Module 3 (XML Persistance) Specific Instructions

Module 3 will build on the structure initially developed in Module 2.

For Module ***3,*** you will develop a mechanism to *persist* the data between executions of your programs. There are obviously lots of ways to store information, but in this course we will use XML to store the *metadata* that describes Tables. All information entered using DEFINE, RENAME, or DROP will need to be saved in an XML file of your design, so that the information is persistent over different executions of the program. I suggest you use SAX to manipulate your XML file. If the user enters any of these statements, the structure of the tables available to the database will of course change, and you will have to completely rewrite the XML file with the new data. As a design decision, you will need to decide when you want to rewrite the XML file. Do you want to do it every time one of these commands is entered? Or do you want to do it only when the program terminates? There are advantages and disadvantages with each mechanism, and this is a design decision that you will need to make. By the way, to clarify, you only need to track the currently defined tables – once a user DROPS a table, you can just eliminate it from your XML file – there is no "UNDO" provision for recovering changes to the tables.

This unit will require continued use of javadocs unless I have specifically told you I am not reviewing yours anymore. You will also need to write JUnit Tests for this module. Remember, use JUnit to test the "building block" methods from you program, and that the methods that you decide to test must return something so you have something to test. So I would write methods in my WriteXML routine that create Strings which are the components of the xml file you are creating and then your test should make sure they are accurate. I would also test the xml reading – during the "Before" phase of the testing I would write out an XML file, then to test I would read it and make sure your result (ie, collection of tables) is what you expected it to be.

Module submission should include appropriate JUnit tests for your code as well as javadocs as appropriate, and source code, class files, and docs should be submitted by email in a functioning jar file. Make sure SVN is up to date with your current code. Be sure to include your mixID in your email submission.