CST8221 – JAP Assignment/Lab 13 – Java Database Programming

Objectives

The purpose of the lab is to give you a hand-on experience of how to write Java Database Applications using JDBC.

The Nature of Things

In this lab you are going to go through the process of creating a traditional single-tier client/server Java databases application using JDBC and JavaDB (Apache Derby) database. All instructions assume that the database software has been already installed and the installation of the database software is located in **c:\Program Files\Java\jdk1.8.X_XX\db** folder. You must verify the database location and the Java JDK version of your installation, and then you **must make an appropriate adjustment to all of the provided batch (*.bat*) files**.

VERY IMPORTANT NOTE:

If you do not find **db** folder in your **jdk1.8.X_XX** installation, download the db.zip file from Brightspace, unzip it, and copy (as Administrator) it in your **jdk1.8.X_XX.** You can also copy the **db** folder at any other location and **make the adjustment to all of the provided batch (*.bat*) files** Once you make the adjustment you can proceed with the steps outlined below

Step 0 – Download and install the required files.

Download the **CST8221_Lab13_code.zip** file. Extract the contents in the root of your hard drive. After the extraction you should have a folder **JAP_JDBC** with the following folders inside:

```
DB
ExecSQL
QueryDB
TestDB
```

Compile all the provided java code files in their corresponding folders. The installation is complete and you can proceed to Step 1.

Step 1 – Starting the database

Open a command window and make the **DB** folder current. Run the **startdb.bat** command batch file.

You must receive a message saying "`Apache Derby Network Server … started and ready to accept connection on port 1527...`"

If you see this message, go to step 2. Do not close the command window.

Step 2 – Checking the connection and creating a database schema

Open a command window and make the **DB** folder current. Run the **checkdb.bat** batch file. The following must appear on your screen:

```
CONNECTION0* – jdbc:derby//localhost:1527/BOOKSTORE;create=true
* = current connection
 ij>
```

At the prompt, type **`EXIT;`** Note: Use capital letters and do not forget the semicolon at the end. The command prompt should return and you should have a folder **BOOKSTORE** created in your **DB** folder. Close the command window. Move to next step.

Step 3 - Testing the database operation

Open a command window and make the **TestDB** folder current. Run the **testdb.bat** batch file. The following should appear on your screen:

```
Hello, Database World!
```

If you see the message, your database connection is ready for work.

Open the TestDB.java file and explore the code. We will discuss the Java code of this file in class. The class loads the database connection parameters (JDBC drivers name, url, username, and password) from a file named ***database.properties*** and tries to connect to the database using the ***getConnection()*** static method of the ***DriverManager*** class. After the connection is established, the program executes several SQL statements using a statement object obtained from the connection object. Then it executes a query and gets the result from the query into a ResultSet object. It prints the query, and, finally, it removes the table by executing the DROP SQL statement.

If you have problems, you must open a command window, make the *DB* folder current, run the *stopdb.bat* batch file, delete the *BOOKSTOR*E folder, and repeat steps 1 to 3.

Step 4 - Creating and populating the BOOKSTORE database tables

In a command window, go to the ExecSQL folder current. At the command prompt, run the following commands in order:

```
createtb Authors.sql
createtb Books.sql
createtb BooksAuthors.sql
createtb Publishers.sql
```

After each command is executed, you should see the content of the corresponding table on the screen.

If the operation is successful, your BOOKSTORE database is ready for use.

Explore the code and the .sql files. The program reads SQL instructions from a .sql file and executes them using JDBC It uses the generic *execute()* method of the *Statement* object. If it returns *true*, the statement has a result set. If there is a result set, the program prints the result. Because this is a generic result set, the program uses a *ResultSetMetaData* object to obtain information about the result. If there is any SQL exception, the program prints the exception and any chained exceptions that may be contained in it. Finally, the program closes the connection to the database.

When you are ready, move to the next step.

Step 5 – Querying the BOOKSTORE database

In a command window, make the QueryDB folder current. At the command prompt, run the **querydb.bat** command file.

A GUI with a title *QueryDB* should appear on your screen with both combo boxes displaying **AnyAuthor** and **AnyPublisher**, and a table populated with **AnyAuthor** and AnyPublisher query result. Choose an author from the first combo box list. Leave the publisher as **Any.** Click the **Query** button again**.** See the results**.** Next**,** select a publisher and type the price change amount into the text field next to the **Change prices** button. When you click the button, all prices of that publisher will be adjusted by to the amount entered (the amount will be added to or subtracted

from the current price). The status line at the bottom will display the number of records affected by the change. Run a query to verify the changes to the prices.

Explore the code. The program executes queries against the BOOKSTORE database. It also allows you to change the prices of the books for a specific publisher. The program is using JTable and JTableModel to display the query results. Try to understand how the code works. In your textbook you have a similar example that can help you to understand the lab code.

The program uses a new feature called ***prepared statements***. Rather than building a separate query statement every time the user launches such a query, you can prepare a query with a ***host*** variable and use it many times, each time providing a different string for the *host* variable. Each host variable in a prepared query is indicated with a **?**. If there is more than one host variable, then you must keep track of the positions when setting the values. Before, executing the prepared statement, you must bind the host variable to actual values with a *set* method. For example,

```
publisherQueryStmt.setString(1, publisher);
```

sets the string to a publisher name.

To implement database updates the program uses the executeUpdate() method. This method does not return a result set. It returns the count of the changed rows of a table.

If you have reached this point you concluded the lab exercise successfully. You must submit your code to Brightspace. I will test it by launching your code myself.

Note: When you are finished running the software, you should stop your database. To do that, open a command window, make the DB folder current, and run the **stopdb.bat** command file.

If you want to learn more about Java DB and download the technical documentation, visit the following link: http://docs.oracle.com/javadb/

Marks: 2% of your course mark