Relational Databases with MySQL Week 5 Coding 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.

GitHub: https://github.com/Rowan-Bear/MySQL_Week4.git

Screenshots of Code:

```java
package com.rowanbear.application;

import java.sql.SQLException;

public class Menu {
    //connects menu to DB
    private favorite_food_dao favorite_food_dao = new favorite_food_dao();
    private Scanner scanner = new Scanner(System.in);
    private List<String> options = Arrays.asList(
            "Display Favorite Foods",
            "Create Favorite Food",
            "Update Favorite Food",
            "Delete Favorite Food");


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


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

            try {

            if (selection.equals("1")) {
                displayFavoriteFoods();
            }
            else if (selection.equals("2")) {
                createFavoriteFood();
            }
            else if (selection.equals("3")) {
                updateFavoriteFood();
            }
            else if (selection.equals("4")) {
                deleteFavoriteFood();
            }

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

```
            System.out.println("Press enter to continue.............");
            scanner.nextLine();

        } while(!selection.equals("-1"));
    }

    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));
        }
    }
    //Read
    private void displayFavoriteFoods() throws SQLException {
        List<Favorite_Food> foods = favorite_food_dao.getFoods();
        for (Favorite_Food food : foods) {
            System.out.println(food.getFoodName() + ": " + food.getFoodOrigin() +
": " + food.getFoodID());
        }
    }
    //Create
    private void createFavoriteFood() throws SQLException {
        System.out.println("Enter name of food: ");
        String foodName = scanner.nextLine();
        System.out.println("Enter where the food came from: ");
        String foodOrigin = scanner.nextLine();
        System.out.println("Enter the foodID");
        int foodId = Integer.parseInt(scanner.nextLine());
        favorite_food_dao.createNewFood(foodName, foodOrigin, foodId);
    }
    //Update
    private void updateFavoriteFood() throws SQLException {
        System.out.println("Changed your mind?\n------------------------\n" +
"Enter the name of the new food: ");
        String foodName = scanner.nextLine();
        System.out.println("Enter where the new food came from: ");
        String foodOrigin = scanner.nextLine();
        System.out.println("Enter the new foodID");
        int newFoodId = Integer.parseInt(scanner.nextLine());
        System.out.println("Enter the food ID you would like to update: ");
        int oldFoodId = Integer.parseInt(scanner.nextLine());
```

```java
package com.rowanbear.dao;

import java.sql.Connection;

public class favorite_food_dao {

    private Connection connection;
    //these are all the SQL commands used in the menu
    private final String GET_FOOD_QUERY = "SELECT * FROM favorite_food";
    private final String CREATE_NEW_FOOD_QUERY = "INSERT INTO favorite_food(food_name, food_origin, food_id) VALUES(?, ?, ?)";
    private final String UPDATE_EXISTING_FOOD_QUERY = "UPDATE favorite_food SET food_name = ?, food_origin = ?, food_id = ? WHERE food_id = ?";
    private final String DELETE_FOOD_QUERY = "DELETE FROM favorite_food WHERE food_name = ?"  ;

    public favorite_food_dao() {
         connection = DBConnection.getConnection();
    }
    //for display/Read method in menu
    public List<Favorite_Food> getFoods() throws SQLException {
        ResultSet rs = connection.prepareStatement(GET_FOOD_QUERY).executeQuery();
        List<Favorite_Food> Foods = new ArrayList<Favorite_Food>();

        while (rs.next()) {
            Foods.add(new Favorite_Food(rs.getString(1), rs.getString(2), rs.getInt(3)));
        }

        return Foods;
    }
    //for create method in menu
    public void createNewFood(String foodName, String foodOrigin, int foodId) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(CREATE_NEW_FOOD_QUERY);
        ps.setString(1, foodName);
        ps.setString(2, foodOrigin);
        ps.setInt(3, foodId);
        ps.executeUpdate();
    }
      String foodName = scanner.nextLine();
      favorite_food_dao.deleteFood(foodName);
   }

}

    //for update method in menu
    public void updateFood(String foodName, String foodOrigin, int newFoodId, int oldFoodId) throws SQLException{
        PreparedStatement ps = connection.prepareStatement(UPDATE_EXISTING_FOOD_QUERY);
        ps.setString(1, foodName);
        ps.setString(2, foodOrigin);
        ps.setInt(3, newFoodId);
        ps.setInt(4, oldFoodId);
        ps.executeUpdate();
    }
    // for delete method in menu
    public void deleteFood(String foodName) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(DELETE_FOOD_QUERY);
        ps.setString(1, foodName);
        ps.executeUpdate();
    }

}
```

```java
package com.rowanbear.entities;

public class Favorite_Food {

    private String foodName;
    private String foodOrigin;
    private int foodID;

        public Favorite_Food(String foodName, String foodOrigin, int foodID) {
        this.setFoodName(foodName);
        this.setFoodOrigin(foodOrigin);
        this.setFoodID(foodID);
    }



    public String getFoodOrigin() {
        return foodOrigin;
    }

    public void setFoodOrigin(String foodOrigin) {
        this.foodOrigin = foodOrigin;
    }

    public int getFoodID() {
        return foodID;
    }

    public void setFoodID(int foodID) {
        this.foodID = foodID;
    }



    public String getFoodName() {
        return foodName;
    }

```

```java
package com.rowanbear.application;
//encapsulates menu, so if you needed to add more aspects of the application
later, it would all be organized.
public class Application {

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

}
```

```java


    public void setFoodName(String foodName) {
        this.foodName = foodName;
    }



}
```

```java
package com.rowanbear.dao;
//connects to the favorite_food DB(.getConnection() is used to connect
favorite_food_dao to the database so it can use the entitity and work with the
methods in the Menu class.
import java.sql.Connection;

public class DBConnection {

private final static String URL = "jdbc:mysql://localhost:3306/favorite_food";
private final static String USERNAME = "root";
private final static String PASSWORD = "root";
private static Connection  connection;
private static DBConnection instance;

private DBConnection(Connection connection) {
    this.connection = connection;
}

public static Connection getConnection() {
    if (instance == null) {
        try {
            connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
            instance = new DBConnection(connection);
            System.out.println("Connection Successful");
        } catch(SQLException e) {
            e.printStackTrace();
        }
    }
    return DBConnection.connection;
}

}

```