CMPE150 ASSIGNMENT 3

PROBLEM DESCRIPTION:

In this assignment we were asked to apply different procedures on images to change some features of the images. Images were kept in ppm format and we applied the procedure according to ppm format rules. We used arguments to specify the parts. In first part we created the same file and named it “output.ppm”. In second part we took the average values and created the next file named “black-and-white.ppm”. In third part we used filters(kernel matrix) to emphasize the vertical or horizantal parts of the images. This process is called convolution. In final part we decreased the quality of the image by equating the pixel values of the neighbour pixels whose values are close enough depending on the range. This is called color quantization. To do this we used recursion.

PROBLEM SOLUTION:

I used different methods to do each part. In first part i created a method called “fileCreating” which takes an input array and creates a file named “output.ppm” which is identical to original image.In second part i created a method called gettingAverage which takes 2 parameters. First parameter’s name is the initial file and the second parameter is the name of the output file. I traversed the first file and found the averages and then created a file whose values are the average values. This led to a black and white image since all the color channels of the pixels are equal to each other. I named this file as “black-and-white.ppm”. In third part i used a method called convolution. In this method i used nested 5 for loops to calculate the desired value of the pixels. 3 loops(for each dimension of the array) + 2 loops(for each dimension of the kernel matrix). After i calculated the desired values and created the array, i print this array to another ppm file called “convolution.ppm” using fileCreating method. In the final part i used a method called quantization and quantizationUtil. My solution’s logic was based on the mazeSolution. I created an array called check whose values are initiliazed as true. If a pixel’s value changes through the process i kept this information in this array at corresponding index. I created a method called isSafe which checks if the index is valid and also if it is in the range. I used this method in quantizationUtil and made recursive calls. By doing this i was able to generate the desired quantized image.

IMPLEMENTATION:

```java
import java.io.*;
import java.util.*;

public class AT2019400288 {
      public static void main(String[] args) throws FileNotFoundException {

            int mode = Integer.parseInt(args[0]); //mode number
            String InputFile = args[1]; // name of the original image.
            String Filter = args[2]; //filter's name or range value.

            //I explained the functions at the point i created them.
            //So i didn't explain them below.
            // first part.
            if (mode == 0) {
                  int[][][] numbers = arrayCreating(InputFile);
                  fileCreating(numbers, "output.ppm");
            //second part

            }
            if (mode == 1) {
                  int[][][] numbers = arrayCreating(InputFile);
                  gettingAverage(InputFile,"black-and-white.ppm");
            //third part
            //after i calculate the convolution part, i call the getting
average method.
            }
            if (mode == 2) {
                  int[][][] numbers = arrayCreating(InputFile);

                  convolution(InputFile,numbers,Filter);
                  gettingAverage("convolution.ppm","convolution.ppm");
            }
            // fourth part
            if(mode == 3){
                  int[][][] numbers = arrayCreating(InputFile);
                  int range = Integer.parseInt(Filter);
                  quantization(numbers,range);
            }

      }

       // I calculate the average values of the triplet pixels and assign
      // them the same value(average value)
      public static void gettingAverage(String InputFile,String target)
                  throws FileNotFoundException {
            int[][][] numbers = arrayCreating(InputFile); //i get the
original array.
            for (int i = 0; i < numbers.length; i++) {
                  for (int j = 0; j < numbers[i].length; j++) {
                        int average = 0;
                        for (int k = 0; k < 3; k++) {
                              average += numbers[i][j][k]; //i calculate the
average
                              if (k == 2) {
                                    numbers[i][j][0] = average / 3; //i
assign their new value.
                                    numbers[i][j][1] = average / 3;
                                    numbers[i][j][2] = average / 3;
                              }
                        }
                  }
            }
            fileCreating(numbers, target); // i create the file using the
updated array.

      }
 // I apply the convolution part here. First i create the array and then
apply the convolution process.
      public static void convolution(String
InputFile,int[][][]numbers,String Filter)throws FileNotFoundException{


            Scanner conv = new Scanner(new File(Filter));
            String s = conv.next();
            int p = s.indexOf("x");
            s = s.substring(0,p); // i find the size of the dimensions.
            int m = Integer.parseInt(s); //i change its type.
            int n = m;
            conv.nextLine();
            int[][] filter = new int[m][n]; // i initiliaz the filter 
array.
            for(int i=0;i<m;i++){
                  for(int j=0;j<n;j++){
                        filter[i][j] = conv.nextInt(); // i create the 
filter array by traversing
                                                            // the filter file.

                  }
            } //filter is created.
            int row = numbers.length-2;
            int column = numbers.length-2;
            int[][][] convoluted = new int[row][column][3]; //this will be 
the output array.
            // I calculate the value of the index inside these 5 nested 
loops.
            for(int k=0;k<3;k++){
            for(int i=0;i<row;i++){
                  for(int j=0;j<column;j++){
                        int convedvalue = 0;
                        for(int a=0;a<3;a++){
                              for(int b=0;b<3;b++){
                                    convedvalue += 
numbers[i+a][j+b][k]*filter[a][b];
                                    //I find its value.
                              }
                        }
                        convoluted[i][j][k]=convedvalue;
                        if(convoluted[i][j][k]<0){ //if they are out of the 
boundary values
                              convoluted[i][j][k]=0; //i make them equal to 
the boundary values.
                                                      //lower boundary.
                        }
                        if(convoluted[i][j][k]>255){ // upper boundary.
                              convoluted[i][j][k]=255;
                        }
                  }
            }
      }
            fileCreating(convoluted,"convolution.ppm"); //when it is 
finished
                                                            //i create the 
convolution file.
            //i call the average method in main!

      //In this function i perform the quantization operation.
      }
      public static void quantization(int[][][]quant,int range)throws 
FileNotFoundException{
            int i = quant.length; //size of the dimensions
            int j = quant[0].length;
            int[][][] sol = new int[i][j][3]; //This will be my solution 
array.
            boolean[][][] check = new boolean[i][j][3]; //In this array i 
keep the information if
                                                            // their values are 
changed or not.
            for(int a=0;a<check.length;a++){
                  for(int b=0;b<check[a].length;b++){
                        for(int c=0;c<3;c++){
                         check[a][b][c]= true; //Initially all of them are
unchanged. So i make them true.
                        }
                  }
            }



            //Inside these 3 loops i traverse the array
            //First row by row, then i move to next channel.
            for(int k=0;k<3;k++){ //for channel
            for(int m=0;m<quant.length;m++){ //for row
                  for(int n=0;n<quant[0].length;n++){ //for column

                             // below here i create the basepixel which
will be the
                            // values of its neighbours if they are in
range.
                             int basepixel=0;
                             if(sol[m][n][k]==0){
                               basepixel= quant[m][n][k];}
                             if(sol[m][n][k]!=0){
                                basepixel= sol[m][n][k];}
                             //First call of the neighbours. inside this
function i make the recursive calls.

      quantizationUtil(quant,range,m,n,k,sol,check,basepixel);

                        }
                  }
            }
             // After these loops and recursive calls are finished my
solution array is created.
            // So i create the quantization file.
            fileCreating(sol,"quantization.ppm");
            }
      //In this file i change the values and make recursive calls.
      public static void quantizationUtil(int[][][]quant,int range,int
x,int y,int z,int[][][]sol,boolean[][][] check,int basepixel){

            //If it is not safe it does not enter.
            if(isSafe(quant,check,x,y,z,range,basepixel)){ //indexler doğru
mu, daha önce değişmiş mi.
                  sol[x][y][z] = basepixel; //If it is safe i change its
value to basepixel.
                  check[x][y][z] = false; //And i keep the information that
it is changed.

                  //Recursive calls of its neighbours.They also check their
neighbours and so on.

      quantizationUtil(quant,range,x+1,y,z,sol,check,basepixel);
                  quantizationUtil(quant,range,x-
1,y,z,sol,check,basepixel);

      quantizationUtil(quant,range,x,y+1,z,sol,check,basepixel);
                  quantizationUtil(quant,range,x,y-
1,z,sol,check,basepixel);

      quantizationUtil(quant,range,x,y,z+1,sol,check,basepixel);
```

```
                        quantizationUtil(quant,range,x,y,z-
1,sol,check,basepixel);


                }

        }
        //This is very similar to the maze question and pretty small but
practical function.
        //It checks 3 things. 1)Is it a valid index 2) Has it changed before
3) Is it in range.
        //If all of them are satisfied it returns true.
        public static boolean isSafe(int[][][]quant,boolean[][][]check,int
x,int y,int z,int range,int basepixel){
                return(x>=0 &&x< quant.length &&y>=0 && y<quant[0].length
&&z>=0 &&z<3 && check[x][y][z]&&Math.abs(basepixel-quant[x][y][z])<=range);
        }


        // It creates an array of the given file. File's name is the
parameter.
        public static int[][][] arrayCreating(String InputFile)
                        throws FileNotFoundException {
                File f = new File(InputFile);
                Scanner input = new Scanner(f); //Scanner object is created.

                input.nextLine(); //consume the next line.
                int i = input.nextInt(); // row dimension
                int j = input.nextInt(); // column dimension

                input.nextInt(); // consumes the next token.(max quanta).
                int numbers[][][] = new int[i][j][3];

                //By 3 nested for loops array is created.
                for (int m = 0; m < numbers.length; m++) {
                        for (int n = 0; n < numbers[m].length; n++) {
                                for (int k = 0; k < 3; k++) {
                                        numbers[m][n][k] = input.nextInt();
                                }

                        }
                }
                return numbers; // I return the created array.
        }
         //It creates a ppm file of the given array. Its name is the second
parameter.
                public static void fileCreating(int[][][] numbers, String name)
                                throws FileNotFoundException {

                        int i = numbers.length;
                        int j = numbers[1].length; // nur topu gibi array'imiz
var.

                        PrintStream output = new PrintStream(new File(name)); //
output file

                              // is created.
                        output.println("P3");
                        output.println(i + " " + j);
                        output.println("255"); //Header part is finished.
                        for (int m = 0; m < numbers.length; m++) { // for row
```

```
                              for (int n = 0; n < numbers[m].length; n++) { //
for column
                                   for (int k = 0; k < 3; k++) { // for channel
                                        output.print(numbers[m][n][k]);
                                        if (k != 2) {
                                             output.print(" "); // if it is in
triplet print 1 space.
                                        } else {
                                             output.print("\t"); //if it is
between two triplets print tab.
                                        }
                                   }
                              }
                         }

                    }


}
```

CONCLUSION:

I was able to generate all the output files called output.ppm,black-and white.ppm,convolution.ppm,quantized.ppm respectively. When i changed the range values i got the same files posted on moodle.(Except for the blank spaces between each triplet) Therefore i believe i got the correct results and completed the project successfully.