Lab 7: Database Connectivity with JDBC

Due Date:

Friday, December 13th, 5:00pm

Assignment Preparation

This is a team lab. Team size is three-person minimum, five-person maximum. Your team may be composed of students from sections 01, 03, or 05 of CSC 365.

Lab Assignment

Each team will build a Java JDBC application that implements an Inn reservation system. Implement your application with the following broad 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. The volume of data in our sample dataset would allow an application 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 simultaneously reading and writing the same data.

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

Non-Functional Requirements

NFR1: Filenames. Use the Java programming language (JDK 8 or above). Name your main class `InnReservations`. 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 the command line.*

NFR2: Dataset. Your system must use 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) NOT NULL,
 Beds int(11) NOT NULL,
 bedType varchar(8) NOT NULL,
 maxOcc int(11) NOT NULL,
 basePrice DECIMAL(6,2) NOT NULL,
 decor varchar(20) NOT NULL,
 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),
 UNIQUE (Room, Checkout),
 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 9 YEAR),
  DATE_ADD(Checkout, INTERVAL 9 YEAR),
  Rate, LastName, FirstName, Adults, Kids FROM INN.reservations;
```

NFR3: 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.

NFR4: User Interface. Your system may use either a command line or graphical user interface.

NFR5: Protection against malicious input. Your application should assume that the end user is well-versed in SQL injection attacks. Construct and execute your SQL statements accordingly.

NFR6: Transaction Control. Your application must apply appropriate database transaction control. This involves the use of well-defined transaction boundaries and proper error handling.

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.

FR1: 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 (round to two decimal places)

- Next available check-in date.
- Length in days and check out date of the most recent (completed) stay in the room.

FR2: 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)
- Begin date of stay
- End date of stay
- Number of children
- Number of adults

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

If the requested person count (children plus adults) exceeds the maximum capacity of any one room at the Inn, print a message indicating that no suitable rooms are available. To reserve a block of rooms, it would be up to the user to submit multiple reservation requests.

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 multipled by room base rate
  - Number of weekend days multiplied by 110% of the room base rate

  - An 18% tourism tax applied to the total of the above two calculations

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

FR3: Reservation Change. Allow the user to make changes to an existing reservation. Accept from the user a reservation code and new values for any of the following:

- First name
- Last name
- Begin date
- End date
- Number of children
- Number of adults

Allow the user to provide a new value or to indicate "no change" for a given field. Update the reservation based on any new information provided. If the user requests different begin and/or end dates, *make sure to check whether the new begin and end dates conflict with another reservation in the system.*

FR4: Reservation Cancellation. Allow the user to cancel an existing reservation. Accept from the user a reservation code, confirm the cancellation, then remove the reservation record from the database.

FR5: Detailed Reservation Information. Present a search prompt or form that allows the user 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).

FR6: Revenue. When this option is selected, your system shall provide a month-by-month overview of revenue for an entire year. For reservations that span multiple months, revenue must be computed based on individual days. For example a reservations that begins on October

30th and ends on November 2nd represents 2 nights of October revenue (Oct 30 and 31st) and 1 night of November revenue (Nov 1st).

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 yearly 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 for your application *(please do not include the MySQL JDBC driver JAR file)*
- The name of the database (username) in which you created your lab7-specific INN tables.
- A `README` file which includes (a) the list of all members of the team, (b) any compilation/runtime 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.
- Full output from `git log --stat` or equivalent manually-compiled data that provides detail about every member's contribution to the project (eg. work log, division of responsibilities)

Submit the lab via PolyLearn as a single archive (either `lab7.zip` or `lab7.tar.gz`).