```java
/*
Jafar Hashim
CS50 AP
May 18 2022
 */

package com.library.kumonlibrary;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.stage.Stage;

import java.io.IOException;

public class main extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(main.class.getResource("main-view.fxml"));
        Scene scene = new Scene(fxmlLoader.load());
        stage.getIcons().add(new Image("file:fish.png"));
        stage.setTitle("Hello!");
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch();
```

```
30          new DatabaseManager();
31      }
32  }
```

```java
/*
Jafar Hashim
CS50 AP
May 18 2022
 */

package com.library.kumonlibrary;

public class BookClub {

    private int id;
    private String bookName;
    private String authorName;
    private int totalAmt;
    private int amtAvailable;
    private boolean active;

    public BookClub(int id, String bookName, String authorName, int totalAmt, int amtAvailable) {
        this.id = id;
        this.bookName = bookName;
        this.authorName = authorName;
        this.totalAmt = totalAmt;
        this.amtAvailable = amtAvailable;
    }

    public int getId() {
        return id;
    }

```

```
30      public void setId(int id) {
31          this.id = id;
32      }
33
34      public String getBookName() {
35          return bookName;
36      }
37
38      public void setBookName(String bookName) {
39          this.bookName = bookName;
40      }
41
42      public String getAuthorName() {
43          return authorName;
44      }
45
46      public void setAuthorName(String authorName) {
47          this.authorName = authorName;
48      }
49
50      public int getTotalAmt() {
51          return totalAmt;
52      }
53
54      public void setTotalAmt(int totalAmt) {
55          this.totalAmt = totalAmt;
56      }
57
58      public int getAmtAvailable() {
```

```
59          return amtAvailable;
60      }
61
62      public void setAmtAvailable(int amtAvailable) {
63          this.amtAvailable = amtAvailable;
64      }
65  }
66
```

```java
/*
Jafar Hashim
CS50 AP
May 18 2022
 */

package com.library.kumonlibrary;

import com.jfoenix.controls.JFXButton;
import com.jfoenix.controls.JFXCheckBox;
import com.jfoenix.controls.JFXListView;
import com.jfoenix.controls.JFXTextArea;
import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;
import org.controlsfx.control.Notifications;

import java.io.IOException;
import java.sql.SQLException;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.List;
import java.util.Objects;
```

```
30
31
32 public class MainController {
33
34     @FXML
35     private VBox userViewVbox;
36
37     @FXML
38     private VBox signUpVbox;
39
40     // @FXML
41     //private Material randoButton;
42
43     @FXML
44     private Button signUpButton;
45
46     @FXML
47     private StackPane stackPane;
48
49     @FXML
50     private JFXTextArea addressUpdate;
51
52     @FXML
53     private MenuItem connectDB;
54
55     @FXML
56     private Text welcomeLabel;
57
58     @FXML
```

```
59      public Label detailName;
60
61      @FXML
62      public Label detailMemberSince;
63
64      @FXML
65      public Label detailDOB;
66
67      @FXML
68      public JFXTextArea detailAddress;
69
70      @FXML
71      public JFXTextArea detailEmail;
72
73      @FXML
74      public JFXTextArea detailPhoneNumber;
75
76      @FXML
77      public Text detailBooksCheckedOut;
78
79      @FXML
80      private JFXListView<String> userListView;
81
82      @FXML
83      private MenuItem testSelect;
84
85      @FXML
86      private JFXButton selectUser;
87
```

```
88      @FXML
89      private JFXTextArea signUpFirstName;
90
91      @FXML
92      private JFXTextArea signUpLastName;
93
94      @FXML
95      private DatePicker signUpDOB;
96
97      @FXML
98      private JFXTextArea signUpAddress;
99
100     @FXML
101     private JFXTextArea signUpPhone;
102
103     @FXML
104     private JFXTextArea signUpEmail;
105
106     @FXML
107     private JFXCheckBox signUpTerms;
108
109     @FXML
110     private JFXButton completeSignUp;
111
112     @FXML
113     private JFXButton confirmUpdate;
114
115     @FXML
116     private JFXButton deleteUser;
```

```
117
118     @FXML
119     private JFXButton viewUsers;
120
121     @FXML
122     private TableView<BookClub> books;
123
124     @FXML
125     private TableColumn<BookClub, String> colBookName;
126
127     @FXML
128     private TableColumn<BookClub, String> colAuthorName;
129
130     @FXML
131     private TableColumn<BookClub, Integer> colAmountAvailable;
132
133     @FXML
134     private StackPane tableStackPane;
135
136     @FXML
137     private VBox checkedOutBooks;
138
139     @FXML
140     private JFXListView<String> checkedOutBookList;
141
142     @FXML
143     private JFXListView<String> checkedOutDateList;
144
145     @FXML
```

```
146     private JFXButton viewCheckedOut;
147
148     @FXML
149     private TextField search;
150
151     @FXML
152     private TextField searchBooks;
153
154     @FXML
155     private JFXButton searchBooksButton;
156
157
158     private DatabaseManager databaseManager = new DatabaseManager();
159
160
161     private ObservableList<String> users = FXCollections.observableArrayList();
162     private ObservableList<String> userData = FXCollections.observableArrayList();
163     private ObservableList<String> inventory = FXCollections.observableArrayList();
164
165     String updateName;
166     //public String selectedUser;
167
168
169     @FXML
170     private void onNewUser() {
171         stackPane.getChildren().clear();
172         stackPane.getChildren().add(signUpVbox);
173     }
174
```

```
175     @FXML
176     private void onReturnHome() {
177         stackPane.getChildren().clear();
178         stackPane.getChildren().add(userViewVbox);
179     }
180 
181     @FXML
182     public void onViewUsers() throws IOException {
183         //this.databaseManager.userList();
184         ObservableList<String> users = FXCollections.observableArrayList(this.databaseManager.userList());
185         userListView.setItems(users);
186         //System.out.println(users);
187     }
188 
189     @FXML
190     public void onSelectUser() throws IOException, SQLException {
191         onReturnHome();
192         String selectedItem = userListView.getSelectionModel().getSelectedItem();
193         System.out.println(selectedItem);
194         StringBuilder sb = new StringBuilder();
195         List<String> userData = this.databaseManager.userSelected(selectedItem);
196         sb.append(userData.get(0));
197         String sba = sb.toString();
198         welcomeLabel.setText("Welcome " + sba);
199         sb.delete(0, 30);
200         System.out.println(sb);
201         sb.append(userData.get(0));
202         detailName.setText(String.valueOf(sb));
203         updateName = (String.valueOf(sb));
```

```
204         sb.delete(0, 30);
205         detailMemberSince.setText(String.valueOf(sb.append("Member Since: ").append(userData.get(1))));
206         sb.delete(0, 30);
207         detailDOB.setText(String.valueOf(sb.append("Date of Birth:  ").append(userData.get(2))));
208         sb.delete(0, 30);
209         detailAddress.setText(String.valueOf(sb.append(userData.get(3))));
210         sb.delete(0, 30);
211         detailEmail.setText(String.valueOf(sb.append(userData.get(5))));
212         sb.delete(0, 30);
213         detailPhoneNumber.setText(String.valueOf(sb.append(userData.get(4))));
214         sb.delete(0, 30);
215         detailBooksCheckedOut.setText(String.valueOf(sb.append(userData.get(6))));
216         //String  bco = String.valueOf(Integer.parseInt(userData.get(6)));
217         //String bco = userData.get(6);
218         //System.out.println(bco);
219         //return bco;
220         onBrowse();
221     }
222 
223 
224     @FXML
225     public void onCompleteSignUp() throws SQLException, IOException {
226 
227         String newUserName = signUpFirstName.getText();
228         // String newUserLastName = signUpLastName.getText();
229         LocalDate x = signUpDOB.getValue();
230         if(x == null){ System.out.println("BOI"); return;}
231         String newSignUpDOB = x.format(DateTimeFormatter.ofPattern("dd-MMM-yy"));
232         String newUserSince = LocalDate.now().format(DateTimeFormatter.ofPattern("dd-MMM-yy"));
```

```
233         String newUserAddress = signUpAddress.getText();
234         String newUserPhone = signUpPhone.getText();String newUserEmail = signUpEmail.getText();
235 
236 
237         databaseManager.newUserCreation(newUserName, newUserSince, newSignUpDOB, newUserAddress, newUserPhone,
    newUserEmail);
238         onReturnHome();
239         onViewUsers();
240 
241         Alert alert = new Alert(Alert.AlertType.INFORMATION);
242         alert.setTitle("About");
243         alert.setHeaderText(null);
244         alert.setContentText("Welcome to ");
245 
246         alert.showAndWait();
247     }
248 
249     @FXML
250     public void onConfirm() {
251 
252         String updateAddress = detailAddress.getText();
253         String updateEmail = detailEmail.getText();
254         String updatePhone = detailPhoneNumber.getText();
255 
256         Notifications.create()
257                 .title("Edits Made")
258                 .showInformation();
259 
260         databaseManager.updateUserInfo(updateName, updateAddress, updatePhone, updateEmail);
```

```
261     }
262
263     @FXML
264     public void onDelete() throws SQLException, IOException {
265         Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
266         alert.setTitle("Confirmation");
267         alert.setHeaderText(null);
268         alert.setContentText("Are you sure you want to delete your account?");
269         //Optional<ButtonType> result = alert.showAndWait();
270         if (alert.showAndWait().get() == ButtonType.OK) {
271             databaseManager.deleteUser(updateName);
272             databaseManager.connectDB();
273             Notifications.create()
274                     .title("Account Deleted :(")
275                     .showInformation();
276             onReturnHome();
277             onViewUsers();
278         } else {
279             return;
280         }
281     }
282
283     @FXML
284     private void onViewCheckedOut() throws SQLException {
285         tableStackPane.getChildren().clear();
286         tableStackPane.getChildren().add(checkedOutBooks);
287
288         ObservableList<String> checkedOutBooksObservable = FXCollections.observableArrayList(this.
    databaseManager.getCheckedOutBooks(detailName.getText()));
```

```
289         checkedOutBookList.setItems(checkedOutBooksObservable);
290 
291         ObservableList<String> checkedOutDatesObservable = FXCollections.observableArrayList(this.
    databaseManager.getCheckedOutDates(detailName.getText()));
292         checkedOutDateList.setItems(checkedOutDatesObservable);
293     }
294 
295     @FXML
296     private void onReturnBook() throws SQLException, IOException {
297         String selectedItem = checkedOutBookList.getSelectionModel().getSelectedItem();
298         String thing1 = databaseManager.getCheckedOutBooks(detailName.getText()).get(0);
299         String thing2 = databaseManager.getCheckedOutBooks(detailName.getText()).get(1);
300         String thing3 = databaseManager.getCheckedOutBooks(detailName.getText()).get(2);
301 
302         if (Objects.equals(selectedItem, thing1)){
303             databaseManager.returnBook("Book1", "DateCheckedOutBook1", detailName.getText(), selectedItem);
304         }
305         if (Objects.equals(selectedItem, thing2)){
306             databaseManager.returnBook("Book2", "DateCheckedOutBook2", detailName.getText(), selectedItem);
307         }
308         if (Objects.equals(selectedItem, thing3)){
309             databaseManager.returnBook("Book3", "DateCheckedOutBook3", detailName.getText(), selectedItem);
310         }
311         Notifications.create()
312                 .title("Book Returned!")
313                 .showInformation();
314 
315         onViewCheckedOut();
316         onSelectUser();
```

```
317     }
318
319
320     @FXML
321     private void onBrowse() throws SQLException {
322         tableStackPane.getChildren().clear();
323         tableStackPane.getChildren().add(books);
324
325         ObservableList<BookClub> bookList = databaseManager.getBooks();
326         colBookName.setCellValueFactory(new PropertyValueFactory<BookClub, String>("bookName"));
327         colAuthorName.setCellValueFactory(new PropertyValueFactory<BookClub, String>("authorName"));
328         colAmountAvailable.setCellValueFactory(new PropertyValueFactory<BookClub, Integer>("amtAvailable"));
329         books.setItems(bookList);
330     }
331
332     @FXML
333     private void onSearchUsers() throws SQLException {
334         String searchKeyword = search.getText();
335         //databaseManager.searchUser(searchKeyword);
336         ObservableList<String> users = FXCollections.observableArrayList(this.databaseManager.searchUser(
    searchKeyword));
337         userListView.setItems(users);
338     }
339
340     @FXML
341     private void onSearchBook() throws SQLException {
342         String searchKeyword = searchBooks.getText();
343         ObservableList<BookClub> bookList = databaseManager.searchBooks(searchKeyword);
344         colBookName.setCellValueFactory(new PropertyValueFactory<>("bookName"));
```

```
345         colAuthorName.setCellValueFactory(new PropertyValueFactory<BookClub, String>("authorName"));
346         colAmountAvailable.setCellValueFactory(new PropertyValueFactory<BookClub, Integer>("amtAvailable"));
347         books.setItems(bookList);
348 
349     }
350 
351     @FXML
352     private void onCheckOut() throws SQLException, IOException {
353         //String bco = userData.get(6);
354         System.out.println(detailBooksCheckedOut.getText());
355         LocalDate localDate = LocalDate.now();
356         DateTimeFormatter dtf = DateTimeFormatter.ofPattern("MM/dd/uuuu");
357         String dateFr = dtf.format(localDate);
358 
359         String selectedBook = books.getSelectionModel().getSelectedItem().getBookName();
360         String wBook;
361         String wDateX;
362         if(books.getSelectionModel().getSelectedItem().getAmtAvailable() <= 0){return;}
363         if(detailBooksCheckedOut.getText().equals("3")){return;}
364         if(detailBooksCheckedOut.getText().equals("2")){
365             wBook = "Book3";
366             wDateX = "DateCheckedOutBook3";
367             databaseManager.checkOutBook(selectedBook, detailName.getText(), wBook, wDateX, dateFr);
368         }
369         if(detailBooksCheckedOut.getText().equals("1")){
370             wBook = "Book2";
371             wDateX = "DateCheckedOutBook2";
372             databaseManager.checkOutBook(selectedBook, detailName.getText(), wBook, wDateX, dateFr);
373         }
```

```
374         if(detailBooksCheckedOut.getText().equals("0")){
375             wBook = "Book1";
376             wDateX = "DateCheckedOutBook1";
377             databaseManager.checkOutBook(selectedBook, detailName.getText(), wBook, wDateX, dateFr);
378         }
379         Notifications.create()
380                 .title("Checked Out: " + selectedBook)
381                 .showInformation();
382         onSelectUser();
383         onBrowse();
384     }
385 
386     @FXML
387     public void onExit(javafx.event.ActionEvent event) {
388         Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
389         alert.setTitle("Exit");
390         alert.setHeaderText("Press Ok to exit");
391 
392         if(alert.showAndWait().get() == ButtonType.OK){
393             Platform.exit();
394         }
395     }
396 }
```

```java
/*
Jafar Hashim
CS50 AP
May 18 2022
 */

package com.library.kumonlibrary;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

import java.io.File;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import static java.sql.DriverManager.getConnection;

public class DatabaseManager {

    private final File databaseFile = new File("data/database.db");

    private Connection connection;

    public DatabaseManager() {
        this.connection = null;
```

```
30          try {
31              if (!databaseFile.exists()) {
32                  databaseFile.getParentFile().mkdirs();
33                  databaseFile.createNewFile();
34              }
35              connectDB().prepareStatement("CREATE TABLE IF NOT EXISTS users(Name varchar(30), MemberSince varchar
    (15), DOB varchar(15), Address varchar(50), PhoneNumber varchar(15), BooksCheckedOut int, Book1 varchar(100),
    Book2 varchar(100), Book3 varchar(100), id int UNIQUE AUTO_INCREMENT);").executeUpdate();
36              connectDB().prepareStatement("CREATE TABLE IF NOT EXISTS books(Name varchar(100), AuthorName varchar
    (40), TotalAmount int, AmountAvailable int, id int UNIQUE AUTO_INCREMENT);").executeUpdate();
37
38          } catch (Exception ex) {
39              ex.printStackTrace();
40          }
41      }
42
43      public Connection connectDB() {
44          try {
45              return getConnection("jdbc:mysql://localhost:3306/library", "root", "224668Jh");
46          } catch (SQLException e) {
47              e.printStackTrace();
48          }
49          return null;
50      }
51
52      public List<String> userSelected(String selectedItem) {
53          List<String> userData= new ArrayList<>();
54          try {
55              System.out.println(selectedItem);
```

```
56              PreparedStatement ps = (PreparedStatement) connectDB().prepareStatement("SELECT * FROM users WHERE
    Name = ?");
57              ps.setString(1, selectedItem);
58              ResultSet rs = ps.executeQuery();
59
60              while (rs.next()) {
61                  userData.add(rs.getString("Name"));
62                  //userData.add(rs.getString("LastName"));
63                  userData.add(rs.getString("MemberSince"));
64                  userData.add(rs.getString("DOB"));
65                  userData.add(rs.getString("Address"));
66                  userData.add(rs.getString("PhoneNumber"));
67                  userData.add(rs.getString("Email"));
68                  userData.add(rs.getString("BooksCheckedOut"));
69                  userData.add(rs.getString("Book1"));
70                  userData.add(rs.getString("DateCheckedOutBook1"));
71                  userData.add(rs.getString("Book2"));
72                  userData.add(rs.getString("DateCheckedOutBook2"));
73                  userData.add(rs.getString("Book3"));
74                  userData.add(rs.getString("DateCheckedOutBook3"));
75
76              }
77              System.out.println(userData);
78              //return FirstName;
79          } catch (SQLException e) {
80              e.printStackTrace();
81          }
82
83          return userData;
```

```
84      }
85
86      public List<String> userList() {
87          List<String> users = new ArrayList<>();
88          try{
89              ResultSet rs = connectDB().prepareStatement("SELECT * FROM users;").executeQuery();
90
91              while (rs.next()) {
92                  users.add(rs.getString("Name")); //+ (" ") + rs.getString("LastName"));
93              }
94              System.out.println(users);
95          } catch (SQLException e) {
96              e.printStackTrace();
97          }
98          return users;
99      }
100
101     public void newUserCreation(String newUserName, String newUserSince, String newUserDOB, String
    newUserAddress, String newUserPhone, String newUserEmail) throws SQLException {
102         try {
103             PreparedStatement ps = connectDB().prepareStatement("INSERT INTO users (Name, MemberSince, DOB,
    Address, PhoneNumber, Email, BooksCheckedOut, Book1, DateCheckedOutBook1, Book2, DateCheckedOutBook2, Book3,
    DateCheckedOutBook3) VALUES (?, ?, ?, ?, ?, ?, 0, ' ', ' ', ' ', ' ', ' ', ' ')");
104             ps.setString(1, newUserName);
105             ps.setString(2,newUserSince);
106             ps.setString(3, newUserDOB);
107             ps.setString(4, newUserAddress);
108             ps.setString(5, newUserPhone);
109             ps.setString(6, newUserEmail);
```

```
110             ps.executeUpdate();
111
112         } catch (SQLException e) {
113             e.printStackTrace();
114         }
115     }
116
117     public void updateUserInfo(String updateName, String updateAddress, String updatePhone, String updateEmail
    ) {
118         try{
119             PreparedStatement ps = connectDB().prepareStatement("Update users SET Address = (?), PhoneNumber
     = (?), Email = (?) WHERE name = (?)");
120             ps.setString(1, updateAddress);
121             ps.setString(2, updatePhone);
122             ps.setString(3, updateEmail);
123             ps.setString(4, updateName);
124             ps.executeUpdate();
125         } catch (SQLException e) {
126             e.printStackTrace();
127         }
128     }
129
130
131     public void deleteUser(String updateName) throws SQLException {
132         PreparedStatement ps = connectDB().prepareStatement("DELETE FROM users WHERE Name = ?");
133         ps.setString(1, updateName);
134         ps.executeUpdate();
135         System.out.println(updateName);
136     }
```

```
137
138     public List<String> searchUser(String searchKeyword) throws SQLException {
139         List<String> searchedUser = FXCollections.observableArrayList();
140         try {
141             PreparedStatement ps = connectDB().prepareStatement("SELECT * FROM users WHERE Name LIKE ? ");
142             ps.setString(1, "%" + searchKeyword + "%");
143             ResultSet rs = ps.executeQuery();
144             System.out.println(ps);
145
146             while (rs.next()){
147                 searchedUser.add(rs.getString("Name"));
148             }
149         } catch (SQLException e) {
150             e.printStackTrace();
151         }
152         return searchedUser;
153     }
154
155     public ObservableList<BookClub> searchBooks(String searchKeyword) throws SQLException {
156         ObservableList<BookClub> searchedBookList = FXCollections.observableArrayList();
157         try {
158             PreparedStatement ps = connectDB().prepareStatement("SELECT * FROM books WHERE Name LIKE ? ");
159             ps.setString(1, "%" + searchKeyword + "%");
160             ResultSet rs = ps.executeQuery();
161             System.out.println(ps);
162
163             while (rs.next()){
164                 //searchedBookList.add(rs.getString("Name"));
165                 BookClub book = new BookClub(rs.getInt("id"), rs.getString("name"), rs.getString("authorname"
```

```
), rs.getInt("totalamount"), rs.getInt("amountavailable"));
                searchedBookList.add(book);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return searchedBookList;
    }

    public ObservableList<BookClub> getBooks() throws SQLException {
        ObservableList<BookClub> bookList = FXCollections.observableArrayList();
        PreparedStatement ps = connectDB().prepareStatement("SELECT * FROM books");
        ResultSet rs = ps.executeQuery();

        while(rs.next()){
            BookClub book = new BookClub(rs.getInt("id"), rs.getString("name"), rs.getString("authorname"), rs.
    getInt("totalamount"), rs.getInt("amountavailable"));
            bookList.add(book);
        }
        System.out.println(bookList);
        return bookList;
    }



    public List <String> getCheckedOutBooks(String name){
        List <String> checkedOutBooksList = new ArrayList<>();
        try{
            PreparedStatement ps = connectDB().prepareStatement("SELECT Book1, Book2, Book3 FROM users WHERE
```

```
192 name = ?");
193             ps.setString(1, name);
194             ResultSet rs = ps.executeQuery();
195 
196             while (rs.next()){
197                 checkedOutBooksList.add(rs.getString("Book1"));
198                 checkedOutBooksList.add(rs.getString("Book2"));
199                 checkedOutBooksList.add(rs.getString("Book3"));
200             }
201         } catch (SQLException e) {
202             e.printStackTrace();
203         }
204         System.out.println(checkedOutBooksList);
205         return checkedOutBooksList;
206     }
207 
208     public List <String> getCheckedOutDates(String name){
209         List <String> checkedOutDatesList = new ArrayList<>();
210         try{
211             PreparedStatement ps = connectDB().prepareStatement("SELECT DateCheckedOutBook1,
    DateCheckedOutBook2, DateCheckedOutBook3 FROM users WHERE name = ?");
212             ps.setString(1, name);
213             ResultSet rs = ps.executeQuery();
214 
215             while (rs.next()){
216                 checkedOutDatesList.add(rs.getString("DateCheckedOutBook1"));
217                 checkedOutDatesList.add(rs.getString("DateCheckedOutBook2"));
218                 checkedOutDatesList.add(rs.getString("DateCheckedOutBook3"));
219             }
```

```
220         } catch (SQLException e) {
221             e.printStackTrace();
222         }
223         System.out.println(checkedOutDatesList);
224         return checkedOutDatesList;
225     }
226 
227     public void returnBook(String book, String date, String name, String selectedItem) throws SQLException {
228         PreparedStatement ps = connectDB().prepareStatement("UPDATE users SET " + book + " = ' ' WHERE name
     = ?");
229         PreparedStatement psa = connectDB().prepareStatement("UPDATE Users SET " + date + "= ' ' WHERE name
     = ?");
230         PreparedStatement psb = connectDB().prepareStatement("UPDATE users SET BooksCheckedOut = IF (
    BooksCheckedOut > 0, BooksCheckedOut-1, BooksCheckedOut) WHERE name = ?;");
231         PreparedStatement psc = connectDB().prepareStatement("UPDATE books SET AmountAvailable =
    AmountAvailable+1 WHERE name = ? ");
232         //PreparedStatement psd = connectDB().prepareStatement("select BooksCheckedOUt from users where name
     = ? ");
233 
234         ps.setString(1, name);
235         psa.setString(1, name);
236         psb.setString(1, name);
237         psc.setString(1, selectedItem);
238         //psd.setString(1, name);
239         ps.execute();
240         psa.execute();
241         psb.execute();
242         psc.execute();
243 //         ResultSet rs = psd.executeQuery();
```

```
244
245
246     }
247
248     public void checkOutBook(String book, String name, String wBook, String wDateX, String wDate) throws
    SQLException {
249         PreparedStatement ps = connectDB().prepareStatement("UPDATE books SET AmountAvailable = AmountAvailable
    -1 WHERE name= ?");
250         PreparedStatement psa = connectDB().prepareStatement("UPDATE users SET " + wBook + "= ? WHERE name = ?"
    );
251         PreparedStatement psb = connectDB().prepareStatement("UPDATE users SET BooksCheckedOut =
    BooksCheckedOut+1 WHERE name = ?");
252         PreparedStatement psc = connectDB().prepareStatement("UPDATE users SET " + wDateX + " = ? WHERE name
     = ?");
253         ps.setString(1, book);
254         psa.setString(1, book);
255         psa.setString(2, name);
256         psb.setString(1, name);
257         psc.setString(1, wDate);
258         psc.setString(2, name);
259         ps.execute();
260         psa.execute();
261         psb.execute();
262         psc.execute();
263         System.out.println("yay");
264     }
265
266
267
```

```
268 }
```