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:

vehiclesDB.sql

13:2

```
1  create database if not exist vehicles;
2
3  use vehicles;
4
5  drop table if exist vehicles;
6
7  create table vehicles (
8          id int(10) not null auto_increment,
9          vehicle type varchar(10) not null,
10         make varchar(50) not null,
11         model varchar(50) not null,
12         primary key(id)
13 );
```

Application.jav | Menu.java | DBConnection.ja

```
1  package application;
2
3  public class Application {
4
5    public static void main(String[] args) {
6      Menu menu = new Menu();
7      menu.start();
8    }
9
10 }
```

```java
package application;

import java.sql.SQLException;

public class Menu {

  private VehiclesDao vehiclesDao = new VehiclesDao();
  private Scanner scanner = new Scanner(System.in);
  private List<String> options = Arrays.asList(
          "Display Vehicles",
          "Display a Vehicle",
          "Create a New Vehicle",
          "Update Vehicle Information",
          "Delete a Vehicle");

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

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

          try {
               if (selection.equals("1")) {
                   displayVehicles();
               }else if (selection.equals("2")) {
                   displayVehicle();
               }else if (selection.equals("3")) {
                   createVehicle();
               }else if (selection.equals("4")) {
                   updateVehicle();
               }else if (selection.equals("5")) {
                   deleteVehicle();
               }
          }catch (SQLException e) {
            e.printStackTrace();
          }

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

```
48
49  private void printMenu() {
50    System.out.println("Select an Option:\n-------------------------");
51    for(int i = 0; i < options.size(); i++) {
52      System.out.println(i + 1 + ") " + options.get(i));
53    }
54  }
55  private void displayVehicles() throws SQLException {
56    List<Vehicles> vehicles = vehiclesDao.getVehicles();
57    for (Vehicles vehicle : vehicles) {
58      System.out.println(vehicle.getVehiclesId() + ": " + vehicle.getVehicleType() + "| "
59                         + vehicle.getMake() + "| " + vehicle.getModel());
60    }
61  }
62  private void displayVehicle() throws SQLException {
63    System.out.println("Enter Vehicle ID: ");
64    int id = Integer.parseInt(scanner.nextLine());
65    Vehicles vehicle = vehiclesDao.getVehiclesdById(id);
66    System.out.println(vehicle.getVehiclesId() + ": " + vehicle.getVehicleType() + ", "
67        + vehicle.getMake() + ", " + vehicle.getModel());
68  }
69  private void createVehicle() throws SQLException {
70    System.out.print("Enter new vehicle type:");
71    String vehicleType = scanner.nextLine();
72    System.out.print("Enter make of vehicle:");
73    String make = scanner.nextLine();
74    System.out.print("Enter model of vehicle:");
75    String model = scanner.nextLine();
76    vehiclesDao.createNewVehicles(vehicleType, make, model);
77
78   }
79  private void updateVehicle() throws SQLException {
80    System.out.println("Enter vehicle ID:");
81    int id = Integer.parseInt(scanner.nextLine());
82    System.out.print("Enter vehicle type:");
83    String vehicleType = scanner.nextLine();
84    System.out.print("Enter make of vehicle:");
85    String make = scanner.nextLine();
86    System.out.print("Enter model of vehicle:");
87    String model = scanner.nextLine();
88    vehiclesDao.updateVehicles(id);
89   }
```

```
83    String vehicleType = scanner.nextLine();
84    System.out.print("Enter make of vehicle:");
85    String make = scanner.nextLine();
86    System.out.print("Enter model of vehicle:");
87    String model = scanner.nextLine();
88    vehiclesDao.updateVehicles(id);
89   }
90  private void deleteVehicle() throws SQLException {
91    System.out.print("Enter vehicle id to delete:");
92    int id = Integer.parseInt(scanner.nextLine());
93    vehiclesDao.deleteVehicles(id);
94
95  }
96 }
97
```

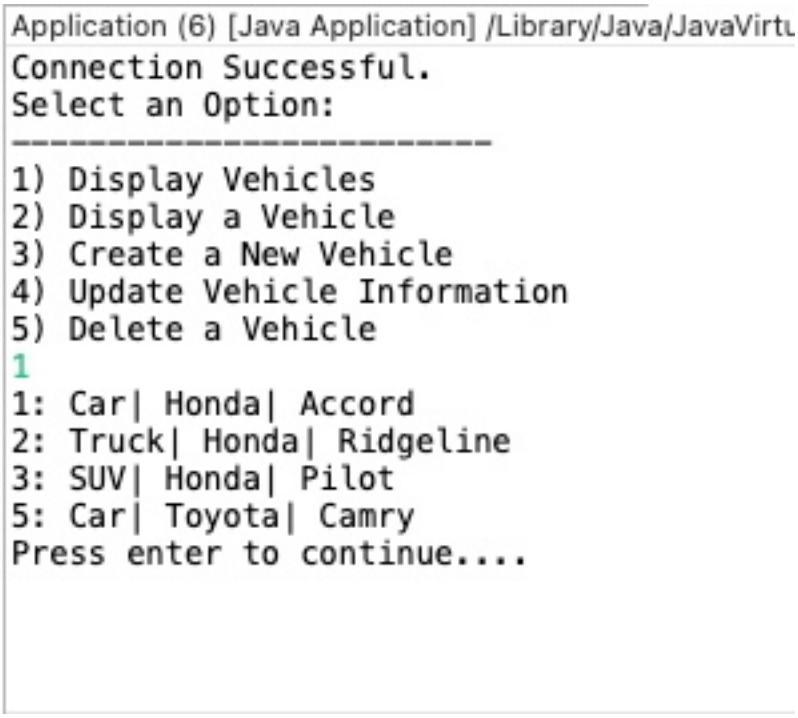
```java
package dao;

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

public class DBConnection {


  private final static String URL = "jdbc:mysql://localhost:3306/vehicles";
  private final static String USERNAME = "root";
  private final static String PASSWORD = "password1";
  private static Connection connection;
  private static DBConnection instance;

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

```

```java
package dao;

import java.sql.Connection;

public class VehiclesDao {

    private Connection connection;
    private final String GET_VEHICLES_QUERY = "SELECT * FROM vehicles";
    private final String GET_VEHICLE_BY_ID_QUERY = "SELECT * FROM vehicles WHERE id = ?";
    private final String CREATE_NEW_VEHICLE_QUERY = "INSERT INTO vehicles (vehicle_type, make, model) VALUES (?, ?, ?)";
    private final String UPDATE_VEHICLE_QUERY = "UPDATE vehicles SET vehicle_type = ?, make = ?, model = ? WHERE id = ?";
    private final String DELETE_VEHICLE_QUERY = "DELETE FROM vehicles WHERE id = ?";

    public VehiclesDao() {
      connection = DBConnection.getConnection();
    }

    public List<Vehicles> getVehicles() throws SQLException {
      ResultSet rs = connection.prepareStatement(GET_VEHICLES_QUERY).executeQuery();
          List<Vehicles> vehicles = new ArrayList<Vehicles>();

          while (rs.next()) {
              vehicles.add(generateVehiclesItem(rs));
          }

          return vehicles;
      }

      public Vehicles getVehiclesdById(int id) throws SQLException {
          PreparedStatement ps = connection.prepareStatement(GET_VEHICLE_BY_ID_QUERY);
          ps.setInt(1, id);
          ResultSet rs = ps.executeQuery();

          if (rs.next()) {
              return generateVehiclesItem(rs);
          }

          return new Vehicles(0, "", "", "");
      }

      public void createNewVehicles(String vehicletype, String make, String model) throws SQLException {
          PreparedStatement ps = connection.prepareStatement(CREATE_NEW_VEHICLE_QUERY);
          ps.setString(1, vehicletype);
          ps.setString(2, make);
          ps.setString(3, model);
          ps.executeUpdate();
      }

      public void updateVehicles (Vehicles vehicleItem) throws SQLException {
          PreparedStatement ps = connection.prepareStatement(UPDATE_VEHICLE_QUERY);

          ps.setString(1, vehicleItem.getVehicleType());
          ps.setString(2, vehicleItem.getMake());
          ps.setString(3, vehicleItem.getModel());
          ps.setInt(4, vehicleItem.getVehiclesId());

          ps.executeUpdate();
      }

      public void deleteVehicles(int id) throws SQLException {
          PreparedStatement ps = connection.prepareStatement(DELETE_VEHICLE_QUERY);
          ps.setInt(1, id);
```

```
69
70          ps.executeUpdate();
71      }
72
73      private Vehicles generateVehiclesItem(ResultSet rs) throws SQLException {
74
75          return new Vehicles(
76                  rs.getInt(1),
77                  rs.getString(2),
78                  rs.getString(3),
79                  rs.getString(4)
80          );
81
82      }
83
84      public void updateVehicles(int id) {
85
86
87      }
88  }
89
```

```
1  package entity;
2
3  public class Vehicles {
4
5    private int vehiclesId;
6    private String vehicleType;
7    private String make;
8    private String model;
9
10   public Vehicles(int vehiclesId, String vehicleType, String make, String model) {
11     this.setVehiclesId(vehiclesId);
12     this.setVehicleType(vehicleType);
13     this.setMake(make);
14     this.setModel(model);
15   }
16
17   public int getVehiclesId() {
18     return vehiclesId;
19   }
20
21   public void setVehiclesId(int vehiclesId) {
22     this.vehiclesId = vehiclesId;
23   }
24
25   public String getVehicleType() {
26     return vehicleType;
27   }
28
29   public void setVehicleType(String vehicleType) {
30     this.vehicleType = vehicleType;
31   }
32
33   public String getMake() {
34     return make;
35   }
36
37   public void setMake(String make) {
38     this.make = make;
39   }
40
41   public String getModel() {
42     return model;
43   }
44
45   public void setModel(String model) {
46     this.model = model;
47   }
48  }
```

Screenshots of Running Application:



URL to GitHub Repository:

https://github.com/AaronL1981/Java-Week-10-MySQL-Assignment.git