Relational Databases with MySQL Week 10 Assignment

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document to the repository. Additionally, push an .sql file with all your queries and your Java project code to the same repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

In this week's coding activity, you will create a menu driven application backed by a MySQL database.

To start, choose one item that you like. It could be vehicles, sports, foods, etc....

Create a new Java project in Eclipse.

Create a SQL script in the project to create a database with one table. The table should be the item you picked.

Write a Java menu driven application that allows you to perform all four CRUD operations on your table.

Tips:

The application does not need to be as complex as the example in the video curriculum.

You need an option for each of the CRUD operations (Create, Read, Update, and Delete).

Remember that PreparedStatment.executeQuery() is only for Reading data and .executeUpdate() is used for Creating, Updating, and Deleting data.

Remember that both parameters on PreparedStatements and the ResultSet columns are based on indexes that start with 1, not 0.

Screenshots of Code:

```java
package application;

public class Application {

    public static void main(String[] args) {
        Menu menu = new Menu();
        menu.start();
    }
}
```

```java
package application;

import dao.MedDao;
import entity.Med;
import java.sql.SQLException;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class Menu {

    private MedDao medDao = new MedDao();
    private Scanner scanner = new Scanner(System.in);
    private List<String> options = Arrays.asList(
            "Create a Med",
            "Display one Med",
            "Display all Meds",
            "Update a Med",
            "Delete a Med");

    public void start() {
        String selection = "";

        do {
            printMenu();
            selection = scanner.nextLine();

```

```java
            try {
                if (selection.equals("1")) {
                    createMed();
                } else if (selection.equals("2")) {
                    displayOneMed();
                } else if (selection.equals("3")) {
                    displayAllMeds();
                } else if (selection.equals("4")) {
                    updateMedName();
                } else if (selection.equals("5")) {
                    deleteMed();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }

            System.out.println("Press enter to continue...");
            scanner.nextLine();
        } while (!selection.equals("-1"));
    }

    private void createMed() throws SQLException {
        System.out.print("Enter medication ID NUMBER for the new medication: ");
        int medId = Integer.parseInt(scanner.nextLine());
        System.out.print("Enter generic name: ");
        String genericName = scanner.nextLine();
        medDao.createNewMed(medId, genericName);
    }

```

```java
    private void displayOneMed() throws SQLException {
        System.out.println("Search med by name: ");
        String genericName = scanner.nextLine();
        Med result = (Med) medDao.getMedByName(genericName);
    }

    private void displayAllMeds() throws SQLException {
        List<Med> allMeds = medDao.getAllMeds();
        for (Med med : allMeds) {
            System.out.println(med.getMedId() + ": " + med.getGenericName());
        }
    }

    private void updateMedName() throws SQLException {
        System.out.println("Enter the name of the med you need to update: ");
        String inputName = scanner.nextLine();
        System.out.print("Enter the updated name: ");
        String updatedName = scanner.nextLine();
        medDao.updateMedName(inputName, updatedName);
        System.out.println(inputName + " has been updated to " + updatedName);
    }

    private void deleteMed() throws SQLException {
        System.out.print("Enter the medication name to delete:");
        String genericName = scanner.nextLine();
        medDao.deleteMedByName(genericName);
    }

    private void printMenu() {
        System.out.println("Select an Option:\n-----------------------");
        for (int i = 0; i < options.size(); i++) {
            System.out.println(i + 1 + ") " + options.get(i));
        }
    }

}
```

```java
package dao;

import exception.MedsException;

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

public class DbConnection {

    private static final String HOST = "localhost";
    private static final int PORT = 3306;
    private static final String USERNAME = "crud_meds";
    private static final String PASSWORD = "crud_meds";
    private static final String SCHEMA = "crud_meds";

    public static Connection getConnection() {

        String uri = String.format("jdbc:mysql://%s:%d/%s", HOST, PORT, SCHEMA);
        System.out.println("Connection to " + uri + " successful!");
        try {
            Connection conn = DriverManager.getConnection(uri, USERNAME, PASSWORD);

            return conn;
        } catch (SQLException e) {
            System.err.println("Could not connect to " + uri);
            throw new MedsException(e);
        }
    }

}
```

```java
package dao;

import entity.Med;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class MedDao {

    private Connection connection;
    private final String CREATE_NEW_MED_QUERY = "INSERT INTO meds(med_id, generic_name) VALUES(?,?)";
//    private final String GET_MED_BY_ID_QUERY = "SELECT * FROM meds WHERE med_id = ?";
    private final String GET_MED_BY_NAME_QUERY = "SELECT * FROM meds WHERE generic_name = ?";
    private final String GET_ALL_MEDS = "SELECT *
    private final String UPDATE_MED_NAME_QUERY =                                    generic_name = ?";
    private final String DELETE_MED_QUERY = "DELETE FROM meds WHERE generic_name = ?";

    public MedDao() { connection  = DbConnection.getConnection(); }

    public void createNewMed(int medId, String genericName) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(CREATE_NEW_MED_QUERY);
        ps.setInt( parameterIndex: 1, medId);
        ps.setString( parameterIndex: 2, genericName);
        ps.executeUpdate();
    }
```

No documentation found.

```java
    public Med getMedByName(String genericName) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(GET_MED_BY_NAME_QUERY);
        ps.setString( parameterIndex: 1, genericName);
        ResultSet displayOneMed = ps.executeQuery();
        if (displayOneMed.next()) {
            Med newMed = new Med(displayOneMed.getInt( columnIndex: 1), displayOneMed.getString( columnIndex: 2));
            return newMed;
        } else {
            return null;
        }
    }

    public List<Med> getAllMeds() throws SQLException {
        ResultSet rs = connection.prepareStatement(GET_ALL_MEDS).executeQuery();
        List<Med> meds = new ArrayList<Med>();

        while (rs.next()) {
            meds.add(addMedToMedsList(rs.getInt( columnIndex: 1), rs.getString( columnIndex: 2)));
        }
        return meds;
    }

    private Med addMedToMedsList(int medId, String genericName) {
        return new Med(medId, genericName);
    }

    public void updateMedName(String inputName, String updatedName) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(UPDATE_MED_NAME_QUERY);
        ps.setString( parameterIndex: 1, updatedName);
        ps.setString( parameterIndex: 2, inputName);
        ps.executeUpdate();
    }
```

```java

    public void deleteMedByName(String genericName) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(DELETE_MED_QUERY);
        ps.setString( parameterIndex: 1, genericName);
        ps.executeUpdate();
    }

}
```

```java
package entity;

public class Med {

    private int medId;
    private String genericName;

    public Med(int medId, String genericName) {
        this.setMedId(medId);
        this.setGenericName(genericName);
    }

    public String getGenericName() {
        return genericName;
    }

    public void setGenericName(String genericName) {
        this.genericName = genericName;
    }

    public int getMedId() {
        return medId;
    }

    public void setMedId(int medId) {
        this.medId = medId;
    }

}
```

```java
package exception;

public class MedsException extends RuntimeException {

    public MedsException() {
    }

    public MedsException(String message) { super(message); }

    public MedsException(Throwable cause) { super(cause); }

    public MedsException(String message, Throwable cause) { super(message, cause); }
}
```

```sql
DROP TABLE if exists meds;

CREATE TABLE meds (
    med_id int NOT NULL AUTO_INCREMENT PRIMARY key,
    generic_name varchar(40) NOT NULL
);
```

Screenshots of Running Application: Lisa—there are a couple of bugs, I need to move on; I could play with this all day!

```
/Users/lindaforlizzi/Library/Java/JavaVirtualMachines/openjdk-17.0.
Connection to jdbc:mysql://localhost:3306/crud_meds successful!
Select an Option:
------------------------
1) Create a Med
2) Display one Med
3) Display all Meds
4) Update a Med
5) Delete a Med
```

```
Select an Option:
------------------------
1) Create a Med
2) Display one Med
3) Display all Meds
4) Update a Med
5) Delete a Med
1
Enter medication ID NUMBER for the new medication: 10
Enter generic name: haldol
Press enter to continue...
```

```
Select an Option:
------------------------
1) Create a Med
2) Display one Med
3) Display all Meds
4) Update a Med
5) Delete a Med
2
Search med by name:
tetracycline
```

```
Select an Option:
------------------------
1) Create a Med
2) Display one Med
3) Display all Meds
4) Update a Med
5) Delete a Med
3
10: haldol
20: tetracycline
```

```
1) Create a Med
2) Display one Med
3) Display all Meds
4) Update a Med
5) Delete a Med
4
Enter the name of the med you need to update:
tetracycline
Enter the updated name: not tetracycline
tetracycline has been updated to not tetracycline
```

```
1) Create a Med
2) Display one Med
3) Display all Meds
4) Update a Med
5) Delete a Med
5
Enter the medication name to delete:haldol
Press enter to continue...

Select an Option:
------------------------
1) Create a Med
2) Display one Med
3) Display all Meds
4) Update a Med
5) Delete a Med
3
20: not tetracycline
```

URL to GitHub Repository: https://github.com/thisLinda/SQLWeek10Assignment