Lab 7: Database Connectivity with JDBC

Due date: Tuesday, June 3, **before the beginning of the lab**

Assignment Preparation

This is a team lab. You can organize yourselves in teams of 2-3 people each. Once you build your team, one member of the team needs to email me at dekhtyar@csc.calpoly.edu. The email message should have the header ``CSC 365: Lab 7". In the body of the message, please put the names and emails of all team members.

Due to the nature of the lab assignment, it makes sense to stay in the same groups as for Lab 1. However, this is not a requirement, new group arrangements may be used.

Lab Assignment

In a Nutshell

Each group is to build a Java JDBC application that implements a variation of the Lab 1 project.

Notice, that for this lab, you will be using the data from the STUDENTS dataset! (and not the data from the Lab 1 assignment).

The Task

As in Lab 1, you are given a list of students of a local elementary school together with their class assignment. The STUDENTS dataset available from the course web page contains the data files. (DO NOT use the .txt files provided to you in Lab 1). Recall that the dataset contains two files, `list.csv`, storing information about students, and `teachers.csv` storing information about teachers.

The format of `list.csv` is

```
StLastName, StFirstName, Grade, Classroom
```

A sample line from `list.csv` is

```
DROP, SHERMAN, 0, 104
```

("Sherman Drop is a kindergarden student assigned to classroom 104.")

The format of `teachers.txt` is

```
TLastName, TFirstName, Classroom
```

A sample line from `teachers.csv` is

```
NIBLER,  JERLENE, 104
```

("Jerlene Nibler teaches in classroom 104.")

Your goal is to design and implement the `schoolsearch` program which uses JDBC to (a) connect to the Oracle server, (b) create the database (as needed) and load the dataset data into it (as needed), (c) retrieve information from the database basedon queries presented in interactive and batch modes.

Specs

You are two write a program called `schoolsearch`, which implements the requested functionality. The program shall be written in Java. The program must satisfy the following requirements.

R1. The program shall run from the command line.

R2. The program shall have two modes: interactive and batch.

R3. In the batch mode, the program is run with a single instruction, specified as command-line parameters. The program shall perform the requested operation and output the results. The command-line should look as follows:

```
> java schoolsearch <INSTRUCTION>
```

R4. The interactive mode of the program is invoked if the program is run without command-line parameters. In the interactive mode, the program shall provide the user with a prompt, read search instructions entered by the user, print the result, and wait for the next user instruction until the termination command is recieved.

R5. *[new]* The program shall use the contents of `list.csv` and `teachers.csv` files located in the same directory as the executable of the program. Unlike

Lab 1, where the contents of these files were to be read at the beginning of each run, your program shall not do so automatically. Rather, the language of instructions (see requirement **R6.**) shall be extended to include commands for populating the database.

You shall design the relational database structure for storing the contents of the `list.csv` and `teachers.csv` files. The specific design is left up to individual teams, but the database must support all functionality specified in the requirements.

R6. ***[Updated]*** The following language of instructions shall be implemented.

- `DB: T[est]| S[etup] | L[oad] | C[lear]`
- `S[tudent]: <lastname>`
- `T[eacher]: <lastname> [C[ount]]`
- `C[lassroom]: <lastname> [T[eacher]] [C[ount]]`
- `G[rade]: <number> [T[eacher]] [C[ount]]`

This language is in effect for both the batch and the intercative mode. In addition, in interactive mode, your program shall also recognize the following commands:

- `Q[uit]`
- `R[epeat]`

Notes: For simplicity all instructions are case sensitive. You are welcome but do not have to make them insensitive of the case. In the specs above, square brackets (`[]`) indicate optional parts (e.g., the `Student` instruction can be abbreviated to `S`), while items in angle brackets (`<>`) are the values provided by the user. Vertical bar `"|"` indicates that only one of the items separated by it can be used at a time.

R7. ***[new]*** When the `DB` instruction is issued, your program shall perform the followning:

- `DB: T[est]`. When this command is issued, your program shall connect to the Oracle server and check if the database storing information about students exists. It should output an affirmative message (e.g., `"Database found"` if the database exists, and a negative message (e.g., `"Database not found"`) if the database (relations) do not exist. Note that existance of all expected tables is sufficient for an affirmative response, even if the tables are not populated.
- `DB: S[etup]`. When this command is issued, your program shall attemt to create ALL tables necessary for the database representing the STUDENTS dataset. Your program should be able to detect and convey to the user the following three situations:

    - no tables from the STUDENT database existed prior to the command; the command successfully created all tables.
    - all tables from the STUDENT database existed prior to the command; the command did not result in creation of any tables;
    - some tables in the STUDENT databe existed prior to the command, but not all; the command resulted in creation of some, but not all tables.

  Note, that in each case, after this command is issued, all tables for the STUDENTS database must exist on the server.

- `DB: L[oad]`. Upon this command, your program shall attempt to insert all records from the `list.csv` and `teachers.csv` files into the database. If any of the relational tables do not exist, an error message must be reported, but insertion into other tables must proceed. As output, for each table report the number of new records inserted. E.g.,

```
> DB Load

Students table: 60 new records inserted
Teachers table: 12 new records inserted

>
```

- `DB: C[lear]`. When this command is issued, your program must delete all tables for the STUDENTS database.

R8. `S[tudent]: <lastname>`

When this instruction is issued, your program shall perform the following:

- search the database for the entry (or entries) for students with the given last name.
- For each entry found, print the last name, first name, grade and classroom assignment for each student found and the name of their teacher (last and first name).

R9.***[updated]*** `T[eacher]: <lastname> [C[ount]]`

When this instruction is issued, your program shall perform the following:

- Search the contents of the `list.txt` file for the entries where the last name of the teacher matches the name provided in the instruction.
- If the `C[ount]` option is present, output the number of records retrieved. Otherwise, for each entry found, print the last and the first name of the student.

R10.*[updated]* `[G]rade: <Number> [C[ount]]`

When this instruction is issued in the form above (without the optional `T[eacher]` suffix), your program shall perform the following actions:

- Search the database for the entries where the student's grade matches the number provided in the instruction.
- If the `C[ount]` option is present, output the number of records retrieved. Otherwise, for each entry, output the name (last and first) of the student.

R11.*[updated]* `[G]rade: <Number> T[eacher] [C[ount]]`

When this instruction is issued with the `T[eacher]` suffix, your program shall

- Search the database for the teachers who teach specified grade.
- If the `C[ount]` option is present, output the number of records retrieved. Otherwise, output the names (last and first) of all teachers who teach the given grade. Each teacher's name shall be printed exactly once.

R12.*[new(ish)]* `[C]lassroom: <Number> [C[ount]]`

When this instruction is issued in the form above (without the optional `T[eacher]` suffix), your program shall perform the following actions:

- Search the database for the entries where the student's classroom matches the number provided in the instruction.
- If the `C[ount]` option is present, output the number of records retrieved. Otherwise, for each entry, output the name (last and first) of the student.

R13.*[updated]* `[Classroom]: <Number> T[eacher] [C[ount]]`

When this instruction is issued with the `T[eacher]` suffix, your program shall

- Search the database for the teachers who teach in the specified classroom.
- If the `C[ount]` option is present, output the number of records retrieved[1]. Otherwise, output the names (last and first) of all teachers who teach the given grade. Each teacher's name shall be printed exactly once.

[1]In the current database, this will always be 1, since there is a one-to-one correspondence between teachers and classrooms. This particular instruction format is kept for consistency and to ensure that the program does not have to be changed should the one-to-one mapping stop holding.

R14.*[new]* `R[epeat]`

When this instruction is issued in the interactive mode, your program shall execute the previous query command that has been specified in the current interactive session. Here, query command is any of the `S[tudent]`, `G[rade]`, `C[lassroom]`, `T[eacher]` commands.

Examples: In following session:

```
> G: 3

  database not found
> DB: Test

  database not found
> DB: Setup

  database created
> DB: Load

  Students: 60 records added
  Teachers: 12 records added
> R
```

the last command shall result in the repetition of the `G: 3` command.

In the following session,

```
>DB: Test

 no database found
>DB: Setup

  database created
> DB: Load

  Students: 60 records added
  Teachers: 12 records added
> R
```

there is no query instruction, and therefore, `R` command should not produce any result. An appropriate diagnostic message (e.g., `"No query to repeat"`) shall be displayed.

E1.*[new]* Your program shall provide basic error-checking and handle all Java exceptions resulting from communication with Oracle graciously, without quitting and without dumping core. The following specific diagnostics must be reported to the user:

E1.1. An attempt to execute any query command (`S,C,G,T`) while no database exists on the Oracle server. A `"no database found"` or similar message must be displayed to the user. The program then shall wait for the next instruction.

Notice that if the database tables exist but have not been populated, there shall be no error messages returned — the program simply shall report that no records were retrieved.

E1.2. A failed attempt to load the database driver. The program shall inform the user that it could not load the driver and graciously exit.

E1.3. A failed attempt to establish a database connection. The program shall inform the user that it could not connect to the database server, and graciously exit.

E1.4. An attempt to create (`DB: S[etup]`) the database, when it (or parts of it) already exist(s). See requirement **R7.** for the required reactions. The program shall return the appropriate diagnostic message and, in the interactive mode, wait for the next instruction.

E1.5. An attempt to delete (`DB: C[clear]`) a non-existing database. The program shall output a diagnostic message and, in the interactive mode, wait for the next instruction.

E1.6. An attempt to load data into non-existing, or partially existing database. The program shall output a diagnostic message and, in the interactive mode, wait for the next instruction.

Note, that an attempt to load the same data again shall not be treated as an error. The report on such action shall simply state that 0 new records were added to the tables.

E1.7. An attempt to repeat (`R[epeat]`) a query before a single query has been issued in a session in the interactive mode (see requirement **R14**). The program shall output a diagnostic message and wait for the next instruction.

E1.8. An attempt to execute a command that is not recognized as a valid instruction. The program shall output an appropriate diagnostic message and, in the interactive mode, wait for the next instruction.

Note that `R[epeat]` and `[Q]uit` commands shall not be recognized as valid instructions in batch mode.

C1. To ensure that your program can run on any server and under any username, your program shall work with a connection configuration file. This file shall be named `connection.config` and shall be located in the same directory as the executable of your program. The format of `connection.config` is described below:

```
# <comment>
url = <url>
login = <loginid>
password = <password>
```

There are three settings found in the configuration file: `url`, `login` and `password`. In addition, empty lines can be inserted anywhere in the file, as well as comments. Each comment line starts with the unix comment symbol (#). Comment lines can be located anywhere in the file.

A sample `connection.config` file is shown below:

```
# CSC 365: Lab 7
#
# Sample connection.config file

# this is the connection URL
url = oracle:jdbc:thin@hercules.csc.calpoly.edu:1522:ora10g

# Login ID
login = ST44

# password
password = empty

# <EOF>
```

Your program shall read and parse the contents of the `connection.config` file, and attempt to establish all connections using the settings specified in it. Note that there are no quotation marks around the values of the `url`, `login` and `password` parameters, and there is no termination symbol. Make sure that you strip any possible whitespace from the parsed values.

Deliverables

You have to submit your code and a README file describing your implementation. There is no need to submit `conection.config` file — I will use my own to test your program.

As usual, submit all files as a `.zip` or a `.tar.gz` archive. If any specific instructions are needed to compile your program, please include them in the README file. Name the file `lab7-group<IJK>.zip` or `lab7-group<IJK>.tar.gz` where `<IJK>` are the first letters of the last names of all group members in alphabetical order.

Also, as usual, *each file you submit must contain the names of all group members.*

Good Luck!