CS 638 Web Programming, Fall 2007 **http://pages.cs.wisc.edu/~cs638-1/**

Assignment 2: Database interaction

Assigned October 16, 2007 *Due: October 23, 2007 11:59 PM*

1 Overview

The second assignment will involve C# programming and interaction with the database. You will have to extend the "event-driven" console application provided on the class web site to implement the functionality described below by adding a single extra .cs file with your code. You will add new commands and implement the handlers executed in response to the user entering these new commands. You will write these handlers whose role is to interact with the user to get the arguments of the commands and to interact with the database using ODBC.

2 Commands for interacting with the database

You should implement **`open`** and **`close`** commands for opening and closing the connection to the database. The arguments of the **`open`** command are the driver, the server name, the server port, the database name, the user name, and the password. The **`close`** command has no arguments. You should not allow the user to call **`open`** twice in a row without calling **`close`** between them.

You should implement **`listfromstate`** and **`classroster`** commands listing the students from a given state, and those who took a given class, respectively. **`listfromstate`** takes as argument the name of the state, and **`classroster`** the semester, the year and the catalog number of the course the roster is for. Each of these commands should give the first name, last name and student id number for all students listed, with each student being listed on a separate line.

You should implement the **`transcript`** command which takes as its single argument a student identifier. It should list the catalog numbers and names of all the classes the student took, the semester and year in which the student took them and the grade with each class.

You should implement the **`updategrade`** command which updates the database. It takes as arguments the student identifier, the catalog number for the class, the year, the semester and the new grade. Both these commands should display the new data to be entered in the database and ask the user for an explicit confirmation before updating it.

3 User interaction

The user should enter arguments after the command, one by one, each on a separate line. You may allow the user to enter the arguments on the same line as the command, but this is not a required feature.

When prompting the user for the value of an argument you should always make it clear what the user is entering by displaying an informative prompt. If a default value is available for the piece of data the user is entering, you should use the default value if the user doesn't enter anything. The default value should be displayed between square brackets when the user is prompted for input. See the Greeter.SetName method from ExtensionExample.cs for an example of how user input with default values can be handled. A few arguments will never have default values; some will not have default values until they are used at least once.

For all parameters of the **`open`** command you should add pre-set default values in your code and update them whenever the user selects another value. These pre-set defaults should connect to your copy of the database with your user name and password. No other arguments should have pre-set default values in your code. You should keep as default student id number the one last used in a **`transcript`** or **`updategrade`** command. You should keep as default catalog number, semester, and year the values last entered by the user in a **`classroster`** or **`updategrade`** command.

Hint: Defining one or more separate classes and/or methods for handling user interaction and defaults can simplify your code. Not doing so may lead to many bits of code doing the same thing in various places.

4 Application structure

You will extend an existing program by adding new commands and the corresponding handlers. Your methods may throw exceptions. Consult the ExampleExtension.cs file for an example of how you can add new commands and handlers.

3 Grading criteria

The grading criteria for this assignment will include: the extent to which all the required features are covered, correctness, the use of parameterized queries, and proper use of classes and methods to structure your code.

4 Submission

Please email as attachment to a single message to the TA the single .cs file implementing your assignment.