```java
package impl;

import cecs323.jdbcproject.interconnect.DatabaseOperations;

/**
 * The OpsImplFactory class is a factory for classes that implement the
 * *Operations interfaces defined in {@link cecs323.jdbcproject.interconnect}.
 *
 * @author Nicholas Utz
 */
public class OpsImplFactory {

    public static DatabaseOperations getOperationsImpl(Connection con) throws SQLException {
        return new OpsImpl(con);
    }
}

class OpsImpl implements DatabaseOperations {

    private final Connection con;

    private static final String SQL_GET_TITLES = "SELECT booktitle FROM books";
    private static final String SQL_GET_BOOKS = "SELECT * FROM books";
    private static final String SQL_GET_BOOK = "SELECT * FROM books WHERE booktitle=? AND groupname=?";
    private static final String SQL_INSERT_BOOK = "INSERT INTO books (groupname, booktitle, "
            + "publishername, yearpublished, numberpages) VALUES (?,?,?,?,?)";
    private static final String SQL_UPDATE_PUBLISHERS = "UPDATE books SET publishername=? WHERE publishername=?";
    private static final String SQL_DELETE_BOOK = "DELETE FROM books WHERE booktitle=? AND groupname=?";
    private static final String SQL_GET_PUBLISHER_NAMES = "SELECT publishername FROM publishers";
    private static final String SQL_GET_PUBLISHERS = "SELECT * FROM publishers";
    private static final String SQL_GET_PUBLISHER = "SELECT * FROM publishers WHERE publishername=?";
    private static final String SQL_INSERT_PUBLISHER = "INSERT INTO publishers (publishername, "
            + "publisheraddress, publisherphone, publisheremail) VALUES (?, ?, ?, ?)";
    private static final String SQL_DELETE_PUBLISHER = "DELETE FROM publishers WHERE publishername=?";
    private static final String SQL_GET_GROUP_NAMES = "SELECT groupname FROM writinggroups";
    private static final String SQL_GET_GROUPS = "SELECT * FROM writinggroups";
    private static final String SQL_GET_WRITING_GROUP = "SELECT * FROM writinggroups WHERE groupname=?";
    private static final String SQL_INSERT_WRITING_GROUP = "INSERT INTO writinggroups (groupname, "
            + "headwriter, yearformed, subject) VALUES (?,?,?,?)";
    private static final String SQL_DELETE_WRITING_GROUP = "DELETE FROM writinggroups WHERE groupname=?";

    private final PreparedStatement PSTMT_GET_BOOK;
    private final PreparedStatement PSTMT_INSERT_BOOK;
    private final PreparedStatement PSTMT_UPDATE_PUBLISHERS;
    private final PreparedStatement PSTMT_DELETE_BOOK;
    private final PreparedStatement PSTMT_GET_PUBLISHER;
    private final PreparedStatement PSTMT_INSERT_PUBLISHER;
```

```
    private final PreparedStatement PSTMT_DELETE_PUBLISHER;
    private final PreparedStatement PSTMT_GET_WRITING_GROUP;
    private final PreparedStatement PSTMT_INSERT_WRITING_GROUP;
    private final PreparedStatement PSTMT_DELETE_WRITING_GROUP;

    public OpsImpl(Connection con) throws SQLException {
        this.con = con;
        this.PSTMT_GET_BOOK = con.prepareStatement(SQL_GET_BOOK);
        this.PSTMT_INSERT_BOOK = con.prepareStatement(SQL_INSERT_BOOK);
        this.PSTMT_UPDATE_PUBLISHERS = con.prepareStatement(SQL_UPDATE_PUBLISHERS);
        this.PSTMT_DELETE_BOOK = con.prepareStatement(SQL_DELETE_BOOK);
        this.PSTMT_GET_PUBLISHER = con.prepareStatement(SQL_GET_PUBLISHER);
        this.PSTMT_INSERT_PUBLISHER = con.prepareStatement(SQL_INSERT_PUBLISHER);
        this.PSTMT_DELETE_PUBLISHER = con.prepareStatement(SQL_DELETE_PUBLISHER);
        this.PSTMT_GET_WRITING_GROUP = con.prepareStatement(SQL_GET_WRITING_GROUP);
        this.PSTMT_INSERT_WRITING_GROUP = con.prepareStatement(SQL_INSERT_WRITING_GROUP);
        this.PSTMT_DELETE_WRITING_GROUP = con.prepareStatement(SQL_DELETE_WRITING_GROUP);
    }

    @Override
    public List<String> listBookTitles() throws SQLException {
        Statement stmt = this.con.createStatement();
        List<String> titles = new LinkedList<>();
        ResultSet results = stmt.executeQuery(SQL_GET_TITLES);

        while (results.next()) {
            titles.add(results.getString(1));
        }

        results.close();
        stmt.close();

        return titles;
    }

    @Override
    public List<Book> listBooks() throws SQLException {
        Statement stmt = this.con.createStatement();
        List<Book> books = new LinkedList<>();
        ResultSet results = stmt.executeQuery(SQL_GET_BOOKS);

        while (results.next()) {
            books.add(new Book(results.getString(2), results.getString(1),
  results.getString(3), results.getString(4),
                    results.getInt(5)));
        }

        results.close();
        stmt.close();

        return books;
    }

    @Override
    public Book getBook(String title, String writingGroup) throws SQLException {
        this.PSTMT_GET_BOOK.setString(1, title);
        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),
    results.getString(3), results.getString(4),
121                     results.getInt(5));
122             results.close();
123             return book;
124 
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,
    SQLException {
161         if (!checkYearString(book.yearPublished)) {
162             throw new IllegalArgumentException("Book yearPublished is inproperly formatted");
163         }
164         this.PSTMT_INSERT_BOOK.setString(1, book.groupName);
165         this.PSTMT_INSERT_BOOK.setString(2, book.bookTitle);
166         this.PSTMT_INSERT_BOOK.setString(3, book.publisherName);
167         this.PSTMT_INSERT_BOOK.setString(4, book.yearPublished);
168         this.PSTMT_INSERT_BOOK.setInt(5, book.numberPages);
169 
170         PSTMT_INSERT_BOOK.executeUpdate();
171     }
```

```
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
235         } else {
236             return null;
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;
```

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