Library Database Testing

SSE 657 Project 3

Dr. Paul MacNeil

Usman Khan

December 8, 2020

Table of Contents

[Table of Figures 3](#_Toc58155680)

[Table of Tables 4](#_Toc58155681)

[Introduction 5](#_Toc58155682)

[Black Box Testing 6](#_Toc58155683)

[Equivalence Class Testing 6](#_Toc58155684)

[Decision Table Testing 7](#_Toc58155685)

[Use Case Testing 7](#_Toc58155686)

[Controller Class 20](#_Toc58155687)

[Conclusion 21](#_Toc58155688)

[References 22](#_Toc58155689)

[Appendix 1. LibraryMain Class 23](#_Toc58155690)

[Appendix 2. Login Class 25](#_Toc58155691)

[Appendix 3. User Class 28](#_Toc58155692)

[Appendix 4. Admin Class 29](#_Toc58155693)

[Appendix 5. Controller Class 32](#_Toc58155694)

[Appendix 6. SqlServerDbAccessor class 44](#_Toc58155695)

# Table of Figures

[Figure 1. Black Box Testing Levels 6](#_Toc57998246)

[Figure 2. List of Valid Users 6](#_Toc57998247)

[Figure 3. Results of Testing Invalid Cases 6](#_Toc57998248)

[Figure 4. Admin Menu for user, ukhan 7](#_Toc57998249)

[Figure 5. User Menu for user, jmercer 7](#_Toc57998250)

[Figure 6. Use Case Diagram from Project 2. 8](#_Toc57998251)

[Figure 7. Adding a book into the database. 9](#_Toc57998252)

[Figure 8. Book added in the database. 10](#_Toc57998253)

[Figure 9. Example of assigning Life of Pi to user, Usman Khan 11](#_Toc57998254)

[Figure 10. Life of Pi has been checked out to Usman Khan 12](#_Toc57998255)

[Figure 11. Assigning a book that's already issued to Jesse Mercer 12](#_Toc57998256)

[Figure 12. The system reassigns the owner of Life of Pi to Jesse Mercer 12](#_Toc57998257)

[Figure 13. Adding user Paul MacNeil to database. 14](#_Toc57998258)

[Figure 14. User Paul MacNeil is added to the database. 14](#_Toc57998259)

[Figure 15. User Usman Khan was added twice with the same credentials. Member ID is different. 14](#_Toc57998260)

[Figure 16. Returning Life of Pi 15](#_Toc57998261)

[Figure 17. Book is no longer assigned an owner. 16](#_Toc57998262)

[Figure 18. Deleting book Life of Pi 17](#_Toc57998263)

[Figure 19. Life of Pi is deleted from the database. 17](#_Toc57998264)

[Figure 20. Deleting user jmercer 18](#_Toc57998265)

[Figure 21. User jmercer is no longer in the database. 19](#_Toc57998266)

# Table of Tables

[Table 1. Login Equivalence Testing 6](#_Toc57998267)

[Table 2. Test Case Table 7](#_Toc57998268)

[Table 3. Use Case Test for Add Book 8](#_Toc57998269)

[Table 4. Use case for Issue Book 10](#_Toc57998270)

[Table 5. Use case test for Add User function. 13](#_Toc57998271)

[Table 6. Use case test for the Return Book function 14](#_Toc57998272)

[Table 7. Use case test for Delete Books function. 16](#_Toc57998273)

[Table 8. Use case test for Delete Users function. 17](#_Toc57998274)

# Introduction

The MUS Development team has been assigned to rebuild the library database system for Bibb County Public Library. In the previous report, the team described both the Inception and Elaboration processes regarding the creation of the library database system. The Inception process details system development regarding the development and design of software and is typically the first step completed in typical system development scenarios. Inception defines the scope and vision of the project, as well as includes the design of use cases that demonstrates the system’s capabilities and functions. The process typically only covers about 10 percent of total use cases and does not entirely define all the system’s capabilities. Inception is meant to establish the project vision and worth.

The purpose of this report is to develop a test plan for the Library Database system, discussed in previous reports. In this report, you will find test cases, and code written to execute test cases. We will be testing at all 4 levels of testing: unit testing, integration testing, system testing, and acceptance testing.

# Black Box Testing

Throughout this project I will be using the black box strategy to perform tests on the library database, discussed in Project 2. The process for black box testing consists of, analyzing the specifications, choosing valid inputs to determine the system under test process, invalid inputs must also be chosen to detect how our system handles it. Black box testing is applicable at all levels of system development, as shown in **Figure 1**.

A picture containing graphical user interface

Description automatically generated

Figure 1. Black Box Testing Levels

## Equivalence Class Testing

I will be testing the login function of this system using the equivalence class testing. For this testing, I will be testing with valid passwords and usernames and invalid passwords and usernames. A list of valid username and passwords is provided below (**Figure 2**).

A picture containing graphical user interface, application

Description automatically generated

Figure 2. List of Valid Users

Table 1. Login Equivalence Testing

|  |  |  |
| --- | --- | --- |
| **Username** | **Password** | **Results** |
| ukhan | bears123 | Valid |
| ukhan | bears234 | Invalid |
| pmacneil | gobears | Invalid |

The results of the testing of the invalid username/password are shown below in **Figure 3**.

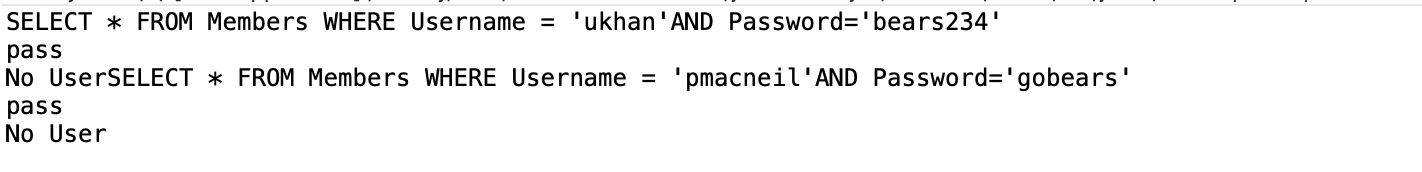


Figure 3. Results of Testing Invalid Cases

This test concludes that the login class is successful.

## Decision Table Testing

The decision table testing method will be used to test whether or not the system identifies a user or an admin. This is critical to test because we do not want random users to have admin privileges. Refer to **Figure 2**, and look at the “Status” column to understand the role of each login user. If “Status” is “member”, then the user class needs to be called, if “Status” is “Admin”, then the admin class needs to be called. Refer to the decision table, below in **Table 2**.

Table 2. Test Case Table

|  |  |  |
| --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** |
| **Inputs** |  |  |
| Username | ukhan | jmercer |
| **Expected Result** |  |  |
| Action | Run admin class. | Run user class. |

**Figure 4** shows the admin class being run after the username, “ukhan”, was inputted.

Graphical user interface, application

Description automatically generated

Figure 4. Admin Menu for user, ukhan

**Figure 5** shows the user class being run after the username, “jmercer”, was inputted.

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 5. User Menu for user, jmercer

## Use Case Testing

Referring back to Project 2, we discussed use cases which are illustrated in the figure below (**Figure 6**). We will be using a few of the use cases discussed and implement use case testing. Use case testing holds a critical foundation at the system and acceptance level.

Diagram

Description automatically generated

Figure 6. Use Case Diagram from Project 2.

The add book function allows the admin to add books into the database. The table below (**Table 3**), is the use case test for the add book function.

Table 3. Use Case Test for Add Book

|  |  |  |
| --- | --- | --- |
| Use Case Component | Description | |
| Use Case Number or Identifier | UC1 | |
| Use Case Name | Add Book | |
| Goal in Context |  | |
| Scope | Library Database System | |
| Level | User Goal | |
| Primary Actor | Admin | |
| Preconditions | Admin enters correct username and password to login in successfully. | |
| Success End Conditions | Book is added to library database. | |
| Failed End Conditions |  | |
| Trigger | Admin selects add book. | |
| Main Success Scenario A:  Actor S: System | Step | Action |
| 1. | Admin navigates the user menu to Add Books. |
| 2. | Admin enters the book ID, title, genre, and owner. |
| 3. | Book is added to the database. |
| 4. | Admin is notified the book is added. |
| 5. | Admin is sent back to the user menu and navigates to View Books. |
| 6. | Admin sees the book has been added with corresponding descriptions. |
| Extensions | 1. | If Book ID is already in use, admin is notified to put in a different book ID. |
| Sub-Variations | Admin may use library database GUI. | |
| Priority | Critical | |
| Response Time | 30 seconds or less | |
| Frequency | Once the book is entered, the process is complete. | |
| Channels to Primary Actor | Login| Add Book| Enter book ID, title, genre, and owner| … | |
| Secondary Actors | None | |
| Channels to Secondary Actors | N/A | |
| Date Due | 8 Dec 2020 | |
| Completeness Level | 1.0 | |
| Open Issues | Same book can be inputted more than once. | |

Graphical user interface, application

Description automatically generated

Figure 7. Adding a book into the database.

Table

Description automatically generated

Figure 8. Book added in the database.

This test concludes that there is an open issue to this use case. The team will need to revisit our software and rewrite some code to fix this issue.

The issue book function allows the admin to check out books to users. The table below (**Table** **4**) is the use case test for the issue book function.

Table 4. Use case for Issue Book

|  |  |  |
| --- | --- | --- |
| Use Case Component | Description | |
| Use Case Number or Identifier | UC3 | |
| Use Case Name | Issue Book | |
| Goal in Context |  | |
| Scope | Library Database System | |
| Level | User Goal | |
| Primary Actor | Admin | |
| Preconditions | Admin enters correct username and password to login in successfully. | |
| Success End Conditions | Book is issued to an owner registered in the database. | |
| Failed End Conditions |  | |
| Trigger | Admin selects issue book. | |
| Main Success Scenario A:  Actor S: System | Step | Action |
| 1. | Admin navigates the user menu to Issue Books. |
| 2. | Admin enters the book title and the new owner. |
| 3. | Book is assigned an owner in the database. |
| 4. | Admin is notified the book is issued. |
| 5. | Admin is sent back to the user menu and navigates to View Issued Books. |
| 6. | Admin sees the book has been added with corresponding descriptions. |
| Extensions | 1. | If the book is already in checked out, admin is notified to inform user that this book is not available. |
| Sub-Variations | Admin may use library database GUI. | |
| Priority | Critical | |
| Response Time | 30 seconds or less | |
| Frequency | Once the book is checked out, the process is complete. | |
| Channels to Primary Actor | Login| Issue Book| Enter book title and new owner| … | |
| Secondary Actors | None | |
| Channels to Secondary Actors | N/A | |
| Date Due | 8 Dec 2020 | |
| Completeness Level | 1.0 | |
| Open Issues | Books can be checked out if the book is already checked out. | |

Graphical user interface, application

Description automatically generated

Figure 9. Example of assigning Life of Pi to user, Usman Khan

Graphical user interface, text, application

Description automatically generated

Figure 10. Life of Pi has been checked out to Usman Khan

Graphical user interface, application

Description automatically generated

Figure 11. Assigning a book that's already issued to Jesse Mercer

Graphical user interface, text, application

Description automatically generated

Figure 12. The system reassigns the owner of Life of Pi to Jesse Mercer

The test concludes that there is an open issue and the team needs to set a revisit and set a condition where if the book is already checked out that the user cannot check this book out. Shown above in **Figure 9,** the admin is *Life of Pi* to user, Usman Khan. This is successful as shown in **Figure 10**; however the same book can be checkout again to another user while Usman Khan already has checked out this book, as shown in **Figure 11** and **Figure 12**.

The add user function allows the admin to add users to the library database system. The table below (**Table 5**) is the use case test for the add user function.

Table 5. Use case test for Add User function.

|  |  |  |
| --- | --- | --- |
| Use Case Component | Description | |
| Use Case Number or Identifier | UC4 | |
| Use Case Name | Add User | |
| Goal in Context |  | |
| Scope | Library Database System | |
| Level | User Goal | |
| Primary Actor | Admin | |
| Preconditions | Admin enters correct username and password to login in successfully. | |
| Success End Conditions | User is added to library database. | |
| Failed End Conditions |  | |
| Trigger | Admin selects add user. | |
| Main Success Scenario A:  Actor S: System | Step | Action |
| 1. | Admin navigates the user menu to Add User. |
| 2. | Admin enters Member name, username, password, and status. |
| 3. | User is added to the database. |
| 4. | Admin is sent back to the user menu and navigates to View Users. |
| 5. | Admin sees the user has been added with corresponding descriptions. |
| Extensions | 1. | If user is already in use, admin is notified that the user is already in there. |
| Sub-Variations | Admin may use library database GUI. | |
| Priority | Critical | |
| Response Time | 30 seconds or less | |
| Frequency | Once the user is added, the process is complete. | |
| Channels to Primary Actor | Login| Add User| Enter Member name, username, password, and status| … | |
| Secondary Actors | None | |
| Channels to Secondary Actors | N/A | |
| Date Due | 8 Dec 2020 | |
| Completeness Level | 1.0 | |
| Open Issues | User can be added twice. | |

Graphical user interface, application

Description automatically generated

Figure 13. Adding user Paul MacNeil to database.

A picture containing table

Description automatically generated

Figure 14. User Paul MacNeil is added to the database.

**Table

Description automatically generated**

Figure 15. User Usman Khan was added twice with the same credentials. Member ID is different.

The test concludes that there is an open issue and the team needs to set a revisit and set a condition where if the user is added to the database then the same user can be added again with another member ID, shown above in **Figure 15**. This would lead to overflow of users in the database.

The return book function allows the admin to return a book that was checked out into the database. The table below (**Table 6**) is the use case for the return book function.

Table 6. Use case test for the Return Book function

|  |  |  |
| --- | --- | --- |
| Use Case Component | Description | |
| Use Case Number or Identifier | UC5 | |
| Use Case Name | Return Book | |
| Goal in Context |  | |
| Scope | Library Database System | |
| Level | User Goal | |
| Primary Actor | Admin | |
| Preconditions | Admin enters correct username and password to login in successfully. | |
| Success End Conditions | The book is returned into the database with no owner. | |
| Failed End Conditions |  | |
| Trigger | Admin selects Return Books. | |
| Main Success Scenario A:  Actor S: System | Step | Action |
| 1. | Admin navigates the user menu to Return Books. |
| 2. | Admin enters title of the book. |
| 3. | Book is free of owner. |
| 4. | Admin is sent back to the user menu and navigates to View Issued Books. |
| 5. | Admin sees that the book is no longer assigned to an owner. |
| Extensions |  |  |
| Sub-Variations | Admin may use library database GUI. | |
| Priority | Critical | |
| Response Time | 30 seconds or less | |
| Frequency | Once the book is returned, the process is complete. | |
| Channels to Primary Actor | Login| Return Books| Enter book title|… | |
| Secondary Actors | None | |
| Channels to Secondary Actors | N/A | |
| Date Due | 8 Dec 2020 | |
| Completeness Level | 1.0 | |
| Open Issues | Book should not be listed in Issued Books. | |

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 16. Returning Life of Pi

Graphical user interface, text, application

Description automatically generated

Figure 17. Book is no longer assigned an owner.

The test concludes that there is an open issue and the team needs to set a revisit and have our system remove books from the issued books menu instead of just removing the owner’s name, shown above in **Figure 17**. This would be confusing over the time after many books have been checked in and out of the system.

The delete books function allows the admin to delete a book that from the database. The table below (**Table 7**) is the use case for the delete books function.

Table 7. Use case test for Delete Books function.

|  |  |  |
| --- | --- | --- |
| Use Case Component | Description | |
| Use Case Number or Identifier | UC6 | |
| Use Case Name | Delete Books | |
| Goal in Context |  | |
| Scope | Library Database System | |
| Level | User Goal | |
| Primary Actor | Admin | |
| Preconditions | Admin enters correct username and password to login in successfully. | |
| Success End Conditions | The book is deleted from the database. | |
| Failed End Conditions |  | |
| Trigger | Admin selects Delete Books. | |
| Main Success Scenario A:  Actor S: System | Step | Action |
| 1. | Admin navigates the user menu to Delete Books. |
| 2. | Admin enters title of the book. |
| 3. | Book is deleted from database. |
| 4. | Admin is sent back to the user menu and navigates to View Books. |
| 5. | Admin sees that the book is no longer in the database. |
| Extensions |  |  |
| Sub-Variations | Admin may use library database GUI. | |
| Priority | Critical | |
| Response Time | 30 seconds or less | |
| Frequency | Once the book is deleted, the process is complete. | |
| Channels to Primary Actor | Login| Delete Books| Enter book title|… | |
| Secondary Actors | None | |
| Channels to Secondary Actors | N/A | |
| Date Due | 8 Dec 2020 | |
| Completeness Level | 1.0 | |
| Open Issues | N/A | |

Graphical user interface, text, application

Description automatically generated

Figure 18. Deleting book Life of Pi

Graphical user interface

Description automatically generated

Figure 19. Life of Pi is deleted from the database.

The test concludes that there is no open issue with the . This has been proven in the figure above, **Figure 19**.

The final function, Delete Users function, allows the admin to delete the users from the database that no longer want to be in the library system. The table below () is the use case test for the Delete Users function.

Table 8. Use case test for Delete Users function.

|  |  |  |
| --- | --- | --- |
| Use Case Component | Description | |
| Use Case Number or Identifier | UC7 | |
| Use Case Name | Delete Users | |
| Goal in Context |  | |
| Scope | Library Database System | |
| Level | User Goal | |
| Primary Actor | Admin | |
| Preconditions | Admin enters correct username and password to login in successfully. | |
| Success End Conditions | The user is deleted from the database. | |
| Failed End Conditions |  | |
| Trigger | Admin selects Delete Users. | |
| Main Success Scenario A:  Actor S: System | Step | Action |
| 1. | Admin navigates the user menu to Delete Users. |
| 2. | Admin enters username that is to be deleted. |
| 3. | User is deleted from database. |
| 4. | Admin is sent back to the user menu and navigates to View Users. |
| 5. | Admin sees that the user is no longer in the database. |
| Extensions |  |  |
| Sub-Variations | Admin may use library database GUI. | |
| Priority | Critical | |
| Response Time | 30 seconds or less | |
| Frequency | Once the user is deleted, the process is complete. | |
| Channels to Primary Actor | Login| Delete User| Enter username|… | |
| Secondary Actors | None | |
| Channels to Secondary Actors | N/A | |
| Date Due | 8 Dec 2020 | |
| Completeness Level | 1.0 | |
| Open Issues |  | |

Graphical user interface, text, application

Description automatically generated

Figure 20. Deleting user jmercer

Table

Description automatically generated

Figure 21. User jmercer is no longer in the database.

The test concludes that there is no open issue with the Delete Users function. This has been proven in the figure above, **Figure 21**.

# Controller Class

The Controller class is where the menus’ functions are stored. Each method in this class corresponds with a button in both the Admin and User classes. This class also handles the connectivity to the library database using Java’s SQL library. The code for each of the described classes can be found in the Appendix below. The Controller Class is testing at a unit level.

When a button is pressed, the controller class is called and uses a corresponding method. For example, when the “Add User” button is pressed, the addUser() method is called from the controller class.

# Conclusion

In this report, we saw the testing of mainly Black Box testing techniques including equivalence testing, decision table testing, and unit case testing. In the Copeland book, many individual tests are discussed but I used the testing techniques I found suitable for our library system. We also briefly discussed the Controller Class testing at a unit level.

All levels of testing were covered in this report. Of course, multiple tests can be conducted at all 4 levels; however, that would be excessive. Considering all the design flaws found from the testing, the software team will need to get back together and consider different solutions to these problems.

# References

Copeland, L. (2004). *A practitioner's guide to software test design*. Boston, MA: Artech House.

Larman, C. (2004). *Applying UML and patterns: An introduction to object-oriented analysis and design and the unified process*. Prentice-Hall.

Library Icons – Free Download, PNG and SVG. (n.d). Retrieved September 23, 2020, from https://8.com/incons/set/library

# Appendix 1. LibraryMain Class

|  |
| --- |
| **package** library;  **import** java.awt.Dimension;  **import** java.awt.Image;    **import** javax.swing.\*;  **public** **class** LibraryMain **extends** JPanel {    **private** Image iconOnTitleBar;  //private Image backgroundPhoto;    **public** **void** libraryMain() {  Dimension windowSize = **new** Dimension(900, 600);  **this**.setPreferredSize(windowSize);  **this**.setMaximumSize(windowSize);  **this**.setMinimumSize(windowSize);  }  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  Login login = **new** Login();  login.gui();  //myBooks();  //addBook();  //addUser();  //viewIssued();  //deleteBook();  }    **public** **void** setIconOnTitleBar(Image image) {  // **TODO** Auto-generated method stub  iconOnTitleBar = image;  }    **public** Image getIconOnTitleBar() {  // **TODO** Auto-generated method stub  **return** iconOnTitleBar;  }    **public** **static** **void** myBooks() {  Controller c = **new** Controller();  c.myBooks("mikae");  }    **public** **static** **void** addBook() {  Controller c = **new** Controller();  c.addBook();  }    **public** **static** **void** addUser() {  Controller c = **new** Controller();  c.addUser();  }    **public** **static** **void** viewIssued() {  Controller c = **new** Controller();  c.viewIssued();  }    **public** **static** **void** deleteBook() {  Controller c = **new** Controller();  c.deleteBook();  }    **public** **static** **void** viewBooks() {  Controller c = **new** Controller();  c.viewBooks();  }    **public** **static** **void** returnBook() {  Controller c = **new** Controller();  c.returnBook();  }    **public** **static** **void** issueBooks() {  Controller c = **new** Controller();  c.issueBook();  }    **public** **static** **void** deleteUser() {  Controller c = **new** Controller();  c.deleteUser();  }    **public** **static** **void** viewUsers() {  Controller c = **new** Controller();  c.viewUser();  }  } |

# Appendix 2. Login Class

|  |
| --- |
| **package** library;  **import** javax.swing.\*;  **import** csdbdao.SqlServerDbAccessor;  **import** java.awt.event.ActionEvent;  **import** java.awt.event.ActionListener;  **import** java.awt.event.WindowEvent;  **import** java.sql.\*;  **public** **class** Login {  **public** **void** gui() {  // **TODO** Auto-generated method stub  JFrame window=**new** JFrame("Login");  JLabel l1,l2;  l1=**new** JLabel("Username");  l1.setBounds(30,15, 100,30); //x axis, y axis, width, height    l2=**new** JLabel("Password");  l2.setBounds(30,50, 100,30);    JTextField F\_user = **new** JTextField();  F\_user.setBounds(110, 15, 200, 30);    JPasswordField F\_pass=**new** JPasswordField();  F\_pass.setBounds(110, 50, 200, 30);    JButton login\_but=**new** JButton("Login");  login\_but.setBounds(130,90,80,25);  login\_but.addActionListener(**new** ActionListener() {  @Override  **public** **void** actionPerformed(ActionEvent e) {  // **TODO** Auto-generated method stub  String username = F\_user.getText();  @SuppressWarnings("deprecation")  String password = F\_pass.~~getText~~();    **if**(username.equals(""))  {  JOptionPane.*showMessageDialog*(**null**, "Please enter username");  }  **else** **if** (password.equals(""))  {  JOptionPane.*showMessageDialog*(**null**, "Please enter password");  }  **else** {  SqlServerDbAccessor sqda = **new** SqlServerDbAccessor();    sqda.setDbName("SSE657-Library");  sqda.connectToDb();  String sql = ("SELECT \* FROM Members WHERE Username = '"+username+"'AND Password='"+password+"'");  System.***out***.println(sql);    **try** {  Statement stmt = sqda.getConnection().createStatement();  ResultSet rs = stmt.executeQuery(sql);  System.***out***.println("pass");  //stmt.executeUpdate(sql);  **if**(rs.next()==**false**) {  System.***out***.print("No User");  JOptionPane.*showMessageDialog*(**null**, "Wrong Username/Password");  }  **else** {    System.***out***.println("pass2");  System.***out***.println("pass3");  String member = rs.getString("Status");  System.***out***.println(member);  //String MID = rs.getString("MemberId");  //System.out.println(MID);    **if**(member.equals("Admin"))  {    Admin.*menu*(username);  window.dispatchEvent(**new** WindowEvent(window, WindowEvent.***WINDOW\_CLOSING***));  }  **else** {  //User user = new User();  User.*menu*(username);  window.dispatchEvent(**new** WindowEvent(window, WindowEvent.***WINDOW\_CLOSING***));  }  }  }  **catch**(Exception ex){  ex.printStackTrace();  }  }  }    });    LibraryMain lm = **new** LibraryMain();    lm.setIconOnTitleBar(**new** ImageIcon("images/libraryIcon.png").getImage());  //Library Icons - Free Download, PNG and SVG. (n.d.). Retrieved September 23, 2020, from https://icons8.com/icons/set/library  window.setIconImage(lm.getIconOnTitleBar());    window.add(F\_pass); //add password  window.add(login\_but);//adding button in JFrame  window.add(F\_user); //add user  window.add(l1); // add label1 i.e. for username  window.add(l2); // add label2 i.e. for password    window.setSize(400,180);//400 width and 500 height  window.setLayout(**null**);  window.setVisible(**true**);//making the frame visible  window.setLocationRelativeTo(**null**);  }  }    /\*  login\_but.addActionListener(new ActionListener() { //Perform action    public void actionPerformed(ActionEvent e){    String username = F\_user.getText();  String password = F\_pass.getText();    if(username.equals("")) //If username is null  {  JOptionPane.showMessageDialog(null,"Please enter username");  }  else if(password.equals("")) //If password is null  {  JOptionPane.showMessageDialog(null,"Please enter password");  }  else  {  if(password.equals("password") && username.equals("admin"))//hardcoded for now  {  Admin admin = new Admin();  admin.menu(username);  window.dispatchEvent(new WindowEvent(window, WindowEvent.WINDOW\_CLOSING));  }    else if(username.equals("user") && password.equals("password")) //hardcoded for now  {  User user = new User();  user.menu(username);  window.dispatchEvent(new WindowEvent(window, WindowEvent.WINDOW\_CLOSING));  }    else  {  JOptionPane.showMessageDialog(null,"Username or password is not correct");  }  }  }  });\*/ |

# Appendix 3. User Class

|  |
| --- |
| **package** library;  **import** java.awt.event.ActionEvent;  **import** java.awt.event.ActionListener;  **import** javax.swing.\*;  **public** **class** User **extends** JFrame {    **public** **static** **void** menu(String username) {  JFrame userMenu = **new** JFrame("User Menu");    JButton my\_books = **new** JButton("My Books");  my\_books.setBounds(50,20,120,25);  my\_books.addActionListener(**new** ActionListener() {  **public** **void** actionPerformed(ActionEvent e){  Controller library = **new** Controller();  library.myBooks(username);  }  });    JButton view\_books = **new** JButton("View Books");  view\_books.setBounds(180,20,120,25);  view\_books.addActionListener(**new** ActionListener() {  **public** **void** actionPerformed(ActionEvent e){  Controller library = **new** Controller();  library.viewBooks();  }  });    LibraryMain lm = **new** LibraryMain();    lm.setIconOnTitleBar(**new** ImageIcon("images/libraryIcon.png").getImage());  userMenu.setIconImage(lm.getIconOnTitleBar());    userMenu.add(my\_books);  userMenu.add(view\_books);    userMenu.setSize(360,100);//360 width and 100 height  userMenu.setLayout(**null**);  userMenu.setVisible(**true**);//making the frame visible  userMenu.setLocationRelativeTo(**null**);  }  } |

# Appendix 4. Admin Class

|  |
| --- |
| **package** library;  **import** java.awt.event.ActionEvent;  **import** java.awt.event.ActionListener;  **import** javax.swing.\*;  **public** **class** Admin {  **public** **static** **void** menu(String username) {  JFrame adminMenu = **new** JFrame("Admin Menu");    JButton view\_books = **new** JButton("View Books");  view\_books.setBounds(50,20,150,25);  view\_books.addActionListener(**new** ActionListener() {  **public** **void** actionPerformed(ActionEvent e){  Controller library = **new** Controller();  library.viewBooks();  }  });    JButton add\_book = **new** JButton("Add Book");  add\_book.setBounds(250,20,150,25);  add\_book.addActionListener(**new** ActionListener() {  **public** **void** actionPerformed(ActionEvent e){  Controller library = **new** Controller();  library.addBook();  }  });    JButton view\_issue = **new** JButton("View Issued Books");  view\_issue.setBounds(450,20,150,25);  view\_issue.addActionListener(**new** ActionListener() {  **public** **void** actionPerformed(ActionEvent e){  Controller library = **new** Controller();  library.viewIssued();  }  });    JButton issue\_book = **new** JButton("Issue Book");  issue\_book.setBounds(650,20,150,25);  issue\_book.addActionListener(**new** ActionListener() {  **public** **void** actionPerformed(ActionEvent e){  Controller library = **new** Controller();  library.issueBook();  }  });    JButton add\_user=**new** JButton("Add User");  add\_user.setBounds(50,60,150,25);  add\_user.addActionListener(**new** ActionListener() {  **public** **void** actionPerformed(ActionEvent e){  Controller library = **new** Controller();  library.addUser();  }  });    JButton view\_user =**new** JButton("View Users");  view\_user.setBounds(250,60,150,25);  view\_user.addActionListener(**new** ActionListener() {  **public** **void** actionPerformed(ActionEvent e){  Controller library = **new** Controller();  library.viewUser();  }  });    JButton return\_book=**new** JButton("Return Books");  return\_book.setBounds(450,60,150,25);  return\_book.addActionListener(**new** ActionListener() {  **public** **void** actionPerformed(ActionEvent e){  Controller library = **new** Controller();  library.returnBook();  }  });    JButton create\_data=**new** JButton("My Books");  create\_data.setBounds(650,60,150,25);  create\_data.addActionListener(**new** ActionListener() {  **public** **void** actionPerformed(ActionEvent e){  Controller library = **new** Controller();  library.myBooks(username);  }  });    JButton delete\_book = **new** JButton("Delete Books");  delete\_book.setBounds(50,100,150,25);  delete\_book.addActionListener(**new** ActionListener() {  **public** **void** actionPerformed(ActionEvent e){  Controller library = **new** Controller();  library.deleteBook();  }  });    JButton delete\_users = **new** JButton("Delete Users");  delete\_users.setBounds(250,100,150,25);  delete\_users.addActionListener(**new** ActionListener() {  **public** **void** actionPerformed(ActionEvent e){  Controller library = **new** Controller();  library.deleteUser();  }  });  LibraryMain lm = **new** LibraryMain();    lm.setIconOnTitleBar(**new** ImageIcon("images/libraryIcon.png").getImage());  adminMenu.setIconImage(lm.getIconOnTitleBar());    adminMenu.add(view\_books);  adminMenu.add(create\_data);  adminMenu.add(add\_book);  adminMenu.add(view\_issue);  adminMenu.add(issue\_book);  adminMenu.add(add\_user);  adminMenu.add(view\_user);  adminMenu.add(return\_book);  adminMenu.add(delete\_book);  adminMenu.add(delete\_users);    adminMenu.setSize(850,180);//850 width and 180 height  adminMenu.setLayout(**null**);  adminMenu.setVisible(**true**);//making the frame visible  adminMenu.setLocationRelativeTo(**null**);  }  } |

# Appendix 5. Controller Class

|  |
| --- |
| **package** library;  **import** java.awt.event.ActionEvent;  **import** java.awt.event.ActionListener;  **import** java.awt.event.WindowEvent;  **import** java.sql.\*;  **import** java.time.LocalDate;  **import** javax.swing.ImageIcon;  **import** javax.swing.JButton;  **import** javax.swing.JFrame;  **import** javax.swing.JLabel;  **import** javax.swing.JOptionPane;  **import** javax.swing.JScrollPane;  **import** javax.swing.JTable;  **import** javax.swing.JTextField;  **import** net.proteanit.sql.DbUtils;  **import** csdbdao.SqlServerDbAccessor;  **public** **class** Controller {    **public** **void** viewBooks() {  //both user and admin  JFrame f = **new** JFrame("Books Available");  SqlServerDbAccessor sqda = **new** SqlServerDbAccessor();  sqda.setDbName("SSE657-Library");  sqda.connectToDb();  String sql = "SELECT \* FROM Book\_Lookup";  **try** {    Statement stmt = sqda.getConnection().createStatement();  ResultSet rs = stmt.executeQuery(sql);    JTable book\_list= **new** JTable();  book\_list.setModel(DbUtils.*resultSetToTableModel*(rs));    JScrollPane scrollPane = **new** JScrollPane(book\_list);    LibraryMain lm = **new** LibraryMain();    lm.setIconOnTitleBar(**new** ImageIcon("images/libraryIcon.png").getImage());  //Library Icons - Free Download, PNG and SVG. (n.d.). Retrieved September 23, 2020, from https://icons8.com/icons/set/library  f.setIconImage(lm.getIconOnTitleBar());  f.add(scrollPane);  f.setSize(800, 400);  f.setVisible(**true**);  f.setLocationRelativeTo(**null**);  }  **catch** (SQLException e1) {  // **TODO** Auto-generated catch block  JOptionPane.*showMessageDialog*(**null**, e1);  }  }    **public** **void** myBooks(String username) {  JFrame f = **new** JFrame("My Books");  SqlServerDbAccessor sqda = **new** SqlServerDbAccessor();  sqda.setDbName("SSE657-Library");  sqda.connectToDb();  String sql = "SELECT \* FROM Book\_Lookup "  + "WHERE owner IN ( '"+username+"')";  **try** {    Statement stmt = sqda.getConnection().createStatement();  ResultSet rs = stmt.executeQuery(sql);    JTable book\_list= **new** JTable();  book\_list.setModel(DbUtils.*resultSetToTableModel*(rs));    JScrollPane scrollPane = **new** JScrollPane(book\_list);    LibraryMain lm = **new** LibraryMain();    lm.setIconOnTitleBar(**new** ImageIcon("images/libraryIcon.png").getImage());  //Library Icons - Free Download, PNG and SVG. (n.d.). Retrieved September 23, 2020, from https://icons8.com/icons/set/library  f.setIconImage(lm.getIconOnTitleBar());  f.add(scrollPane);  f.setSize(800, 400);  f.setVisible(**true**);  f.setLocationRelativeTo(**null**);  }  **catch** (SQLException e1) {  // **TODO** Auto-generated catch block  JOptionPane.*showMessageDialog*(**null**, e1);  }  }    **public** **void** viewIssued() {  //admin only  JFrame f = **new** JFrame("Books Available");  SqlServerDbAccessor sqda = **new** SqlServerDbAccessor();  sqda.setDbName("SSE657-Library");  sqda.connectToDb();  String sql = "SELECT \* FROM Book\_Lookup WHERE NOT owner = 'NULL'";  **try** {    Statement stmt = sqda.getConnection().createStatement();  ResultSet rs = stmt.executeQuery(sql);    JTable book\_list= **new** JTable();  book\_list.setModel(DbUtils.*resultSetToTableModel*(rs));    JScrollPane scrollPane = **new** JScrollPane(book\_list);    LibraryMain lm = **new** LibraryMain();    lm.setIconOnTitleBar(**new** ImageIcon("images/libraryIcon.png").getImage());  //Library Icons - Free Download, PNG and SVG. (n.d.). Retrieved September 23, 2020, from https://icons8.com/icons/set/library  f.setIconImage(lm.getIconOnTitleBar());  f.add(scrollPane);  f.setSize(800, 400);  f.setVisible(**true**);  f.setLocationRelativeTo(**null**);  }  **catch** (SQLException e1) {  // **TODO** Auto-generated catch block  JOptionPane.*showMessageDialog*(**null**, e1);  }  }    **public** **void** addUser() {  //only admin  JFrame window=**new** JFrame("Add User");  JLabel l1,l2, l3, l4, l5;    l2=**new** JLabel("Member Name");  l2.setBounds(30,15, 100,30);    JTextField F\_membername=**new** JTextField();  F\_membername.setBounds(120, 15, 200, 30);    l3=**new** JLabel("Username");  l3.setBounds(30,50, 100,30);    JTextField F\_username=**new** JTextField();  F\_username.setBounds(120, 50, 200, 30);    l4=**new** JLabel("Password");  l4.setBounds(30,85, 100,30);    JTextField F\_password=**new** JTextField();  F\_password.setBounds(120, 85, 200, 30);    l5=**new** JLabel("Status");  l5.setBounds(30,120, 100,30);    JTextField F\_status=**new** JTextField();  F\_status.setBounds(120, 120, 200, 30);    JButton add\_but=**new** JButton("Add User");  add\_but.setBounds(120,155,200,25);  add\_but.addActionListener(**new** ActionListener() { //Perform action    **public** **void** actionPerformed(ActionEvent e){  //String memberid = F\_memberID.getText();  String membername = F\_membername.getText();  String username = F\_username.getText();  String password = F\_password.getText();  String status = F\_status.getText();    SqlServerDbAccessor sqda = **new** SqlServerDbAccessor();  sqda.setDbName("SSE657-Library");  sqda.connectToDb();  String sql = "INSERT INTO Members (MemberName, Username, Password, Status)"  + "VALUES ('" +membername+ "' , '" + username + "' , '" + password + "' , '" + status + "')" ;  **try** {    Statement stmt = sqda.getConnection().createStatement();  //ResultSet rs = stmt.executeQuery(sql);  stmt.executeUpdate(sql);  window.dispatchEvent(**new** WindowEvent(window, WindowEvent.***WINDOW\_CLOSING***));    }  **catch** (SQLException e1) {  // **TODO** Auto-generated catch block  JOptionPane.*showMessageDialog*(**null**, e1);  }    }  });    LibraryMain lm = **new** LibraryMain();    lm.setIconOnTitleBar(**new** ImageIcon("images/libraryIcon.png").getImage());  //Library Icons - Free Download, PNG and SVG. (n.d.). Retrieved September 23, 2020, from https://icons8.com/icons/set/library  window.setIconImage(lm.getIconOnTitleBar());    window.add(F\_password); //add password  //window.add(F\_memberID);//adding button in JFrame  window.add(F\_username); //add user  window.add(F\_membername);  window.add(F\_status);  window.add(add\_but);  //window.add(l1); // add label1 i.e. for username  window.add(l2); // add label2 i.e. for password  window.add(l3);  window.add(l4);  window.add(l5);    window.setSize(400,300);//400 width and 500 height  window.setLayout(**null**);  window.setVisible(**true**);//making the frame visible  window.setLocationRelativeTo(**null**);  }    **public** **void** viewUser() {  //only admin  JFrame f = **new** JFrame("Current Members");  SqlServerDbAccessor sqda = **new** SqlServerDbAccessor();  sqda.setDbName("SSE657-Library");  sqda.connectToDb();  String sql = "SELECT \* FROM Members";  **try** {    Statement stmt = sqda.getConnection().createStatement();  ResultSet rs = stmt.executeQuery(sql);    JTable book\_list= **new** JTable();  book\_list.setModel(DbUtils.*resultSetToTableModel*(rs));    JScrollPane scrollPane = **new** JScrollPane(book\_list);    LibraryMain lm = **new** LibraryMain();    lm.setIconOnTitleBar(**new** ImageIcon("images/libraryIcon.png").getImage());  //Library Icons - Free Download, PNG and SVG. (n.d.). Retrieved September 23, 2020, from https://icons8.com/icons/set/library  f.setIconImage(lm.getIconOnTitleBar());  f.add(scrollPane);  f.setSize(800, 400);  f.setVisible(**true**);  f.setLocationRelativeTo(**null**);  }  **catch** (SQLException e1) {  // **TODO** Auto-generated catch block  JOptionPane.*showMessageDialog*(**null**, e1);  }  }  **public** **void** issueBook() {  //only admin  //only admin  //change the owner the book in book\_lookup and book\_instance  //take note of issue date and return date  JFrame window=**new** JFrame("Issue Book");  JLabel l1,l2;  l1=**new** JLabel("Title");  l1.setBounds(30,15, 100,30); //x axis, y axis, width, height    JTextField F\_title = **new** JTextField();  F\_title.setBounds(110, 15, 200, 30);    l2=**new** JLabel("Owner");  l2.setBounds(30,50, 100,30);    JTextField F\_owner=**new** JTextField();  F\_owner.setBounds(110, 50, 200, 30);    JButton add\_but=**new** JButton("Issue Book");  add\_but.setBounds(110,155,200,25);  add\_but.addActionListener(**new** ActionListener() { //Perform action    **public** **void** actionPerformed(ActionEvent e){  String title = F\_title.getText();  String owner = F\_owner.getText();    System.***out***.print(title);  System.***out***.print(" ");  System.***out***.print(owner);    LocalDate date = LocalDate.*now*();  //System.out.print(date);    LocalDate newDate = date.plusDays(7);  //System.out.println(newDate);      //JFrame f = new JFrame("Current Members");  SqlServerDbAccessor sqda = **new** SqlServerDbAccessor();  sqda.setDbName("SSE657-Library");  sqda.connectToDb();  String sql = "UPDATE Books\_Instance " +"SET owner ='"+owner+"'"  + " WHERE Title='"+title+"'";  String sql2 = "UPDATE Book\_Lookup " +"SET owner ='"+owner+"'"  + " WHERE Title='"+title+"'";  String sql3 = "UPDATE Books\_Instance " +"SET CheckoutDate ='"+date+"'"  + " WHERE Title='"+title+"'";  String sql4 = "UPDATE Books\_Instance " +"SET ReturnDate ='"+newDate+"'"  + " WHERE Title='"+title+"'";    **try** {    Statement stmt = sqda.getConnection().createStatement();  stmt.executeUpdate(sql);  stmt.executeUpdate(sql2);  stmt.executeUpdate(sql3);  stmt.executeUpdate(sql4);  window.dispatchEvent(**new** WindowEvent(window, WindowEvent.***WINDOW\_CLOSING***));  }  **catch** (SQLException e1) {  // **TODO** Auto-generated catch block  JOptionPane.*showMessageDialog*(**null**, e1);  }    //System.out.println("title "+title+" genre "+genre);  }  });    LibraryMain lm = **new** LibraryMain();    lm.setIconOnTitleBar(**new** ImageIcon("images/libraryIcon.png").getImage());  //Library Icons - Free Download, PNG and SVG. (n.d.). Retrieved September 23, 2020, from https://icons8.com/icons/set/library  window.setIconImage(lm.getIconOnTitleBar());    //window.add(F\_author); //add password  window.add(add\_but);//adding button in JFrame  window.add(F\_title); //add user  // window.add(F\_genre);  //window.add(F\_bookID);  window.add(F\_owner);  window.add(l1);  window.add(l2);  //window.add(l3);  //window.add(l4);  //window.add(l5);    window.setSize(400,300);  window.setLayout(**null**);  window.setVisible(**true**);//making the frame visible  window.setLocationRelativeTo(**null**);    }    **public** **void** addBook() {  //only admin  JFrame window=**new** JFrame("Add Book");  JLabel l1,l2, l3, l4, l5;  l1=**new** JLabel("Title");  l1.setBounds(30,15, 100,30); //x axis, y axis, width, height    JTextField F\_title = **new** JTextField();  F\_title.setBounds(110, 15, 200, 30);    l2=**new** JLabel("Author");  l2.setBounds(30,50, 100,30);    JTextField F\_author=**new** JTextField();  F\_author.setBounds(110, 50, 200, 30);    l4=**new** JLabel("Genre");  l4.setBounds(30,85, 100,30);    JTextField F\_genre=**new** JTextField();  F\_genre.setBounds(110, 85, 200, 30);    l5=**new** JLabel("Owner");  l5.setBounds(30,120, 100,30);    JTextField F\_owner=**new** JTextField();  F\_owner.setBounds(110, 120, 200, 30);    JButton add\_but=**new** JButton("Add Book");  add\_but.setBounds(110,155,200,25);  add\_but.addActionListener(**new** ActionListener() { //Perform action    **public** **void** actionPerformed(ActionEvent e){  String author = F\_author.getText();  String title = F\_title.getText();  String genre = F\_genre.getText();  //String bookID = F\_bookID.getText();  String owner = F\_owner.getText();  String status = "In-Stock";    //JFrame f = new JFrame("Current Members");  SqlServerDbAccessor sqda = **new** SqlServerDbAccessor();  sqda.setDbName("SSE657-Library");  sqda.connectToDb();  String sql = "INSERT INTO Book\_Lookup (Title, Name, Genre, owner)"  + "VALUES ('" + title+ "', '" +author+ "' , '" + genre + "' , '" + owner + "')" ;  String sql2 = "INSERT INTO Books\_Instance (Title, Status, owner)"  + "VALUES ('" + title+ "', '" +status+ "' , '" + owner + "')" ;  **try** {    Statement stmt = sqda.getConnection().createStatement();  stmt.executeUpdate(sql);  stmt.executeUpdate(sql2);    window.dispatchEvent(**new** WindowEvent(window, WindowEvent.***WINDOW\_CLOSING***));  }  **catch** (SQLException e1) {  // **TODO** Auto-generated catch block  JOptionPane.*showMessageDialog*(**null**, e1);  }    //System.out.println("title "+title+" genre "+genre);  }  });    LibraryMain lm = **new** LibraryMain();    lm.setIconOnTitleBar(**new** ImageIcon("images/libraryIcon.png").getImage());  //Library Icons - Free Download, PNG and SVG. (n.d.). Retrieved September 23, 2020, from https://icons8.com/icons/set/library  window.setIconImage(lm.getIconOnTitleBar());    window.add(F\_author); //add password  window.add(add\_but);//adding button in JFrame  window.add(F\_title); //add user  window.add(F\_genre);  //window.add(F\_bookID);  window.add(F\_owner);  window.add(l1);  window.add(l2);  //window.add(l3);  window.add(l4);  window.add(l5);    window.setSize(400,300);  window.setLayout(**null**);  window.setVisible(**true**);//making the frame visible  window.setLocationRelativeTo(**null**);  }    **public** **void** returnBook() {  //only admin  //only admin  //restore owner back to null and get rid of dates  //undo what is done in issue book  //only admin  //change the owner the book in book\_lookup and book\_instance  //take note of issue date and return date  JFrame window=**new** JFrame("Return Book");  JLabel l1,l2;  l1=**new** JLabel("Title");  l1.setBounds(30,15, 100,30); //x axis, y axis, width, height    JTextField F\_title = **new** JTextField();  F\_title.setBounds(110, 15, 200, 30);    JButton add\_but=**new** JButton("Return Book");  add\_but.setBounds(110,155,200,25);  add\_but.addActionListener(**new** ActionListener() { //Perform action    **public** **void** actionPerformed(ActionEvent e){  String title = F\_title.getText();  String owner = " ";    System.***out***.print(title);  System.***out***.print("meow");  System.***out***.print(owner);    String date = " ";  //System.out.print(date);    String newDate = " ";  //System.out.println(newDate);      //JFrame f = new JFrame("Current Members");  SqlServerDbAccessor sqda = **new** SqlServerDbAccessor();  sqda.setDbName("SSE657-Library");  sqda.connectToDb();  String sql = "UPDATE Books\_Instance " +"SET owner ='"+owner+"'"  + " WHERE Title='"+title+"'";  String sql2 = "UPDATE Book\_Lookup " +"SET owner ='"+owner+"'"  + " WHERE Title='"+title+"'";  String sql3 = "UPDATE Books\_Instance " +"SET CheckoutDate ='"+date+"'"  + " WHERE Title='"+title+"'";  String sql4 = "UPDATE Books\_Instance " +"SET ReturnDate ='"+newDate+"'"  + " WHERE Title='"+title+"'";    **try** {    Statement stmt = sqda.getConnection().createStatement();  stmt.executeUpdate(sql);  stmt.executeUpdate(sql2);  stmt.executeUpdate(sql3);  stmt.executeUpdate(sql4);  window.dispatchEvent(**new** WindowEvent(window, WindowEvent.***WINDOW\_CLOSING***));  }  **catch** (SQLException e1) {  // **TODO** Auto-generated catch block  JOptionPane.*showMessageDialog*(**null**, e1);  }    //System.out.println("title "+title+" genre "+genre);  }  });    LibraryMain lm = **new** LibraryMain();    lm.setIconOnTitleBar(**new** ImageIcon("images/libraryIcon.png").getImage());  //Library Icons - Free Download, PNG and SVG. (n.d.). Retrieved September 23, 2020, from https://icons8.com/icons/set/library  window.setIconImage(lm.getIconOnTitleBar());    //window.add(F\_author); //add password  window.add(add\_but);//adding button in JFrame  window.add(F\_title); //add user  // window.add(F\_genre);  //window.add(F\_bookID);  // window.add(F\_owner);  window.add(l1);  //window.add(l2);  //window.add(l3);  //window.add(l4);  //window.add(l5);    window.setSize(400,300);  window.setLayout(**null**);  window.setVisible(**true**);//making the frame visible  window.setLocationRelativeTo(**null**);  }    **public** **void** deleteBook() {  JFrame window=**new** JFrame("Delete Book");  JLabel l1;  l1=**new** JLabel("Title");  l1.setBounds(30,15, 100,30); //x axis, y axis, width, height    JTextField F\_title = **new** JTextField();  F\_title.setBounds(110, 15, 200, 30);    JButton add\_but=**new** JButton("Delete Book");  add\_but.setBounds(110,50,200,25);  add\_but.addActionListener(**new** ActionListener() { //Perform action    **public** **void** actionPerformed(ActionEvent e){  //String author = F\_author.getText();  String title = F\_title.getText();    SqlServerDbAccessor sqda = **new** SqlServerDbAccessor();  sqda.setDbName("SSE657-Library");  sqda.connectToDb();  String sql = "DELETE FROM Book\_Lookup WHERE Title = '" + title + "'" ;  //String sql2 = "DELETE FROM Books\_Instance WHERE Title = '" + title + "'" ;  **try** {    Statement stmt = sqda.getConnection().createStatement();  stmt.executeUpdate(sql);  //stmt.executeUpdate(sql2);    window.dispatchEvent(**new** WindowEvent(window, WindowEvent.***WINDOW\_CLOSING***));  }  **catch** (SQLException e1) {  // **TODO** Auto-generated catch block  JOptionPane.*showMessageDialog*(**null**, e1);  }    //System.out.println("title "+title+" genre "+genre);  }  });    LibraryMain lm = **new** LibraryMain();    lm.setIconOnTitleBar(**new** ImageIcon("images/libraryIcon.png").getImage());  //Library Icons - Free Download, PNG and SVG. (n.d.). Retrieved September 23, 2020, from https://icons8.com/icons/set/library  window.setIconImage(lm.getIconOnTitleBar());    window.add(add\_but);//adding button in JFrame  window.add(F\_title);  window.add(l1);      window.setSize(400,300);  window.setLayout(**null**);  window.setVisible(**true**);//making the frame visible  window.setLocationRelativeTo(**null**);  }    **public** **void** deleteUser() {  JFrame window=**new** JFrame("Delete User");  JLabel l1;  l1=**new** JLabel("Username");  l1.setBounds(30,15, 100,30); //x axis, y axis, width, height    JTextField F\_title = **new** JTextField();  F\_title.setBounds(110, 15, 200, 30);    JButton add\_but=**new** JButton("Delete User");  add\_but.setBounds(110,50,200,25);  add\_but.addActionListener(**new** ActionListener() { //Perform action    **public** **void** actionPerformed(ActionEvent e){  //String author = F\_author.getText();  String user = F\_title.getText();    SqlServerDbAccessor sqda = **new** SqlServerDbAccessor();  sqda.setDbName("SSE657-Library");  sqda.connectToDb();  String sql = "DELETE FROM Members WHERE username = '" + user + "'" ;  //String sql2 = "DELETE FROM Books\_Instance WHERE Title = '" + title + "'" ;  **try** {    Statement stmt = sqda.getConnection().createStatement();  stmt.executeUpdate(sql);  //stmt.executeUpdate(sql2);    window.dispatchEvent(**new** WindowEvent(window, WindowEvent.***WINDOW\_CLOSING***));  }  **catch** (SQLException e1) {  // **TODO** Auto-generated catch block  JOptionPane.*showMessageDialog*(**null**, e1);  }    //System.out.println("title "+title+" genre "+genre);  }  });    LibraryMain lm = **new** LibraryMain();    lm.setIconOnTitleBar(**new** ImageIcon("images/libraryIcon.png").getImage());  //Library Icons - Free Download, PNG and SVG. (n.d.). Retrieved September 23, 2020, from https://icons8.com/icons/set/library  window.setIconImage(lm.getIconOnTitleBar());    window.add(add\_but);//adding button in JFrame  window.add(F\_title);  window.add(l1);      window.setSize(400,300);  window.setLayout(**null**);  window.setVisible(**true**);//making the frame visible  window.setLocationRelativeTo(**null**);  }  } |

# Appendix 6. SqlServerDbAccessor class

|  |
| --- |
| package csdbdao;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  //import java.sql.ResultSet;  //import java.sql.Statement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.sql.Statement;  import java.util.ArrayList;  import library.Login;  public class SqlServerDbAccessor {  private Connection con;  // private Statement stmt;  private PreparedStatement prepStmt;  // private ResultSet rs;    private String connectionUrl;    private String defaultConnUrl = "jdbc:sqlserver://;" +  "servername=csdata.cd4sevot432y.us-east-1.rds.amazonaws.com;"  + "user=csc312cloud;password=c3s!c2Cld;";  // + "databaseName=JLBookstore;";    public SqlServerDbAccessor() {  connectionUrl = defaultConnUrl;  }    public SqlServerDbAccessor(String serverName, String user, String pwd,  String dbName) {  connectionUrl = "jdbc:sqlserver://;";  connectionUrl += "servername=" + serverName + ";";  connectionUrl += "user=" + user + ";";  connectionUrl += "password=" + pwd + ";";  connectionUrl += "databaseName=" + dbName + ";";  }      public void setDbName(String dbName) {  connectionUrl += "databaseName=" + dbName;  }    public void connectToDb() {  try {  // Establish the connection.  Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  con = DriverManager.getConnection(connectionUrl);  } catch (Exception e) {  e.printStackTrace();  }  }  public PreparedStatement getPrepStmt() {  // TODO Auto-generated method stub  return prepStmt;  }  public Connection getConnection() {  // TODO Auto-generated method stub  return con;  }  public String getUrl() {  // TODO Auto-generated method stub  return connectionUrl;  }    /\*public static ArrayList<String> loadEntriesFromDb(String file)  {  ArrayList<String> subjects = new ArrayList<String>();  SqlServerDbAccessor sqda = new SqlServerDbAccessor();  String SQL = "SELECT \* FROM " + file;  String[] row = new String[4];  try {  Statement stmt = sqda.getConnection().createStatement();  ResultSet rs = stmt.executeQuery(SQL);    while(rs.next()) {  for (int i=1; i<=4; i++) {  row[i-1] = ((rs.getString(i) == null) ? "" : rs.getString(i));  subjects.add(row[i-1]);  }    }    return subjects;  } catch (SQLException e) {  e.printStackTrace();  }  return null;      }    public static void main(String[] args) {  // TODO Auto-generated method stub    SqlServerDbAccessor sqda = new SqlServerDbAccessor();  sqda.setDbName("SSE657-Library");  sqda.connectToDb();  sqda.loadEntriesFromDb("Books\_Instance");  }\*/ |