```java
package cecs323.jdbcproject.interconnect;

import cecs323.jdbcproject.pojos.Book;

/**
 * The DatabaseOperations interface defines the interface for a class that can
 * perform operations on the database.
 *
 * @author Nicholas Utz
 */
public interface DatabaseOperations {
    /**
     * Returns a {@link List} of Strings, containing the titles of all of the
     * entries in the Books table.
     *
     * @return list of book titles
     */
    public List<String> listBookTitles() throws SQLException;

    /**
     * Returns a {@link List} of {@link Book}s, containing all of the
     * information in the Books table.
     *
     * @return list of all books
     */
    public List<Book> listBooks() throws SQLException;

    /**
     * Returns a {@link Book} storing all of the data pertaining to the book
     * with the given title, written by the WritingGroup with the given name.
     *
     * @param title the title of the book to fetch
     * @param writingGroup the writing group that wrote the book to fetch
     * @return book info
     */
    public Book getBook(String title, String writingGroup) throws SQLException;

    /**
     * Returns a {@link Book} storing all of the data pertaining to the book
     * with the primary key data.
     *
     * @param key the key data of the book to fetch
     * @return book info
     */
    public Book getBook(BookKeyData key) throws SQLException;

    /**
     * Returns a {@link BookDetail} object, containing all available data
     * pertaining to the book with the given title, written by the given writing
     * group, including the publisher and writing group.
     *
     * @param title the title of the book to fetch
     * @param writingGroup the writing group name of the book to fetch
     * @return book details
     */
    public BookDetail getBookDetails(String title, String writingGroup) throws SQLException;

```

```java
    /**
     * Returns a {@link BookDetail} object, storing all of the data pertaining
     * to the book with the given primary key data, and the publisher and
     * writing group of the book.
     *
     * @param key the key of the book to fetch
     * @return book details
     */
    public BookDetail getBookDetails(BookKeyData key) throws SQLException;

    /**
     * Inserts the given book into the books table.
     *
     * @param book the book to insert
     * @throws SQLException if a SQLException occurs while attempting to insert
     */
    public void insertBook(Book book) throws SQLIntegrityConstraintViolationException,
  SQLException;

    /**
     * Replaces the given old publisher name with a new one, for all books
     * published by the old publisher.
     *
     * @param oldName the name of the publisher to be replaced
     * @param newName the name of the publisher replacing the old one
     * @throws SQLException if a SQLException occurs while updating
     */
    public void replacePublisher(String oldName, String newName)
            throws SQLIntegrityConstraintViolationException, SQLException;

    /**
     * Deletes the book with the given title and writing group from the books
     * table.
     *
     * @param title the title of the book to delete
     * @param writingGroup the writing group who wrote the book to delete
     * @throws SQLException if a SQLException occurs while deleting
     */
    public void deleteBook(String title, String writingGroup) throws SQLException;

    /**
     * Deletes the book with the given primary key data from the books table.
     *
     * @param key the primary key data of the book to delete
     * @throws SQLException if a SQL exception occurs while deleting
     */
    public void deleteBook(BookKeyData key) throws SQLException;

    /**
     * Returns a {@link List} of {@link String}s, representing the names of all
     * of the entries in the Publishers table.
     *
     * @return list of publisher names
     */
    public List<String> listPublisherNames() throws SQLException;

    /**
```

```
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      *
```

```
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
    SQLIntegrityConstraintViolationException, SQLException;
191 
192     /**
193      * Deletes the writing group with the given name.
194      *
195      * @param groupName the name of the writing group to delete
196      * @throws SQLException if an exception is raised while deleting
197      */
198     public void deleteWritingGroup(String groupName) throws
    SQLIntegrityConstraintViolationException, SQLException;
199 }
200 
```