Lab 7: Database Connectivity with JDBC

Due date: Friday, December 1st, **9:00pm**

Assignment Preparation

Please form two-person teams for this assignment. Team formation is your responsibility. If you do not find teammates due to missing class/lab period, it is your responsibility to complete the full assignment.

Lab Assignment

Each group will build a Java JDBC application that implements a simple Bed & Breakfast Inn reservation system. Implement your application with the following goals in mind: (1) minimize the amount of data transferred between your Java application and the the database (2) minimize the number of individual SQL statements you send to the database. Clearly, the data volumes in the sample dataset would permit us to `SELECT *` from every table in the database upon startup, then work entirely from memory (neatly achieving both goals.) However, we will imagine that the Inn's full dataset does not fit into main memory. Furthermore, we will assume that there are many other users reading and writing the same data concurrently.

Wherever possible, calculations, filtering, summarization and other logic should be expressed in SQL for evaluation by the database.

General Requirements

Filenames. Use the Java programming language (JDK 8 or below). Name your main class `InnReservations` (source file `InnReservations.java`) The structure of your application and inclusion of any other files is left up to you. You are required to submit a `README` file which explains how to compile and run your program from command line (note: even if you are developing using

an IDE, please make sure you know how your code runs from command line. You may include a `make` file).

Dataset. Your system should utilize a standardized `INN` dataset, based on the DDL/DML statements below (*do not use your own table structure from Lab 2!*) Choose one team member's course database, create the tables there, then use the SQL `GRANT` command to allow access by other team members.

```
CREATE TABLE IF NOT EXISTS lab7_rooms (
 RoomCode char(5) PRIMARY KEY,
 RoomName varchar(30),
 Beds int(11),
 bedType varchar(8),
 maxOcc int(11),
 basePrice DECIMAL(6,2),
 decor varchar(20),
 UNIQUE (RoomName)
);

CREATE TABLE IF NOT EXISTS lab7_reservations (
 CODE int(11) PRIMARY KEY,
 Room char(5) NOT NULL,
 CheckIn date NOT NULL,
 Checkout date NOT NULL,
 Rate DECIMAL(6,2) NOT NULL,
 LastName varchar(15) NOT NULL,
 FirstName varchar(15) NOT NULL,
 Adults int(11) NOT NULL,
 Kids int(11) NOT NULL,
 UNIQUE (Room, CheckIn),
 FOREIGN KEY (Room) REFERENCES lab7_rooms (RoomCode)
);

INSERT INTO lab7_rooms SELECT * FROM INN.rooms;

INSERT INTO lab7_reservations SELECT CODE, Room,
  DATE_ADD(CheckIn, INTERVAL 7 YEAR),
  DATE_ADD(Checkout, INTERVAL 7 YEAR),
  Rate, LastName, FirstName, Adults, Kids FROM INN.reservations;
```

DBMS Access. Your system shall access the database server using JDBC. The URL, username and password must be passed to your application using shell environment variables. The environment variable names you use must be clearly documented in your README file.

UI. Your system may use either a command line or graphical user interface.

Malicious Input. Your application should assume that the end user is well versed in SQL injection attacks. Construct and execute your SQL statements accordingly.

Functional Requirements

Your application must support the following features, accessed via a main menu. Again, you may choose to implement using either a GUI or command line interface. If you choose the latter, display of lower case ASCII characters is (reluctantly) permitted.

Rooms and Rates. When this option is selected, the system shall output a list of rooms to the user sorted by popularity (highest to lowest, see below for definition of "popularity") Include all information from the `rooms` table, as well as the following:

- Room popularity score (number of days the room has been occupied during the previous 180 days, divided by 180)
- Next available check-in date.
- Length of the most recent stay in the room and most recent check out date.

Reservations. When this option is selected, your system shall accept from the user the following information:

- First name
- Last name
- A room code to indicate the specific room desired (or "Any" to indicate no preference)
- A desired bed type (or "Any" to indicate no preference)
- Desired begin and end dates of stay
- Number of children
- Number of adults

With this information, the system shall produce a numbered list of matching rooms available for booking, along with a prompt to allow booking by option number. If no exact matches are found, the system must suggest 5 possibilities for different rooms or dates. These 5 possibilities should be chosen intelligently based on similarity to the desired reservation ("similarity" defined as nearby dates or rooms with similar features or decor) For every option presented, maximum room occupancy must be considered and the dates must not overlap with another existing reservation.

At the prompt, the user may decide to cancel the current request, which will return the user to the main menu. If the user chooses to book one of the room options presented, they will enter the option number at the prompt. After a choice is made, provide the user with a confirmation screen that displays the following:

- First name, last name
- Room code, room name, bed type
- Begin and end date of stay
- Number of adults
- Number of children
- Total cost of stay, based on a sum of the following:
  - Number of weekdays * room base rate
  - Number of weekend days * (110% of the room base rate)
  - An 18% tourism tax applied to the total of the above two lines

Allow the user to either cancel, returning to the main menu, or confirm, which will finalize their reservation and create an entry in the `lab7_reservations` table.

Detailed Reservation Information. Present the user with a search prompt or form that allows them to enter any combination of the fields listed below (a blank entry should indicate "Any"). For all fields except dates, permit partial values using SQL LIKE wildcards (for example: GL% should be allowed as a last name search value)

- First name
- Last name
- A range of dates
- Room code
- Reservation code

Using the search information provided, display a list of all matching reservations found in the database.

The list shall show the contents of every attribute from the lab7_reservations table (as well as the full name of the room, and any extra information about the room you wish to add).

Revenue. When this option is selected, your system shall provide a month-by-month overview of revenue for an entire year. For the purpose of this assignment, all revenue from the reservation is assigned to the month

and year when the reservation **ended**. For example a seven-day hotel stay that started on October 30 will be treated as November revenue.

Your system shall display a list of rooms, and, for each room, 13 columns: 12 columns showing dollar revenue for each month and a 13th column to display total year revenue for the room. There shall also be a "totals" row in the table, which contains column totals. All amounts should be rounded to the nearest whole dollar.

Submission

Submit one set of deliverables per team:

- The entire code base for your application *(please do not include the MySQL JDBC driver JAR file)*
- The name of the database in which you created the lab7-specific INN tables.
- A `README` file which includes (a) the list of all members of the team, (b) any compilation/running instructions, including the names of environment variables used to pass JDBC URL, username and password, and (c) information about any known bugs and/or deficiencies.

Submit the lab via PolyLearn as a single archive (either `lab9.zip` or `lab9.tar.gz`).

Good luck!