Gebze Technical University
Computer Engineering

CSE 222 - 2019 Spring

HOMEWORK 5 REPORT

STUDENT NAME
Can BEYAZNAR
STUDENT NUMBER
161044038

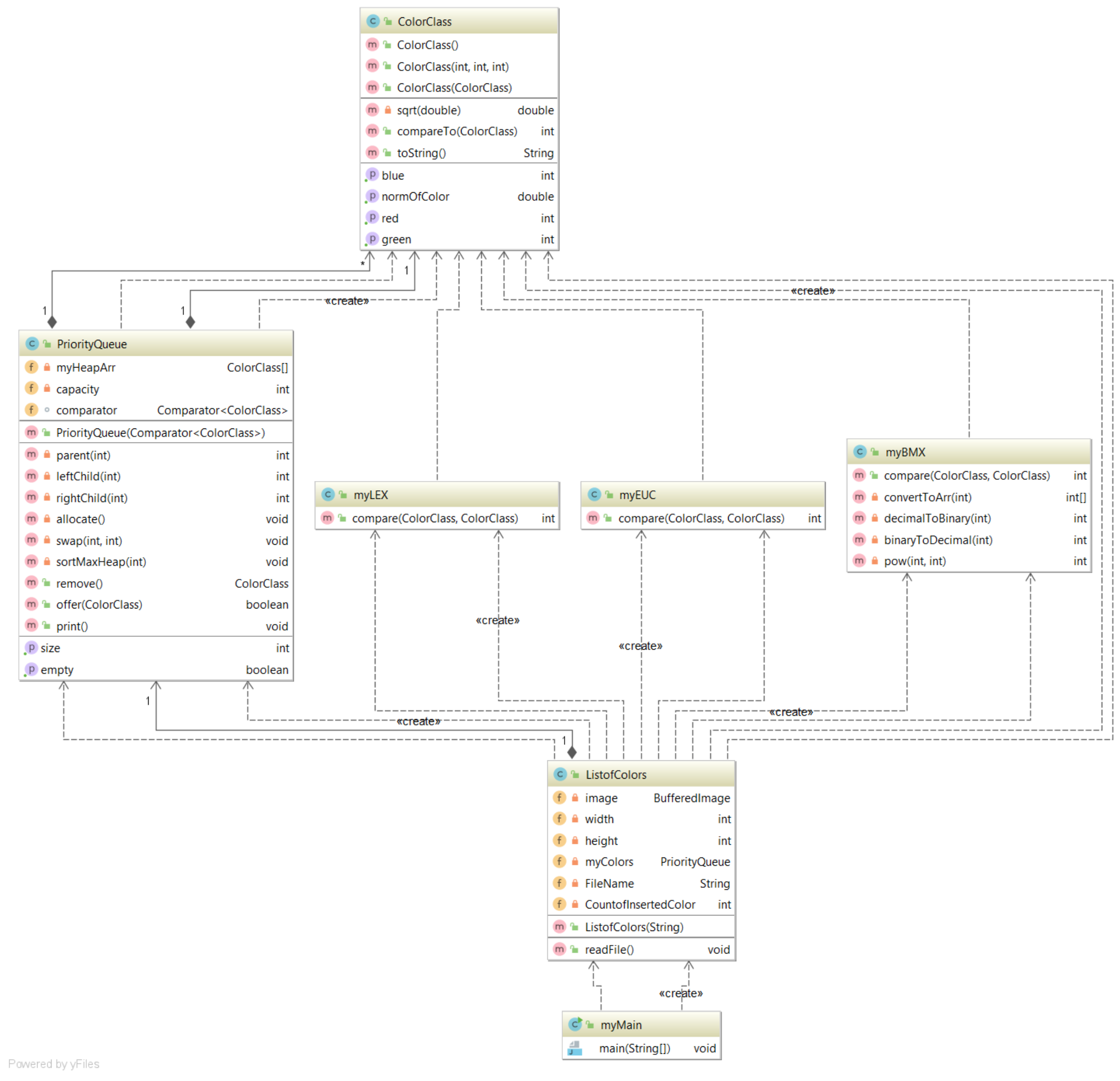
1 INTRODUCTION

1.1 Problem Definition

In this homework, the user gives us a image (which is in .png format) and we will read its pixels one by one. In doing so, we assign each pixel we read to our PriorityQueue, which we created. And after the number of these pixels that we send is 100, we delete each pixel we put on our PriorityQueue from this list according to certain rankings. These sorting shapes are 3 pieces; **LEX**, **EUC** and **BMX**. We are asked to sort each pixel we place in our PriorityQueue in a special way using these sort types. When deleting the elements in these lists, which are sorted by 3 conditions, the deletion command of the 3 lists is requested to work simultaneously. To do this, we are also asked to take advantage of **threads**.

2 METHOD

2.1 Class Diagrams



2.2 Problem Solution Approach

2.2.1 PriorityQueue Class

In this class, we need to create the maximum heap to sort the pixels. To do this, we need to create a ColorClass type array. And we need to keep the length and capacity of this series. Thus, according to the length of the series can be reached from the parents over their children. In addition, we can easily delete the maximum element of our heap'ı can put our new element. As I wrote in the definetion part of the problem, I listed the pixels in three different conditions. To distinguish these sorting shapes from others, I keep the value of comparator in the Comparator<ColorClass> type to determine what sort type that user wants. After keeping this sort type in comparator, I create a Comparator class of the type of my ColorClass class, and I override the compare method. In this compare method, I look at the comparator to find out what type of sorting user wants to perform, and then apply it to it. And so I'm sorting out all the colors I get in my PriorityQueue class. In the **LEX** ranking, I used my compareTo method which is in my ColorClass class and I compare two colors according to the ratios of red, green, blue colors. In the **EUC** ranking, I use the normOfColor element in my ColorClass to compare between two colors. In the **BMX** ranking, firstly I took the red, green and blue colors from the two ColorClass elements, and after that I convert these colors (red, green, and blue) to the **8-bit** binary. Then I created two **24-bit** arrays based on bit mixing and I assigned binary numbers as specified. Then I convert these two arrays to decimal and I made a comparison between the two numbers. I create three class to compare these two colors. These classes are **myBMX**, **myEUC**, **myLEX**.

2.2.2 ColorClass Class

I created this class to keep the color ratios of each pixel I read. I took three different integer elements for red, green and blue. In addition, after calculating the norm of a pixel, I created a double type NormOfColor element to hold it in this class. When I was calculating the norm of a pixel, I took the squares of these three colors and took the root of the result. And also I overrided the compareTo method to compare the colors of two elements (Red > Green > Blue).

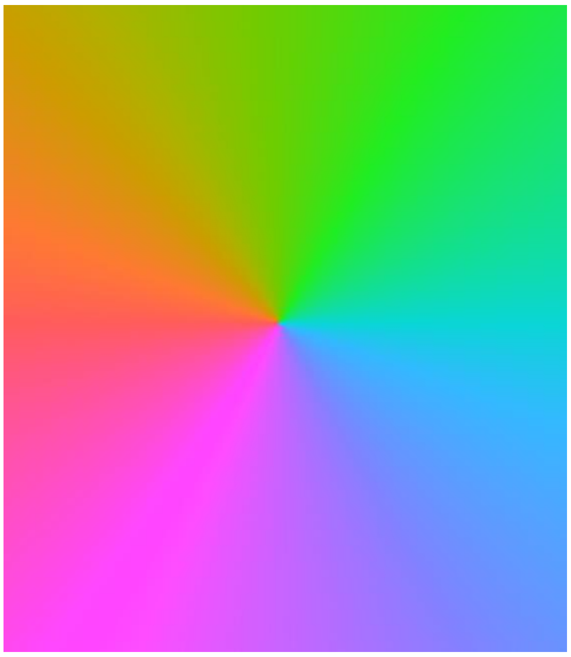
2.2.3 ListofColors Class

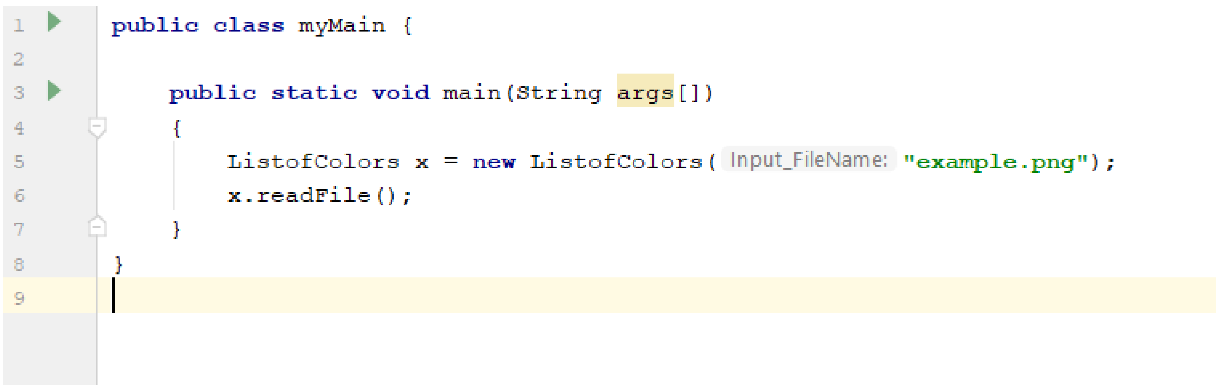
I created this class to read all the pixels from the file and keep it in my PriorityQueue class. To do this firstly I took the filename in my constructor. And then I create three PriorirtyQueue elements in my readFile method. The three elements I create will help to sort by three different conditions every time I add pixels. After that I create my first thread and overrided its **run** method and I added my elements to this three PriorirtyQueue elements one by one. While we are adding, if the count of we added elemets is bigger than 99, our three threads will work too. Then we start them too and we will delete our elements in these three PriorirtyQueue elements. Actually, if we have too many pixels the working time of this class takes too long.

3 RESULT

3.1 Test Cases

Our image is:

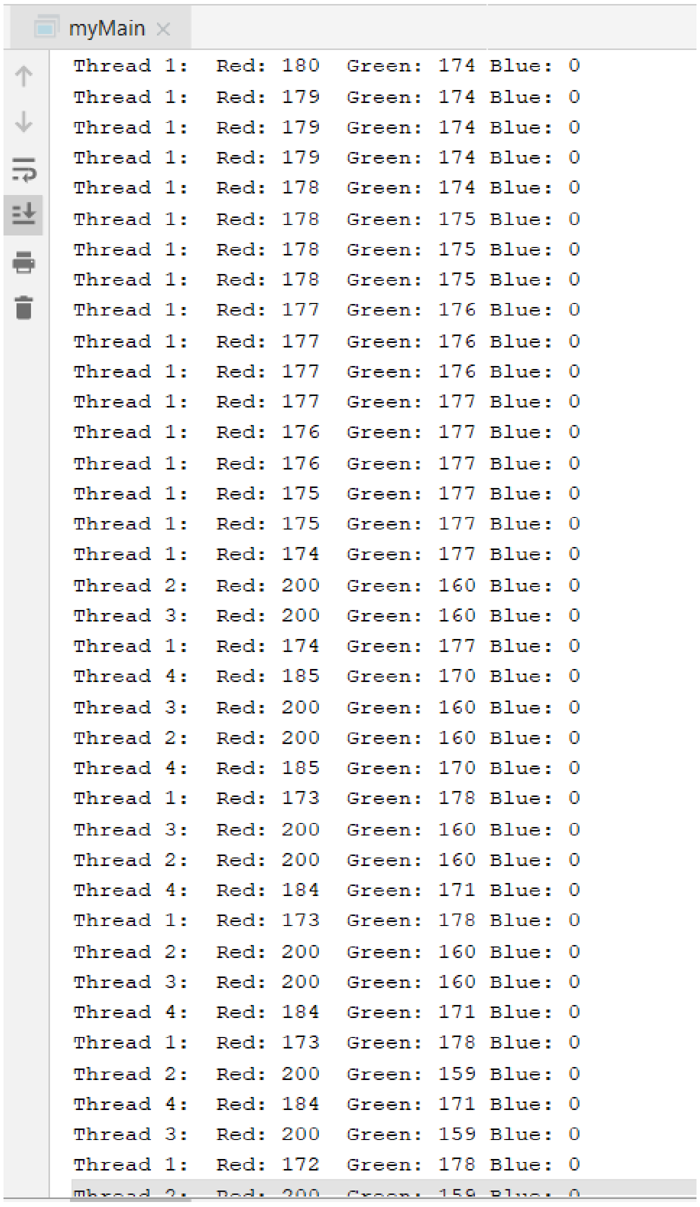




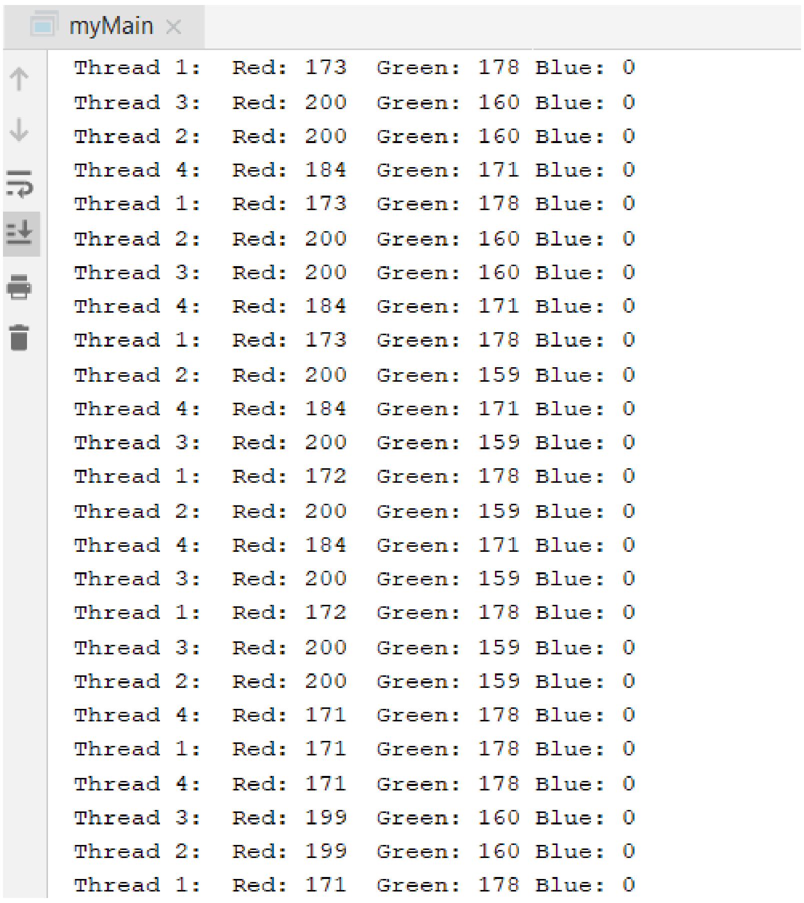
```
public class myMain {

    public static void main(String args[])
    {
        ListofColors x = new ListofColors( Input_FileName: "example.png");
        x.readFile();
    }
}

```

3.2 Running Results

The result is too long but we can show a little part of our result.



```
Thread 1:  Red: 180  Green: 174 Blue: 0
Thread 1:  Red: 179  Green: 174 Blue: 0
Thread 1:  Red: 179  Green: 174 Blue: 0
Thread 1:  Red: 179  Green: 174 Blue: 0
Thread 1:  Red: 178  Green: 174 Blue: 0
Thread 1:  Red: 178  Green: 175 Blue: 0
Thread 1:  Red: 178  Green: 175 Blue: 0
Thread 1:  Red: 178  Green: 175 Blue: 0
Thread 1:  Red: 177  Green: 176 Blue: 0
Thread 1:  Red: 177  Green: 176 Blue: 0
Thread 1:  Red: 177  Green: 176 Blue: 0
Thread 1:  Red: 177  Green: 177 Blue: 0
Thread 1:  Red: 176  Green: 177 Blue: 0
Thread 1:  Red: 176  Green: 177 Blue: 0
Thread 1:  Red: 175  Green: 177 Blue: 0
Thread 1:  Red: 175  Green: 177 Blue: 0
Thread 1:  Red: 174  Green: 177 Blue: 0
Thread 2:  Red: 200  Green: 160 Blue: 0
Thread 3:  Red: 200  Green: 160 Blue: 0
Thread 1:  Red: 174  Green: 177 Blue: 0
Thread 4:  Red: 185  Green: 170 Blue: 0
Thread 3:  Red: 200  Green: 160 Blue: 0
Thread 2:  Red: 200  Green: 160 Blue: 0
Thread 4:  Red: 185  Green: 170 Blue: 0
Thread 1:  Red: 173  Green: 178 Blue: 0
Thread 3:  Red: 200  Green: 160 Blue: 0
Thread 2:  Red: 200  Green: 160 Blue: 0
Thread 4:  Red: 184  Green: 171 Blue: 0
Thread 1:  Red: 173  Green: 178 Blue: 0
Thread 2:  Red: 200  Green: 160 Blue: 0
Thread 3:  Red: 200  Green: 160 Blue: 0
Thread 4:  Red: 184  Green: 171 Blue: 0
Thread 1:  Red: 173  Green: 178 Blue: 0
Thread 2:  Red: 200  Green: 159 Blue: 0
Thread 4:  Red: 184  Green: 171 Blue: 0
Thread 3:  Red: 200  Green: 159 Blue: 0
Thread 1:  Red: 172  Green: 178 Blue: 0
Thread 2:  Red: 200  Green: 159 Blue: 0
```



```
Thread 1:   Red: 173   Green: 178 Blue: 0
Thread 3:   Red: 200   Green: 160 Blue: 0
Thread 2:   Red: 200   Green: 160 Blue: 0
Thread 4:   Red: 184   Green: 171 Blue: 0
Thread 1:   Red: 173   Green: 178 Blue: 0
Thread 2:   Red: 200   Green: 160 Blue: 0
Thread 3:   Red: 200   Green: 160 Blue: 0
Thread 4:   Red: 184   Green: 171 Blue: 0
Thread 1:   Red: 173   Green: 178 Blue: 0
Thread 2:   Red: 200   Green: 159 Blue: 0
Thread 4:   Red: 184   Green: 171 Blue: 0
Thread 3:   Red: 200   Green: 159 Blue: 0
Thread 1:   Red: 172   Green: 178 Blue: 0
Thread 2:   Red: 200   Green: 159 Blue: 0
Thread 4:   Red: 184   Green: 171 Blue: 0
Thread 3:   Red: 200   Green: 159 Blue: 0
Thread 1:   Red: 172   Green: 178 Blue: 0
Thread 3:   Red: 200   Green: 159 Blue: 0
Thread 2:   Red: 200   Green: 159 Blue: 0
Thread 4:   Red: 171   Green: 178 Blue: 0
Thread 1:   Red: 171   Green: 178 Blue: 0
Thread 4:   Red: 171   Green: 178 Blue: 0
Thread 3:   Red: 199   Green: 160 Blue: 0
Thread 2:   Red: 199   Green: 160 Blue: 0
Thread 1:   Red: 171   Green: 178 Blue: 0
```