CS1003 Week 6 Exercise 1: JDBC

As with all lab exercises, this exercise is not assessed. It is intended solely to help you understand the module material.

The aim of this exercise is to practice processing of relational data in Java, using JDBC. You can use VS Code or any other Java development tools of your choosing.

1. Define a relational schema

Design a relational schema with a single table containing information on recording artists (pop stars). For each artist it should include at least the name, record label and highest chart position achieved. Consider what the primary key should be, and add additional attributes if necessary.

Connect to your database either using the `sqlite3` tool or DBeaver as in last week's exercise, and set up the table. Refine and test your SQL syntax as necessary. Delete the table again.

2. Write JDBC code to set up the table

Write a Java program that creates a JDBC connection to your database, and executes the SQL statement to create the table, as developed in the previous step. You may want to reuse code from the first example in this week's lectures ***JDBCExample1*** at:

https://studres.cs.st-andrews.ac.uk/CS1003/Lectures/W06-Examples/JDBC

To connect to SQLite from Java you need a JDBC database connector like this:

```
Connection connection = DriverManager.getConnection(
"jdbc:sqlite:test.db" );
// assumes file name is test.db
```

Run your program and then check via the `sqlite3` command line interface that the table has been created correctly. Run the program again without deleting the table, and check that it still works. The desired behaviour at this stage is that the program creates the table if it's not already present, and does nothing if it is. Debug as necessary.

3. Populate the table

Add JDBC code to add two rows to the table, with data of your choosing. Check via the command line interface and DBeaver.

Define a method *insertArtist* with appropriate parameters, that inserts a new row with the given data. Refine the previous insertion code so that it uses this method.

4. Implement queries

Define a method *printArtistsWithLabel* that takes as a parameter the name of a record label, and prints out the names of all the artists in the table that have that record label. Test and debug.

Repeat with a method *deleteUnpopularArtists* that deletes all artists whose highest chart position is below a given number.

5. Multiple tables

Repeat steps 1-3 with another table that contains information on albums, this table should contain a foreign key for the artist that made the album.

Define a method *printLabelAlbums* that prints out the titles and performers of all the albums released by artists of a given record label.