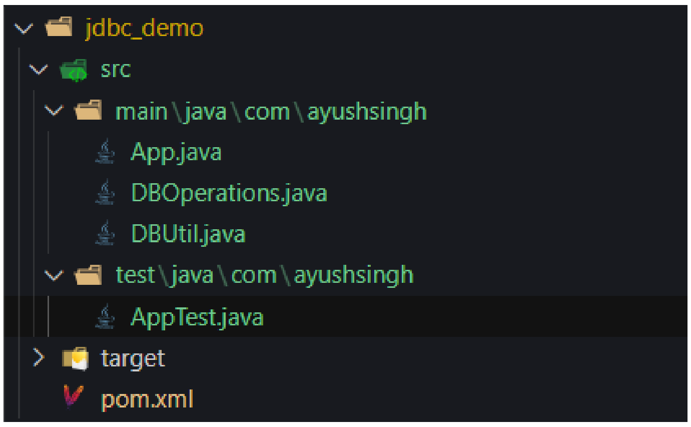
Wk08 Assignment: JDBC

Problem Statement

1. Create simple JAVA application to connect to MYSQL Database
2. Perform CRUD Operation. Display the results in Console.
3. Prepare the steps performed along with screenshot of the result.

Project Description

Project Structure



```
jdbc_demo
  src
    main\java\com\ayushsingh
      App.java
      DBOperations.java
      DBUtil.java
    test\java\com\ayushsingh
      AppTest.java
  target
  pom.xml
```

Steps

1. Create a **Maven** project with the **maven-archetype-quickstart.**
2. Add the following dependency for **MySQL-**

```xml
 <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.33</version>
 </dependency>
```

3. Create a class **DBUtil.java-**

```java
package com.ayushsingh;


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

/**
 * This class provides utility method for database connectivity.
 * It contains a static method {@link #getConnection()} to establish a connection to the MySQL
database.
 *
 * @author Ayush Singh
 * @version 1.0
 * @since 12-05-2024
 */
public class DBUtil {

    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/ecom";
    private static final String USERNAME = "hbstudent";
    private static final String PASSWORD = "hbstudent";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(JDBC_URL, USERNAME, PASSWORD);
    }
}
```

In this class, we are obtaining the connection to the mysql database using the connection string, username and password.

4. Create a class **DBOperations.java-**

Add the imports-

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
/**
 * This class represents the operations to interact with the database regarding product 
management.
 * It provides methods to create a table, insert, read, update, and delete products from the 
database.
 * The methods are used in the {@link App} class.
 *
 * @author Ayush Singh
 * @version 1.0
 * @since 12-05-2024
 */
public class DBOperations {


}
```

This class will have all the CRUD methods-

a. Create table-

```
    /**
     * Creates the 'products' table in the database
     * The table contains columns for product ID, name, price, and quantity.
     *
     * @throws SQLException If a SQL exception occurs while executing the create table query.
     */
    public static void createTable() throws SQLException {
        try (Connection conn = DBUtil.getConnection(); Statement statement = conn.createStatement()) {
            String createTableQuery = "CREATE TABLE IF NOT EXISTS products (" + "product_id INT AUTO_INCREMENT 
PRIMARY KEY, " + "product_name VARCHAR(100), " + "product_price DECIMAL(10, 2), " + "product_quantity INT)";
            System.out.println("Creating table... " + createTableQuery);
            statement.executeUpdate(createTableQuery);
            System.out.println("Table created... ");

        }
    }
```

b. Insert product-

```
 /**
     * Inserts a new product into the 'products' table using a prepared statement.
     *
     * @param productName     The name of the product to be inserted.
     * @param productPrice    The price of the product to be inserted.
     * @param productQuantity The quantity of the product to be inserted.
     * @throws SQLException If a SQL exception occurs while executing the insert query.
     */
    public static void insertProduct(String productName, double productPrice, int productQuantity) throws
SQLException {
        String query = "INSERT INTO products (product_name, product_price, product_quantity) VALUES (?, ?,
?)";
        try (Connection conn = DBUtil.getConnection(); PreparedStatement preparedStatement =
conn.prepareStatement(query)) {
            preparedStatement.setString(1, productName);
            preparedStatement.setDouble(2, productPrice);
            preparedStatement.setInt(3, productQuantity);
            System.out.println("Inserting product... " + preparedStatement);
            int res = preparedStatement.executeUpdate();
            System.out.println("Rows inserted: " + res);
        }
    }
```

c. Read products-

```
/**
     * Reads all products from the 'products' table and returns the result set.
     *
     * @return ResultSet containing product data retrieved from the database.
     * @throws SQLException If a SQL exception occurs while executing the select query.
     */
    public static ResultSet readProducts() throws SQLException {
        String selectQuery = "SELECT * FROM products";
        Connection conn = DBUtil.getConnection();
        Statement statement = conn.createStatement();
        System.out.println("Reading products... " + statement.toString());
        return statement.executeQuery(selectQuery);
    }
```

d. Update price-

```
 /**
     * Updates the price of a product in the 'products' table based on the product ID.
     *
     * @param productId The ID of the product to update.
     * @param newPrice  The new price to set for the product.
     * @throws SQLException If a SQL exception occurs while executing the update query.
     */
    public static void updateProductPrice(int productId, double newPrice) throws SQLException {
        String updateQuery = "UPDATE products SET product_price = ? WHERE product_id = ?";
        try (Connection conn = DBUtil.getConnection(); PreparedStatement preparedStatement =
conn.prepareStatement(updateQuery)) {
            preparedStatement.setDouble(1, newPrice);
            preparedStatement.setInt(2, productId);
```

```java
            System.out.println("Updating product price... " + preparedStatement);
            int res = preparedStatement.executeUpdate();
            System.out.println("Rows updated: " + res);
        }
    }
```

e. Delete product-

```java
    /**
     * Deletes a product from the 'products' table based on the product ID.
     *
     * @param productId The ID of the product to delete.
     * @throws SQLException If a SQL exception occurs while executing the delete query.
     */
    public static void deleteProduct(int productId) throws SQLException {
        String deleteQuery = "DELETE FROM products WHERE product_id = ?";
        try (Connection conn = DBUtil.getConnection(); PreparedStatement preparedStatement =
conn.prepareStatement(deleteQuery)) {
            preparedStatement.setInt(1, productId);
            System.out.println("Deleting product... " + preparedStatement);
            int res = preparedStatement.executeUpdate();
            System.out.println("Rows deleted: " + res);
        }
    }
```

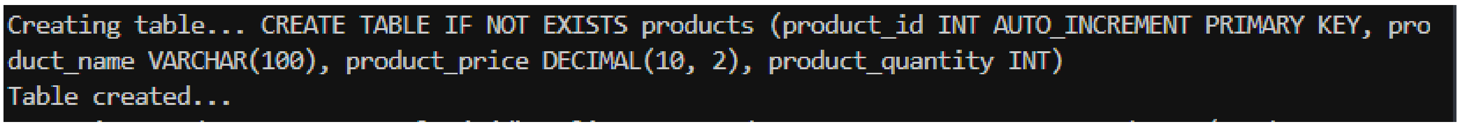
5. Use these methods in the **App.java class-**

```java
package com.ayushsingh;


import java.sql.ResultSet;
import java.sql.SQLException;

/**
 * This class is the Main starting class for the project.
 * In the {@link #main(String[])} method, the database operations are performed.
 * In the {@link #displayResultSet(ResultSet)} method, the result set is displayed.
 *
 * @author Ayush Singh
 * @version 1.0
 * @since 12-05-2024
 */
public class App {
    public static void main(String[] args) {
        try {
            DBOperations.createTable();

            // -Insert product
            DBOperations.insertProduct("Product A", 5674, 100);
            DBOperations.insertProduct("Product B", 1800, 200);
            DBOperations.insertProduct("Product C", 4556, 453);

            // - Read products
            ResultSet resultSet = DBOperations.readProducts();
            displayResultSet(resultSet);

            // - Update product price
            DBOperations.updateProductPrice(8, 25.75);
            System.out.println("Product price updated");
            // - Delete product
            DBOperations.deleteProduct(14);
            System.out.println("Product deleted");

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void displayResultSet(ResultSet resultSet) throws SQLException {
        System.out.println("Product Data:");
        while (resultSet.next()) {
            int productId = resultSet.getInt("product_id");
            String productName = resultSet.getString("product_name");
            double productPrice = resultSet.getDouble("product_price");
            int productQuantity = resultSet.getInt("product_quantity");
            System.out.println("ID: " + productId + ", Name: " + productName + ", Price: $" + productPrice +
", Quantity: " + productQuantity);
        }
        resultSet.close();
    }
}
```

Output

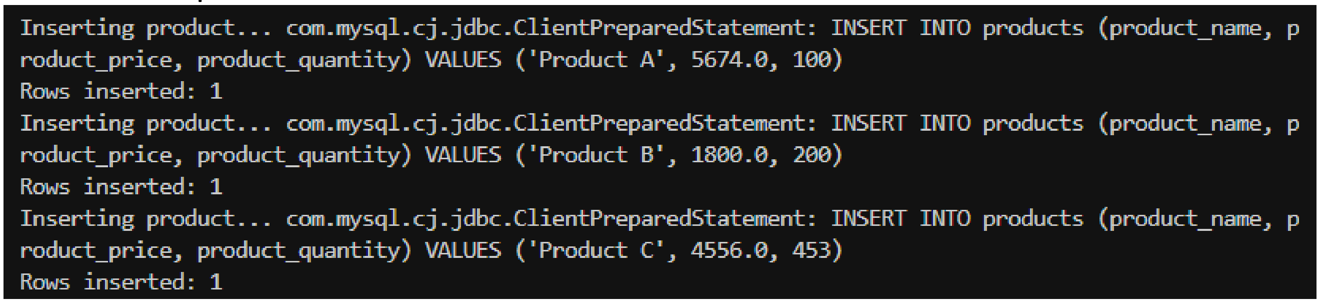
The output for all the operations is as follows-

1. Create table-



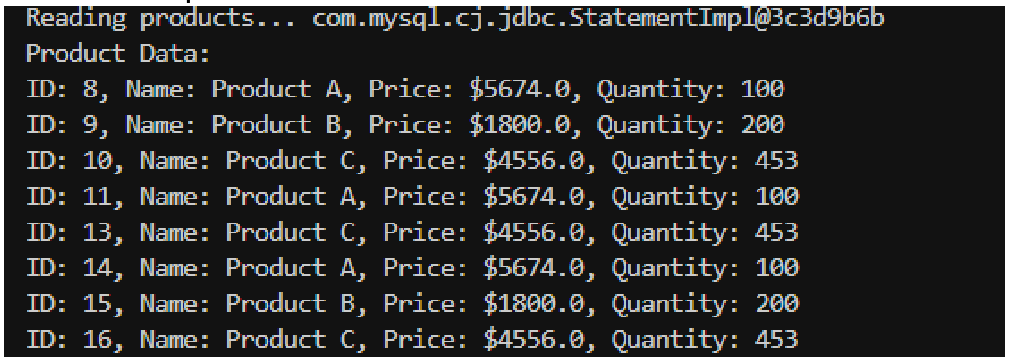
```
Creating table... CREATE TABLE IF NOT EXISTS products (product_id INT AUTO_INCREMENT PRIMARY KEY, pro
duct_name VARCHAR(100), product_price DECIMAL(10, 2), product_quantity INT)
Table created...
```

2. Insert product-



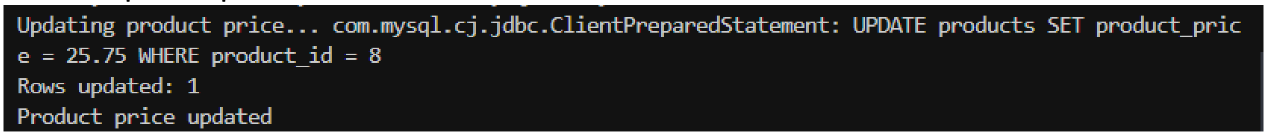
```
Inserting product... com.mysql.cj.jdbc.ClientPreparedStatement: INSERT INTO products (product_name, p
roduct_price, product_quantity) VALUES ('Product A', 5674.0, 100)
Rows inserted: 1
Inserting product... com.mysql.cj.jdbc.ClientPreparedStatement: INSERT INTO products (product_name, p
roduct_price, product_quantity) VALUES ('Product B', 1800.0, 200)
Rows inserted: 1
Inserting product... com.mysql.cj.jdbc.ClientPreparedStatement: INSERT INTO products (product_name, p
roduct_price, product_quantity) VALUES ('Product C', 4556.0, 453)
Rows inserted: 1
```

3. Read products-



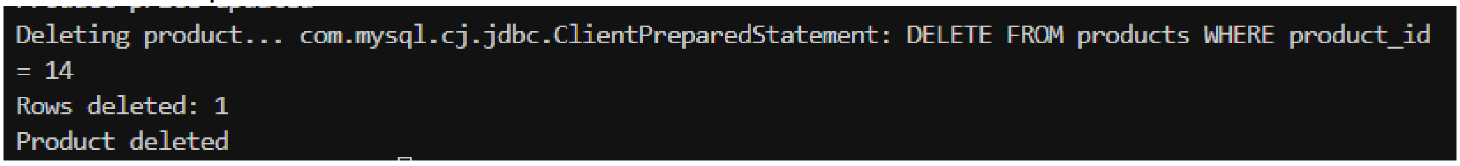
```
Reading products... com.mysql.cj.jdbc.StatementImpl@3c3d9b6b
Product Data:
ID: 8, Name: Product A, Price: $5674.0, Quantity: 100
ID: 9, Name: Product B, Price: $1800.0, Quantity: 200
ID: 10, Name: Product C, Price: $4556.0, Quantity: 453
ID: 11, Name: Product A, Price: $5674.0, Quantity: 100
ID: 13, Name: Product C, Price: $4556.0, Quantity: 453
ID: 14, Name: Product A, Price: $5674.0, Quantity: 100
ID: 15, Name: Product B, Price: $1800.0, Quantity: 200
ID: 16, Name: Product C, Price: $4556.0, Quantity: 453
```

4. Update product-



```
Updating product price... com.mysql.cj.jdbc.ClientPreparedStatement: UPDATE products SET product_pric
e = 25.75 WHERE product_id = 8
Rows updated: 1
Product price updated
```

5. Delete product-



```
Deleting product... com.mysql.cj.jdbc.ClientPreparedStatement: DELETE FROM products WHERE product_id
= 14
Rows deleted: 1
Product deleted
```