Exercise Sheet 12: Application Programming II

(until Thursday 24.01.2013)

Please note that you need **50%** of all exercise points to receive the "Studienleistung". Exercises have to be turned in until **Thursday** of each respective week and must be completed in teams of two students each. You may hand in your solutions either on paper **before the lecture** or into the mailbox at the IFIS floor (Informatikzentrum 2nd floor). Please do not forget to write your "Matrikelnummer" and your tutorial group number on your solutions. Your solutions may be in German or English. Please note: To pass the "RDB I Modul" you need both the exercise points and the exam!

This week we are going to do some practical exercises using JDBC. To do so please first download the following zip archive (http://www.ifis.cs.tu-bs.de/webfm_send/1210) and unzip it in an arbitrary folder of your choice. The zip archive contains some prepared java code with some TODOs. Your task will it be to implement those parts annotated with a TODO.

To test your code it is necessary to establish a connection to a RDBMS server. If you are attending the SQL lab, you should already have username, password and server URL to connect to our DB2 server. If you are not attending the SQL lab you have several choices to connect to a RDBMS server:

1. You can ask your hiwi if you can get a login to our DB2 server and consult the according documentation from the SQL lab (http://www.ifis.cs.tu-bs.de/webfm_send/1168) in order to find out how to connect to a RDBMS server.
2. You can install an RDBMS to your computer, include the respective JDBC Level 4 driver to your CLASSPATH and connect to the RDBMS to your localhost.
3. You can use Apache derby. Derby is a very light weight RDBMS written in pure Java. To use it no installation is necessary. You just have to include the derby.jar contained in the lib folder into your java CLASSPATH. If you connect to a derby database, derby will create a folder in the path of execution to store your data. More details are annotated in the source code contained in the archive.

Please hand in your solution via email. Zip your modified source files and send them to your according hiwi (gruppe[your-group-nr]@ifis.cs.tu-bs.de).

Exercise 12.1 (20 points)

The program to be implemented should wrap some data manipulation and querying tasks for an application that has something to do with movies ☺. This is done by the MovieExplorer contained in src/main/MovieExplorer.java. In the file src/main/Main.java the application will try to connect to a database and then it will attempt to create a MovieExplorer object with the obtained Connection. The MovieExplorer can then be used to manipulate or query the database on a high level of abstraction.

a. Implement the `getConnection()` method contained in src/main/Main.java. Please create a comment that describes what RDBMS is used and how you connected to it. (4 points)
b. Implement the `createSchema(Connection)` method contained in src/main/MovieExplorer.java. With the submitted connection the method should create the tables described by the comments in the method body. (4 points)
c. Use the `createSchema(Connection)` method to create the schema in the `MovieExplorer` constructor. Handle the case in which the schema is already set. In this case the tables should not be recreated. You can do this by checking explicitly if the tables exist or by catching the `SQLException` that occurs if the tables could not be created. (4 points)
d. Using the MovieExplorer object in the Main class, insert the following movies: (2 points)
    - (1, "movie 1", 1999)
    - (2, "movie 2", 1980)
    - (3, "movie 3", 2005)
    - (4, "movie 4", 2003)
e. Expand the functionality of the `MovieExplorer` by implementing methods to insert persons and actor tuples to the according tables. Implement this methods analogous to the `insertMovie(Movie)` method. (5 points)
   This means you have to do the following steps
    - Declare `PreparedStatements` as private members of the `MovieExplorer`
    - Prepare the according `INSERT` statements in the `MovieExplorer` constructor
    - Implement the `insertPerson(Person)` and `insertActorTuple(int, int, String)` methods.
f. Expand the functionality of the `MovieExplorer` by implementing a method `getActorsToAMovie(int)` that returns a list of all persons who played a role in a given movie (described by its id). Implement this method analogous to the `getMovieBetweenYears(int, int)` method. (4 points)
   This means you have to do the following steps
    - Declare `PreparedStatements` as private members of the `MovieExplorer`
    - Prepare the according `SELECT` statements in the `MovieExplorer` constructor
    - Implement the `getActorsToAMovie(int)` and method.