

CECS323JDBCProject.java

```

1
2 package cecs323.jdbcproject;
3
4 import java.sql.*;
13
14 /**
15  * <h1>CECS323JDBCProject</h1> This is program is designed to be operate in
16  * conjunction with a database of books, publisher, and writing group.
17  *
18  * @author Sotheanith Sok
19  * @version 1.5
20  * @since 03-16-2017
21  */
22 public class CECS323JDBCProject {
23
24     public static void main(String[] args) {
25         // TODO code application logic here
26         Scanner in = new Scanner(System.in);
27         // Input data if required
28         String DBNAME = "JDBCProjectDatabase";
29         String USER = "IAmNotARobot";
30         String PASS = "IAmNotARobot";
31
32         // Database URL
33         String DB_URL = "jdbc:derby://localhost:1527/" + DBNAME + ";user=" + USER +
";password=" + PASS;
34         Connection conn = null;
35         // Statement stmt=null;
36         boolean done = false;
37         try {
38             // Register JDBC driver
39             Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();
40             // Open Connection
41             System.out.println("Connecting to database...");
42             conn = DriverManager.getConnection(DB_URL);
43
44             // Create Database object
45             DatabaseOperations d = OpsImplFactory.getOperationsImpl(conn);
46             do {
47                 int choice = menu(in);
48                 switch (choice) {
49                     case 1:
50                         System.out.println("-Listing All Writing Groups-");
51                         listAllWritingGroups(d);
52                         break;
53                     case 2:
54                         System.out.println("-Listing all data for a writing group-");
55                         listDataForAWritingGroup(d, in);
56                         break;
57                     case 3:
58                         System.out.println("-Listing all publishers-");
59                         listAllPublisher(d);
60                         break;
61                     case 4:
62                         System.out.println("-Listing all the data for publisher-");
63                         printDataForAPublisher(d, in);
64                         break;

```

```

65         case 5:
66             System.out.println("-Listing all book titles-");
67             listAllBookTitle(d);
68             break;
69         case 6:
70             System.out.println("-Listing all the data for a book-");
71             listDataForABook(d, in);
72             break;
73         case 7:
74             System.out.println("-Inserting a new book-");
75             insertABook(d, in);
76             break;
77         case 8:
78             System.out.println("-Inserting a new publisher-");
79             insertAPublisher(d, in);
80             break;
81         case 9:
82             System.out.println("-Removing a book-");
83             removeABook(d, in);
84             break;
85         case 10:
86             System.out.println("-Exiting-");
87             done = true;
88             break;
89         default:
90             System.out.println("-Invalid Input-");
91             break;
92     }
93     } while (!done);
94     // Close resource when there isn't any error.
95     conn.close();
96 } catch (SQLException se) {
97     System.out.println("ERROR: Connection to Database failed!!!");
98 } catch (Exception e) {
99     // Testing for unexpected exception.
100    System.out.println("Unknown Exception was threw to main");
101    System.out.println(e);
102 } finally {
103     // Error caused by closing resources.
104     try {
105         conn.close();
106         in.close();
107     } catch (SQLException se) {
108         System.out.println("H3");
109         se.printStackTrace();
110         // This error is mostly caused by wrong database info.
111     } catch (NullPointerException np) {
112         System.out.println("ERROR: Database related informations are incorrect.");
113     }
114 }
115 }
116
117 /**
118  * Print menu and get user input
119  *
120  * @param in Scanner for keyboard
121  * @return valid user choice

```

```

122     */
123     public static int menu(Scanner in) {
124         boolean done = false;
125         int choice = 0;
126         System.out.println("--Menu--");
127         System.out.println("1. List all writing groups ");
128         System.out.println("2. List all data for a writing group");
129         System.out.println("3. List all publishers");
130         System.out.println("4. List all the data for a publisher");
131         System.out.println("5. List all book titles");
132         System.out.println("6. List all the data for a book");
133         System.out.println("7. Insert a new book");
134         System.out.println("8. Insert a new publisher");
135         System.out.println("9. Remove a book");
136         System.out.println("10. Exit");
137         while (!done) {
138             try {
139                 System.out.print("Enter: ");
140                 choice = in.nextInt();
141                 if (!(choice >= 1 && choice <= 10)) {
142                     throw new NumberFormatException();
143                 }
144                 done = true;
145             } catch (InputMismatchException ime) {
146                 in.next();
147                 System.out.print("Invalid Input. Re-enter: ");
148             } catch (NumberFormatException nfe) {
149                 System.out.print("Invalid Input. Re-enter: ");
150             }
151         }
152         return choice;
153     }
154
155     /**
156     * List all data related to all writing groups
157     *
158     * @param w WritingGroupOperations object.
159     * @throws SQLException
160     */
161     public static void listAllWritingGroups(DatabaseOperations w) {
162         try {
163             List<WritingGroup> list = w.listWritingGroups();
164             // Check if WritingGroups is empty.
165             if (list.size() == 0) {
166                 throw new SQLException();
167             }
168             // Print WritingGroups.
169             System.out.printf("%-20s%-20s%-20s%-20s\n", "GroupName", "HeadWriter",
"YearFormed", "Subject");
170             for (int i = 0; i < list.size(); i++) {
171                 System.out.printf("%-20s%-20s%-20s%-20s\n", list.get(i).groupName,
list.get(i).headWriter,
list.get(i).subject, list.get(i).yearFormed);
172             }
173         } catch (SQLException s) {
174             System.out.println("ERROR: WritingGroups is empty.");
175         }
176     }

```

```

177     }
178
179     /**
180     * List all data for a specific WritingGroup (4)
181     *
182     * @param w WritingGroupOperations object
183     * @param in Scanner for keyboard
184     * @throws SQLException
185     */
186     public static void listDataForAWritingGroup(DatabaseOperations w, Scanner in) {
187         try {
188             List<String> list = w.listWritingGroupNames();
189             // Check if WritingGroups is empty.
190             if (list.size() == 0) {
191                 throw new SQLException();
192             }
193             // Print available WritingGroups.
194             System.out.println("-Available Group-");
195             for (int i = 0; i < list.size(); i++) {
196                 System.out.println(list.get(i));
197             }
198             // Get input.
199             System.out.print("Enter group name: ");
200             String groupName = in.next();
201             WritingGroup k = w.getWritingGroup(groupName);
202             // Print result
203             if (k.groupName != null) {
204                 System.out.printf("%-20s%-20s%-20s%-20s\n", "GroupName", "HeadWriter",
205 "YearFormed", "Subject");
206                 System.out.printf("%-20s%-20s%-20s%-20s\n", k.groupName, k.headWriter,
207 k.yearFormed, k.subject);
208             } catch (SQLException s) {
209                 System.out.println("ERROR: WritingGroups is empty.");
210             } catch (NullPointerException np) {
211                 System.out.println("ERROR: WritingGroup was not found.");
212             }
213         }
214
215         /**
216         * List information related to all publishers (4)
217         *
218         * @param p PublisherOperations object
219         * @throws SQLException
220         */
221         public static void listAllPublisher(DatabaseOperations p) {
222             try {
223                 List<Publisher> list = p.listPublishers();
224                 // Check if publishers is empty.
225                 if (list.size() == 0) {
226                     throw new SQLException();
227                 }
228                 // Print result.
229                 System.out.printf("%-20s%-30s%-20s%-20s\n", "PublisherName", "PublisherAddress",
230 "PublisherPhone",
231 "PublisherEmail");
232                 for (int i = 0; i < list.size(); i++) {

```

```

231         System.out.printf("%-20s%-30s%-20s%-20s\n", list.get(i).publisherName,
list.get(i).publisherAddress,
232             list.get(i).publisherPhone, list.get(i).publisherEmail);
233     }
234 } catch (SQLException s) {
235     System.out.println("ERROR: Publishers is empty.");
236 }
237 }
238
239 /**
240  * Print all data for a publisher
241  *
242  * @param d DatabaseOperations object
243  */
244 public static void printDataForAPublisher(DatabaseOperations d, Scanner in) {
245     try {
246         List<String> list = d.listPublisherNames();
247         // Check if Publishers is empty.
248         if (list.size() == 0) {
249             throw new SQLException();
250         }
251         // Print available publishers
252         System.out.println("-Available Publishers-");
253         for (int i = 0; i < list.size(); i++) {
254             System.out.println(list.get(i));
255         }
256         // Get input
257         System.out.print("Enter publisher name: ");
258         in.nextLine();
259         String pubName = in.nextLine();
260         Publisher p = d.getPublisher(pubName);
261         // Print result
262         if (p.publisherName != null) {
263             System.out.printf("%-20s%-30s%-20s%-20s\n", "PublisherName",
"PublisherAddress", "PublisherPhone",
264                 "PublisherEmail");
265             System.out.printf("%-20s%-30s%-20s%-20s\n", p.publisherName,
p.publisherAddress, p.publisherPhone,
266                 p.publisherEmail);
267         }
268     } catch (SQLException e) {
269         System.out.println("ERROR: Publishers is empty.");
270     } catch (NullPointerException np) {
271         System.out.println("ERROR: Publisher was not found.");
272     }
273 }
274
275 /**
276  * List all the book title
277  *
278  * @param b
279  * @throws SQLException
280  */
281 public static void listAllBookTitle(DatabaseOperations b) {
282     try {
283         List<String> list = b.listBookTitles();
284         // Check if Books is empty.

```

```

285         if (list.size() == 0) {
286             throw new SQLException();
287         }
288         // Print result.
289         System.out.printf("%-10s", "BookTitle\n");
290         for (int i = 0; i < list.size(); i++) {
291             System.out.printf("%-10s\n", list.get(i));
292         }
293     } catch (SQLException s) {
294         System.out.println("ERROR: Books is empty.");
295     }
296 }
297
298 /**
299  * List all data for a specific book including related writing group and
300  * publisher.
301  *
302  * @param b BookOperations object
303  * @param in Scanner for keyboard
304  * @throws SQLException
305  */
306 public static void listDataForABook(DatabaseOperations b, Scanner in) {
307     try {
308         // Check if Books is empty.
309         if (b.listBookTitles().size() == 0) {
310             throw new SQLException();
311         }
312         // Print available Publishers and WritingGroups.
313         printAvaialbeBooks(b);
314         // Get input.
315         System.out.print("Enter BookTitle:");
316         in.nextLine();
317         String bookTitle = in.nextLine();
318         System.out.print("Enter groupName: ");
319         String groupName = in.nextLine();
320         Book book = b.getBook(new BookKeyData(bookTitle, groupName));
321         BookDetail bookDetail = b.getBookDetails(new BookKeyData(bookTitle, groupName));
322         // Print result.
323         if (book.groupName != null) {
324             System.out.printf("%-40s%-20s%-20s%-20s%-20s%-20s%-20s%-30s%-30s%-20s\n",
"BookTitle",
325                                     "YearPublished", "NumberPages", "GroupName", "HeadWriter",
"YearFormed", "Subject",
326                                     "PublisherName", "PublisherAddress", "PublisherPhone",
"PublisherEmail");
327             System.out.printf("%-40s%-20s%-20s%-20s%-20s%-20s%-20s%-30s%-30s%-20s\n",
book.bookTitle,
328                                     book.yearPublished, book.numberPages,
bookDetail.writingGroup.groupName,
329                                     bookDetail.writingGroup.headWriter,
bookDetail.writingGroup.yearFormed,
330                                     bookDetail.writingGroup.subject, bookDetail.publisher.publisherName,
331                                     bookDetail.publisher.publisherAddress,
bookDetail.publisher.publisherPhone,
332                                     bookDetail.publisher.publisherEmail);
333         }
334     } catch (SQLException s) {

```

```

335         System.out.println("ERROR: Books is empty.");
336     } catch (NullPointerException np) {
337         System.out.println("ERROR: Book was not found.");
338     }
339 }
340
341 /**
342  * Insert a new book into database
343  *
344  * @param d DatabaseOperations object
345  * @param in Scanner for keyboard
346  * @throws SQLIntegrityConstraintViolationException
347  * @throws SQLException
348  */
349 public static void insertABook(DatabaseOperations d, Scanner in) {
350     try {
351         if (d.listWritingGroupNames().size() == 0 || d.listPublisherNames().size() == 0) {
352             throw new SQLException();
353         }
354         // Print out available publisher and group.
355         System.out.println("-Available Publishers-");
356         List<String> s = d.listPublisherNames();
357         for (int i = 0; i < s.size(); i++) {
358             System.out.println(s.get(i));
359         }
360         System.out.println("-Available WritingGroups-");
361         s = d.listWritingGroupNames();
362         for (int i = 0; i < s.size(); i++) {
363             System.out.println(s.get(i));
364         }
365         // Get input.
366         System.out.print("Enter BookTitle: ");
367         in.nextLine();
368         String bookTitle = in.nextLine();
369         System.out.print("Enter YearPublished: ");
370         String yearPublished = in.nextLine();
371         System.out.print("Enter NumberPages: ");
372         int numberPages = in.nextInt();
373         System.out.print("Enter GroupName: ");
374         in.nextLine();
375         String groupName = in.nextLine();
376         System.out.print("Enter PublisherName: ");
377         String publisherName = in.nextLine();
378         // Insert into Books.
379         d.insertBook(new Book(bookTitle, groupName, publisherName, yearPublished,
380             numberPages));
381     } catch (SQLException s) {
382         System.out.println("ERROR: Unable to insert book when publishers or writing groups
383         is empty.");
384     } catch (IllegalArgumentException iae) {
385         System.out.println("ERROR: YearPublished should be integer. Insertion Fail!!!");
386     } catch (InputMismatchException im) {
387         System.out.println("ERROR: NumberPages should be integer. Insertion Fail!!!");
388     }
389 }
390 /**

```

```

390  * Insert and replace old publisher with a new one
391  *
392  * @param d DatabaseOperations objects
393  * @param in Scanner for keyboard
394  * @throws SQLIntegrityConstraintViolationException
395  * @throws SQLException
396  */
397  public static void insertAPublisher(DatabaseOperations d, Scanner in) {
398      try {
399          // Check if Publishers is empty.
400          if (d.listPublisherNames().size() == 0) {
401              throw new SQLException();
402          }
403          // Print available publishers
404          System.out.println("-Avaialbe Publishers-");
405          List<String> s = d.listPublisherNames();
406          for (int i = 0; i < s.size(); i++) {
407              System.out.println(s.get(i));
408          }
409          // Get input.
410          System.out.print("Enter OldPublisherName: ");
411          in.nextLine();
412          String oldPub = in.nextLine();
413          System.out.println("-Get New Publisher Info-");
414          System.out.print("Enter PublisherName: ");
415          String publisherName = in.nextLine();
416          System.out.print("Enter PublisherAddress: ");
417          String publisherAddress = in.nextLine();
418          System.out.print("Enter PublisherPhone: ");
419          String publisherPhone = in.nextLine();
420          System.out.print("Enter PublisherEmail: ");
421          String publisherEmail = in.nextLine();
422          // Check if oldPublisher actually exist.
423          if (d.getPublisher(oldPub).publisherName == null) {
424              // Replace publisher
425              d.insertPublisher(new Publisher(publisherName, publisherAddress, publisherPhone,
426                  publisherEmail));
427              d.replacePublisher(oldPub, publisherName);
428              d.deletePublisher(oldPub);
429          } catch (NullPointerException np) {
430              System.out.println("ERROR: Old publisher was not found. Insertion Fail!!!");
431          } catch (SQLException s) {
432              System.out.println("ERROR: Publisher is empty");
433          }
434      }
435
436  /**
437   * Remove book based on title and group name
438   *
439   * @param d DatabaseOperations object
440   * @param in Scanner for keyboard
441   * @throws SQLException
442   */
443  public static void removeABook(DatabaseOperations d, Scanner in) {
444      try {
445          // Check if Books is empty.

```



```

446         if (d.listBookTitles().size() == 0) {
447             throw new SQLException();
448         }
449         // Print available Publishers and WritingGroups
450         printAvaialbeBooks(d);
451         // Get input
452         System.out.print("Enter BookTitle:");
453         in.nextLine();
454         String bookTitle = in.nextLine();
455         System.out.print("Enter groupName: ");
456         String groupName = in.nextLine();
457         // Check if the book existed.
458         if (d.getBook(bookTitle, groupName).bookTitle == null) {
459             throw new NullPointerException();
460         }
461         // Perform operation
462         d.deleteBook(new BookKeyData(bookTitle, groupName));
463     } catch (SQLException s) {
464         System.out.println("ERROR: Books is empty.");
465     } catch (NullPointerException iie) {
466         System.out.println("ERROR: Book doesn't not existed in the database.");
467     }
468 }
469
470 /**
471  * This function is used to print available bookTitle and groupName
472  * combination.
473  *
474  * @param d DatabaseOperations object
475  * @throws SQLException the exception which will be handle by other
476  *         functions.
477  */
478 public static void printAvaialbeBooks(DatabaseOperations d) throws SQLException {
479     List<Book> list = d.listBooks();
480     System.out.println("-Avaialbe Book-");
481     System.out.printf("%-40s%-10s\n", "BookTitle", "GroupName");
482     for (int i = 0; i < list.size(); i++) {
483         System.out.printf("%-40s%-10s\n", list.get(i).bookTitle, list.get(i).groupName);
484     }
485 }
486 }
487

```

DatabaseOperations.java

```
1 package cecs323.jdbcproject.interconnect;
2
3 import cecs323.jdbcproject.pojos.Book;
11
12 /**
13  * The DatabaseOperations interface defines the interface for a class that can
14  * perform operations on the database.
15  *
16  * @author Nicholas Utz
17  */
18 public interface DatabaseOperations {
19     /**
20      * Returns a {@link List} of Strings, containing the titles of all of the
21      * entries in the Books table.
22      *
23      * @return list of book titles
24      */
25     public List<String> listBookTitles() throws SQLException;
26
27     /**
28      * Returns a {@link List} of {@link Book}s, containing all of the
29      * information in the Books table.
30      *
31      * @return list of all books
32      */
33     public List<Book> listBooks() throws SQLException;
34
35     /**
36      * Returns a {@link Book} storing all of the data pertaining to the book
37      * with the given title, written by the WritingGroup with the given name.
38      *
39      * @param title the title of the book to fetch
40      * @param writingGroup the writing group that wrote the book to fetch
41      * @return book info
42      */
43     public Book getBook(String title, String writingGroup) throws SQLException;
44
45     /**
46      * Returns a {@link Book} storing all of the data pertaining to the book
47      * with the primary key data.
48      *
49      * @param key the key data of the book to fetch
50      * @return book info
51      */
52     public Book getBook(BookKeyData key) throws SQLException;
53
54     /**
55      * Returns a {@link BookDetail} object, containing all available data
56      * pertaining to the book with the given title, written by the given writing
57      * group, including the publisher and writing group.
58      *
59      * @param title the title of the book to fetch
60      * @param writingGroup the writing group name of the book to fetch
61      * @return book details
62      */
63     public BookDetail getBookDetails(String title, String writingGroup) throws SQLException;
64
```

DatabaseOperations.java

```
65  /**
66   * Returns a {@link BookDetail} object, storing all of the data pertaining
67   * to the book with the given primary key data, and the publisher and
68   * writing group of the book.
69   *
70   * @param key the key of the book to fetch
71   * @return book details
72   */
73  public BookDetail getBookDetails(BookKeyData key) throws SQLException;
74
75  /**
76   * Inserts the given book into the books table.
77   *
78   * @param book the book to insert
79   * @throws SQLException if a SQLException occurs while attempting to insert
80   */
81  public void insertBook(Book book) throws SQLIntegrityConstraintViolationException,
82  SQLException;
83
84  /**
85   * Replaces the given old publisher name with a new one, for all books
86   * published by the old publisher.
87   *
88   * @param oldName the name of the publisher to be replaced
89   * @param newName the name of the publisher replacing the old one
90   * @throws SQLException if a SQLException occurs while updating
91   */
92  public void replacePublisher(String oldName, String newName)
93  throws SQLIntegrityConstraintViolationException, SQLException;
94
95  /**
96   * Deletes the book with the given title and writing group from the books
97   * table.
98   *
99   * @param title the title of the book to delete
100  * @param writingGroup the writing group who wrote the book to delete
101  * @throws SQLException if a SQLException occurs while deleting
102  */
103  public void deleteBook(String title, String writingGroup) throws SQLException;
104
105  /**
106   * Deletes the book with the given primary key data from the books table.
107   *
108   * @param key the primary key data of the book to delete
109   * @throws SQLException if a SQL exception occurs while deleting
110   */
111  public void deleteBook(BookKeyData key) throws SQLException;
112
113  /**
114   * Returns a {@link List} of {@link String}s, representing the names of all
115   * of the entries in the Publishers table.
116   *
117   * @return list of publisher names
118   */
119  public List<String> listPublisherNames() throws SQLException;
120  /**
```

DatabaseOperations.java

```
121     * Returns a {@link List} of {@link Publisher}s, representing all of the
122     * data in the Publishers table.
123     *
124     * @return list of publishers
125     */
126     public List<Publisher> listPublishers() throws SQLException;
127
128     /**
129     * Returns a {@link Publisher} object, storing all the data stored for the
130     * publisher with the given name in the Publishers table.
131     *
132     * @param name the name of the publisher to fetch
133     * @return publisher info
134     */
135     public Publisher getPublisher(String name) throws SQLException;
136
137     /**
138     * Inserts a new entry into the Publishers table, using the given
139     * {@link Publisher} as a source of attribute data.
140     *
141     * @param info the info of the publisher to insert
142     * @throws java.sql.SQLException if a SQLException occurs while inserting
143     */
144     public void insertPublisher(Publisher info) throws
SQLIntegrityConstraintViolationException, SQLException;
145
146     /**
147     * Deletes the Publisher with the given name from the publishers table.
148     *
149     * @param name the name of the publisher to delete
150     * @throws SQLIntegrityConstraintViolationException if there is a Book
151     *         dependent on the named publisher
152     * @throws SQLException if there is a problem deleting the publisher
153     */
154     public void deletePublisher(String name) throws SQLIntegrityConstraintViolationException,
SQLException;
155
156     /**
157     * Returns a {@link List} of Strings, containing the names of all of the
158     * WritingGroups in the WritingGroups table.
159     *
160     * @return list of writing groups' names
161     */
162     public List<String> listWritingGroupNames() throws SQLException;
163
164     /**
165     * Returns a {@link List} of {@link WritingGroup}s, representing all of the
166     * data in the WritingGroups table.
167     *
168     * @return list of WritingGroups
169     */
170     public List<WritingGroup> listWritingGroups() throws SQLException;
171
172     /**
173     * Returns a {@link WritingGroup} object containing the data stored in the
174     * WritingGroups table for the WritingGroup with the given name.
175     *
```

DatabaseOperations.java

```
176     * @param name the name of the WritingGroup to fetch
177     * @return writing group info
178     * @throws NullPointerException if there is no entry in the writing groups
179     *         table with the given name.
180     */
181     public WritingGroup getWritingGroup(String name) throws SQLException;
182
183     /**
184     * Inserts a row in the WritingGroups table with the attribute values stored
185     * in the given {@link WritingGroup} object.
186     *
187     * @param group the writing group to insert
188     * @throws SQLException if an exception is thrown while trying to insert
189     */
190     public void insertWritingGroup(WritingGroup group) throws
191     SQLIntegrityConstraintViolationException, SQLException;
192
193     /**
194     * Deletes the writing group with the given name.
195     *
196     * @param groupName the name of the writing group to delete
197     * @throws SQLException if an exception is raised while deleting
198     */
199     public void deleteWritingGroup(String groupName) throws
200     SQLIntegrityConstraintViolationException, SQLException;
```

OpsImplFactory.java

```

1 package impl;
2
3 import cecs323.jdbcproject.interconnect.DatabaseOperations;
17
18 /**
19  * The OpsImplFactory class is a factory for classes that implement the
20  * *Operations interfaces defined in {@link cecs323.jdbcproject.interconnect}.
21  *
22  * @author Nicholas Utz
23  */
24 public class OpsImplFactory {
25
26     public static DatabaseOperations getOperationsImpl(Connection con) throws SQLException {
27         return new OpsImpl(con);
28     }
29 }
30
31 class OpsImpl implements DatabaseOperations {
32
33     private final Connection con;
34
35     private static final String SQL_GET_TITLES = "SELECT booktitle FROM books";
36     private static final String SQL_GET_BOOKS = "SELECT * FROM books";
37     private static final String SQL_GET_BOOK = "SELECT * FROM books WHERE booktitle=? AND
groupname=?";
38     private static final String SQL_INSERT_BOOK = "INSERT INTO books (groupname, booktitle, "
39         + "publishername, yearpublished, numberpages) VALUES (?, ?, ?, ?, ?)";
40     private static final String SQL_UPDATE_PUBLISHERS = "UPDATE books SET publishername=?
WHERE publishername=?";
41     private static final String SQL_DELETE_BOOK = "DELETE FROM books WHERE booktitle=? AND
groupname=?";
42     private static final String SQL_GET_PUBLISHER_NAMES = "SELECT publishername FROM
publishers";
43     private static final String SQL_GET_PUBLISHERS = "SELECT * FROM publishers";
44     private static final String SQL_GET_PUBLISHER = "SELECT * FROM publishers WHERE
publishername=?";
45     private static final String SQL_INSERT_PUBLISHER = "INSERT INTO publishers (publishername,
"
46         + "publisheraddress, publisherphone, publisheremail) VALUES (?, ?, ?, ?)";
47     private static final String SQL_DELETE_PUBLISHER = "DELETE FROM publishers WHERE
publishername=?";
48     private static final String SQL_GET_GROUP_NAMES = "SELECT groupname FROM writinggroups";
49     private static final String SQL_GET_GROUPS = "SELECT * FROM writinggroups";
50     private static final String SQL_GET_WRITING_GROUP = "SELECT * FROM writinggroups WHERE
groupname=?";
51     private static final String SQL_INSERT_WRITING_GROUP = "INSERT INTO writinggroups
(groupname, "
52         + "headwriter, yearformed, subject) VALUES (?, ?, ?, ?)";
53     private static final String SQL_DELETE_WRITING_GROUP = "DELETE FROM writinggroups WHERE
groupname=?";
54
55     private final PreparedStatement PSTMT_GET_BOOK;
56     private final PreparedStatement PSTMT_INSERT_BOOK;
57     private final PreparedStatement PSTMT_UPDATE_PUBLISHERS;
58     private final PreparedStatement PSTMT_DELETE_BOOK;
59     private final PreparedStatement PSTMT_GET_PUBLISHER;
60     private final PreparedStatement PSTMT_INSERT_PUBLISHER;

```

```

61     private final PreparedStatement PSTMT_DELETE_PUBLISHER;
62     private final PreparedStatement PSTMT_GET_WRITING_GROUP;
63     private final PreparedStatement PSTMT_INSERT_WRITING_GROUP;
64     private final PreparedStatement PSTMT_DELETE_WRITING_GROUP;
65
66     public OpsImpl(Connection con) throws SQLException {
67         this.con = con;
68         this.PSTMT_GET_BOOK = con.prepareStatement(SQL_GET_BOOK);
69         this.PSTMT_INSERT_BOOK = con.prepareStatement(SQL_INSERT_BOOK);
70         this.PSTMT_UPDATE_PUBLISHERS = con.prepareStatement(SQL_UPDATE_PUBLISHERS);
71         this.PSTMT_DELETE_BOOK = con.prepareStatement(SQL_DELETE_BOOK);
72         this.PSTMT_GET_PUBLISHER = con.prepareStatement(SQL_GET_PUBLISHER);
73         this.PSTMT_INSERT_PUBLISHER = con.prepareStatement(SQL_INSERT_PUBLISHER);
74         this.PSTMT_DELETE_PUBLISHER = con.prepareStatement(SQL_DELETE_PUBLISHER);
75         this.PSTMT_GET_WRITING_GROUP = con.prepareStatement(SQL_GET_WRITING_GROUP);
76         this.PSTMT_INSERT_WRITING_GROUP = con.prepareStatement(SQL_INSERT_WRITING_GROUP);
77         this.PSTMT_DELETE_WRITING_GROUP = con.prepareStatement(SQL_DELETE_WRITING_GROUP);
78     }
79
80     @Override
81     public List<String> listBookTitles() throws SQLException {
82         Statement stmt = this.con.createStatement();
83         List<String> titles = new LinkedList<>();
84         ResultSet results = stmt.executeQuery(SQL_GET_TITLES);
85
86         while (results.next()) {
87             titles.add(results.getString(1));
88         }
89
90         results.close();
91         stmt.close();
92
93         return titles;
94     }
95
96     @Override
97     public List<Book> listBooks() throws SQLException {
98         Statement stmt = this.con.createStatement();
99         List<Book> books = new LinkedList<>();
100        ResultSet results = stmt.executeQuery(SQL_GET_BOOKS);
101
102        while (results.next()) {
103            books.add(new Book(results.getString(2), results.getString(1),
104                results.getString(3), results.getString(4),
105                results.getInt(5)));
106        }
107
108        results.close();
109        stmt.close();
110
111        return books;
112    }
113
114    @Override
115    public Book getBook(String title, String writingGroup) throws SQLException {
116        this.PSTMT_GET_BOOK.setString(1, title);
117        this.PSTMT_GET_BOOK.setString(2, writingGroup);

```

```

117     ResultSet results = this.PSTMT_GET_BOOK.executeQuery();
118
119     if (results.next()) {
120         Book book = new Book(results.getString(2), results.getString(1),
121             results.getString(3), results.getString(4),
122                 results.getInt(5));
123         results.close();
124         return book;
125     } else {
126         results.close();
127         return null;
128     }
129 }
130
131 @Override
132 public Book getBook(BookKeyData key) throws SQLException {
133     return getBook(key.bookTitle, key.writingGroup);
134 }
135
136 @Override
137 public BookDetail getBookDetails(String title, String writingGroup) throws SQLException {
138     Book book = getBook(title, writingGroup);
139
140     if (book == null) {
141         return null;
142     }
143
144     Publisher pub = getPublisher(book.publisherName);
145     WritingGroup wg = getWritingGroup(writingGroup);
146
147     if (pub == null || wg == null) {
148         return null;
149     }
150
151     return new BookDetail(book, pub, wg);
152 }
153
154 @Override
155 public BookDetail getBookDetails(BookKeyData key) throws SQLException {
156     return getBookDetails(key.bookTitle, key.writingGroup);
157 }
158
159 @Override
160 public void insertBook(Book book) throws SQLIntegrityConstraintViolationException,
161     SQLException {
162     if (!checkYearString(book.yearPublished)) {
163         throw new IllegalArgumentException("Book yearPublished is improperly formatted");
164     }
165     this.PSTMT_INSERT_BOOK.setString(1, book.groupName);
166     this.PSTMT_INSERT_BOOK.setString(2, book.bookTitle);
167     this.PSTMT_INSERT_BOOK.setString(3, book.publisherName);
168     this.PSTMT_INSERT_BOOK.setString(4, book.yearPublished);
169     this.PSTMT_INSERT_BOOK.setInt(5, book.numberPages);
170
171     PSTMT_INSERT_BOOK.executeUpdate();
172 }

```



```

172
173 @Override
174 public void replacePublisher(String oldName, String newName)
175     throws SQLIntegrityConstraintViolationException, SQLException {
176     this.PSTMT_UPDATE_PUBLISHERS.setString(1, newName);
177     this.PSTMT_UPDATE_PUBLISHERS.setString(2, oldName);
178
179     this.PSTMT_UPDATE_PUBLISHERS.executeUpdate();
180 }
181
182 @Override
183 public void deleteBook(String title, String writingGroup) throws SQLException {
184     this.PSTMT_DELETE_BOOK.setString(1, title);
185     this.PSTMT_DELETE_BOOK.setString(2, writingGroup);
186     this.PSTMT_DELETE_BOOK.executeUpdate();
187 }
188
189 @Override
190 public void deleteBook(BookKeyData key) throws SQLException {
191     deleteBook(key.bookTitle, key.writingGroup);
192 }
193
194 @Override
195 public List<String> listPublisherNames() throws SQLException {
196     Statement stmt = this.con.createStatement();
197     List<String> pubs = new LinkedList<>();
198     ResultSet results = stmt.executeQuery(SQL_GET_PUBLISHER_NAMES);
199
200     while (results.next()) {
201         pubs.add(results.getString(1));
202     }
203
204     results.close();
205     stmt.close();
206
207     return pubs;
208 }
209
210 @Override
211 public List<Publisher> listPublishers() throws SQLException {
212     Statement stmt = this.con.createStatement();
213     List<Publisher> pubs = new LinkedList<>();
214     ResultSet results = stmt.executeQuery(SQL_GET_PUBLISHERS);
215
216     while (results.next()) {
217         pubs.add(new Publisher(results.getString(1), results.getString(2),
results.getString(3),
218             results.getString(4)));
219     }
220
221     return pubs;
222 }
223
224 @Override
225 public Publisher getPublisher(String name) throws SQLException {
226     this.PSTMT_GET_PUBLISHER.setString(1, name);
227     ResultSet results = this.PSTMT_GET_PUBLISHER.executeQuery();

```

```

228
229     if (results.next()) {
230         Publisher pub = new Publisher(results.getString(1), results.getString(2),
results.getString(3),
231             results.getString(4));
232         results.close();
233         return pub;
234     } else {
235         return null;
236     }
237 }
238
239
240 @Override
241 public void insertPublisher(Publisher info) throws
SQLIntegrityConstraintViolationException, SQLException {
242     PSTMT_INSERT_PUBLISHER.setString(1, info.publisherName);
243     PSTMT_INSERT_PUBLISHER.setString(2, info.publisherAddress);
244     PSTMT_INSERT_PUBLISHER.setString(3, info.publisherPhone);
245     PSTMT_INSERT_PUBLISHER.setString(4, info.publisherEmail);
246     PSTMT_INSERT_PUBLISHER.execute();
247 }
248
249 @Override
250 public void deletePublisher(String name) throws SQLIntegrityConstraintViolationException,
SQLException {
251     PSTMT_DELETE_PUBLISHER.setString(1, name);
252     PSTMT_DELETE_PUBLISHER.execute();
253 }
254
255 @Override
256 public List<String> listWritingGroupNames() throws SQLException {
257     Statement stmt = con.createStatement();
258     List<String> names = new LinkedList<>();
259     ResultSet results = stmt.executeQuery(SQL_GET_GROUP_NAMES);
260
261     while (results.next()) {
262         names.add(results.getString(1));
263     }
264
265     results.close();
266     stmt.close();
267
268     return names;
269 }
270
271 @Override
272 public List<WritingGroup> listWritingGroups() throws SQLException {
273     Statement stmt = con.createStatement();
274     List<WritingGroup> groups = new LinkedList<>();
275     ResultSet results = stmt.executeQuery(SQL_GET_GROUPS);
276
277     while (results.next()) {
278         groups.add(new WritingGroup(results.getString(1), results.getString(2),
results.getString(3),
279             results.getString(4)));
280     }

```

```

281
282     results.close();
283
284     return groups;
285 }
286
287 @Override
288 public WritingGroup getWritingGroup(String name) throws SQLException {
289     this.PSTMT_GET_WRITING_GROUP.setString(1, name);
290     ResultSet result = this.PSTMT_GET_WRITING_GROUP.executeQuery();
291
292     if (result.next()) {
293         WritingGroup group = new WritingGroup(result.getString(1), result.getString(2),
result.getString(3),
294         result.getString(4));
295         result.close();
296         return group;
297     }
298
299     return null;
300 }
301
302 @Override
303 public void insertWritingGroup(WritingGroup group) throws
SQLIntegrityConstraintViolationException, SQLException {
304     this.PSTMT_INSERT_WRITING_GROUP.setString(1, group.groupName);
305     this.PSTMT_INSERT_WRITING_GROUP.setString(2, group.headWriter);
306     this.PSTMT_INSERT_WRITING_GROUP.setString(3, group.yearFormed);
307     this.PSTMT_INSERT_WRITING_GROUP.setString(4, group.subject);
308     this.PSTMT_INSERT_WRITING_GROUP.executeUpdate();
309 }
310
311 @Override
312 public void deleteWritingGroup(String groupName) throws
SQLIntegrityConstraintViolationException, SQLException {
313     this.PSTMT_DELETE_WRITING_GROUP.setString(1, groupName);
314     this.PSTMT_DELETE_WRITING_GROUP.executeUpdate();
315 }
316
317 /**
318  * Checks the given string is a valid year string, that is, exactly 4
319  * characters in length, all of which are digits.
320  *
321  * @param year the string to check
322  * @return is year valid
323  */
324 public static boolean checkYearString(String year) {
325     if (year.length() > 4) {
326         return false;
327     }
328     for (int i = 0; i < year.length(); i++) {
329         if (!Character.isDigit(year.charAt(i))) {
330             return false;
331         }
332     }
333
334     return true;

```

OpsImplFactory.java

```
335     }  
336  
337 }  
338
```

BookDetail.java

```
1
2 package cecs323.jdbcproject.pojos;
3
4 /**
5  * The BookDetail class combines the {@link Book}, {@link Publisher}, and
6  * {@link WritingGroup} classes into one object.
7  *
8  * @author Nicholas
9  */
10 public class BookDetail {
11     /**
12      * The {@link Book} that this <code>BookDetails</code> stores details of.
13      */
14     public final Book book;
15
16     /**
17      * The {@link Publisher} of {@link #book}.
18      */
19     public final Publisher publisher;
20
21     /**
22      * The {@link WritingGroup} that wrote {@link #book}.
23      */
24     public final WritingGroup writingGroup;
25
26     /**
27      * Creates a new <code>BookDetail</code> for the given {@link Book} with the
28      * given {@link Publisher} and {@link WritingGroup}.
29      *
30      * @param b the book to detail
31      * @param p the publisher of the book
32      * @param wg the writing group of the book
33      */
34     public BookDetail(Book b, Publisher p, WritingGroup wg) {
35         this.book = b;
36         this.publisher = p;
37         this.writingGroup = wg;
38     }
39 }
40
```

Book.java

```
1 package cecs323.jdbcproject.pojos;
2
3 /**
4  * The Book class is a POJO (Plain Old Java Object) that is used to encapsulate
5  * the attributes of an entry in the Books table.
6  *
7  * @author Nicholas
8  */
9 public class Book {
10     /**
11      * The name of the {@link WritingGroup} that wrote this Book.
12      *
13      * Must be no more than 30 characters in length, and the name of an existing
14      * <code>WritingGroup</code>.
15      */
16     public String groupName;
17
18     /**
19      * The title of this Book.
20      *
21      * Must be no more than 40 characters in length, and unique within the
22      * {@link WritingGroup} that wrote this Book.
23      *
24      * @see #groupName
25      */
26     public String bookTitle;
27
28     /**
29      * The name of the {@link Publisher} that published this Book.
30      *
31      * Must be no more than 30 characters in length, and equivalent to the name
32      * of an existing <code>Publisher</code>.
33      */
34     public String publisherName;
35
36     /**
37      * The year in which this Book was published.
38      *
39      * Must be exactly 4 characters in length.
40      */
41     public String yearPublished;
42
43     /**
44      * The number of pages in this Book.
45      */
46     public int numberPages;
47
48     /**
49      * Creates a new Book object with the given values.
50      *
51      * @param title the title of the book
52      * @param groupName the name of the WritingGroup that wrote this book
53      * @param pubName the name of the publisher that published this book
54      * @param year the year that this book was published
55      * @param pages the number of pages in this book
56      */
57     public Book(String title, String groupName, String pubName, String year, int pages) {
```

Book.java

```
58     this.bookTitle = title;
59     this.groupName = groupName;
60     this.publisherName = pubName;
61     this.numberPages = pages;
62     this.yearPublished = year;
63 }
64 }
65
```

BookKeyData.java

```
1 package cecs323.jdbcproject.pojos;
2
3 /**
4  * The BookKeyData class is a POJO (Plain Old Java Object) that stores the
5  * primary key attributes of an entry in the Books table.
6  *
7  * @author Nicholas
8  */
9 public class BookKeyData {
10     /**
11      * The title of the book.
12      */
13     public String bookTitle;
14
15     /**
16      * The name of the writing group that wrote the book.
17      */
18     public String writingGroup;
19
20     /**
21      * Constructor for BookKeyData
22      *
23      * @param bookTitle title of a book
24      * @param writingGroup name of a writing group
25      */
26     public BookKeyData(String bookTitle, String writingGroup) {
27         this.bookTitle = bookTitle;
28         this.writingGroup = writingGroup;
29     }
30 }
31
```


Publisher.java

```
1 package cecs323.jdbcproject.pojos;
2
3 /**
4  * The Publisher class is a POJO (Plain Old Java Object) used to encapsulate the
5  * attributes of an entry in the Publishers table.
6  *
7  * @author Nicholas
8  */
9 public class Publisher {
10     /**
11      * The name of this Publisher.
12      *
13      * Must be no more than 30 characters in length.
14      */
15     public String publisherName;
16
17     /**
18      * The address of this Publisher.
19      *
20      * Must be no more than 30 characters in length.
21      */
22     public String publisherAddress;
23
24     /**
25      * The phone number of this Publisher.
26      *
27      * Must be no more than 20 characters in length.
28      */
29     public String publisherPhone;
30
31     /**
32      * The email address of this Publisher.
33      *
34      * Must be no more than 50 characters in length.
35      */
36     public String publisherEmail;
37
38     /**
39      * Creates a new Publisher object with the given name, address, phone
40      * number, and email address.
41      *
42      * @param name the name of the publisher
43      * @param address the address of the publisher
44      * @param phone the phone number of the publisher
45      * @param email the email address of the publisher
46      */
47     public Publisher(String name, String address, String phone, String email) {
48         this.publisherName = name;
49         this.publisherAddress = address;
50         this.publisherPhone = phone;
51         this.publisherEmail = email;
52     }
53 }
54
```

WritingGroup.java

```
1 package cecs323.jdbcproject.pojos;
2
3 /**
4  * The WritingGroup class is a POJO (Plain Old Java Object) that is used to
5  * compartmentalize the attributes of an entry in the WritingGroups table.
6  *
7  * @author Nicholas
8  */
9 public class WritingGroup {
10     /**
11      * The name of the WritingGroup.
12      *
13      * Must be no more than 30 characters in length.
14      */
15     public String groupName;
16
17     /**
18      * The name of the head writer in this WritingGroup.
19      *
20      * Must be no more than 30 characters in length.
21      */
22     public String headWriter;
23
24     /**
25      * The year that this WritingGroup was formed.
26      *
27      * Must be exactly 4 characters in length.
28      */
29     public String yearFormed;
30
31     /**
32      * The subject that this WritingGroup writes about.
33      *
34      * Must be no more than 50 characters in length.
35      */
36     public String subject;
37
38     /**
39      * Creates a new WritingGroup object with the given name, head writer, year
40      * and subject.
41      *
42      * @param name the name of the writing group
43      * @param head the head writer of the writing group
44      * @param year the year in which the group formed
45      * @param subj the subject that the group writes about
46      */
47     public WritingGroup(String name, String head, String year, String subj) {
48         this.groupName = name;
49         this.headWriter = head;
50         this.yearFormed = year;
51         this.subject = subj;
52     }
53 }
54
```

```

1 CREATE TABLE publishers(
2     publishername VARCHAR(40) NOT NULL,
3     publisheraddress VARCHAR(40),
4     publisherphone VARCHAR(40),
5     publisheremail VARCHAR(40),
6     CONSTRAINT publisher_pk PRIMARY KEY (publishername)
7 );
8 CREATE TABLE writinggroups(
9     groupname VARCHAR(40) NOT NULL,
10    headwriter VARCHAR(40),
11    yearformed VARCHAR(40),
12    subject VARCHAR(40),
13    CONSTRAINT writinggroup_pk PRIMARY KEY (groupname)
14 );
15 CREATE TABLE books(
16     groupname VARCHAR(40) NOT NULL,
17     booktitle VARCHAR(40) NOT NULL,
18     publishername VARCHAR(40) NOT NULL,
19     yearpublished VARCHAR(40),
20     numberpages INT,
21     CONSTRAINT books_pk PRIMARY KEY (groupname, booktitle),
22     CONSTRAINT publishers_books_FK FOREIGN KEY (publishername) REFERENCES
23     publishers(publishername),
24     CONSTRAINT writinggroup_books_FK FOREIGN KEY (groupname) REFERENCES
25     writinggroups(groupname)
26 );
27 --INSERT VALUE IN publishers--
28 INSERT INTO publishers (publishername, publisheraddress, publisherphone,
29 publisheremail) VALUES
30 ('Apple Inc.', 'New York, New York', '617501426', 'apple@live.com'),
31 ('Orange Inc.', 'Cape Town, South Africa', '901524395', 'orange@hotmail.com'),
32 ('Mango Inc.', 'Tokyo, Japan', '444370190', 'mango@gmail.com');
33
34 --INSERT VALUE INTO writinggroups--
35 INSERT INTO writinggroups (groupname, headwriter, yearformed, subject) VALUES
36 ('Alpha', 'John', '1538', 'English'),
37 ('Omega', 'Lucy', '1982', 'History'),
38 ('Epsilon', 'Marry', '2007', 'Chemistry'),
39 ('Delta', 'Zake', '2014', 'Physic'),
40 ('Beta', 'Niko', '1753', 'Math');
41
42 --INSERT VALUE INTO books--
43 INSERT INTO books (groupname, booktitle, publishername, yearpublished,
44 numberpages)VALUES
45 ('Alpha', 'In Search of Lost Time by Marcel Proust', 'Orange Inc.', '1563', 156),
46 ('Omega', 'Don Quixote by Miguel de Cervantes', 'Orange Inc.', '1792', 387),
47 ('Delta', 'Ulysses by James Joyce', 'Orange Inc.', '2017', 4562),
48 ('Beta', 'The Odyssey by Homer', 'Apple Inc.', '2000', 1234),
49 ('Epsilon', 'War and Peace by Leo Tolstoy', 'Mango Inc.', '1982', 7453),
50 ('Epsilon', 'Moby Dick by Herman Melville', 'Mango Inc.', '1970', 4125),
51 ('Beta', 'The Divine Comedy by Dante Alighieri', 'Orange Inc.', '1999', 127),
52 ('Omega', 'Hamlet by William Shakespeare', 'Mango Inc.', '2003', 852),
53 ('Delta', 'Madame Bovary by Gustave Flaubert', 'Apple Inc.', '2005', 465),
54 ('Alpha', 'The Iliad by Homer', 'Apple Inc.', '1985', 1357);

```

JDBC Project Result

1. List all writing groups

```
Connecting to database...
--Menu--
1. List all writing groups
2. List all data for a writing group
3. List all publishers
4. List all the data for a publisher
5. List all book titles
6. List all the data for a book
7. Insert a new book
8. Insert a new publisher
9. Remove a book
10. Exit
Enter: 1
-Listing All Writing Groups-
GroupName      HeadWriter      YearFormed      Subject
Alpha          John            English         1538
Omega          Lucy            History         1982
Epsilon        Marry           Chemistry       2007
Delta          Zake            Physic          2014
Beta           Niko            Math            1753
```

2. List all data for a writing group

```
--Menu--
1. List all writing groups
2. List all data for a writing group
3. List all publishers
4. List all the data for a publisher
5. List all book titles
6. List all the data for a book
7. Insert a new book
8. Insert a new publisher
9. Remove a book
10. Exit
Enter: 2
-Listing all data for a writing group-
-Available Group-
Alpha
Beta
Delta
Epsilon
Omega
Enter group name: Epsilon
GroupName      HeadWriter      YearFormed      Subject
Epsilon        Marry           2007            Chemistry
```

3. List all publishers

```
--Menu--
1. List all writing groups
2. List all data for a writing group
3. List all publishers
4. List all the data for a publisher
5. List all book titles
6. List all the data for a book
7. Insert a new book
8. Insert a new publisher
9. Remove a book
10. Exit
Enter: 3
-Listing all publishers-
PublisherName      PublisherAddress      PublisherPhone      PublisherEmail
Apple Inc.         New York, New York    617501426           apple@live.com
Orange Inc.        Cape Town, South Africa 901524395           orange@hotmail.com
Mango Inc.         Tokyo, Japan          444370190           mango@gmail.com
```

4. List all the data for a publisher

```
--Menu--
1. List all writing groups
2. List all data for a writing group
3. List all publishers
4. List all the data for a publisher
5. List all book titles
6. List all the data for a book
7. Insert a new book
8. Insert a new publisher
9. Remove a book
10. Exit
Enter: 4
-Listing all the data for publisher-
-Available Publishers-
Apple Inc.
Mango Inc.
Orange Inc.
Enter publisher name: Orange Inc.
PublisherName      PublisherAddress      PublisherPhone      PublisherEmail
Orange Inc.        Cape Town, South Africa 901524395           orange@hotmail.com
```

5. List all book titles

```
--Menu--
1. List all writing groups
2. List all data for a writing group
3. List all publishers
4. List all the data for a publisher
5. List all book titles
6. List all the data for a book
7. Insert a new book
8. Insert a new publisher
9. Remove a book
10. Exit
Enter: 5
-Listing all book titles-
BookTitle
In Search of Lost Time by Marcel Proust
The Iliad by Homer
The Divine Comedy by Dante Alighieri
The Odyssey by Homer
Madame Bovary by Gustave Flaubert
Ulysses by James Joyce
Moby Dick by Herman Melville
War and Peace by Leo Tolstoy
Don Quixote by Miguel de Cervantes
Hamlet by William Shakespeare
```

6. List all the data for a book

```
--Menu--
1. List all writing groups
2. List all data for a writing group
3. List all publishers
4. List all the data for a publisher
5. List all book titles
6. List all the data for a book
7. Insert a new book
8. Insert a new publisher
9. Remove a book
10. Exit
Enter: 6
-Listing all the data for a book-
-Availabe Book-
BookTitle      GroupName
In Search of Lost Time by Marcel Proust Alpha
Don Quixote by Miguel de Cervantes Omega
Ulysses by James Joyce Delta
The Odyssey by Homer Beta
War and Peace by Leo Tolstoy Epsilon
Moby Dick by Herman Melville Epsilon
The Divine Comedy by Dante Alighieri Beta
Hamlet by William Shakespeare Omega
Madame Bovary by Gustave Flaubert Delta
The Iliad by Homer Alpha
Enter BookTitle:The Odyssey by Homer
Enter groupName: Beta
BookTitle      YearPublished  NumberPages  GroupName  HeadWriter  YearFormed  Subject
The Odyssey by Homer  2000        1234        Beta      Niko        1753        Math
```

Subject	PublisherName	PublisherAddress	PublisherPhone	PublisherEmail
Math	Apple Inc.	New York, New York	617501426	apple@live.com

7. Insert a new book

```

--Menu--
1. List all writing groups
2. List all data for a writing group
3. List all publishers
4. List all the data for a publisher
5. List all book titles
6. List all the data for a book
7. Insert a new book
8. Insert a new publisher
9. Remove a book
10. Exit
Enter: 7
-Inserting a new book-
-Avaialbe Publishers-
Apple Inc.
Mango Inc.
Orange Inc.
-Avaialbe WritingGroups-
Alpha
Beta
Delta
Epsilon
Omega
Enter BookTitle: HelloWorld
Enter YearPublished: 2003
Enter NumberPages: 856
Enter GroupName: Alpha
Enter PublisherName: Orange Inc.

Enter PublisherName: Orange Inc.
--Menu--
1. List all writing groups
2. List all data for a writing group
3. List all publishers
4. List all the data for a publisher
5. List all book titles
6. List all the data for a book
7. Insert a new book
8. Insert a new publisher
9. Remove a book
10. Exit
Enter: 5
-Listing all book titles-
BookTitle
HelloWorld
In Search of Lost Time by Marcel Proust
The Iliad by Homer
The Divine Comedy by Dante Alighieri
The Odyssey by Homer
Madame Bovary by Gustave Flaubert
Ulysses by James Joyce
Moby Dick by Herman Melville
War and Peace by Leo Tolstoy
Don Quixote by Miguel de Cervantes
Hamlet by William Shakespeare

```

8. Insert a new publisher

```
--Menu--
1. List all writing groups
2. List all data for a writing group
3. List all publishers
4. List all the data for a publisher
5. List all book titles
6. List all the data for a book
7. Insert a new book
8. Insert a new publisher
9. Remove a book
10. Exit
Enter: 8
-Inserting a new publisher-
-Availabe Publishers-
Apple Inc.
Mango Inc.
Orange Inc.
Enter OldPublisherName: Apple Inc.
-Get New Publisher Info-
Enter PublisherName: Sky Comp.
Enter PublisherAddress: Beijin, China
Enter PublisherPhone: 51915856
Enter PublisherEmail: Sky@comp.com
--Menu--
1. List all writing groups
2. List all data for a writing group
3. List all publishers
4. List all the data for a publisher
5. List all book titles
6. List all the data for a book
7. Insert a new book
8. Insert a new publisher
9. Remove a book
10. Exit
Enter: 3
-Listing all publishers-


| PublisherName | PublisherAddress        | PublisherPhone | PublisherEmail     |
|---------------|-------------------------|----------------|--------------------|
| Orange Inc.   | Cape Town, South Africa | 901524395      | orange@hotmail.com |
| Mango Inc.    | Tokyo, Japan            | 444370190      | mango@gmail.com    |
| Sky Comp.     | Beijin, China           | 51915856       | Sky@comp.com       |


```

9. Remove a book

```
--Menu--
1. List all writing groups
2. List all data for a writing group
3. List all publishers
4. List all the data for a publisher
5. List all book titles
6. List all the data for a book
7. Insert a new book
8. Insert a new publisher
9. Remove a book
10. Exit
Enter: 9
-Removing a book-
-Availabe Book-


| BookTitle                               | GroupName |
|-----------------------------------------|-----------|
| In Search of Lost Time by Marcel Proust | Alpha     |
| Don Quixote by Miguel de Cervantes      | Omega     |
| Ulysses by James Joyce                  | Delta     |
| The Odyssey by Homer                    | Beta      |
| War and Peace by Leo Tolstoy            | Epsilon   |
| Moby Dick by Herman Melville            | Epsilon   |
| The Divine Comedy by Dante Alighieri    | Beta      |
| Hamlet by William Shakespeare           | Omega     |
| Madame Bovary by Gustave Flaubert       | Delta     |
| The Iliad by Homer                      | Alpha     |
| HelloWorld                              | Alpha     |


Enter BookTitle:Ulysses by James Joyce
Enter groupName: Delta
```



```
Enter groupName: Delta
--Menu--
1. List all writing groups
2. List all data for a writing group
3. List all publishers
4. List all the data for a publisher
5. List all book titles
6. List all the data for a book
7. Insert a new book
8. Insert a new publisher
9. Remove a book
10. Exit
Enter: 5
-Listing all book titles-
BookTitle
HelloWorld
In Search of Lost Time by Marcel Proust
The Iliad by Homer
The Divine Comedy by Dante Alighieri
The Odyssey by Homer
Madame Bovary by Gustave Flaubert
Moby Dick by Herman Melville
War and Peace by Leo Tolstoy
Don Quixote by Miguel de Cervantes
Hamlet by William Shakespeare
```