Gebze Technical University
Computer Engineering

CSE 222 - 2019 Spring

HOMEWORK 5 REPORT

Gökhan HAS
161044067

Course Assistant: Özgü GÖKSU

1 INTRODUCTION

1.1 Problem Definition

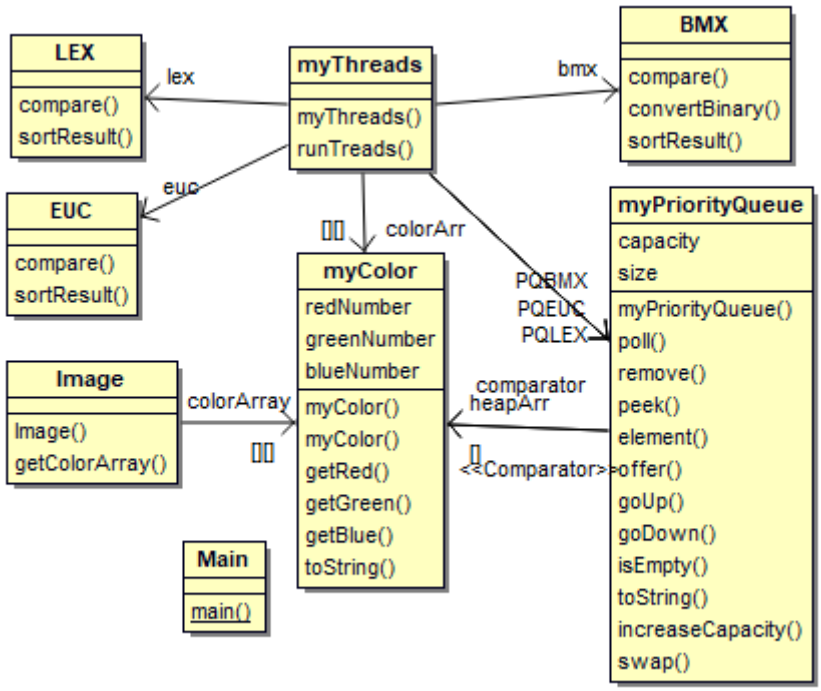
In this assignment, an image file is read and various operations are required. First, we asked for a priorityqueue implementation. Then, using this priorityqueue, a number of threads can be used to add or remove threads. Some important points have been specified for PriorityQueue. These are LEX, EUC and BMX. It was necessary to follow the rules.

1.2 System Requirements

I tested this code on two different systems. One is a desktop computer with a high-performance processor, the other is a laptop with a middle-featured processor. On the desktop the program was completed in less time. And I got results that I believe are more accurate. The process on the other computer took a long time and I got different output.

2 METHOD

2.1 Class Diagrams



2.2 Problem Solution Approach

First of all I started with the homework to understand the requirements. I decided to do myColor class for reading from the file. This class will hold the red, green and blue number as integer. I wrote the get methods of these variables for this class. They were used when pressing the screen. Two constructors were also written. One is receiving 3 integers as a parameter while the other is receiving another myColor reference. And I override the toString method to help with test work.

The second class I used for the homework was Image class. In this class, I read the image from the file and put it in a two-dimensional array (example.png). I think the threads would normally be used here, but I created another class for them. I have written a getColorArray () method in Image.java class for use in class. This method returns a two-dimensional array. **I tried to read the threads from the file, but I got the same results. So I wrote a different class to make it look nice.**

In myThread class, I am reading the file and starting to run my threads. I've arranged to run Thread 2,3,4 in Thread 1. The threads worked faster on the computer with performance. To run threads, we need to override the run () method. Display outputs are put on the test section. The threads look like they're doing the same thing towards the end. Some operations are not displayed.

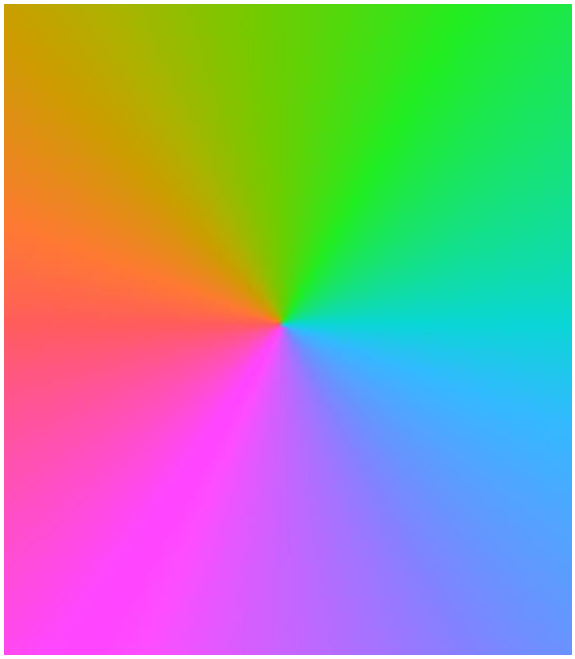
LEX, EUC and BMX classes were written for comparator objects. Each class has its own name. Fx , LEX is the biggest one in red.

myPriorityQueue class is the main class in homework. A heap implementation is required. I've used the array in myPriorityQueue class. At first I gave arraye 20 capacity. And when you equalize with capacity e, the increaseCapacity () method will be increased. When using Heap, when using array, if i index is 2 * i + 1 left child, 2 * i + 2 right child. And the parent always needs to be bigger than the children (in the minimum Heap is the opposite). Some methods from PriorityQueue have been implemented. With the poll () method, the remove () method does the same. Heap’s element is removing and deleted element is returning. The only difference is poll (), if size == 0 it returns null, and remove () throws the NoSuchElementException.

The difference between peek () and element () method is likewise. The most challenging methods were the offer () and remove () methods. The offer () method is used when adding elements. Of course, after this addition process, the array needs to be checked whether there is another great element. Because, if the element added is the largest, it must remain at the top. And if there is an element in the middle, it needs to be placed in the right place. They are controlled both in the insertion process and in the deletion process. Size smallness situations are controlled according to the comparator object. There is also a constructor with reference to Comparotor. Heap also offer the method O (logN), because the depth of the tree is going, all the elements are not navigated. For the same reason, O (logN) in deleting methods such as remove (). Peek () and element () methods O (1). Because they return the first element of the array.

3 RESULT

3.1 Test Cases



3.2 Running Results

```
Thread3-PQEUC: [255, 80, 188]
Thread3-PQEUC: [255, 80, 188]
Thread2-PQLEX: [255, 79, 191]
Thread2-PQLEX: [255, 79, 190]
Thread4-PQBMX: [255, 77, 207]
Thread4-PQBMX: [255, 78, 198]
Thread2-PQLEX: [255, 79, 190]
Thread3-PQEUC: [255, 80, 187]
Thread4-PQBMX: [255, 81, 180]
Thread2-PQLEX: [255, 79, 189]
Thread3-PQEUC: [255, 80, 187]
Thread4-PQBMX: [255, 81, 180]
Thread2-PQLEX: [255, 78, 202]
Thread4-PQBMX: [255, 95, 88]
Thread2-PQLEX: [255, 78, 202]
Thread2-PQLEX: [255, 78, 202]
Thread3-PQEUC: [255, 80, 182]
Thread4-PQBMX: [255, 79, 191]
Thread4-PQBMX: [255, 81, 179]
Thread3-PQEUC: [255, 80, 182]
Thread2-PQLEX: [255, 78, 203]
Thread4-PQBMX: [255, 79, 191]
Thread2-PQLEX: [255, 78, 203]
Thread3-PQEUC: [255, 80, 186]
Thread4-PQBMX: [255, 81, 179]
Thread2-PQLEX: [255, 78, 204]
Thread2-PQLEX: [255, 78, 201]
Thread2-PQLEX: [255, 78, 201]
Thread4-PQBMX: [255, 79, 191]
Thread3-PQEUC: [255, 81, 179]
Thread3-PQEUC: [255, 80, 183]
Thread3-PQEUC: [255, 81, 176]
```

```
Thread3-PQEUC: [208, 154, 4] size => 109
Thread4-PQBMX: [201, 159, 0] size => 112
Thread3-PQEUC: [208, 154, 4] size => 108
Thread4-PQBMX: [208, 154, 4] size => 112
Thread2-PQLEX: [209, 153, 5] size => 112
Thread2-PQLEX: [208, 154, 4] size => 108
Thread2-PQLEX: [208, 154, 4] size => 107
Thread4-PQBMX: [208, 154, 4] size => 108
Thread3-PQEUC: [209, 153, 5] size => 116
Thread 1: [206, 156, 2]
Thread2-PQLEX: [209, 153, 5] size => 113
Thread3-PQEUC: [209, 153, 5] size => 113
Thread 1: [206, 156, 2]
Thread4-PQBMX: [208, 154, 4] size => 114
Thread 1: [206, 156, 2]
Thread3-PQEUC: [209, 153, 5] size => 112
Thread4-PQBMX: [203, 157, 0] size => 114
Thread 1: [206, 155, 2]
Thread 1: [206, 155, 2]
Thread 1: [206, 155, 2]
Thread 1: [207, 155, 3]
Thread4-PQBMX: [208, 154, 4] size => 116
Thread3-PQEUC: [209, 153, 5] size => 114
Thread 1: [207, 155, 3]
Thread4-PQBMX: [210, 153, 6] size => 118
Thread4-PQBMX: [210, 153, 6] size => 117
Thread2-PQLEX: [209, 153, 5] size => 114
Thread2-PQLEX: [208, 154, 4] size => 101
Thread3-PQEUC: [210, 153, 6] size => 117
Thread 1: [207, 155, 3]
Thread 1: [207, 155, 3]
Thread 1: [208, 154, 4]
Thread 1: [208, 154, 4]
Thread 1: [208, 154, 4]
Thread 1: [208, 154, 4]
Thread 1: [208, 154, 4]
Thread 1: [208, 154, 4]
Thread 1: [208, 154, 4]
Thread 1: [209, 153, 5]
Thread 1: [209, 153, 5]
Thread 1: [209, 153, 5]
Thread 1: [210, 153, 6]
Thread4-PQBMX: [210, 153, 6] size => 122
Thread3-PQEUC: [210, 153, 6] size => 122
Thread3-PQEUC: [209, 153, 5] size => 121
Thread2-PQLEX: [210, 153, 6] size => 122
Thread4-PQBMX: [202, 158, 0] size => 119
Thread3-PQEUC: [209, 153, 5] size => 120
```