EECS 3421 Project II: APPs User Manual

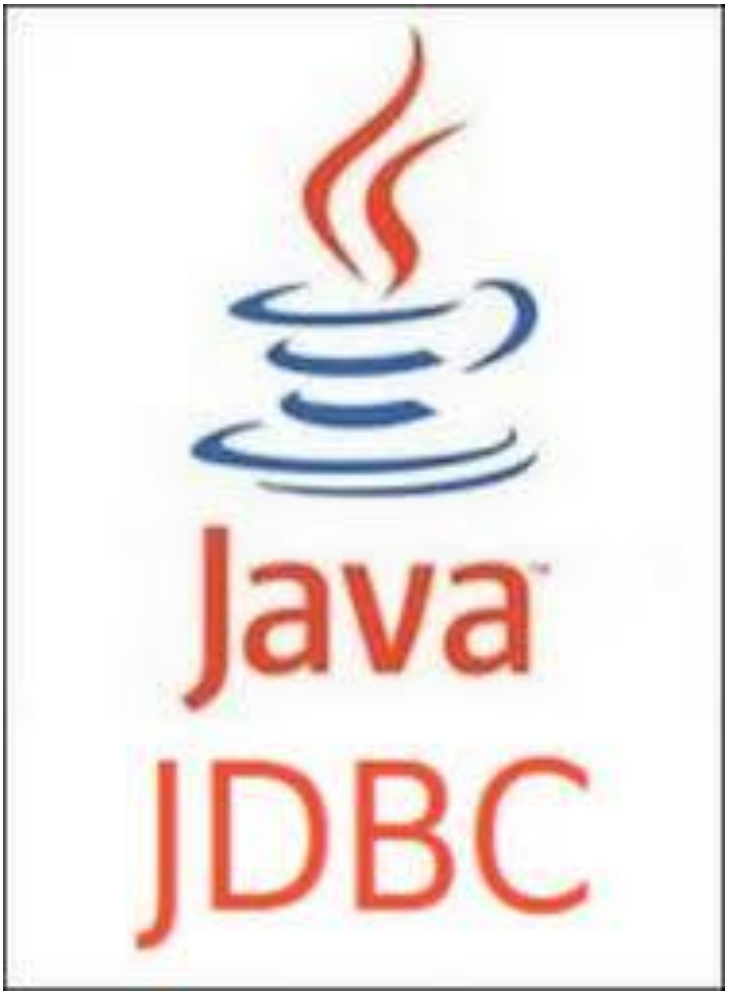
Prof. Jarek Gryz

York University

Vladimir Martintsov

Student #: 212829206

Email: vlad95@my.yorku.ca



Welcome to YRB App!

This application lets you to purchase the books from York River Bookseller's Database.

Setting up the connection to database:

To run this program appropriately, you need to be familiar with Linux Terminal and have the following required files in **SAME** directory:

yrb-create

YRBApp.class

YRBApp.java

MainClass.class

MainClass.java

Once you have the files in the **SAME** directory, open Terminal from that directory and start typing the following commands **one by one**:

source ~db2leduc/cshrc.runtime

db2 connect to c3421a

db2 -tf yrb-create

db2 connect reset

db2 terminate

javac MainClass.java

java MainClass

At this point, you should be greeted with a greeting message from the program. As you type in the commands you should be getting messages that indicate successful connection and execution of the commands you type.

Should you not be greeted with a welcome message, that means that something went wrong. You need to make sure that you have connection to database and retype the commands above.

Operating the program:

The operation of the program is very intuitive. At any point of the program you can use the keyword "EXIT" – **case sensitive** – to terminate the program. If the program is asking you to enter **yes/no or y/n**, please do so, otherwise it will not accept anything other than yes/no or y/n or EXIT.

At the points when you are asked to enter **integers** as inputs, please enter integers, otherwise you will be prompted with an error message.

Tools used in this project:

The tools that were used in this project were Eclipse (Java development framework), as well as the following Java libraries:

```
import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Map;
import java.util.NoSuchElementException;
import java.util.Scanner;
import java.util.TreeMap;
import java.util.*;
```

Screenshots of using the program:

As we enter the ID of the User, the information about the user will appear on the screen. Then the program will ask the user if they want to update the City or Name. *Should the customer enter an invalid input to question with a yes/no or y/n answer, it will show an error:*

```
Customer ID: 20       Customer Name: Finwick Cooper        City: Dublin
Before we show you the list of book categories,would you like to update the cust
omer info?Enter yes/no or y/n:
asdasd
You have entered an invalid answer.Kindly use yes/no or y/n as the inputs, and t
ry again.
```

If the customer enters yes, we are prompted to an update customer menu:

```
yes
At this time, you can only update your Name or CityWhat would you like to update
? Kindly enter:
1 - to update your Name
2 - to update your City
3 - to Exit
```

If we exit the update menu, we are shown the list of categories:

```
Would you like to update anything else?no
Alright, here are the available book cateogries:1 - children
2 - cooking
3 - drama
4 - guide
5 - history
6 - horror
7 - humor
8 - mystery
9 - phil
10 - romance
11 - science
12 - travel
Please enter the number that corresponds tothe category of your interest
```

Should we enter an integer that is not corresponding to the category, we will be prompted to an error and asked to enter the proper input again. Once we enter the correct input we are shown the list of book titles in this category:

```
Please enter the number that corresponds tothe category of your interest
45
You have entered an invalid selectionPlease enter a number corresponding to your
category of interest
2
Here are the books we have in selected Category:
cooking
Available Books:
1 - Recipes for Humans, 2000, Plutonian,705;
2 - Vegetables are Good!, 1987, English,292;
3 - Tampopo Oishii, 1995, Japanese,276;
4 - Radiator Barbecuing, 1998, English,154;
5 - Food for Dummies, 2000, English,234;
6 - Ringo to Nashi, 1993, Japanese,334;
7 - Aubergines!, 1996, French,296;
8 - Nothing but Steak, 1991, English,338;
9 - Yum, Yum, English Cooking, 1993, English,57;
10 - Cuisine Anglaise!?, 1993, French,49;
```

The program also allows the user to enter 0, in case if they made a mistake in selecting their category:

```
Available Books:
1 - Recipes for Humans, 2000, Plutonian,705;
2 - Vegetables are Good!, 1987, English,292;
3 - Tampopo Oishii, 1995, Japanese,276;
4 - Radiator Barbecuing, 1998, English,154;
5 - Food for Dummies, 2000, English,234;
6 - Ringo to Nashi, 1993, Japanese,334;
7 - Aubergines!, 1996, French,296;
8 - Nothing but Steak, 1991, English,338;
9 - Yum, Yum, English Cooking, 1993, English,57;
10 - Cuisine Anglaise!?, 1993, French,49;
11 - Rabbits are nice, 2000, English,186;
12 - The Fickle Pickle, 2000, English,285;
Please enter the number that corresponds tothe title of your interest
If you made a mistake, or you don'tsee the title of your interest, press 0 to go
 back tothe list of Categories
```

Other than that, the program works in this manner until it lets the user buy the books.

The source code:

MainClass.java:

```
//Main class that calls the working class

import java.util.*;
public class MainClass {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		YRBApp app = new YRBApp(sc);

	}

}
```

YRBApp.java:

```
//YRBAPP CLASS
//Done by Vladimir Martintsov, EECS Login: vlad95

//Available on https://github.com/vladimir95/DB2_SQL_Project

import java.math.BigDecimal;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Map;
import java.util.NoSuchElementException;
import java.util.Scanner;
import java.util.TreeMap;


public class YRBApp {

	private Scanner scanner; //Scanner used to take in user's input
	private Connection conDB;   // Connection to the database system.
	private String url;         // URL of the database?
	private final int MAXNAMELENGTH = 20; //Max name length allowed in this
database
	private final int MAXCITYLENGTH = 15; //Max City length allowed in this
database
	private String userStringInput; //Current String input by the user
	private String userIntInput; //current Integer Input by the user
	private int custID;     // Customer ID
	private String  custName;   // Name of that customer.
	private String custCity;      //Customer's City
	private int userChoice;       //User's choices in the Update details
menu

	private Map<Integer,String> categories =
			new TreeMap<Integer,String>(); //categories of the books

	private Map<Integer,ArrayList<String>> bookTitles =
			new TreeMap<Integer,ArrayList<String>>(); //book titles in
selected category

	private int categoryChosen; //Category chosen from the drop down list
	private int titleNumberChosen; //Book Title chosen form drop down menu
list

	private Map<Integer,ArrayList<String>> bookInformation =
			new TreeMap<Integer,ArrayList<String>>(); //Book
Information displayed to the customer

	private int bookNumberChosen; //Book number chosen after title menu
	private int resetFlag = 0; //Flag needed to reset the transaction
	private int bookYear; //Year of the book
	private String bookTitle; //Title of the book
	private float minPrice; //Minimum Price of the book
	private int numberOfBooks; //Number of the books customer wants
	private String clubName; //Customer's Club name
	private float totalPrice; //Total Price for the selected books
```

```
	//Reserved word to exit throughout any point of the program
	private final String abort = "EXIT";

	public YRBApp (Scanner in){

		//Initialize the scanner
		scanner = in;



		//End of File input handling
		try {
			// 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);
			}

			//Welcome message
			message();

			//Start taking User's Input
			//Make sure that only integers are entered
			userIntInput = scanner.nextLine();
			while(!intInputCheck(userIntInput)){
				userIntInput = scanner.nextLine();
			};
			custID = Integer.parseInt(userIntInput);

			//Check if the ID is not in database
			while(!find_customer(custID)){
				System.out.print("We are sorry, the customer ID you
entered" +
```

```
                                    "is not in our database. Please enter a
valid customer ID: ");

                        userIntInput = scanner.nextLine();

                        while(!intInputCheck(userIntInput)){
                              userIntInput = scanner.nextLine();
                        };
                        custID = Integer.parseInt(userIntInput);
                  }

                  //Found the customer, ask to update the City and Name
                  System.out.println("Before we show you the list of book
categories,"+
                              "would you like to update the customer info?"+
                              "Enter yes/no or y/n:");

                  userStringInput = scanner.nextLine();

                  while(!yesNoInputCheck(userStringInput)){
                        userStringInput = scanner.nextLine();

                  }

                  //If customer selects yes, run the Update_cusomter

      while(userStringInput.equals("yes")||userStringInput.equals("y")){
                        update_customer(custID);
                        userStringInput = scanner.nextLine();
                        while(!yesNoInputCheck(userStringInput)){
                              userStringInput = scanner.nextLine();

                        }

                  }


                  //Done with updates, let's show the categories
                  //THE "CATEGORIES" LOOP
                  while (resetFlag==0){

                  System.out.print("Alright, here are the available book
cateogries:");

                  //Should there be a problem with fetching categories,
abort.
                  //Must be some database issue on user's end
                  while(!fetch_categories()){
                        System.out.println("We are sorry about this."+
                                    "This session is now terminated. Please
rerun the"+
                                    "program and try again");
                        System.exit(0);

                  };

                  //Categories fetched fine
```

```
                  System.out.println("Please enter the number that
corresponds to"+
                              "the category of your interest");
                  userIntInput = scanner.nextLine();
                  while(!intInputCheck(userIntInput)){
                        userIntInput = scanner.nextLine();
                  };

                  categoryChosen = Integer.parseInt(userIntInput);

                  //If customer enters categories not in the list
                  while(!categories.containsKey(new
Integer(categoryChosen))){
                        System.out.println("You have entered an invalid
selection"+
                                    "Please enter a number corresponding to
your"+
                                    "category of interest");
                        userIntInput = scanner.nextLine();
                        while(!intInputCheck(userIntInput)){
                              userIntInput = scanner.nextLine();
                        };

                        categoryChosen = Integer.parseInt(userIntInput);


                  }


                  //We now display all the books in the category
                  //for smooth program flow.

                  System.out.println("Here are the books we have in selected
Category:");

                  //Fetching the book titles. Abort if something is wrong
                  while(!fetch_titles(categoryChosen)){
                        System.out.println("We are sorry about this."+
                                    "This session is now terminated. Please
rerun the"+
                                    "program and try again");
                        System.exit(0);

                  };

                  //Ask customer for the title they are interested in
                  //If they feel like they don't see the book allow them
                  //to press 0 to go back to previous menu

                  System.out.println("Please enter the number that
corresponds to"+
                              "the title of your interest");
                  System.out.println("If you made a mistake, or you don't"+
                              "see the title of your interest, press 0 to go
back to"
                              + "the list of Categories");
```

```
                    //Take the input, perform the generic input check
                    userIntInput = scanner.nextLine();
                    while(!intInputCheck(userIntInput)){
                          userIntInput = scanner.nextLine();
                    };

                    titleNumberChosen = Integer.parseInt(userIntInput);



                    // If customer enters a title that is not in the list and
not 0
                    // then they must be entering some gibberish
                    while((titleNumberChosen!=0)&&!bookTitles.containsKey(new
Integer(titleNumberChosen))){
                          System.out.println("You have entered an invalid
selection"+
                                      "Please enter a number corresponding to
your"+
                                      "category of interest");
                          userIntInput = scanner.nextLine();
                          while(!intInputCheck(userIntInput)){
                                userIntInput = scanner.nextLine();
                          };

                          titleNumberChosen = Integer.parseInt(userIntInput);


                    }

                    //Should customer enter 0, keep the flag 0
                    //Rerun the "CATEGORIES" loop
                    if(titleNumberChosen==0){
                          resetFlag = 0;
                          continue;
                          //bookInformation.clear();

                    }
                    else {
                          resetFlag = 1;
                    }



                    //Show the customer the books with selected title
                    System.out.println("Here is the book we have of the
selected title: ");

                    //Perform book fetch from database, and see if anything is
wrong
                    while(!find_book(titleNumberChosen,categoryChosen)){
                          System.out.println("We are sorry about this."+
                                      "This session is now terminated. Please
rerun the"+
                                      "program and try again");
                          System.exit(0);
```

```
                }


                //Ask customer for the title they are interested in
                System.out.println("Please enter the number that
corresponds to"+
                                   "the book of your interest for more
information. " +
                                   "If you don't like this book, please use the
aborting keyword EXIT and" +
                                   " and rerun the program");

                //Take in cusotmer's input
                userIntInput = scanner.nextLine();

                while(!intInputCheck(userIntInput)){
                      userIntInput = scanner.nextLine();
                }

                bookNumberChosen = Integer.parseInt(userIntInput);


                //Make sure that the customer selects that book
                while (!bookInformation.containsKey(new
Integer(bookNumberChosen))){
                      System.out.println("You have entered an invalid
selection."+
                                         "Please enter a number corresponding to
your"+
                                         "book of interest");
                      userIntInput = scanner.nextLine();
                      while(!intInputCheck(userIntInput)){
                            userIntInput = scanner.nextLine();
                      };

                      bookNumberChosen = Integer.parseInt(userIntInput);

                }

                }

                //Alight, the customer has selected the book of his
interest.
                //Now, we need to offer the price to the customer based on
his club

                System.out.println("We select the minimum price for the
book"+
                                   " you chose based on your club membership.");


                //Check if fetching the min Price is not running
                while(!find_minPrice(custID, bookTitle, bookYear)){
                      System.out.print("Something went wrong with
calculating the price"+
                                         "Please try agian");
```

```
                        System.exit(0);

                  }

                  //Check if finding the club did not work
                  //We need the club to calculate the price of the book
                  while(!(find_club(custID, minPrice))){
                        System.out.print("Something went wrong with fetching
your club"+
                                    "Please try agian");
                        System.exit(0);

                  }

                  //Justify the price of the book
                  System.out.println("You are getting this price because you
are a part of" +
                              " this club " + clubName);



                  //Alright, now we ask how many books the customer wants

                  System.out.println("How many books were you looking for?"+
                              "Please enter an amount that is bigger than
0");
                  userIntInput = scanner.nextLine();

                  while(!intInputCheck(userIntInput)){
                        userIntInput = scanner.nextLine();
                  }




                  numberOfBooks = Integer.parseInt(userIntInput);

                  //Check if the input is valid
                  while(numberOfBooks <=0){
                        System.out.println("You entered an invalid amount."+
                                    "Please enter an amount bigger or equal
to 1");
                        userIntInput = scanner.nextLine();

                        while(!intInputCheck(userIntInput)){
                              userIntInput = scanner.nextLine();
                        }
                        numberOfBooks = Integer.parseInt(userIntInput);

                  }

                  //Calculate the price of the book
                  totalPrice = minPrice*numberOfBooks;

                  System.out.println("The total price for this many books is"
+
                              totalPrice);
```

```
                  System.out.print("Would you like to buy these books?
yes/no?");

                  //Ask if the user wants to buy these books or not
                  userStringInput = scanner.nextLine();

                  while(!yesNoInputCheck(userStringInput)){
                        userStringInput = scanner.nextLine();

                  }

                  //If yes, insert the purchase into the database

      if(userStringInput.equals("yes")||userStringInput.equals("y")){

      while(!(insert_purchase(custID,clubName,bookTitle,bookYear,numberOfBook
s))){
                              System.out.println("Something went wrong with
inserting your purchase");
                              System.exit(0);

                        }
                        System.out.println("Transaction Complete. Thank
you!");

                  }

                  //If not, then terminate the program
                  else if
(userStringInput.equals("no")||userStringInput.equals("n")){

                        System.out.println("That is unfortunate. Thank you
for visiting us!");
                  }



                  // 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);
                  }
```

```
                  //Catching End of File
            }
            catch(NoSuchElementException e){
                  System.out.println("We are sorry but you have reached the
End of File."+
                              "Please check your inputs and  rerun the
program again.");
            }



      }

      private void message(){
            System.out.print(" Welcome to the York River Bookseller's
Database." +
            "To use this application appropriately, make sure you have your
connection set up "+
            "and your ID ready. To set up teh connection properly, please
refer to user manual \n" +
            "At any point in this program, feel free to type SPECIFIC keyword
EXIT to terminate "+
            "the program. The keyword IS case SENSITIVE! \n" +
            "Please use integers on your keypad as inputs OR " + ""
            + "YES/Y or NO/N (yes/y or no/n are also acceptable) in the
prompts that ask "+
            "for your action. \n"
            + "Remember: the Name supported by the database is of 20
characters and City - 15 \n"+
            "Should there be a critical error, the program will terminate
automatically. Simply "+
            "rerun the program to start again. The program cannot and should
not run if something " +
            "goes wrong internally.\n"+
            "You are all set and good to go! \n"
            + "Let's start with entering your customer ID:");
      }

      //Helper Method that enforces the user to enter yes/no or y/no
      private boolean yesNoInputCheck(String s){
            try{
                  String temporary = s.toLowerCase();}
            catch(NullPointerException e){
                  System.out.println("We are sorry, but you have reached some
error"+
                              "The program will end now. Please relaunch it
to try again.");
                  System.exit(0);
            }

            String temporary = s.toLowerCase();
            if(temporary.equals(abort)){
                  System.out.println("This operation has been aborted. Thank
you, good bye!");
```

```
                System.exit(0);
            }

            if (temporary.equals("yes")||temporary.equals("no")
                        ||temporary.equals("n")||temporary.equals("y")){
                return true;
            }
            else if (temporary.isEmpty()){
                System.out.println("You have entered an invalid answer."+
                            "Kindly use yes/no or y/n as the inputs, and
try again.");
                return false;
            }

            else {
                System.out.println("You have entered an invalid answer."+
                            "Kindly use yes/no or y/n as the inputs, and
try again.");
                return false;
            }
      }

      // This method checks if there are no abnormal inputs that are passed
as strings
      private boolean stringInputCheck(String s){
            boolean correctInput = false;
            try{
                String temporary = s.toLowerCase();
                correctInput = true;
            }
            catch(NullPointerException e){
                System.out.println("We are sorry, but you have reached the
End of File."+
                            "The program will end now. Please relaunch it
to try again.");
                System.exit(0);
            }

            String temporary = s.toLowerCase();

            if(temporary.equals(abort)){
                System.out.println("This operation has been aborted. Thank
you, good bye!");
                System.exit(0);
            }

            if (temporary.isEmpty()){
                System.out.println("You have entered an empty answer."+
                            "Kindly type your answer, and try again.");
                correctInput = false;
            }


            return correctInput;
      }
```

```
	//This method ensures that the user is entering integers only as
required
	private boolean intInputCheck(String a){


		if(a.equals(abort)){
			System.out.println("This operation has been aborted. Thank
you, good bye!");
			System.exit(0);
		}

		if (a.isEmpty()){
			System.out.println("You have entered an empty answer."+
					"Kindly type your answer, and try again.");
			return false;
		}

		try{
			int temp = Integer.parseInt(a);}
		catch(NumberFormatException e){
			System.out.println("We are sorry, but you have entered an
invalid integer"+
					"The program requires integers as inputs.
Please try again.");
			return false;

		}
		return true;
	}

	//This method retrieves customer ID from database
	public boolean find_customer(int input) {
		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, 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()) {
                    inDB = true;
                    custID = answers.getInt("cid");
                    custName = answers.getString("name");
                    custCity = answers.getString("city");
                    //Output the customer information
                    System.out.println("Customer ID: " + custID
                                + "       Customer Name: " + custName
                                + "        City: " + 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;
        }

        //This method allows the user to update it's information in Database
        public void update_customer(int id){

            //Handling user's inputs
            System.out.println("At this time, you can only update your Name
or City"+
                        "What would you like to update? Kindly enter: \n" +
                        "1 - to update your Name \n" +
                        "2 - to update your City \n" +
```

```
                        "3 - to Exit");
            userIntInput = scanner.nextLine();
            while(!intInputCheck(userIntInput)){
                  userIntInput = scanner.nextLine();
            };
            userChoice = Integer.parseInt(userIntInput);

            System.out.println("We got to this point");

            //Making sure the customer enters integers only
            while(!(userChoice==1||userChoice==2||userChoice==3)){
                  System.out.println("You have entered an invalid
selection."+
                              "What would you like to update? Kindly enter:
\n" +
                              "1 - to update your Name \n" +
                              "2 - to update your City \n" +
                              "3 - to Exit");
                  userIntInput = scanner.nextLine();
                  while(!intInputCheck(userIntInput)){
                        userIntInput = scanner.nextLine();
                  };
                  userChoice = Integer.parseInt(userIntInput);


            }

            //Now finally execute the query
            //Updating the name
            if (userChoice == 1){
                  System.out.println("Please enter your new Name:");
                  userStringInput = scanner.nextLine();

                  while(!stringInputCheck(userStringInput)){
                        userStringInput = scanner.nextLine();

                  }

                  //Handle too long inputs
                  if (userStringInput.length()>=MAXNAMELENGTH){
                        System.out.print("We are sorry, but this name is too
long for this database"+
                                    "to accept. Please have your input up to
and including 20 characters"+
                                    "Would you like to still update anything
else?");
                        return;
                  }
                        //Run the helper method
                        while(!updateCustomerName(id,userStringInput)){
                              System.out.print("Something went wrong. Please
enter your name again");
                              userStringInput = scanner.nextLine();
                        }

                        //updateDB = true;
```

```
                        System.out.print("Would you like to update anything
else?");
                        return;
                  }



            //Change the city
            if (userChoice == 2){
                  System.out.print("Please enter your new City:");
                  userStringInput = scanner.nextLine();

                  while(!stringInputCheck(userStringInput)){
                        userStringInput = scanner.nextLine();

                  }

                  //Handle long input
                  if (userStringInput.length()>=MAXCITYLENGTH){
                        System.out.print("We are sorry, but this city name is
too long for this database"+
                                    "to accept. Please have your input up to
and including 15 characters"+
                                    "Would you like to still update anything
else?");
                        return;
                  }
                  while(!updateCustomerCity(id,userStringInput)){
                        System.out.print("Something went wrong. Please enter
your city again");
                        userStringInput = scanner.nextLine();
                        }

                  //updateDB = true;
                  System.out.print("Would you like to update anything
else?");
                  return;
            }
            //Allow customer to exit if they change their mind
            if (userChoice == 3){
                  System.out.print("Alright, nothing to update." +
                              "Would you like to still update something?
Enter yes or y to do so. ");
                  return;
            }

            //In case if they come back and enter wrong input, they need to
enter 1-3 only
            System.out.println("You have entered an invalid selection. Kindly
use numbers from"+
                        " 1-3. Would you like to try again? Type yes or no");
            return;
      }



//Helper method to update the Name in the database
```

```
private boolean updateCustomerName(int id, String name){
	//Now finally execute the query
	String            queryText = "";      // The SQL text.
	PreparedStatement querySt   = null;    // The query handle.
	ResultSet         answers   = null;    // A cursor.

	boolean           updateDB = false; //Return variable


	queryText =
	            "UPDATE yrb_customer SET name = ? 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.setString(1, name);
		querySt.setInt(2, id);
		querySt.executeUpdate();
		System.out.print("Update Successful!");
		updateDB = true;
	} catch(SQLException e) {
		System.out.println("SQL#1 failed in execute");
		System.out.println(e.toString());
		System.exit(0);
	}

	//Show the updated info
	while(!find_customer(id)){
		System.out.println("Something went wrong in fetching your updated
info."+
				"Please try again");
		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 updateDB;

}

//Helper method to update the City in the database
```

```
private boolean updateCustomerCity(int id, String city){
	//Now finally execute the query
	String            queryText = "";      // The SQL text.
	PreparedStatement querySt   = null;    // The query handle.
	ResultSet         answers   = null;    // A cursor.

	boolean           updateDB = false;
	boolean inDB      = false;  // Return.

	queryText =
	            "UPDATE yrb_customer SET 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 {
		querySt.setString(1, city);
		querySt.setInt(2, id);
		querySt.executeUpdate();
		System.out.print("Update Successful!");
		updateDB = true;
	} catch(SQLException e) {
		System.out.println("SQL#1 failed in execute");
		System.out.println(e.toString());
		System.exit(0);
	}


	//Show the updated info
	while(!find_customer(id)){
		System.out.println("Something went wrong in fetching your updated
info."+
				"Please try again");
		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 updateDB;

}
```

```
//Method to fetch the available categories form the database
public boolean fetch_categories(){
	String queryText = ""; // The SQL text.
	PreparedStatement querySt = null; // The query handle.
	ResultSet answers = null; // A cursor.
	boolean inDB = false;

	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);
	}

	// Any answer?
	try {
		for (int i = 1; answers.next(); i++) {
			String category = answers.getString("cat");
			categories.put(i, category);
		}
	} 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);
```

```
		}

		//Now let's print the categories
		try{
			for (int i = 1; i<categories.size()+1;i++){
				System.out.println(i + " - " + categories.get(i));
				inDB = true;

			}
		}catch(NullPointerException e){
			System.out.println("Something went wrong with fetching" +
					"the available book cateogries");

		}


		return inDB;
	}

	//Method that fetches the book titles associated with this category
	public boolean fetch_titles(int categoryNumber) {
		String queryText = ""; // The SQL text.
		PreparedStatement querySt = null; // The query handle.
		ResultSet answers = null; // A cursor.
		boolean inDB = false;


		queryText = "SELECT B.title, B.year, B.language, B.weight"+
				" FROM yrb_book B"+
				" WHERE B.cat = ?";

		// Prepare the query.
		try {
			querySt = conDB.prepareStatement(queryText);
		} catch (SQLException e) {
			System.out.println("SQL#3 failed in prepare");
			System.out.println(e.toString());
			System.exit(0);
		}

		// Execute the query.
		try {
			//temporary variable
			String categoryName = categories.get(categoryNumber);

			System.out.println(categoryName);
			//Now run the query
			querySt.setString(1, categoryName);
			answers = querySt.executeQuery();
		} catch (SQLException e) {
			System.out.println("SQL#3 failed in execute");
			System.out.println(e.toString());
			System.exit(0);
		}

		// Any answer?
		try {
```

```
            String title;
            Integer weight;
            String language;
            Integer year;
            for (int i = 1; answers.next(); i++) {
                  title = answers.getString("title");
                  year = answers.getInt("year");
                  language = answers.getString("language");
                  weight = answers.getInt("weight");
                  bookTitles.put(i, new ArrayList < String >
(Arrays.asList(title,
                              Integer.toString(year), language,
Integer.toString(weight))));

            }
      } catch (SQLException e) {
            System.out.println("SQL#3 failed in cursor.");
            System.out.println(e.toString());
            System.exit(0);
      }

      // Close the cursor.
      try {
            answers.close();
      } catch (SQLException e) {
            System.out.print("SQL#3 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#3 failed closing the handle.\n");
            System.out.println(e.toString());
            System.exit(0);
      }

      //Now let's try printing the book titles
      try{
            String title;
            Integer weight;
            String language;
            Integer year;
            System.out.println("Available Books:");
            for (int i = 1; i<bookTitles.size()+1;i++){
                  title = bookTitles.get(i).get(0);
                  year = Integer.parseInt(bookTitles.get(i).get(1));
                  language = bookTitles.get(i).get(2);
                  weight = Integer.parseInt((bookTitles.get(i).get(3)));

                  System.out.println(i + " - " + title +", " + year +", " +
                              language + "," + weight + ";");
                  inDB = true;
            }
```

```
	}catch(NullPointerException e){
		System.out.println("Something went wrong with fetching" +
				"the available book titles");

	}



	return inDB;
}


//Based on selected category and book numbers, we now find book information
public boolean find_book(int titleNumber, int categoryNumber) {
	String queryText = ""; // The SQL text.
	PreparedStatement querySt = null; // The query handle.
	ResultSet answers = null; // A cursor.
	boolean inDB = false;
	String titleChosen = bookTitles.get(new Integer(titleNumber)).get(0);
	String categoryName = categories.get(new Integer(categoryNumber));


	System.out.print(titleChosen + "   "+  categoryName);
	queryText = "SELECT * " +
			"FROM yrb_book " +
			"WHERE title = ? and cat = ?";



	// Prepare the query.
	try {
		querySt = conDB.prepareStatement(queryText);
	} catch (SQLException e) {
		System.out.println("SQL#4 failed in prepare");
		System.out.println(e.toString());
		System.exit(0);
	}

	// Execute the query.
	try {
		querySt.setString(1, titleChosen);
		querySt.setString(2, categoryName);
		answers = querySt.executeQuery();
	} catch (SQLException e) {

		System.out.println("SQL#4 failed in execute");
		System.out.println(e.toString());
		System.exit(0);
	}

	// Any answer?
	try {
```

```
            String title;
            Integer weight;
            String language;
            Integer year;
            for (int i = 1; answers.next(); i++) {
                  title = answers.getString("title");
                  year = answers.getInt("year");
                  language = answers.getString("language");
                  weight = answers.getInt("weight");
                  bookInformation.put(i, new ArrayList < String >
(Arrays.asList(title,
                              Integer.toString(year), language,
Integer.toString(weight))));

                  inDB = true;

            }

      } catch (SQLException e) {
            System.out.println("SQL#4 failed in cursor.");
            System.out.println(e.toString());
            System.exit(0);
      }



      //Now let's try printing the book information
      try{
            String title;
            Integer weight;
            String language;
            Integer year;
            System.out.println("The book that you chose:");
            for (int i = 1; i<bookInformation.size()+1;i++){
                  title = bookInformation.get(i).get(0);
                  year = Integer.parseInt(bookInformation.get(i).get(1));
                  language = bookInformation.get(i).get(2);
                  weight = Integer.parseInt((bookInformation.get(i).get(3)));

                  System.out.println(i + " - " + title +", " + year +", " +
                              language + "," + weight + ";");
                  inDB = true;

                  bookTitle = title;
                  bookYear = year;

            }
      }catch(NullPointerException e){
            System.out.println("Something went wrong with fetching" +
                        "the available book titles");

      }
      // Close the cursor.
      try {
            answers.close();
      } catch (SQLException e) {
```

```
            System.out.print("SQL#4 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#4 failed closing the handle.\n");
            System.out.println(e.toString());
            System.exit(0);
      }
      return inDB;
}


//Method to find the minimum price in for the selected book
private boolean find_minPrice(int customerID, String title, int year){
      String            queryText = "";      // The SQL text.
      PreparedStatement querySt   = null;    // The query handle.
      ResultSet         answers   = null;    // A cursor.

      boolean           inDB      = false;  // Return.


      System.out.println(bookTitle + "  " + bookYear);
      queryText =
                  "SELECT min(price)       "
                              + "FROM yrb_offer O, yrb_member M "
                              + "WHERE M.cid = ? AND M.club = O.club AND
O.title = ? AND O.year = ?";

      // 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, customerID);
            querySt.setString(2, title);
            querySt.setInt(3, year);
            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 = true;
                  minPrice = answers.getFloat(1);
                  System.out.println("Your minimum price is for this book is
"
                              + minPrice);

            } 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;



}


//Helper method needed to fetch Customer's club
private boolean find_club(int customerID, float price){
      String            queryText = "";      // The SQL text.
      PreparedStatement querySt   = null;    // The query handle.
      ResultSet         answers   = null;    // A cursor.

      BigDecimal minPrice = new BigDecimal(Float.toString(price));

      boolean           inDB      = false;  // Return.


      queryText =
                  "SELECT O.club       "
                              + "FROM yrb_offer O, yrb_member M "
                              + "WHERE M.cid = ? AND M.club = O.club 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, customerID);
		querySt.setBigDecimal(2, minPrice);
		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 = true;
			clubName = answers.getString("club");

			System.out.println("Your minimum price is for this book is
"
					+ minPrice);

		} 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;


}


//Final method needed to insert the purchase into the list of purchases byt
the customer
public boolean 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.
	boolean inDB = false;

	Timestamp currentTime = new Timestamp(System.currentTimeMillis());
	DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-HH.mm.ss");
	String stamp = dateFormat.format(currentTime);

	queryText = "INSERT INTO yrb_purchase values (?,?,?,?,?,?) ";
	try {
		querySt = conDB.prepareStatement(queryText);
	} catch (SQLException e) {
		System.out.println("SQL#6 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, stamp);
		querySt.setInt(6, quantity);
		querySt.executeUpdate();
		System.out.println("Purchase inserted!");
		inDB = true;
	} 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);
	}
	return inDB;
```

```
}
```

```
}
```