EECS 3421 Project 2

Java API for YRB_Bookstore DB

Kenneth Faria
213 846 597
April 2017

Coding Methodology

Introduction

The API was written in Java using the JDBC driver for IBM's DB2 database. The database used was the provided YRB_Bookstore data base which was comprised of various tables which described student-club member relationships, and book-club memberships. The following queries have been written:

1. Customer ID (CID) lookup
2. Category (cat) lookup
3. Book (bookTitle) lookup
4. Minimum book price lookup (for the clubs that the customer is a part of)
5. Update query to update the YRB_PURCHASE with any purchases a customer may make

Data Structures

Instead of coding the entire program in a main program, a separate "dbInit" class was created to modularize the program to allow room for future scaling. Additionally, data structures were used wherever it was sensible to do so. Using data structures allowed to use of a comparable method which cut down the amount of coding required for certain methods such as find_books where multiple books may be returned and min_price where the lowest price for a book and club member is listed.

Error Handling

Although the command line interface was used to reduce the amount of user errors as possible, errors in input are still possible. Reasonable errors were dealt with, unreasonable errors may require a program restart. Examples of error handling will be included in the relevant sections below.

Customer Lookup

The following code looks up the user within the YRB_CUSTOMER database.

```
public String find_customer(int CID) throws SQLException {
    sql = "SELECT cid AS #cid, name AS #name, city AS #city FROM YRB_CUSTOMER WHERE cid = " + CID;
    PreparedStatement queryBody = null;
    queryBody = conDB.prepareStatement(sql);
    ResultSet res = null;
    String response ="";
        res = queryBody.executeQuery();

    try {
        res.next();
        response = res.getString( columnLabel: "#cid") + "\t" + res.getString( columnLabel: "#name") +
                "\t"  + res.getString( columnLabel: "#city");
    } catch (SQLException e) {
        System.out.println("Sorry user not found! Please enter another CID.");
        response = null;
    }
    return response;
}
```

Terminal Output

```
Enter CID: 12
12      Fanny Mae       Roanoke
```

Error Handling

Since there are only a limited number of customers in the YRB_CUSTOMER table, valid integers where checked against the the users in the database before continuing.

```
Enter CID: -1
Sorry user not found! Please enter another CID.
46
Sorry user not found! Please enter another CID.
45
45      Jack Daniels    Charlottesville
```

Users are warned if they enter non integer values and will have to re-run the program.

Category Lookup

The category query searches the YRB_CATEGORY tables for categories and places the returned values into an ArrayList.

```
public class dbInit {

    private String userInformation;
    private Connection conDB;
    private ArrayList<String> cat = new ArrayList<String>();
    private ArrayList<HashMap> book_test = new ArrayList<HashMap>();
    private TreeMap<Double, String> priceLookup = new TreeMap<>();
    private String sql;

    public dbInit() throws ClassNotFoundException, SQLException, IllegalAccessException, InstantiationException {...}

    public String find_customer(int CID) throws SQLException {...}

    public ArrayList<String> fetch_categories() throws SQLException {

        sql = "SELECT cat AS #cat FROM YRB_CATEGORY";
        PreparedStatement queryBody = null;
        queryBody = conDB.prepareStatement(sql);
        ResultSet res = null;
        int i = 0;
        res = queryBody.executeQuery();

        try{
            while(res.next()){
                cat.add(res.getString( columnLabel: "#cat"));
            }

        } catch(SQLException e) {
            System.out.println("There was an error with the category query");
        }
        return cat;
    }
```

Terminal Output

```
Select a category number to search a title through:
0: children
1: cooking
2: drama
3: guide
4: history
5: horror
6: humor
7: mystery
8: phil
9: romance
10: science
11: travel
```

Error Handling

If a number other than the above is entered, the program will prompt the user to enter another value.

```
Select a category number to search a title through:
0: children
1: cooking
2: drama
3: guide
4: history
5: horror
6: humor
7: mystery
8: phil
9: romance
10: science
11: travel
-1
Sorry, invalid category input, please try again
12
Sorry, invalid category input, please try again
1
The category 'cooking' was selected
```

Book Lookup

This method searches the YRB_Book table with an entered book title and previously entered category. The returned object is of type ArrayList<HashMap> for many reasons. First, since ArrayLists are dynamic data structures in case the query returns multiple books. Second, the ArrayLists accept HashMaps since the book query returns vital information useful later on in the program. HashMaps store data in <Key, Value> pairs which makes storing, Book title, year, language and weight simple.

```
public class dbInit {

    private String userInformation;
    private Connection conDB;
    private ArrayList<String> cat = new ArrayList<String>();
    private ArrayList<HashMap> book_test = new ArrayList<HashMap>();
    private TreeMap<Double, String> priceLookup = new TreeMap<>();
    private String sql;

    public dbInit() throws ClassNotFoundException, SQLException, IllegalAccessException, InstantiationException {...}

    public String find_customer(int CID) throws SQLException {...}

    public ArrayList<String> fetch_categories() throws SQLException {...}

    public ArrayList<HashMap> find_book(String bookTitle, String category) throws SQLException {

        bookTitle = bookTitle.replaceAll("'", "''");

        sql = "SELECT title AS #title, year AS #year, language AS #language, weight AS #weight FROM YRB_BOOK"
                + " WHERE cat = '" + category
                + "' AND title = '" + bookTitle
                + "'";
        PreparedStatement queryBody = conDB.prepareStatement(sql);
        ResultSet res = null;
        res = queryBody.executeQuery();

        try{
            boolean test = res.next();
            if(!test){
                System.out.println("Book does not exist within this category, please enter a new book title");
                bookTitle = new Scanner(System.in).nextLine();
                find_book(bookTitle, category);
            }

            while(test){
                HashMap bookInfo = new HashMap();
                bookInfo.put("Title", res.getString( columnLabel: "#title"));
                bookInfo.put("Year", res.getString( columnLabel: "#year"));
                bookInfo.put("Language", res.getString( columnLabel: "#language"));
                bookInfo.put("Weight", res.getString( columnLabel: "#weight") );
                book_test.add(bookInfo);
                test = res.next();
            }

        } catch(SQLException e){
            System.out.printf("There was an error with the find book query");
        }
        return book_test;
    }
```

Terminal Output

```
Please enter a book title for a query
Rabbits are nice
1: Year: 2000   Language: English       Title: Rabbits are nice Weight: 186
```

Error Handling

SQL requires single apostrophes to be replaced with double apostrophes:

```
public ArrayList<HashMap> find_book(String bookTitle, String category) throws SQLException {
    bookTitle = bookTitle.replaceAll("'", "''");
```

If the book cannot be found within the table, a message will be displayed prompting the user to re-enter the book title. In case the user enters the wrong title, minimal work is required to enter the correct title.

```
Please enter a book title for a query
Rabits are nice
Book does not exist within this category, please enter a new book title
Rabbits are nice
1: Year: 2000   Language: English       Title: Rabbits are nice Weight: 186
```

If the user enters the wrong number from above, an error is thrown and the user is prompted to enter the number properly.

```
Please enter a book title for a query
Rabbits are nice
1: Year: 2000   Language: English       Title: Rabbits are nice Weight: 186

Select book to purchase by choosing the line number above.
0
Sorry, invalid book input, please try again
-1
Sorry, invalid book input, please try again
1
The cheapest book available costs: $18.95
Would you like to purchase the book? [Y/N]:
```

Once the user enters the correct value, the program will advance onwards to the next part of the program

Minimum Price Lookup

This method uses another data structure, TreeMap which is a structure that inserts Key (price of the book) + Value (club it belongs to) in natural order, meaning the first element will have the cheapest price. This makes finding the cheapest book as simple as returning the first element in the TreeMap, as is done below. Since “values” in <Key, Value> pairings cannot be obtained without the key, and since the key contains valuable information as well, the first element of the TreeMap is returned as a Map.Entry object which allows the client code to iterate through the key value pair through simple accessor methods.

```
public class dbInit {

    private String userInformation;
    private Connection conDB;
    private ArrayList<String> cat = new ArrayList<String>();
    private ArrayList<HashMap> book_test = new ArrayList<HashMap>();
    private TreeMap<Double, String> priceLookup = new TreeMap<>();
    private String sql;

    public dbInit() throws ClassNotFoundException, SQLException, IllegalAccessException, InstantiationException {...}

    public String find_customer(int CID) throws SQLException {...}

    public ArrayList<String> fetch_categories() throws SQLException {...}

    public ArrayList<HashMap> find_book(String bookTitle, String category) throws SQLException {...}

    public Map.Entry<Double, String> min_price(int CID, String bookTitle, String category, int year) throws SQLException
        bookTitle = bookTitle.replaceAll("'", "''");
        ArrayList<String> cid_club = find_club(CID);
        for(int i = 0 ; i < cid_club.size(); i++){
            sql = "SELECT * FROM YRB_OFFER" +
                    " WHERE title='" + bookTitle +"'" +
                    " AND club='" + cid_club.get(i) + "'" +
                    " AND year=" + year;
            PreparedStatement queryBody = conDB.prepareStatement(sql);
            ResultSet res = queryBody.executeQuery();

            try {
                while(res.next()){
                    priceLookup.put(res.getDouble( columnLabel: "price"), res.getString( columnLabel: "club"));
                }
            } catch (SQLException e){
                System.out.println("There was a problem with the min_price query");
            }
        }
    return priceLookup.firstEntry();
    }
```

Terminal Output

```
Select book to purchase by choosing the line number above.
1
The cheapest book available costs: $18.95
Would you like to purchase the book? [Y/N]:
```

Error Handling

In case the user enters the wrong letter, they will be prompted to re-enter the suggest values.

```
The cheapest book available costs: $18.95
Would you like to purchase the book? [Y/N]: A
Inalvid entry. Would you like to purchase the book? [Y/N]:
s
Inalvid entry. Would you like to purchase the book? [Y/N]:
Y
```

The user is then prompted to try enter the number of books they would like to purchase. More error handling is done here for non-integer values less than 0:

```
Would you like to purchase the book? [Y/N]: Y
How many books do you want?: -1
Sorry, invalid input, please try again enter a positive integer
1
This will cost you $18.95

Are you okay with this? [Y/N]:
```

More error handling exists to ensure the user enters the above options:

```
Are you okay with this? [Y/N]: S
Invalid entry.
Would you like to go ahead with the purchase [Y/N]:
Y
```

Update Query

The update query takes in arguments and performs a “purchase” for the customer ID. This is done by performing and inserting the values found below into the YRB_PURCHASE table.

```
public class dbInit {

    private String userInformation;
    private Connection conDB;
    private ArrayList<String> cat = new ArrayList<String>();
    private ArrayList<HashMap> book_test = new ArrayList<HashMap>();
    private TreeMap<Double, String> priceLookup = new TreeMap<>();
    private String sql;

    public dbInit() throws ClassNotFoundException, SQLException, IllegalAccessException, InstantiationException {...}

    public String find_customer(int CID) throws SQLException {...}

    public ArrayList<String> fetch_categories() throws SQLException {...}

    public ArrayList<HashMap> find_book(String bookTitle, String category) throws SQLException {...}

    public Map.Entry<Double, String> min_price(int CID, String bookTitle, String category, int year) throws SQLException

    public void insert_purchase(int CID, String club, String bookTitle, int year, int qnty) throws SQLException {
        bookTitle = bookTitle.replaceAll("'", "''");
        String date = String.valueOf(new Timestamp(System.currentTimeMillis()));
        sql = "INSERT INTO YRB_PURCHASE VALUES(" +
                "'" + CID + "', " +
                "'" + club + "', " +
                "'" + bookTitle + "', " +
                "'" + year + "', " +
                "'" + date + "', " +
                "'" + qnty + "'" +
                ")";
        Statement update = conDB.createStatement();
        try {
            update.executeUpdate(sql);
            System.out.println("Purchase successful! ");

        } catch (SQLException e) {
            System.out.println("There was an error making the purchase");
            e.printStackTrace();
        }
    }
```

Terminal Output

At this point of the program, there is simply a message that prints on a successful print.

```
Would you like to go ahead with the purchase [Y/N]:
Y
Purchase successful!
```

Error Handling

There isn't much error handling required for this method since all of the passed in values have been previously handled in other methods. The only code that is handled further handled is the passed in book title to watch for " ' " in book titles that may have them. This is indicated by the highlighted line below:

```
public void insert_purchase(int CID, String club, String bookTitle, int year, int qnty) throws SQLException {
    bookTitle = bookTitle.replaceAll("'", "''");
```

This replaces the single apostrophe with two apostrophes.

Source Code

The source code is found on the following page:

File - /Users/kendotron/IdeaProjects/EECS3421_Project2/src/com/company/Main.java

```java
import java.io.IOException;
import java.sql.*;
import java.util.*;

/**
 * Start up text:
 * cd Documents/EECS3421/Project2/EECS3421_Project2/
 * source ~db2leduc/cshrc.runtime
 * scp -r com/ kfaria@red.eecs.yorku.ca:/eecs/home/kfaria/Documents/EECS3421/Project2/EECS3421_Project2/src
 *
 * Created by Kenneth Faria - 213846597 on 2017-03-23.
 *
 *
 */

public class Main {

    public static void main(String[] args) throws ClassNotFoundException, SQLException, IllegalAccessException, InstantiationException, IOException {

        //Main vars
        ArrayList<String> categories = null;
        ArrayList<HashMap> book = null;
        int userCatInput;
        String userCat = "";
        String userTitleInput = "";
        String bookTitle;
        int bookYear;
        int purchaseQnty = 1;
        Map.Entry minimumBookPriceSearch = null;
        double minBookPrice = 0;
        String minBookClub = null;
        Scanner sc = new Scanner(System.in);
        dbInit yrb_bookstore = new dbInit();
        int bookPurchaseInput = 0;
        int CID;


        //CID Lookup
        System.out.print("Enter CID (Integers Only): ");
        CID = sc.nextInt();
        String user = yrb_bookstore.find_customer(CID);

        while(user == null){
            CID = sc.nextInt();
            user = yrb_bookstore.find_customer(CID);
        }
        System.out.println(user);


        //Category print + selection
        if(user!= null){

           categories = yrb_bookstore.fetch_categories();
            System.out.println("\nSelect a category number to search a title through:");
            Iterator catIt = categories.iterator();
            for(int i = 0 ; catIt.hasNext() ; i++){
                System.out.printf("%d: %s\n", i, catIt.next());
```

```
58              }
59
60              userCatInput = sc.nextInt();
61
62              while(userCatInput < 0 || userCatInput >11){
63                  System.out.println("Sorry, invalid category input,
    please try again");
64                  userCatInput = sc.nextInt();
65              }
66
67              switch(userCatInput) {
68                  case 0 : userCat = "children";
69                      break;
70                  case 1 : userCat = "cooking";
71                      break;
72                  case 2 : userCat = "drama";
73                      break;
74                  case 3 : userCat = "guide";
75                      break;
76                  case 4 : userCat = "history";
77                      break;
78                  case 5 : userCat = "horror";
79                      break;
80                  case 6: userCat = "humor";
81                      break;
82                  case 7 : userCat = "mystery";
83                      break;
84                  case 8 : userCat = "phil";
85                      break;
86                  case 9 : userCat = "romance";
87                      break;
88                  case 10 : userCat = "science";
89                      break;
90                  case 11 : userCat = "travel";
91                      break;
92              }
93
94              System.out.printf("The category '%s' was selected \n",
    userCat);
95          }
96
97          //Book query
98          System.out.println("\nPlease enter a book title for a query")
    ;
99
100         while(userTitleInput.isEmpty()){
101             userTitleInput = sc.nextLine();
102         }
103
104         book = yrb_bookstore.find_book(userTitleInput, userCat);
105
106         if (book != null){
107             Iterator bookHMIt = book.iterator();
108             String output = "";
109
110             for(int i = 1 ; bookHMIt.hasNext() ; i++){
111
112                 HashMap hmObj = (HashMap) bookHMIt.next();
113                 Set set = hmObj.entrySet();
114                 Iterator setIt = set.iterator();
115                 while (setIt.hasNext()){
```

```
116                     Map.Entry me = (Map.Entry)setIt.next();
117                     output += me.getKey() + ": " + me.getValue() + "\
    t";
118                 }
119                 output += '\n';
120                 System.out.printf("%d: %s", i, output);
121             }
122             System.out.println("\nSelect book to purchase by choosing
     the line number above.");
123             bookPurchaseInput = sc.nextInt();
124 
125         }
126 
127 
128         while(bookPurchaseInput <=  0 || bookPurchaseInput > book.
    size()){
129             System.out.println("Sorry, invalid book input, please try
     again");
130             bookPurchaseInput = sc.nextInt();
131         }
132 
133         bookTitle = (String)book.get(bookPurchaseInput - 1).get("
    Title");
134         bookYear = Integer.parseInt((String)book.get(
    bookPurchaseInput - 1).get("Year"));
135 
136         //Minimum price of book
137         minimumBookPriceSearch = yrb_bookstore.min_price(CID,
    bookTitle, userCat, bookYear);
138 
139         if(minimumBookPriceSearch != null){
140                 minBookPrice = (double) minimumBookPriceSearch.getKey
    ();
141                 minBookClub = (String) minimumBookPriceSearch.
    getValue();
142         }
143 
144         System.out.printf("The cheapest book available costs: $%.2f\n
    ", minBookPrice);
145 
146         String decision = null;
147         boolean flag = false;
148         System.out.print("Would you like to purchase the book? [Y/N
    ]: ");
149         decision = sc.next();
150 
151         while (!(decision.equals("Y") | decision.equals("y") |
    decision.equals("N") | decision.equals("n"))){
152             System.out.println("Inalvid entry. Would you like to
    purchase the book? [Y/N]: ");
153             decision = sc.next();
154         }
155 
156         if ((decision.equals("Y") | decision.equals("y"))){
157             System.out.print("How many books do you want?: ");
158             purchaseQnty = sc.nextInt();
159 
160             while(purchaseQnty < 0 ){
161                 System.out.println("Sorry, invalid input, please try
    again enter a positive integer");
162                 purchaseQnty = sc.nextInt();
```

```
163             }
164
165             System.out.printf("This will cost you $%.2f\n", (
    minBookPrice * purchaseQnty));
166             System.out.println();
167
168         } else if ((decision.equals("N") | decision.equals("n"))){
169             System.out.println("Ok goodbye.");
170             System.exit(0);
171         }
172
173         System.out.print("Are you okay with this? [Y/N]: ");
174         decision = sc.next();
175
176         while (!(decision.equals("Y") | decision.equals("y") |
    decision.equals("N") | decision.equals("n"))){
177             System.out.println("Invalid entry.");
178             System.out.println("Would you like to go ahead with the
    purchase [Y/N]: ");
179             decision = sc.next();
180         }
181
182         if ((decision.equals("Y") | decision.equals("y"))){
183             yrb_bookstore.insert_purchase(CID, minBookClub, bookTitle
    , bookYear, purchaseQnty);
184         } else if ((decision.equals("N") | decision.equals("n"))){
185             System.out.println("Ok goodbye.");
186             System.exit(0);
187         }
188     }
189 }
```

File - /Users/kendotron/IdeaProjects/EECS3421_Project2/src/com/company/dbInit.java

```java
import java.sql.*;
import java.util.*;


/**
 * Created by Kenneth Faria - 213846597 on 2017-03-23.
 */

public class dbInit {

    private String userInformation;
    private Connection conDB;
    private ArrayList<String> cat = new ArrayList<String>();
    private ArrayList<HashMap> book_test = new ArrayList<HashMap>();
    private TreeMap<Double, String> priceLookup = new TreeMap<>();
    private String sql;

    public dbInit() throws ClassNotFoundException, SQLException, IllegalAccessException, InstantiationException {
        Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();
        String url = "jdbc:db2:c3421m";
        conDB = DriverManager.getConnection(url);
    }

    public String find_customer(int CID) throws SQLException {

        sql = "SELECT cid AS #cid, name AS #name, city AS #city FROM YRB_CUSTOMER WHERE cid = " + CID;
        PreparedStatement queryBody = null;
        queryBody = conDB.prepareStatement(sql);
        ResultSet res = null;
        String response ="";
            res = queryBody.executeQuery();

        try {
            res.next();
            response = res.getString("#cid") + "\t" + res.getString("#name") +
                    "\t"  + res.getString("#city");
        } catch (SQLException e) {
            System.out.println("Sorry user not found! Please enter another CID.");
            response = null;
        }
        return response;
    }

    public ArrayList<String> fetch_categories() throws SQLException {

        sql = "SELECT cat AS #cat FROM YRB_CATEGORY";
        PreparedStatement queryBody = null;
        queryBody = conDB.prepareStatement(sql);
        ResultSet res = null;
        int i = 0;
        res = queryBody.executeQuery();

        try{
            while(res.next()){
                cat.add(res.getString("#cat"));
            }

```

```
58          } catch(SQLException e) {
59              System.out.println("There was an error with the category
    query");
60          }
61          return cat;
62      }
63
64      public ArrayList<HashMap> find_book(String bookTitle, String
    category) throws SQLException {
65          bookTitle = bookTitle.replaceAll("'", "''");
66
67          sql = "SELECT title AS #title, year AS #year, language AS #
    language, weight AS #weight FROM YRB_BOOK"
68                  + " WHERE cat = '" + category
69                  + "' AND title = '" + bookTitle
70                  + "'";
71          PreparedStatement queryBody = conDB.prepareStatement(sql);
72          ResultSet res = null;
73          res = queryBody.executeQuery();
74
75          try{
76              boolean test = res.next();
77              if(!test){
78                  System.out.println("Book does not exist within this
    category, please enter a new book title");
79                  bookTitle = new Scanner(System.in).nextLine();
80                  find_book(bookTitle, category);
81              }
82
83              while(test){
84                  HashMap<String, String> bookInfo = new HashMap<>();
85                  bookInfo.put("Title", res.getString("#title"));
86                  bookInfo.put("Year", res.getString("#year"));
87                  bookInfo.put("Language", res.getString("#language"));
88                  bookInfo.put("Weight", res.getString("#weight") );
89                  book_test.add(bookInfo);
90                  test = res.next();
91              }
92
93          } catch(SQLException e){
94              System.out.printf("There was an error with the find book
    query");
95          }
96          return book_test;
97      }
98
99      public Map.Entry<Double, String> min_price(int CID, String
    bookTitle, String category, int year) throws SQLException {
100         bookTitle = bookTitle.replaceAll("'", "''");
101         ArrayList<String> cid_club = find_club(CID);
102         for(int i = 0 ; i < cid_club.size(); i++){
103             sql = "SELECT * FROM YRB_OFFER" +
104                     " WHERE title='" + bookTitle +"'" +
105                     " AND club='" + cid_club.get(i) + "'" +
106                     " AND year=" + year;
107             PreparedStatement queryBody = conDB.prepareStatement(sql)
    ;
108             ResultSet res = queryBody.executeQuery();
109
110             try {
111                 while(res.next()){
```

```
112                     priceLookup.put(res.getDouble("price"), res.
    getString("club"));
113                 }
114             } catch (SQLException e){
115                 System.out.println("There was a problem with the
    min_price query");
116             }
117         }
118     return priceLookup.firstEntry();
119     }
120
121     public void insert_purchase(int CID, String club, String
    bookTitle, int year, int qnty) throws SQLException {
122         bookTitle = bookTitle.replaceAll("'", "''");
123         String date = String.valueOf(new Timestamp(System.
    currentTimeMillis()));
124         sql = "INSERT INTO YRB_PURCHASE VALUES(" +
125                 "'" + CID + "', " +
126                 "'" + club + "', " +
127                 "'" + bookTitle + "', " +
128                 "'" + year + "', " +
129                 "'" + date + "', " +
130                 "'" + qnty + "'" +
131                 ")";
132         Statement update = conDB.createStatement();
133         try {
134             update.executeUpdate(sql);
135             System.out.println("Purchase successful! ");
136
137         } catch (SQLException e) {
138             System.out.println("There was an error making the
    purchase");
139             e.printStackTrace();
140         }
141     }
142
143     private ArrayList<String> find_club(int CID) throws SQLException
    {
144
145         ArrayList<String> cid_club = new ArrayList<String>();
146
147         sql = "SELECT club AS #club FROM YRB_MEMBER where cid = " +
    CID;
148         PreparedStatement queryBody = null;
149         queryBody = conDB.prepareStatement(sql);
150         ResultSet res = null;
151         res = queryBody.executeQuery();
152
153         try {
154             while(res.next()){
155                 cid_club.add(res.getString("#club"));
156             }
157
158         } catch (SQLException e){
159             System.out.println("There was an error with the find club
     query");
160         }
161         return cid_club;
162     }
163
164 }
```