

APPs Project

EECS 3421 – Introduction to Database Management Sysytems.

Rajkumar Balakrishnan Lakshmi - 213141197

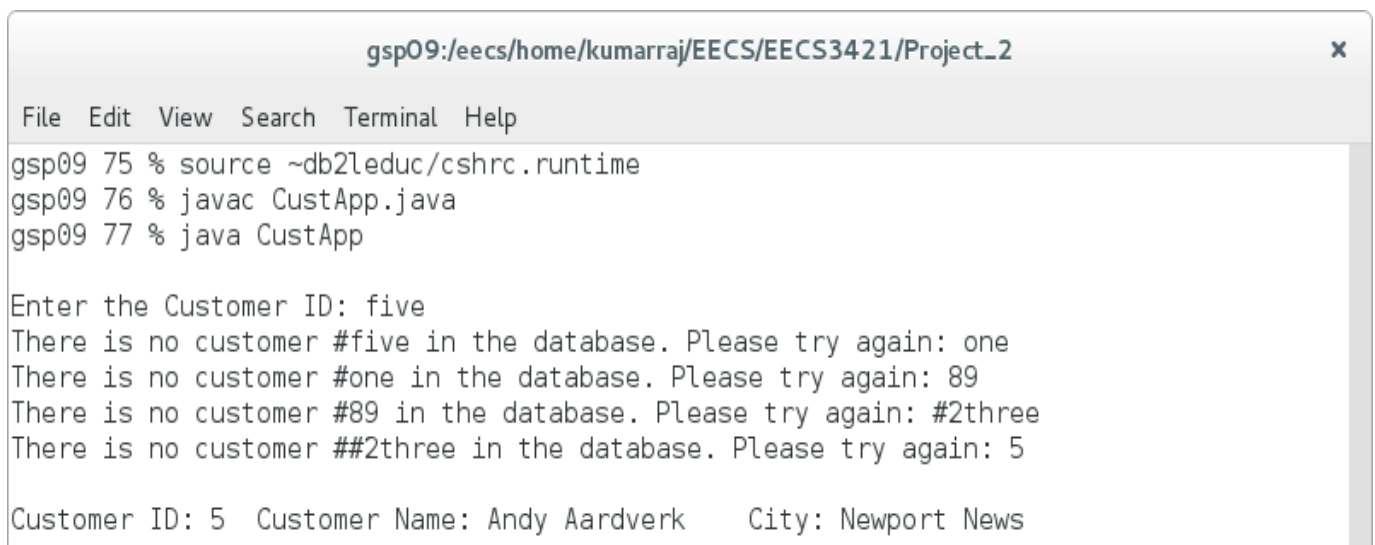
5/12/17

Report

The aim of this project is to create an application program for the YRB database and the main purpose of this application is to search through the database and try to purchase books that are available on the database.

The program begins by asking the user to input the Customer id and if the entered id exists in the database it displays the details of that customer. Whereas if the entered id is not available on the database or if the user inputs some invalid type values or characters, then it prints an error message and prompts the user to input the customer id again. This process keeps looping through until a valid customer id is entered by the user. A glimpse of this process is shown below:

Input example for – Enter the customer ID: Integer value (1 to n) n- number of customers in the database.



```
gsp09:/eecs/home/kumarraj/EECS/EECS3421/Project_2
File Edit View Search Terminal Help
gsp09 75 % source ~db2leduc/cshrc.runtime
gsp09 76 % javac CustApp.java
gsp09 77 % java CustApp

Enter the Customer ID: five
There is no customer #five in the database. Please try again: one
There is no customer #one in the database. Please try again: 89
There is no customer #89 in the database. Please try again: #2three
There is no customer ##2three in the database. Please try again: 5

Customer ID: 5 Customer Name: Andy Aardverk City: Newport News
```

Now it asks if the user wants to update the information of the displayed customer and if the customer agrees, all possible ways of updating the information is displayed and asks for the user to choose from one of the options. Here again if the user inputs an invalid type of characters or value. It prints an error message and prompts the user to input the option again. This process keeps looping through until a valid option is chosen and based on the option chosen the update is performed. A glimpse of the above process is shown below:

Input example for – Updating Customer Information: String (Name | City | Both)



```

gsp09:/eecs/home/kumarraj/EECS/EECS3421/Project_2
File Edit View Search Terminal Help
gsp09 79 % java CustApp
Enter the Customer ID: 5
Customer ID: 5   Customer Name: Andy Aardverk   City: Newport News
Would you like to update the customer information? Enter Yes || No : s
The entered option is invalid. Please try again: yup
The entered option is invalid. Please try again: #01
The entered option is invalid. Please try again: yes
Possible ways of updating customer information are:
1.Name
2.City
3.Both
Choose from one of the above options:nmaecity
The entered option is invalid. Please try again: name&city
The entered option is invalid. Please try again: 3
The entered option is invalid. Please try again: both
Enter the new name: Jarek Gryz
Enter the new city: Toronto
Name has been successfully updated to Jarek Gryz !!!
City has been successfully updated to Toronto !!!

```

After updating the customer information, the program displays a list of all categories of books available on the database and asks the user to input or to choose a category from the list to see all the titles available under that category. If a right category is chosen from one of the categories in the list, then the book titles associated with that title is displayed, otherwise if an invalid type or any other invalid option is entered by the user, the program prints an error message and asks the user to input the category again. This process keeps looping through until a valid option is entered.

Now if the user chooses the right category and all the book titles associated with it is displayed, then the program asks for the user to enter a book title from the list of books displayed to see the details of that book. Here if the user enters a book that is not in the list or if the user enters any invalid characters or values, then the program displays an error message and takes back the user to the previous step of choosing the categories again. We assume that the user is looking for a book in a different category, so we go back to the previous step. This routine continues until the user chooses a right book title associated to the right category. A glimpse of the above two processes is shown below:

Input example for – Choosing Category: String (children | cooking | | travel)

Input example for – Choosing Book title: String (Database aren't | ... | Plato Rocks)

Note: For the above don't enter a integer value, the input should be the category name or title name only.

```

gsp09:/eecs/home/kumarraj/EECS/EECS3421/Project_2
File Edit View Search Terminal Help
The categories of books in our database are:
1.children
2.cooking
3.drama
4.guide
5.history
6.horror
7.humor
8.mystery
9.phil
10.romance
11.science
12.travel
Choose from one of the above categories to see the book titles: mystery

The book titles under the category "mystery" are:
1.Rigor Mortis
2.The Wiley Snark
3.Under Your Bed
4.Where are my Socks?
5.Will Snoopy find Lucy?
Choose from one of the above book titles to view its details: Plato Rocks

The book "Plato Rocks" under the category "mystery" doesn't exist. Please try again!

-----
The categories of books in our database are:
1.children
2.cooking
3.drama
4.guide
5.history
6.horror
7.humor
8.mystery
9.phil
10.romance
11.science
12.travel
Choose from one of the above categories to see the book titles: phil

The book titles under the category "phil" are:
1.Databases aren't
2.Is Math is fun?
3.Plato Rocks
Choose from one of the above book titles to view its details: Plato Rocks

-----
-----

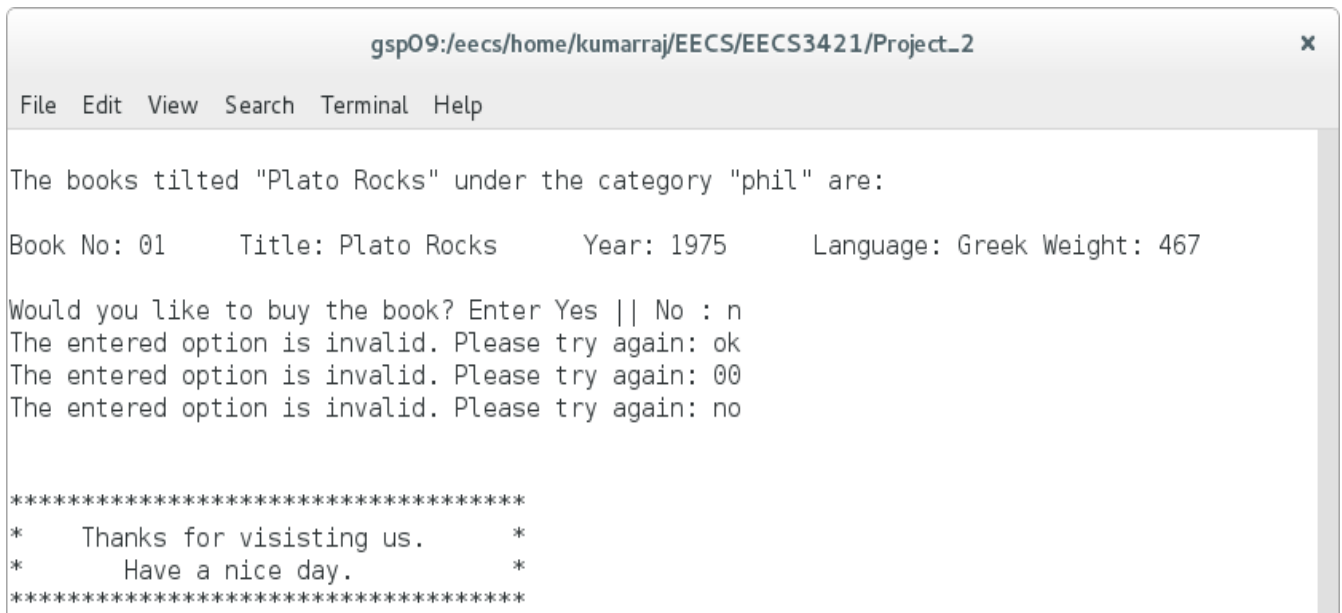
The books titled "Plato Rocks" under the category "phil" are:

Book No: 01      Title: Plato Rocks      Year: 1975      Language: Greek Weight: 467

```

If the entered book title of the chosen category exists, then we display all the details of the book. If there exists a book with same name of the same category but different year of publication or language, then all such books are displayed for the user to choose from. Each such books are displayed with a book number beside the details, making it easier for the user to choose from the books. Now we ask if the user is interested in purchasing the book with yes/no option. If the user enters no, we simply print a thanking message and we exit the program. If the user enters anything other the options provided, the program keeps prompting until a right option is entered as seen in above cases. A glimpse of the above process is shown below:

Input example for – Buying the book: String (Yes | No)



```

gsp09:/eecs/home/kumarraj/EECS/EECS3421/Project_2
File Edit View Search Terminal Help

The books tilted "Plato Rocks" under the category "phil" are:

Book No: 01      Title: Plato Rocks      Year: 1975      Language: Greek Weight: 467

Would you like to buy the book? Enter Yes || No : n
The entered option is invalid. Please try again: ok
The entered option is invalid. Please try again: 00
The entered option is invalid. Please try again: no

*****
*   Thanks for visisting us.   *
*   Have a nice day.         *
*****

```

If the user enters “yes”, then we ask the user to input the book number that he/she wishes to buy. If an invalid value or type is entered, an error message is displayed, and we continue to prompt until a valid number is entered. Then we display the minimum price for the chosen book number and ask the user to input the quantity of the book to be bought. Here again, if an invalid value or type is entered, an error message is displayed, and we continue to prompt until a valid number is entered. After that being done, we update the yrb_purchase table with the new transaction details and display the user with the conformation message including the total price

(Quantity * minimum price) and exits the program with a thanking message. A glimpse of the above process is shown below:

Input example for – Book number to buy: Integer (1 to n) n-number of books

Input example for – Quantity of books: Integer (1 to n) n-number of books to buy

```
The books tilted "Plato Rocks" under the category "phil" are:
Book No: 01      Title: Plato Rocks      Year: 1975      Language: Greek Weight: 467

Would you like to buy the book? Enter Yes || No : Yes

Enter the book no you wish to buy : one
Invalid Book No! Please enter a valid number to proceed: 02
Invalid Book No! Please enter a valid number to proceed: 1

The minimum price of the book "btitle" is : 36.95
Enter the number of books you wish to buy : 0
Sorry the cart seems to be empty, Please enter a valid number to proceed: ten
Sorry the cart seems to be empty, Please enter a valid number to proceed: 10

*****
*Transaction Complete!
*Customer ID : 5
*Club       : AARP
*Title      : Plato Rocks
*Year       : 1975
*Quantity   : 10
*Total Price : 369.5
*Time       : 2017-12-05-14.30.50
*Thank you, please visit us again.
*****
gsp09 91 % █
```

Whenever there is prompt to input something during the execution of the program, no matter what trash or invalid type values the user inputs related to the asked value, the program is designed in such a way that it takes in the input and reacts accordingly. This was achieved through the help of matching the user inputs to the required regular expressions in java.

Instructions Set

Assuming this is the first time the program application is being set up or executed. We first must set up a connection to the database using the command *% db2 connect to c3421a* and use the script yrb-create, using the command *% db2 -tf yrb-create* which will create the YRB DB schema for you. It will also populate the tables with mock data. Once this is completed successfully, we do the sourcing using the command *% source ~db2leduc/cshrc.runtime*. Then we compile and execute our java program using the commands *% javac CustApp.java* and *% java CustApp* respectively. As the program is running we search through purchase the books from the database. We do this by reacting to the input asked by the program, if everything is entered properly the program will reach the end and exit. If we mess things up to the tables of the database, we can use the script yrb-drop using the command *% db2 -tf yrb-drop* and re-create our copy of YRB DB from the DB2 schema space. A glimpse of the instruction on the terminal is shown below:

Once you have a window (and thus, shell) open on a PRISM machine (e.g., red.cs.yorku.ca), go to the directory that contains the required java and yrb files and you can do the following:

Step 1: *% db2 connect to c3421a*

Step 2: *% db2 -tf yrb-create*

Step 3: *% source ~db2leduc/cshrc.runtime*

Step 4: *% javac CustApp.java*

Step 5: *% java CustApp*

```

gsp09:/eecs/home/kumarraj/EECS/EECS3421/Project_2
File Edit View Search Terminal Help
gsp09 68 % db2 connect to c3421a

Database Connection Information

Database server          = DB2/LINUX8664 11.1.1
SQL authorization ID    = KUMARRAJ
Local database alias    = C3421A

gsp09 69 % db2 -tf yrb-create

DB20000I The SQL command completed successfully.
DB20000I The SQL command completed successfully.
DB20000I The SQL command completed successfully.
DB20000I The SQL command completed successfully.
DB20000I The SQL command completed successfully.
DB20000I The SQL command completed successfully.
DB20000I The SQL command completed successfully.
DB20000I The SQL command completed successfully.

DB20000I The SQL command completed successfully.
DB20000I The SQL command completed successfully.
DB20000I The SQL command completed successfully.
DB20000I The SQL command completed successfully.
DB20000I The SQL command completed successfully.
DB20000I The SQL command completed successfully.
DB20000I The SQL command completed successfully.
DB20000I The SQL command completed successfully.
DB20000I The SQL command completed successfully.

gsp09 70 % source ~/db2educ/cshrc.runtime
gsp09 71 % javac CustApp.java
gsp09 72 % java CustApp

Enter the Customer ID: 5

Customer ID: 5 Customer Name: Andy Aardverk City: Newport News

Would you like to update the customer information? Enter Yes || No : █

```




CustApp.Java



```

/*=====
* CustApp      : A JDBC APP to purchase a book for a customer.
* EECS3421     : Introduction to Database Management Systems, Fall 2017
* Project 2    : CustApp.java
* Student Name : Balakrishnan Lakshmi, Rajkumar
* Student Login : kumarraj
* Student ID   : 213141197
=====*/

```

```

import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.net.*;
import java.text.*;
import java.lang.*;
import java.io.*;
import java.sql.*;

```

```

/*=====
CLASS CustApp
=====*/

```

```

public class CustApp
{
    private Connection conDB;           // Connection to the database system.
    private String url="";              // URL: Which database?
    private String myCustId="";         // Customer Id.
    private Integer custID=0;           // Who are we tallying?
    private String custName="";         // Name of that customer.
    private String custCity="";         // City of that customer.
    private String custUpdate1 = "";    // Update for - New name of the customer.
    private String custUpdate2 = "";    // Update for -New city of the customer.
    private boolean cupname = false;    // Boolean new name
    private boolean cupcity = false;    // Boolean new city
    private ArrayList <String> cat = new ArrayList<>(); // list of categories in the database
    private ArrayList <String> titles;  // list of titles under the catop.
    private int list=0;
    private String catop = "";          // Category option.
    private String titleop = "";        // Title option.
    private String btitle= "";          // Book's title option.
    private int byear=0;                // Book's year.
    private String blanguage = "";      // Book's language.
    private int bweight=0;              // Book's weight.
    private String cart= "";            // Item's in the cart.
    private String booknum="";          // Book's Number from the list of books.
    private double minprice=0;          // Minimum price of the book.
    private String quantity="";         // Quantity of the books to be bought.
    private double finalprice=0;        // Final price of the book to be bought.
    private String bclub="";            // Book's club that offers minimal price.
    Timestamp ctime;                   // Current time.
    DateFormat cdate;                  // Current date.
    String when;                       // When was the book bought - Date & Time.

```

```

Map <Integer,ArrayList<String>> books = new HashMap <Integer,ArrayList<String>> ();

```

```

// Constructor
public CustApp () {
    // 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);
}

// Let's have autocommit turned off. No particular reason here.
try {
    conDB.setAutoCommit(false);
} catch(SQLException e) {
    System.out.print("\nFailed trying to turn autocommit off.\n");
    e.printStackTrace();
    System.exit(0);
}

//First part of the app to get the customer information.
custInfo();

//Second part of the app to get the category and book information.
catnbook();

//Third part of the app to purchase of the book.
buybook();

// 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);
}

// 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 void custInfo()
{
    // Who are we tallying?
    Pattern idPattern = Pattern.compile("\\d+");
    Scanner input1 = new Scanner(System.in);
    System.out.print("\nEnter the Customer ID: ");
    myCustId = input1.nextLine();
    Matcher idmatcher = idPattern.matcher(myCustId);

    //Is this custID for real?
    while(!idmatcher.matches() || !find_customer(Integer.parseInt(myCustId)))
    {
        System.out.print("There is no customer #"+myCustId+" in the database. Please try again: ");
        myCustId = input1.nextLine();
        idmatcher = idPattern.matcher(myCustId);
    }

    // Update the Customer Information
    Scanner input2= new Scanner(System.in);

```

```

Pattern catPattern = Pattern.compile("(?:yes|no)$");
System.out.print("\nWould you like to update the customer information? Enter Yes || No : ");
custUpdate1 = input1.nextLine().toLowerCase();
Matcher catmatcher = catPattern.matcher(custUpdate1);

while(!catmatcher.matches())
{
    System.out.print("The entered option is invalid. Please try again: ");
    custUpdate1 = input1.nextLine().toLowerCase();
    catmatcher = catPattern.matcher(custUpdate1);
}

if (custUpdate1.equals("yes"))
{
    Pattern updatePattern = Pattern.compile("(?:name|city|both)$");
    System.out.print("\nPossible ways of updating customer information are:");
    System.out.print("\n1.Name\n2.City\n3.Both");
    System.out.print("\nChoose from one of the above options:");
    custUpdate2 = input2.nextLine().toLowerCase();
    Matcher updatematcher = updatePattern.matcher(custUpdate2);

    while(!updatematcher.matches())
    {
        System.out.print("The entered option is invalid. Please try again: ");
        custUpdate2 = input2.nextLine().toLowerCase();
        updatematcher = updatePattern.matcher(custUpdate2);
    }

    if(custUpdate2.equals("name"))
    {
        cupname=true;
        System.out.print("\nEnter the new name: ");
        custUpdate2 = input2.nextLine();
        update_customer(Integer.parseInt(myCustId),custUpdate2);
        System.out.print("\nName has been successfully updated to "+custUpdate2 +" !!!\n");
    }
    else if(custUpdate2.equals("city"))
    {
        cupcity=true;
        System.out.print("Enter the new city: ");
        custUpdate2 = input2.nextLine();
        update_customer(Integer.parseInt(myCustId),custUpdate2);
        System.out.print("\nCity has been successfully updated to "+custUpdate2 +" !!!\n");
    }
    else if(custUpdate2.equals("both"))
    {
        System.out.print("Enter the new name: ");
        custUpdate1 = input2.nextLine();
        System.out.print("Enter the new city: ");
        custUpdate2 = input2.nextLine();
        update_customer(Integer.parseInt(myCustId),custUpdate1,custUpdate2);
        System.out.print("\nName has been successfully updated to "+custUpdate1 +" !!!");
        System.out.print("\nCity has been successfully updated to "+custUpdate2 +" !!!\n");
    }
    System.out.print("\n-----");
}
System.out.print("\n-----\n");

}

```

//=====

```

public void catnbook()
{
    Scanner input3 = new Scanner(System.in);
    System.out.print("\nThe categories of books in our database are:\n");
    cat = fetch_categories();
    list=1;
    for(String c: cat)
    {
        System.out.println(list+"."+c);
        list++;
    }

    System.out.print("Choose from one of the above categories to see the book titles: ");
}

```

```

catop = input3.nextLine().toLowerCase();
while(!cat.contains(catop))
{
    System.out.print("There is no category \""+catop+"\" in the database. Please try again!");
    System.out.print("\nEnter the category:");
    catop = input3.nextLine().toLowerCase();
}

System.out.print("\nThe book titles under the category \""+catop+"\" are:\n");
titles = fetch_titles(catop);
list=1;
for(String t: titles)
{
    System.out.println(list+"."+t);
    list++;
}
System.out.print("Choose from one of the above book titles to view its details: ");
Scanner input4 = new Scanner(System.in);
titleop = input4.nextLine();
while(!titles.contains(titleop))
{
    System.out.print("\nThe book \""+titleop+"\" under the category \""+catop+"\" doesn't exist. Please try again!\n");
    System.out.print("\n-----");
    catnbook();
}
System.out.print("\n-----\n");
}

//=====

public void buybook()
{
    books=find_book(titleop,catop);
    if(books.size()>0)
    {
        System.out.println("\nThe books titled \""+titleop+"\" under the category \""+catop+"\" are: ");
        int j = 0;

        while (j <=(books.size()-1))
        {
            btitle = books.get(j).get(0);
            byear = Integer.parseInt(books.get(j).get(1));
            blanguage = books.get(j).get(2);
            bweight = Integer.parseInt(books.get(j).get(3));
            System.out.println("\nBook No: "+j+1+"\tTitle: "+btitle +"\tYear: "+byear+"\tLanguage: "+blanguage+"\tWeight: "+bweight);
            j++;
        }
    }

    Pattern buyPattern = Pattern.compile("^(?:yes|no)$");
    Scanner input5 = new Scanner(System.in);
    System.out.print("\nWould you like to buy the book? Enter Yes || No : ");
    cart = input5.nextLine().toLowerCase();
    Matcher buymatcher = buyPattern.matcher(cart);

    while(!buymatcher.matches())
    {
        System.out.print("The entered option is invalid. Please try again: ");
        cart = input5.nextLine().toLowerCase();
        buymatcher = buyPattern.matcher(cart);
    }
    // System.out.print("cid: "+custID+"\ncatop: "+catop+"\nbtitle: "+btitle+"\nbyear: "+byear+"\nlanguage: "+blanguage+"\nbweight: "+bweight);

    if(cart.equals("yes"))
    {
        Pattern bookNumPattern = Pattern.compile("\\d+");
        System.out.print("\nEnter the book no you wish to buy : ");
        booknum = input5.nextLine();
        Matcher bookNummatcher = bookNumPattern.matcher(booknum);

        while(!bookNummatcher.matches() || !(Integer.parseInt(booknum)>0 && Integer.parseInt(booknum)<=books.size()))
        {

```

```

        System.out.print("Invalid Book No! Please enter a valid number to proceed: ");
        booknum = input5.nextLine();
        bookNummatcher = bookNumPattern.matcher(booknum);
    }

    btitle = books.get(Integer.parseInt(booknum)-1).get(0);
    byear = Integer.parseInt(books.get(Integer.parseInt(booknum)-1).get(1));
    blanguage = books.get(Integer.parseInt(booknum)-1).get(2);
    bweight = Integer.parseInt(books.get(Integer.parseInt(booknum)-1).get(3));

    minprice = min_price(custID, catop, btitle, byear);
    System.out.print("\nThe minimum price of the book \""+btitle+"\" is : "+minprice);

    Pattern quanPattern = Pattern.compile("\\d+");
    System.out.print("\nEnter the number of books you wish to buy : ");
    Scanner input6= new Scanner(System.in);
    quantity = input6.nextLine();
    Matcher quanmatcher = quanPattern.matcher(quantity);

    while(!quanmatcher.matches() || Integer.parseInt(quantity)<=0)
    {
        System.out.print("Sorry the cart seems to be empty, Please enter a valid number to proceed: ");
        quantity = input6.nextLine();
        quanmatcher = quanPattern.matcher(quantity);
    }
    finalprice = minprice*Integer.parseInt(quantity);
    //System.out.print("cid: "+custID+"\ncatop: "+catop+"\nbtitle: "+btitle+"\nbyear: "+byear+"\nlanguage: "+blanguage+"\nbweight: "+bweight+"\nQuantity: "+quantity+"\nClub: "+bclub+"\n");
    insert_purchase(custID, bclub, btitle, byear, Integer.parseInt(quantity));
    System.out.print("\n*****");
    System.out.print("\n*Transaction Complete!");
    System.out.print("\n*Customer ID : "+custID);
    System.out.print("\n*Club      : "+bclub);
    System.out.print("\n*Title       : "+btitle);
    System.out.print("\n*Year        : "+byear);
    System.out.print("\n*Quantity    : "+quantity);
    System.out.print("\n*Total Price : "+finalprice);
    System.out.print("\n*Time       : "+when);
    System.out.print("\n*Thank you, please visit us again.");
    System.out.print("\n*****\n");

}
else
{
    System.out.print("\n\n*****");
    System.out.print("\n*   Thanks for visisting us.   *");
    System.out.print("\n*   Have a nice day.           *");
    System.out.print("\n*****\n");
}

}

```

/******

```

public boolean find_customer(int myCustId) {
    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#1 failed in prepare");
        System.out.println(e.toString());
        System.exit(0);
    }
}

```

```

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

// Answer
try {
    if (answers.next()) {
        inDB = true;
        custId = answers.getInt("cid");
        custName = answers.getString("name");
        custCity = answers.getString("city");
        System.out.println("\nCustomer ID: " + custId + "\tCustomer Name: " + custName + "\tCity: " + custCity);
    } else {
        inDB = false;
        custName = 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 inDB;
}

//-----

public boolean update_customer(int myCustId2, String... custUpdate2) {
    String    queryText = "";    // The SQL text.
    PreparedStatement querySt = null; // The query handle.

    boolean    inDB    = false; // Return.

    if(cupname)
    {
        queryText =
            "UPDATE yrb_customer "
            + "SET name = ?      "
            + "WHERE cid = ?      ";
    }
    else if(cupcity)
    {
        queryText =
            "UPDATE yrb_customer "
            + "SET city = ?      "
            + "WHERE cid = ?      ";
    }
    else
    {

```

```

        queryText =
            "UPDATE yrb_customer  "
            + "SET name = ?, city = ? "
            + "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 {
        if(cupname != null || cupcity != null)
        {
            querySt.setString(1, custUpdate2[0]);
            querySt.setInt(2, myCustId2);
        }
        else
        {
            querySt.setString(1, custUpdate2[0]);
            querySt.setString(2, custUpdate2[1]);
            querySt.setInt(3, myCustId2);
        }
        querySt.executeUpdate();
        inDB = true;
    } 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);
    }

    return inDB;
}

//-----

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

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

    // Prepare the query.
    try {
        querySt = conDB.prepareStatement(queryText);
    } catch (SQLException e) {
        System.out.println("SQL#2 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#2 failed in execute");
    }
}

```



```

        System.out.println(e.toString());
        System.exit(0);
    }

    // Answer
    try {
        for (int i=1; answers.next(); i++)
        {
            String mycategories = answers.getString("cat");
            cats.add(mycategories);
        }
    } catch (SQLException e) {
        System.out.println("SQL#2 failed in cursor.");
        System.out.println(e.toString());
        System.exit(0);
    }

    // Close the cursor.
    try {
        answers.close();
    } catch (SQLException e) {
        System.out.print("SQL#2 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#2 failed closing the handle.\n");
        System.out.println(e.toString());
        System.exit(0);
    }

    return cats;
}

//-----

public ArrayList <String> fetch_titles(String category) {
    String      queryText = ""; // The SQL text.
    PreparedStatement querySt = null; // The query handle.
    ResultSet      answers = null; // A cursor.
    ArrayList <String> titles = new ArrayList < String > ();

    queryText =
        "Select distinct title " + " from yrb_book where cat = ? "+"and title in (select o.title from " +
        "yrb_offer o " + "where o.club in " + "(Select club " + "from yrb_member " +
        "where cid = ?)) and year in " + "(select o.year " + "from yrb_offer o " +
        "Where o.club in (Select club " + "from yrb_member " + "where cid = ?))";

    // Prepare the query.
    try {
        querySt = conDB.prepareStatement(queryText);
    } catch (SQLException e) {
        System.out.println("SQL#2 failed in prepare");
        System.out.println(e.toString());
        System.exit(0);
    }

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

    // Answer
    try {

```

```

        for (int i=0; answers.next(); i++)
        {
            String mytitles= answers.getString("title");
            titles.add(mytitles);
        }
    } catch (SQLException e) {
        System.out.println("SQL#2 failed in cursor.");
        System.out.println(e.toString());
        System.exit(0);
    }

    // Close the cursor.
    try {
        answers.close();
    } catch (SQLException e) {
        System.out.print("SQL#2 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#2 failed closing the handle.\n");
        System.out.println(e.toString());
        System.exit(0);
    }

    return titles;
}
//-----

```

```

public Map <Integer,ArrayList<String>> find_book(String title,String category) {
    String queryText = ""; // The SQL text.
    PreparedStatement querySt = null; // The query handle.
    ResultSet answers = null; // A cursor.
    ArrayList <String> bookinfo = new ArrayList < String > ();
    Map <Integer,ArrayList<String>> bookDetails = new HashMap <Integer,ArrayList<String>> ();
    String btitles = "";
    Integer bweights;
    String blanguages;
    Integer byears;

    queryText =
        "SELECT * "
        + "FROM yrb_book "
        + "WHERE title = ? and cat = ?";
    // Prepare the query.
    try {
        querySt = conDB.prepareStatement(queryText);
    } catch (SQLException e) {
        System.out.println("SQL#2 failed in prepare");
        System.out.println(e.toString());
        System.exit(0);
    }

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

    // Answer.
    try {
        int i;
        for (i=0; answers.next(); i++)
        {
            bookinfo.clear();

```

```

        btitles = answers.getString("title");bookinfo.add(btitles);
        byears = answers.getInt("year");bookinfo.add(byears.toString());
        blanguages = answers.getString("language");bookinfo.add(blanguages);
        bweights = answers.getInt("weight");bookinfo.add(bweights.toString());
        bookDetails.put(i, bookinfo);
    }
} catch (SQLException e) {
    System.out.println("SQL#2 failed in cursor.");
    System.out.println(e.toString());
    System.exit(0);
}

// Close the cursor.
try {
    answers.close();
} catch (SQLException e) {
    System.out.print("SQL#2 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#2 failed closing the handle.\n");
    System.out.println(e.toString());
    System.exit(0);
}

return bookDetails;
}

//-----

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

    String      queryText2 = ""; // The SQL text.
    PreparedStatement querySt2 = null; // The query handle.
    ResultSet    answers2 = null; // A cursor.

    double price=0;

    queryText =
        "SELECT min(price) "
        +"FROM yrb_offer "
        +"WHERE title = ? AND year = ? "
        +"and club in (SELECT club FROM yrb_member WHERE cid = ?)";

    queryText2 =
        "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);
        querySt2 = conDB.prepareStatement(queryText2);
    } 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.setInt(2, year);
        querySt.setInt(3, cid);
        answers = querySt.executeQuery();
    } catch (SQLException e) {

```

```

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

// Answer.
    try {
        if (answers.next())
        {
            price = answers.getDouble(1);
        }
    } catch (SQLException e) {
        System.out.println("SQL#2 failed in cursor.");
        System.out.println(e.toString());
        System.exit(0);
    }

// Execute the query2.
    try {
        querySt2.setInt(1, byear);
        querySt2.setInt(2, custID);
        querySt2.setString(3, btitle);
        querySt2.setDouble(4, price);
        answers2 = querySt2.executeQuery();
    } catch (SQLException e) {
        System.out.println("SQL#1 failed in execute");
        System.out.println(e.toString());
        System.exit(0);
    }

// Answer2.
    try {
        if (answers2.next())
        {
            bclub = answers2.getString(1);
        }
    } catch (SQLException e) {
        System.out.println("SQL#2 failed in cursor.");
        System.out.println(e.toString());
        System.exit(0);
    }

// Close the cursor.
    try {
        answers.close();
        answers2.close();
    } catch (SQLException e) {
        System.out.print("SQL#2 failed closing cursor.\n");
        System.out.println(e.toString());
        System.exit(0);
    }

// We're done with the handle.
    try {
        querySt.close();
        querySt2.close();
    } catch (SQLException e) {
        System.out.print("SQL#2 failed closing the handle.\n");
        System.out.println(e.toString());
        System.exit(0);
    }

    return price;
}

//-----

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

    ctime = new Timestamp(System.currentTimeMillis());
    cdate = new SimpleDateFormat("yyyy-MM-dd-HH.mm.ss");
    when = cdate.format(ctime);

```

```

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, cid);
    querySt.setString(2, club);
    querySt.setString(3, title);
    querySt.setInt(4, year);
    querySt.setString(5, when);
    querySt.setInt(6, qnty);
    querySt.executeUpdate();
} catch (SQLException e) {
    System.out.println("SQL#6 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#6 failed closing the handle.\n");
    System.out.println(e.toString());
    System.exit(0);
}

}

//*****

public static void main(String[] args)
{
    CustApp ct = new CustApp();
}
}

```