

Relational Databases with MySQL Week 4 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:

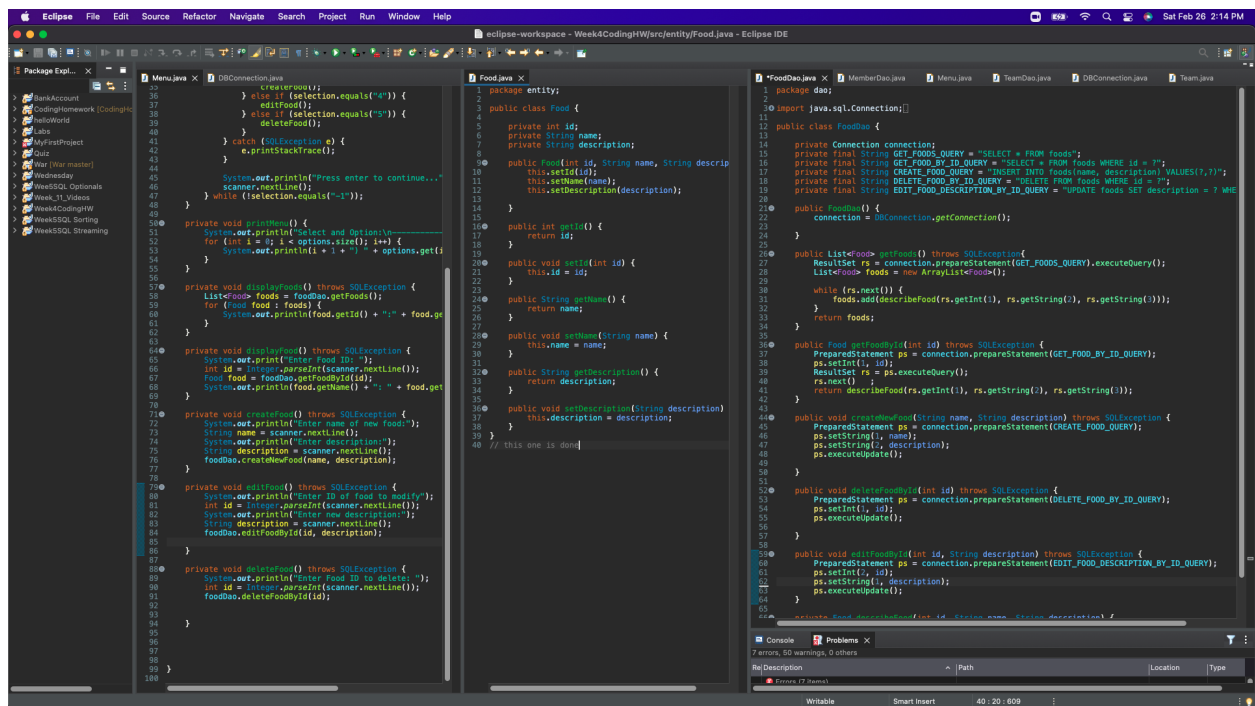
The application does not need to be as complex as the example in the video curriculum.

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

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

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

Screenshots of Code:



```
1 package entity;
2 public class Food {
3     private int id;
4     private String name;
5     private String description;
6
7     public Food(int id, String name, String description) {
8         this.setId(id);
9         this.setName(name);
10        this.setDescription(description);
11    }
12
13    public int getId() {
14        return id;
15    }
16
17    public void setId(int id) {
18        this.id = id;
19    }
20
21    public String getName() {
22        return name;
23    }
24
25    public void setName(String name) {
26        this.name = name;
27    }
28
29    public String getDescription() {
30        return description;
31    }
32
33    public void setDescription(String description) {
34        this.description = description;
35    }
36
37    // this one is done
38 }
39
40 // this one is done
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
}
```

```
1 package dao;
2
3 import java.sql.Connection;
4
5 public class FoodDao {
6     private Connection connection;
7     private final String GET_FOODS_QUERY = "SELECT * FROM foods";
8     private final String GET_FOOD_BY_ID_QUERY = "SELECT * FROM foods WHERE id = ?";
9     private final String CREATE_FOOD_QUERY = "INSERT INTO foods(name, description) VALUES(?,?)";
10    private final String DELETE_FOOD_BY_ID_QUERY = "DELETE FROM foods WHERE id = ?";
11    private final String EDIT_FOOD_DESCRIPTION_BY_ID_QUERY = "UPDATE foods SET description = ? WHERE id = ?";
12
13    public FoodDao() {
14        connection = DBConnection.getConnection();
15    }
16
17    public List<Food> getFoods() throws SQLException {
18        ResultSet rs = connection.prepareStatement(GET_FOODS_QUERY).executeQuery();
19        List<Food> foods = new ArrayList<Food>();
20
21        while (rs.next()) {
22            Food food = new Food(rs.getInt(1), rs.getString(2), rs.getString(3));
23            foods.add(food);
24        }
25        return foods;
26    }
27
28    public Food getFoodById(int id) throws SQLException {
29        PreparedStatement ps = connection.prepareStatement(GET_FOOD_BY_ID_QUERY);
30        ps.setInt(1, id);
31        ResultSet rs = ps.executeQuery();
32        rs.next();
33        return describeFood(rs.getInt(1), rs.getString(2), rs.getString(3));
34    }
35
36    public void createFood(String name, String description) throws SQLException {
37        PreparedStatement ps = connection.prepareStatement(CREATE_FOOD_QUERY);
38        ps.setString(1, name);
39        ps.setString(2, description);
40        ps.executeUpdate();
41    }
42
43    public void deleteFoodById(int id) throws SQLException {
44        PreparedStatement ps = connection.prepareStatement(DELETE_FOOD_BY_ID_QUERY);
45        ps.setInt(1, id);
46        ps.executeUpdate();
47    }
48
49    public void editFoodById(int id, String description) throws SQLException {
50        PreparedStatement ps = connection.prepareStatement(EDIT_FOOD_DESCRIPTION_BY_ID_QUERY);
51        ps.setInt(1, id);
52        ps.setString(2, description);
53        ps.executeUpdate();
54    }
55
56    private Food describeFood(int id, String name, String description) {
57        return new Food(id, name, description);
58    }
59 }
60
61 // this one is done
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
}
```

```
1 package dao;
2
3 import java.sql.Connection;
4
5 public class FoodDao {
6     private Connection connection;
7     private final String GET_FOODS_QUERY = "SELECT * FROM foods";
8     private final String GET_FOOD_BY_ID_QUERY = "SELECT * FROM foods WHERE id = ?";
9     private final String CREATE_FOOD_QUERY = "INSERT INTO foods(name, description) VALUES(?,?)";
10    private final String DELETE_FOOD_BY_ID_QUERY = "DELETE FROM foods WHERE id = ?";
11    private final String EDIT_FOOD_DESCRIPTION_BY_ID_QUERY = "UPDATE foods SET description = ? WHERE id = ?";
12
13    public FoodDao() {
14        connection = DBConnection.getConnection();
15    }
16
17    public List<Food> getFoods() throws SQLException {
18        ResultSet rs = connection.prepareStatement(GET_FOODS_QUERY).executeQuery();
19        List<Food> foods = new ArrayList<Food>();
20
21        while (rs.next()) {
22            Food food = new Food(rs.getInt(1), rs.getString(2), rs.getString(3));
23            foods.add(food);
24        }
25        return foods;
26    }
27
28    public Food getFoodById(int id) throws SQLException {
29        PreparedStatement ps = connection.prepareStatement(GET_FOOD_BY_ID_QUERY);
30        ps.setInt(1, id);
31        ResultSet rs = ps.executeQuery();
32        rs.next();
33        return describeFood(rs.getInt(1), rs.getString(2), rs.getString(3));
34    }
35
36    public void createFood(String name, String description) throws SQLException {
37        PreparedStatement ps = connection.prepareStatement(CREATE_FOOD_QUERY);
38        ps.setString(1, name);
39        ps.setString(2, description);
40        ps.executeUpdate();
41    }
42
43    public void deleteFoodById(int id) throws SQLException {
44        PreparedStatement ps = connection.prepareStatement(DELETE_FOOD_BY_ID_QUERY);
45        ps.setInt(1, id);
46        ps.executeUpdate();
47    }
48
49    public void editFoodById(int id, String description) throws SQLException {
50        PreparedStatement ps = connection.prepareStatement(EDIT_FOOD_DESCRIPTION_BY_ID_QUERY);
51        ps.setInt(1, id);
52        ps.setString(2, description);
53        ps.executeUpdate();
54    }
55
56    private Food describeFood(int id, String name, String description) {
57        return new Food(id, name, description);
58    }
59 }
60
61 // this one is done
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
}
```

Screenshots of Running Application:

```
Console Problems
<terminated> Application [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.0.1.jdk/Contents/Home/bin/java (Feb 26, 2022, 1
3) Create Food
4) Edit Food
5) Delete Food
4
Enter ID of food to modify
5
Enter new description:
test
Press enter to continue...

Select and Option:
-----
1) Display Foods
2) Display Food
3) Create Food
4) Edit Food
5) Delete Food
1
3:Carly's Stir Fry
5:Olive
7:Apple (Honeycrisp)
8:test
Press enter to continue...
2
Select and Option:
-----
1) Display Foods
2) Display Food
3) Create Food
4) Edit Food
5) Delete Food
2
Enter Food ID: 5
Olive: test
Press enter to continue...

Select and Option:
-----
1) Display Foods
2) Display Food
3) Create Food
4) Edit Food
5) Delete Food
4
Enter ID of food to modify
5
Enter new description:
These things are awful, they ruin every food.
Press enter to continue...

Select and Option:
-----
1) Display Foods
2) Display Food
3) Create Food
4) Edit Food
5) Delete Food
2
Enter Food ID: 5
Olive: These things are awful, they ruin every food.
Press enter to continue...

Select and Option:
-----
1) Display Foods
```

URL to GitHub Repository: <https://github.com/edsupler/week4SQLCodingHW.git>