Assignment 3

Cezar Ionescu, lecturer DIT948

October 17th, 2013

Abstract

In this assignment, you are asked to populate a database with tables read from a given file. You will then design and implement a GUI for displaying information from these tables, updating the tables and answering simple user queries.

Short description

1. Create an empty database using either SQLite or MySQL and use JDBC to populate it with tables read from the file tables.txt. The result will be a database similar to the SPJ one you encountered in DIT630 and in the JDBC lecture of DIT948.
2. Design and implement a simple GUI allowing users to
   (a) choose a table and
      i. display a row from it
      ii. delete a row from it
      iii. insert a new row in it
   (b) perform the following queries:
      i. find the projects employing a given supplier
      ii. find the parts supplied by a give supplier
      iii. find the total quantity of a given part which is supplied to the projects.

Details

The `tables.txt` file

The `tables.txt` file contains the names of the tables and, for each table, the column names and types, and the values in the rows of the table.

The table name is signaled by the heading "Table name:" and appears either on the same line with this heading or on a line below it.

The column names and types are headed by "Table fields:". They are a list of column name–SQL type name pairs, each pair on a different line, with the SQL type name in parentheses. The first name–type pair may start on the same line as the header.

The values are headed by "Table values:" and are a list of lines, each line containing the value of each column specified under "Table fields". Thus, all lines in this group contain the same number of entries, which is also the number of name–type pairs in "Table fields".

If you have read the file and performed the JDBC operations correctly, you should have exactly the same tables as in the SPJ database of DIT630 and our lecture 13.

The program should still work correctly if the file contains a different number of tables, or different numbers of rows in each table. No unnecessary hard-coding of values!

The user interface

The following description is indicative rather than compulsory. Feel free to design your own interface, but the functionality mentioned here is mandatory (i.e., no matter what your interface looks like, it should still allow choosing tables, displaying rows, etc.). A couple of important requirements are signaled as such.

Once the `tables.txt` file has been read in and the database has been created, the user should be presented with the choice between the possible operation of groups 2a and 2b in the short description.

For example, you could have a menu with items "Choose table" and "Query". Another menu could offer the choice between "Display row", "Delete row", or "Insert row".

If "Choose table" is selected, then a dialog is opened to retrieve the table name from the user. The chosen table then becomes the table that the options of displaying, deleting, or inserting a row apply to.

If one of these options is chosen from the corresponding menu and no table has been selected, the user should receive an error message.

If the user chooses one of the options "Display row" or "Delete row", then the (primary) key for the row should be retrieved, for example via another dialog. The user is informed if the key is not valid. If the key is valid, then the operation is performed. Displaying a row should display all the columns in the row. **Important:** it should be possible to select the row with the mouse, copy it, and paste it somewhere else. That means that a choice of, e.g., a plain `JLabel` to display a row is unsuitable.

If the user chooses to insert a new row in the table, then he should be presented with an appropriate interface. For example, a text field that is "long enough". **Important:** it should be possible to see a row and insert another row at the same time. For example, the user might want to copy values from the displayed row into the one being prepared for insertion.

Selections from the "Query" menu should lead to a dialog in order to retrieve the relevant information (e.g., part name) and then a reasonable displaying of the result. The user should be informed if the information entered is invalid and given the possibility of entering it again (no crashes!).

Grading

All 3 assignments for the course are mandatory and must be handed in. Each assignment will be graded and will be given a passing grade (G) or a failing grade (U). If an assignment doesn't fulfill the requirements when handed in, you will be given two weeks to correct it. However, if the assignment has major flaws when first handed in (e.g. is far from being complete) it will given a failing grade. Returned assignments will be given back with comments on what must be done to get a passing grade. If the assignment doesn't fulfill the requirements after a maximum of 2 re-submissions it will be given a failing grade (U). For assignments with a failing grade the next opportunity to do the assignment will be the next time the course is given.

Submission

The assignment is to be completed individually or in pairs.

Assignment 2 must be uploaded to the GUL system before 17.00 on November 7th and it must have been given a passing grade before November 28th, 17.00.

Only java source files should be uploaded, no class files. Your java files should be put in a zip-archive with the following name:

`assign1_author1_author2.zip`

where author1 and author2 are the surnames (family names) of the group members. If you solve the assignment by yourself only one name is necessary. You must also supply a README-file containing the names of the authors and their social security numbers. All comments etc. will be posted on GUL.