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, with your Java project code, to the repository. Lastly, in the Learning Management System, click the "Add Submission" button and paste the URL to your GitHub repository.

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:

Game.java

*Game.java

```
1  package entity;
2
3
4      public class Game {
5
6          private int gameId;
7          private String name;
8          private int minNumberPlayers;
9          private int maxNumberPlayers;
10         private int playTime;
11         private double difficultyRating;
12         private String mechanism;
13
14         public Game (int gameId, String name, int minNumberPlayers, int maxNumberPlayers, int playTime,
15                 double difficultyRating, String mechanism ) {
16             this.setGameId(gameId);
17             this.setName(name);
18             this.setMinNumberPlayers(minNumberPlayers);
19             this.setMaxNumberPlayers(maxNumberPlayers);
20             this.setPlayTime(playTime);
21             this.setDifficultyRating(difficultyRating);
22             this.setMechanism(mechanism);
23         }
24
25         public int getGameId() {
26             return gameId;
27         }
28
29         public void setGameId(int gameId) {
30             this.gameId = gameId;
31         }
32
33         public String getName() {
34             return name;
35         }
36
37         public void setName(String name) {
38             this.name = name;
39         }
40
41         public int getMinNumberPlayers() {
42             return minNumberPlayers;
43         }
44
45         public void setMinNumberPlayers(int minNumberPlayers) {
46             this.minNumberPlayers = minNumberPlayers;
47         }
48
49         public int getMaxNumberPlayers() {
50             return maxNumberPlayers;
51         }
55         }
56         public int getPlayTime() {
57             return playTime;
58         }
59
60         public void setPlayTime(int playTime) {
61             this.playTime = playTime;
62         }
63
64         public double getDifficultyRating() {
65             return difficultyRating;
66         }
67
68         public void setDifficultyRating(double difficultyRating) {
69             this.difficultyRating = difficultyRating;
70         }
71
72         public String getMechanism() {
73             return mechanism;
74         }
75
76         public void setMechanism(String mechanism) {
77             this.mechanism = mechanism;
78         }
79
80
81     }
82
83
```

DBConnection.java

```java
package dao;

import java.sql.Connection;

public class DBConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/board_game_database";
    private static final String USER_NAME = "root";
    private static final String PASSWORD = "";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER_NAME, PASSWORD);

    }

}

```

GameDao.java

```java
package dao;

import java.sql.Connection;

public class GameDao {

    public void createBoardGame (String name, int minNumberPlayers,
            int maxNumberPlayers, int playTime, double difficultyRating, String mechanism) {
        try (Connection connection = DBConnection.getConnection()) {
            String create = "INSERT INTO games (name, min_number_players, max_number_players, play_time, difficulty_rating, mechanism) VALUES (?,?,?,?,?,?)";

            try(PreparedStatement statement = connection.prepareStatement(create)) {
                statement.setString(1, name);
                statement.setInt(2, minNumberPlayers);
                statement.setInt(3, maxNumberPlayers);
                statement.setInt(4, playTime);
                statement.setDouble(5, difficultyRating);
                statement.setString(6, mechanism);
                statement.execute();
            }

        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }

    public List<Game> listAllGames() throws SQLException {
        try (Connection connection = DBConnection.getConnection()) {
        String select = "SELECT * FROM games";

        try(PreparedStatement statement = connection.prepareStatement(select)) {
            try(ResultSet rs = statement.executeQuery()) {
                List<Game> games = new ArrayList<>(); {
                    while (rs.next()) {
                        int gameId = rs.getInt("id");
                        String name = rs.getString("name");
                        int minNumberPlayers = rs.getInt("min_number_players");
                        int maxNumberPlayers = rs.getInt("max_number_players");
                        int playTime = rs.getInt("play_time");
                        double difficultyRating = rs.getDouble("difficulty_rating");
                        String mechanism = rs.getString("mechanism");

                        Game game = new Game(gameId, name, minNumberPlayers, maxNumberPlayers, playTime, difficultyRating, mechanism);
                        games.add(game);
                }
        return games;
        }

    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
        }
}
    }
    public void modifyGameTime(int gameId, int playTime) {
         try (Connection connection = DBConnection.getConnection()) {
             String update = "UPDATE games SET play_time=? WHERE id =?";

             try(PreparedStatement statement = connection.prepareStatement(update)) {
                 statement.setInt(1, playTime);
                 statement.setInt(2, gameId);

                 statement.executeUpdate();
             }
             }
         catch(SQLException e) {
             throw new RuntimeException(e);
         }
     }

    public void deleteBoardGame(int gameId) {
         try (Connection connection = DBConnection.getConnection()) {
             String delete = "DELETE FROM games WHERE id =?";

             try(PreparedStatement statement = connection.prepareStatement(delete)) {
                 statement.setInt(1, gameId);

                 statement.executeUpdate();
             }
             }
         catch(SQLException e) {
             throw new RuntimeException(e);
         }

     }

}
```

Menu.java

```java
package application;

import java.sql.SQLException;

public class Menu {

    private Scanner scanner = new Scanner(System.in);
    private GameDao gameDao = new GameDao();


    public void start() {
        boolean done = false;

        while(!done) {
            printMenu();

            try {
                switch(scanIntValue("Enter a menu item: ")){
                case -1:
                done = true;
                break;

                case 1:
                    listBoardGames();
                    break;

                case 2:
                    addBoardGame();
                    break;

                case 3:
                    modifyBoardGameTime();
                    break;

                case 4:
                    deleteBoardGame();
                    break;

                default:
                    System.out.println("Enter a value from 1 to 4!");
                break;
                }
            }

            catch(Exception e) {
                System.out.println("Error!" + e.toString());
                done = true;
            }
        }
        System.out.println("Goodbye!");
    }

    private void deleteBoardGame() {
        System.out.println("You're deleting a Board Game!");
        int gameId = scanIntValue("Enter the Board Game ID:");

        gameDao.deleteBoardGame(gameId);
        System.out.println("You deleted a Board Game, hope you got some money for it.");


    }

    private void modifyBoardGameTime() {
        System.out.println("You're changing a Board Game's playtime");
        int gameId = scanIntValue("Enter the Board Game ID:");
        int playTime = scanIntValue("Enter the new play time:");

        gameDao.modifyGameTime(gameId, playTime);
        System.out.println("Board Game play time changed to: " + playTime);

    }

    private void addBoardGame() {
        System.out.println("You're adding a Board Game");
        String name = scanStringValue("Enter Board Game Name:");
        int minPlayers = scanIntValue("Minimum number of players:");
        int maxPlayers = scanIntValue("Maximum number of players:");
        int playTime = scanIntValue("Playtime(in minutes):");
        double difficultyRating =scanDoubleValue("Difficulty Rating:");
        String mechanism = scanStringValue("Mechanism:");

        gameDao.createBoardGame(name, minPlayers, maxPlayers, playTime, difficultyRating, mechanism);

        System.out.println("You created a Board Game with name " + name + "\n minimum number of players: " + minPlayers
                + "\n maximum number of players: " + maxPlayers + "\n Playtime of: " + playTime + "\n Difficulty Rating of: "
                + difficultyRating + "\n and the mechanism is: " + mechanism);

    }

    private void listBoardGames() throws SQLException {
        System.out.println("Here are all the Board Games:");
        List<Game> games = gameDao.listAllGames();

        System.out.println();

        for(Game game : games) {
            System.out.println("   " + game.getGameId() + ": " + game.getName() + ": the playtime is "+ game.getPlayTime() +
                    " minutes, the max number of players is " + game.getMaxNumberPlayers()+ ", the difficulty rating is " + game.getDifficultyRating() +
                    ", and it is a " + game.getMechanism() + " game.");
        }

```

```
106
107
108     }
109
110     private String scanStringValue(String message) {
111             System.out.print(message);
112             return scanner.nextLine();
113         }
114
115     private int scanIntValue(String message) {
116             System.out.print(message);
117             String value = scanner.nextLine();
118         return Integer.parseInt(value);
119         }
120
121     private double scanDoubleValue(String message) {
122         System.out.print(message);
123         String value = scanner.nextLine();
124     return Double.parseDouble(value);
125     }
126
127     private void printMenu() {
128         System.out.println();
129         System.out.println("1) List all Board Games");
130         System.out.println("2) Add a Board Game");
131         System.out.println("3) Modify a Board Game's Playtime");
132         System.out.println("4) Delete a Board Game");
133         System.out.println("-1) to quit");
134     }
135
136
137 }
138
```

Application.java

```
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
11 }
12
```

Screenshots of Running Application:

```
1) List all Board Games
2) Add a Board Game
3) Modify a Board Game's Playtime
4) Delete a Board Game
-1) to quit
Enter a menu item: 1
Here are all the Board Games:

   1: Gloomhaven: the playtime is 120 minutes, the max number of players is 4, the difficulty rating is 3.5, and it is a Cooperative game.
   3: The Resistance : the playtime is 60 minutes, the max number of players is 10, the difficulty rating is 1.61, and it is a Team game.
   4: Revolution!: the playtime is 60 minutes, the max number of players is 4, the difficulty rating is 2.04, and it is a Competitive game.
   6: Street Fighter: the playtime is 45 minutes, the max number of players is 5, the difficulty rating is 2.11, and it is a Competitive game.
```

```
Enter a menu item: 2
You're adding a Board Game
Enter Board Game Name:King of New York
Minimum number of players:2
Maximum number of players:6
Playtime(in minutes):40
Difficulty Rating:1.87
Mechanism:Competitive
You created a Board Game with name King of New York
 minimum number of players: 2
 maximum number of players: 6
 Playtime of: 40
 Difficulty Rating of: 1.87
 and the mechanism is: Competitive
```

```
1) List all Board Games
2) Add a Board Game
3) Modify a Board Game's Playtime
4) Delete a Board Game
-1) to quit
Enter a menu item: 3
You're changing a Board Game's playtime
Enter the Board Game ID:7
Enter the new play time:45
Board Game play time changed to: 45
```

```
Enter a menu item: 1
Here are all the Board Games:

   1: Gloomhaven: the playtime is 120 minutes, the max number of players is 4, the difficulty rating is 3.5, and it is a Cooperative game.
   3: The Resistance : the playtime is 60 minutes, the max number of players is 10, the difficulty rating is 1.61, and it is a Team game.
   4: Revolution!: the playtime is 60 minutes, the max number of players is 4, the difficulty rating is 2.04, and it is a Competitive game.
   6: Street Fighter: the playtime is 45 minutes, the max number of players is 5, the difficulty rating is 2.11, and it is a Competitive game.
   7: King of New York: the playtime is 45 minutes, the max number of players is 6, the difficulty rating is 1.87, and it is a Competitive game.
```

```
Enter a menu item: 4
You're deleting a Board Game!
Enter the Board Game ID:6
You deleted a Board Game, hope you got some money for it.
```

```
Enter a menu item: 1
Here are all the Board Games:

   1: Gloomhaven: the playtime is 120 minutes, the max number of players is 4, the difficulty rating is 3.5, and it is a Cooperative game.
   3: The Resistance : the playtime is 60 minutes, the max number of players is 10, the difficulty rating is 1.61, and it is a Team game.
   4: Revolution!: the playtime is 60 minutes, the max number of players is 4, the difficulty rating is 2.04, and it is a Competitive game.
   7: King of New York: the playtime is 45 minutes, the max number of players is 6, the difficulty rating is 1.87, and it is a Competitive game.
```

URL to GitHub Repository:

https://github.com/Christinalytle/boardgamelibrary.git