# 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:**

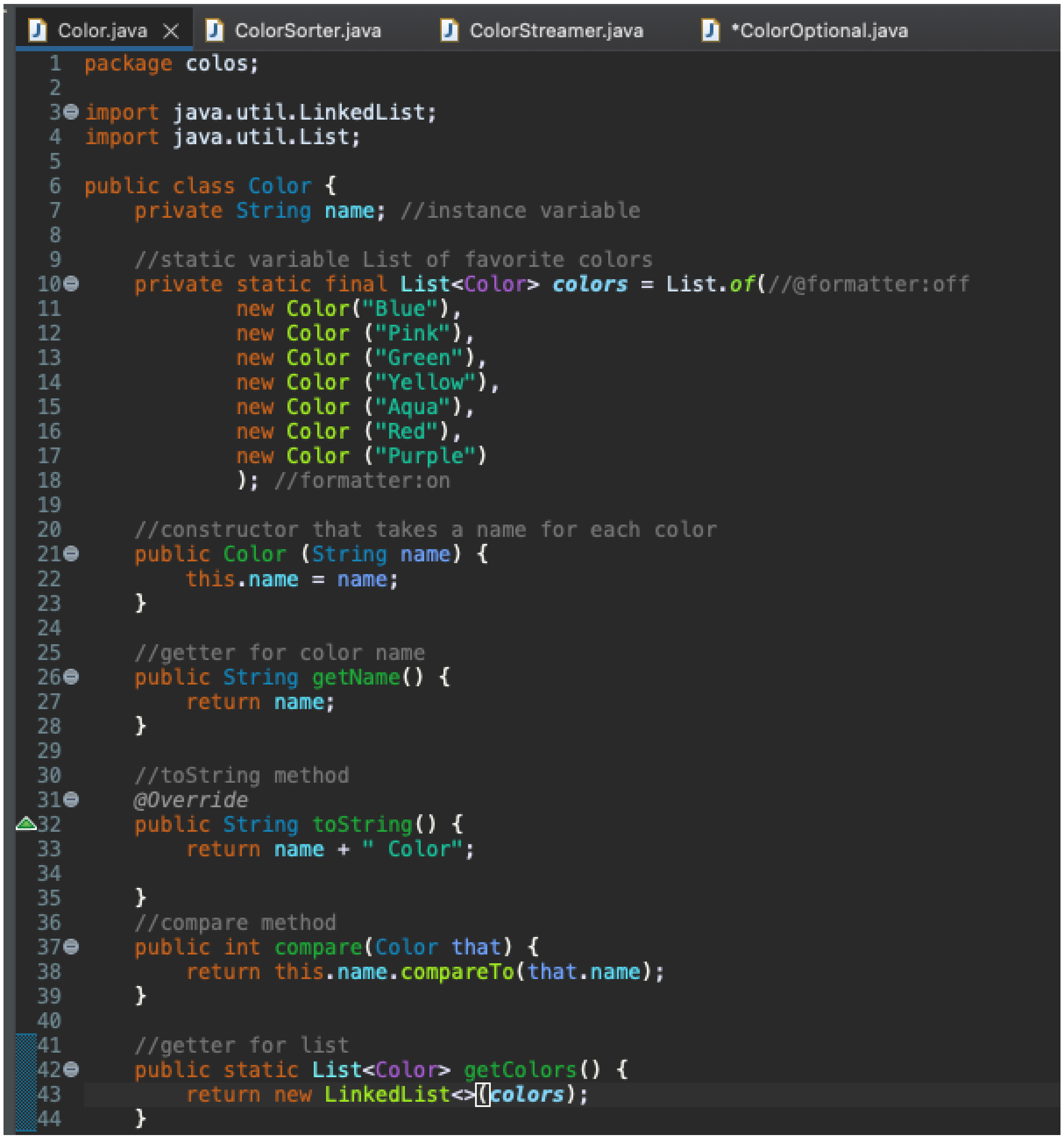
1. ~~Create a class of whatever type you want (Animal, Person, Camera, Cheese, etc.).~~
    a. ~~Do not implement the Comparable interface.~~
    b. ~~Add a name instance variable so that you can tell the objects apart.~~
    c. ~~Add getters, setters and/or a constructor as appropriate.~~
    d. ~~Add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

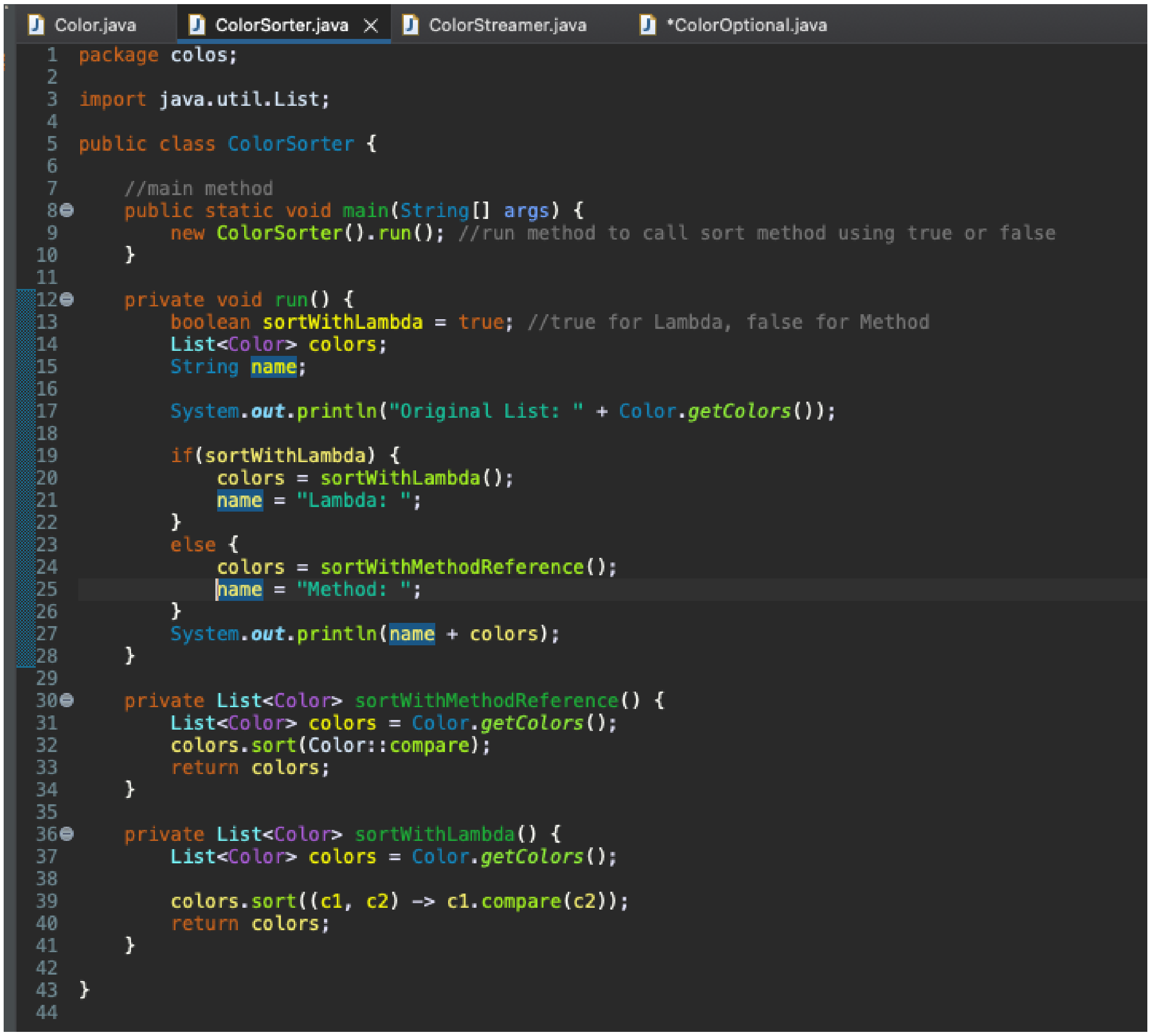
    b. ~~The method should throw a NoSuchElementException with a custom message if the object is not present.~~
    c. ~~Create another method (method b) that calls method a with an object wrapped by an Optional. Show that the object is returned unwrapped from the Optional (i.e., print the object).~~
    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:**

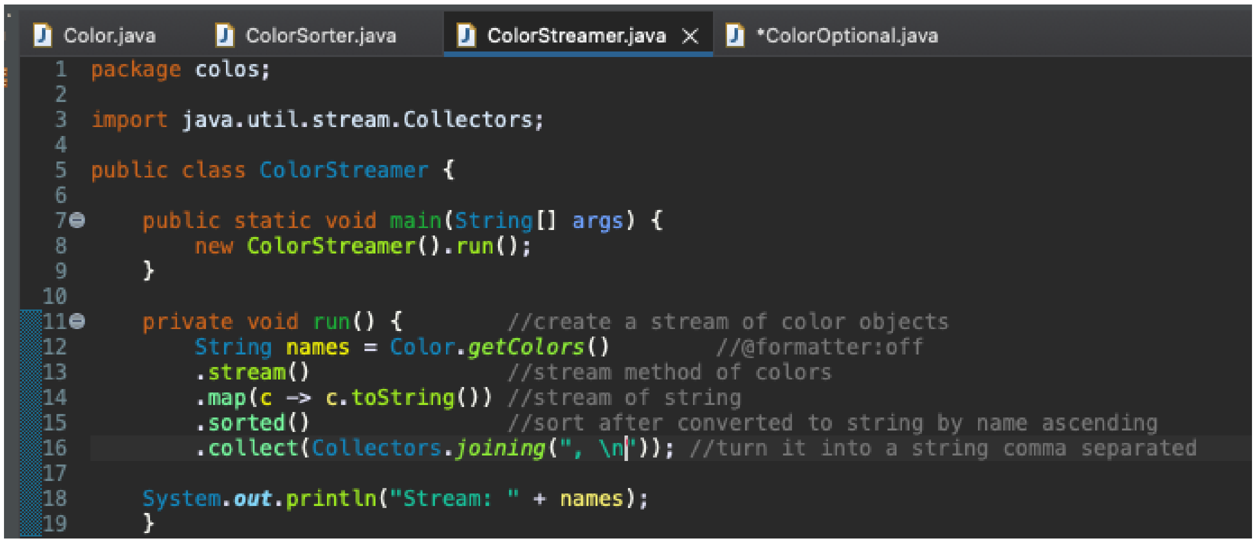


Color.java ColorSorter.java ColorStreamer.java *ColorOptional.java

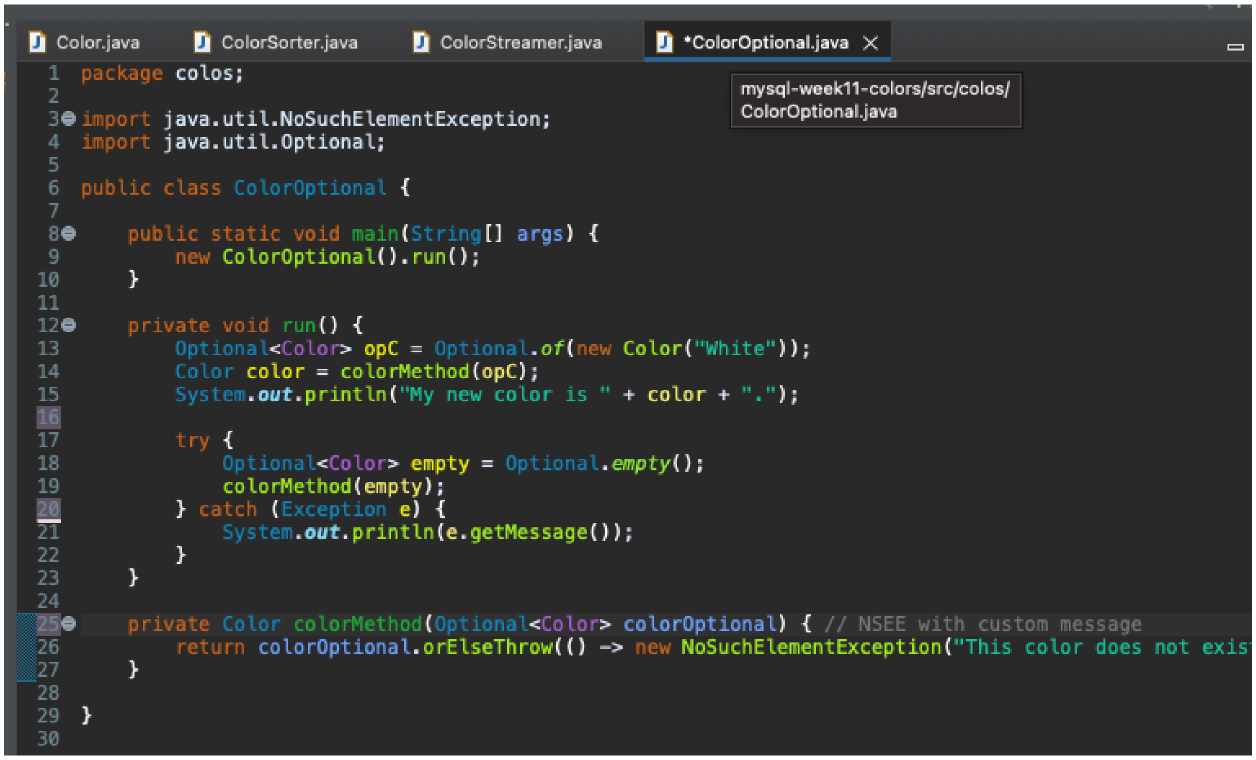
```java
package colos;

import java.util.LinkedList;
import java.util.List;

public class Color {
    private String name; //instance variable

    //static variable List of favorite colors
    private static final List<Color> colors = List.of(//@formatter:off
            new Color("Blue"),
            new Color ("Pink"),
            new Color ("Green"),
            new Color ("Yellow"),
            new Color ("Aqua"),
            new Color ("Red"),
            new Color ("Purple")
            ); //formatter:on

    //constructor that takes a name for each color
    public Color (String name) {
        this.name = name;
    }

    //getter for color name
    public String getName() {
        return name;
    }

    //toString method
    @Override
    public String toString() {
        return name + " Color";

    }
    //compare method
    public int compare(Color that) {
        return this.name.compareTo(that.name);
    }

    //getter for list
    public static List<Color> getColors() {
        return new LinkedList<>(colors);
    }
```


Color.java | ColorSorter.java | ColorStreamer.java | *ColorOptional.java

```java
package colos;

import java.util.List;

public class ColorSorter {

    //main method
    public static void main(String[] args) {
        new ColorSorter().run(); //run method to call sort method using true or false
    }

    private void run() {
        boolean sortWithLambda = true; //true for Lambda, false for Method
        List<Color> colors;
        String name;

        System.out.println("Original List: " + Color.getColors());

        if(sortWithLambda) {
            colors = sortWithLambda();
            name = "Lambda: ";
        }
        else {
            colors = sortWithMethodReference();
            name = "Method: ";
        }
        System.out.println(name + colors);
    }

    private List<Color> sortWithMethodReference() {
        List<Color> colors = Color.getColors();
        colors.sort(Color::compare);
        return colors;
    }

    private List<Color> sortWithLambda() {
        List<Color> colors = Color.getColors();

        colors.sort((c1, c2) -> c1.compare(c2));
        return colors;
    }

}
```

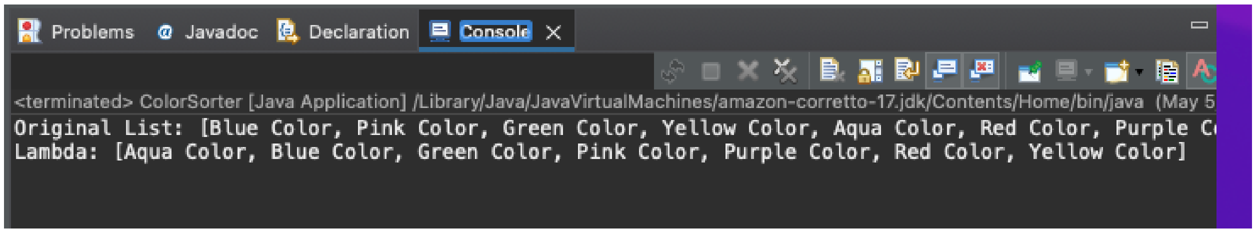


```java
package colos;

import java.util.stream.Collectors;

public class ColorStreamer {

    public static void main(String[] args) {
        new ColorStreamer().run();
    }

    private void run() {        //create a stream of color objects
        String names = Color.getColors()        //@formatter:off
        .stream()               //stream method of colors
        .map(c -> c.toString()) //stream of string
        .sorted()               //sort after converted to string by name ascending
        .collect(Collectors.joining(", \n")); //turn it into a string comma separated

    System.out.println("Stream: " + names);
    }
```

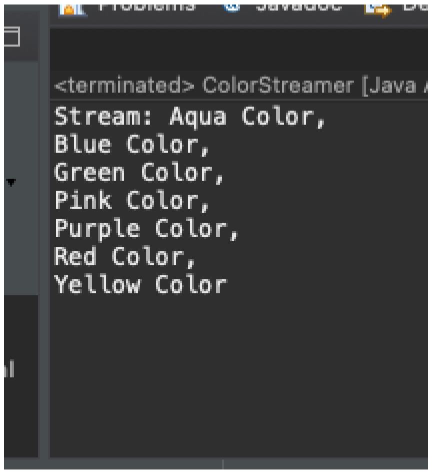


```java
package colos;

import java.util.NoSuchElementException;
import java.util.Optional;

public class ColorOptional {

    public static void main(String[] args) {
        new ColorOptional().run();
    }

    private void run() {
        Optional<Color> opC = Optional.of(new Color("White"));
        Color color = colorMethod(opC);
        System.out.println("My new color is " + color + ".");

        try {
            Optional<Color> empty = Optional.empty();
            colorMethod(empty);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    private Color colorMethod(Optional<Color> colorOptional) { // NSEE with custom message
        return colorOptional.orElseThrow(() -> new NoSuchElementException("This color does not exis
    }

}
```

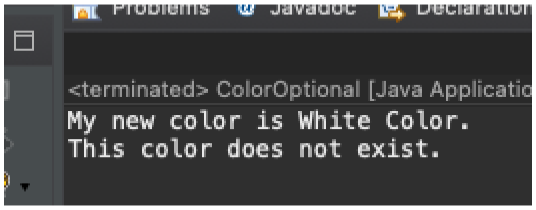
## Screenshots of Running Application Results:



```
<terminated> ColorSorter [Java Application] /Library/Java/JavaVirtualMachines/amazon-corretto-17.jdk/Contents/Home/bin/java  (May 5
Original List: [Blue Color, Pink Color, Green Color, Yellow Color, Aqua Color, Red Color, Purple Co
Lambda: [Aqua Color, Blue Color, Green Color, Pink Color, Purple Color, Red Color, Yellow Color]
```



```
<terminated> ColorStreamer [Java A
Stream: Aqua Color,
Blue Color,
Green Color,
Pink Color,
Purple Color,
Red Color,
Yellow Color
```



```
<terminated> ColorOptional [Java Applicatio
My new color is White Color.
This color does not exist.
```

**URL to GitHub Repository:**

https://github.com/cperrine19/Week11_MySQL_Colors/tree/main