

# Relational Databases with MySQL Week 11 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:** Complete the coding steps. 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 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 of Code:

```
1 package sorting;
2
3 import java.util.List;
4
5 public class MySort {
6
7     private SortService sortService = new SortService();
8
9     public static void main(String[] args) {
10
11         // Calling sort method
12         System.out.println("Sorted Objects using Lambda expression: ");
13         new MySort().sortLambda();
14
15         System.out.println();
16
17         System.out.println("Sorted Objects using Method Reference: ");
18         new MySort().methodReference();
19
20     }
21
22     // Method that uses our 'compare' method and lambda to sort our
23     // list of object
24     private void sortLambda() {
25         List<Animal> animal = sortService.getAnimal(SortType.LAMBDA);
26         print(animal, SortType.LAMBDA);
27     }
28
29     // Method Reference
30     private void methodReference() {
31         List<Animal> animal = sortService.getAnimal(SortType.METHOD_REFERENCE);
32         print(animal, SortType.METHOD_REFERENCE);
33     }
34
35     private void print(List<Animal> animal, SortType type) {
36         switch(type) {
37             case LAMBDA:
38                 animal.forEach(animal -> System.out.println(animal.getAnimal()));
39                 break;
40             case METHOD_REFERENCE:
41                 animal.forEach(System.out::println);
42                 break;
43             default:
44                 break;
45         }
46     }
47
48 }
```

```
1 package service;
2
3 import java.util.Comparator;
4
5 public class SortService {
6
7     private SortDao sortDao = new SortDao();
8
9     public List<Animal> getAnimal(SortType type) {
10         List<Animal> animal = sortDao.getAnimals();
11         Comparator<Animal> comp = null;
12
13         switch(type) {
14             case LAMBDA:
15                 comp = new MySort();
16                 comp = (animal a1, Animal a2) -> {
17                     return Animal.compare(a1, a2);
18                 };
19             case METHOD_REFERENCE:
20                 comp = Animal::compare;
21                 break;
22             default:
23                 throw new RuntimeException("Unhandled sort type: " + type);
24         }
25
26         animal.sort(comp);
27         return animal;
28     }
29
30     static class MySort implements Comparator<Animal> {
31
32         @Override
33         public int compare(Animal a1, Animal a2) {
34             return Animal.compare(a1, a2);
35         }
36     }
37
38 }
```

```
1 package sort.model;
2
3 public class Animal {
4
5     private String animalName;
6
7     public Animal(String animalName) {
8         this.animalName = animalName;
9     }
10
11     @Override
12     public String toString() {
13         return animalName + " Animal";
14     }
15
16     public String getAnimal() {
17         return animalName + " Animal";
18     }
19
20
21     public static int compare(Animal a1, Animal a2) {
22         if(a1.animalName.compareTo(a2.animalName) < a2.animalName.compareTo(a1.animalName)) {
23             return -1;
24         } else if (a1.animalName.compareTo(a2.animalName) > a2.animalName.compareTo(a1.animalName)) {
25             return 1;
26         } else {
27             return 0;
28         }
29     }
30
31 }
32
33
```

```
1 package dao;
2
3 import java.util.ArrayList;
4
5 public class SortDao {
6
7     // Static list of objects
8     public List<Animal> animals = new ArrayList<>(List.of(
9         new Animal("Dog"),
10        new Animal("Cat"),
11        new Animal("Tig"),
12        new Animal("Horse"),
13        new Animal("lion"),
14        new Animal("Eagle"));
15
16     public List<Animal> getAnimals() {
17         return animals;
18     }
19
20 }
21
22
23
24
25
```

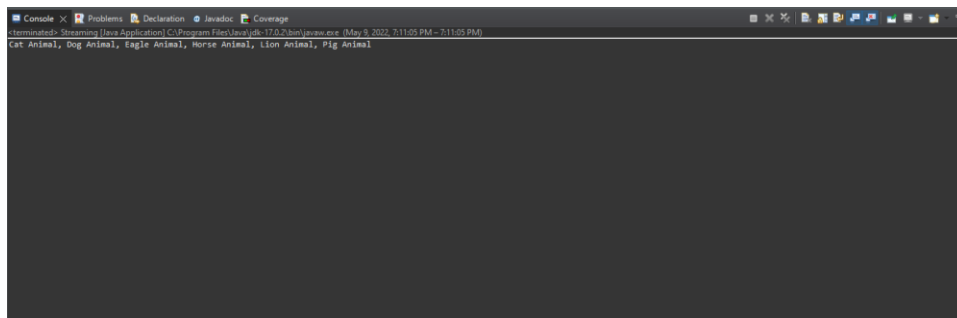
```
Streaming.java  MySort.java  SortService.java  SortDao.java  Optional.java  Animal.java
1 package optional;
2
3 import java.util.NoSuchElementException;
4
5 public class Optional {
6
7     public static Animal a(Optional<Animal> animal) {
8         return animal.orElseThrow(() -> new NoSuchElementException("It appears that the object does not exist"));
9     }
10
11     public void a(Optional<Animal> animal) {
12         animal = Optional.empty();
13
14         try {
15             Optionalis.a(animal);
16             System.out.println(animal);
17         } catch (NoSuchElementException e) {
18             System.out.println(e.getMessage());
19         }
20     }
21
22     public static void main(String[] args) {
23         Optional<Animal> animal = Optional.empty();
24         Optional option = new Optional();
25         option.a(animal);
26     }
27 }
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
```

```
Streaming.java  MySort.java  SortService.java  SortDao.java  Optional.java  Animal.java
1 package stream;
2
3 import java.util.List;
4
5 public class Streaming {
6
7     public static void main(String[] args) {
8         SortDao sortDao = new SortDao();
9
10         // Create the stream of animals
11         List<Animal> animal = sortDao.getAnimals();
12
13         // Turn stream of object to a stream of String
14         Stream<String> stream = animal.stream().map(Object::toString);
15
16         // Sort the stream in the natural order
17         // collect stream and returns a comma separated list of names as single string.
18         // and prints it
19         String s = stream.sorted().map(Object::toString).collect(Collectors.joining(", "));
20         System.out.println(s);
21     }
22 }
23
24
```

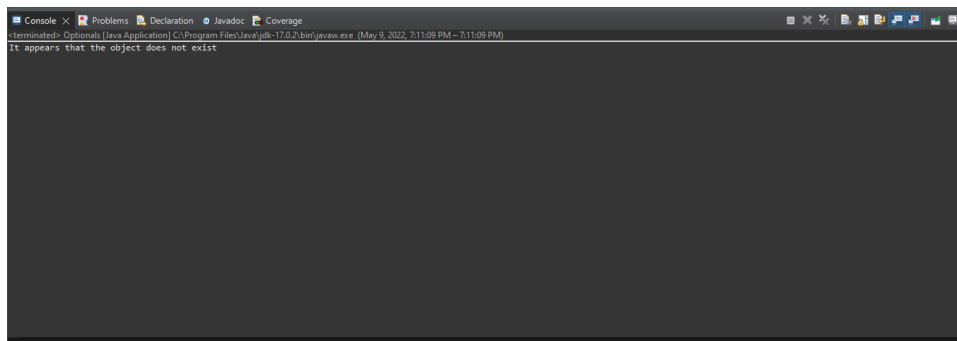
## Screenshots of Running Application Results:

```
Console  Problems  Declaration  Javadoc  Coverage
C:\Program Files\Java\jdk-17.0.2\bin\java.exe (May 9, 2022, 7:10:56 PM - 7:10:56 PM)
Sorted Objects using Lambda expression:
Cat Animal
Dog Animal
Eagle Animal
Horse Animal
Lion Animal
Pig Animal

Sorted Objects using Method Reference:
Cat Animal
Dog Animal
Eagle Animal
Horse Animal
Lion Animal
Pig Animal
```



A screenshot of an IDE's console window. The title bar shows tabs for Console, Problems, Declaration, Javadoc, and Coverage. The console text reads: "terminated> Streaming [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\java.exe (May 9, 2022, 7:11:05 PM)" followed by "Cat Animal, Dog Animal, Eagle Animal, Horse Animal, Lion Animal, Pig Animal".



A screenshot of an IDE's console window. The title bar shows tabs for Console, Problems, Declaration, Javadoc, and Coverage. The console text reads: "terminated> Optional [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\java.exe (May 9, 2022, 7:11:09 PM)" followed by the error message "It appears that the object does not exist".

**URL to GitHub Repository:**

**<https://github.com/ArturoAquino/Week11Assignment>**