CS 348 - Project 3

Due: Sunday April 7, 2019 at 11:59PM on Blackboard.

(There will be a 10% penalty for each late calendar-day. After five calendar days, the homework will not be accepted.)

Instructions:

- Please read all the information below as they are very much essential in guiding you through the project.
- Grading is automated. So please follow exactly the given input/output format, which includes spaces, new lines, and case (capital vs. small letters) taken into consideration.

Project Description

In this project, you will implement a simple access control scheme (role-based) for the given database schema. The project has two goals:

- Using JDBC to query/update your database from within a Java program
- Be able to simulate an admin-user interaction where the admin creates roles and grant privileges to users.

Details of the Project

Access Control Mechanism

In this project, you will simulate an admin-user interaction, where the operations a user can perform on any data are determined by the user's access roles. This is popularly called Role Based Access Control Protocol. Here we see that user can perform a said operation on a said table only if he has the access right to perform that operation on that table. Users can have multiple roles and are entitled to all privileges of their assigned roles. Here, users of the database are assigned roles by a single administrator, who also assigns specific permissions regarding the use of the database tables to each role. The operations regular users of the database can perform on the data are restricted to INSERT and SELECT for this project. There is an OwnerRole column in all the user tables and when a SELECT command is issued and a user has the SELECT privilege for that table, he/she should still not be able to see the OwnerRole column.

Connecting to your Oracle database

For this project, you will use your Oracle account to host the database of the application. Your program should connect to this database using a database connection library (JDBC). To get started using JDBC, you can go through the tutorial at http://www.tutorialspoint.com/jdbc/jdbc-quick-guide.htm. This tutorial should provide the sufficient knowledge about JDBC for this project's purposes.

Note that when registering the database driver in your code, you should pass the argument **"oracle.jdbc.OracleDriver" to the Class.forName() method**, as you will be connecting to an Oracle database. Also, you will need to include the JDBC driver **ojdbc8.jar** (provided to you) in your classpath at runtime to be able to use the database connectivity function.

You should use the URL **"jdbc:oracle:thin:@claros.cs.purdue.edu:1524:strep"** with the getConnection method of the DriverManager class when connecting to the database and **include your Oracle account username (without @csora) and password** as the username and password parameter values.

Database Schema

The database in this project consists of the tables listed below. The scripts to create/drop the database are already provided to you.

Admin tables

These tables are for administrative tasks only and are not accessible by regular users: Users, Roles, UserRoles, Privileges, and UserPrivileges.

User tables

These are accessible by regular users. The schema uses a hospital environment and the tables are named in that way. The users will need to have the appropriate permissions to insert data into or view data from these tables. Note again that **the OwnerRole column should not be output when a SELECT command is issued by any user**.

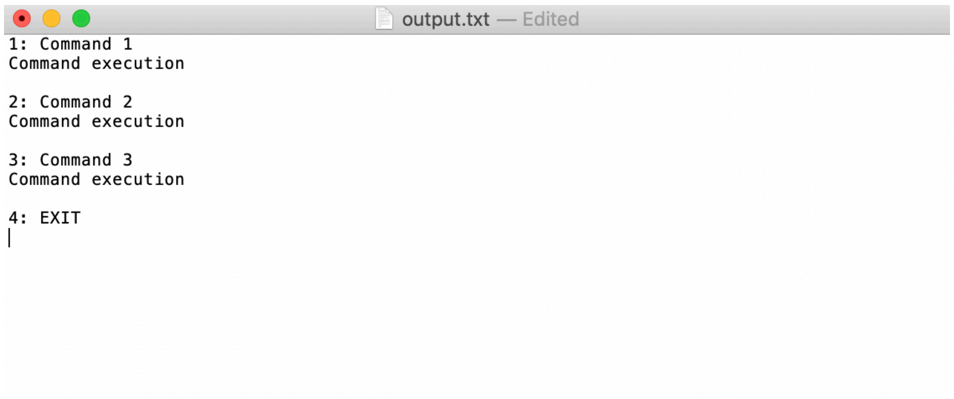
Note: You can create your own corner testcases to further test your code.

Command Set

Your program should accept input from a text file (with each command terminated with a newline character), process the input and produce as output the responses in another text file. Ultimately, your submission should be executed as follows:

```
java -cp .:ojdbc8.jar Project3 input.txt output.txt
```

Sample input and output files are also provided for you. input.txt is the file containing the commands and output.txt is a file you are going to create. output.txt should include the output for each command separated by newlines, and ends with a newline after the “EXIT” command. For example, if we have three commands in input.txt before the "EXIT" command, output.txt should look like



```
1: Command 1
Command execution

2: Command 2
Command execution

3: Command 3
Command execution

4: EXIT

```

SAMPLE OUTPUT.TXT

Notice that "Command 1" is the actual first command text you read from input.txt. Following any other format will result in a problem and eventually delay in grading.

The program should implement the command set listed below. Note that the commands will always be input in the correct format, i.e. you do not need to check for syntax errors.

- **`LOGIN`** username password

  When this command is issued, you need to check that the provided username and password pair matches the one in the Users table of the Admin tables in the database. If not, you should print the error message “**Invalid login**” (use this format exactly). If a match is found, the current user should be made to the one specified by this username and also print the exact message “**Login successful**”.
  Note that you should be having a record in your Users table for the admin user (UserId=1, Username=’admin’, Password=’password’). You should also be having a record for the ADMIN role in the Roles table (RoleId=1, RoleName='ADMIN'). You should have a record in your UserRoles table, for which the UserId is 1 and the RoleId is 1 (this means that the user admin has role ADMIN). A script with this data initialization is also

provided with this handout. While checking whether the current user is the admin, you can either check that the username is admin or that the user has the ADMIN role in the UserRoles table (either approach will fetch points). Note that if the login command is issued before a user quits the program, you should just switch the current user to the user specified in the login command. You can assume that if the login is unsuccessful, the user will keep trying to login until it succeeds. **The very first command issued to your program will always be the login command.**

- `CREATE ROLE` roleName

  If the current user is the admin when this command is issued, you should insert a new record in the Roles table with the values (roleName) and print "**Role created successfully**" (generate role Id dynamically). If the current user is not the admin, then print the error message "**Authorization failure**".

- `CREATE USER` username password

  If the current user is the admin when this command is issued, you need to insert a new record in the Users table with the values (username, password) and print "**User created successfully**" (generate user id dynamically). If the current user is not the admin, then print the error message "**Authorization failure**".

- `ASSIGN ROLE` username roleName

  If the current user is the admin when this command is issued, you should insert a new record in the UserRoles table with ids corresponding to (username, roleName) and print "**Role assigned successfully**". If the current user is not the admin, print the error message "**Authorization failure**".

- `GRANT PRIVILEGE` privName `TO` roleName `ON` tableName

  In the already populated Privileges table (with two rows)
    - Insert Privilege: Id = 1, Name = INSERT
    - Select Privilege: Id = 2, Name = SELECT

  If the current user is the admin when this command is issued, you should insert a new record in the UserPrivileges table with values corresponding to the parameters (roleName, privName, tableName) and print "**Privilege granted successfully**". If the current user is not the admin, print the error message "**Authorization failure**".

- **REVOKE PRIVILEGE** privName **FROM** roleName **ON** tableName

  If the current user is the admin when this command is issued, you need to delete the record in the UserPrivileges table with values corresponding to the parameters (roleName, tableName, privName) and print "**Privilege revoked successfully**". If the current user is not the admin, print the error message **"Authorization failure".**

- **INSERT INTO** tableName **VALUES** (valueList)
  GET ownerRole

  Upon issuance of this command, you should first check if any of the roles of the current user has the "INSERT" privilege on the table `tableName`. If not, you should print the error message "**Authorization failure**". Otherwise, you should insert the values in valueList (which is a list of comma-separated strings enclosed in single quotes) into the table tableName. While inserting the tuple, you should set the OwnerRole attribute value to the id corresponding to ownerRole. Your output should be "**Row inserted successfully**" if the user is authorized.

- **SELECT** * **FROM** tableName

  Upon issuance of this command, you should first check if any of the roles of the current user has the "SELECT" privilege on the table `tableName`. If not, you should print the error message "**Authorization failure**". Otherwise, you should print the names of the attributes in this table (in upper case) on a line by itself, each separated by a comma and all records in the table `tableName`, one record per line, with each attribute value separated by a comma (attribute values in the same order as the attribute names listed on the first line). Note that OwnerRole should NOT be printed to output. You can assume that there will be at least one record in the table tableName when this command is issued.

- **EXIT**

  Your program should terminate upon issuance of this command (you should ignore any command that appears after the EXIT command in the input file).

Submission Instructions

Please follow these instructions strictly.
Grading is automated, and your grade would be significantly affected otherwise.

- The project should be implemented in Java and your main file should be named **Project3.java**.

- Before you submit your project, make sure to delete all data from your database tables, and drop all the tables, except for the data in the initialization script (To guarantee a completely clean database before testing your submission).

- If needed, create a README file containing identifying information, and include anything you might want us to know when grading your project.

- A sample run.sh script is given. This is used to clean and initialize the database, and compile and run your code. Make sure you edit the file to add your oracle username and password. As shown in the script file:
  Your submission should be compiled using the following command
  ```
  javac -cp .:ojdbc8.jar Project3.java
  ```
  and should be run using the following command
  ```
  java -cp .:ojdbc8.jar Project3 input.txt output.txt
  ```

- If you add other java files, you might need to edit the run.sh file to compile these too.

- Please submit on Blackboard a ZIP file named "username_project3.zip" where username is your Purdue career account username (i.e. username@purdue.edu). The ZIP file should contain **ONLY** the following files with no subfolders:
  - **Project3.java** (Your java main file) and any other .java files you might be using
  - The modified **run.sh** file with your username and password
  - [Optional] **README**: your readme file, if needed