Homework 11

Due: 12 Nov 2007

All the setup for this homework is written up in Homework 9. Your code from that assignment is your starting point for this one.

What you should have so far:

- A solid database design that well represents the structure of your data at the granularity you care about
- A relational schema that implements that design
- An implementation of a minimal set of SQL updates and queries required to interact with the database
- A crude GUI to run those updates and queries

It is the last two items that should draw your attention: the SQL is minimal, and the GUI is crude. One or the other of these should be fixed for this assignment. Which one is up to you.

If you decide to improve the UI, turn to page 2.

If you decide to increase the complexity of the data manipulation, turn to page 3.

UI improvements

To actually give yourself a UI worthy of the name, you're going to need to upgrade from an applet to an application. The two options for that would be AWT—which would let all your widgets continue on unchanged—or Swing. Swing does require you to change your widgets (in most cases, just preceding the widget classname with a J: `JButton`, `JTextField`, etc), but is somewhat more powerful and high-level; also, Swing has more resources available online in terms of implementation and tutorials. If you do choose this track, I recommend using Swing.

When you graduate from applets to applications, you get to take a lot more control over many things. The flipside of this is, you *have* to take more control; just getting a window on the screen requires more work than before. The class responsible for controlling the window is `JFrame`; your topmost class will likely extend `JFrame`. Getting the application started, then, will once again involve our friend the `main` method: when run, it will instantiate (some subclass of) `JFrame` and set everything in motion.

We've talked at some length about the principles of UI design, and in this track you'll put them to use. Based on your knowledge of the domain and the target audience, fix the user interface to be not lame. That's a very broad charge, but keep in mind that you *aren't* really changing the database manipulations being done. You won't really have access to complex queries, which somewhat limits the scope of what you can do there. Rather, you'll be using the GUI library to present the data in a more useful arrangement and let the user enter information more conveniently.

Part of this track involves you going out and learning more about the GUI library you're using. There are elements we've not seen or discussed, that might make your design easier and/or better. Use them!

Turn to page 4.

Query/update/report improvements

In the Homework 9 version of the project, all the queries are simple select/where lookups, which makes all the data theoretically accessible but may require the user to do a lot of work to assemble the answers to the sorts of questions they really ask about their data. For this track, you should remedy that problem.

Part of the reason I wanted you to pick a domain that you knew and cared about is that I wanted you to be familiar with the sorts of data manipulation you'd want to do with it. You may even want to make a feature that generates some sort of report format. These will all require some rather more involved data manipulation than simple select/where queries.

If you choose this track, you will have to make some changes to the UI, say, adding a button to trigger a new query, but other than that you should not worry about the aesthetics of the thing. The main focus here will be in writing code that, via the Java GUI, builds complex SQL queries, interfaces with the database, and processes the `ResultSet`. While the general charge is somewhat broad, there is a certain limit to the scope of the project in that the complexity of your data should not be too great, and so there's only so much you can do. On the other hand, you do need to do *something*, and one or two new queries that are just a projection or something are not going to cut it.

NB: You should *not* be using Java to manipulate the data here. Obviously, one way to "solve" the problem would be to read all the data into hand-built Java data structures and process it, but that would be no fun and missing the point. The only thing Java should be doing here is pulling stuff out of a `ResultSet` and sending it to the GUI or a file.

Turn to page 4.

Submission, demo, and evaluation

Your main submission should include:

- the source code itself,
- instructions on how to compile and run it,
- sample data,[1] and
- an explanation of the improvements.

The explanation should probably be at least a half-page or so; for the UI track it should appeal to specific UI principles and goals in detailing why you made the choices you did, and for the SQL track it should clearly explain what each of the new query/updates does and why, in this domain, that should be relevant.

The sample data will be important for me to test your code myself, but ultimately the best way for me to see what you've done is for you to show me. As a result, I've reallocated the last day of class to demoing this project. You'll each have five minutes or so to show off what you've done to me and to the rest of the class.

This project will be graded out of twenty points, i.e. the equivalent of four regular homework problems, or ($\times 4$). Of that, five points will go to the explanation and the rest to the design and implementation.

[1]Note that the sample data doesn't need to be *generated* by your GUI program; particularly if you are doing the SQL track, generating the db data this way could be rather more painful than typing it through `sqlite3` or even rolling your own one-shot JDBC program to read a text file and dump it to the binary database.