

# Relational Databases with MySQL Week 4 Coding Assignment

Points possible: 70

Category	Criteria	% of Grade
<b>Functionality</b>	Does the code work?	25
<b>Organization</b>	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
<b>Creativity</b>	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
<b>Completeness</b>	All requirements of the assignment are complete.	25

**Instructions:** Complete the coding steps. Take screenshots of the steps 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 the Java project 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:

1. Create a class of whatever type you want (Animal, Person, Camera, Cheese, etc.).
  - a. Do not implement the Comparable interface.
  - b. Add a name instance variable so that you can tell the objects apart.
  - c. Add getters, setters and/or a constructor as appropriate.
  - d. Add a `toString` method that returns the name and object type (like "Pentax Camera").
  - e. Create a static method named `compare` in the class that returns an int and takes two of the objects as parameters. Return -1 if parameter 1 is "less than" parameter 2. Return 1 if parameter 1 is "greater than" parameter 2. Return 0 if the two parameters are "equal".

- f. Create a static list of these objects, adding at least 4 objects to the list.
  - g. In another class, write a method to sort the objects using a Lambda expression using the compare method you created earlier.\*\*\*\*\*
  - h. Write a method to sort the objects using a Method Reference to the compare method you created earlier.
    - i. Create a main method to call the sort methods.
    - j. Print the list after sorting (System.out.println).
2. Create a new class with a main method. Using the list of objects you created in the prior step.
  - a. Create a Stream from the list of objects.
  - b. Turn the Stream of object to a Stream of String (use the map method for this).
  - c. Sort the Stream in the natural order. (Note: The String class implements the Comparable interface, so you won't have to supply a Comparator to do the sorting.)
  - d. Collect the Stream and return a comma-separated list of names as a single String.  
Hint: use Collectors.joining(", ") for this.
  - e. Print the resulting String.
3. Create a new class with a main method. Create a method (method a) that accepts an Optional of some type of object (Animal, Person, Camera, etc.).
  - a. The method should return the object unwrapped from the Optional if the object is present. For example, if you have an object of type Cheese, your method signature should look something like this:

```
public Cheese cheesyMethod(Optional<Cheese> optionalCheese) { ... }
```
  - b. The method should throw a NoSuchElementException with a custom message if the object is not present.
  - c. Create another method (method b) that calls method a with an object wrapped by an Optional. Show that the object is returned unwrapped from the Optional (i.e., print the object).
  - d. Method b should also call method a with an empty Optional. Show that a NoSuchElementException is thrown by method a by printing the exception message. Hint: catch the NoSuchElementException as parameter named "e" and do System.out.println(e.getMessage()).

- e. Note: your method should handle the Optional as shown in the video on Optionals using the orElseThrow method. For the missing object, you must use a Lambda expression in orElseThrow to return a NoSuchElementException with a custom message.

**Screenshots:**

```
Users > borisgoetz > Desktop > Pokémon
1  create database if not exists pokémon;
2
3  use pokémon;
4
5  drop table if exists pokémon_;
6  drop table if exists pokémon_type;
7
8  create table pokémon_type (
9      pokedex_no int(10) not null,
10     mon_name VARCHAR(50) not null,
11    primary_type VARCHAR(50) not null,
12   secondary_type VARCHAR(50) not null,
13      primary key(pokedex_no)
14  );
15
16  create table pokémon_ (
17      pokedex_no int(10) not null,
18     mon_name VARCHAR(50) not null,
19    move_one VARCHAR(25) not null,
20    move_two VARCHAR(25) not null,
21    attack_iv int(10) not null,
22    defense_iv int(10) not null,
23    hp_iv int(10) not null,
24    gen_no int(10) not null
25  );
26
27
```

Caption

DBeaver 21.2.4 - <Sample1> Script-5

Database Navigator    Projects

Enter a part of object name here

Sample1 - localhost:3306

- Databases
- MySQLWeek3
- Pokémon
  - Tables
    - pokémon\_
  - Constraints
  - Foreign Keys
  - References
  - Triggers
  - Indexes

Project - General

DataSource

Name

Bookmarks

ER Diagrams

Scripts

Script-5

```
select * from Pokémon.pokémon_p ;
insert into pokémon_ values(214, 'Houndour', 'Snarl', 'Flamethrower', 1, 7, 1, 2);
insert into pokémon_ values(215, 'Houndoom', 'Bite', 'Crunch', 1, 1, 1, 2);
```

pokemon\_1

	pokedex_no	mon_name	move_one	move_two	Value
1	215	Houndoom	Bite	Crunch	215
2	214	Houndour	Snarl	Flamethrower	
3	214	Houndour	Snarl	Flamethrower	

Grid

Text

3 row(s) fetched - 5ms

MST en\_US Writable

Smart Insert

3 : 76 : 193

Caption

DBeaver 21.2.4 - <Sample1> Script-5

Database Navigator X Projects

Enter a part of object name here

- > Triggers
- > Indexes
- > Partitions
- pokemon\_type** 16K
  - > Columns
    - pokedex\_no (int)
    - mon\_name (varchar(50))
    - primary\_type (varchar(50))
    - secondary\_type (varchar(50))
  - > Constraints
  - > Foreign Keys
  - > References
  - > Triggers
  - > Indexes
  - > Partitions
  - > Views
  - > Indexes
  - > Procedures
  - > Triggers
  - > Events
- > Sample1

Project - General X

Name DataSource

Bookmarks

ER Diagrams

Scripts

\*<Sample1> Pokemon.sql \*<Sample1> Script-4 \*<Sample1> Script-5

```
select * from Pokemon.pokemon_p;
insert into pokemon_values(214, 'Houndour', 'Snarl', 'Flamethrower', 1, 7, 1, 2);
insert into pokemon_values(215, 'Houndoom', 'Bite', 'Crunch', 1, 1, 1, 2);
```

pokemon\_1 X

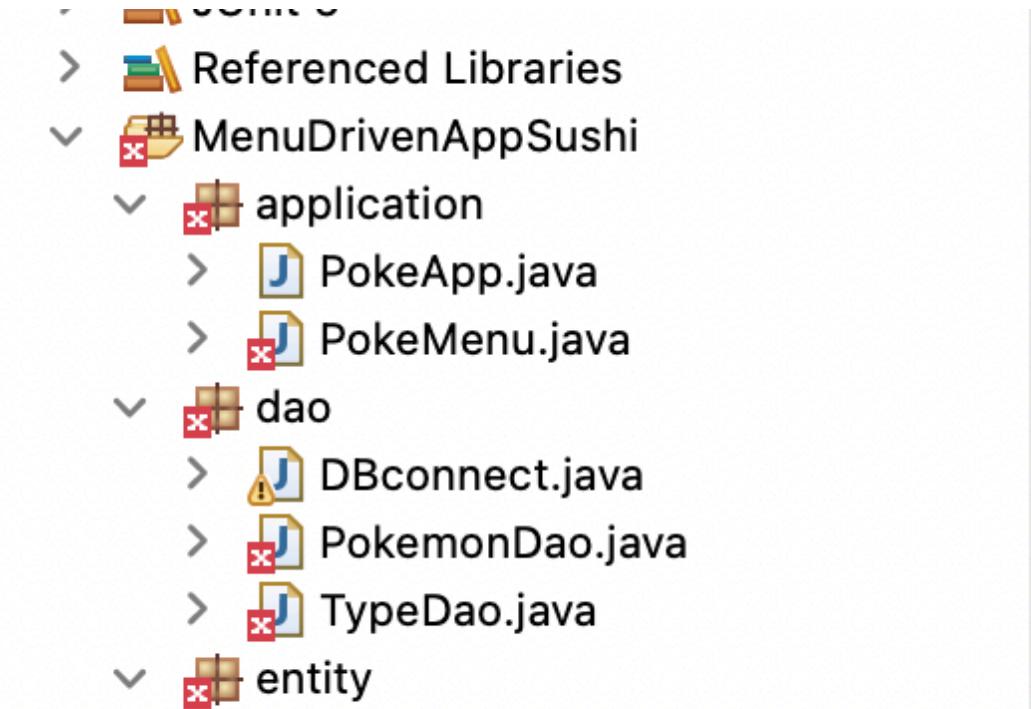
	pokedex_no	mon_name	move_one	move_two	Value
1	215	Houndoom	Bite	Crunch	215
2	214	Houndour	Snarl	Flamethrower	
3	214	Houndour	Snarl	Flamethrower	

Grid Text

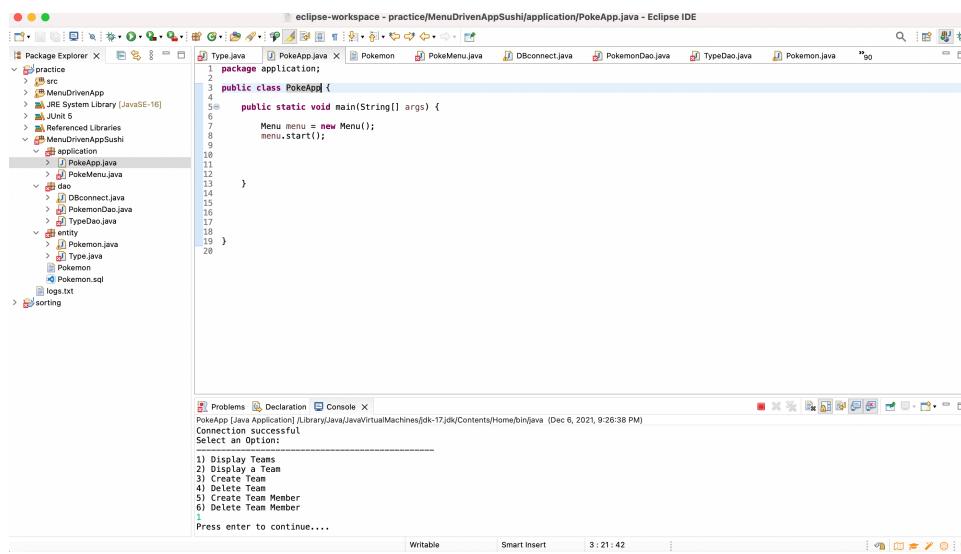
3 row(s) fetched - 5ms

MST en\_US Writable Smart Insert 3 : 76 : 193

Caption



Caption



Caption

eclipse-workspace - practice/MenuDrivenAppSushi/application/PokeMenu.java - Eclipse IDE

```

1 package application;
2
3 import java.util.Scanner;
4
5 public class PokeMenu {
6
7     private PokeDatabase typeDB = new PokeDB();
8     private PokeDatabase pokemonsDB = new PokeDB();
9
10    private String[] options = Arrays.asList("Display Teams", "Display a Team", "Create Team", "Delete Team", "Display Type", "Create Type", "Delete Type", "Display Pokemons", "Delete Pokemon");
11
12    public void start() {
13        Scanner scanner = new Scanner(System.in);
14        String selection = "";
15
16        do {
17            printMenu();
18            selection = scanner.nextLine();
19
20            try {
21                if (selection.equals("1")) {
22                    displayTeams();
23                } else if (selection.equals("2")) {
24                    displayTeam();
25                } else if (selection.equals("3")) {
26                    createTeam();
27                } else if (selection.equals("4")) {
28                    deleteTeam();
29                } else if (selection.equals("5")) {
30                    displayType();
31                } else if (selection.equals("6")) {
32                    createType();
33                } else if (selection.equals("7")) {
34                    deleteType();
35                } else if (selection.equals("8")) {
36                    displayPokemon();
37                } else if (selection.equals("9")) {
38                    deletePokemon();
39                }
40            } catch (SQLException e) {
41                e.printStackTrace();
42            }
43        } while (!selection.equals("10"));
44
45    }
46
47    private void printMenu() {
48        System.out.println("Select an Option:");
49        for (int i = 0; i < options.size(); i++) {
50            System.out.print(i + 1 + ") " + options.get(i));
51        }
52    }
53
54    private void displayTeams() throws SQLException {
55        List<Team> teams = typeDB.getTeams();
56        for (Team team : teams) {
57            System.out.println(team.getTeamId() + ": " + team.getName());
58        }
59    }
60
61    private void displayTeam() throws SQLException {
62        System.out.println("Enter Team ID: ");
63        int id = Integer.parseInt(scanner.nextLine());
64        Team team = typeDB.getTeam(id);
65        System.out.println(team);
66    }
67
68    private void createTeam() throws SQLException {
69        System.out.println("Enter Team Name: ");
70        String name = scanner.nextLine();
71        typeDB.createTeam(name);
72    }
73
74    private void deleteTeam() throws SQLException {
75        System.out.println("Enter Team ID: ");
76        int id = Integer.parseInt(scanner.nextLine());
77        typeDB.deleteTeam(id);
78    }
79
80    private void displayType() throws SQLException {
81        List<Type> types = typeDB.getTypes();
82        for (Type type : types) {
83            System.out.println(type.getTypeId() + ": " + type.getName());
84        }
85    }
86
87    private void createType() throws SQLException {
88        System.out.println("Enter Type Name: ");
89        String name = scanner.nextLine();
90        typeDB.createType(name);
91    }
92
93    private void deleteType() throws SQLException {
94        System.out.println("Enter Type ID: ");
95        int id = Integer.parseInt(scanner.nextLine());
96        typeDB.deleteType(id);
97    }
98
99    private void displayPokemon() throws SQLException {
100        List<Pokemon> pokemons = pokemonsDB.getPokemons();
101        for (Pokemon pokemon : pokemons) {
102            System.out.println(pokemon.getPokemonId() + ": " + pokemon.getName());
103        }
104    }
105
106    private void deletePokemon() throws SQLException {
107        System.out.println("Enter Pokemon ID: ");
108        int id = Integer.parseInt(scanner.nextLine());
109        pokemonsDB.deletePokemon(id);
110    }
111}

```

Problems Declaration Console X

PokeApp [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (Dec 6, 2021, 9:26:38 PM)

Connection successful

Select an Option:

1) Display Teams  
2) Display a Team  
3) Create Team  
4) Delete Team  
5) Create Team Member  
6) Delete Team Member  
7) Press enter to continue....

Caption



eclipse-workspace - practice/MenuDrivenAppSushi/dao/TypeDao.java - Eclipse IDE

```
package dao;
import java.sql.Connection;
public class TypeDao {
    private Connection connection;
    public void insertTeam(Connection connection) {
        String insertTeam = "INSERT INTO TEAM (TEAM_NAME, TEAM_TYPE_ID, TEAM_DESCRIPTION, TEAM_STATUS) VALUES (?, ?, ?, ?)";
        try {
            PreparedStatement ps = connection.prepareStatement(insertTeam);
            ps.setString(1, "Team 1");
            ps.setInt(2, 1);
            ps.setString(3, "A team for testing purposes");
            ps.setString(4, "Active");
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    public void insertMember(Connection connection) {
        String insertMember = "INSERT INTO TEAM_MEMBER (TEAM_ID, MEMBER_ID, MEMBER_NAME, MEMBER_ROLE) VALUES (?, ?, ?, ?)";
        try {
            PreparedStatement ps = connection.prepareStatement(insertMember);
            ps.setInt(1, 1);
            ps.setInt(2, 1);
            ps.setString(3, "John Doe");
            ps.setString(4, "Captain");
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    public void updateTeamStatus(Connection connection) {
        String updateTeamStatus = "UPDATE TEAM SET TEAM_STATUS = ? WHERE TEAM_ID = ?";
        try {
            PreparedStatement ps = connection.prepareStatement(updateTeamStatus);
            ps.setString(1, "Inactive");
            ps.setInt(2, 1);
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    public void deleteTeam(Connection connection) {
        String deleteTeam = "DELETE FROM TEAM WHERE TEAM_ID = ?";
        try {
            PreparedStatement ps = connection.prepareStatement(deleteTeam);
            ps.setInt(1, 1);
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    public void deleteMemberFromTeam(Connection connection) {
        String deleteMemberFromTeam = "DELETE FROM TEAM_MEMBER WHERE TEAM_ID = ? AND MEMBER_ID = ?";
        try {
            PreparedStatement ps = connection.prepareStatement(deleteMemberFromTeam);
            ps.setInt(1, 1);
            ps.setInt(2, 1);
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Package Explorer X | PokeApp.java | Pokemon.java | PokeMenu.java | DBconnect.java | PokemonDao.java | TypeDao.java | Pokemon.sql | logs.txt | sorting

Problems Declaration Console X

PokeApp [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (Dec 6, 2021, 9:26:38 PM)

Connection successful

Select an Option:

1) Display Teams  
2) Display Team  
3) Create Team  
4) Delete Team  
5) Create Team Member  
6) Delete Team Member

Press enter to continue...

Caption

eclipse-workspace - practice/MenuDrivenAppSushi/dao/PokemonDao.java - Eclipse IDE

```

1 package dao;
2
3 import java.sql.Connection;
4
5 public class PokemonDao {
6
7     private Connection connection;
8
9     private final String GET_MON_BY_TYPE = "SELECT * FROM members WHERE type_id = ?";
10    private final String CREATE_NEW_MON_QUERY = "INSERT INTO pokemons(pokedex_no, mon_name, move_one, move_two, attack_iv, defense_iv, hp_iv, gen_no) VALUES(?, ?, ?, ?, ?, ?, ?, ?)";
11    private final String DELETE_MON_BY_POKEDEX_NO_QUERY = "DELETE FROM pokemons WHERE pokedex_no = ?";
12
13    public PokemonDao() {
14        connection = DBconnection.getConnection();
15    }
16
17    public List<Member> getMonByPokedexNo(int pokedexNo) throws SQLException {
18        PreparedStatement ps = connection.prepareStatement(GET_MON_BY_TYPE);
19        ps.setInt(1, pokedexNo);
20        ResultSet rs = ps.executeQuery();
21        List<Member> result = new ArrayList<Member>();
22
23        while (rs.next()) {
24            Member mon = new Member();
25            mon.setId(rs.getInt("id"));
26            mon.setName(rs.getString("mon_name"));
27            mon.setMoveOne(rs.getString("move_one"));
28            mon.setMoveTwo(rs.getString("move_two"));
29            mon.setAttackIV(rs.getInt("attack_iv"));
30            mon.setDefenseIV(rs.getInt("defense_iv"));
31            mon.setHPIV(rs.getInt("hp_iv"));
32            mon.setGenNo(rs.getInt("gen_no"));
33
34            result.add(mon);
35        }
36
37        return result;
38    }
39
40    public void createNewPokemon(String monName, String moveOne, String moveTwo, int attack, int defense, int hp, int genNo) throws SQLException {
41        PreparedStatement ps = connection.prepareStatement(CREATE_NEW_MON_QUERY);
42        ps.setString(1, pokedexNo);
43        ps.setString(2, monName);
44        ps.setString(3, moveOne);
45        ps.setString(4, moveTwo);
46        ps.setInt(5, attack);
47        ps.setInt(6, defense);
48        ps.setInt(7, hp);
49        ps.setInt(8, genNo);
50
51        ps.executeUpdate();
52    }
53
54    public void deleteMonByPokedexNo(int pokedexNo) throws SQLException {
55        PreparedStatement ps = connection.prepareStatement(DELETE_MON_BY_POKEDEX_NO_QUERY);
56        ps.setInt(1, pokedexNo);
57        ps.executeUpdate();
58    }
59
60    public void deleteMemberByID(int id) throws SQLException {
61        PreparedStatement ps = connection.prepareStatement(DELETE_MON_BY_ID_QUERY);
62        ps.setInt(1, id);
63        ps.executeUpdate();
64    }

```

Problems Declaration Console X

PokeApp [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (Dec 6, 2021, 9:26:38 PM)

Connection successful

Select an Option:

- 1) Display Teams
- 2) Display a Team
- 3) Create Team
- 4) Delete Team
- 5) Create Team Member
- 6) Delete Team Member

Press enter to continue....

eclipse-workspace - practice/MenuDrivenAppSushi/entity/Type.java - Eclipse IDE

```

1 package entity;
2
3 import java.util.List;
4
5 public class Type {
6
7     private int pokedexNo;
8     private String monName;
9     private String primaryType;
10    private String secondaryType;
11
12    public Type(int pokedexNo, String monName, String primaryType, String secondaryType) {
13        this.setPokedex(pokedexNo);
14        this.setMonName(monName);
15        this.setPrimaryType(primaryType);
16        this.setSecondaryType(secondaryType);
17    }
18
19
20
21
22    private void setSecondaryType(String secondaryType2) {
23        // TODO Auto-generated method stub
24    }
25
26
27    private void setPrimaryType(String primaryType2) {
28        // TODO Auto-generated method stub
29    }
30
31
32    private void setMonName(String monName2) {
33        // TODO Auto-generated method stub
34    }
35
36
37    private void setPokedex(int pokedexNo2) {
38        // TODO Auto-generated method stub
39    }
40
41
42
43
44}

```

Problems Declaration Console X

PokeApp [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (Dec 6, 2021, 9:26:38 PM)

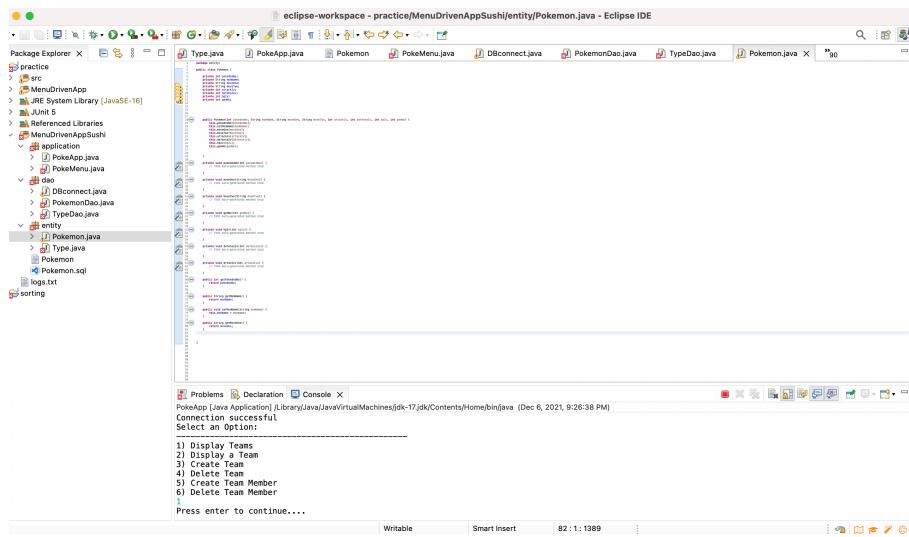
Connection successful

Select an Option:

- 1) Display Teams
- 2) Display a Team
- 3) Create Team
- 4) Delete Team
- 5) Create Team Member
- 6) Delete Team Member

Press enter to continue....

Caption



Caption

**URL to GitHub Repository:**