Assignment: JDBC & Simple Object-Relational Mapping

Avraham Leff

COM3580: Fall 2020

1 Assignment-Specific Packaging

The general packaging is unchanged from the basic "Homework Requirements" (see slides from first lecture and "Homework Policies for COM 3563" on Piazza). You may want to review the "programming assignment" requirements, because I plan to build, execute, and test your code.

This assignment's "DIR" **must be named** *AuthorsAndBooks*.

2 Motivation

The purpose of this assignment is to reinforce your JDBC skills in the context of a specific relational database and set of database tables.

The assignment will require you to:

- Code Java classes that integrates data spread over three database tables into a more useful end-user API. Any difficulties you experience will give you a sense of some of the real-world issues faced by object-relational mapping techniques.
- Code SQL queries as necessary to enable your Java classes to access (via JDBC) the appropriate result sets.

3 Getting Started

Download *AuthorsAndBooks.sql* from the *AuthorsAndBooks* sub-directory in the COM3563StubbedAssignments github repository.

The *AuthorsAndBooks* directory contains a set of "starter code" interface classes. This may be a mixed blessing since coding to interfaces can be tricky, but it provides you with "compilable" API requirements.

Do not, under any circumstances, change these interface classes! Check them into your code base, if you'd like, but do not modify them in any way. Show your creativity by creating any number of other interface and implementation classes ☺

4 Requirements

Your task is to implement the *QueryEngine* API such that it runs correctly against the database specified in `AuthorsAndBooks.sql`.

The API consists of three queries and two factory methods. You will need to create implementation classes for each of the five interface classes. You can name the implementation classes as you please except that **you must name the class that implements the *QueryEngine* interface as *QueryEngineImpl***. The *QueryEngineImpl* class **must only have a no-argument constructor**.

As usual, how you implement the requirements (in general) is up to you, but it must be possible for me to invoke the QueryEngine APIs (through the *QueryEngineImpl* instance) in a way that depends on no other classes besides the aforementioned interface classes.

4.1 Tips

- As you develop, be sure to use the `configureConnection` method as the basis of connecting to the database. I will be using that method to drive my test code against your implementation!

- I've documented the query APIs with Javadoc: if you find the required behavior ambiguous, please reach out to discuss!
- Although I am not requiring you to use *PreparedStatements* (as opposed to *Statement*, I suggest that you implement at least one the queries using *PreparedStatements*.

5 Credits

The "book database" concept and schema are based on Chapter 24 from *Java How To Program, Late Objects (11th Edition)* by Deitel & Deitel. I highly recommend the book despite the ratings and cost. I've morphed the requirements considerably, however, from the initial concept.