CMSI 370

INTERACTION DESIGN

Fall 2006

Assignment 1026

At this point, you should have most of the information needed to put together an end-to-end Swing user interface for some application. For this assignment, we'll try to do just that.

Not for Submission

Read Chapter 7 from Shneiderman/Plaisant (we'll go back to Chapter 6 later).

For Submission

Design and implement a Swing user interface for a standalone Swing application, using the interaction design metrics, guidelines, principles, and theories discussed in class so far to inform the design decisions that you make for the application.

Since the focus of this assignment is interaction design, and not the application itself, it helps to have pre-existing code on top of which the user interface can be created. For the so-called *domain objects* and *business logic* in your application, you have a number of choices.

Code from a Previous Class

Use code that you have written in a prior course, such as 185, 186, 281, etc. For this option, devise something interactive for the user to do, refactoring your code as needed to accommodate this. For example, with the "schoolgirl problem," you can have the user build a set of "schoolgirl combinations" which your program can then check for validity (e.g., does the series of combinations adhere to the constraints of the problem?). Other options include a card-related program using "deck-of-cards" code, or an interactive "make change" program that allows different sets of denominations.

Profs. Dorin and Toal might still have your old code, in case you no longer do. You may use someone else's code as a basis; just make sure to give proper credit.

You'll get **extra credit** on this option if you build *two* user interfaces over the same code (where the second interface doesn't have to be Swing) — this will show that you have cleanly separated the domain objects and business logic of the program from the presentation and interaction components.

Code from the NSF Recourse Project

The domain objects and some of the business logic for the NSF Recourse project may be used if you wish; the general application that you would be able to build with this code is something that can manages a database of courses, students, faculty, assignments, and solutions. The code isn't entirely finished, but there's enough to build something nontrivial. **Extra credit** if you: (a) figure out how to get the code to use a live relational database and (b) provide a GUI for specifying database settings.

"Deal or No Deal"-Like Game

I have the building blocks for a game that resembles the *Deal or No Deal* game show. The code implements the data structures needed for the game, but not the stages of the game nor an automated banker. Think of it as having a "board game" version, with human beings still needing to manage the game's mechanics. The code primarily keeps track of the objects in the game, such as the suitcases and the dollar amounts that they hold.

A pure "board game" implementation will be fine for this assignment. You get **extra credit** if you cleanly extend the existing game logic and user interface to: (a) manage the number of suitcases to be opened before the banker's next offer comes up, and (b) manage the banker's offers themselves.

Roll Your Own

You can also write something totally new — but keep it simple! You don't have much time to do this assignment, though it does coincide with the undergraduate holiday on purpose. Don't forget, the focus of this assignment is designing and building a user interface, so don't kill yourself with fancy AI or complicated logic. You get **extra credit** if your design shows clean MVC separation.

Commit the program to CVS, under */homework/cmsi370/fullswing*. As usual, include an Ant *build.xml* file, and tag the submission as *hw-1026*.