CECS323JDBCProject.java

```java

package cecs323.jdbcproject;

import java.sql.*;

/**
 * <h1>CECS323JDBCProject</h1> This is program is designed to be operate in
 * conjunction with a database of books, publisher, and writing group.
 *
 * @author Sotheanith Sok
 * @version 1.5
 * @since 03-16-2017
 */
public class CECS323JDBCProject {

    public static void main(String[] args) {
        // TODO code application logic here
        Scanner in = new Scanner(System.in);
        // Input data if required
        String DBNAME = "JDBCProjectDatabase";
        String USER = "IAmNotARobot";
        String PASS = "IAmNotARobot";

        // Database URL
        String DB_URL = "jdbc:derby://localhost:1527/" + DBNAME + ";user=" + USER +
";password=" + PASS;
        Connection conn = null;
        // Statement stmt=null;
        boolean done = false;
        try {
            // Register JDBC driver
            Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();
            // Open Connection
            System.out.println("Connecting to database...");
            conn = DriverManager.getConnection(DB_URL);

            // Create Datebase object
            DatabaseOperations d = OpsImplFactory.getOperationsImpl(conn);
            do {
                int choice = menu(in);
                switch (choice) {
                case 1:
                    System.out.println("-Listing All Writing Groups-");
                    listAllWritingGroups(d);
                    break;
                case 2:
                    System.out.println("-Listing all data for a writing group-");
                    listDataForAWritingGroup(d, in);
                    break;
                case 3:
                    System.out.println("-Listing all publishers-");
                    listAllPublisher(d);
                    break;
                case 4:
                    System.out.println("-Listing all the data for publisher-");
                    printDataForAPublisher(d, in);
                    break;
                case 5:
                    System.out.println("-Listing all book titles-");
                    listAllBookTitle(d);
                    break;
                case 6:
                    System.out.println("-Listing all the data for a book-");
                    listDataForABook(d, in);
                    break;
                case 7:
                    System.out.println("-Inserting a new book-");
                    insertABook(d, in);
                    break;
                case 8:
                    System.out.println("-Inserting a new publisher-");
                    insertAPublisher(d, in);
                    break;
                case 9:
                    System.out.println("-Removing a book-");
                    removeABook(d, in);
                    break;
                case 10:
                    System.out.println("-Exiting-");
                    done = true;
                    break;
                default:
                    System.out.println("-Invalid Input-");
                    break;
                }
            } while (!done);
            // Close resource when there isn't any error.
            conn.close();
        } catch (SQLException se) {
            System.out.println("ERROR: Connection to Database failed!!!");
        } catch (Exception e) {
            // Testing for unexpected exception.
            System.out.println("Unknown Exception was threw to main");
            System.out.println(e);
        } finally {
            // Error caused by closing resources.
            try {
                conn.close();
                in.close();
            } catch (SQLException se) {
                System.out.println("H3");
                se.printStackTrace();
                // This error is mostly caused by wrong database info.
            } catch (NullPointerException np) {
                System.out.println("ERROR: Database related informations are incorrect.");
            }
        }
    }

    /**
     * Print menu and get user input
     *
     * @param in Scanner for keyboard
     * @return valid user choice
```

```java
     */
    public static int menu(Scanner in) {
        boolean done = false;
        int choice = 0;
        System.out.println("--Menu--");
        System.out.println("1. List all writing groups ");
        System.out.println("2. List all data for a writing group");
        System.out.println("3. List all publishers");
        System.out.println("4. List all the data for a publisher");
        System.out.println("5. List all book titles");
        System.out.println("6. List all the data for a book");
        System.out.println("7. Insert a new book");
        System.out.println("8. Insert a new publisher");
        System.out.println("9. Remove a book");
        System.out.println("10. Exit");
        while (!done) {
            try {
                System.out.print("Enter: ");
                choice = in.nextInt();
                if (!(choice >= 1 && choice <= 10)) {
                    throw new NumberFormatException();
                }
                done = true;
            } catch (InputMismatchException ime) {
                in.next();
                System.out.print("Invalid Input. Re-enter: ");
            } catch (NumberFormatException nfe) {
                System.out.print("Invalid Input. Re-enter: ");
            }
        }
        return choice;
    }

    /**
     * List all data related to all writing groups
     *
     * @param w WritingGroupOperations object.
     * @throws SQLException
     */
    public static void listAllWritingGroups(DatabaseOperations w) {
        try {
            List<WritingGroup> list = w.listWritingGroups();
            // Check if WritingGroups is empty.
            if (list.size() == 0) {
                throw new SQLException();
            }
            // Print WritingGroups.
            System.out.printf("%-20s%-20s%-20s%-20s\n", "GroupName", "HeadWriter",
"YearFormed", "Subject");
            for (int i = 0; i < list.size(); i++) {
                System.out.printf("%-20s%-20s%-20s%-20s\n", list.get(i).groupName,
list.get(i).headWriter,
                        list.get(i).subject, list.get(i).yearFormed);
            }
        } catch (SQLException s) {
            System.out.println("ERROR: WritingGroups is empty.");
        }
    }

    /**
     * List all data for a specific WritingGroup (4)
     *
     * @param w WritingGroupOperations object
     * @param in Scanner for keyboard
     * @throws SQLException
     */
    public static void listDataForAWritingGroup(DatabaseOperations w, Scanner in) {
        try {
            List<String> list = w.listWritingGroupNames();
            // Check if WritingGroups is empty.
            if (list.size() == 0) {
                throw new SQLException();
            }
            // Print available WritingGroups.
            System.out.println("-Available Group-");
            for (int i = 0; i < list.size(); i++) {
                System.out.println(list.get(i));
            }
            // Get input.
            System.out.print("Enter group name: ");
            String groupName = in.next();
            WritingGroup k = w.getWritingGroup(groupName);
            // Print result
            if (k.groupName != null) {
                System.out.printf("%-20s%-20s%-20s%-20s\n", "GroupName", "HeadWriter",
"YearFormed", "Subject");
                System.out.printf("%-20s%-20s%-20s%-20s\n", k.groupName, k.headWriter,
k.yearFormed, k.subject);
            }
        } catch (SQLException s) {
            System.out.println("ERROR: WritingGroups is empty.");
        } catch (NullPointerException np) {
            System.out.println("ERROR: WritingGroup was not found.");
        }
    }

    /**
     * List information related to all publishers (4)
     *
     * @param p PublisherOperations object
     * @throws SQLException
     */
    public static void listAllPublisher(DatabaseOperations p) {
        try {
            List<Publisher> list = p.listPublishers();
            // Check if publishers is empty.
            if (list.size() == 0) {
                throw new SQLException();
            }
            // Print result.
            System.out.printf("%-20s%-30s%-20s%-20s\n", "PublisherName", "PublisherAddress",
"PublisherPhone",
                    "PublisherEmail");
            for (int i = 0; i < list.size(); i++) {
                System.out.printf("%-20s%-30s%-20s%-20s\n", list.get(i).publisherName, list.get(i).publisherAddress,
                        list.get(i).publisherPhone, list.get(i).publisherEmail);
            }
        } catch (SQLException s) {
            System.out.println("ERROR: Publishers is empty.");
        }
    }

    /**
     * Print all data for a publisher
     *
     * @param d DatabaseOperations object
     */
    public static void printDataForAPublisher(DatabaseOperations d, Scanner in) {
        try {
            List<String> list = d.listPublisherNames();
            // Check if Publishers is empty.
            if (list.size() == 0) {
                throw new SQLException();
            }
            // Print available publishers
            System.out.println("-Available Publishers-");
            for (int i = 0; i < list.size(); i++) {
                System.out.println(list.get(i));
            }
            // Get input
            System.out.print("Enter publisher name: ");
            in.nextLine();
            String pubName = in.nextLine();
            Publisher p = d.getPublisher(pubName);
            // Print result
            if (p.publisherName != null) {
                System.out.printf("%-20s%-30s%-20s%-20s\n", "PublisherName", "PublisherAddress", "PublisherPhone",
                        "PublisherEmail");
                System.out.printf("%-20s%-30s%-20s%-20s\n", p.publisherName, p.publisherAddress, p.publisherPhone,
                        p.publisherEmail);
            }
        } catch (SQLException e) {
            System.out.println("ERROR: Publishers is empty.");
        } catch (NullPointerException np) {
            System.out.println("ERROR: Publisher was not found.");
        }
    }

    /**
     * List all the book title
     *
     * @param b
     * @throws SQLException
     */
    public static void listAllBookTitle(DatabaseOperations b) {
        try {
            List<String> list = b.listBookTitles();
            // Check if Books is empty.
            if (list.size() == 0) {
                throw new SQLException();
            }
            // Print result.
            System.out.printf("%-10s", "BookTitle\n");
            for (int i = 0; i < list.size(); i++) {
                System.out.printf("%-10s\n", list.get(i));
            }
        } catch (SQLException s) {
            System.out.println("ERROR: Books is empty.");
        }
    }

    /**
     * List all data for a specific book including related writing group and
     * publisher.
     *
     * @param b BookOperations object
     * @param in Scanner for keyboard
     * @throws SQLException
     */
    public static void listDataForABook(DatabaseOperations b, Scanner in) {
        try {
            // Check if Books is empty.
            if (b.listBookTitles().size() == 0) {
                throw new SQLException();
            }
            // Print available Publishers and WritingGroups.
            printAvaialbeBooks(b);
            // Get input.
            System.out.print("Enter BookTitle:");
            in.nextLine();
            String bookTitle = in.nextLine();
            System.out.print("Enter groupName: ");
            String groupName = in.nextLine();
            Book book = b.getBook(new BookKeyData(bookTitle, groupName));
            BookDetail bookDetail = b.getBookDetails(new BookKeyData(bookTitle, groupName));
            // Print result.
            if (book.groupName != null) {
                System.out.printf("%-40s%-20s%-20s%-20s%-20s%-20s%-20s%-20s%-30s%-30s%-20s\n",
"BookTitle",
                        "YearPublished", "NumberPages", "GroupName", "HeadWriter",
"YearFormed", "Subject",
                        "PublisherName", "PublisherAddress", "PublisherPhone",
"PublisherEmail");
                System.out.printf("%-40s%-20s%-20s%-20s%-20s%-20s%-20s%-20s%-30s%-30s%-20s\n",
book.bookTitle,
                        book.yearPublished, book.numberPages,
bookDetail.writingGroup.groupName,
                        bookDetail.writingGroup.headWriter,
bookDetail.writingGroup.yearFormed,
                        bookDetail.writingGroup.subject, bookDetail.publisher.publisherName,
                        bookDetail.publisher.publisherAddress,
bookDetail.publisher.publisherPhone,
                        bookDetail.publisher.publisherEmail);
            }
        } catch (SQLException s) {
            System.out.println("ERROR: Books is empty.");
        } catch (NullPointerException np) {
            System.out.println("ERROR: Book was not found.");
        }
    }

    /**
     * Insert a new book into database
     *
     * @param d DatabaseOpeartions object
     * @param in Scanner for keyboard
     * @throws SQLIntegrityConstraintViolationException
     * @throws SQLException
     */
    public static void insertABook(DatabaseOperations d, Scanner in) {
        try {
            if (d.listWritingGroupNames().size() == 0 || d.listPublisherNames().size() == 0) {
                throw new SQLException();
            }
            // Print out available publisher and group.
            System.out.println("-Avaialbe Publishers-");
            List<String> s = d.listPublisherNames();
            for (int i = 0; i < s.size(); i++) {
                System.out.println(s.get(i));
            }
            System.out.println("-Avaialbe WritingGroups-");
            s = d.listWritingGroupNames();
            for (int i = 0; i < s.size(); i++) {
                System.out.println(s.get(i));
            }
            // Get input.
            System.out.print("Enter BookTitle: ");
            in.nextLine();
            String bookTitle = in.nextLine();
            System.out.print("Enter YearPublished: ");
            String yearPublished = in.nextLine();
            System.out.print("Enter NumberPages: ");
            int numberPages = in.nextInt();
            System.out.print("Enter GroupName: ");
            in.nextLine();
            String groupName = in.nextLine();
            System.out.print("Enter PublisherName: ");
            String publisherName = in.nextLine();
            // Insert into Books.
            d.insertBook(new Book(bookTitle, groupName, publisherName, yearPublished,
numberPages));
        } catch (SQLException s) {
            System.out.println("ERROR: Unable to insert book when publishers or writing groups
is empty.");
        } catch (IllegalArgumentException iae) {
            System.out.println("ERROR: YearPublished should be integer. Insertion Fail!!!");
        } catch (InputMismatchException im) {
            System.out.println("ERROR: NumberPages should be integer. Insertion Fail!!!");
        }
    }

    /**
```

```
     * Insert and replace old publisher with a new one
     *
     * @param d DatabaseOperations objects
     * @param in Scanner for keyboard
     * @throws SQLIntegrityConstraintViolationException
     * @throws SQLException
     */
    public static void insertAPublisher(DatabaseOperations d, Scanner in) {
        try {
            // Check if Publishers is empty.
            if (d.listPublisherNames().size() == 0) {
                throw new SQLException();
            }
            // Print available publishers
            System.out.println("-Avaialbe Publishers-");
            List<String> s = d.listPublisherNames();
            for (int i = 0; i < s.size(); i++) {
                System.out.println(s.get(i));
            }
            // Get input.
            System.out.print("Enter OldPublisherName: ");
            in.nextLine();
            String oldPub = in.nextLine();
            System.out.println("-Get New Publisher Info-");
            System.out.print("Enter PublisherName: ");
            String publisherName = in.nextLine();
            System.out.print("Enter PublisherAddress: ");
            String publisherAddress = in.nextLine();
            System.out.print("Enter PublisherPhone: ");
            String publisherPhone = in.nextLine();
            System.out.print("Enter PublisherEmail: ");
            String publisherEmail = in.nextLine();
            // Check if oldPublisher actually exist.
            if (d.getPublisher(oldPub).publisherName == null) {
            }
            // Replace publisher
            d.insertPublisher(new Publisher(publisherName, publisherAddress, publisherPhone,
publisherEmail));
            d.replacePublisher(oldPub, publisherName);
            d.deletePublisher(oldPub);
        } catch (NullPointerException np) {
            System.out.println("ERROR: Old publisher was not found. Insertion Fail!!!");
        } catch (SQLException s) {
            System.out.println("ERROR: Publisher is empty");
        }
    }

    /**
     * Remove book based on title and group name
     *
     * @param d DatabaseOperations object
     * @param in Scanner for keyboard
     * @throws SQLException
     */
    public static void removeABook(DatabaseOperations d, Scanner in) {
        try {
            // Check if Books is empty.
            if (d.listBookTitles().size() == 0) {
                throw new SQLException();
            }
            // Print available Publishers and WritingGroups
            printAvaialbeBooks(d);
            // Get input
            System.out.print("Enter BookTitle:");
            in.nextLine();
            String bookTitle = in.nextLine();
            System.out.print("Enter groupName: ");
            String groupName = in.nextLine();
            // Check if the book existed.
            if (d.getBook(bookTitle, groupName).bookTitle == null) {
                throw new NullPointerException();
            }
            // Perform operation
            d.deleteBook(new BookKeyData(bookTitle, groupName));
        } catch (SQLException s) {
            System.out.println("ERROR: Books is empty.");
        } catch (NullPointerException iie) {
            System.out.println("ERROR: Book doesn't not existed in the database.");
        }
    }

    /**
     * This function is used to print available bookTitle and groupName
     * combination.
     *
     * @param d DatabaseOperations object
     * @throws SQLException the exception which will be handle by other
     *             functions.
     */
    public static void printAvaialbeBooks(DatabaseOperations d) throws SQLException {
        List<Book> list = d.listBooks();
        System.out.println("-Avaialbe Book-");
        System.out.printf("%-40s%-10s\n", "BookTitle", "GroupName");
        for (int i = 0; i < list.size(); i++) {
            System.out.printf("%-40s%-10s\n", list.get(i).bookTitle, list.get(i).groupName);
        }
    }
}
```