Thomas Le

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:

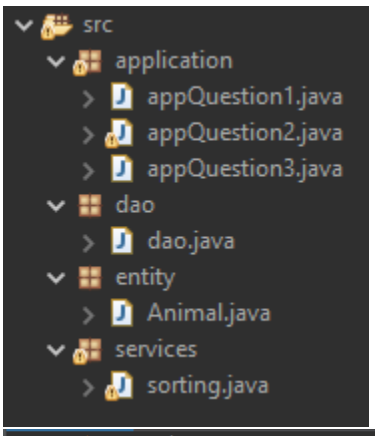
1. Create a class of whatever type you want (Animal, Person, Camera, Cheese, etc.).
    a. Do not implement the Comparable interface.
    b. Add a name instance variable so that you can tell the objects apart.
    c. Add getters, setters and/or a constructor as appropriate.
    d. Add a toString method 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:




```java
package entity;

public class Animal {

    private String name;
    private String sound;
    private int numLegs;

    public Animal(String name, String sound, int numLegs) {
        this.setName(name);
        this.setSound(sound);
        this.setNumLegs(numLegs);
    }

    public String getName() {
        return name;
    }

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

    public String getSound() {
        return sound;
    }

    public void setSound(String sound) {
        this.sound = sound;
    }

    public int getNumLegs() {
        return numLegs;
    }

    public void setNumLegs(int numLegs) {
        this.numLegs = numLegs;
    }

    @Override                                //overrides the superclass that's already present in the Object methods
    public String toString() {
        return "This is the " + name + " animal";
    }

}
```

```java
package dao;

import java.util.ArrayList;
import java.util.List;
import java.util.NoSuchElementException;
import java.util.Optional;

import entity.Animal;

public class dao {

    public List<Animal> animalList = new ArrayList <Animal> (List.of(
            new Animal ("Dog", "Woof", 4),
            new Animal ("Cat", "Meow", 4),
            new Animal ("Kangaroo", "Yip", 2),
            new Animal ("Snake", "Hiss", 0)));

    public static int compareName(Animal a, Animal b) {
        return a.getName().compareTo(b.getName());
    }

    public Animal animalExists(Optional<Animal> optionalAnimal) { //First method takes in an Optional of Animal called optionalAnimal then returns the Animal object if it exists
        if (optionalAnimal.isPresent()) {
            return optionalAnimal.get();
        } else {
            throw new NoSuchElementException("There is not an object in the optionalAnimal Optional"); //throws a NoSuchElementException if the Animal object does not exist.
        }
    }
}
```

```java
package services;

import java.util.List;

public class sorting {

    public dao d = new dao();

    public void sortWithLambda(List<Animal> animalList) {
        animalList.sort((Animal a, Animal b) -> d.compareName(a, b));
        animalList.forEach(System.out::println);
    }
}
```

```java
package application;

import dao.dao;

public class appQuestion1 { //Sorts the Animalist using the Lambda expression from the sorting class then prints the said list.

    public static void main(String args[]) {

        dao d = new dao();
        sorting s = new sorting();

        s.sortWithLambda(d.animalList);
    }

}
```

```java
package application;

import java.util.List;

public class appQuestion2 { //Creates a stream from the list of objects then maps each Animal toString, sorted in the natural order, and joins everything together into one String

    public static void main(String args[]) {

        dao d = new dao();

        String sortedAnimalList = d.animalList.stream()
                                              .map(Animal :: toString)
                                              .sorted()
                                              .collect(Collectors.joining(","));

        System.out.println(sortedAnimalList);
    }

}
```

```java
package application;

import java.util.NoSuchElementException;

public class appQuestion3 { //Methods using Optionals

    private static dao d = new dao();

    public static void main(String[] args) { //Optional nullTest to see the error statement if optional is null. Optional animalTest to see the output for an optional that exists.
        Animal test = new Animal("Oyster" , "Yap" , 1);
        Optional<Animal> nullTest = Optional.ofNullable(null);
        Optional<Animal> animalTest = Optional.ofNullable(test);
        run(animalTest);
        run(nullTest);
    }

    public static void run(Optional<Animal> optionalAnimal) { //Second method  calls method a and also calls method a with an empty optional which will throw an error if needed.
        try {
        optionalAnimal.orElseThrow(() -> new NoSuchElementException(
                "The object " + optionalAnimal + " is missing!"));
        System.out.println(d.animalExists(optionalAnimal));
        } catch (NoSuchElementException e) {
            System.out.println(e.getMessage());
        }
    }

}
```

Screenshots of Running Application Results:

Question 1:

```
This is the Cat animal
This is the Dog animal
This is the Kangaroo animal
This is the Snake animal
```

Question 2:

```
This is the Cat animal,This is the Dog animal,This is the Kangaroo animal,This is the Snake animal
```

Question 3:

```
This is the Oyster animal
The object Optional.empty is missing!
```

URL to GitHub Repository:

https://github.com/AkemiTCGyt/PromineoTechWeek11CodingAssignment