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.

Tasks

- 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)
  - **java.sql.Resultset**
    - absolute(int row)
    - afterLast()
    - beforeFirst()
    - close()
    - findColumn(String columnLabel)
    - first()
    - getInt(int columnIndex)
    - getInt(String columnLabel)
    - getMetaData()
    - getObject(int columnIndex)
    - getStatement()
    - getString(int columnIndex)
    - getString(String columnLabel)
    - 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.
- 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 can use JDK logger, log4J, ...etc. packages for this purpose.

- Any configurations (including the logging configuration) 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 in it.
- 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.
- You are required to create UML diagrams
  - Use Case
  - State Diagram for 3 scenarios
  - Detailed class diagram
  - Sequence Diagram

Testing

You can call it this way:

```
Driver driver = new MyDriver();
Properties info = new Properties();
File dbDir = new File(/*your database folder location*/);
info.put("path", dbDir.getAbsoluteFile());
Connection connection = driver.connect("jdbc:xmldb://localhost", info);
```

And this is the impl of the connect function

```
public Connection connect(String url, Properties info) throws SQLException {
File dir = (File) info.get("path");
String path = dir.getAbsolutePath();
return connectionManager.getConnection(path); // pool
}
```

Resources

- **JDBC Tutorial**.
- **DBMS** .

Project setup

- Download the project from this link.
- Add test_supporting_files.jar to your external jars.
- Organize your code under package with name eg.edu.alexu.csd.oop.jdbc.cs <your-two-digit-class-number>.
- Just write class number of one member of the team.

Deliverables

- You should work in groups of four,
- You should deliver a report that contains the required UML diagrams, describes your design thoroughly, and contains snapshots of your UI and a user guide that explains how to use your application. Any design decisions that you have made should be listed clearly.
- Develop this assignment in Java.
- A jar file that should be test "as a library you include in a new project" in a new project as the data base connector to your DBMS you implemented in the previous assignment.
- A self-executable Jar: The program should be executable by simply double clicking the icon, given that you have a running JRE.
- Upload your report, jar files, and source code zipped to Git.
- Delivering a copy will be severely penalized for both parties, so delivering nothing is so much better than delivering a copy.

Good Luck