### 

## College of the North Atlantic

**CP3566 – Applied Java Programming**

**Assignment 1**

**Due Date: As Noted in the Dropbox**

**Instructor demo required prior to submission!**

**Part 1: Database Setup**

It’s assumed that your database setup is complete on your development machine. But if not, Install and configure MariaDB by following the installation guide posted on D2L.

**Part 2: Book Database**

Using HeidiSQL (or your management interface of choice), which was configured in Part 1, set up the **books** database by following these steps:

1. Open HeidiSQK
2. Create a new database called “books”
3. Open a query in the books database and copy in all the SQL statements contained in the “books.sql” file. (Alternatively, use another method of your choice)
4. Execute all the statements and confirm that the data has been correctly inserted into the database. The select statements at the end of the script should echo out all the data that you can see in the script.

Note: You can remove the “drop” statements at the beginning of the SQL file the first time you run the script. Also, you can remove the select statements at the end as you populate the database. Technically, you can leave both in, but you will see errors related to the drop statements the first time, as there will be nothing to drop.

**Part 3: Using the Book Database in Java**

1. Create a “Book” class which will represent the books from the “titles” table in the books database. Each book object will have a list of its authors – a private attribute *List<Author> authorList*. (Make sure to populate the authorList attribute when loading the objects from the database.)
   * Hint: When you add an Author to a Book make sure to add the book to the Author
   * Hint: Always check that the Author is not already an Author of the book to avoid circular adds
2. Create an “Author” class which will represent the authors from the “authors” table in the books database. Each author object will have a list of the books they have written – a private attribute *List<Book> bookList*. (Make sure to populate the bookList attribute when loading the objects from the database.)
   * Hint: When you add an Author to a Book make sure to add the book to the Author
   * Hint: Always check that the Author is not already an Author of the book to avoid circular adds
3. Create a “BookDatabaseManager” class which will handle all database connections and queries as well as store lists of Books and Authors to be used by the main application. It is recommended that you do the following:
   * Build all the methods to Add Books and Authors here so that the database is in sync with the application objects
   * Use PreparedStatements to do CRUD operations on the database using Book and author objects
     + Create – store new book or author, include their relationships
     + Read – load in individual, or all, books or authors into a list
     + Update – update or insert an individual book or author back to the database with any change
     + Delete – remove a book or author from the database
4. You must manage the relationships between books and authors using the lists in each class. Be careful not to have duplicate objects loaded. **A good approach is to load the books and authors and then build up the relationships as you load them.**
5. You may want to use a Library object which has a collection of books and authors.

**Part 4: Create a Simple Application**

1. Create a Java class called “BookApplication” with a main method that prompts the user to select any of the following options:
   1. Print all the books from the database (showing the authors)
   2. Print all the authors from the database (showing the books)
   3. Edit a book’s attributes or an authors attributes
   4. Add a book to the database for existing author(s) or new author(s)
   5. Make sure the relationships are maintained
   6. Quit
2. Note that the user should be able to continue making choices until they quit.

**Export your project as a zip file and submit to the dropbox.**