

EECS 3421

PROJECT 2

MICHAEL WILLIAMS

211087798

ABSTRACT

The goal of this project is to write an application program in java that carries out various tasks of a transaction for a user that wishes to purchase a book. The program is entirely coded in java but uses the YRB database from the first project and the JDBC on PRISM in order to allow the program to interact with the YRB database.

The application program has many steps for the user interacting with it. The exact specifications of the steps can be seen in the project description in the course website but below is a shorter summarized version

The program starts by requesting a customer ID continuously until a valid one is given. Once a valid ID has been recognized and is displayed, the user is asked if they wish to update the information. If they do, the user agrees and supplies the new information to the application program. If not, the program simply moves forward. Next, the program displays all categories and allows the user to choose 1. It then displays all the books under that category and again allows the user to choose a book title. If the user enters an incorrect category, they are prompted to try again. If the user enters a book title that does not exist, they are taken back to the category menu to reselect a category.

Once the user selects an existing book title, all books of that name will appear in a list and the user may select the version of the book they wish to buy. The minimum price for that book under their current club memberships will automatically be chosen and displayed for the user to see. After this, the user enters the quantity they wish to purchase and the total price is displayed. The user is then asked to approve the purchase and if they do, the purchase information is stored in the database. Otherwise, the user is notified that the program is ending and the program closes.

The specifics of the program and varying results per varied responses are shown in greater detail in the USER MANUAL/WALKTHROUGH portion of the lab. This section also includes the step to step guide on starting the program.

TOOLS

- MobaXTerminal, used as an alternative to PUTTY for SSH connections
- Java Eclipse for programming the beginning portion of the program
- Jedit for programming the large remainder of the program
- Snipping tool used for screenshots
- Microsoft Word used for writing the report

USER MANUAL/WALKTHROUGH

PART A: OPENING THE PROGRAM:

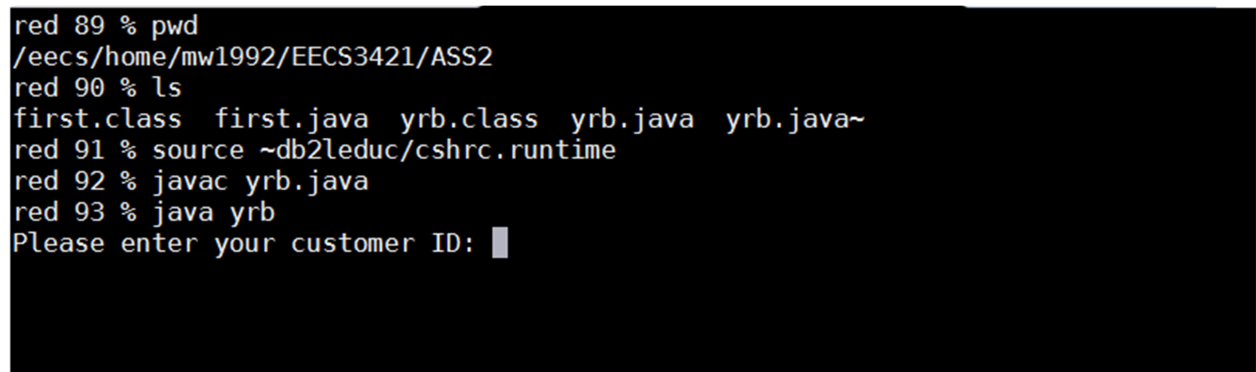
To run the program, first open the command or terminal window and go to the folder in which the program resides. Afterwards, type in the following:

```
source ~db2leduc/cshrc.runtime
```

This will prime the shell and allow the program to run. Afterwards, while still in the same folder, type in the following to compile the program:

```
javac yrb.java
```

Then, type in “java yrb” to execute the program. You should now be running and the program should ask for a customer ID.



```
red 89 % pwd
/eecs/home/mw1992/EECS3421/ASS2
red 90 % ls
first.class first.java yrb.class yrb.java yrb.java~
red 91 % source ~db2leduc/cshrc.runtime
red 92 % javac yrb.java
red 93 % java yrb
Please enter your customer ID: █
```

Figure 1: showing exact method of running program, from line red 91

PART B: ACQUIRING CUSTOMER ID AND UPDATING INFORMATION:

At this point, the user must enter a customer ID and the program will search to see if the given ID exists in the database. If it does, the program displays the customer

information, comprised of the customer ID, name and city. If it does not, the program will return an error message and prompt the user to reenter a valid ID. The customer ID is always a number and searching for otherwise (ie words or characters) will cause an exception to be thrown, and close the program.

```
Please enter your customer ID: -23
The provided ID isn't in our database. Please enter a registered customer ID: -1
The provided ID isn't in our database. Please enter a registered customer ID: 0
The provided ID isn't in our database. Please enter a registered customer ID: 99
The provided ID isn't in our database. Please enter a registered customer ID: 123
The provided ID isn't in our database. Please enter a registered customer ID: 4
ID: 4      Name: ciahel      City: William
```

Figure 2: inserting ID

After a valid ID has been given and the information is displayed, the program then asks the user if they would like to update the customer information. The user may respond with “yes” or “no”. Any other response will result in an error message and a prompt to insert a valid answer. If the user chooses “yes”, they will be asked to insert their new desired name and city. After providing these two, the program updates the customer’s information in the database and displays the new customer information.

```
ID: 4      Name: ciahel      City: William
Would you like to update your current information?
If so, enter Yes, otherwise Enter No to continue.
isdfsfd
Sorry, we can't understand what you said. Please Answer again with Yes or No.

3242
Sorry, we can't understand what you said. Please Answer again with Yes or No.

-1
Sorry, we can't understand what you said. Please Answer again with Yes or No.

yes
Great! Let's proceed to update your info.

Please enter your new name:
Michael

Please enter your new city:
Brampton
Info updated! Here is the new official info:
ID: 4      Name: Michael      City: Brampton
```

Figure 3: Updating customer information

If the user chooses “no”, the program will send a message of acknowledgement and proceed with the program

```
ID: 4          Name: ciahel      City: William
Would you like to update your current information?
If so, enter Yes, otherwise Enter No to continue.
no
Ok, then let's move on.
```

Figure 4: Deciding not to update customer information

PART C: CHOOSING A CATEGORY AND BOOK TITLE:

Next, the program displays all categories in the database and prompts the user to select one. Failure to type in a category amongst the list will result in an error message and a prompt to reenter a correct category choice.

```
Please choose a category:
23ewr

That category choice does not exist. Try again. The categories are displayed below:
children
cooking
drama
guide
history
horror
humor
mystery
phil
romance
science
travel

Please choose a category:
PHIL

Here are the books in the category phil:
Plato Rocks
Databases aren't
Is Math is fun?
```

Figure 5: Choosing a category

Once a given category has been selected, the program will display all available books from that category. The program will then prompt the user to select a book from the list. The book title choices are case sensitive and failure to type in the chosen book title exactly as it appears in the program will result in an error message. Furthermore, if the given title is not found in the database, the program will display the list of categories again and revert back to asking the user to select one, along with a subsequent book title.

```
Here are the books in the category phil:
Plato Rocks
Databases aren't
Is Math is fun?

Choose a book from the list. Remember, the selections are case sensitive
plato rocks

No book with that title exists, please try again.

The categories are displayed below:
children
cooking
drama
guide
history
horror
humor
mystery
phil
romance
science
travel

Please choose a category:
█
```

Figure 6: Incorrectly choosing a book title

When the user enters an existing category and book title amongst the given lists, the book information is displayed, comprised of the book title, year, language, and weight. The program may find multiple books with the same category and name and thus will display all such books and their information.

```

Please choose a category:
science

Here are the books in the category science:
Tensor Calculus made Easy
Pluto Collides With Mars
Quarks and Other Matter
Transmorgification
Rats Like Us
Databases made Real Hard
Eigen Eigen
SQL Kills!
Dogs are not Cats
Math is fun!
Cats are not Dogs
Chats ne sont pas Chiens
Quantum Databases

Choose a book from the list. Remember, the selections are case sensitive
Transmorgification

Listing # 1:
Title: Transmorgification      Year: 2000      Language: Plutonian      Weight: 2911

```

Figure 7: Correctly choosing a category and book title

PART D: SELECTING BOOK AND QUANTITY FOR PURCHASE:

Next, the user selects the listing number of the book they wish to purchase. The selection must be one of the given listings values and failure to provide a valid number will result in an error message and a prompt to reenter a valid listed number. Providing other input (ie words or characters) will result in an exception being thrown and the program closing. Once a valid number is given, the lowest price available to the user, for that book, will be retrieved and displayed.

```

Listing # 1:
Title: Transmorgification      Year: 2000      Language: Plutonian      Weight: 2911

Enter the listings number of the book you wish to buy:
-2
That is not a valid choice.

Enter the listings number of the book you wish to buy:
0
That is not a valid choice.

Enter the listings number of the book you wish to buy:
2
That is not a valid choice.

Enter the listings number of the book you wish to buy:
1

The minimum price available with your current memberships for that book is: $282.73

```

Figure 8: Choosing a listings number

After, the program will prompt the user to enter the quantity they wish to purchase. Failure to provide a valid number (anything less than 1) will result in an error message and a prompt to reenter a valid quantity. Providing other input (ie words or characters) will result in an exception being thrown and the program closing. Once a valid quantity is entered, the program will display the total price of the purchase.

```
Enter the amount of this title you would like to buy:
0
You have entered an invalid quantity. Please enter a value of at least 1. -3
You have entered an invalid quantity. Please enter a value of at least 1. 4
The total price of your purchase is $147.80.
```

Figure 9: Inserting quantity

PART E: CONFIRMING OR REJECTING THE PURCHASE:

The program will now ask the user if they approve of the price and wish to confirm the purchase. The user may respond with “yes” or “no”. Any other response will result in an error message and a prompt to re-insert a valid answer. If the user chooses “yes”, the purchase information is stored in the database and a thank you message is displayed to the user to confirm their purchase. The program then closes.

```
Would you like to make this purchase? Please respond yes or no
werd
That is an incorrect response. Please respond with yes or no
2343s
That is an incorrect response. Please respond with yes or no
yes
TRANSACTION SUCCESSFUL. Here are the details:
Customer ID: 2      Club: YRB Gold      Title: Plato Rocks      Year: 1975      Quantity: 3
Price: $110.85
Program is now closing
```

Figure 10: Confirming purchase

If the user selects “no”, a message is displayed acknowledging that no purchase is to be made. The program then proceeds to close

```
Would you like to make this purchase? Please respond yes or no
no
Oh well. Have a nice day! Program is now closing.
```

Figure 11: Reecting purchase

SOURCE CODE

```
/*  
  
* This application program carries out various tasks of a transaction for the user.  
  
* Generally speaking, the program allows a user to purchase a book from the database.  
  
* The database will display relevant information to allow the user to carry out the purchase  
  
* and will insert a record of the purchase into the database upon completion  
  
*/  
  
  
//imported classes  
  
import java.util.*;  
  
import java.net.*;  
  
import java.text.*;  
  
import java.lang.*;  
  
import java.io.*;  
  
import java.sql.*;  
  
  
/*  
  
* This is the class that will run when invoked in the main method. Basically this class  
  
* will run the entirety of the program's specification, with the assistance of additional methods  
  
* which will be invoked throughout the programs execution.  
  
*/  
  
public class yrb {  
  
  
    //declare variables
```

```

private Connection conDB; // Connection to the database system.

private String url;      // URL

private String ID;       //Customer ID


private Integer customerID; //integer version of the Customer ID

private String customerName; // Name of the customer

private String customerCity; //city the customer belongs to


private String ACK;      //used for accepting strings of yes or

private String ACK2;     //used in the same way as ACK

private Boolean moveON;   //used for deciding whether to move on with code during
certain looping segments

private String newName;   //new Name when updating

private String newCity;   //New city when updating


private ArrayList <String> categories; //stores categories for user interaction


    Map <Integer, ArrayList <String>> map = new HashMap <Integer, ArrayList <String>> ();
//stores listings

private Integer size;     //holds numberof book listings

private String categoryChoice; //holds user's category choice


private ArrayList <String> titleList; //contains list of all book titles

private String book;      //name of book to be searched for

private Integer bookNum;   //number used to represent a book

```

```

        private String title;        //title of query

private Integer year;        //year of query

private String language;        //language of query

private Integer weight;        //weight of query


private Double lowestPrice;    //stores lowest price of given book title

private Integer quantity;    //quantity of books to be bought

private Double total;        //total price of purchase

private String club;        //club name


/*

* Constructor for class. Program is implemented here so that the entire sequence
* of tasks is performed once class instance is executed

*/

public yrb() {

    // Set up the DB connection.

    try {

        Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();

        //check for possible exceptions

    } catch (ClassNotFoundException e) {

        e.printStackTrace();

```

```
System.exit(0);
```

```
} catch (InstantiationException e) {
```

```
    e.printStackTrace();
```

```
    System.exit(0);
```

```
} catch (IllegalAccessException e) {
```

```
    e.printStackTrace();
```

```
    System.exit(0);
```

```
}//all exception checking done
```

```
// URL establishment
```

```
url = "jdbc:db2:c3421a";
```

```
// Initialize the connection.
```

```
try {
```

```
    // Connect with a fall-thru id & password
```

```
    conDB = DriverManager.getConnection(url);
```

```
} catch (SQLException e) {
```

```
    System.out.print("\nSQL: database connection error.\n");
```

```
    System.out.println(e.toString());
```

```
    System.exit(0);
```

```

    } //catch exception

    // Turn autocommit off here

    try {

        conDB.setAutoCommit(false);

    } catch (SQLException e) {

        System.out.print("\nFailed trying to turn autocommit off.\n");

        e.printStackTrace();

        System.exit(0);

    } //catch exception

    titleList = new ArrayList < String > (); //used later in the program

    //obtain customer ID

    System.out.print("Please enter your customer ID: ");

    Scanner input = new Scanner(System.in);

    ID = input.nextLine();

    //keep asking for ID until proper ID is given

    while (find_customer(ID) == false) {

        System.out.print("The provided ID isn't in our database. Please enter a
registered customer ID: ");

        ID = input.nextLine();

```

```

} //while loop to obtain valid customer ID

System.out.println("Would you like to update your current information? ");
System.out.println("If so, enter Yes, otherwise Enter No to continue. ");
ACK = input.nextLine().toLowerCase();
moveON = false;

//keep loping until proper answer is given
while (moveON == false) {

    if (ACK.equals("yes")) {

        System.out.println("Great! Let's proceed to update your info.\n");

        System.out.println("Please enter your new name: ");
        newName = input.nextLine();

        System.out.println("\nPlease enter your new city: ");
        newCity = input.nextLine();

        update_customer(newName, newCity, customerID);

        System.out.println("");
        moveON = true;

    } //if yes, update info

```

```

else if (ACK.equals("no")) {

    System.out.println("Ok, then let's move on. \n");

    moveON = true;

}

//if no, don't do anything

else {

    System.out.println("Sorry, we can't understand what you said.
Please Answer again with Yes or No. \n");

    moveON = false;

    ACK = input.nextLine().toLowerCase();

}

//if they put neither, make them enter the correct statement

}

//while loop

categories = new ArrayList < String > ();

categories = fetch_categories();

size = 0;

while (size < 1) {

    System.out.println("The categories are displayed below:

");

    for (String x: categories) { System.out.println(x); }

```

```

        System.out.println("");

        System.out.println("Please choose a category: ");

        categoryChoice = input.nextLine().toLowerCase();

        System.out.println("");

        while (categories.contains(categoryChoice) == false) {

            System.out.print("That category choice does not
exist. Try again. ");

            System.out.println("The categories are displayed
below:");

            for (String x: categories) { System.out.println(x); }

            System.out.println("");

            System.out.println("Please choose a category: ");

            categoryChoice = input.nextLine().toLowerCase();

            System.out.println("");

        } //while loop to recieve correct category choice

        System.out.println("Here are the books in the category " + categoryChoice + ": ");

        titleList = get_titles(categoryChoice);

        for (String x: titleList) { System.out.println(x); }

        System.out.println("");

        System.out.println("Choose a book from the list. Remember, the selections are case
sensitive ");

```



```

    book = input.nextLine();

    System.out.println("");

    find_book(book, categoryChoice);

    size = map.size();

    if (size == 0) {

        System.out.println("");

        System.out.println("No book with that title exists, please try again.");

        System.out.println("");

        continue;

    } // if statement

} // once out of this loop, it means a valid book has been chosen

    bookNum = 0;

    while (bookNum < 1 || bookNum > size) {

        System.out.println("");

        System.out.println("Enter the listings number of the book you wish to
buy: ");

        bookNum = input.nextInt();

        if (bookNum < 1 || bookNum > size) { System.out.println("That is not a
valid choice. "); }

    } // while loop to pick book number

```

```

title = map.get(bookNum).get(0);

year = Integer.parseInt(map.get(bookNum).get(1));

language = map.get(bookNum).get(2);

weight = Integer.parseInt((map.get(bookNum).get(3)));


        lowestPrice = min_price(categoryChoice, title, year, customerID);

club = get_club(year, customerID, title, lowestPrice);


        System.out.println("");

        System.out.println("The minimum price available with your current memberships
for that book is: $" + lowestPrice);

        System.out.println("\nEnter the amount of this title you would like to buy: ");

        quantity = input.nextInt();


        while (quantity < 1) {

                System.out.print("You have entered an invalid quantity. Please enter a value of
at least 1. ");

                quantity = input.nextInt();


        } //while loop to obtain correct quantity

```

```

        total = lowestPrice * quantity;

        System.out.printf("The total price of your purchase is $%.2f. \nWould you like to
make this purchase? ", total);

        System.out.println("Please respond yes or no ");

        moveON = false;

        int num = 0;

        while (moveON == false){

            ACK = input.nextLine().toLowerCase();

                                if (ACK.equals("yes")) {

                                    insert_purchase(customerID, club,
title, year, quantity);

                                    System.out.println("");

                                    System.out.print("TRANSACTION
SUCCESSFUL. ");

                                    System.out.printf("Here are the
details: \nCustomer ID: %d    Club: %s    Title: %s    Year: %d    Quantity: %d    Price:
$%.2f\n", customerID, club, title, year, quantity, total);

                                    moveON = true;

                                    System.out.println("Program is now
closing");

                                } else if (ACK.equals("no")){

                                    System.out.println("");

                                    System.out.println("Oh well. Have a nice
day! Program is now closing. ");

```

```
        moveON = true;

    } else if(num >= 1){

        System.out.println("That is an incorrect
response. Please respond with yes or no ");
```

```
    } else { num++;}

} //while loop
```

```
// Commit the transaction
```

```
try {

    conDB.commit();

} catch (SQLException e) {

    System.out.print("\nFailed trying to commit.\n");

    e.printStackTrace();

    System.exit(0);

} //catch exception
```

```
// Close the connection.
```

```
try {

    conDB.close();

} catch (SQLException e) {

    System.out.print("\nFailed trying to close the
connection.\n");
```

```
        e.printStackTrace();  
        System.exit(0);  
    } //catch exception
```

```
} //constructor
```

```
/*  
 * main method, used to execute instance of the program  
 */  
public static void main(String[] args) {  
    yrb x = new yrb();  
} //main method
```

```
/*  
 * find customer will check to see if the given ID matches a cid in the database  
 * If it does, method returns true, else false  
 */  
public boolean find_customer(String ID) {  
    String queryText = ""; // The SQL text.  
    PreparedStatement querySt = null; // The query handle.  
    ResultSet answers = null; // A cursor.  
    boolean inDB = false; // Return.
```

```
queryText = "SELECT *" + "FROM yrb_customer " + "WHERE cid = ? ";

// Prepare the query.

try {

    querySt = conDB.prepareStatement(queryText);

} catch (SQLException e) {

    System.out.println("SQL find_customer failed in prepare");

    System.out.println(e.toString());

    System.exit(0);

}

}

//catch exception

// Execute the query

try {

    querySt.setInt(1, Integer.parseInt(ID));

    answers = querySt.executeQuery();

} catch (SQLException e) {

    System.out.println("SQL find_customer failed in execute");

    System.out.println(e.toString());

    System.exit(0);

}
```

```

    } //catch exception

    // Any answer?
    try {
        if (answers.next()) {
            inDB = true;

            customerID = answers.getInt("cid");

            customerName = answers.getString("name");

            customerCity = answers.getString("city");

            System.out.println("ID: " + customerID + "      Name: " +
customerName + "      City: " + customerCity);

        } else { inDB = false; }

    } catch (SQLException e) {
        System.out.println("SQL find_customer failed in cursor.");
        System.out.println(e.toString());
        System.exit(0);

    } //catch exception

    // Close the cursor.
    try {

```

```

        answers.close();

    } catch (SQLException e) {

        System.out.print("SQL find_customer failed closing cursor.\n");

        System.out.println(e.toString());

        System.exit(0);

    } //catch exception

    // We're done with the handle.

    try {

        querySt.close();

    } catch (SQLException e) {

        System.out.print("SQL find_customer failed closing the
handle.\n");

        System.out.println(e.toString());

        System.exit(0);

    } //catch exception

    return inDB;

} //find_customer method

```



```

/*
    * update_customer updates the customer information in the database with the
given new name and city

    * The information replaced is that of the given customer ID.
*/

public void update_customer(String name, String city, Integer customerID){

    String queryText = ""; // The SQL text.

    PreparedStatement querySt = null; // The query handle.

    ResultSet answers = null; // A cursor.

    int intAnswers = 0; //return from query execution

    queryText = "UPDATE yrb_customer " + "SET name = ?, city = ? " + "WHERE cid =
? ";

    // Prepare the query.

    try {

        querySt = conDB.prepareStatement(queryText);

    } catch (SQLException e) {

        System.out.println("SQL update_customer failed in prepare");

        System.out.println(e.toString());

        System.exit(0);

    } //catch exception

```

```

        // Execute the query
    try {

        querySt.setString(1, name);

        querySt.setString(2, city);

        querySt.setInt(3, customerID);

        intAnswers = querySt.executeUpdate();

    } catch (SQLException e) {

        System.out.println("SQL update_customer failed in execute");

        System.out.println(e.toString());

        System.exit(0);

    } //catch exception

    // Any answer?

    if (intAnswers == 1) {

        System.out.println("Info updated! Here is the new official
info: ");

        find_customer(customerID.toString());

        System.out.println("");

    } //if statement

    // We're done with the handle.

```

```

        try {

            querySt.close();

        } catch (SQLException e) {

            System.out.print("SQL customer_update failed closing the
handle.\n");

            System.out.println(e.toString());

            System.exit(0);

        } //catch exception

    } //update_customer method

    /*
    *fetch_categories returns a list of the categories of the database
    */

    public ArrayList < String > fetch_categories(){

        String queryText = ""; // The SQL text.

        PreparedStatement querySt = null; // The query handle.

        ResultSet answers = null; // A cursor.

        ArrayList < String > temp = new ArrayList < String > ();

        queryText = "SELECT *" + "FROM yrb_category ";

```

```
// Prepare the query.

try {

    querySt = conDB.prepareStatement(queryText);

} catch (SQLException e) {

    System.out.println("SQL fetch_categories failed in prepare");

    System.out.println(e.toString());

    System.exit(0);

} //catch exception


// Execute the query.

try {

    answers = querySt.executeQuery();

} catch (SQLException e) {

    System.out.println("SQL fetch_categories failed in execute");

    System.out.println(e.toString());

    System.exit(0);

} //catch exception


// Any answer?
```

```
try {  
    for (int i = 1; answers.next(); i++) {  
        String category = answers.getString("cat");  
        temp.add(category);  
    } //for loop to add categories  
  
} catch (SQLException e) {  
    System.out.println("SQL fetch_categories failed in cursor.");  
    System.out.println(e.toString());  
    System.exit(0);  
  
} //catch exception  
  
// Close the cursor.  
try {  
    answers.close();  
  
} catch (SQLException e) {  
    System.out.print("SQL fetch_categories failed closing cursor.\n");  
    System.out.println(e.toString());  
    System.exit(0);  
  
} //catch exception
```

```

        // We're done with the handle.

        try {

            querySt.close();

        } catch (SQLException e) {

            System.out.print("SQL fetch_categories failed closing the
handle.\n");

            System.out.println(e.toString());

            System.exit(0);

        } // catch exception

        return temp;

    } // fetch_Categories method

    /*
    * get_titles obtains the titles of all books from the given category
    */

    public ArrayList <String> get_titles(String category) {

        String queryText = ""; // The SQL text.

        PreparedStatement querySt = null; // The query handle.

        ResultSet answers = null; // A cursor.

```

```
ArrayList <String> temp = new ArrayList <String> ();
```

```
queryText = "SELECT DISTINCT title, year " + " FROM yrb_book " + " WHERE cat =  
? ";
```

```
// Prepare the query.
```

```
try {
```

```
    querySt = conDB.prepareStatement(queryText);
```

```
} catch (SQLException e) {
```

```
    System.out.println("SQL get_titles failed in prepare");
```

```
    System.out.println(e.toString());
```

```
    System.exit(0);
```

```
//catch exception
```

```
// Execute the query.
```

```
try {
```

```
    querySt.setString(1, category);
```

```
    answers = querySt.executeQuery();
```

```
} catch (SQLException e) {
```

```
    System.out.println("SQL get_titles failed in execute");
```

```
    System.out.println(e.toString());
```

```
        System.exit(0);

    }

    // Any answer?
    try {

        for (int i = 1; answers.next(); i++) {

            String title = answers.getString("title");

            temp.add(title);

        }

    } catch (SQLException e) {

        System.out.println("SQL get_titles in cursor.");

        System.out.println(e.toString());

        System.exit(0);

    } //catch exception

    // Close the cursor.
    try {

        answers.close();

    } catch (SQLException e) {
```



```
        System.out.print("SQL get_titles failed closing cursor.\n");
        System.out.println(e.toString());
        System.exit(0);
```

```
    } //catch exception
```

```
    // We're done with the handle.
```

```
    try {
        querySt.close();
```

```
    } catch (SQLException e) {
        System.out.print("SQL get_titles failed closing the handle.\n");
        System.out.println(e.toString());
        System.exit(0);
```

```
    } //catch exception
```

```
    return temp;
```

```
    } //get_titles method
```

```
    /*
```

```
    *find_book creates a map of all the book listings and their respective
information
```

```

*/

public void find_book(String book, String category) {

    String queryText = ""; // The SQL text.

    PreparedStatement querySt = null; // The query handle.

    ResultSet answers = null; // A cursor

    queryText = "SELECT * " + "FROM yrb_book " + "WHERE title = ?" + " AND
cat = ?";

    String title;

    Integer weight;

    String language;

    Integer year;

    // Prepare the query.

    try {

        querySt = conDB.prepareStatement(queryText);

    } catch (SQLException e) {

        System.out.println("SQL find_book failed in prepare");

        System.out.println(e.toString());

        System.exit(0);

    } // catch exception

```

```
// Execute the query.

try {

    querySt.setString(1, book);

    querySt.setString(2, category);

    answers = querySt.executeQuery();

} catch (SQLException e) {

    System.out.println("SQL find_book failed in execute");

    System.out.println(e.toString());

    System.exit(0);

} //catch exception

int i = 0;

int j = 1;

// Any answer?

try {

    for (i = 0; answers.next(); i++) {

        title = answers.getString("title");

        year = answers.getInt("year");

        language = answers.getString("language");

        weight = answers.getInt("weight");

    }

}
```

```
        map.put(i + 1, new ArrayList <String> (Arrays.asList(title,
Integer.toString(year), language, Integer.toString(weight))));
```

```
    } //insert tuples into map
```

```
    if (i > 0) {
```

```
        while (j <= i) {
```

```
            title = map.get(j).get(0);
```

```
            year = Integer.parseInt(map.get(j).get(1));
```

```
            language = map.get(j).get(2);
```

```
            weight = Integer.parseInt((map.get(j).get(3)));
```

```
            System.out.println("Listing # " + j + ":");
```

```
            System.out.println("Title: " + title + "      Year: " +
year + "      Language: " + language + "      Weight: " + weight);
```

```
            j++;
```

```
        } //while loop
```

```
    }
```

```
    } catch (SQLException e) {
```

```
        System.out.println("SQL find_books failed in cursor.");
```

```
        System.out.println(e.toString());
```

```
        System.exit(0);
```

```
}//catch exception
```

```
// Close the cursor.
```

```
try {
```

```
    answers.close();
```

```
} catch (SQLException e) {
```

```
    System.out.print("SQL find_books failed closing cursor.\n");
```

```
    System.out.println(e.toString());
```

```
    System.exit(0);
```

```
}//catch exception
```

```
// We're done with the handle.
```

```
try {
```

```
    querySt.close();
```

```
} catch (SQLException e) {
```

```
    System.out.print("SQL find_books failed closing the handle.\n");
```

```
    System.out.println(e.toString());
```

```
    System.exit(0);
```

```
}//catch exception
```

```
}//find_book method
```

```
/*
```

*min_price returns the minimum available price of a book, given the category, book title, year of publication and the customerID

```
*/
```

```
public Double min_price(String category, String title, int year, int ID) {
```

```
    String queryText = ""; // The SQL text.
```

```
    PreparedStatement querySt = null; // The query handle.
```

```
    ResultSet answers = null; // A cursor.
```

```
    Double price = 0.0;
```

```
        queryText = " SELECT MIN(price) " + "FROM yrb_offer " + " WHERE title =  
? AND year = ? " + "AND club IN " +
```

```
                                                                " (SELECT club " + " FROM  
yrb_member " + " WHERE cid = ?)";
```

```
    // Prepare the query.
```

```
    try {
```

```
        querySt = conDB.prepareStatement(queryText);
```

```
    } catch (SQLException e) {
```

```
        System.out.println("SQL min_price failed in prepare");
```

```
        System.out.println(e.toString());
```

```
System.exit(0);
```

```
}//catch exception
```

```
// Execute the query.
```

```
try {
```

```
    querySt.setString(1, title);
```

```
    querySt.setInt(2, year);
```

```
    querySt.setInt(3, ID);
```

```
    answers = querySt.executeQuery();
```

```
} catch (SQLException e) {
```

```
    System.out.println("SQL min_price failed in execute");
```

```
    System.out.println(e.toString());
```

```
    System.exit(0);
```

```
}//catch exception
```

```
// Any answer?
```

```
try {
```

```
    if (answers.next()) {
```

```
        //inDB = true;
```

```
        price = answers.getDouble(1);
```

```

        } //else { inDB = false; }

    } catch (SQLException e) {

        System.out.println("SQL min_price failed in cursor.");

        System.out.println(e.toString());

        System.exit(0);

    } //catch exception

    // Close the cursor.

    try {

        answers.close();

    } catch (SQLException e) {

        System.out.print("SQL min_price failed closing cursor.\n");

        System.out.println(e.toString());

        System.exit(0);

    } //catch exception

    // We're done with the handle.

    try {

        querySt.close();

```



```

        } catch (SQLException e) {

            System.out.print("SQL min_price failed closing the handle.\n");

            System.out.println(e.toString());

            System.exit(0);

        } //catch exception

        return price;

    } //min_price

    /*
    *insert_purchase inserts the purchase information into the database
    */

    public void insert_purchase(int CID, String club, String title, int year, int quantity) {

        String queryText = ""; // The SQL text.

        PreparedStatement querySt = null; // The query handle.

        ResultSet answers = null; // A cursor

        Timestamp x = new Timestamp(System.currentTimeMillis());

        DateFormat y = new SimpleDateFormat("yyyy-MM-dd-HH.mm.ss");

        String time = y.format(x);

        queryText = "INSERT into yrb_purchase " + "values (?, ?, ?, ?, ?) ";

```

```
try {  
    querySt = conDB.prepareStatement(queryText);  
  
} catch (SQLException e) {  
    System.out.println("SQL insert_purchase failed in prepare");  
    System.out.println(e.toString());  
    System.exit(0);  
  
} //catch exception  
  
// Execute the query.  
try {  
    querySt.setInt(1, CID);  
    querySt.setString(2, club);  
    querySt.setString(3, title);  
    querySt.setInt(4, year);  
    querySt.setString(5, time);  
    querySt.setInt(6, quantity);  
    querySt.executeUpdate();  
  
} catch (SQLException e) {  
    System.out.println("SQL insert_purchase failed in update");  
    System.out.println(e.toString());
```

```

        System.exit(0);

    } // catch exception

    // We're done with the handle.
    try {
        querySt.close();

    } catch (SQLException e) {
        System.out.print("SQL insert_purchase failed closing the handle.\n");
        System.out.println(e.toString());
        System.exit(0);

    } // catch exception

} // insert_purchase

/*
 * get_club retrieves the club name given the associated year, customer ID, book title and
price
 */

private String get_club(Integer year, Integer customerID, String title, Double lowestPrice) {
    String queryText = ""; // The SQL text.

    PreparedStatement querySt = null; // The query handle.

```

```
ResultSet answers = null; // A cursor.
```

```
String club = "";
```

```
        queryText = "SELECT o.club " + " FROM yrb_member m, yrb_offer o" +  
WHERE o.club = m.club AND o.year = ? " +
```

```
        "AND m.cid = ? AND o.title = ? AND o.price = ? ";
```

```
// Prepare the query.
```

```
try {
```

```
        querySt = conDB.prepareStatement(queryText);
```

```
    } catch (SQLException e) {
```

```
        System.out.println("SQL get_club failed in prepare");
```

```
        System.out.println(e.toString());
```

```
        System.exit(0);
```

```
    } //catch exception
```

```
// Execute the query.
```

```
try {
```

```
        querySt.setInt(1, year);
```

```
        querySt.setInt(2, customerID);
```

```
        querySt.setString(3, title);
```

```

        querySt.setDouble(4, lowestPrice);

        answers = querySt.executeQuery();

    } catch (SQLException e) {

        System.out.println("SQL get_club failed in execute");

        System.out.println(e.toString());

        System.exit(0);

    } //catch exception

    // any answer?

    try {

        if (answers.next()) { club = answers.getString(1); }

    } catch (SQLException e) {

        System.out.println("SQL get_club failed in cursor.");

        System.out.println(e.toString());

        System.exit(0);

    } //catch exception

    // Close the cursor.

    try {

        answers.close();

```

```

    } catch (SQLException e) {

        System.out.print("SQL get_club failed closing cursor.\n");

        System.out.println(e.toString());

        System.exit(0);

    } //catch exception

    // We're done with the handle.
    try {

        querySt.close();

    } catch (SQLException e) {

        System.out.print("SQL get_club failed closing the handle.\n");

        System.out.println(e.toString());

        System.exit(0);

    } //catch exception

    return club;

} //get_club method

} //public class yrb

```