Projects II: Apps

EECS 3421, Fall 2017

Student: Zhehao Zheng

Student #: 212286522

Email: jack93@my.yorku.ca

Contents

Introduction

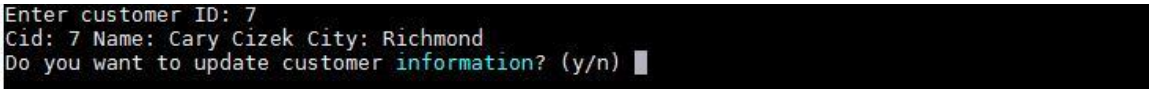
In this project, we are working with the existing YRB database from project I. We are tasked to write a java application program that complies SQL application programs and it will modify the YRB database. In this assignment we are tasked to find a customer if he exists in the database by given id and display the customer information and then update his information if the user desires, next the application will all the categories in the database and display all of them in a list, user can then decide which category to choose from, after choosing the category the user will enter a title of the book that he wants to purchase, if the entered book is in the data base, the book information will be shown, if the book does not exist in the said category, an error message will be displayed and let the users to choose from a category again, if the book exists in the category the program promotes the user to buy the book in a list and the minimum cost of the book will be shown. User will then enter the quantity that he desires to buy then the program will calculate the total cost using quantity * minimum price and show it to the user. If user approves the purchase will be made and the information of the purchase such as user id, book title, club name,

book year, current date and time and the quantity brought will be store in the purchase table.

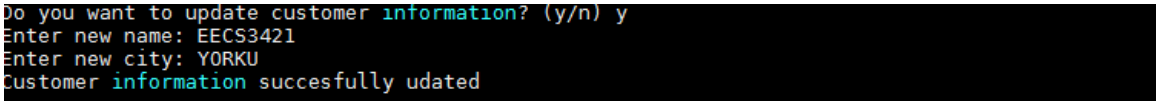
User Manual

This a straight forward program, first we can set up the program in the following way.

Then buy running the program it will as user to enter the customer id, and the customer information will be displayed.



Then, it will promote to ask user if he wants to update customer's information if user enters N or n it will show the category of the books, if user enters Y or y I will then asking to enter new information.



User information is updated in the database.

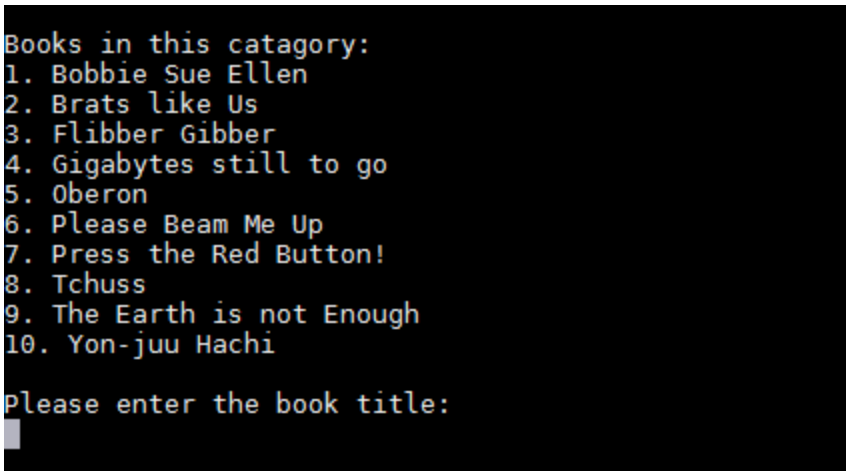
```
red 308 % db2 "select * from yrb_customer where cid = 7"


CID    NAME                 CITY
------ -------------------- ---------------
     7 EECS3421             YORKU

  1 record(s) selected.
```

```
Enter customer ID: 7
Cid: 7 Name: EECS3421 City: YORKU
Do you want to update customer information? (y/n)
```

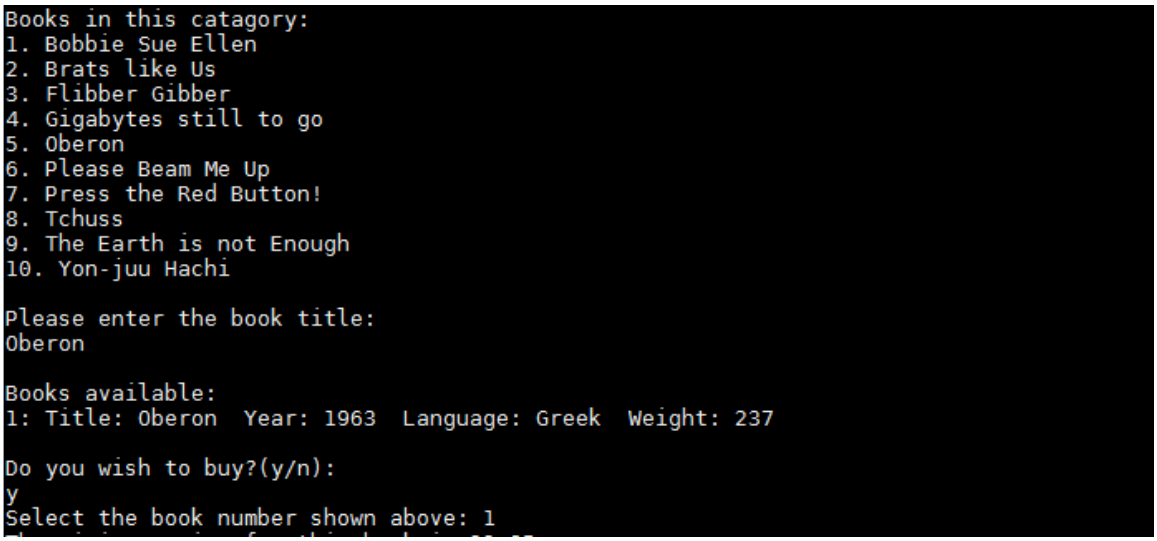
Then it will ask user to choose aa category from the list that user wish to purchase.

```
Please choose a category from the list:
1.children
2.cooking
3.drama
4.guide
5.history
6.horror
7.humor
8.mystery
9.phil
10.romance
11.science
12.travel
Enter your choice:
```

After you enter the desired choice and the books in that category shows up in the list, then customer can enter the title of the book customer wants to purchase.



After customer enters the book title it will show the list of book in that title and ask if customer wish to purchase that book, then customer can select one of the choices in the drop list.



After picking the book, the minimum price of the book is shown and program promote to ask the quantity of the books customer wants to purchase, and total cost is calculated by the minimum price * quantity,

and display to the customer and waiting for customer to approve the purchase.

```
Select the book number shown above: 1
The minimum price for this book is $22.95
How many would you like to purchase? 6
The total price for the purchase is: $137.7
Do you want to purchase the books? (y/n)
```

If customer picks y it will confirm the purchase, else it will abort the purchase.

```
Do you want to purchase the books? (y/n) y
Purchsed made!
```

```
Do you want to purchase the books? (y/n) n
No purchase was made!
```

After confirming the purchase info will be added to the purchase table.

```
red 324 % db2 "select * from yrb_purchase where cid = 7"


CID    CLUB            TITLE                     YEAR   WHEN                       QNTY
------ --------------- ------------------------- ------ -------------------------- ------
     7 Basic           Ringo to Nashi              1993 1999-06-15-12.13.00.000000      1
     7 Basic           Ringo to Nashi              1993 2000-05-05-14.49.00.000000      1
     7 YRB Gold        Cats are not Dogs           1992 2000-09-26-16.32.00.000000      1
     7 YRB Gold        Oberon                      1963 2017-12-05-18.25.54.000000      6
     7 YRB Gold        Richmond Underground        1997 1999-06-15-12.13.00.000000      1

  5 record(s) selected.
```

The CID, CLUB, TITLE, YEAR, WHEN (time) and QUANTITY is recorded on the purchase table.

Methods Used

In this assignment stored procedure is used and will be shown below.

1.

```
public boolean find_customer (int id){
```

find_customer is used to get a number as the customer id and returen

the customer information (cid, name, city).

Example:

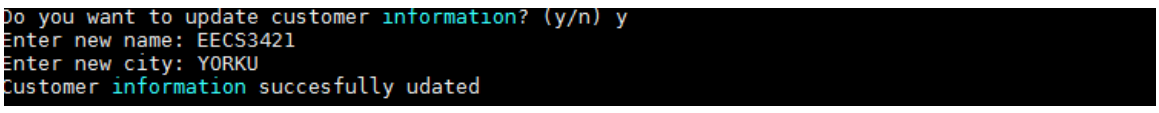
2.

```
public void update_customer(int cid, String name, String city){
```

update_customer is used to update customer information with the new

input name and city.

Example :



```
red 308 % db2 "select * from yrb_customer where cid = 7"


CID    NAME                 CITY
------ -------------------- ---------------
     7 EECS3421             YORKU

  1 record(s) selected.
```

3.

```
public void fetch_categories() {
```

fetch_categories is used to retrive all the category names in the database.

Example:

```
Please choose a category from the list:
1.children
2.cooking
3.drama
4.guide
5.history
6.horror
7.humor
8.mystery
9.phil
10.romance
11.science
12.travel
```

4.

```
public void displayBooks(String cat)
```

displayBooks is used to display book in the input category and show the books in that category, easier for customer to choose their book.

Example

```
Books in this catagory:
1. Bobbie Sue Ellen
2. Brats like Us
3. Flibber Gibber
4. Gigabytes still to go
5. Oberon
6. Please Beam Me Up
7. Press the Red Button!
8. Tchuss
9. The Earth is not Enough
10. Yon-juu Hachi
```

5.

```
public boolean find_book(String title, String cat)
{
```

find_book is use to return the title, year , language and wright if the book

exists. If not return false.

```
Books available:
1: Title: Oberon  Year: 1963  Language: Greek  Weight: 237
```

```
Please enter the book title:
EECS3421

The book does not exist, pick the category again!
```

6.

```
public double min_price(int cid, String title, String year)
{
```

min_price is used to returns the minimum price for the book.

Example:

```
The minimum price for this book is $22.95
```

7.

```
private void insert_purchase(int amount, String title, int year)
{
```

insert_purchase is used to insert the new purchase information and adds

a new tuple(s) for the new purchase in the yrb_purchase table.

Example:

```
Do you want to purchase the books? (y/n) y
Purchsed made!
```

```
red 324 % db2 "select * from yrb_purchase where cid = 7"


CID    CLUB           TITLE                    YEAR   WHEN                         QNTY
------ -------------- ------------------------ ------ -------------------------- ------
     7 Basic          Ringo to Nashi             1993 1999-06-15-12.13.00.000000      1
     7 Basic          Ringo to Nashi             1993 2000-05-05-14.49.00.000000      1
     7 YRB Gold       Cats are not Dogs          1992 2000-09-26-16.32.00.000000      1
     7 YRB Gold       Oberon                     1963 2017-12-05-18.25.54.000000      6
     7 YRB Gold       Richmond Underground       1997 1999-06-15-12.13.00.000000      1

  5 record(s) selected.
```

Tools Used

In this assignment we used Java via JDBC on CSE's PRIMSM environment to acces DB2. The program is talking with the database server using the script % source ~db2leduc/cshrc.runtime to do it at runtime.

By the connecting to the server, the app establishes the appropriate driver with the DriverManager to set up BD connection and app will be talking via the JDBC API.

```
// Set up the DB connection.
try {
    // Register the driver with DriverManager.
    Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();
} catch (ClassNotFoundException e) {
```

And then establish connection with the PRISM's BD2 system using the url for the course.

```
try {
    // Connect with a fall-thru id & password
    conDB = DriverManager.getConnection(url);

// URL: which database!
url = "jdbc:db2:c3421a";
```

After the application terminates, it closes the connection.

```
try {
    conDB.close();
} catch(SQLException e) {
    System.out.print("\nFailed trying to close the connection.\n");
    e.printStackTrace();
    System.exit(0);
}
```

In java queries are presented in queryText and get send to the database server and get executed. First we can prepare the query setup in java.

```
String queryText = ""; // The SQL text.
PreparedStatement querySt = null; // The query handle.
ResultSet answers = null; // A cursor.
```

Then we write the query in the queryText for the database to execute

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

Then we prepare the query for the BD server

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

The query is executed in

```
// Execute the query.
try {
    querySt.setInt(1, id);
    answers = querySt.executeQuery();
```

Then we get the answer from the database if there is any

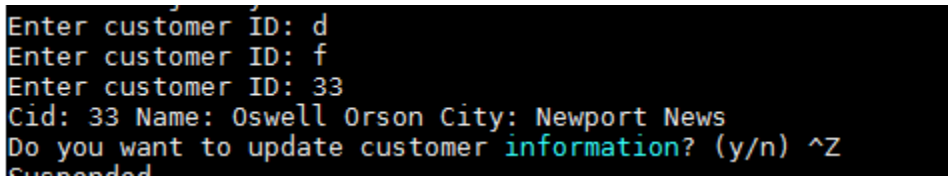
```
// Any answer?
try {
    if (answers.next()) {
        exist = true;
        custID = answers.getInt(1);
        System.out.println("Cid: " + answers.getInt(1) + " Name: " + answers.getString(2) + " City
    } else {
```

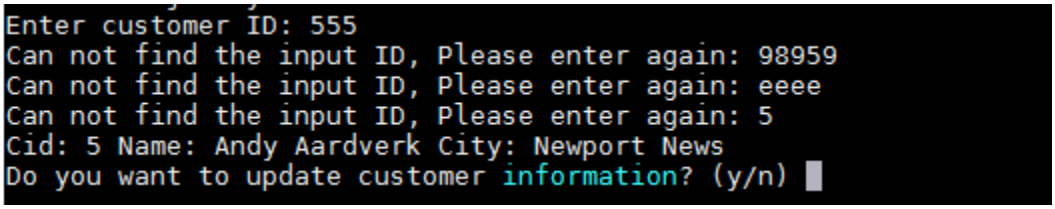
Lastly we close the cursor and the query handle

```
    // Close the cursor.
    try {
        answers.close();
    } catch(SQLException e) {
        System.out.print("SQL#1 failed closing cursor.\n");
        System.out.println(e.toString());
        System.exit(0);
    }

    // We're done with the handle.
    try {
        querySt.close();
    } catch(SQLException e) {
        System.out.print("SQL#1 failed closing the handle.\n");
        System.out.println(e.toString());
        System.exit(0);
    }
```

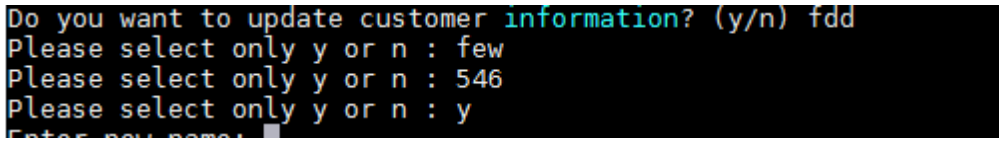
Input scenarios

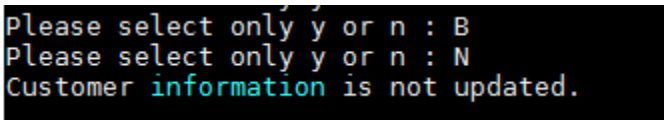
The program can handle different situations depending to the user input.





The user can only enter the customer id in the database.

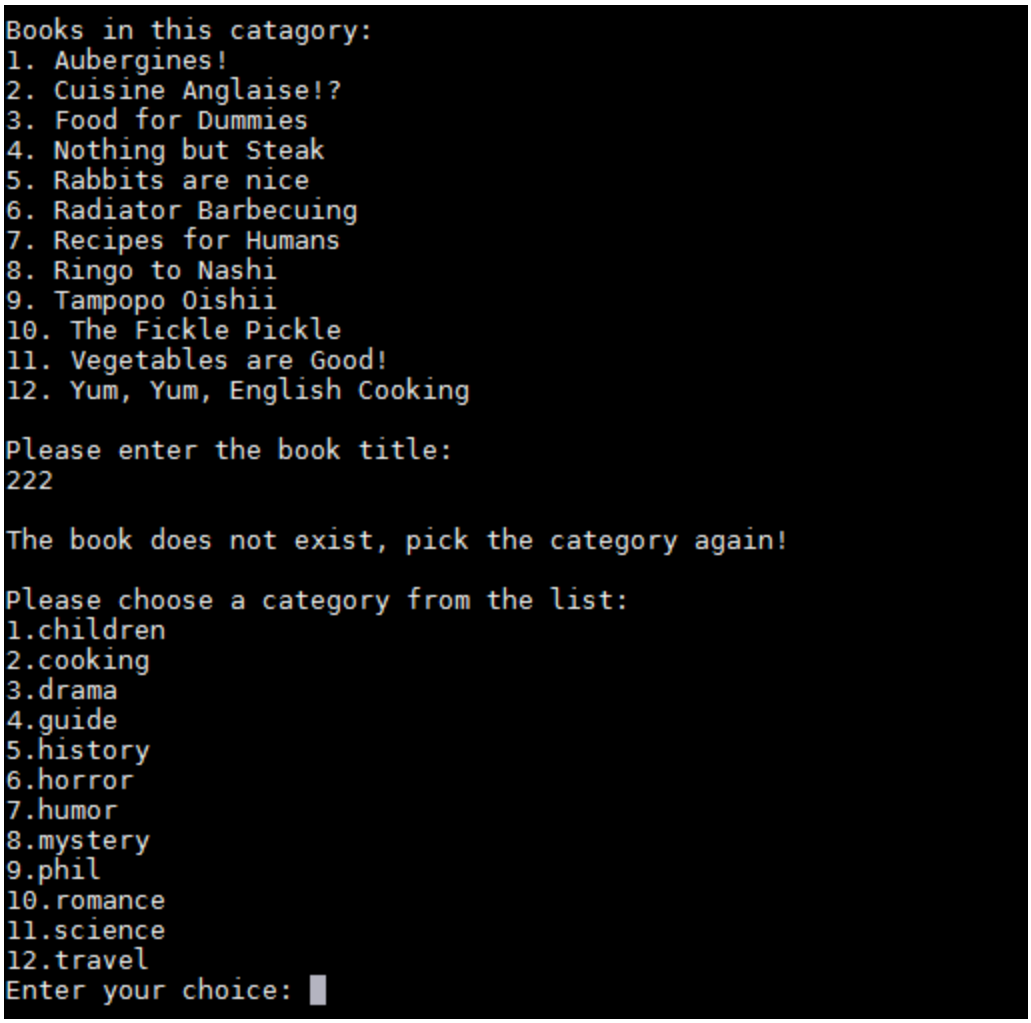




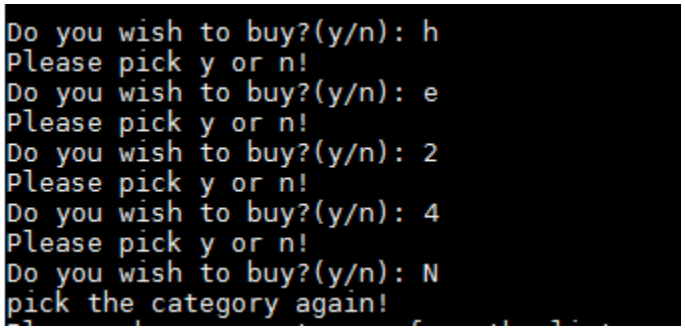
The user can only enter y/Y or n/N.

```
Please choose a category from the list:
1.children
2.cooking
3.drama
4.guide
5.history
6.horror
7.humor
8.mystery
9.phil
10.romance
11.science
12.travel
Enter your choice: dd
Enter your choice: ww
Enter your choice: fwfw
Enter your choice: 789
Please choose a category from the list: dd
Please choose a category from the list: 888
Please choose a category from the list: 2

Books in this catagory:
1. Aubergines!
2. Cuisine Anglaise!?
3. Food for Dummies
4. Nothing but Steak
5. Rabbits are nice
6. Radiator Barbecuing
7. Recipes for Humans
8. Ringo to Nashi
9. Tampopo Oishii
10. The Fickle Pickle
11. Vegetables are Good!
12. Yum, Yum, English Cooking

Please enter the book title:
```

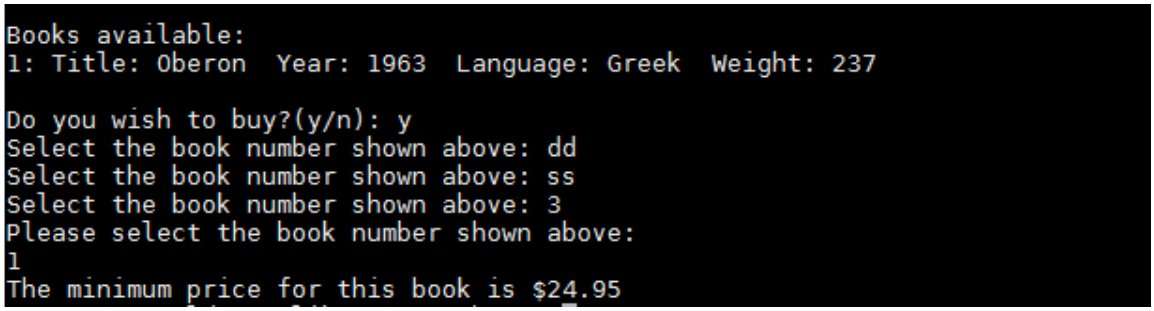
The program only accepts input on the list.



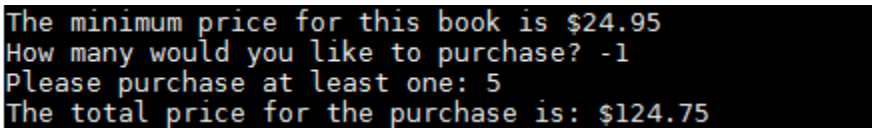
The program let the user to choose the category again if the entered title does not exist.



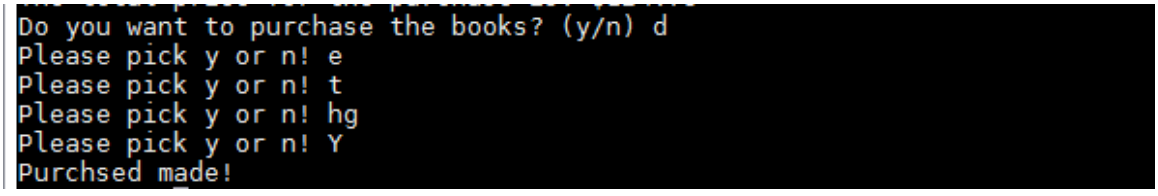
Program asks the user if he wants to buy the displayed book. Only y/Y or n/N input would be accepted.



Only accept the shown input.



Only accept positive numbers.



Only accept Y/y or N/n as input.

Source Code

```
import java.util.*;
import java.net.*;
import java.text.*;
import java.lang.*;
import java.io.*;
import java.sql.*;


public class yrbStore {

    private Connection conDB;   // Connection to the database system.
    private String url;         // URL: Which database?
    private Integer custID;     // Who are we tallying?
    private String club;
    private HashMap<Integer, String[]> bookMap = new HashMap<Integer,
String[]>();
    private int bookNum;

    public yrbStore() {

        // Set up the DB connection.
        try {
            // Register the driver with DriverManager.
            Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            System.exit(0);
        } catch (InstantiationException e) {
            e.printStackTrace();
            System.exit(0);
        } catch (IllegalAccessException e) {
            e.printStackTrace();
            System.exit(0);
        }

        // URL: Which database?
        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);
        }

        // Commit.  Okay, here nothing to commit really, but why not...
        try {
            conDB.commit();
```

```
        } catch(SQLException e) {
            System.out.print("\nFailed trying to commit.\n");
            e.printStackTrace();
            System.exit(0);
        }


        System.out.print("Enter customer ID: ");
        Scanner input = new Scanner(System.in);
        while(!input.hasNextInt()) {
            input.next();
            System.out.print("Enter customer ID: ");
        }
        int custID = input.nextInt();

        boolean exi = false;
        while(!exi){
            if(find_customer(custID)){
                exi = true;
            }
            else{
                System.out.print("Can not find the input ID, Please
enter again: ");
                while(!input.hasNextInt()) {
                    input.next();
                    System.out.print("Can not find the input ID, Please
enter again: ");
                }
                custID = input.nextInt();
                exi = false;
            }
        }

        System.out.print("Do you want to update customer information?
(y/n) ");
        Scanner input1 = new Scanner(System.in);
        String update = input1.nextLine();

        boolean checkupdate = false;

        while(!checkupdate){
            if(update.equals("Y") || update.equals("y")){
                System.out.print("Enter new name: ");
                Scanner input2 = new Scanner(System.in);
                String updatename = input2.nextLine();
                System.out.print("Enter new city: ");
                Scanner input3 = new Scanner(System.in);
                String updatecity = input3.nextLine();
                update_customer(custID, updatename, updatecity);
                System.out.println("Customer information succesfully
udated");
                System.out.println(" ");
                checkupdate = true;
            }
            else if(update.equals("N") || update.equals("n")){
                System.out.println("Customer information is not
updated.");
```

```
                System.out.println(" ");
                break;

            }
            else {
                System.out.print("Please select only y or n : ");
                update = input1.nextLine();
                continue;
            }
        }


        boolean reverse = false;

        while(!reverse)
        {


        System.out.println("Please choose a category from the list: ");
        fetch_categories();
        System.out.print("Enter your choice: ");
        Scanner input4 = new Scanner(System.in);
        while(!input4.hasNextInt()) {
            input4.next();
            System.out.print("Enter your choice: ");
        }
        int category = input4.nextInt();
        while(category <=0 || category > 12){
            System.out.print("Please choose a category from the list:
");
            while(!input4.hasNextInt()) {
                input4.next();
                System.out.print("Please choose a category from the
list: ");
            }
            category = input4.nextInt();
        }

        String catEnter = "";
        if(category == 1){
            catEnter = "children";
        }
        else if(category == 2){
            catEnter = "cooking";
        }
        else if(category == 3){
            catEnter = "drama";
        }
        else if(category == 4){
            catEnter = "guide";
        }
        else if(category == 5){
            catEnter = "history";
        }
        else if(category == 6){
            catEnter = "horror";
        }
```

```
        else if(category == 7){
            catEnter = "humor";
        }
        else if(category == 8){
            catEnter = "mystery";
        }
        else if(category == 9){
            catEnter = "phil";
        }
        else if(category == 10){
            catEnter = "romance";
        }
        else if(category == 11){
            catEnter = "science";
        }
        else if(category == 12){
            catEnter = "travel";
        }

        System.out.println("");
        System.out.println("Books in this catagory: ");
        displayBooks(catEnter);
        System.out.println("");
        System.out.println("Please enter the book title: ");
        Scanner input5 = new Scanner(System.in);
        String booklist = input5.nextLine();

        boolean xxx = find_book(booklist,catEnter);

        if(xxx){

            boolean wish = false;
            while(!wish){
            System.out.println(" ");
            System.out.print("Do you wish to buy?(y/n): ");
            Scanner input6 = new Scanner(System.in);
            String buybuy = input6.nextLine();
            if(buybuy.equals("Y") || buybuy.equals("y")){
            reverse = true;
            wish = true;
            }
            else if(buybuy.equals("N") || buybuy.equals("n")){
                System.out.println("pick the category again!");
                reverse = false;
                break;
            }
            else{
                System.out.print("Please pick y or n!");
                continue;
            }
            }
        }
        else{
            System.out.println(" ");
            System.out.println("The book does not exist, pick the
category again!");
            System.out.println(" ");
```

```
            reverse = false;
        }

        }


        System.out.print("Select the book number shown above: ");
        Scanner input333 = new Scanner(System.in);
        while(!input333.hasNextInt()) {
            input333.next();
            System.out.print("Select the book number shown above: ");
        }
        int Selected = input333.nextInt();
        int sNumber = 0;
        boolean choice1 = false;
        while(!choice1){
            if(Selected <=0 || Selected > bookNum){
                choice1 = false;
                System.out.print("Please select the book number shown
above: ");
                Selected = input333.nextInt();
            }
            else{
                choice1 = true;
                sNumber = Selected;

            }
        }

        String[] storevalue = bookMap.get(sNumber);
        String title = storevalue[0];
        String year = storevalue[1];
        double minPrice = min_price(custID, title, year);
        System.out.println("The minimum price for this book is $" +
minPrice);
        System.out.print("How many would you like to purchase? ");
        Scanner input444 = new Scanner(System.in);
        while(!input444.hasNextInt()) {
            input444.next();
            System.out.print("How many would you like to purchase? ");
        }
        int purchase = input444.nextInt();

        boolean purchasecheck = false;
        while(!purchasecheck){
            if(purchase <=0){
                System.out.print("Please purchase at least one: ");
                while(!input444.hasNextInt()) {
                    input444.next();
                    System.out.print("Please purchase at least one: ");
                }
                purchase = input444.nextInt();
                purchasecheck = false;
            }
            else{
                purchasecheck = true;
            }
```

```
        }

        club = getClub(title, year, minPrice);
        double sum;
        DecimalFormat df = new DecimalFormat("#.##");
        System.out.println("The total price for the purchase is: $" +
df.format((sum = minPrice * purchase)));
        System.out.print("Do you want to purchase the books? (y/n) ");
        Scanner inputgt = new Scanner(System.in);
        String purchaseC = inputgt.nextLine();

        boolean lastpurchase = false;
        while(!lastpurchase){
            if(purchaseC.equals("Y") || purchaseC.equals("y")){
                insert_purchase(purchase, title,
Integer.parseInt(year));
                System.out.println("Purchsed made!");
                lastpurchase = true;
            }else if(purchaseC.equals("N") || purchaseC.equals("n")){
                System.out.println("No purchase was made!");
                break;
            }
            else{
                System.out.print("Please pick y or n! ");
                purchaseC = inputgt.nextLine();
                continue;
            }

        }

        // Close the connection.
        try {
            conDB.close();
        } catch(SQLException e) {
            System.out.print("\nFailed trying to close the
connection.\n");
            e.printStackTrace();
            System.exit(0);
        }




    }


    public boolean find_customer (int id){
        String queryText = ""; // The SQL text.
        PreparedStatement querySt = null; // The query handle.
        ResultSet answers = null; // A cursor.

        boolean exist = false;

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

        // Prepare the query.
```

```
        try {
            querySt = conDB.prepareStatement(queryText);
        } catch(SQLException e) {
            System.out.println("SQL#1 failed in prepare");
            System.out.println(e.toString());
            System.exit(0);
        }

        // Execute the query.
        try {
            querySt.setInt(1, id);
            answers = querySt.executeQuery();
        } catch(SQLException e) {
            System.out.println("SQL#1 failed in execute");
            System.out.println(e.toString());
            System.exit(0);
        }

        // Any answer?
        try {
            if (answers.next()) {
                exist = true;
                custID = answers.getInt(1);
                System.out.println("Cid: " + answers.getInt(1) + " Name:
" + answers.getString(2) + " City: " + answers.getString(3) );
            } else {
                exist = false;
                custID = null;
            }
        } catch(SQLException e) {
            System.out.println("SQL#1 failed in cursor.");
            System.out.println(e.toString());
            System.exit(0);
        }

        // Close the cursor.
        try {
            answers.close();
        } catch(SQLException e) {
            System.out.print("SQL#1 failed closing cursor.\n");
            System.out.println(e.toString());
            System.exit(0);
        }

        // We're done with the handle.
        try {
            querySt.close();
        } catch(SQLException e) {
            System.out.print("SQL#1 failed closing the handle.\n");
            System.out.println(e.toString());
            System.exit(0);
        }

        return exist;


    }
```

```
    public void update_customer(int cid, String name, String city){

        String queryText = ""; // The SQL text.
        PreparedStatement querySt = null; // The query handle.

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

        try {
            querySt = conDB.prepareStatement(queryText);
        } catch(SQLException e) {
            System.out.println("SQL#1 failed in prepare");
            System.out.println(e.toString());
            System.exit(0);
        }

        // Execute the query.
        try {
            querySt.setString(1, name);
            querySt.setString(2, city);
            querySt.setInt(3, cid);

            querySt.executeUpdate();
        } catch(SQLException e) {
            System.out.println("SQL#1 failed in Update");
            System.out.println(e.toString());
            System.exit(0);
        }

        // We're done with the handle.
        try {
            querySt.close();
        } catch (SQLException e) {
            System.out.print("SQL#1 failed closing the handle.\n");
            System.out.println(e.toString());
            System.exit(0);
        }

    }

    public void fetch_categories() {
        String queryText = ""; // The SQL text.
        PreparedStatement querySt = null; // The query handle.
        ResultSet answers = null; // A cursor.

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

        // Prepare the query.
        try {
            querySt = conDB.prepareStatement(queryText);
        } catch (SQLException e) {
            System.out.println("SQL#1 failed in prepare");
            System.out.println(e.toString());
            System.exit(0);
        }
```

```
        // Execute the query.
        try {
            answers = querySt.executeQuery();
        } catch (SQLException e) {
            System.out.println("SQL#1 failed in execute");
            System.out.println(e.toString());
            System.exit(0);
        }

        // Any answer?
        try {
            for (int i = 1; answers.next(); i++) {
                String list = answers.getString(1);
                System.out.println(i + "." + list);
            }
        } catch (SQLException e) {
            System.out.println("SQL#1 failed in cursor.");
            System.out.println(e.toString());
            System.exit(0);
        }

        // Close the cursor.
        try {
            answers.close();
        } catch (SQLException e) {
            System.out.print("SQL#1 failed closing cursor.\n");
            System.out.println(e.toString());
            System.exit(0);
        }

        // We're done with the handle.
        try {
            querySt.close();
        } catch (SQLException e) {
            System.out.print("SQL#1 failed closing the handle.\n");
            System.out.println(e.toString());
            System.exit(0);
        }
    }


    public void displayBooks(String cat)
    {
        String queryText = ""; // The SQL text.
        PreparedStatement querySt = null; // The query handle.
        ResultSet answers = null; // A cursor.

        queryText = "select distinct title from yrb_book where cat = ?";



        // Prepare the query.
        try {
            querySt = conDB.prepareStatement(queryText);
        } catch (SQLException e) {
            System.out.println("SQL#1 failed in prepare");
            System.out.println(e.toString());
```

```
            System.exit(0);
        }

        // Execute the query.
        try {
            querySt.setString(1, cat);
            answers = querySt.executeQuery();
        } catch (SQLException e) {
            System.out.println("SQL#1 failed in execute");
            System.out.println(e.toString());
            System.exit(0);
        }

        // Any answer?
        try {
            boolean exist = answers.next();
            int bookNum1 = 1;
            String bookTitle;
            for (; exist; bookNum1++)
            {
                bookTitle = answers.getString(1);
                System.out.println(bookNum1 + ". " + bookTitle);
                if(answers.next() == true)
                {
                    exist = true;
                }
                else
                {
                    exist = false;
                }
            }
        } catch (SQLException e) {
            System.out.println("SQL#1 failed in cursor.");
            System.out.println(e.toString());
            System.exit(0);
        }

        // Close the cursor.
        try {
            answers.close();
        } catch (SQLException e) {
            System.out.print("SQL#1 failed closing cursor.\n");
            System.out.println(e.toString());
            System.exit(0);
        }

        // We're done with the handle.
        try {
            querySt.close();
        } catch (SQLException e) {
            System.out.print("SQL#1 failed closing the handle.\n");
            System.out.println(e.toString());
            System.exit(0);
        }

    }
```

```
    public boolean find_book(String title, String cat)
    {
        String queryText = ""; // The SQL text.
        PreparedStatement querySt = null; // The query handle.
        ResultSet answers = null; // A cursor.

        boolean inDB = false; // Return.

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


        String[] findBook = new String[4];

        // Prepare the query.
        try {
            querySt = conDB.prepareStatement(queryText);
        } catch (SQLException e) {
            System.out.println("SQL#1 failed in prepare");
            System.out.println(e.toString());
            System.exit(0);
        }

        // Execute the query.
        try {
            querySt.setString(1, title);
            querySt.setString(2, cat);
            answers = querySt.executeQuery();
        } catch (SQLException e) {
            System.out.println("SQL#1 failed in execute");
            System.out.println(e.toString());
            System.exit(0);
        }

        // Any answer?
        try {
            if(!answers.next())
            {
                inDB = false;
            }
            else
            {
                System.out.println("\nBooks available:");
                inDB = true;
                boolean nExist = true;
                int pos = 1;
                for (; nExist; pos++)
                {
                    findBook = new String[4];
                    findBook[0] = answers.getString(1);
                    findBook[1] = String.valueOf(answers.getInt(2));
                    findBook[2] = answers.getString(3);
                    findBook[3] = String.valueOf(answers.getInt(5));
                    bookMap.put(pos, findBook);
```

```
                    System.out.println(pos + ": " + "Title: " +
answers.getString(1) + "  Year: " + answers.getInt(2) + "  Language: "
+ answers.getString(3) + "  Weight: " + answers.getInt(5));
                    if(answers.next() == true)
                    {
                        nExist = true;
                    }
                    else
                    {
                        nExist = false;
                    }
                }

                bookNum = pos - 1;
            }


        } catch (SQLException e) {
            System.out.println("SQL#1 failed in cursor.");
            System.out.println(e.toString());
            System.exit(0);
        }

        // Close the cursor.
        try {
            answers.close();
        } catch (SQLException e) {
            System.out.print("SQL#1 failed closing cursor.\n");
            System.out.println(e.toString());
            System.exit(0);
        }

        // We're done with the handle.
        try {
            querySt.close();
        } catch (SQLException e) {
            System.out.print("SQL#1 failed closing the handle.\n");
            System.out.println(e.toString());
            System.exit(0);
        }

        return inDB;

    }


    public double min_price(int cid, String title, String year)
    {
        String queryText = ""; // The SQL text.
        PreparedStatement querySt = null; // The query handle.
        ResultSet answers = null; // A cursor.
        double price = 0.0;

        queryText = "SELECT distinct min(price) "
                + "FROM yrb_offer WHERE year = ? and title = ? 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#1 failed in prepare");
    System.out.println(e.toString());
    System.exit(0);
}

// Execute the query.
try {
    querySt.setInt(1, Integer.parseInt(year));
    querySt.setString(2, title);
    querySt.setInt(3, custID);
    answers = querySt.executeQuery();
} catch (SQLException e) {
    System.out.println("SQL#1 failed in execute");
    System.out.println(e.toString());
    System.exit(0);
}

// Any answer?
try {
    if(answers.next())
    {
        price = answers.getDouble(1);
    }
    else
    {
        System.out.println("Can't find the minimum.");
    }
} catch (SQLException e) {
    System.out.println("SQL#1 failed in cursor.");
    System.out.println(e.toString());
    System.exit(0);
}

// Close the cursor.
try {
    answers.close();
} catch (SQLException e) {
    System.out.print("SQL#1 failed closing cursor.\n");
    System.out.println(e.toString());
    System.exit(0);
}

// We're done with the handle.
try {
    querySt.close();
} catch (SQLException e) {
    System.out.print("SQL#1 failed closing the handle.\n");
    System.out.println(e.toString());
    System.exit(0);
}
```

```
        return price;
    }


    private void insert_purchase(int amount, String title, int year)
    {
        String queryText = ""; // The SQL text.
        PreparedStatement querySt = null; // The query handle.

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

        DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-
HH.mm.ss");
        String todaytime = dateFormat.format(timestamp);

        queryText = "INSERT INTO yrb_purchase VALUES (?,?,?,?,?,?)";



        // Prepare the query.
        try {
            querySt = conDB.prepareStatement(queryText);
        } catch (SQLException e) {
            System.out.println("SQL#1 failed in prepare");
            System.out.println(e.toString());
            System.exit(0);
        }

        // Execute the query.
        try {

            querySt.setInt(1, custID);
            querySt.setString(2, club);
            querySt.setString(3, title);
            querySt.setInt(4, year);
            querySt.setString(5, todaytime);
            querySt.setInt(6, amount);

            querySt.executeUpdate();
        } catch (SQLException e) {
            System.out.println("SQL#1 failed in update");
            System.out.println(e.toString());
            System.exit(0);
        }

        // We're done with the handle.
        try {
            querySt.close();
        } catch (SQLException e) {
            System.out.print("SQL#1 failed closing the handle.\n");
            System.out.println(e.toString());
            System.exit(0);
        }
```

```
}

private String getClub(String title, String year, double price)
{

    String club = "";

    String queryText = ""; // The SQL text.
    PreparedStatement querySt = null; // The query handle.
    ResultSet answers = null; // A cursor.

    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#1 failed in prepare");
        System.out.println(e.toString());
        System.exit(0);
    }

    // Execute the query.
    try {

        querySt.setInt(1, Integer.parseInt(year));
        querySt.setInt(2, custID);
        querySt.setString(3, title);
        querySt.setDouble(4, price);
        answers = querySt.executeQuery();
    } catch (SQLException e) {
        System.out.println("SQL#1 failed in execute");
        System.out.println(e.toString());
        System.exit(0);
    }

    // Any answer?
    try {
        if(answers.next())
        {
            club = answers.getString(1);
        }
        else
        {
            System.out.println("No club found");
        }
    } catch (SQLException e) {
        System.out.println("SQL#1 failed in cursor.");
        System.out.println(e.toString());
        System.exit(0);
```

```
        }

        // Close the cursor.
        try {
            answers.close();
        } catch (SQLException e) {
            System.out.print("SQL#1 failed closing cursor.\n");
            System.out.println(e.toString());
            System.exit(0);
        }

        // We're done with the handle.
        try {
            querySt.close();
        } catch (SQLException e) {
            System.out.print("SQL#1 failed closing the handle.\n");
            System.out.println(e.toString());
            System.exit(0);
        }


        return club;
    }



    public static void main(String[] args) {
        // TODO Auto-generated method stub
        yrbStore ct = new yrbStore();
    }

}
```