CSCI 470/502 -- Assignment 6 -- Spring 2023

Note: This is the last assignment and no late submission is allowed.

Description

This project will implement a JDBC program. Your program will have to run on turing/hopper in order to be allowed to connect to a mysql/MariaDB database server installed in the CS department. You will not be able to connect directly to the database from an external machine (e.g. your laptop) due to security restriction.

You may edit/compile your program on your laptop, then ftp the program to turing for running and testing.

You will work with a designated database, base on the first letter of your last name (or the last name of the student who does the submission if you work in a group):

> *A-C: JavaCust01; D-F: JavaCust02; G-I: JavaCust03; J-L: JavaCust04; M-O: JavaCust05;* P-R: JavaCust06; S-Z: JavaCust07;

The spare database JavaCust08, JavaCust09, and JavaCust10 can be used by anyone for testing.

In your program, you should set your designated database as the default database name. Additionally, you should enable the program to receive the database name as a command line argument. When your program is running, it will also be receiving user input through command line. See Example Output later. Note that command line arguments are different from user inputs received through the command line (System.in).

Each database contains a single table whose name is **cust**, so this is the table name you should specify in all of your SQL queries (e.g., select * from cust where name = 'nameString'). The cust table has 4 columns. These are:

Column Number	Column Name	Data Type
1	name	char(20)
2	ssn	char(10)
3	address	char(40)
4	code	int

The ssn column has a unique primary index on it and thus all values in this column must be unique. All table name and column names are lowercase and must be coded as such in SQL statements.

These databases/tables might not be empty, so your first transaction type to implement might be GETALL (see below). It might also be the easiest among all operations without the need of user input.

You should perform the following operations on the database.

ADD a new customer. The user enter information of the 4 fields. The program needs to parse the input. If there is any error, e.g., one or more fields are blank, display an error message. The program should insert this new information into the database and inform the user of its success or failure. Note that attempting to insert a record with a duplicate ssn will raise an SQLException.

DELETE a customer. The user provides an ssn. The program should attempt to delete the record with that customer's ssn and should inform the user of its success or failure.

UPDATE a customer **address**. The user has entered an ssn and the new address. The program needs to parse the input. If there is any error, e.g., one or both fields are blank, display an error message. The program should change the address for this ssn (if the ssn is found) and report to the user on update success or failure. Specifically, you are asked to use a ***PreparedStatement*** for this action, which means your program should just prepare and execute a PreparedStatement, report on whether it succeeded of failed. Only **one** database transaction. (It should **not** use two transactions by first reading the existing record to ensure that the ssn exists in the database.)

GETALL - the program should retrieve and display all the database records. These records need not be neatly formatted - just dump the raw information, one record per line.

Example input/output:

Start with a simple menu. The blue fonts are from the program. The black fonts are user input.

turing:~% java JDBCAssign JavaCust02

The program is now successfully connected to the database JavaCust02.

1. ADD
2. DELETE
3. UPDATE ADDRESS
4. GETALL
5. EXIT

=>Choose the action: 1

Please enter [name;ssn;address;code]: Jane Smith;123456789;800 Dekalb Ave;112

The customer is successfully inserted into the table.

=>Choose the action: 2

Please enter [ssn]: 120003456

The customer is not found for deletion.

=>Choose the action: 3

Please enter [ssn; address]: 123456789; 666 Dekalb Ave

The customer's address has been updated.

=>Choose the action: 4

Jane Smith;123456789; 800 Dekalb Ave; 112

Hint: In order to get user input for some action, such as action 1, and implement the aforementioned user interaction, it may be necessary to call nextLine() on the Scanner object twice, depending on the method employed. For example, if nextInt() was used to retrieve an integer choice of the action, the first call to nextLine() would be to bypass the new line from user's selection for an action, and the second call to nextLine() would be to capture the user input for the information.

Development and Testing Details

For information on JDBC and development details, see "JDBC notes", which also includes details on how to set the necessary classpath.

As mentioned earlier, you can't connect directly to the database from an external machine (e.g. your laptop) due to security restriction (database ports are notorious hack points). One alternative would be writing a client-server application: Let a client on your laptop connect to a server running on turing. And the server is the one that does JDBC. Such setup is *not* required for this assignment. If interested, you may explore on your own using Java networking API.

Submission

Submit your source code in Blackboard similar as before. Make sure you name the zip file as required.