CS377, Fall 2022

Assignment 3 - Embedded SQL, JDBC

Due: Friday Nov 11, 2022, at 11:59PM [Late submissions: see syllabus for policy].

Artist DB Browser: A Java Application

For this assignment, you are tasked with implementing a Java application that retrieves and displays information about artists. We will use the same database tables defined in A2 (the artistdb schema). If, for some reason, you feel you need to use an intermediate view to execute a query inside one of the Java methods, you must create the view in that method. However, you must make sure to drop the view before exiting the method.

Before you begin, read these general instructions:

1. You may not use standard input or output. Doing so even once will result in the autotester terminating, causing you to receive zero credit.
2. Be sure to close all unused statements and result sets.
3. The database, username, and password must be passed as parameters, never "hard-coded". We will use the `connectDB()` and `disconnectDB()` methods to connect to the database with our own credentials. Our test code will use these to connect to a database. You should **not** call these in the other methods we ask you to implement; you can assume that they will be called before and after, respectively, any other method calls.
4. All of your code must be written in `ArtistBrowser.java`. This is the only file you will submit for this part.
5. You may not change the method signature of any of the functions we've asked you to implement. However, you are encouraged to write helper functions to maintain good code quality ;-).
6. Your JDBC code should have proper exception-handling logic, and must use *prepared* SQL statements (for ensuring better performance and security standards).
7. As you saw in lecture and during one of the preps: to run your code, you will need to include the JDBC driver in your class path. You may wish to review the related exercises available under the JDBC folder on Canvas.

Here's a summary of the required Java methods inside `ArtistBrowser.java`. The first two are implemented for you. For the rest, read the comments in the code carefully for additional instructions and requirements (beyond what is written here). Remember, you must not change the names and signatures of any of these core methods!

1. `connectDB`: Connect to a database with the supplied credentials.
2. `disconnectDB`: Disconnect from the database.
3. **[14 points]** `findArtistsInBands`: Return a list of all musicians who were part of a band at any point during a given timeframe.
4. **[14 points]** `findArtistsInGenre`: Return a list of all musicians and bands who released at least one album in a given genre.
5. **[18 points]** `findCollaborators`: Return a list of artists who collaborated with a given artist on some song. The list of artists should include both the case when an artist was a guest collaborator, as well as the main artist.
6. **[18 points]** `findSongwriters`: Return a list of all the songwriters that wrote songs for a given artist. Do not include the artist themself, for artists that wrote their own music (either partially or exclusively).
7. **[18 points]** `findCommonAcquaintances`: Return the common acquaintances for two given artists. A common acquaintance is considered a collaborator (either as a guest or main artist), or songwriter that worked with both given artists at any point during their careers.

8. **[18 points]** `artistConnectivity`: We define a path between two artists P1 and P2 as a sequence of artists [P1, Q1, Q2, ..., Qn, P2] such that P1 and Q1 collaborated on some song, Q1 and Q2 collaborated on some song, ..., and Qn and P2 collaborated on some song. We say that two artists are **connected** if there exists a path [P1, P2] connecting them. For example, in the test data you are given, artists 'Z' and 'Usher' are considered *connected* even though they have not collaborated directly on any song; however, 'Z' collaborated with 'Alicia Keys' who in turn had collaborated with 'Usher', therefore there is a collaboration path connecting artists 'Z' and 'Usher'. Note that accomplishing this task in pure SQL can be quite challenging. Luckily we can utilize the power of JDBC programming to tackle this problem.

For all queries, except the connectDB, disconnectDB, and artistConnectivity, the result must be returned as an `ArrayList` of strings, as per the function signatures. Hint: You can optionally implement some of these methods without writing any additional SQL code, and while utilizing Java methods that you have already implemented in the ArtistBrowser Java class!

Getting Started: Step by Step

- Download PostgreSQL and install it on your computer, if you haven't already: https://www.postgresql.org/download/
- Download all starter code in the A3 folder on Canvas.
- Execute the content of `artistdb.sql`. You can either copy its content into a pgAdmin Query Browser window and run it, or, using a terminal/shell you can use the `psql` command with `-f` flag to execute the SQL script on your computer. If you want to use the latter approach, here's an example you can follow (after modifying the file paths according to your local setup):

```
cd /Library/PostgreSQL/10/bin/     # navigate to psql binary location
./psql -U postgres -f  /replace/this/with/full/path/to/your/a3/folder/artistdb.sql
```

- Implement all the missing methods in `ArtistBrowser.java`.
- Compile the Java class:

```
javac ArtistBrowser.java
```

- Run the Java class with the JDBC driver in the classpath (replace the colon ':' with a semicolon ';' if you're using Windows OS). Provide the two required program arguments, the DB user name and password (see the code in `main()`):

```
java -cp /path/to/postgresql-42.5.0.jar: ArtistBrowser postgres postgres
```

Java Resources: You should have a basic knowledge of Java from CS171, which is a prerequisite for this course. Nevertheless, if you have Java-specific questions, feel free to consult the documentation. For your reference, the main built-in Java class that you'll need to refresh is `ArrayList`, for which you can find documentation here: `http://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html`. You can find some additional material in section 3.2 of here: `http://www.vogella.com/tutorials/JavaCollections/article.html#javacollections_lists`. For java.sql package questions, refer to the documentation here: `https://docs.oracle.com/javase/8/docs/api/java/sql/package-summary.html`.

Discussion and asking for clarifications: Use our course Piazza page to post questions about the assignment if you need a clarification about any of the methods. Or visit teaching staff during office hours!

Honor Code: The assignment is expected to be solved **individually** and is governed by the College Honor Code and Departmental Policy. Remember, any code you submit must be your own; otherwise you risk being investigated by the Honor Council and facing the consequences of that. We do check for code plagiarism (across student submissions and against online resources). Please remember to have the following comment included at the top of the Java file.

```
/*
THIS CODE WAS MY OWN WORK, IT WAS WRITTEN WITHOUT CONSULTING
CODE WRITTEN BY OTHER STUDENTS OR COPIED FROM ONLINE RESOURCES.
 _Student_Name_Here_
*/
```