Suzanne

On caret update event of **id**



```
private void idCaretUpdate(javax.swing.event.CaretEvent evt) {
    try {
        Class.forName("java.sql.DriverManager");
        Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
        Statement s = (Statement) con.createStatement();
        ResultSet rs = s.executeQuery("SELECT * FROM members where id = " + id.getText());
        while (rs.next()) {
            name.setText(rs.getString(2));
            address.setText(rs.getString(4));
            mobile.setText(rs.getString(3));
        }

        Statement s1 = (Statement) con.createStatement();
        ResultSet rs1 = s1.executeQuery("SELECT * FROM issuedbooks WHERE issuedto = " + id.getText());
        String issuedBook = "";
        while (rs1.next()) {
            issuedBook = rs1.getString("name") + " by " + rs1.getString("author") + " | " + rs1.getString("code");
        }
        book.setText(issuedBook);
    } catch (Exception exp) {
        //DoNothing();
    }
}
```

Remove Member Frame

Description: This frame is used to **remove a member** by entering member id. It will ask what to do with the book issued, if any.

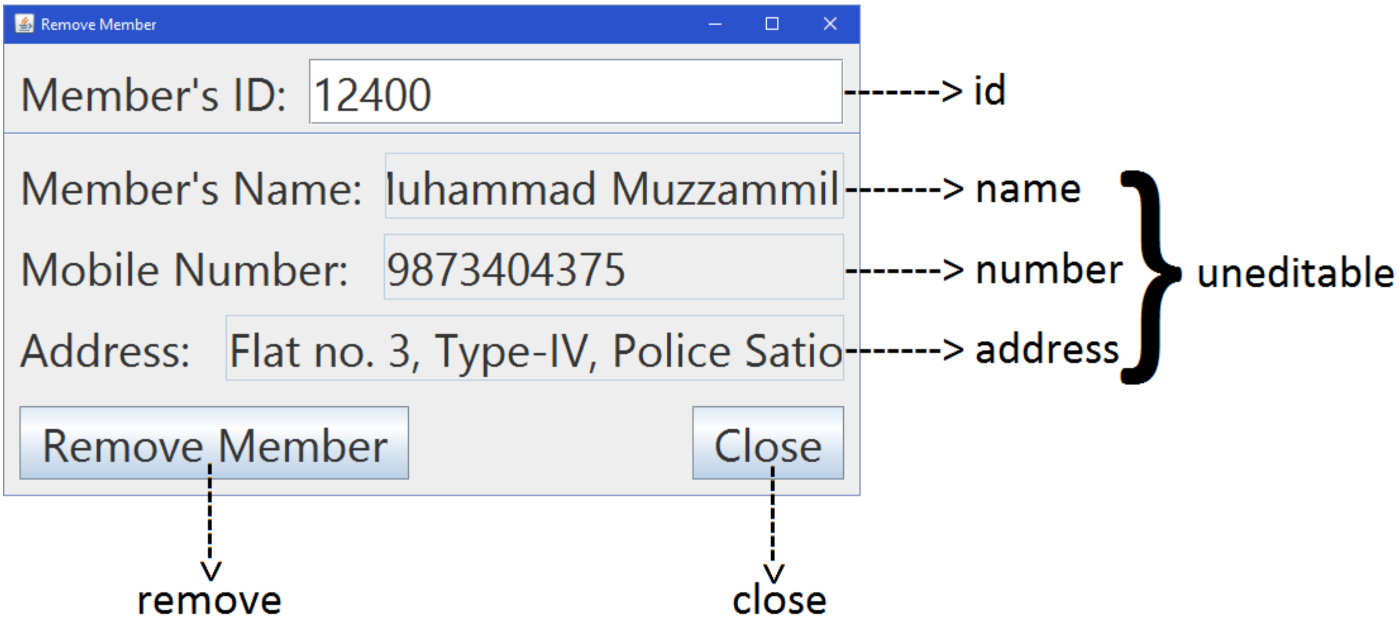
Import Statements:

```
import java.sql.*;
import java.swing.*;
```

Custom Method:

```
public void clr()
```

removeMember frame design



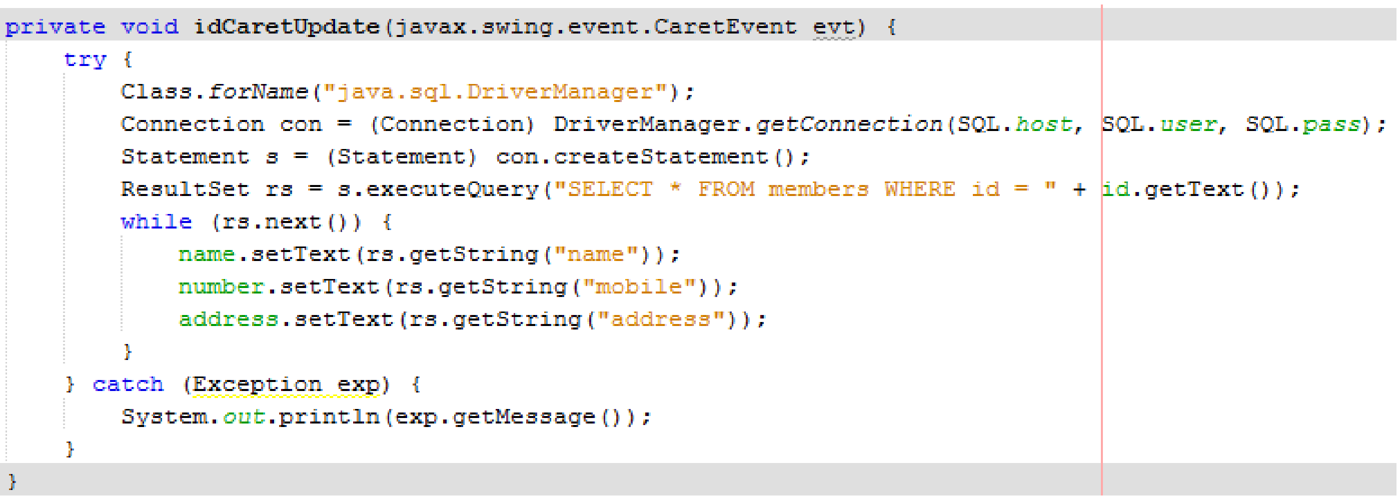
Member's ID: 12400

Member's Name: luhammad Muzzammil

Mobile Number: 9873404375

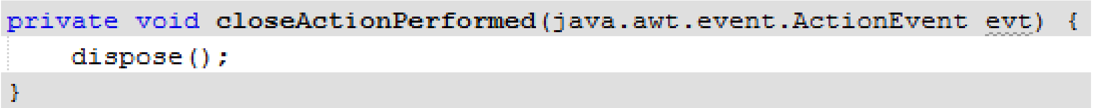
Address: Flat no. 3, Type-IV, Police Satio

On caret update event of id



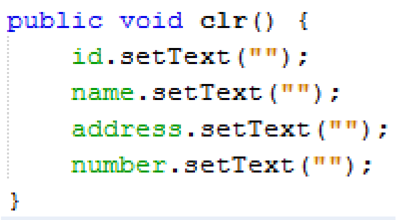
```
private void idCaretUpdate(javax.swing.event.CaretEvent evt) {
    try {
        Class.forName("java.sql.DriverManager");
        Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
        Statement s = (Statement) con.createStatement();
        ResultSet rs = s.executeQuery("SELECT * FROM members WHERE id = " + id.getText());
        while (rs.next()) {
            name.setText(rs.getString("name"));
            number.setText(rs.getString("mobile"));
            address.setText(rs.getString("address"));
        }
    } catch (Exception exp) {
        System.out.println(exp.getMessage());
    }
}
```

On click event of close



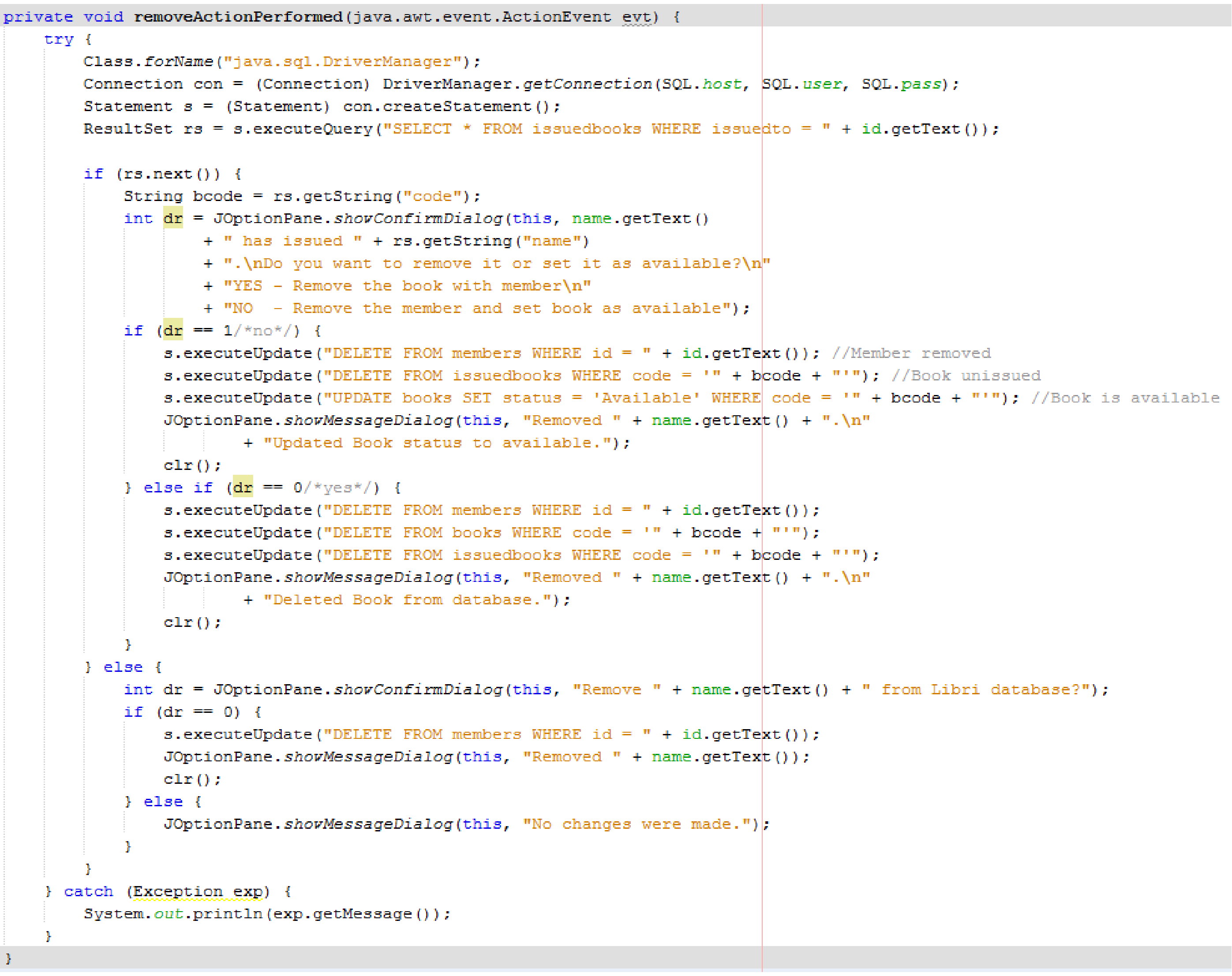
```
private void closeActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```

clr() method

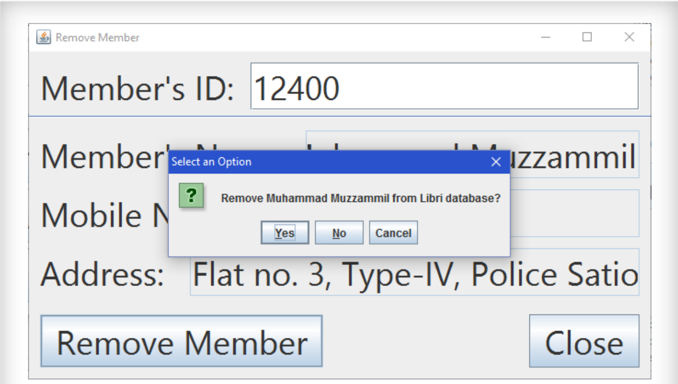


```
public void clr() {
    id.setText("");
    name.setText("");
    address.setText("");
    number.setText("");
}
```

On click event of **remove**



```java
private void removeActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Class.forName("java.sql.DriverManager");
        Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
        Statement s = (Statement) con.createStatement();
        ResultSet rs = s.executeQuery("SELECT * FROM issuedbooks WHERE issuedto = " + id.getText());

        if (rs.next()) {
            String bcode = rs.getString("code");
            int dr = JOptionPane.showConfirmDialog(this, name.getText()
                    + " has issued " + rs.getString("name")
                    + ".\nDo you want to remove it or set it as available?\n"
                    + "YES - Remove the book with member\n"
                    + "NO  - Remove the member and set book as available");
            if (dr == 1/*no*/) {
                s.executeUpdate("DELETE FROM members WHERE id = " + id.getText()); //Member removed
                s.executeUpdate("DELETE FROM issuedbooks WHERE code = '" + bcode + "'"); //Book unissued
                s.executeUpdate("UPDATE books SET status = 'Available' WHERE code = '" + bcode + "'"); //Book is available
                JOptionPane.showMessageDialog(this, "Removed " + name.getText() + ".\n"
                        + "Updated Book status to available.");
                clr();
            } else if (dr == 0/*yes*/) {
                s.executeUpdate("DELETE FROM members WHERE id = " + id.getText());
                s.executeUpdate("DELETE FROM books WHERE code = '" + bcode + "'");
                s.executeUpdate("DELETE FROM issuedbooks WHERE code = '" + bcode + "'");
                JOptionPane.showMessageDialog(this, "Removed " + name.getText() + ".\n"
                        + "Deleted Book from database.");
                clr();
            }
        } else {
            int dr = JOptionPane.showConfirmDialog(this, "Remove " + name.getText() + " from Libri database?");
            if (dr == 0) {
                s.executeUpdate("DELETE FROM members WHERE id = " + id.getText());
                JOptionPane.showMessageDialog(this, "Removed " + name.getText());
                clr();
            } else {
                JOptionPane.showMessageDialog(this, "No changes were made.");
            }
        }
    } catch (Exception exp) {
        System.out.println(exp.getMessage());
    }
}
```



Member's ID: 12400

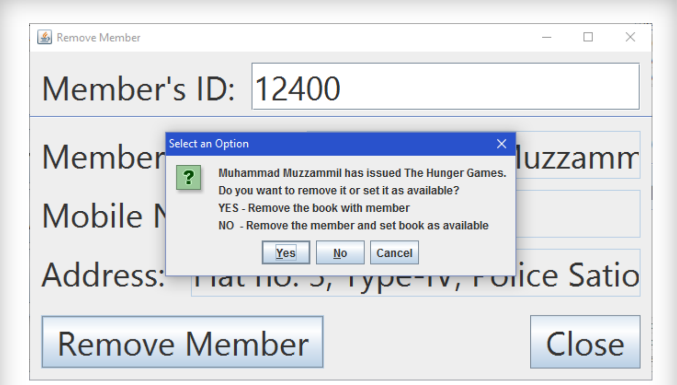
Member Name: ...uzzammil

Mobile N...

Address: Flat no. 3, Type-IV, Police Satio

Remove Muhammad Muzzammil from Libri database?

When member doesn't have a book issued, a simple message will appear.



Member's ID: 12400

Member ...uzzamm

Mobile N...

Address: ...ice Satio

Muhammad Muzzammil has issued The Hunger Games.
Do you want to remove it or set it as available?
YES - Remove the book with member
NO - Remove the member and set book as available

When member does have a book issued, a message showing name of the book will appear, asking to remove book as well or set it as available.

Update Member Frame

Description: This frame is used to **update details of a member** by entering member id.

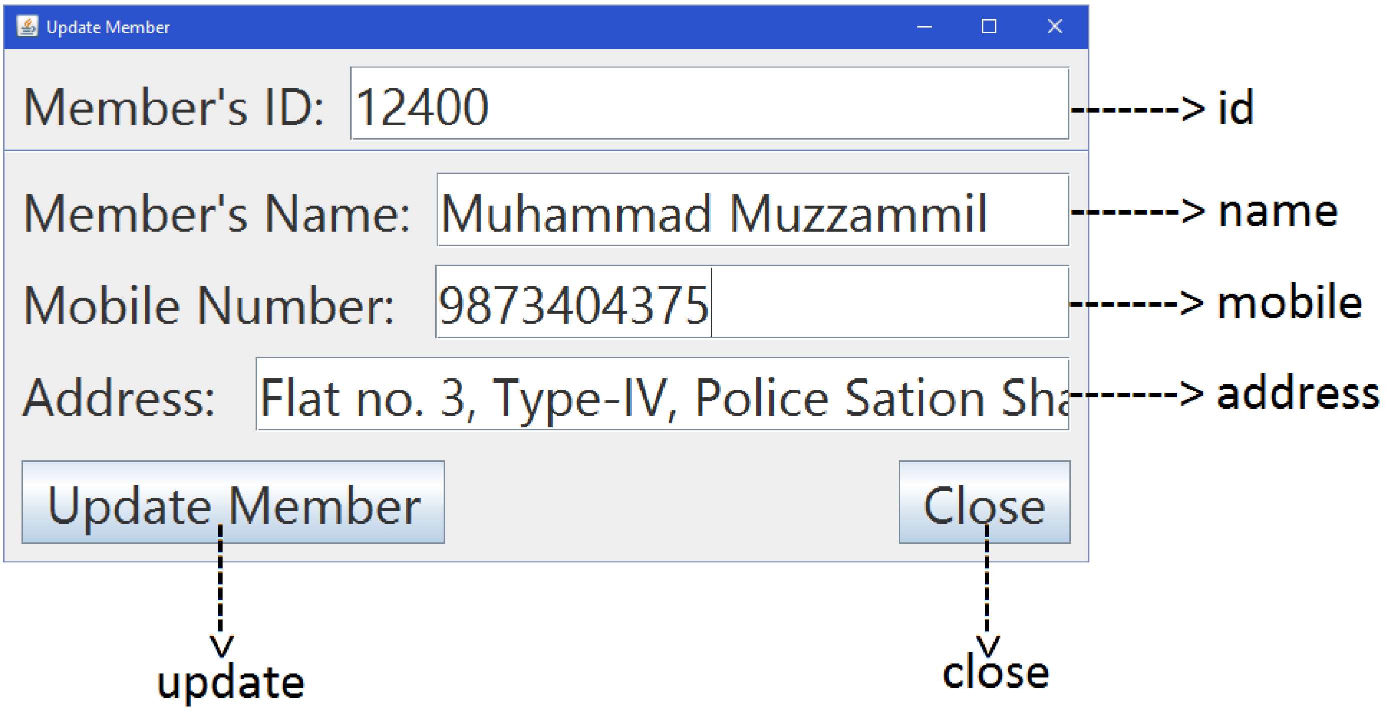
Import Statements:

```
import java.sql.*;
import java.swing.*;
```

Global Variables:

```
String NAME, MOBILE, ADDRESS;
```

updateMember frame design



Update Member

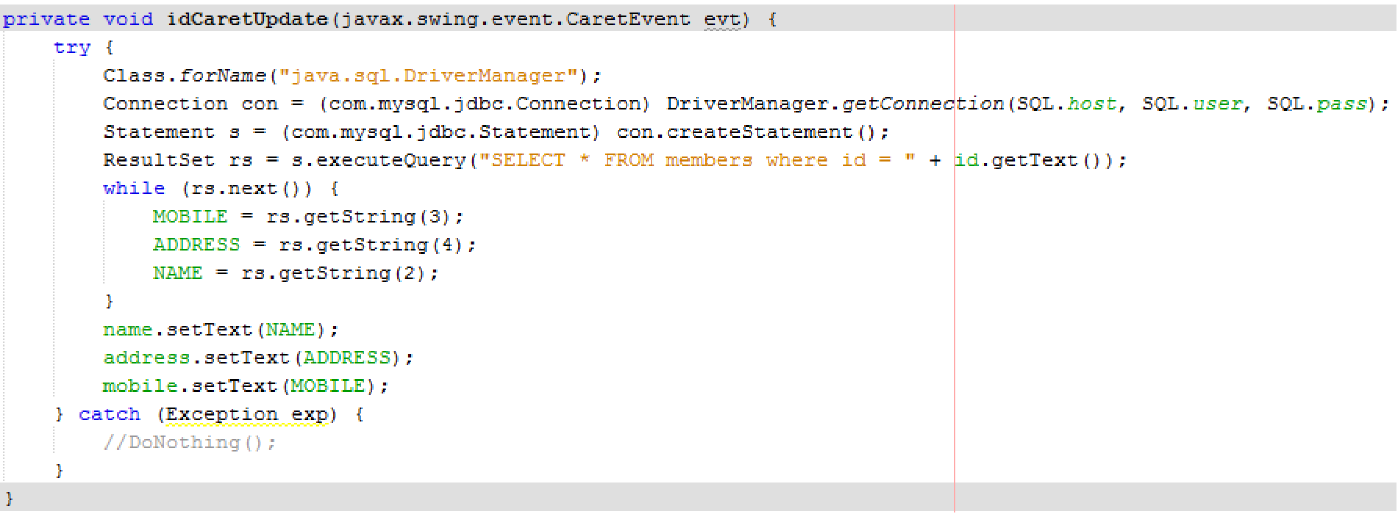
Member's ID: 12400

Member's Name: Muhammad Muzzammil

Mobile Number: 9873404375

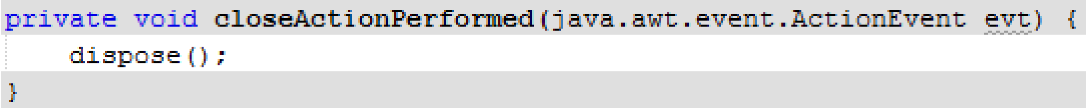
Address: Flat no. 3, Type-IV, Police Sation Sha

On caret update event of **id**



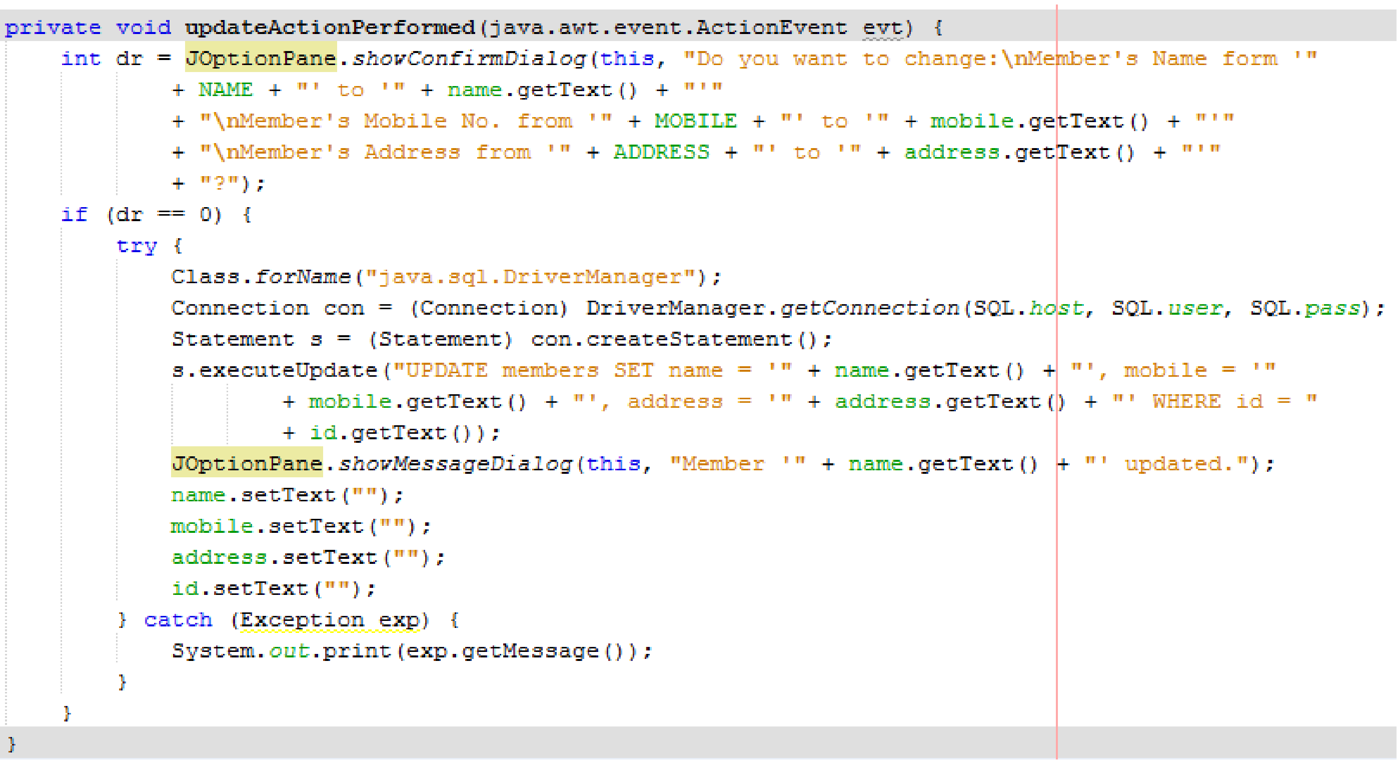
```
private void idCaretUpdate(javax.swing.event.CaretEvent evt) {
    try {
        Class.forName("java.sql.DriverManager");
        Connection con = (com.mysql.jdbc.Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
        Statement s = (com.mysql.jdbc.Statement) con.createStatement();
        ResultSet rs = s.executeQuery("SELECT * FROM members where id = " + id.getText());
        while (rs.next()) {
            MOBILE = rs.getString(3);
            ADDRESS = rs.getString(4);
            NAME = rs.getString(2);
        }
        name.setText(NAME);
        address.setText(ADDRESS);
        mobile.setText(MOBILE);
    } catch (Exception exp) {
        //DoNothing();
    }
}
```

On click event of **close**

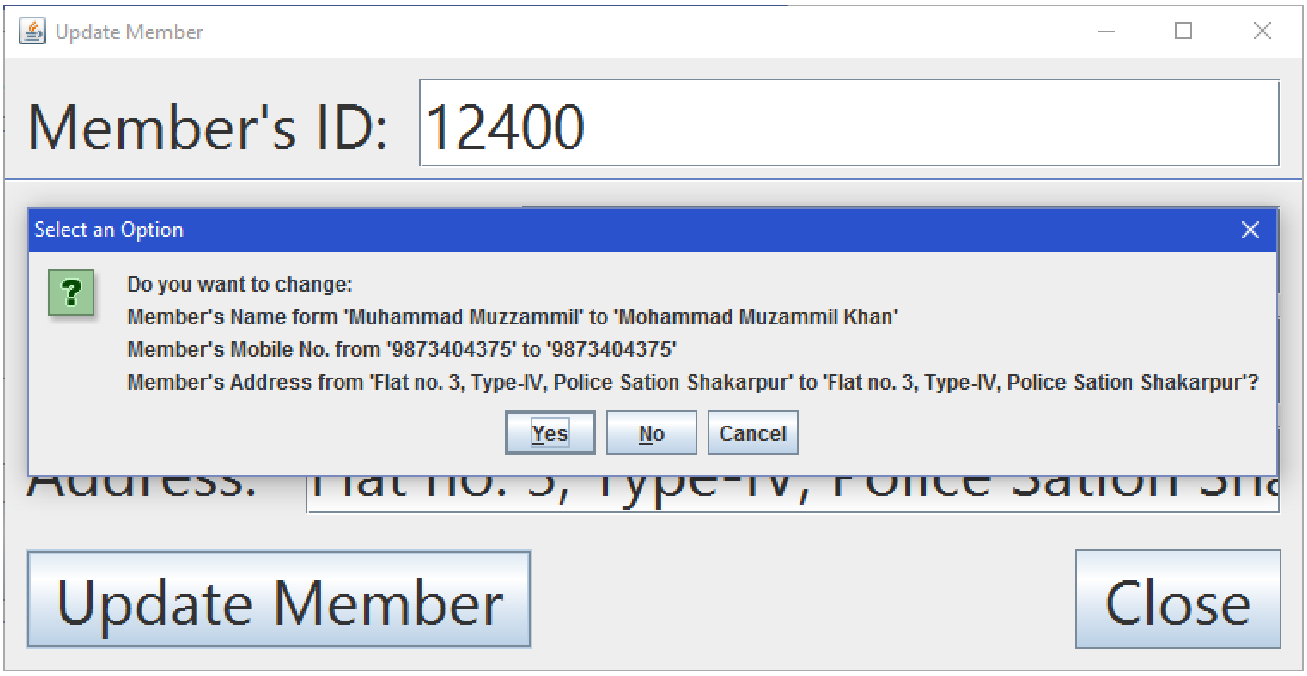


```
private void closeActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```

On click event of **update**



```
private void updateActionPerformed(java.awt.event.ActionEvent evt) {
    int dr = JOptionPane.showConfirmDialog(this, "Do you want to change:\nMember's Name form '"
            + NAME + "' to '" + name.getText() + "'"
            + "\nMember's Mobile No. from '" + MOBILE + "' to '" + mobile.getText() + "'"
            + "\nMember's Address from '" + ADDRESS + "' to '" + address.getText() + "'"
            + "?");
    if (dr == 0) {
        try {
            Class.forName("java.sql.DriverManager");
            Connection con = (Connection) DriverManager.getConnection(SQL.host, SQL.user, SQL.pass);
            Statement s = (Statement) con.createStatement();
            s.executeUpdate("UPDATE members SET name = '" + name.getText() + "', mobile = '"
                    + mobile.getText() + "', address = '" + address.getText() + "' WHERE id = "
                    + id.getText());
            JOptionPane.showMessageDialog(this, "Member '" + name.getText() + "' updated.");
            name.setText("");
            mobile.setText("");
            address.setText("");
            id.setText("");
        } catch (Exception exp) {
            System.out.print(exp.getMessage());
        }
    }
}
```



Member's ID: 12400

Do you want to change:
Member's Name form 'Muhammad Muzzammil' to 'Mohammad Muzammil Khan'
Member's Mobile No. from '9873404375' to '9873404375'
Member's Address from 'Flat no. 3, Type-IV, Police Sation Shakarpur' to 'Flat no. 3, Type-IV, Police Sation Shakarpur'?

Address: Flat no. 3, Type-IV, Police Sation Sha

On click event of update, a confirm dialog is shown with the changes.

THE END