CSCE 4523 Database Management Systems
Homework 4
Due: Tuesday, March 31st at 11:59pm
By: Amber Yates #010757937 and Garret Gardenhire #010715713

Objective

The objective of this homework is to create a program in Java or C++ that is capable of interacting with an SQL database by using ODBC or JDBC. This program must be capable of inserting, deleting, and updating the database through a command-line database. The data used for this assignment will be the same tables regarding clients, agents, and policies that were used in homework 3. This database does not have to be created by this program, but if it does then it must do so via the "init" function.

Approach

This program was built using Java where the user interface was implemented through a switch function that read in user input though the Java Scanner module. Each menu choice was handled through a different java method that would manipulate created SQL tables according to the user's input. The tasks of this project were split among the partners, worked on separately, and then brought together after communicating through google hangouts or text messaging.

The mysql database manipulated by this program is accessed through JDBC, which stands for "Java Database Connectivity driver", which requires a username and password for a specific mysql user to connect. This information can be manipulated by changing the strings present in the main function of HW4.java, as the group elected not to prompt the user for their information to adhere to the requirements of this project.

Sample code from the Database Class website was utilized for the skeleton of this program in order to better understand how the program needed to manipulate SQL tables. This code is broke up into six sections:

1) Find all agents and clients in a given city
2) Add a new client, then purchase an available policy from a particular agent
3) List all policies sold by a particular agent
4) Cancel a policy
5) Add a new agent to a city
6) Quit

These sections were implemented in their own functions within the HW4.java class. Some of these functions were even broke down into smaller methods, For example, #2 has many sub-tasks it needs to complete so it runs across several methods within the class. Each method except for "quit" creates a string query of SQL syntax that is then sent to a method called query(), that will then apply the created SQL statement to the databases attached to the logged in user.

In functions that required adding an entry to the SQL tables such as in function 2 and 5, the max ID in the respective tables were found, and then increased by one to get the new entry's ID. This method is not the best way to get a unique ID as when the last entry is deleted a new entry with the deleted ID is added, and, depending on the use of the database, this may not be ideal.

Results

This program was able to fulfill the requirements stated on the Database Class website. To keep the user in the desired function, while loops were used to ensure the user entered a value that is applicable to the SQL tables and that the values were formatted correctly. When a user's input was too large, the program truncates their inputs to ensure a matching type in the SQL table. Similar methods were used to make sure the user inputted valid policy numbers when they already specified the type of policy they wanted to purchase, and to make sure all zip codes were numbers.

There is one issue, where if the user inputs a character or string for the main menu input, there will then be an infinite loop issue. However, for this assignment, it was assumed that the user would properly use the program. In a real world application, this program would need to be modified to make sure invalid inputs would not result in an infinite loop.

A minor adjustment could be made to the user entries for uniformity. The queries were not case sensitive, so no adjustments were made to force capitalize the user entries to match the original tables.

The work of the project was delegated to the following persons:

Main - Amber Y. and Garret G.
1. Find all existing agents in a given city - Amber Y.
2. Purchase an available policy from a particular agent - Amber Y.
3. List all policies sold by a particular agent - Amber Y.
4. Cancel a policy - Garret G.
5. Add a new agent for a city - Garret G.
6. Quit - Garret G.

JDBC connection - Provided
Report - Amber Y. and Garret G.

Testing

Each function was tested to ensure the main feature worked as they were implemented. Once each function was created, more specific tests were done, such as removing empty queries and making sure users weren't brought back to the menu for a mistake.

Many tests focused on making sure that only the correct data for the SQL file could be accepted. The main requirements were each string could only be 50 characters long, zip codes had to be integers, and the amount purchased had to be a float value between 0 and 10,000. During the final refinements, the code was modified to allow string entries to accept spaces, while number values still could not.

In the script file detailed test cases were shown such as finding clients and agents that don't exist, attempting to delete a policy that doesn't exist, and demonstrating that there are while loops in place so that the user is asked for a value until they enter one that is correct.