Mohak Chandani - 191080016
Information Technology

EXPERIMENT NO. 1

AIM

To write a program that reads and displays your own photo on screen without using any inbuilt function to read and write/display image.

THEORY

Java implements a particular type of object called a BufferedImage for images in Java. A BufferedImage can be read from several distinct image types (i.e., BMP, HEIC, etc.). Not all of these are backed by ImageIO itself, but there are plugins to extend ImageIO and other libraries such as Apache Imaging and JDeli.

In Java itself, all the complexity of various image types is hidden, and we only work on BufferedImage. Java provides immediate access to the image pixels and color information and allows conversions and image processing.

1. java.io.File: To read and write an image file, we must import the File class. This class represents file and directory path names in general.

2. java.io.IOException: To handle errors, we use the IOException class.

3. java.awt.image.BufferedImage: To hold the image, we create the BufferedImage object; we use BufferedImage class. This object is used to store an image in RAM.

4. javax.imageio.ImageIO: To perform the image read-write operation, we will import the ImageIO class. This class has static methods to read and write an image.

5. Java.io.console : It is used to read from and write to the console, if one exists. Console is primarily a convenience class because most of its functionality is available through System.in and System.out. However, its use can simplify some types of console interactions, especially when reading strings from the console.

IMPLEMENTATION STEPS

In this program, we have created 3 functions getBlackAndWhite, getNegative and getGrayscale to convert the images into Black And White, Negative and Grayscale respectively. In the main function we have a infinite while loop which will iterate until the user exits.

There is a switch case loop with 4 conditions given below

case 1: getGrayscale(img,f,newFileName,extension);break;

case 2: getBlackAndWhite(img,f,newFileName,extension);break;

case 3: getNegative(img,f,newFileName,extension);break;

case 4: System.exit(0);

Each function takes BufferedImage, file name, extension, as the parameter. They get the image width and height. Two for loops are executed to iterate through each pixel and modification in the pixel value is made according to the function

CODE

```
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.io.Console;
import javax.imageio.ImageIO;

public class Exp1 {

  public static void getBlackAndWhite(BufferedImage img,File f,String
fileName,String extension){
      // get image's width and height
      int width = img.getWidth();
      int height = img.getHeight();

      System.out.println("Height of image is :" + height);
      System.out.println("Width of image is :" + width);

      // convert to black and white
      for (int y = 0; y < height; y++) {
         for (int x = 0; x < width; x++) {

            // Here (x,y)denotes the coordinate of image
            // for modifying the pixel value.
            int p = img.getRGB(x, y);
            int a = (p >> 24) & ((1 << 8) - 1);

            int r = (p >> 16) & ((1 << 8) - 1);
            int g = (p >> 8) & ((1 << 8) - 1);
            int b = p & ((1 << 8) - 1);
            // calculate average
            int avg = (r + g + b) / 3;

            // Black and white
            // Checking for threshold value
            if(avg < 127){
               p = (a << 24) | (0x00 << 16) | (0x00 << 8) | (0x00);
            }
            else{
               p = (a << 24) | (0xff << 16) | (0xff << 8) | (0xff);
            }

            img.setRGB(x, y, p);
         }
      }

      // write image
      try {
         String newImage = fileName+"_b&w."+extension;
         f = new File(newImage);
         ImageIO.write(img, extension, f);
         System.out.println("Operation successful");
         System.out.println("Check your file : " + newImage);
      }
      catch (IOException e) {
         System.out.println(e);
      }
   }

   public static void getNegative(BufferedImage img,File f,String
fileName,String extension){
      // get image's width and height
      int width = img.getWidth();
      int height = img.getHeight();

      System.out.println("Height of image is :" + height);
      System.out.println("Width of image is :" + width);

      // convert to negative
      for (int y = 0; y < height; y++) {
         for (int x = 0; x < width; x++) {

            // Here (x,y)denotes the coordinate of image
            // for modifying the pixel value.
            int p = img.getRGB(x, y);
            int a = (p >> 24) & ((1 << 8) - 1);

            int r = (p >> 16) & ((1 << 8) - 1);
            int g = (p >> 8) & ((1 << 8) - 1);
            int b = p & ((1 << 8) - 1);

            r = 255 - r;
            g = 255 - g;
            b = 255 - b;

            p = (a << 24) | (r << 16) | (g << 8) | b;

            img.setRGB(x, y, p);
         }
      }

      // write image
      try {
         String newImage = fileName+"_negative."+extension;
         f = new File(newImage);
         ImageIO.write(img, extension, f);
         System.out.println("Operation successful");
         System.out.println("Check your file : " + newImage);
      }
      catch (IOException e) {
         System.out.println(e);
      }
   }

   public static void getGrayscale(BufferedImage img,File f,String
fileName,String extension){
      // get image's width and height
```

```java
      int width = img.getWidth();
      int height = img.getHeight();

      System.out.println("Height of image is :" + height);
      System.out.println("Width of image is :" + width);

      // convert to grayscale
      for (int y = 0; y < height; y++) {
         for (int x = 0; x < width; x++) {

            // Here (x,y)denotes the coordinate of image
            // for modifying the pixel value.
            int p = img.getRGB(x, y);
            int a = (p >> 24) & ((1 << 8) - 1);

            int r = (p >> 16) & ((1 << 8) - 1);
            int g = (p >> 8) & ((1 << 8) - 1);
            int b = p & ((1 << 8) - 1);
            // calculate average
            int avg = (r + g + b) / 3;

            // Grayscale
            p = (a << 24) | (avg << 16) | (avg << 8) | avg;


            img.setRGB(x, y, p);
         }
      }

      // write image
      try {
         String newImage = fileName+"_grayscale."+extension;
         f = new File(newImage);
         ImageIO.write(img, extension, f);
         System.out.println("Operation successful");
         System.out.println("Check your file : " + newImage);
      }
      catch (IOException e) {
         System.out.println(e);
      }
   }

   public static void main(String args[]) throws IOException
   {
        Console c = System.console();
        String fileName = c.readLine("Enter file name : ");


        BufferedImage img = null;
        File f = null;

        // read image
        try {
            String []temp = fileName.split("\\.");
            String newFileName = temp[0];
            String extension = temp[1];
            f = new File(fileName);
            img = ImageIO.read(f);

            while(true){

                System.out.println("1. Grayscale");
                System.out.println("2. Black & White");
                System.out.println("3. Negative");
                System.out.println("4. Exit");

                int n = Integer.parseInt(c.readLine("Choose one : "));
                switch(n){
                    case 1: getGrayscale(img,f,newFileName,extension);break;
                    case 2: getBlackAndWhite(img,f,newFileName,extension);break;
                    case 3: getNegative(img,f,newFileName,extension);break;
                    case 4: System.exit(0);
                }

            }
        }
        catch (IOException e) {
            System.out.println(e);
        }


    }
}
```

OUTPUT

CONCLUSION

Thus, we have successfully written a program in Java which can read and view an image in Black and White, negative and also in grayscale.