Gebze Technical University
Computer Engineering

CSE 222 - 2019 Spring

HOMEWORK 05 REPORT

Muhammed ÖZKAN
151044084

Course Assistant:Özgü GÖKSU

1 INTRODUCTION

1.1 Problem Definition

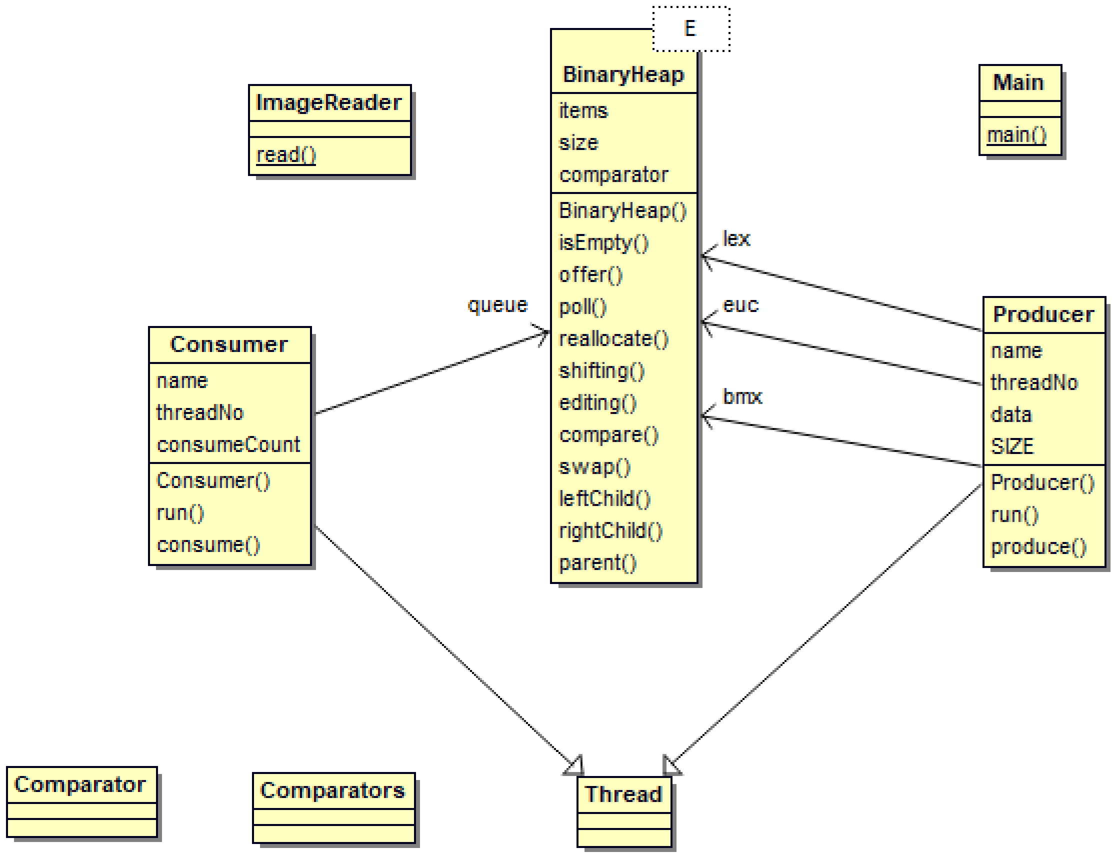
In this assignment, we first add the color information of the pixels to the three priority queue data structures. Each of these three priority queues must have separate prioritizers. These priorities are as follows. 1-LEX 2-EUC 3-BMX. The working structure of the program should be as follows. As soon as the first 100 pixels are inserted, it will create and start the 3 threads, and then continue inserting the remaining pixels. There are two problems we need to solve in this program. Producer Consumer Problem, Synchronization Problem.

1.2 System Requirements

We will develop this program for all devices running Java. The classes used in the solution of the problems have been developed considering the minimum possible memory consumption and execution time. The size of the Binary Heap class on memory is to vary depending on the type of data to keep. Memory size shows a linear increase. In case of proper use, the prepared programs can be used in any environment, even on a smartphone or even on a smart watch.

2 METHOD

2.1 Class Diagrams



2.2 Use Case

The software works on the console screen. The user must specify the path of the files to be used as the parameter to the program before running the programs.

For example:

-java program_name C:\test_file.png or jpg

2.3 Problem Solution Approach

If the PQ that is being processed by a thread happens to be temporarily empty, and the number of pixels it has processed so far is less than WxH, then it must wait some times added new element. Or if thread 1 tries to insert a value into a PQ or thread 2 3 4 try to remove a value from PQ, queues must be locked against other processes in both processes. In consideration of these problems, the following procedures were done to solve the problem. This program is implemented(Producer-Consumer Part) according to the **design pattern** rules. I used the design patterns to avoid code repetitions and to provide code readability. Once implemented more than one used for example thread2 thread3 thread4 was created from the same consumer object type. The comparator system in the Binary Heap(Priority Queue) is also a design pattern.

2.4 Complexity of Functions

The complexity of functions is calculated according to the number and structure of the loops they contain. Since the complexity calculations are considered infinite, the comparison, assignment and similar operations within the functions are not included in the calculations since they do not have any meaning in infinity.

Function Name	**Complexity**	**Big O Notation**
BinaryHeap.isEmpty()	$T_1(n)=1$ =1	*O(1)*
BinaryHeap.offer()	$T_2(n)=T_5(n)+1$ =log n+1	*O(log n)*
BinaryHeap.poll()	$T_3(n)=T_7(n)+T_6(n)+1$ =log n+2	*O(log n)*
BinaryHeap.reallocate()	$T_4(n)=n$ =n	*O(n)*
BinaryHeap.shifting()	$T_5(n)=T_7(n)$*log n =log n	*O(log n)*
BinaryHeap.editing()	$T_6(n)=\log n$ =log n	*O(log n)*
BinaryHeap.swap()	$T_7(n)=1$ =1	*O(1)*
Producer	$T_8(n)=n$ =n	*O(n)*
Consumer	$T_9(n)=n$ =n	*O(n)*

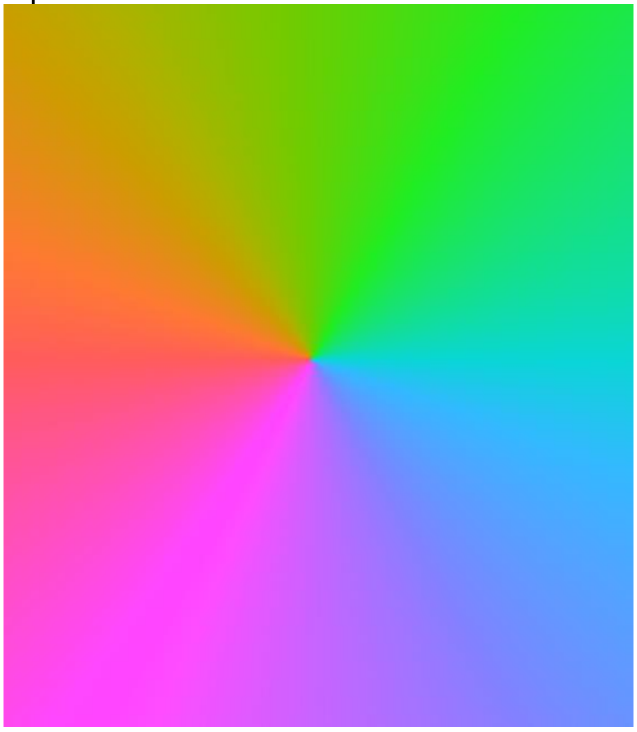
3 RESULT

3.1 Test Cases

I tested the program with images of various sizes and type(png or jpg).

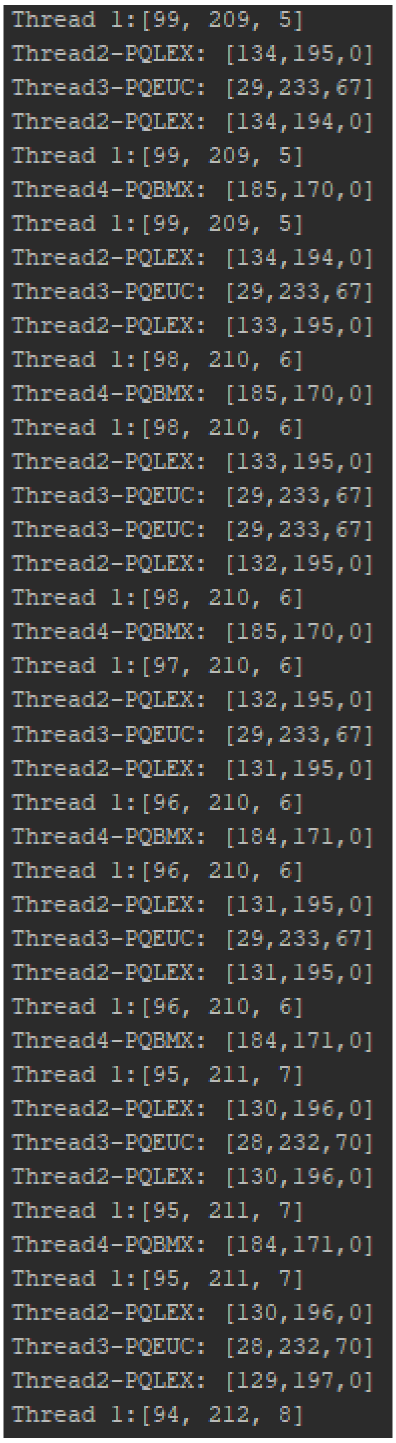
3.2 Running Results

Input:



Size: 567 x 651 px

Output:



```
Thread 1:[99, 209, 5]
Thread2-PQLEX: [134,195,0]
Thread3-PQEUC: [29,233,67]
Thread2-PQLEX: [134,194,0]
Thread 1:[99, 209, 5]
Thread4-PQBMX: [185,170,0]
Thread 1:[99, 209, 5]
Thread2-PQLEX: [134,194,0]
Thread3-PQEUC: [29,233,67]
Thread2-PQLEX: [133,195,0]
Thread 1:[98, 210, 6]
Thread4-PQBMX: [185,170,0]
Thread 1:[98, 210, 6]
Thread2-PQLEX: [133,195,0]
Thread3-PQEUC: [29,233,67]
Thread3-PQEUC: [29,233,67]
Thread2-PQLEX: [132,195,0]
Thread 1:[98, 210, 6]
Thread4-PQBMX: [185,170,0]
Thread 1:[97, 210, 6]
Thread2-PQLEX: [132,195,0]
Thread3-PQEUC: [29,233,67]
Thread2-PQLEX: [131,195,0]
Thread 1:[96, 210, 6]
Thread4-PQBMX: [184,171,0]
Thread 1:[96, 210, 6]
Thread2-PQLEX: [131,195,0]
Thread3-PQEUC: [29,233,67]
Thread2-PQLEX: [131,195,0]
Thread 1:[96, 210, 6]
Thread4-PQBMX: [184,171,0]
Thread 1:[95, 211, 7]
Thread2-PQLEX: [130,196,0]
Thread3-PQEUC: [28,232,70]
Thread2-PQLEX: [130,196,0]
Thread 1:[95, 211, 7]
Thread4-PQBMX: [184,171,0]
Thread 1:[95, 211, 7]
Thread2-PQLEX: [130,196,0]
Thread3-PQEUC: [28,232,70]
Thread2-PQLEX: [129,197,0]
Thread 1:[94, 212, 8]
```