Relational Databases with MySQL Week 4 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: 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.

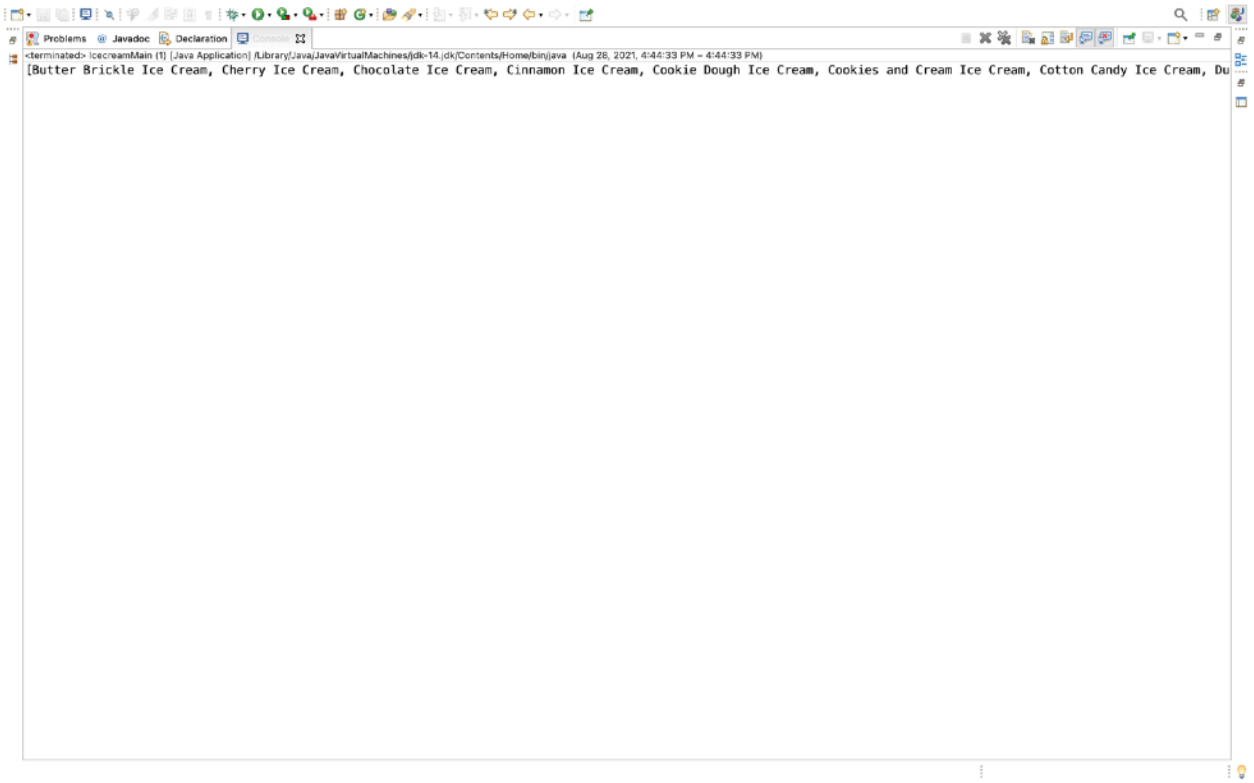
Screenshots:

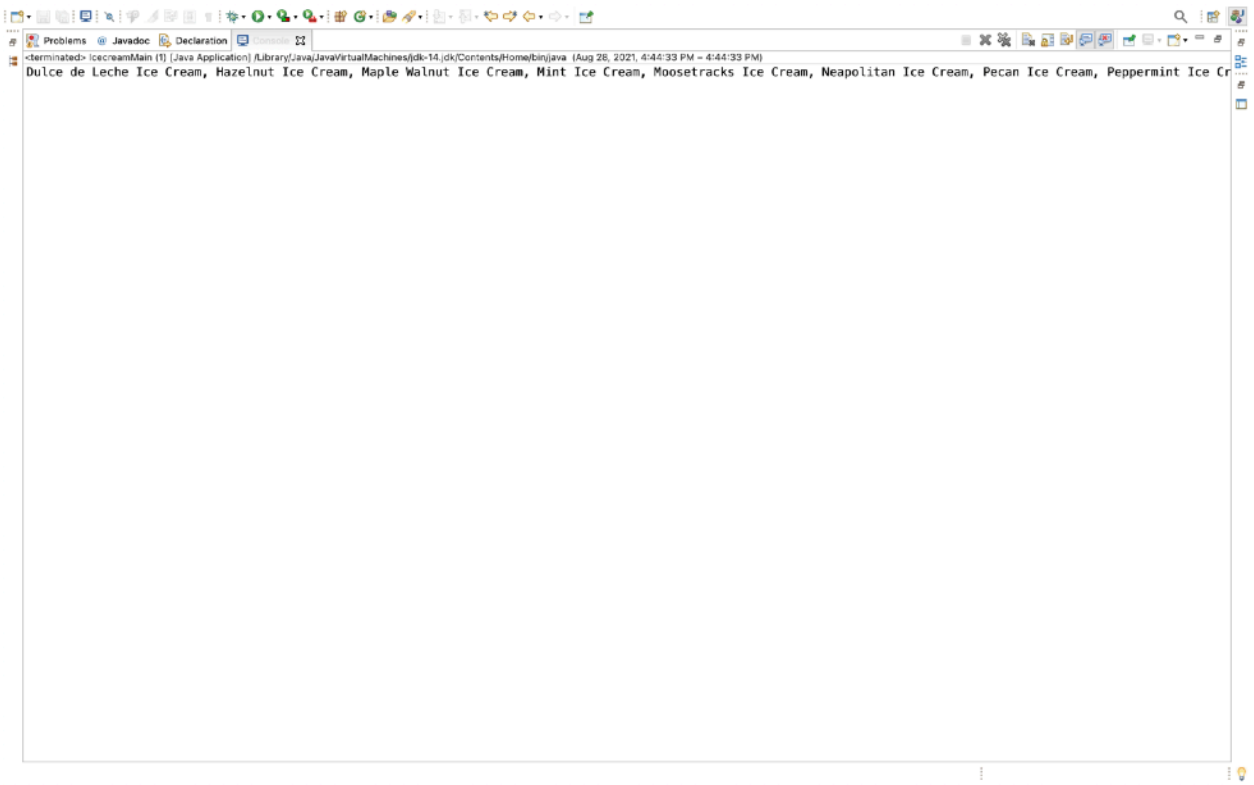
```
1  package object.icecream.sort;
2
3  public class Icecream {
4
5      public static Object compareIcecream;
6      String icecreamType;
7
8      public Icecream(String str) {
9          this.icecreamType = str;
10     }
11
12     public String getIcecreamType() {
13         return icecreamType;
14     }
15
16     @Override
17     public String toString() {
18         return (this.getIcecreamType() + " Ice Cream");
19     }
20
21     public static int compare(Icecream ice1, Icecream ice2) {
22         return ice1.icecreamType.compareTo(ice2.icecreamType);
23     }
24
25 }
26
```

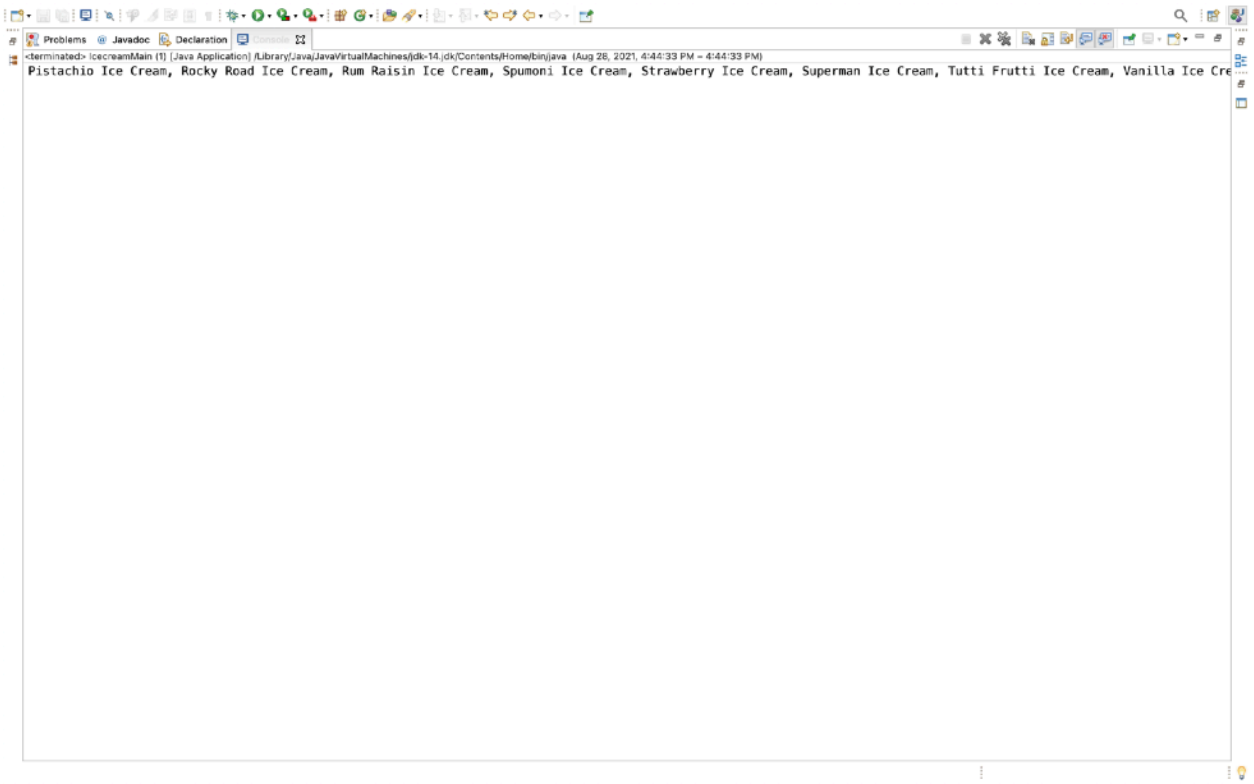
```java
package object.icecream.dao;

import java.util.ArrayList;
import java.util.List;
import object.icecream.sort.Icecream;

public class IcecreamData {

    List<Icecream> icecreams = new ArrayList<Icecream>(List.of(new Icecream("Vanilla"), new Icecream("Chocolate"),
            new Icecream("Mint"), new Icecream("Rocky Road"), new Icecream("Strawberry"), new Icecream("Pecan"),
            new Icecream("Neapolitan"), new Icecream("Hazelnut"), new Icecream("Rum Raisin"), new Icecream("Butter Brickle"),
            new Icecream("Cookie Dough"), new Icecream("Cookies and Cream"), new Icecream("Cotton Candy"), new Icecream("Cherry"),
            new Icecream("Dulce de Leche"), new Icecream("Peppermint"), new Icecream("Cinnamon"), new Icecream("Maple Walnut"),
            new Icecream("Pistachio"), new Icecream("Moosetracks"), new Icecream("Tutti Frutti"), new Icecream("Superman"),
            new Icecream("Spumoni")));

    public List<Icecream> getIcecreams() {
        return icecreams;
    }

}
```

```java
package object.icecream.sort;

import java.util.List;
import object.icecream.dao.IcecreamData;

public class IcecreamSort {

    static IcecreamData icecreamData = new IcecreamData();

    public static List<Icecream> sortIcecream() {

        List<Icecream> icecreamList = icecreamData.getIcecreams();

        icecreamList.sort(Icecream::compare);

        return icecreamList;
    }

}
```

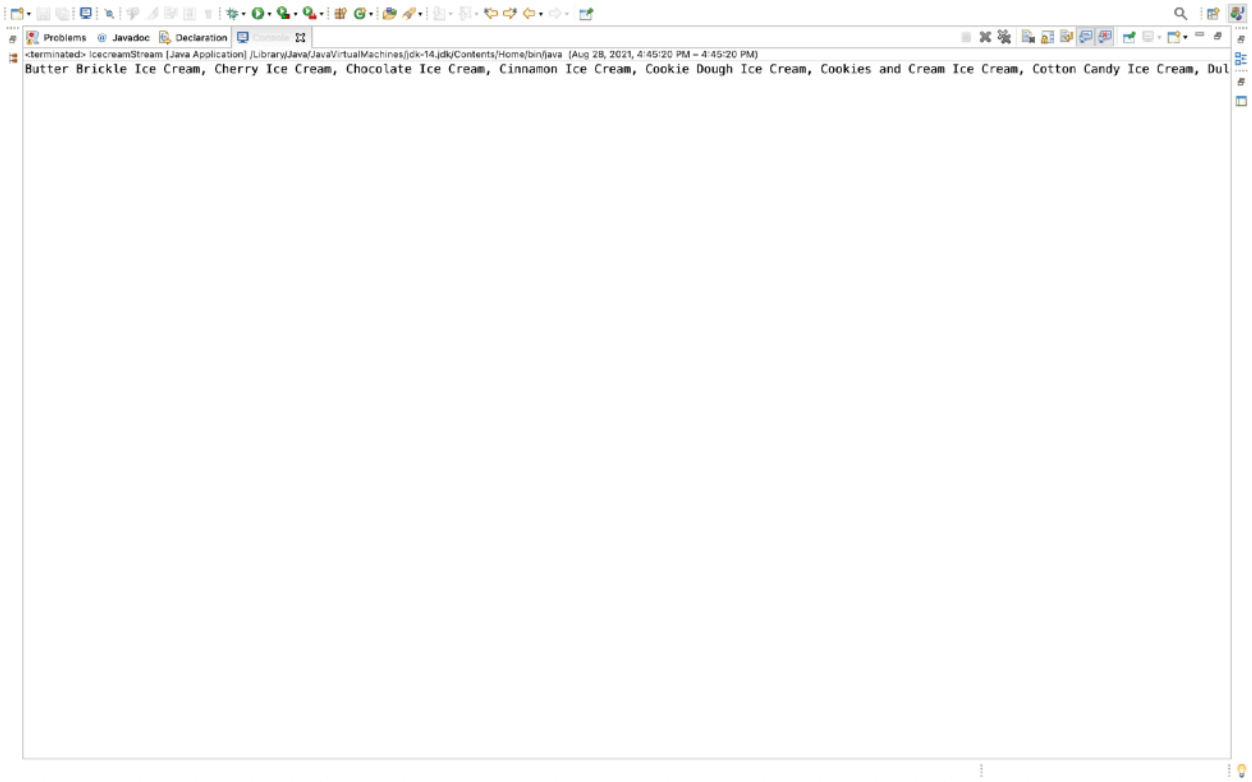
```java
package object.icecream.sort;

import java.util.List;

public class IcecreamMain {

    public static void main(String[] args) {

        List<Icecream> sortedIcecream = IcecreamSort.sortIcecream();

        System.out.println(sortedIcecream);

    }

}
```

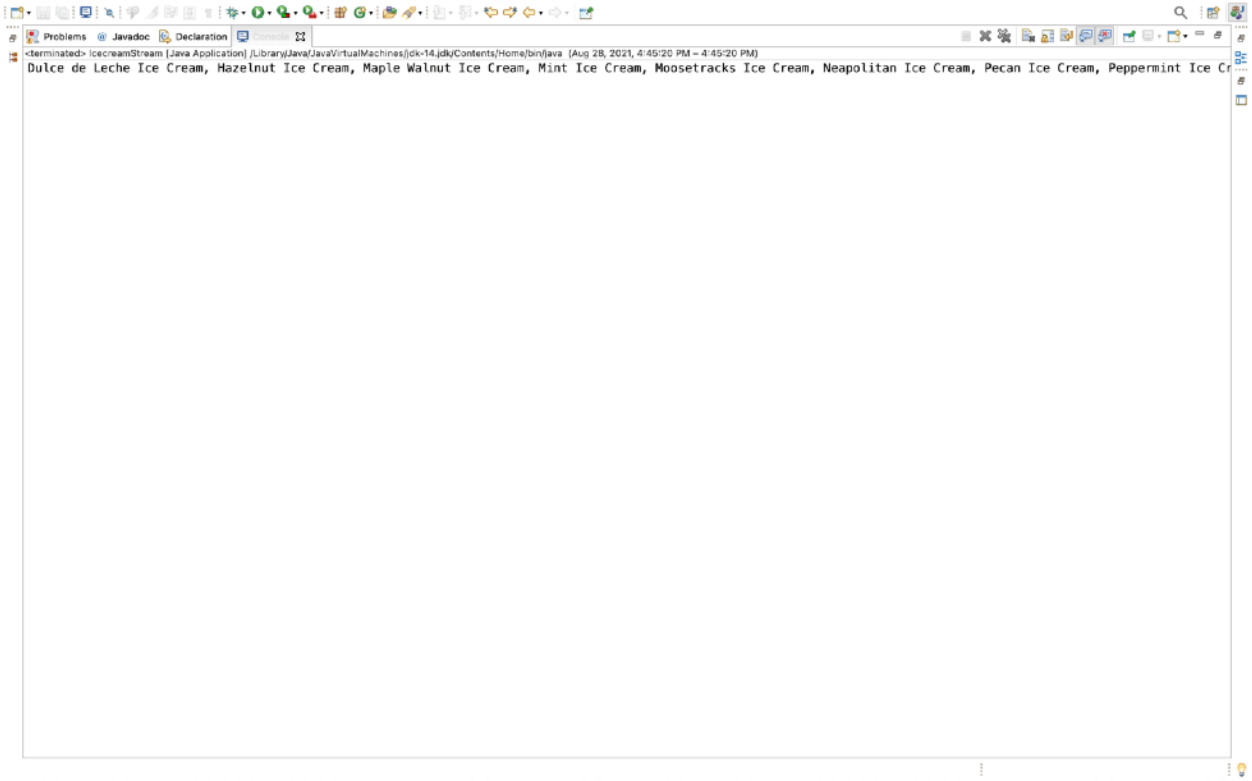

Problems
Javadoc
Declaration
Console
<terminated> IcecreamMain (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.jdk/Contents/Home/bin/java (Aug 28, 2021, 4:44:33 PM – 4:44:33 PM)
[Butter Brickle Ice Cream, Cherry Ice Cream, Chocolate Ice Cream, Cinnamon Ice Cream, Cookie Dough Ice Cream, Cookies and Cream Ice Cream, Cotton Candy Ice Cream, Du

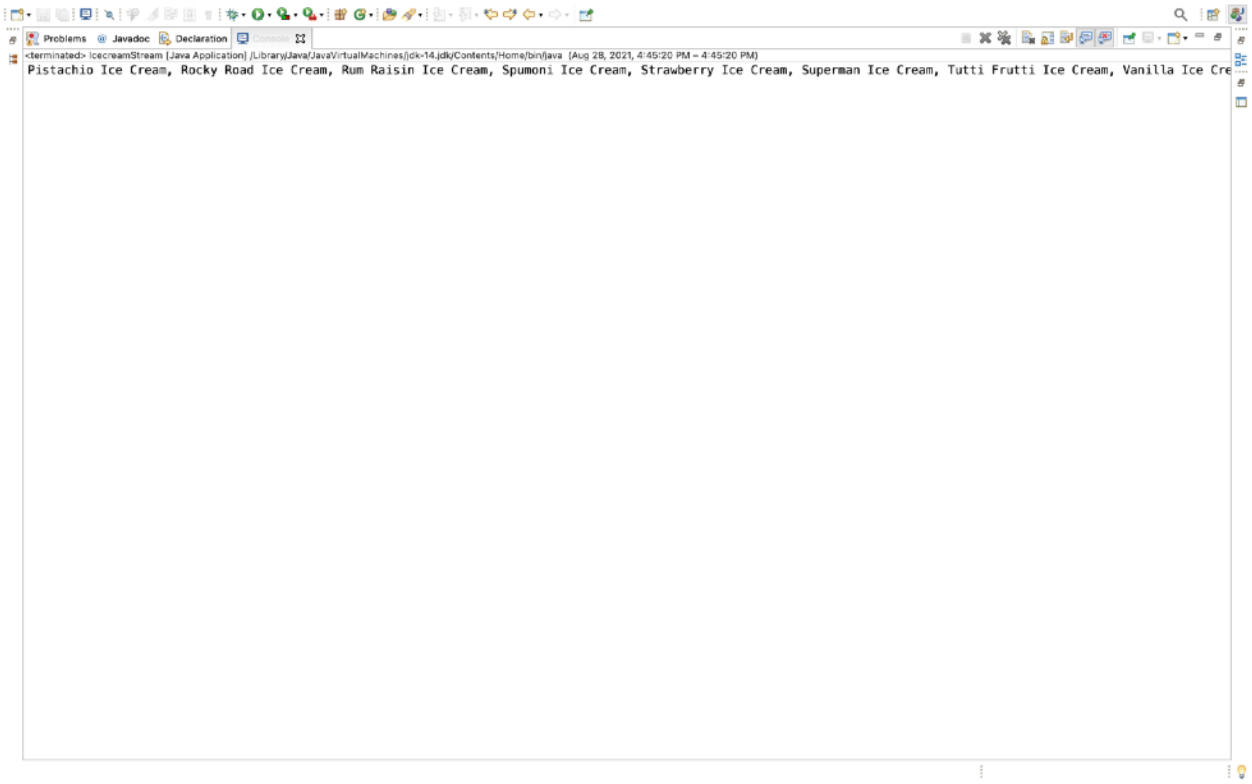

Problems
Javadoc
Declaration
Console
<terminated> IcecreamMain (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.jdk/Contents/Home/bin/java (Aug 28, 2021, 4:44:33 PM – 4:44:33 PM)
Dulce de Leche Ice Cream, Hazelnut Ice Cream, Maple Walnut Ice Cream, Mint Ice Cream, Moosetracks Ice Cream, Neapolitan Ice Cream, Pecan Ice Cream, Peppermint Ice Cr



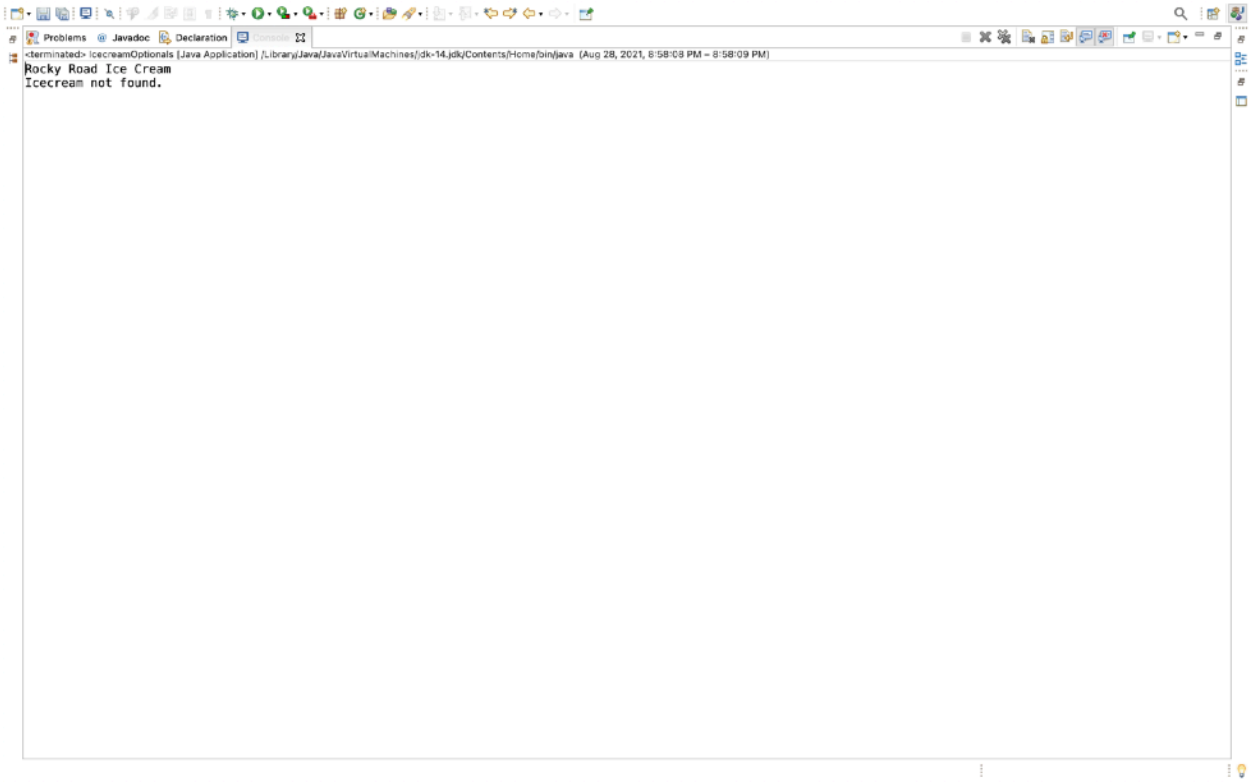

```java
package object.icecream.stream;

import java.util.List;
import object.icecream.sort.Icecream;
import object.icecream.dao.IcecreamData;
import java.util.stream.Collectors;

public class IcecreamStream {

    public static void main(String[] args) {

        IcecreamData icecreamData1 = new IcecreamData();

        List<Icecream> icecreams1 = icecreamData1.getIcecreams();


        String icecreamStr = icecreams1.stream()
              .map(String::valueOf)
              .sorted()
              .collect(Collectors.joining(", "));


        System.out.println(icecreamStr);




    }

}
```


Problems
Javadoc
Declaration
<terminated> IcecreamStream [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.jdk/Contents/Home/bin/java (Aug 28, 2021, 4:45:20 PM – 4:45:20 PM)
Butter Brickle Ice Cream, Cherry Ice Cream, Chocolate Ice Cream, Cinnamon Ice Cream, Cookie Dough Ice Cream, Cookies and Cream Ice Cream, Cotton Candy Ice Cream, Dul


Problems
Javadoc
Declaration
<terminated> IcecreamStream [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.jdk/Contents/Home/bin/java (Aug 28, 2021, 4:45:20 PM – 4:45:20 PM)
Dulce de Leche Ice Cream, Hazelnut Ice Cream, Maple Walnut Ice Cream, Mint Ice Cream, Moosetracks Ice Cream, Neapolitan Ice Cream, Pecan Ice Cream, Peppermint Ice Cr




```java
package object.icecream.optionals;

import java.util.NoSuchElementException;
import java.util.Optional;
import object.icecream.dao.IcecreamData;
import object.icecream.sort.Icecream;


public class IcecreamOptionals {


public static void main(String[] args) {

    icecreamB();

}

    public static Icecream icecreamA(Optional<Icecream> icecream) {

        return icecream.orElseThrow(() ->
        new NoSuchElementException("Icecream not found."));

}

    public static void icecreamB() {

        Optional<Icecream> optional = Optional.of(IcecreamData.icecreams.get(3));

            System.out.println(icecreamA(optional));

            try {
                icecreamA(Optional.empty());
            }
            catch (Exception e) {
                System.out.println(e.getMessage());
            }
}
```



URL to GitHub Repository:

https://github.com/johnbarts/mySQLweek4