CSC 365: Database Systems
Fall 2018

Project A: Database Connectivity with JDBC
Emilia E. Buckalew

Due date:
Saturday, December 8, 2018 @ 11:59pm

Assignment Preparation

This is a pair project. There is too much work here for one person; you *must* not work alone. Please alert me ASAP if you lack a partner and I will do my best to assist you.

Lab Assignment

In a Nutshell

Each pair is to build a Java JDBC application that implements a simple Bed & Breakfast Inn reservation system. You will be using the data from the INN dataset for this lab.

The Task: Overview

The INN dataset is designed to provide support for a simple hotel reservation system. For this lab, you will build a Java application that implements three components of such a system: the ADMIN subsystem: a number of convenience functions to make it easier to set up, reset and manage the system; the OWNER subsystem: the browsing and analytical functionality for the hotel owner; and the GUEST subsystem: the room reservation functionality for the prospective hotel guests.

Detailed requirements are outlined below.

General Requirements

GR-0. Filenames. Your system will be implemented in Java. Name your main class `InnReservations` and your main program `InnReservations.java` respectively. 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 the command line.) See the `Submission` section for more details about the `README` file. You may include a `makefile`, if you wish.

GR-1. Dataset. Your system will utilize the `INN` dataset. It will store all data on the `mySQL` server, and will use `JDBC` functionality to connect to the database and update/query it. You will use

your own database for your INN relations. You may use the class INN database to create your own INN tables using the statements:

```
CREATE TABLE <your table name> AS
   SELECT *
   FROM   INN.rooms;

CREATE TABLE <your table name> AS
   SELECT *
   FROM   INN.reservations;
```

GR-2. DBMS Access. Your system will access the mySQL server using JDBC via the following process. The information about the server URL, the login Id of the user, and the password will NOT be hard-coded into your program. Instead, your program, upon startup, will read the contents of a file named ServerSettings.txt located in the directory from which your program is being run. ServerSettings.txt is organized as follows. It consists of three lines of text. The first line is the URL of the mySQL server to which the program will connect. The second line is the login Id (username) of the user whose account the system will use to access the data. The third line is the password for that account.

If you were connecting to the mySQL server as user ''scott'' whose password is ''tiger'', then the contents of the ServerSettings.txt file would be

```
jdbc:mysql://ambari-head.csc.calpoly.edu/scott
scott
tiger
```

Note that in the sample above, scott is specified as both the database to be used and the user to be logged in.

GR-3. UI. Your system will have a user interface capable of supporting user interactions with the software. To simplify your task, a UI will be supplied to you. Intuitively, there are three main components to the system, which assume three different categories of users for them. For simplicity, all three subsystems will be unified under the same application (to facilitate grading), but within the application, they will be independent.

GR-4. Startup. Upon startup, your system will perform the following activities (not necessarily in the order listed below):

- Read the contents of the ServerSettings.txt file.
- Establish a connection with the mySQL server using the credentials provided in the file.
- Check if the tables from the INN dataset already exist on the account. If they do not exist, create (but do NOT populate) them. You may use the following statements to do this:

```
CREATE TABLE IF NOT EXISTS <your table name>
LIKE INN.rooms;

CREATE TABLE IF NOT EXISTS <your table name>
LIKE INN.reservations;
```

- Display the main UI of your program. The main UI must allow access to all three subsystems as well as provide a graceful exit option.
- Perform a check of whether the `INN` database is full or empty. A full database is a database that contains the tuples from the INN dataset and possibly some more tuples. An empty database is a database that has both `Rooms` and `Reservations` tables, but the tables contain no records[1].
- Wait for user activity.

ADMIN subsystem

AR-0. Overview. The ADMIN subsystem is designed to make it easy to deploy the application and to reset the database state. The following functionality will be provided for the ADMIN subsystem:

- Current status display.
- Viewing facility for the contents of each of the `INN` dataset tables.
- Ability to clear the database of all records.
- Ability to load and/or reload the contents of the database.
- Ability to destroy the database (i.e., drop all the tables).
- Return to `MAIN`[2]

Each function is described below.

AR-1. Current Status Display. The main UI for the `ADMIN` subsystem will be the current status screen, displaying the general information available about the dataset at present moment. All other functionality of the `ADMIN` subsystem will be accessible from the current status display. The following information will be made available:

- *Database Status.* Either *full* or *empty* or *no database* (note, that the latter status will never happen upon startup, but can happen under other circumstances).
- *Reservations.* Total number of tuples in the `RESERVATIONS` table.
- *Rooms.* Total number of tuples in the `ROOMS` table.

[1] You do not need to look at the contents of every single record to determine if the DB is full. Checking the number of records in each table is sufficient.

[2] Yes, it's not secure, but it sure is convenient. And since this will not be deployed, why not?

AR-2. Table display. The status option will provide the facility to display the contents of the rooms and the reservations tables. For example, two choices, ‘‘`View Rooms table`’’ and ‘‘`View Reservations table`’’ can be used. When the user elects to view one of the database tables, a SQL query retrieving the full contents of the table is executed, and the results are displayed in a way designed and implemented by each team.

AR-3. Clear database. A ‘‘`Clear DB`’’ option will be provided on the current status display. When this option is chosen, SQL commands deleting the contents from both the rooms and the reservations tables will be executed. As the result of the ‘‘`Clear DB`’’ command, the database becomes empty and this information will be immediately propagated to the current status display.

AR-4. Load/Reload DB. A ‘‘`Load DB`’’ and/or ‘‘`ReLoad DB`’’ option or options (or other UI constructs facilitating the same functionality) will be provided on the current status display. When this functionality is engaged, the system will behave as follows:

- The system checks if the current database exists and is populated. If the database is populated, your system will return to the ‘‘`Current Status`’’ display. The `STATUS` display will contain a diagnostic message indicating that the table(s) already exist and are populated.
- If the database exists but is empty, the system will insert into the relations the full contents of the `INN` dataset.
- If the database does not exist, your system will execute SQL commands creating the database and then populating it with the records.

Suggestion: You may be able to scavenge code from early labs.

Upon completion, the system returns to the status display screen, which should reflect the current state of the database.

AR-5. Database Removal. When this functionality is engaged, your system will drop the `ROOMS` and `RESERVATIONS` tables and return to the status display screen, which should show whether the deletion was successful.

AR-6. Switch subsystem. The ADMIN subsystem UI will provide the ability to switch between the subsystems at any time. That is, a user of the system will be able to switch at any time from the ADMIN subsystem to either the GUEST or the OWNER subsystem.

OWNER subsystem.

OR-0. Overview. This subsystem is designed to allow the Bed & Breakfast management to monitor the overall state of the reservations. The `OWNER` subsystem UI will list the functionality available within the subsystem and allow the user to select desired functionality. The following functionality will be available to the user in the `OWNER` subsystem.

- Occupancy overview
- View Revenue
- Review Rooms
- Review Reservations
- Review Detailed reservation information
- Return to `MAIN`[3]
- Return to `MAIN`

OR-1. Occupancy overview. When selected, your system will do the following:

- Display a "fresh" UI screen providing a way by which the user can enter two dates.
- When a user enters a date (Month, Day - use 2010 year by default), your system will display a list of rooms. For each room, the system will specify whether it is occupied or empty on the given date. If a user selects an occupied room, the system will display full information about the reservation that covers the given day (see below).
- When a user enters a pair of dates (start date and end date), your system will display a list of rooms. For each room the system will specify whether the room is fully occupied during the time period (i.e., someone is staying in the room every night of the time period), partially occupied (someone is staying in the room part of the time, but not every night of the time period) or fully vacant (the room is not reserved for any night of the time period).

  If a fully occupied or partially occupied room is selected by the user and `''Enter''` is pressed, the system will display a brief list of reservations that cover the time interval. Selecting a specific reservation from the list will yield the display of full information about the reservation.

Note: try getting the information for each of the two cases described above using a single SQL query per task.

OR-2. Revenue. When this option is selected, your system will provide a month-by-month overview of the reservations volume and the cash flow. For the purpose of this assignment, all revenue from the reservation is assigned to the month when the reservation **ended**. For example, a seven-day hotel stay that started on October 30 will be treated as November revenue. Note also that this means that some reservations in the system do not contribute to the year 2010 revenue. You can ignore those reservations in your summaries (the owner can just think of it as "one of those things").

Your system will display a list of rooms, and, for each room, 13 columns of numbers (12 months plus a total). The numbers displayed in the columns will be one of the following:

1. reservation counts, or

[3] Yes, it's still not secure, but it is still convenient. And since this will not be deployed, why not?

2. total number of days occupied, or

3. the dollar revenue for each month.

The final column will be a total of the numbers for each room. There will also be a `TOTALS` row in the table, which adds up the numbers for each month. The UI will provide the option of selecting which set of numbers—reservations, days of occupancy, or revenue—is to be displayed, and will switch the numbers when a different selection is made.

OR-3. Reservations. When this option is selected, the user will be able to browse the list of reservations. The system will provide the means for the user to enter the start day and the end day for the search, and/or select a specific room, and will report all reservations that commence within the specified time period and are for a selected room. When the user selects a reservation, detailed information about it will be displayed.

OR-4. Rooms. When this option is selected, the system will display the list of rooms. Two options will be made available: viewing full information about a room and viewing the list of reservations for a given room.

When the user chooses to view a specific room, a screen containing information about the room will be displayed. In addition to the information about the room from the `ROOMS` table, the following information will be displayed:

- total number of nights of occupancy for the room in 2010
- percent of time the room is occupied
- total revenue the room has generated in 2010
- percent of the overall 2010 revenue generated by the room

Note that the information above must only appear in the OWNER subsystem. Room information screen(s) in other subsystems may only contain data from the `ROOMS` table.

When the user elects to view the reservations for a room, a screen displaying a list of reservations for the selected room sorted in chronological order will be displayed. When the user selects a reservation, the detailed reservation screen will be displayed.

OR-5. Detailed reservation information. When a reservation is selected from a list of reservations (see OR-1, OR-3, OR-4), a display will appear that shows the full details of the reservation. The display will show the contents of every attribute from the reservations table (as well as the correct name of the room and any extra information about the room you want to add).

GUEST subsystem

R-0. The GUEST subsystem will provide the functionality for the prospective hotel guest to reserve a room. Following the traditions of hotel web sites, the following options will be offered to the guest:

- Rooms and Rates
- Reservations (`R-2`—`R-5`)
- Return to `MAIN`[4]

R-1. Rooms and Rates. When this option is selected, the system will display the list of rooms to the user. When a room is selected from the list, the system will show a detailed information screen, similar to the screen shown to the owner (see **OR-4**), except for the room occupancy and revenue information. The information screen will contain specific details about the room and a ‘‘`Check Availability`’’ option.

R-2. Checking Room Availability. When the user chooses the ‘‘`Check Availability`’’ option, the system will offer the following dialog to the user. The system will provide the means for the user to enter a pair of dates (check-in and check-out). Once the dates are entered, the system will show information about whether or not the room is available on each of the nights of the proposed stay[5]. If the room is available for a given night, the system will display the room rate (see **R-3** for room rate computation). If the room is unavailable, the system will display ‘‘`Occupied`’’ for that night. If the room is available for the duration of the selected stay (i.e., available on each night), a ‘‘`Place a Reservation`’’ option will be provided for the user to complete the reservation.

R-3. Pricing. The price of one night of stay is determined as follows. On a weeknight, the price of the stay is the base rate of the room. On a weekend, the price of the stay is 110% of the base for the room. For any reservation or prospective stay that covers the following nights: January 1, July 4, September 6, October 30, the nightly rate for each night will be 125% of the base for the room.

All nights in a stay will be priced the same way, so your system must determine the highest applicable rate and apply it to all nights of the stay. E.g., if a room reservation goes from October 26 to November 2, then the 125% markup will be applied to the nightly rate for all nights.

A discount adjustment (10% `AAA` or a 15% `AARP`) may be applied during the reservation time.

R-4. Reservations. When the user chooses the ‘‘`Reservations`’’ option from the main GUEST subsystem menu, the system will request the user to enter the prospective dates of the stay. When the dates are entered, the system will display the list of rooms that have availability **for each night of the stay**. For each such room, the room name, and the applicable nightly rate (see **R-3**) will be displayed.

When the user chooses a room, a screen containing detailed information about the room (similar to that in **R-2**) will be displayed, with the ‘‘`Make a Reservation`’’ option available.

R-5. Completing a reservation. When the user chooses to complete a reservation, either through the ‘‘`Rooms and Rates`’’ or ‘‘`Reservations`’’ subtasks, the system will collect the information for reservation completion. The system will ask the user to specify the following details

[4]Yes, it's still not secure, but it is still convenient for navigation. And since this will not be deployed, why not?

[5]Note: you may have to execute a separate query for each night of the stay to get this information.

(we will not worry about the payment options in this system, so only the details for the Reservations table in the `INN` database will be collected):

- Name (last, first) of the hotel guest.
- Number of adults staying in the room.
- Number of children staying in the room.
- Applicable rate discounts.

The total number of guests in the room will not exceed the room capacity. If it does your system will provide appropriate feedback and wait for the guest to change the numbers.

There are two applicable rate discounts that can be applied to the nightly rate computed as per **R-3**: `AAA` gives a 10% discount; `AARP` gives a 15% discount. The user can select one of the two (but not both) discounts, or no discount.

The system will provide a `''Place reservation''` option, which will become available once the user enters all necessary and correct information.

R-6. Updating the database. Once the user completes the `''Place reservation''` operation, the system will do the following:

- Generate a six-digit reservation code. The code must be unique - i.e., not found in the current reservations table.
- Insert a record about the new reservation into the reservations table.
- Display a `''Your reservation is complete.''` message which includes the details of the reservation and the reservation code.

The system then will provide the facility for the user to return to the main screen of the GUEST subsystem.

Submission

Submit one set of deliverables per team. The deliverables include:

- the entire code base for your application;
- a `README` file which includes (a) the list of all members of the team, (b) any compilation/running instructions, and (c) information about any known bugs and/or deficiencies.

Submit the lab either as a `projectA.zip` or `projectA.tar.gz` archive using the appropriate `handin` command:

```
$ handin ebuckale projectA.01 <archive>
$ handin ebuckale projectA.03 <archive>
```

Good luck!