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.

URL to GitHub Repository:

https://github.com/ladymanj/Week10CodingAssignment.git

Screenshots of Code:

Application.java:

```java
package application;

public class Application {

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

}

```

DBConnection.java:

```java
package dao;

import java.sql.Connection;

public class DBConnection {

    private final static String URL = "jdbc:mysql://localhost:3306/board_games";
    private final static String USERNAME = "board_games";
    private final static String PASSWORD = "board_games";
    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;
    }

}

```

BoardGame.java:

```java
package entity;

public class BoardGame {

    private int id;
    private String name;
    private int minPlayerCount;
    private int maxPlayerCount;

    public BoardGame(int id, String name, int minPlayerCount, int maxPlayerCount) {
        this.setId(id);
        this.setName(name);
        this.setMinPlayerCount(minPlayerCount);
        this.setMaxPlayerCount(maxPlayerCount);
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getMinPlayerCount() {
        return minPlayerCount;
    }

    public void setMinPlayerCount(int minPlayerCount) {
        this.minPlayerCount = minPlayerCount;
    }

    public int getMaxPlayerCount() {
        return maxPlayerCount;
    }

    public void setMaxPlayerCount(int maxPlayerCount) {
        this.maxPlayerCount = maxPlayerCount;
    }

}
```

Menu.java:

```java
package application;

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

import dao.BoardGameDao;
import entity.BoardGame;

public class Menu {

    private BoardGameDao boardGameDao = new BoardGameDao();
    private Scanner scanner = new Scanner(System.in);
    private List<String> options = Arrays.asList(
            "Display Board Games",
            "Display Board Game",
            "Add Board Game",
            "Delete Board Game",
            "Update Board Game");

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

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

            try {
                if (selection.equals("1")) {
                    displayBoardGames();
                } else if (selection.equals("2")) {
                    displayBoardGame();
                } else if (selection.equals("3")) {
                    addBoardGame();
                } else if (selection.equals("4")) {
                    deleteBoardGame();
                } else if (selection.equals("5")) {
                    updateBoardGame();
                }
            } catch (SQLException e) {
```

Menu.java: (continued)

```
                e.printStackTrace();
            }

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

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

    }
    private void displayBoardGames() throws SQLException {
        List<BoardGame> boardGames = boardGameDao.getBoardGames();
        for (BoardGame boardGame : boardGames) {
            System.out.println(boardGame.getId() + ": " + boardGame.getName());
        }

    }

    private void displayBoardGame() throws SQLException {
        System.out.print("Enter Board Game Id: ");
        int id = Integer.parseInt(scanner.nextLine());
        BoardGame boardGame = boardGameDao.getBoardGameById(id);
        System.out.println(boardGame.getId() + ": " + boardGame.getName() +
                           "\t\tPlayer Count: " + boardGame.getMinPlayerCount() +
                           "-" + boardGame.getMaxPlayerCount());
    }

    private void addBoardGame() throws SQLException {
        System.out.print("Enter New Board Game: ");
        String boardGameName = scanner.nextLine();
        System.out.print("Enter Minimum Number of Players: ");
        int minPlayerCount = Integer.parseInt(scanner.nextLine());
        System.out.print("Enter Maximum Number of Players: ");
        int maxPlayerCount = Integer.parseInt(scanner.nextLine());
        boardGameDao.addBoardGame(boardGameName, minPlayerCount, maxPlayerCount);
    }

    private void deleteBoardGame() throws SQLException {
        System.out.print("Enter Board Game Id to Delete: ");
        int id = Integer.parseInt(scanner.nextLine());
        boardGameDao.deleteBoardGameById(id);
    }

    private void updateBoardGame() throws SQLException {
        System.out.print("Enter Board Game Id to Update: ");
        int id = Integer.parseInt(scanner.nextLine());

        String selection = "";

        do {
            System.out.println("Please select what you would like to change:\n\n" +
                               "\t1) Name\n" +
                               "\t2) Minimum Player Count\n" +
                               "\t3) Maximum Player Count\n" +
                               "\t4) Submit Changes");
            selection = scanner.nextLine();

            if (selection.equals("1")) {
                updateBoardGameName(id);
            } else if (selection.equals("2")) {
                updateBoardGameMinPlayerCount(id);
            } else if (selection.equals("3")) {
                updateBoardGameMaxPlayerCount(id);
            }
        } while (!selection.equals("4"));

    }

    private void updateBoardGameName(int id) throws SQLException {
        System.out.print("Enter New Name: ");
        String newName = scanner.nextLine();
        boardGameDao.updateBoardGameName(id, newName);
    }

    private void updateBoardGameMinPlayerCount(int id) throws SQLException {
        System.out.print("Enter New Minimum Player Count: ");
        int newMinPlayerCount = Integer.parseInt(scanner.nextLine());
        boardGameDao.updateBoardGameMinPlayerCount(id, newMinPlayerCount);
    }

    private void updateBoardGameMaxPlayerCount(int id) throws SQLException {
        System.out.print("Enter New Maximum Player Count: ");
        int newMaxPlayerCount = Integer.parseInt(scanner.nextLine());
        boardGameDao.updateBoardGameMaxPlayerCount(id, newMaxPlayerCount);
    }

}
```

BoardGameDao.java:

```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.BoardGame;

public class BoardGameDao {

    private Connection connection;
    private final String GET_BOARD_GAMES_QUERY = "SELECT * FROM board_games";
    private final String GET_BOARD_GAME_BY_ID_QUERY = "SELECT * FROM board_games WHERE id = ?";
    private final String ADD_BOARD_GAME_QUERY = "INSERT INTO board_games(name, min_player_count, max_player_count) VALUES (?, ?, ?)";
    private final String DELETE_BOARD_GAME_BY_ID_QUERY = "DELETE FROM board_games WHERE id = ?";
    private final String UPDATE_BOARD_GAME_NAME_QUERY = "UPDATE board_games SET name = ? WHERE id = ?";
    private final String UPDATE_BOARD_GAME_MIN_PLAYER_COUNT_QUERY = "UPDATE board_games SET min_player_count = ? WHERE id = ?";
    private final String UPDATE_BOARD_GAME_MAX_PLAYER_COUNT_QUERY = "UPDATE board_games SET max_player_count = ? WHERE id = ?";

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

    public List<BoardGame> getBoardGames() throws SQLException {
        ResultSet rs = connection.prepareStatement(GET_BOARD_GAMES_QUERY).executeQuery();
        List<BoardGame> boardGames = new ArrayList<BoardGame>();

        while (rs.next()) {
            boardGames.add(populateBoardGame(rs.getInt(1), rs.getString(2), rs.getInt(3), rs.getInt(4)));
        }

        return boardGames;
    }

    public BoardGame getBoardGameById(int id) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(GET_BOARD_GAME_BY_ID_QUERY);
        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();
        rs.next();
        return populateBoardGame(rs.getInt(1), rs.getString(2), rs.getInt(3), rs.getInt(4));
    }

    public void addBoardGame(String name, int minPlayerCount, int maxPlayerCount) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(ADD_BOARD_GAME_QUERY);
        ps.setString(1, name);
        ps.setInt(2, minPlayerCount);
        ps.setInt(3, maxPlayerCount);
        ps.executeUpdate();
    }

    public void deleteBoardGameById(int id) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(DELETE_BOARD_GAME_BY_ID_QUERY);
        ps.setInt(1, id);
        ps.executeUpdate();
    }

    public void updateBoardGameName(int id, String newName) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(UPDATE_BOARD_GAME_NAME_QUERY);
        ps.setString(1, newName);
        ps.setInt(2, id);
        ps.executeUpdate();
    }

    public void updateBoardGameMinPlayerCount(int id, int newMinPlayerCount) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(UPDATE_BOARD_GAME_MIN_PLAYER_COUNT_QUERY);
        ps.setInt(1, newMinPlayerCount);
        ps.setInt(2, id);
        ps.executeUpdate();
    }

    public void updateBoardGameMaxPlayerCount(int id, int newMaxPlayerCount) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(UPDATE_BOARD_GAME_MAX_PLAYER_COUNT_QUERY);
        ps.setInt(1, newMaxPlayerCount);
        ps.setInt(2, id);
        ps.executeUpdate();
    }

    private BoardGame populateBoardGame(int id, String name, int minPlayerCount, int maxPlayerCount) {
        return new BoardGame(id, name, minPlayerCount, maxPlayerCount);
    }
}

```

Screenshots of Running Application:

```
Connection successful.
Please select an option below:

1) Display Board Games
2) Display Board Game
3) Add Board Game
4) Delete Board Game
5) Update Board Game
3
Enter New Board Game: Catan
Enter Minimum Number of Players: 3
Enter Maximum Number of Players: 4
Press enter to continue.

Please select an option below:

1) Display Board Games
2) Display Board Game
3) Add Board Game
4) Delete Board Game
5) Update Board Game
1
1: Catan
Press enter to continue.

Please select an option below:

1) Display Board Games
2) Display Board Game
3) Add Board Game
4) Delete Board Game
5) Update Board Game
2
Enter Board Game Id: 1
1: Catan                Player Count: 3-4
Press enter to continue.

Please select an option below:

1) Display Board Games
2) Display Board Game
3) Add Board Game
4) Delete Board Game
5) Update Board Game
5
Enter Board Game Id to Update: 1
Please select what you would like to change:

        1) Name
        2) Minimum Player Count
        3) Maximum Player Count
        4) Submit Changes
1
Enter New Name: The Settlers of Catan
Please select what you would like to change:

        1) Name
        2) Minimum Player Count
        3) Maximum Player Count
        4) Submit Changes
3
Enter New Maximum Player Count: 6
Please select what you would like to change:

        1) Name
        2) Minimum Player Count
        3) Maximum Player Count
        4) Submit Changes
4
Press enter to continue.

Please select an option below:

1) Display Board Games
2) Display Board Game
3) Add Board Game
4) Delete Board Game
5) Update Board Game
2
Enter Board Game Id: 1
1: The Settlers of Catan                Player Count: 3-6
Press enter to continue.

Please select an option below:

1) Display Board Games
2) Display Board Game
3) Add Board Game
4) Delete Board Game
5) Update Board Game
4
Enter Board Game Id to Delete: 1
Press enter to continue.

Please select an option below:

1) Display Board Games
2) Display Board Game
3) Add Board Game
4) Delete Board Game
5) Update Board Game
1
Press enter to continue.

Please select an option below:

1) Display Board Games
2) Display Board Game
3) Add Board Game
4) Delete Board Game
5) Update Board Game
-1
Press enter to continue.
```

Screenshots of SQL Script:

```
CREATE DATABASE IF NOT EXISTS board_games;

USE board_games;

DROP TABLE IF EXISTS board_games;

CREATE TABLE board_games (
                id int(11) NOT NULL AUTO_INCREMENT,
        name varchar(50) NOT NULL,
        min_player_count int(3) NOT NULL,
        max_player_count int(3) NOT NULL,
        PRIMARY KEY(id)
);
```