package models;

import java.sql.\*;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.util.\*;

import controllers.Display;

public class DaoModel extends Display{

DBConnect conn = null;

Statement stmt = null;

public static String IdSelectedFromCombobox = null;

/\*\*

\* The DaoModel is a

\* default constructor.

\*/

public DaoModel() {

//create db object instance

conn = new DBConnect();

}

/\*\*

\* The retrieveRecords method retrieves

\* username and password from the administrator table.

\* @return returns the returned records

\*/

public String[] retrieve\_admin\_Records() {

ResultSet rs1 = null;

String[] values = new String[2];

try {

//open a connection

stmt = conn.connect().createStatement();

System.out.println("Retrieving Records from the table...");

String sql = "SELECT adminuser,adminpass from aak\_padmin";

System.out.println(sql);

// execute the query

rs1 = stmt.executeQuery(sql);

rs1.next();

System.out.println(rs1.getString("adminuser"));

values[0] = rs1.getString("adminuser");

values[1] = rs1.getString("adminpass");

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

}

catch (SQLException se) {

se.printStackTrace();

}

return values;

}

/\*\*

\* The retrieveStudentRecords method retrieves

\* the username and password based on the student data

\* entered in the login page

\* @param emailOrUsernameEntered the username entered by the student

\* @return the values

\*/

public String[] retrieveCustomerRecords(String emailOrUsernameEntered) {

ResultSet rs1 = null;

String[] values = new String[2];

try {

//open a connection

stmt = conn.connect().createStatement();

System.out.println("Connected to the database.\n"+

"\nRetrieving Records from the table...");

String sql = "SELECT custuser,custpass from aak\_pcust WHERE custuser IN ('"+ emailOrUsernameEntered + "')";

System.out.println(sql);

// execute the query

rs1 = stmt.executeQuery(sql);

while(rs1.next()) {

values[0] = rs1.getString("custuser");

values[1] = rs1.getString("custpass");}

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

}

catch (SQLException se) {

se.printStackTrace();

}

return values;

}

/\*\*

\* The retrieveStaffRecords method retrieves

\* the username and password from staff table

\* based on the staff data entered in

\* the login page.

\* @param emailOrUsernameEntered the username entered by the staff

\* @return the values

\*/

public String[] retrieveEmployeeRecords(String emailOrUsernameEntered) {

ResultSet rs1 = null;

String[] values = new String[2];

try {

//open a connection

stmt = conn.connect().createStatement();

System.out.println("Connected to the database.\n"+

"\nRetrieving Records from the table...");

String sql = "SELECT empuser,emppass from aak\_pemp WHERE empuser IN ('"+ emailOrUsernameEntered + "')";

System.out.println(sql);

// execute the query

rs1 = stmt.executeQuery(sql);

while(rs1.next()) {

values[0] = rs1.getString("empuser");

values[1] = rs1.getString("emppass");

}

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

}

catch (SQLException se) {

se.printStackTrace();

}

return values;

}

/\*\*

\* The retrieveSearchTableRecords method retrieves

\* the table records based on article or books selection upon entered

\* data in the Search box.

\* @return the values based on the records retrieved from the table

\* @throws SQLException

\*/

public ResultSet retrieveSearchTableRecords() throws SQLException {

// TODO Auto-generated method stub

ResultSet rs = null;

String sql = null;

String sql1 = null;

String sql2 = null;

stmt = conn.connect().createStatement();

if(articleSelected) {

sql = "SELECT \* from aak\_prooms WHERE id LIKE ";

sql1 = " OR (type LIKE ";

sql2= ") OR (rice LIKE ";

}else if (bookSelected) {

sql = "SELECT \* from aak\_prooms WHERE id LIKE ";

sql1 = " OR (type LIKE ";

sql2= ") OR (price LIKE ";

}

//query along quotes

String sqlQuery = sql +"\""+ query+ "%"+"\""+sql1+"\""+ query+ "%"+"\""+sql2 +"\""+ query+ "%"+"\"" +")";

// execute the query

rs = stmt.executeQuery(sqlQuery);

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

return rs;

}

/\*\*

\* The retrieveStaffBookRecords method retrieves

\* the table records based on article or books selection from staff.

\* @param selection is the toggle selection of staff.

\* @param queryReceived is the data entered by the staff in the Search box.

\* @return the values based on the records retrieved from the table.

\* @throws SQLException

\*/

public ResultSet retrieveStaffBookRecords(int selection,String queryReceived) throws SQLException {

// TODO Auto-generated method stub

ResultSet rs = null;

String sql = null;

String sql1 = null;

String sql2 = null;

stmt = conn.connect().createStatement();

if(selection ==1) {

sql = "SELECT \* from aak\_prooms WHERE id LIKE ";

sql1 = " OR (name LIKE ";

sql2= ") OR (author LIKE ";

}else if (selection ==2) {

sql = "SELECT \* from aak\_prooms WHERE id LIKE ";

sql1 = " OR (name LIKE ";

sql2= ") OR (author LIKE ";

}

System.out.println(queryReceived);

System.out.println(sql);

//query along quotes

String sqlQuery = sql +"\""+ queryReceived+ "%"+"\""+sql1+"\""+ queryReceived+ "%"+"\""+sql2 +"\""+ queryReceived+ "%"+"\"" +")";

// execute the query

rs = stmt.executeQuery(sqlQuery);

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

return rs;

}

/\*\*

\* The retrieveStudentTableRecords method retrieves

\* the table records based on article or books selection from student.

\* @param selection is the toggle selection of staff.

\* @param queryReceived is the data entered by the student in the Search box.

\* @return the values based on the records retrieved from the table.

\* @throws SQLException

\*/

public ResultSet retrieveStudentTableRecords(int selection, String query) throws SQLException {

// TODO Auto-generated method stub

ResultSet rs = null;

String sql = null;

String sql1 = null;

String sql2 = null;

System.out.println(selection);

stmt = conn.connect().createStatement();

if(selection==1) {

sql = "SELECT \* from aak\_prooms WHERE roomid LIKE ";

sql1 = " OR (roomtype LIKE ";

sql2= ") OR (roomprice LIKE ";

}else if (selection==2) {

sql = "SELECT \* from aak\_prooms WHERE roomid LIKE ";

sql1 = " OR (roomtype LIKE ";

sql2= ") OR (roomprice LIKE ";

}

//query along quotes

String sqlQuery = sql +"\""+ query+ "%"+"\""+sql1+"\""+ query+ "%"+"\""+sql2 +"\""+ query+ "%"+"\"" +")";

// execute the query

rs = stmt.executeQuery(sqlQuery);

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

return rs;

}

/\*\*

\* The retrieveIdRecords method retrieves

\* the id from the table based on article or books selection from student.

\* @param selection is the toggle selection of staff.

\* @param query is the data entered by the student in the Search box.

\* @return the id based on the records retrieved from the table.

\* @throws SQLException

\*/

public ResultSet retrieveIdRecords(int selection, String query) throws SQLException {

// TODO Auto-generated method stub

ResultSet rs = null;

String sql = null;

String sql1 = null;

String sql2 = null;

String sql3 =null;

System.out.println(selection);

stmt = conn.connect().createStatement();

if(selection==1) {

System.out.println("Checker");

sql = "SELECT \* from aak\_prooms WHERE roomid LIKE ";

sql2 = " OR (roomtype LIKE ";

sql3 = ") OR (roomprice LIKE" ;

}else if (selection==2) {

sql = "SELECT \* from aak\_prooms WHERE roomid LIKE ";

sql1 = "AND roomisavailable = 'YES'";

sql2 = " OR (roomtype LIKE ";

sql3 = ") OR (roomprice LIKE" ;

}

System.out.println(query);

System.out.println(sql);

//query along quotes

String sqlQuery = sql +"\""+ query+ "%"+"\""+ sql2+"\""+ query+ "%"+"\""+sql3 +"\""+ query+ "%"+"\")";

// execute the query

System.out.println(sqlQuery);

rs = stmt.executeQuery(sqlQuery);

System.out.println("\n\nRetrieving Records successful LOL");

// close the connection

conn.connect().close();

return rs;

}

/\*\*

\* The insertBookRecords method inserts the records entered

\* by the administrator/staff after Add Book is selected.

\* @param insertBooks is the list of records entered in

\* the text boxes by the administrator.

\* @param num is article/book selected

\* @throws SQLException

\*/

public void insertBookRecords(ArrayList<String> insertBooks, int num) throws SQLException {

// TODO Auto-generated method stub

int i = 1;

DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");

LocalDate localDate = LocalDate.now();

System.out.println(dtf.format(localDate));

if(num==1) {

String query = "INSERT INTO aak\_prooms (id, "

+ "name,isbn,author,publishedBy,isRefferal,isAvailable,addedDate) VALUES (?,?,?,?,?,?,?,?)";

}else {

String query = "INSERT INTO aak\_prooms (id, "

+ "name,isbn,author,publishedBy,isRefferal,isAvailable,addedDate) VALUES (?,?,?,?,?,?,?,?)";

}

try(PreparedStatement stmt = conn.connect().prepareStatement(query)) {

for (int index = 0; index < insertBooks.size(); index++) {

stmt.setString(i, insertBooks.get(index));

i++;

}

stmt.setString(7, "Yes");

stmt.setString(8, dtf.format(localDate));

stmt.execute();

} catch (Exception e) {

// log info somewhere at least until it's properly tested/

// you implement a better way of handling the error

e.printStackTrace(System.err);

}

}

/\*\*

\* The insertBookRecords method inserts the records entered

\* by the administrator after Add Staff is selected.

\* @param insertStaff is the list of records entered in

\* the text boxes by the administrator.

\* @throws SQLException

\*/

public void insertStaffRecords(ArrayList<String> insertStaff) throws SQLException {

// TODO Auto-generated method stub

int i = 1;

DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");

LocalDate localDate = LocalDate.now();

System.out.println(dtf.format(localDate));

String query = "INSERT INTO aak\_prooms (employeeId, "

+ "staffName,username,password,phoneNumber,address) VALUES (?,?,?,?,?,?)";

try(PreparedStatement stmt = conn.connect().prepareStatement(query)) {

for (int index = 0; index < insertStaff.size(); index++) {

stmt.setString(i, insertStaff.get(index));

i++;

}

stmt.execute();

} catch (Exception e) {

// log info somewhere at least until it's properly tested/

e.printStackTrace(System.err);

}

}

/\*\*

\* The retrieveIDRecords retrieves the ID's from

\* articlestable for selection of ID in update book page

\* @return resultset containing ID's

\* @throws SQLException

\*/

public ResultSet retrieveIDRecords() throws SQLException {

// TODO Auto-generated method stub

ResultSet rs = null;

String sql = null;

stmt = conn.connect().createStatement();

sql = "SELECT roomid from aak\_prooms";

// execute the query

rs = stmt.executeQuery(sql);

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

return rs;

}

/\*\*

\* The retrieveBookUpdateRecords retrieves the ID's from

\* articlestable for selection of ID in update book page

\* @return resultset containing ID's

\* @throws SQLException

\*/

public ResultSet retrieveBookUpdateRecords() throws SQLException {

// TODO Auto-generated method stub

ResultSet rs = null;

String sql = null;

stmt = conn.connect().createStatement();

sql = "SELECT id from aak\_prooms";

// execute the query

rs = stmt.executeQuery(sql);

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

return rs;

}

/\*\*

\* The retrieveStaffIDRecords retrieves the ID's from

\* staff table for selection of ID in update staff page

\* @return resultset containing employeeID's

\* @throws SQLException

\*/

public ResultSet retrieveStaffIDRecords() throws SQLException {

// TODO Auto-generated method stub

ResultSet rs = null;

String sql = null;

stmt = conn.connect().createStatement();

sql = "SELECT employeeId from aak\_prooms";

// execute the query

rs = stmt.executeQuery(sql);

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

return rs;

}

/\*\*

\* The retrieveRecordsbasedOnId retrieves the

\* records upon selection of ID in the Update Book page

\* @param updateBookId the Id selected the administrator/staff

\* @param selectedButton is selected toggle button articles/books

\* @return the resultset containing records from articlestable/bookstable

\* @throws SQLException

\*/

public ResultSet retrieveRecordsbasedOnId(String updateBookId, int selectedButton) throws SQLException {

ResultSet rs = null;

String sql = null;

stmt = conn.connect().createStatement();

IdSelectedFromCombobox = updateBookId;

if(selectedButton==1) {

sql = "SELECT \* from aak\_prooms WHERE id IN ('"+ updateBookId + "')";

}else if(selectedButton ==2) {

sql = "SELECT \* from aak\_prooms WHERE id IN ('"+ updateBookId + "')";

}

// execute the query

rs = stmt.executeQuery(sql);

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

return rs;

}

/\*\*

\* The retrieveStaffDataId retrieves the staff records based

\* on id selected by the administrator in the delete Staff page.

\* @param deleteStaffId the id selected the administrator

\* @return resultset containing the records upon id selection

\* @throws SQLException

\*/

public ResultSet retrieveStaffDataId(String deleteStaffId) throws SQLException {

ResultSet rs = null;

String sql = null;

stmt = conn.connect().createStatement();

IdSelectedFromCombobox = deleteStaffId;

sql = "SELECT \* from aak\_prooms WHERE employeeId IN ('"+ deleteStaffId + "')";

// execute the query

rs = stmt.executeQuery(sql);

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

return rs;

}

/\*\*

\* The retrieveStaffDataId retrieves the staff records based

\* on id selected by the administrator in the update Staff page.

\* @param updateStaffId the id selected the administrator.

\* @return resultset containing the records upon id selection.

\* @throws SQLException

\*/

public ResultSet retrieveStaffDataOnId(String updateStaffId) throws SQLException {

ResultSet rs = null;

String sql = null;

stmt = conn.connect().createStatement();

IdSelectedFromCombobox = updateStaffId;

sql = "SELECT \* from aak\_prooms WHERE employeeId IN ('"+ updateStaffId + "')";

// execute the query

rs = stmt.executeQuery(sql);

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

return rs;

}

/\*\*

\* The insertupdatedRecords inserts the updated

\* records into articlestable/bookstable

\* @param insertBooks is the list of data

\* entered by the administrator/staff

\* @param num is the selection articles/books

\* @throws SQLException

\*/

public void insertUpdatedRecords(ArrayList<String> insertBooks, int num) throws SQLException {

// TODO Auto-generated method stub

int i = 1;

DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");

LocalDate localDate = LocalDate.now();

System.out.println(dtf.format(localDate));

if(num==1) {

String query = "UPDATE aak\_prooms SET name = ?, isbn = ?,author = ?,publishedBy = ?, isRefferal =?,addedDate =? WHERE id = ?";

try(PreparedStatement stmt = conn.connect().prepareStatement(query)) {

for (int index = 0; index < insertBooks.size(); index++) {

stmt.setString(i, insertBooks.get(index));

i++;

}

stmt.setString(6, dtf.format(localDate));

stmt.setString(7, IdSelectedFromCombobox);

System.out.println(query);

System.out.println(stmt);

stmt.execute();

}catch (Exception e) {

// log info somewhere at least until it's properly tested/

// you implement a better way of handling the error

e.printStackTrace(System.err);

}

}else {

String query = "UPDATE aak\_prooms SET name = ?, isbn = ?,author = ?,publishedBy = ?, isRefferal =?,addedDate =? WHERE id = ?";

try(PreparedStatement stmt = conn.connect().prepareStatement(query)) {

for (int index = 0; index < insertBooks.size(); index++) {

stmt.setString(i, insertBooks.get(index));

i++;

}

stmt.setString(6, dtf.format(localDate));

stmt.setString(7, IdSelectedFromCombobox);

System.out.println(query);

System.out.println(stmt);

stmt.execute();

}catch (Exception e) {

// log info somewhere at least until it's properly tested/

// you implement a better way of handling the error

e.printStackTrace(System.err);

}

}

}

/\*\*

\* The insertupdatedStaffData inserts the updated

\* records into stafftable.

\* @param insertDetails is the list of staff data

\* entered by the administrator

\* @return

\* @throws SQLException

\*/

public ResultSet insertupdatedStaffData(ArrayList<String> insertDetails) throws SQLException {

// TODO Auto-generated method stub

int i = 1;

DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");

LocalDate localDate = LocalDate.now();

System.out.println(dtf.format(localDate));

String query = "UPDATE aak\_prooms SET staffname = ?, username = ?,phoneNumber = ?,address = ? WHERE employeeId = ?";

try(PreparedStatement stmt = conn.connect().prepareStatement(query)) {

for (int index = 0; index < insertDetails.size(); index++) {

stmt.setString(i, insertDetails.get(index));

i++;

}

stmt.setString(5, IdSelectedFromCombobox);

System.out.println(query);

System.out.println(stmt);

stmt.execute();

} catch (Exception e) {

// log info somewhere at least until it's properly tested/

// you implement a better way of handling the error

e.printStackTrace(System.err);

}

return null;

}

/\*\*

\* The retrieveIDRecordsForDeletePage retrieves the

\* articles records for delete page upon id selection.

\* @return the resultset containing the articles table records

\* @throws SQLException

\*/

public ResultSet retrieveIDRecordsForDeletePage() throws SQLException {

// TODO Auto-generated method stub

ResultSet rs = null;

String sql = null;

stmt = conn.connect().createStatement();

sql = "SELECT roomid from aak\_prooms";

// execute the query

rs = stmt.executeQuery(sql);

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

return rs;

}

/\*\*

\* The retrieveStaffIDDeletePage retrieves the

\* employeeid from the staff table for delete page

\* @return the resultset containing employee id's

\* @throws SQLException

\*/

public ResultSet retrieveStaffIDDeletePage() throws SQLException {

// TODO Auto-generated method stub

ResultSet rs = null;

String sql = null;

stmt = conn.connect().createStatement();

sql = "SELECT employeeid from aak\_prooms";

// execute the query

rs = stmt.executeQuery(sql);

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

return rs;

}

/\*\*

\* The deleteRecordsTable method deletes the records from

\* the articlestable/bookstable

\* @param insertBooks is the list containing the

\* data of articles/books

\* @param num is articles/books

\* @throws SQLException

\*/

public void deleteRecordsTable(ArrayList<String> insertBooks, int num) throws SQLException {

// TODO Auto-generated method stub

if(num==1) {

String query = "DELETE FROM aak\_prooms WHERE id = ?";

PreparedStatement stmt = conn.connect().prepareStatement(query);

stmt.setString(1, IdSelectedFromCombobox);

System.out.println(query);

System.out.println(stmt);

stmt.execute();

}else {

String query = "DELETE FROM aak\_prooms WHERE id = ?";

PreparedStatement stmt = conn.connect().prepareStatement(query);

stmt.setString(1, IdSelectedFromCombobox);

System.out.println(query);

System.out.println(stmt);

stmt.execute();

}

}

/\*\*

\* The deleteRecordsTable method deletes the records from

\* the staff table.

\* @param deleteDetails is the list containing the

\* data of staff

\* @return

\* @throws SQLException

\*/

public void deleteStaffData(ArrayList<String> deleteDetails) throws SQLException {

// TODO Auto-generated method stub

String query = "DELETE FROM aak\_prooms WHERE employeeId = ?";

PreparedStatement stmt = conn.connect().prepareStatement(query);

stmt.setString(1, IdSelectedFromCombobox);

System.out.println(query);

System.out.println(stmt);

stmt.execute();

}

/\*\*

\* The insertBorrowDetails method updates the records

\* in the table after the student borrows the book/article

\* @param idSelected is the article/book id selected

\* by the student

\* @param borrowerId is the id of the student

\* @param borrowerName is the name of the student

\* who is borrowing the book

\* @param selection is the article/book selection

\* @throws SQLException

\*/

public void insertBorrowDetails(String idSelected,String borrowerId, String borrowerName,int selection) throws SQLException {

// TODO Auto-generated method stub

DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");

LocalDate localDate = LocalDate.now();

System.out.println(dtf.format(localDate));

LocalDate returningDate = LocalDate.now().plusDays(15);

if(selection==1) {

String query = "UPDATE aak\_prooms SET borrowedByStudentId = ?, borrowedByName = ?,borrowedDate = ?,returnDate = ?, isAvailable =? WHERE id = ?";

try(PreparedStatement stmt = conn.connect().prepareStatement(query)) {

stmt.setString(1, borrowerId);

stmt.setString(2, borrowerName);

stmt.setString(3, dtf.format(localDate));

stmt.setString(4, dtf.format(returningDate));

stmt.setString(5, "No");

stmt.setString(6, idSelected);

System.out.println(query);

System.out.println(stmt);

stmt.execute();

}catch (Exception e) {

// log info somewhere at least until it's properly tested/

// you implement a better way of handling the error

e.printStackTrace(System.err);

}

}

else if(selection ==2) {

String query = "UPDATE aak\_prooms SET borrowedByStudentId = ?, borrowedByName = ?,borrowedDate = ?,returnDate = ?, isAvailable =? WHERE id = ?";

try(PreparedStatement stmt = conn.connect().prepareStatement(query)) {

stmt.setString(1, borrowerId);

stmt.setString(2, borrowerName);

stmt.setString(3, dtf.format(localDate));

stmt.setString(4, dtf.format(returningDate));

stmt.setString(5, "No");

stmt.setString(6, idSelected);

System.out.println(query);

System.out.println(stmt);

stmt.execute();

}catch (Exception e) {

// log info somewhere at least until it's properly tested/

// you implement a better way of handling the error

e.printStackTrace(System.err);

}

}

}

/\*\*

\* The retrieveStudentDetails retrieves the

\* data based on the student email id.

\* @param studEmail is the student email id.

\* @return resultset containing the data of the

\* respective student

\* @throws SQLException

\*/

public ResultSet retrieveStudentDetails(String studEmail) throws SQLException {

ResultSet rs = null;

String sql = null;

stmt = conn.connect().createStatement();

sql = "SELECT \* from aak\_prooms WHERE usernameOrEmail IN ('"+ studEmail + "')";

// execute the query

rs = stmt.executeQuery(sql);

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

return rs;

}

/\*\*

\* The retrieveBorrowedRecords method gives the list

\* of books borrowed by the student

\* @param studId the id of the student

\* @param selected is the articles/books

\* @return resultset containing the records

\* @throws SQLException

\*/

public ResultSet retrieveBorrowedRecords(String custId, int selected) throws SQLException {

ResultSet rs = null;

String sql = null;

stmt = conn.connect().createStatement();

sql = "SELECT custid from aak\_pcust where custuser = ('"+ custId + "')";

System.out.println(sql);

rs = stmt.executeQuery(sql);

rs.next();

if(selected==3){

sql = "SELECT "

+ "borrowedByStudentId,borrowedByName,name,isbn,author,"

+ "publishedby,id,"

+ "borrowedDate,returnDate "

+ "from aak\_prooms WHERE borrowedByStudentId = ('"+ rs.getString("custId") + "')";

}

else if(selected==4){

sql = "SELECT "

+ "borrowedByStudentId,borrowedByName,name,isbn,author,"

+ "publishedby,id,"

+ "borrowedDate,returnDate "

+ "from aak\_prooms WHERE borrowedByStudentId = ('"+ rs.getString("studentId") + "')";

}

rs = stmt.executeQuery(sql);

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

return rs;

}

/\*\*

\* The retrieveStaffData retrieves the employee id

\* from the staff table.

\* @param idReceived is the staff id received

\* upon selection.

\* @return resultset containing the records

\* of staff table

\* @throws SQLException

\*/

public ResultSet retrieveStaffData(String idReceived) throws SQLException {

ResultSet rs = null;

String sql = null;

stmt = conn.connect().createStatement();

//IdSelectedFromCombobox = updateBookId;

sql = "SELECT employeeId from aak\_prooms WHERE employeeId IN ('"+ idReceived + "')";

// execute the query

rs = stmt.executeQuery(sql);

System.out.println("\n\nRetrieving Records successful");

// close the connection

conn.connect().close();

return rs;

}

}