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.

Screenshots of Code:

src > application > Application.java > Java Language Support > Application > main

```java
package application;

public class Application {

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

src > application > Menu.java > ...

```java
package application;

import java.sql.SQLException;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

import dao.TypeDao;
import entity.Brand;
import entity.Type;

public class Menu {

    private TypeDao typeDao = new TypeDao();
    private Scanner scanner = new Scanner(System.in);
    private List<String> options = Arrays.asList(
        "Show Brands",
        "Show a Brand",
        "Create Brand",
        "Delete Brand");

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

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

            try {
                if(selection.equals("1")) {
                    showBrands();
                } else if (selection.equals("2")) {
                    showBrand();
                } else if (selection.equals("3")) {
                    createBrand();
                } else if (selection.equals("4")) {
                    deleteBrand();
                }
            } 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));
        }
    }
    private void showBrands() throws SQLException {
        List<Type> types = typeDao.getType();
        for(Type type : types) {
            System.out.println(type.getTypeId() + ": " + type.getName());
        }
    }
    private void showBrand() throws SQLException {
        System.out.println("Enter Brand id: ");
        int id = Integer.parseInt(scanner.nextLine());
        Type type = typeDao.getTypeById(id);
        System.out.println(type.getTypeId() + ": " +type.getName());
        for(Brand brand : type.getBrands()) {
            System.out.println("\tBrandId: " + brand.getBrandId() + "~Name: " + brand.getModel() + " " + brand.getType());
        }
    }
    private void createBrand() throws SQLException {
        System.out.println("Enter new brand: ");
        String brandName = scanner.nextLine();
        typeDao.createNewBrand(brandName);
    }
    private void deleteBrand() throws SQLException {
        System.out.print("Enter brand Id to delete: ");
        int carId = Integer.parseInt(scanner.nextLine());
        typeDao.deleteBrandByCarId(carId);
    }
}

```

src > dao > BrandDao.java > Java Language Support > BrandDao > deleteBrandByCarId

```
package dao;

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

import entity.Brand;

public class BrandDao {
    private Connection connection;
    private final String GET_BRANDS_BY_TYPE_ID_QUERY = "SELECT * FROM brands WHERE brand_id = ?";
    private final String DELETE_BRANDS_BY_BRAND_ID_QUERY = "DELETE FROM brands WHERE brand_id = ?";

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

    public List<Brand> getBrandsByTypeId(int typeId) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(GET_BRANDS_BY_TYPE_ID_QUERY);
        ps.setInt(1, typeId);
        ResultSet rs = ps.executeQuery();
        List<Brand> brands = new ArrayList<Brand>();

        while (rs.next()) {
            brands.add(new Brand(rs.getInt(1), rs.getString(2), rs.getString(3)));
        }
        return brands;
    }

    public void deleteBrandByCarId(int carId) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(DELETE_BRANDS_BY_BRAND_ID_QUERY);
        ps.setInt(1, carId);
        ps.executeUpdate();
    }

}

```

src > dao > DBConnection.java > Java Language Support > DBConnection

```
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 = "T4afarbd!";
    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 to the Database.");
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        return DBConnection.connection;
    }
}

```

src > dao > TypeDao.java

```
package dao;

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

import entity.Type;

public class TypeDao {

    private Connection connection;
    private BrandDao brandDao;
    private final String GET_BRAND_QUERY = "SELECT * FROM cars";
    private final String GET_BRAND_BY_ID_QUERY = "SELECT * FROM cars WHERE car_id = ?";
    private final String CREATE_NEW_BRAND_QUERY = "INSERT INTO cars(brand_name) VALUES(?)";
    private final String DELETE_BRAND_BY_ID_QUERY = "DELETE FROM cars WHERE car_id = ?";


    public TypeDao() {
        connection = DBConnection.getConnection();
        brandDao = new BrandDao();
    }

    public List<Type> getType() throws SQLException {
        ResultSet rs = connection.prepareStatement(GET_BRAND_QUERY).executeQuery();
        List<Type> types = new ArrayList<Type>();

        while (rs.next()) {
            types.add(populateType(rs.getInt(1), rs.getString(2)));
        }
        return types;
    }
    public Type getTypeById(int id) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(GET_BRAND_BY_ID_QUERY);
        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();
        rs.next();
        return populateType(rs.getInt(1), rs.getString(2));
    }

    public void createNewBrand (String brandName) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(CREATE_NEW_BRAND_QUERY);
```

src > dao > TypeDao.java

```
        ps.setString(1, brandName);
        ps.executeUpdate();
    }

    public void deleteBrandByCarId(int carId) throws SQLException {
        brandDao.deleteBrandByCarId(carId);
        PreparedStatement ps = connection.prepareStatement(DELETE_BRAND_BY_ID_QUERY);
        ps.setInt(1, carId);
        ps.executeUpdate();
    }

    private Type populateType(int id, String name) throws SQLException {
        return new Type(id, name, brandDao.getBrandsByTypeId(id));
    }

}

```

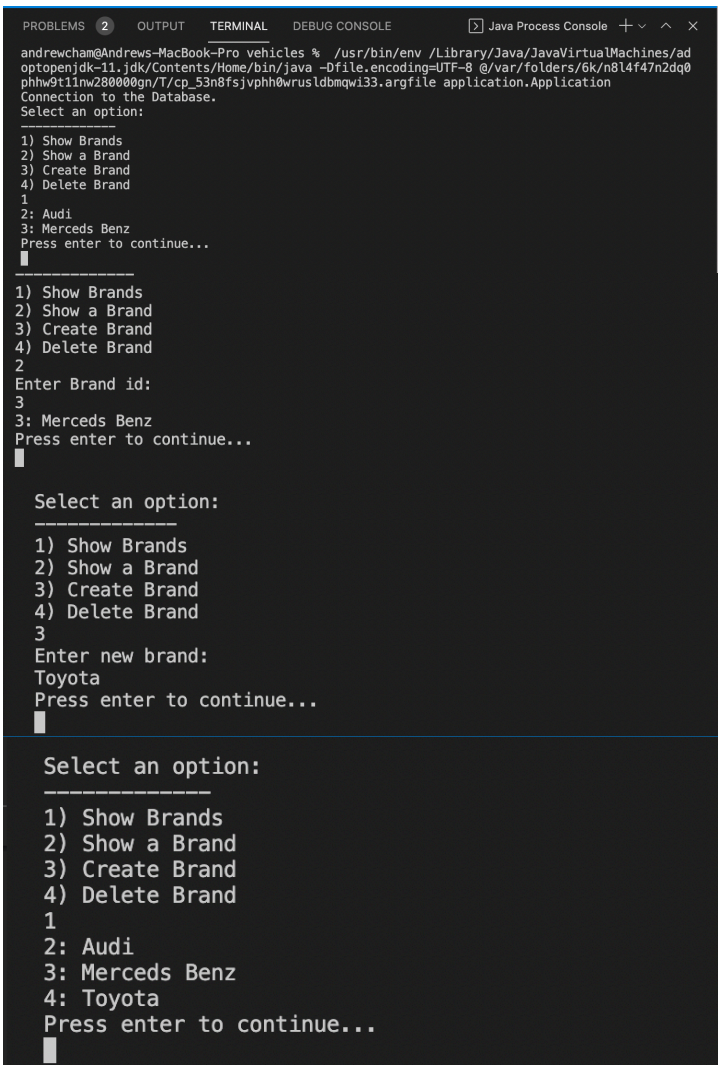
src > entity > Brand.java > Java Language Support > Brand > setType

```
package entity;

public class Brand {
    private int brandId;
    private String model;
    private String type;

    public Brand(int brandId, String model, String type) {
        this.brandId = brandId;
        this.model = model;
        this.type = type;
    }

    public int getBrandId() {
        return brandId;
    }
    public void setBrandId(int brandId) {
        this.brandId = brandId;
    }
    public String getModel() {
        return model;
    }
    public void setModel(String model) {
        this.model = model;
    }
    public String getType() {
        return type;
    }
    public void setType(String type) {
        this.type = type;
    }
}

```

src > entity > Type.java > Java Language Support > Type > getName

```
package entity;
import java.util.List;

public class Type {

    private int typeId;
    private String name;
    private List<Brand> brands;

    public Type(int typeId, String name, List<Brand> brands) {
        this.typeId = typeId;
        this.name = name;
        this.brands = brands;
    }
    public int getTypeId() {
        return typeId;
    }
    public void setTypeId(int typeId) {
        this.typeId = typeId;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public List<Brand> getBrands() {
        return brands;
    }
    public void setBrands(List<Brand> brands) {
        this.brands = brands;
    }
}

```

Screenshots of Running Application:



URL to GitHub Repository:

[Vehicles repository]