Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 5 REPORT

Elif Akgün
1801042251

Course Assistant: Özgü Göksu

1 INTRODUCTION

1.1 Problem Definition

In this assignment, the pixels of a given image were ordered. Each pixel contains three colors: red, blue and green. In fact, each pixel is a three-element vector. To sort the vectors, certain sorting criteria are needed. Three sorting criteria were given for this.

-LEX: standard lexicographical comparison from discrete math.
-EUC: whichever vector has the greater L2 norm is considered greater (this is actually a preordering
relation since it's not anti-symmetric but that's ok for this context).
-BMX:

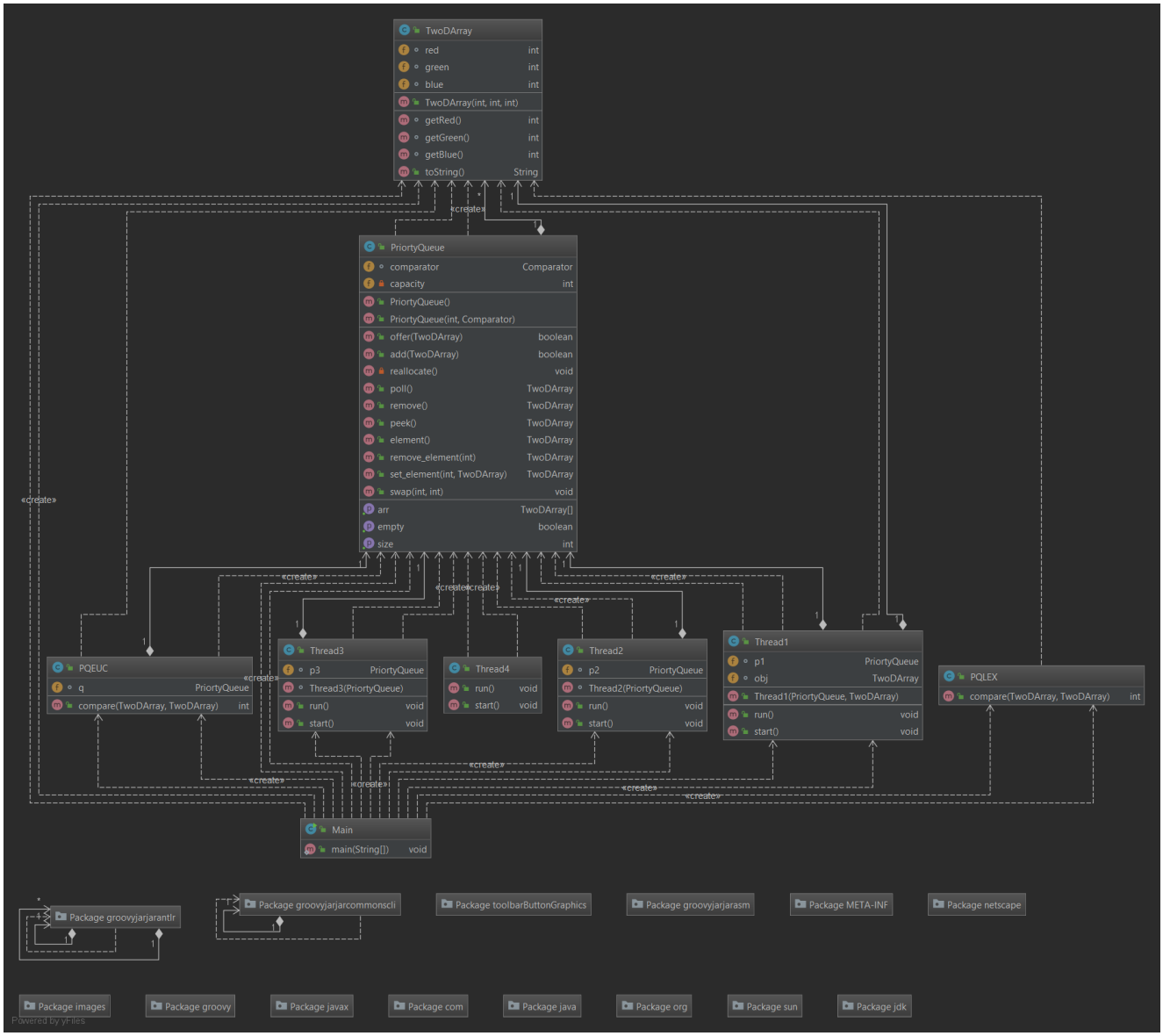
According to these criteria, the vectors will be sorted with the help of the Priorty Queue data structure and the max pixel will be found for each sorting criteria.

1.2 System Requirements

My solution also does not require a specific piece of hardware. This program can work anywhere with JVM and can work with 128KB of memory, it does not keep memory so much and it needs the interface to work on the smart phone. It can work if it is provided.

2 METHOD

2.1 Class Diagrams



2.2 Use Case Diagrams

The only thing that a user needs to do is save the image in the file thats pixels compare. User can run this program to click the run button.

2.3 Problem Solution Approach

During the project development phase, the objects that should be used first were considered and classes were created for those objects.
Priorty Queue data structure was used. Methods were implanted according to PQ structure. While the implementing, course book was used for help.

METHODS
boolean offer(TwoDArray item): Insert an item into the priority queue.
TwoDArray peek(): Returns the top of queue without removing. If array is empty returns null
TwoDArray element(): Returns the top of queue without removing. If array is empty throws NoSuchElementException
TwoDArray remove_element(int index): removes the element which in specified index

HELPER METHODS
boolean add(TwoDArray item): Adds the pixels to array.
void reallocate(): Method to reallocate the array containing the queue data.
TwoDArray poll(): Remove an item from the priority queue
TwoDArray remove(): This method same as poll method. There is a difference: If array is empty throws NoSuchElementException.
TwoDArray set_element(int index, TwoDArray item): Sets the the specified index
void swap(int obj1, int obj2): Swaps the elements in the specified index

COMPLEXITY

All methods space complexity are O(n) because of 1D array.
boolean offer(TwoDArray item): It has a while loop so its time complexity is O(n).
boolean add(TwoDArray item): It does not has a loop, its complexity is O(1).
TwoDArray poll(): It does not has a loop, its complexity is O(1).
TwoDArray remove(): It does not has a loop, its complexity is O(1).
TwoDArray peek(): Just returns first element so its complexity is O(1).
TwoDArray element(): Just returns first element so its complexity is O(1).
TwoDArray remove_element(int index): This method slips the array, complexity is O(n).
TwoDArray set_element(int index, TwoDArray item): It does not has a loop, its complexity is O(1).
boolean isEmpty(): Just returns arrays size is empty or full. Complexity is O(1).
void swap(int obj1, int obj2): Just swaps the elements. Complexity is O(1).

RESULT

2.4 Test Cases

This program has been loaded with an image to be tested and the image's pixels are read individually. Pixeller sent to queues for comparison. Read pixels and deleted pixels printed on the screen. Some part of test:

```
BufferedImage img = null;
File f = null;

// read image
try {
    f = new File( pathname: "C:\\\\Users\\\\user\\\\Desktop\\\\1801042251\\\\images.jpg");
    img = ImageIO.read(f);
} catch (IOException e) {
    System.out.println(e);
}

// get image's width and height
int width = img.getWidth();
int height = img.getHeight();

TwoDArray[][] array = new TwoDArray[width][height];

for (int i = 0; i < width; i++) {
    for (int j = 0; j < height; j++) {

        Color c = new Color(img.getRGB(i, j));
        array[i][j] = new TwoDArray(c.getRed(), c.getGreen(), c.getBlue());

    }
}
```

```
PQEUC p1 = new PQEUC();
PriortyQueue pqeuc = new PriortyQueue();
pqeuc.comparator = p1;

PQLEX p2 = new PQLEX();
PriortyQueue pqlex = new PriortyQueue();
pqlex.comparator = p2;

int k = 0;
int temp=0;

/*Firt Thread1 offer to the queues 100 pixel and other threads does nothing*/
for(int i=0;i<width;++i){
    for(int j=0; j<height; ++j){
        Thread1 t1 = new Thread1(pqeuc, array[i][j]);
        Thread1 t_1 = new Thread1(pqlex, array[i][j]);
        Thread t=new Thread(t1);
        t.start();
        t_1.start();
        /*t1.stopp();
        t_1.stopp();*/
        k++;
        if(k==100){
            temp=1;
            break;
        }
    }
    if (temp==1){
        break;
```

2.5 Running Results

```
Thread2-PQLEX: [Red: 250, Green: 70, Blue: 73]
Thread3-PQEUX: [Red: 250, Green: 70, Blue: 73]
Thread1: [Red: 250, Green: 70, Blue: 73]
Thread1: [Red: 250, Green: 70, Blue: 73]
Thread2-PQLEX: [Red: 250, Green: 70, Blue: 73]
Thread3-PQEUX: [Red: 250, Green: 70, Blue: 73]
Thread1: [Red: 250, Green: 70, Blue: 73]
Thread1: [Red: 250, Green: 70, Blue: 73]
Thread2-PQLEX: [Red: 250, Green: 70, Blue: 73]
Thread3-PQEUX: [Red: 250, Green: 70, Blue: 73]
Thread1: [Red: 250, Green: 70, Blue: 73]
Thread1: [Red: 250, Green: 70, Blue: 73]
Thread2-PQLEX: [Red: 250, Green: 70, Blue: 73]
Thread3-PQEUX: [Red: 250, Green: 70, Blue: 73]
Thread1: [Red: 250, Green: 70, Blue: 73]
Thread1: [Red: 250, Green: 70, Blue: 73]
Thread2-PQLEX: [Red: 250, Green: 70, Blue: 73]
Thread3-PQEUX: [Red: 250, Green: 70, Blue: 73]
Thread1: [Red: 250, Green: 70, Blue: 73]
```

```
Thread3-PQEUX: [Red: 244, Green: 48, Blue: 49]
Thread2-PQLEX: [Red: 241, Green: 89, Blue: 86]
Thread3-PQEUX: [Red: 244, Green: 48, Blue: 49]
Thread2-PQLEX: [Red: 241, Green: 89, Blue: 86]
Thread3-PQEUX: [Red: 244, Green: 48, Blue: 49]
Thread2-PQLEX: [Red: 241, Green: 87, Blue: 85]
Thread3-PQEUX: [Red: 244, Green: 48, Blue: 49]
Thread2-PQLEX: [Red: 241, Green: 87, Blue: 85]
Thread3-PQEUX: [Red: 244, Green: 48, Blue: 49]
Thread2-PQLEX: [Red: 241, Green: 87, Blue: 85]
Thread3-PQEUX: [Red: 244, Green: 48, Blue: 49]
Thread2-PQLEX: [Red: 241, Green: 86, Blue: 82]
Thread3-PQEUX: [Red: 244, Green: 48, Blue: 49]
Thread2-PQLEX: [Red: 241, Green: 86, Blue: 82]
Thread3-PQEUX: [Red: 244, Green: 48, Blue: 49]
Thread2-PQLEX: [Red: 240, Green: 200, Blue: 200]
Thread3-PQEUX: [Red: 244, Green: 48, Blue: 49]
Thread2-PQLEX: [Red: 240, Green: 200, Blue: 200]
Thread3-PQEUX: [Red: 244, Green: 48, Blue: 49]
Thread2-PQLEX: [Red: 240, Green: 200, Blue: 200]
Thread3-PQEUX: [Red: 247, Green: 39, Blue: 39]
Last size of PQEUC: 0
Last size of PQLEX: 0
```