CSCC43/343 Introduction to Databases
Winter 2008 - Assignment 3 - JDBC

Instructions:

For this assignment, you will create the class Assignment3.java which will allow you to process queries using JDBC. We will again use the Airline database tables which we used (and are described) in the last assignment (Assignment3.ddl). The tables have been slightly modified (e.g., employee last and first names have been added.)

Rules:

1. Standard input and output must not be used. This will halt the "automarker" program and you will probably end up with a zero.
2. The database, username, and password must be passed as parameters, never "hard-coded".
3. If you feel you need an intermediate view to execute a query in a method, you must create it in that method. You must also drop it before exiting that method.
4. Be sure to close all unused statements and result sets.
5. All return values will be String, boolean or int values.
6. All dates will be in the format YYYYMMDD where YYYY is the four-digit year; MM, the two-digit month; and DD the two-digit day. (For more information on manipulating date and time values look at http://www.postgresql.org/docs/8.0/interactive/functions-datetime.html)
7. Time will be in the twenty-four hour clock – you can use the format HH:MM where HH is the hours and MM represents the minutes.
8. You will lose marks for not following the signature of each method as it is defined below.
9. A successful action (Update, Delete) is when:
   a. It doesn't throw an SQL exception, and
   b. The number of rows to be updated or deleted is correct.
10. When rows of data are returned as a String, they must be in the following contiguous format:
    - Columns are separated with a colon ":". There is no colon after the last column of a row.
    - Rows are separated with a pound-sign "#". There is no pound-sign after the last row of the result.
    - Leading and trailing spaces are eliminated
    - e.g., a result set that includes multiple rows of the columns firstname, lastname might look like "Renee:Miller#Faye:Baron#Dimitris:Tsirogiannis"

Class and method descriptions:

Class name	**Description**
Assignment3.java	Allows several interactions with a postgreSQL database. The third assignment for CSC343/C43 Winter 2008.

Instance Variables (you may want to add more)

Type and name	**Description**
`Connection a1Con`	The database connection for this session. (**Note**: make this `public` – so that if your method fails, the tester can still create a connection and use it for other methods.)

Methods (you may want to add helper methods for redundant code)

Constructor	**Description**
`Assignment2()`	Identifies the postgreSQL driver using Class.forName method.

Method	**Description**
`boolean connectDB(String URL, String username, String password)`	Using the String input parameters which are the URL, username, and password respectively , establish the `Connection` to be used for this session. Returns **`true`** if the connection was successful.
`boolean disconnectDB()`	Closes the connection. Returns **`true`** if the closure was successful.
`boolean insertImplment (int flightNo, String date, int pilot, int plane)`	A flight is being added for a specific date. Insert a flight implementation record using the information supplied and return `true` if your insertion is successful.. You must use the company ID of the pilot as the two-character company ID. As well, you must verify that the pilot is qualified to fly the plane in question. If the pilot is not qualified for the plane; if there is already a pilot assigned for the same company, flightNo and date; if the pilot, flight number or plane are invalid; or if the flight date is not in the future – then you cannot insert the flight implementation and must return `false`. You must also catch exceptions and return `false` if your insertion fails for any reason.
`String getLanguageCount()`	The Air Transport Safety Association wants to know what languages the flight attendants for each company are fluent in. For each company, for each language that is the primary language of the flight attendants for that company, return the count of airline flight attendants. The output must be sorted in ascending alphabetical order by airline and language within airline. For example, if 5 flight attendants for AC have French as their primary language, 7 have English, and 4 have Chinese, and if all 24 AA employees have English as their primary language, then the String returned would be: `AA:English:24#AC:Chinese:4#AC:English:7#AC:French:5`

Method	Description
`String getCityCrew (String date, String company, String AirportID)`	The ground accommodation planners need to know who will be flying into their city, and when they will be arriving so that they can make ground arrangements (hotel and limo) for them. Return a list of all crew who are flying into the city identified by the three-character apid (`AirportID`) on the `date` specified for the `company` specified. The crew must include the main pilot as well as all other crew. Include the following information for each row: `<flight number>:<arrival time>:<last name>:<first name>` Remember to separate rows with the # symbol. Order your list by first name within last name within arrival time – in ascending order.
`int changePilot(String company, int flight, int oldPilot, int newPilot)`	The pilot for a particular flight is being replaced. From this day forward, the new pilot will handle the flight. Change the old pilot (`oldPilot`) who is currently listed as the main pilot for the flight and company specified to the newPilot which is passed. You must first verify that the new pilot is eligible to fly the plane specified for the flight. Only change the pilot for flights occurring from today onwards (since it makes no sense to switch the pilot on flights that have already flown.) Return the number of records that were changed as a result of your execution.
`int deleteFlight(String company, int flight, String date)`	An airline is cancelling one of their flights as of a particular date. Delete all flight implementations for the flight and company specified effective the date specified and later. Return the number of records that were deleted as a result of your execution.
`int delayFlights( String company, String cityCode, int hours, int mins)`	There is a major schedule change. Flights from a specific city are being delayed by a fixed period of time. All flights departing from the city (`cityCode`) for the specified `company` are delayed by the `hours` and minutes (`mins`) specified. Make the appropriate update and return the number of flights affected. **Note**: that flights may not depart after midnight, so any changes, which would result in next-morning travel, must depart no later than 11:59 p.m. (24:00). You will probably have to research time functions in the Postgres documentation to effect this update.