

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              }
425          // Replace publisher
426          d.insertPublisher(new Publisher(publisherName, publisherAddress, publisherPhone,
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

```