Alexandria University
Faculty of Engineering
Computer and Systems Engineering Dept.
Second Year
Fall 2016

CS221: Object Oriented Programming
Programming Assignment 4
Due Date: Saturday, December 17th, 11:59 PM

JDBC API

Objectives

Upon completion of this assignment, you will be able to:

- Implements JDBC standard interface.
- Integrate JDBC interface with your earlier DBMS implementation.
- Apply different design patterns to your model.
- Extend your previously written code and augment it with new features.

Description:

Java Database Connectivity (JDBC) provides Java developers with a standard API that is used to access databases, regardless of the driver and database product.

JDBC presents a uniform interface to databases - change vendors and your applications only need to change their driver.

A typical program that uses JDBC driver looks like the following:

```
import java.sql.Statement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class JDBCExample {
	public static void main(String args[]) {
		try {
			Class.forName("myDriver.ClassName");
		} catch (java.lang.ClassNotFoundException e) {
			System.err.print("ClassNotFoundException: ");
			System.err.println(e.getMessage());
			System.exit(1);
		}
		// Contains Database name.
		String url = "jdbc:mySubprotocol:myDataSource";
		Connection con;
		Statement stmt;
		String createString = "create table COFFEES "
				+ "(COF_NAME VARCHAR(32), " + "SUP_ID INTEGER, "
				+ "PRICE FLOAT, " + "SALES INTEGER, " + "TOTAL INTEGER)";
		try {
			con = DriverManager.getConnection(url, "myLogin", "myPassword");
			stmt = con.createStatement();
			stmt.executeUpdate(createString); // Execute SQL queries.
			stmt.close();
			con.close();
		} catch (SQLException ex) {
			System.err.println("SQLException: " + ex.getMessage());
		}
	}
}
```

Tasks:

- Extend your previous code for DBMS by:

  1- Adding support for these new data types for your XML DBMS (Date, Float). For "Date", you can use the "java.sql.Date" class as a representation, and for "Float", you can use "java.lang.Float".

  2- Adding support for two statements/operators other than those you implemented on the previous phase. You should choose from: "ALTER TABLE" (Add/delete columns), "SELECT DISTINCT", "SQL UNION", and "SQL ORDER BY". If you implemented any of these as a bonus on the last phase, you are now required to implement different statements/operators. In other words, you cannot deliver something from the previous phase for this requirement.

  3- Add another backend writer for your DBMS, for example, the backend writer for the previous assignment was XML. This time, You need to support another format. You can choose between "JSON" and "Protocol Buffers" ("Protocol Buffers" are highly recommended). Then, you should have a parameter in your DBMS/Database configuration to choose which backend to be used in this session/Database. No schema validation is required for this part.

- These requirement changes are meant to test the goodness of your design. If you had to change many of your previous assumptions and components, then your code was badly designed for extensibility.

- Implement the JDBC main interfaces to access the tables' data.

- The following interfaces/methods should be implemented:

  - **java.sql.Driver**
    - acceptsURL(String url)
    - connect(String url, Properties info)
    - getPropertyInfo(String url, Properties info)

  - **java.sql.Connection**
    - close()
    - createStatement()

  - **java.sql.Statement**
    - addBatch(String sql)
    - clearBatch()
    - close()
    - execute(String sql)
    - executeBatch()
    - executeQuery(String sql)
    - executeUpdate(String sql)
    - getConnection()
    - getQueryTimeout()
    - setQueryTimeout(int seconds)
    - getResultSet()
    - getUpdateCount()

  - **java.sql.Resultset**
    - absolute(int row)
    - afterLast()
    - beforeFirst()
    - close()
    - findColumn(String columnLabel)
    - first()
    - getInt(int columnIndex)
    - getInt(String columnLabel)
    - getDate(int columnIndex)
    - getDate(String columnLabel)
    - getString(int columnIndex)
    - getString(String columnLabel)
    - getFloat(int columnIndex)
    - getFloat(String columnLabel)
    - getMetaData()
    - getObject(int columnIndex)
    - getStatement()
    - isAfterLast()
    - isBeforeFirst()

        - isClosed()
        - isFirst()
        - isLast()
        - last()
        - next()
        - previous()

    - **java.sql.ResultSetMetaData**
        - getColumnCount()
        - getColumnLabel(int column)
        - getColumnName(int column)
        - getColumnType(int column)
        - getTableName(int column)

    Any other method in the interfaces not mentioned above will have empty implementation that throws *java.lang.UnsupportedOperationException*.

    For this requirement, you should read the JDBC docs very carefully. Failing to commit to the given assumptions and definitions will result in severe loss of grades. Your code will be tested for correctness with the OnlineTester, which depends on these assumptions.

- Complete log of the operations done on the database through the DBMS should be shown. Operations include initiating DB connections, query execution, errors and warnings, connections closing. Operations timestamp should be indicated. You should use "log4J" open source package for this purpose.

- Any configurations (including the logging configuration, the format of backend, etc) should be done externally using configuration files, without modifying the source code.

- Build on your implemented DBMS in the previous lab. You should correct any mistakes that was noted to you on the previous discussion.

- Use JUnit to write test scenarios that cover all the functionalities, errors and possible use cases for a developer that uses the JDBC driver. You should make a separate test suit for each module. Your test suits were very bad in the previous assignment, so make sure to learn from your mistakes.

- You should pack your driver in a JAR file and implement a command line interface (CLI) that processes SQL queries as you did in the previous assignment. This time you must use only the driver interface to access the DBMS. You should not use

any of your internal DBMS classes, including the DBMS interface. Your CLI should give meaningful feedbacks to the user in the cases of success or failure.

Resources

- JDBC Tutorial: http://docs.oracle.com/javase/tutorial/jdbc/
- DBMS: http://en.wikipedia.org/wiki/Database_management_system

Deliverables

- You should work in groups of four (Groups of five for cases where groups of four were not found), and your report should also contain the detailed division of labor among group members. You may assign one of the team as a Quality Assurance. This person should perform intensive tests to make sure everything is working fine for a regular user. You may also assign one of the team to follow the Piazza discussions. It is your responsibility to follow any specific illustrations posted there.

- Note that during the delivery of the previous labs we were searching for working scenarios to give you the grade. In this lab, we will be searching for non-working scenarios to decrease the grade. This is mainly to encourage you to **polish** your code and make sure everything is working correctly.

- You should deliver your source code using your git repository. You should also bring a proof of your delivery before the noted deadline. You should also know that your code will be tested on the OnlineTester, so you should follow its proper format of delivery.

- You should deliver a clear, good-looking report that contains the required UML diagram, describes your design thoroughly. Any design decisions that you have made should be listed clearly.

- This time, we will deduct a lot of marks if your design is still bad. You have more than enough time to redesign some parts of your DBMS. So make sure that the design conforms with the object oriented principles you studied.

- Delivering a copy will be severely penalized for both parties, so delivering nothing is much better than delivering a copy.