UNIVERSITY OF MADRAS

GUINDY CAMPUS, CHENNAI – 600 025.

DEPARTMENT OF COMPUTER SCIENCE

MASTER OF COMPUTER APPLICATION

DIGITAL IMAGE PROCESSING LAB

Submitted by

Name : VIGNESH S

Reg.No : 36822026

Subject code : MSIC363

NOVEMBER 2023

UNIVERSITY OF MADRAS

GUINDY CAMPUS, CHENNAI – 600 025.

BONAFIDE CERTIFICATE

Certified that this is the bonafide record work of **DIGITAL IMAGE PROCESSING LAB** done by **VIGNESH S** with register number 36822026 of **II MCA** COMPUTER SCIENCE during the academic period of AUGUST 2023 – NOVEMBER 2023

Station:

Date:

Staff – in – charge
(Dr.PL. CHITHRA)

Head of the Department
(Dr. S. GOPINATH)

Submitted for the University practical Examination held on _____ at the Department of Computer Science, University of Madras, Guindy, Chennai - 600025.

TABLE OF CONTENTS

Signature of staff in charge

EX. NO:1	READING AND WRITING AN IMAGE
DATE:17/07/2023	

AIM:
To write a java program for image read and write using java image class

SOURCE CODE:

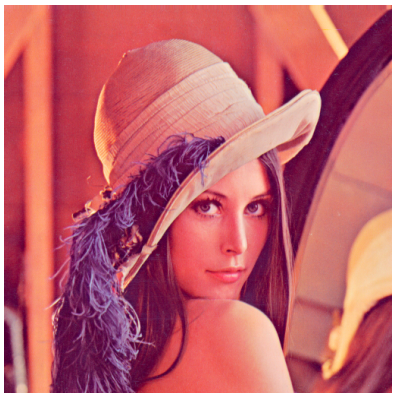
```
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
public class rw {
   public static void main(String[] args) {
      try {
         BufferedImage image = ImageIO.read(new File("Lenna_(test_image).png"));
         ImageIO.write(image, "PNG", new File("output10.png"));
         System.out.println(“writing complete”);
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

OUTPUT:

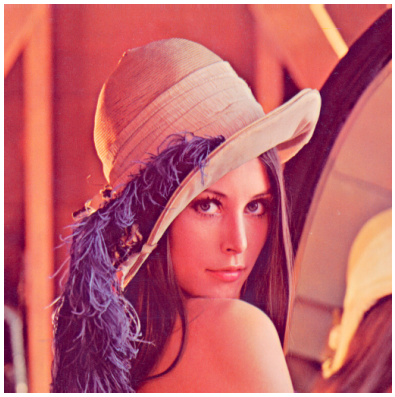
Reading complete

Writing complete

Input image :



Output image:



Result:

Thus the above program executed successfully

EX. NO: 2	GRAYSCALE TO NEGATIVE
DATE:24/07/2023	

AIM:

To convert a Grayscale Image to Negative in Java Programming.

SOURCE CODE:

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

public class gray2nega {
public static void main(String[] args) {
try {
BufferedImage grayscaleImage = ImageIO.read(new File("The-famous-Lena-image-often-
used-as-an-example-in-image-processing.png"));
BufferedImage negativeImage = convertToNegative(grayscaleImage);
ImageIO.write(negativeImage, "png", new File("negative_output2.png"));
System.out.println("Negative image processing complete.");
} catch (IOException e) {
e.printStackTrace();
}
}
private static BufferedImage convertToNegative(BufferedImage grayscaleImage) {
int width = grayscaleImage.getWidth();
int height = grayscaleImage.getHeight();
BufferedImage negativeImage = new BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);

for (int y = 0; y < height; y++)
for (int x = 0; x < width; x++) {
int rgb = grayscaleImage.getRGB(x, y);
int grayValue = rgb & 0xFF;

int negativeGrayValue = 255 - grayValue;
int negativeRGB = (negativeGrayValue << 16) | (negativeGrayValue << 8) |
negativeGrayValue;
negativeImage.setRGB(x, y, negativeRGB);
}
return negativeImage;
}
}
```

OUPTUT:

Input image:

Output image:

Result:

Thus the above program executed successfully

EX. NO:3	COLOR IMAGE TO GRAYSCALE
DATE:31/07/2023	

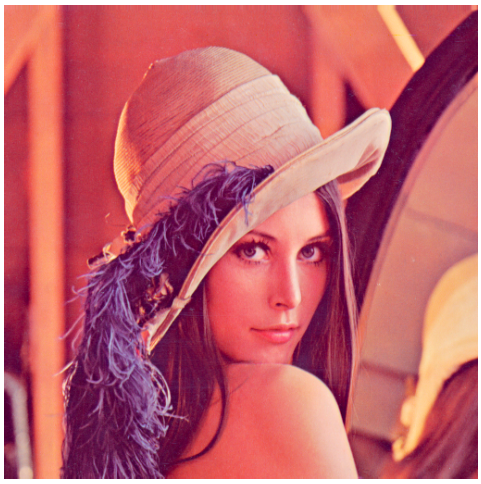
AIM:
To write a java program for convert colour image to gray scale using java image class

SOURCE CODE:

```
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
public class gray {
   public static void main(String[] args) {
      try {
         BufferedImage colorImage = ImageIO.read(new File("Doo.png"));
         BufferedImage bwImage = new BufferedImage(colorImage.getWidth(),
colorImage.getHeight(), BufferedImage.TYPE_BYTE_GRAY);
         bwImage.getGraphics().drawImage(colorImage, 0, 0, null);
         ImageIO.write(bwImage, "jpg", new File("output8.jpg"));
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

OUPTUT:

Input image:



Output Image:

Result:

Thus the above program executed successfully

EX. NO:4	COLOR IMAGE TO BLACK AND WHITE
DATE:07/08/2023	

AIM:

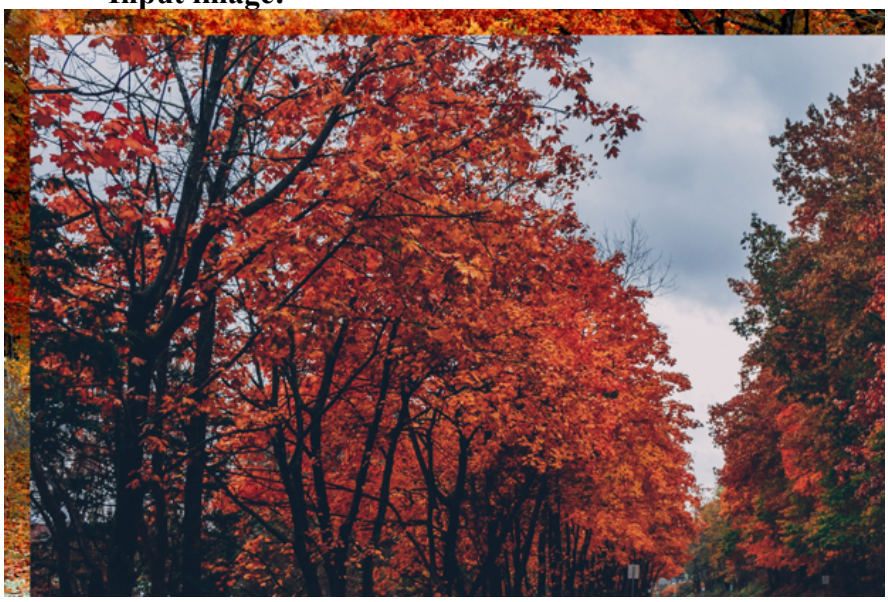
To write a java program for convert colour image to Black & White image using java image class

SOURCE CODE:

```
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
public class bw{
	public static void main(String[] args){
		try{
	BufferedImage image=ImageIO.read(new File("Doo.png"));
	BufferedImage black= new
BufferedImage(image.getWidth(),image.getHeight(),BufferedImage.TYPE_BYTE_BINARY
);
	black.getGraphics().drawImage(image,0,0,null);
	ImageIO.write(black,"png",new File("black1.png"));
		}catch(IOException e){
			e.printStackTrace();
		}
	}
}
```

OUTPUT:

Input image:



Output image:

Result:

Thus the above program executed successfully

EX. NO:5	COLOR IMAGE TO NEGATIVE
DATE:14/08/2023	

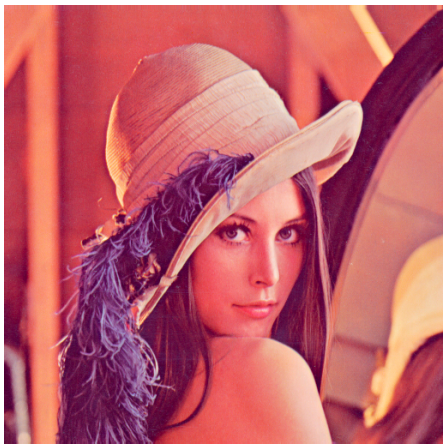
AIM:
To write a java program for convert colour image to Negative image using java image class

SOURCE CODE:

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


public class Negative{
   public static void main(String[] args) {
      try {
         BufferedImage colorImage = ImageIO.read(new File("Doo.png"));
         BufferedImage negativeImage = new
BufferedImage(colorImage.getWidth(),colorImage.getHeight(),
BufferedImage.TYPE_INT_RGB);
         for (int y = 0; y < colorImage.getHeight(); y++) {
            for (int x = 0; x < colorImage.getWidth(); x++) {
               int rgb = colorImage.getRGB(x, y);
               int invertedRGB = ~rgb;
               negativeImage.setRGB(x, y, invertedRGB);
            }
         }
         ImageIO.write(negativeImage, "jpg", new File("output_negative.jpg"));
         System.out.println("Negative image saved to output_negative9.jpg");
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
```

OUTPUT:

Input image:



Output image:

Result:

Thus the above program executed successfully.

EX. NO:6	MERGE A BACKGROUND AND FOREGROUND IMAGE
DATE:21/08/2023	

AIM:
To write a java program for Two image merge using java image class

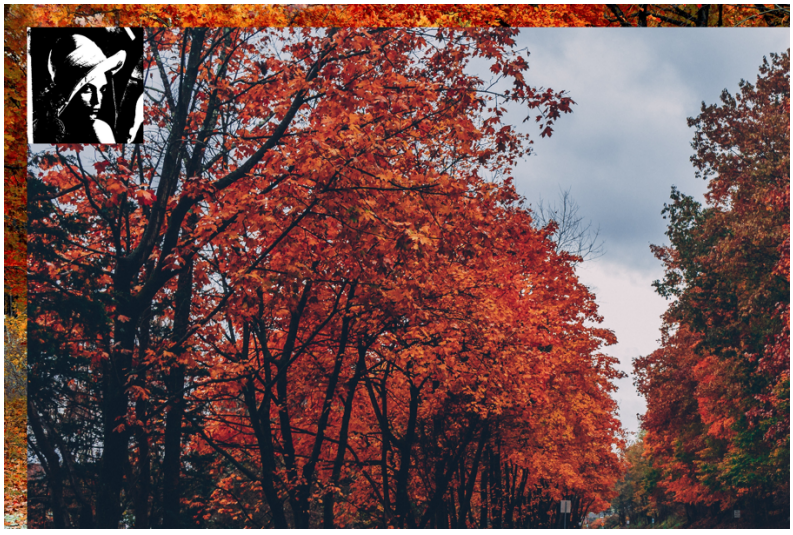
SOURCE CODE:

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

public class Merge{
   public static void main(String[] args) {
      try {
         BufferedImage foregroundImage = ImageIO.read(new File("pexels-craig-adderley-
1563355.jpg"));
         BufferedImage backgroundImage = ImageIO.read(new File("pexels-pixabay-
33109.jpg"));

         int width = backgroundImage.getWidth();
         int height = backgroundImage.getHeight();
         BufferedImage mergedImage = new BufferedImage(width, height,
BufferedImage.TYPE_INT_ARGB);

         Graphics2D g2d = mergedImage.createGraphics();
         g2d.drawImage(backgroundImage, 0, 0, null);

         int foregroundX = 100;
         int foregroundY = 100;

         g2d.drawImage(foregroundImage, foregroundX, foregroundY, null);
         g2d.dispose();

         ImageIO.write(mergedImage, "PNG", new File("out.png"));
         System.out.println("Merged image saved successfully.");
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

OUTPUT :

Input image:

Output image:



Result:
Thus the above program executed successfully.

EX. NO:7	SEGMENTATION OF FOREGROUND FROM BACKGROUND
DATE:28/08/2023	

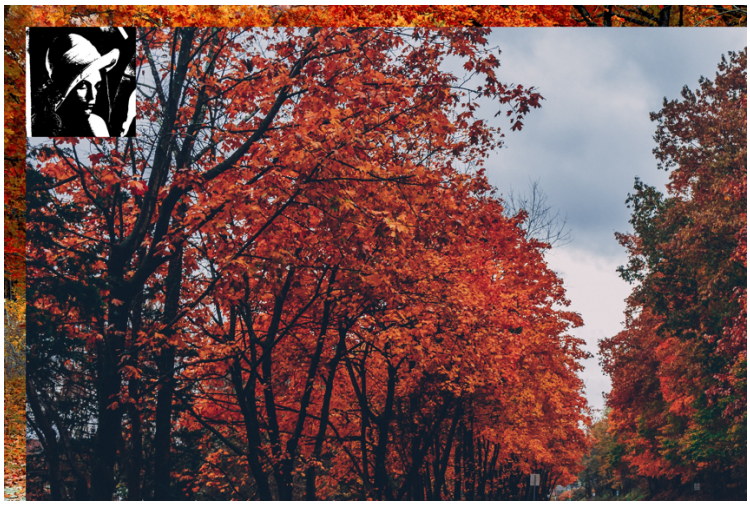
AIM:
To write a java program for segmentation of foreground from background using java image class

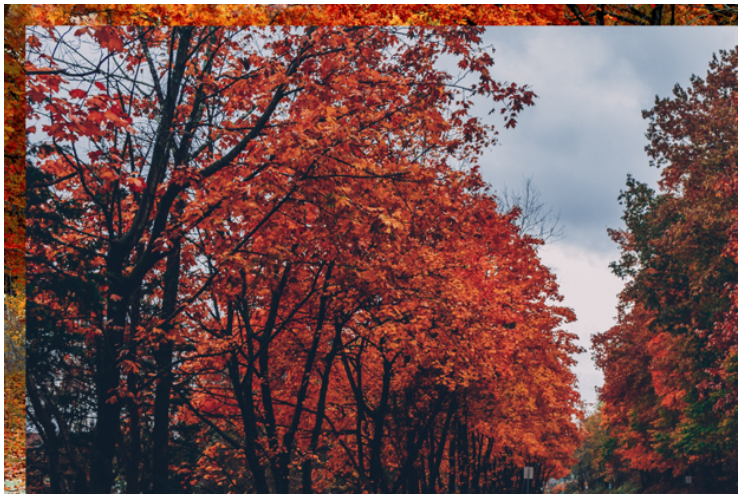
SOURCE CODE:

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

public class seg {
   public static void main(String[] args) {
      try {
         BufferedImage originalImage = ImageIO.read(new File("Lenna_(test_image).png"));
         BufferedImage segmentedImage = segmentForeground(originalImage);
         ImageIO.write(segmentedImage, "png", new File("segmented_output.png"));
         System.out.println("Segmentation complete.");
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
   private static BufferedImage segmentForeground(BufferedImage originalImage) {
      int width = originalImage.getWidth();
      int height = originalImage.getHeight();
      BufferedImage segmentedImage = new BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);

      int threshold = 128; // Adjust this threshold based on your images

      for (int y = 0; y < height; y++)
         for (int x = 0; x < width; x++) {
            int rgb = originalImage.getRGB(x, y);
            int grayValue = rgb & 0xFF;
            int segmentValue = (grayValue > threshold) ? 255 : 0; // Foreground: White,
Background: Black
            int segmentRGB = (segmentValue << 16) | (segmentValue << 8) | segmentValue;
            segmentedImage.setRGB(x, y, segmentRGB);
         }
      return segmentedImage;
   }
}
```

OUTPUT:

Input image:





Output image:

Result:
Thus the above program executed successfully.

EX. NO:8	COLOR IMAGE TO RGB CHANNELS
DATE:04/09/2023	

AIM:

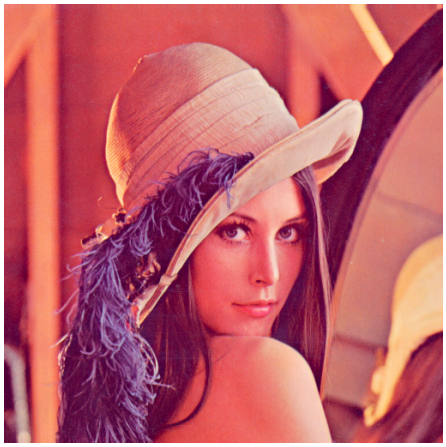
To write a java program for convert color image to RGB channels using java image class

SOURCE CODE:

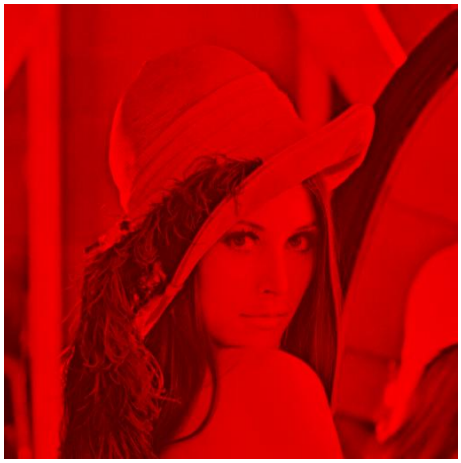
```
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
public class colour {
   public static void main(String[] args) throws IOException {
      BufferedImage colorImage = ImageIO.read(new File("Lenna_(test_image).png"));
      BufferedImage redImage = new BufferedImage(colorImage.getWidth(),
colorImage.getHeight(), BufferedImage.TYPE_INT_RGB);
      BufferedImage greenImage = new BufferedImage(colorImage.getWidth(),
colorImage.getHeight(), BufferedImage.TYPE_INT_RGB);
      BufferedImage blueImage = new BufferedImage(colorImage.getWidth(),
colorImage.getHeight(), BufferedImage.TYPE_INT_RGB);
      for (int y = 0; y < colorImage.getHeight(); y++) {
         for (int x = 0; x < colorImage.getWidth(); x++) {
            int color = colorImage.getRGB(x, y);
            redImage.setRGB(x, y, (color & 0xFF0000) >> 0);
            greenImage.setRGB(x,y, (color & 0xFF0000)>> 8);
            blueImage.setRGB(x,y, (color & 0xFF0000)>> 16);
         }
      }
      ImageIO.write(redImage, "jpg", new File("outpuo8.jpg"));
      ImageIO.write(greenImage, "jpg", new File("outpuo16.jpg"));
      ImageIO.write(blueImage, "jpg", new File("outpu025.jpg"));
   }
}
```

OUTPUT:

Input image:



Output Image:



Result:

Thus the above program executed successfully

EX. NO: 9	LOG TRANSFORMATION OF AN IMAGE
DATE:09/10/2023	

AIM:

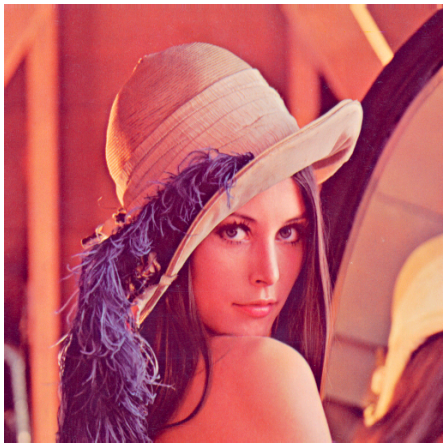
To modify an Image using Log Transformation in Java Programming.

SOURCE CODE:

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

public class log_transform {
   public static void main(String[] args) {
      try {
         BufferedImage inputImage = ImageIO.read(new File("Lenna_(test_image).png"));
         BufferedImage outputImage = new BufferedImage(inputImage.getWidth(),
inputImage.getHeight(), BufferedImage.TYPE_INT_RGB);
         double c = 255 / Math.log(256);

         for (int y = 0; y < inputImage.getHeight(); y++) {
            for (int x = 0; x < inputImage.getWidth(); x++) {
               int rgb = inputImage.getRGB(x, y);
               int r = (rgb >> 16) & 0xFF;
               int g = (rgb >> 8) & 0xFF;
               int b = (rgb >>24)& 0xFF;

               int newR = (int) (c * Math.log(1 + r));
               int newG = (int) (c * Math.log(1 + g));
               int newB = (int) (c * Math.log(1 + b));

               int newRGB = (newR << 16) | (newG << 8) | newB;
               outputImage.setRGB(x, y, newRGB);
            }
         }
         ImageIO.write(outputImage, "jpg", new File("output_log.jpg"));
         System.out.println("Log-transformed image saved to output_log.jpg");
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
```

OUTPUT:

Input image:



Output image:

Result:
Thus the above program executed successfully .

EX. NO: 10	HISTOGRAM EQUALIZATION OF AN IMAGE
DATE:09/10/2023	

AIM:
To write a java program for histogram equalization using java image class

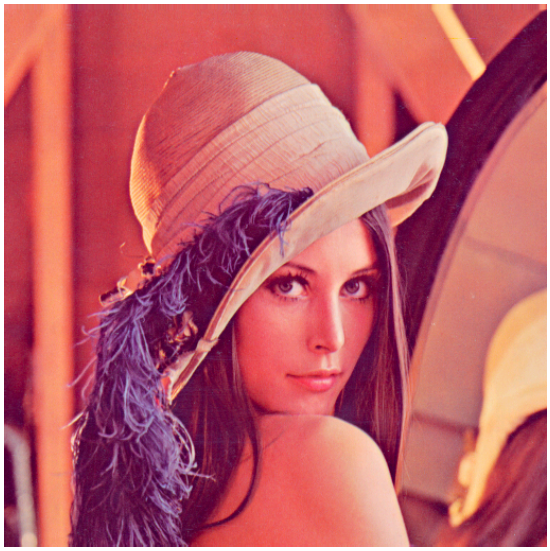
SOURCE CODE:

```
import java.awt.image.BufferedImage;
import java.io.File;
import javax.imageio.ImageIO;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Graphics;

public class hist {
   public static void main(String[] args) {
      try {
         BufferedImage image = ImageIO.read(new File("Doo.png"));
         int[] histogram = new int[256];
         for (int y = 0; y < image.getHeight(); y++)
            for (int x = 0; x < image.getWidth(); x++)
               histogram[(image.getRGB(x, y) >> 16) & 0xFF]++;
         JFrame frame = new JFrame("Image Histogram");
         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         frame.setSize(300, 200);

         JPanel panel = new JPanel() {
            protected void paintComponent(Graphics g) {
               super.paintComponent(g);
               int w = getWidth(), h = getHeight(), max = getMax(histogram);
               for (int i = 0; i < 256; i++) {
                  int barH = (int) ((double) histogram[i] / max * h);
                  g.fillRect(i * w / 256, h - barH, w / 256, barH);
               }
            }
         };
         frame.add(panel);
         frame.setVisible(true);
      } catch (Exception e) {
         System.out.println("Error: " + e.getMessage());
      }
   }
   public static int getMax(int[] array) {
      int max = array[0];
      for (int value : array)
         if (value > max) max = value;
      return max;
   }
```

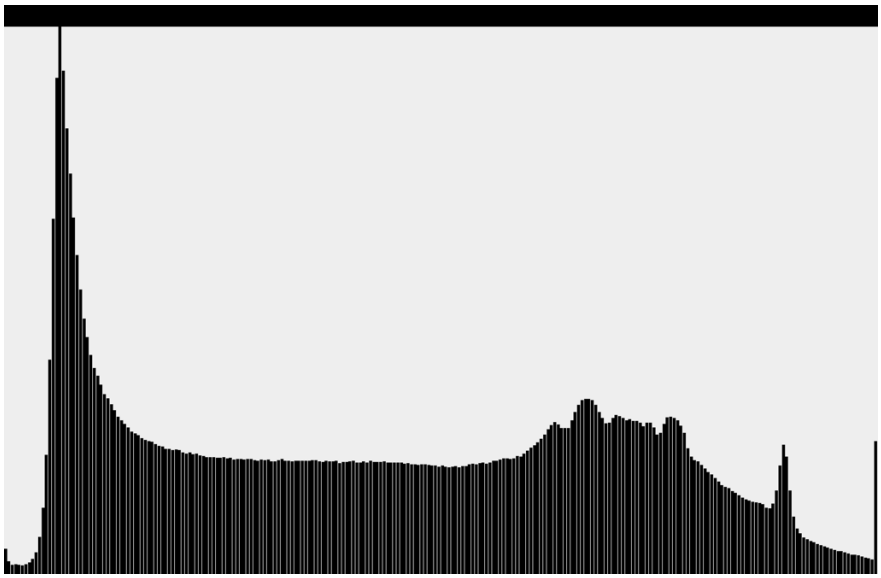
}

OUTPUT:

Input image:



Output image:



Result:

Thus the above program executed successfully .

EX. NO: 11	APPLY MEAN FILTER TO AN IMAGE
DATE:16/10/2023	

AIM:
To write a java program for mean filter using java image class

SOURCE CODE:

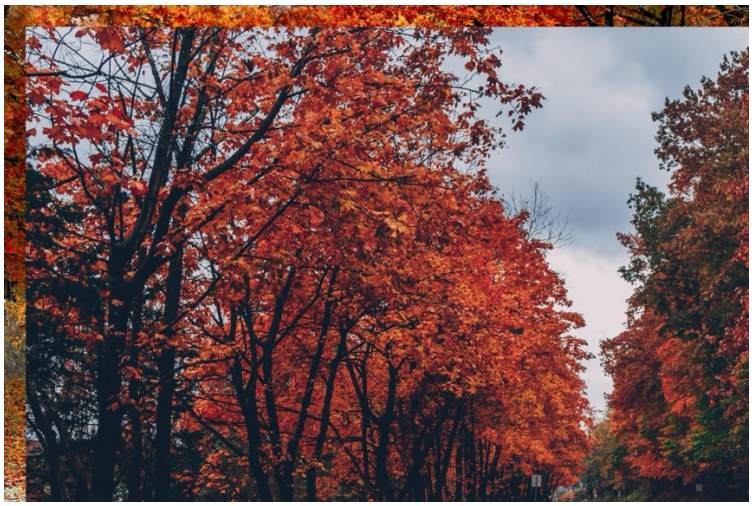
```
import java.awt.image.*;
import java.io.File;
import javax.imageio.ImageIO;

public class mean {
   public static void main(String[] args) {
      try {
         BufferedImage i = ImageIO.read(new File("Lenna_(test_image).png"));
         ImageIO.write(apply(i, 3), "jpg", new File("output78.jpg"));
      } catch (Exception e) {
         System.out.println("Error: " + e.getMessage());
      }
   }

   public static BufferedImage apply(BufferedImage i, int s) {
      int w = i.getWidth(), h = i.getHeight();
      BufferedImage f = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);

      for (int y = 0; y < h; y++)
         for (int x = 0; x < w; x++)
            f.setRGB(x, y, getRGB(i, x, y, s));

      return f;
   }

   public static int getRGB(BufferedImage i, int x, int y, int s) {
      int r = 0, g = 0, b = 0, c = 0;

      for (int j = -s; j <= s; j++)
         for (int k = -s; k <= s; k++) {
            int px = x + k, py = y + j;

            if (px >= 0 && px < i.getWidth() && py >= 0 && py < i.getHeight()) {
               int rgb = i.getRGB(px, py);
               r += (rgb >> 16) & 0xFF;
               g += (rgb >> 8) & 0xFF;
               b += rgb & 0xFF;
               c++;
            }
         }

      return ((r / c) << 16) | ((g / c) << 8) | (b / c);
```

```
    }
}
```

OUTPUT:

Input image:

Output image:



Result:
Thus the above program executed successfully .

EX. NO: 12	APPLY MEDIAN FILTER TO AN IMAGE
DATE:23/10/2023	

AIM:
To write a java program for median filter using java image class

SOURCE CODE:

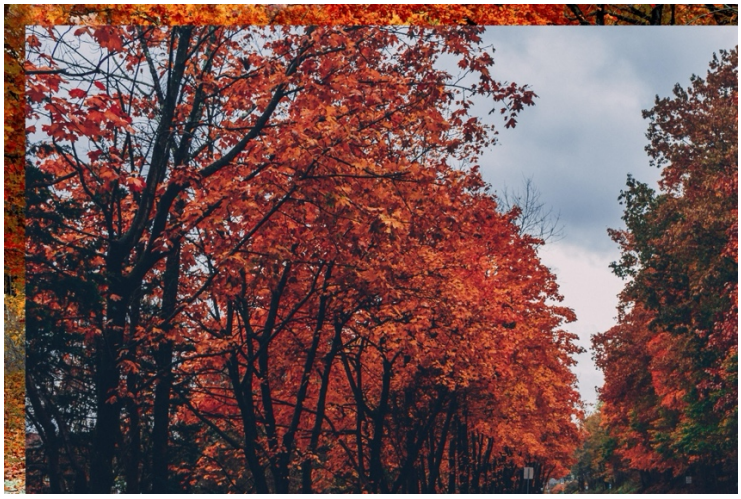
```
import java.awt.image.*;
import java.io.File;
import javax.imageio.ImageIO;
import java.util.Arrays;

public class median {
   public static void main(String[] args) {
      try {
         BufferedImage image = ImageIO.read(new File("Lenna_(test_image).png"));
         ImageIO.write(applyMedianFilter(image, 3), "jpg", new File("output.jpg"));
      } catch (Exception e) {
         System.out.println("Error: " + e.getMessage());
      }
   }
   public static BufferedImage applyMedianFilter(BufferedImage image, int filterSize) {
      int w = image.getWidth(), h = image.getHeight();
      BufferedImage f = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);

      for (int y = 0; y < h; y++)
         for (int x = 0; x < w; x++)
            f.setRGB(x, y, getMedianRGB(image, x, y, filterSize));

      return f;
   }
   public static int getMedianRGB(BufferedImage image, int x, int y, int filterSize) {
      int[] red = new int[filterSize * filterSize];
      int[] green = new int[filterSize * filterSize];
      int[] blue = new int[filterSize * filterSize];
      int i = 0;

      for (int fy = -filterSize; fy <= filterSize; fy++)
         for (int fx = -filterSize; fx <= filterSize; fx++) {
            int ix = x + fx, iy = y + fy;
            if (ix >= 0 && ix < image.getWidth() && iy >= 0 && iy < image.getHeight()) {
               int rgb = image.getRGB(ix, iy);
               red[i] = (rgb >> 16) & 0xFF;
               green[i] = (rgb >> 8) & 0xFF;
               blue[i] = rgb & 0xFF;
               i++;
            }
         }
      Arrays.sort(red);
```

```
        Arrays.sort(green);
        Arrays.sort(blue);

        int ih = i / 2;
        return (red[ih] << 16) | (green[ih] << 8) | blue[ih];
    }
}
```

OUTPUT:

Input image:

Output Image:



Result:
Thus the above program executed successfully .

EX. NO: 13	SHARPENING AN IMAGE USING 1st ORDER DERIVATIVE
DATE:30/10/2023	

AIM:
To write a java program for First Order Derivative using java image class

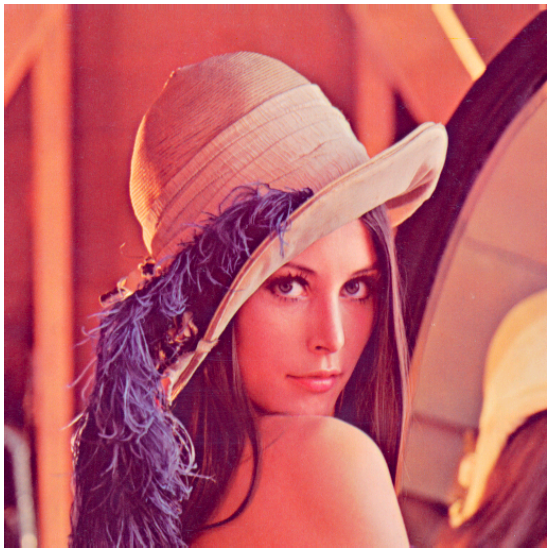
SOURCE CODE:

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

public class firstorder {
   public static void main(String[] args) {
      try {
         BufferedImage image = ImageIO.read(new File("Lenna_(test_image).png"));
         BufferedImage sharpenedImage = applyFirstOrderDerivative(image);
         File output = new File("output9.jpg");
         ImageIO.write(sharpenedImage, "jpg", output);
      } catch (Exception e) {
         System.out.println("Error: " + e.getMessage());
      }
   }
   public static BufferedImage applyFirstOrderDerivative(BufferedImage image) {
      int width = image.getWidth();
      int height = image.getHeight();
      BufferedImage sharpenedImage = new BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);
      int[][] kernel = {{-1, -1, -1}, {-1, 9, -1}, {-1, -1, -1}};

      for (int y = 1; y < height - 1; y++) {
         for (int x = 1; x < width - 1; x++) {
            int[] sum = new int[3];
            for (int ky = 0; ky < 3; ky++) {
               for (int kx = 0; kx < 3; kx++) {
                  int pixel = image.getRGB(x - 1 + kx, y - 1 + ky);
                  sum[0] += ((pixel >> 16) & 0xFF) * kernel[ky][kx];
                  sum[1] += ((pixel >> 8) & 0xFF) * kernel[ky][kx];
                  sum[2] += (pixel & 0xFF) * kernel[ky][kx];
               }
            }
            int newPixel = (clamp(sum[0]) << 16) | (clamp(sum[1]) << 8) | clamp(sum[2]);
            sharpenedImage.setRGB(x, y, newPixel);
         }
      }
      return sharpenedImage;
   }
   public static int clamp(int value) {
      return Math.max(0, Math.min(value, 255));
   }
```

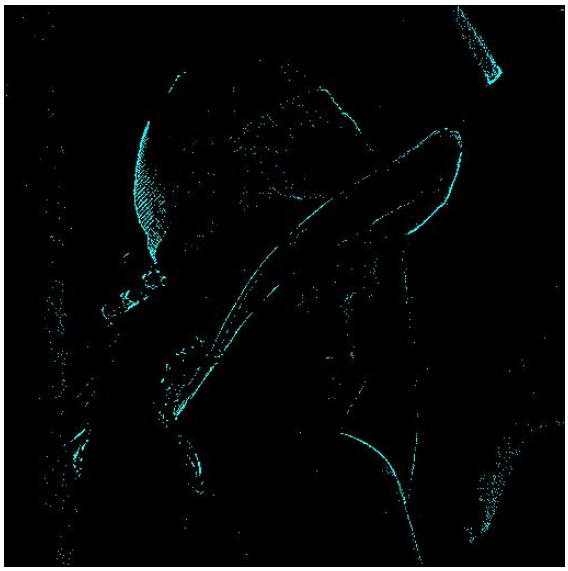
}

OUTPUT:

Input Image:



Output Image:



Result:
Thus the above program executed successfully .

EX. NO: 14	SHARPENING AN IMAGE USING 2nd ORDER DERIVATIVE
DATE:30/10/2023	

AIM:

To write a java program for Second Order Derivative using java image class

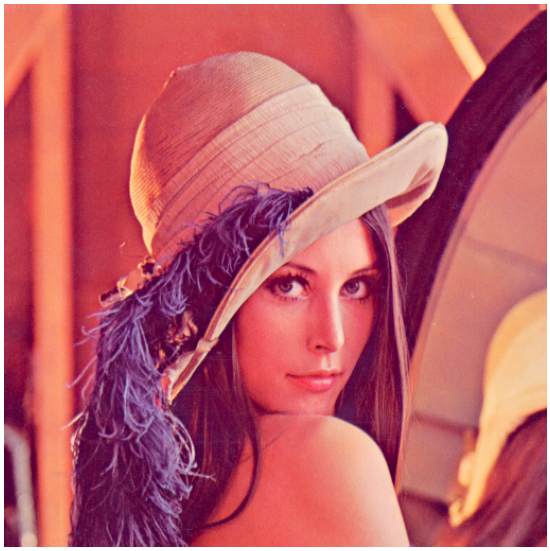
SOURCE CODE:

```
import java.awt.image.BufferedImage;
import java.io.File;
import javax.imageio.ImageIO;
public class secondorder {
   public static void main(String[] args) {
      try {
         BufferedImage image = ImageIO.read(new File("Doo.png"));
         BufferedImage sharpenedImage = applySecondOrderDerivative(image);
         File output = new File("output90.jpg");
         ImageIO.write(sharpenedImage, "jpg", output);
      } catch (Exception e) {
         System.out.println("Error: " + e.getMessage());
      }
   }
   public static BufferedImage applySecondOrderDerivative(BufferedImage image) {
      int width = image.getWidth();
      int height = image.getHeight();
      BufferedImage sharpenedImage = new BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);
      int[][] kernel = {
         {0, -1, 0},
         {-1, 4, -1},
         {0, -1, 0}
      };
      for (int y = 1; y < height - 1; y++) {
         for (int x = 1; x < width - 1; x++) {
            int[] sum = new int[3];
            for (int ky = 0; ky < 3; ky++) {
               for (int kx = 0; kx < 3; kx++) {
                  int pixel = image.getRGB(x - 1 + kx, y - 1 + ky);
                  sum[0] += ((pixel >> 16) & 0xFF) * kernel[ky][kx];
                  sum[1] += ((pixel >> 8) & 0xFF) * kernel[ky][kx];
                  sum[2] += (pixel & 0xFF) * kernel[ky][kx];
               }
            }
            int newPixel = (clamp(sum[0]) << 16) | (clamp(sum[1]) << 8) | clamp(sum[2]);
            sharpenedImage.setRGB(x, y, newPixel);
         }
      }
      return sharpenedImage;
   }
   public static int clamp(int value) {
```

```
        return Math.max(0, Math.min(value, 255));
    }
}
```

OUTPUT:

Input Image:



Output Image:

Result:
Thus the above program executed successfully .

EX. NO: 15	SEGMENTING REGION OF INTEREST IN AN IMAGE
DATE:06/11/2023	

AIM:
To write a java program for Huffman encoding using java image class

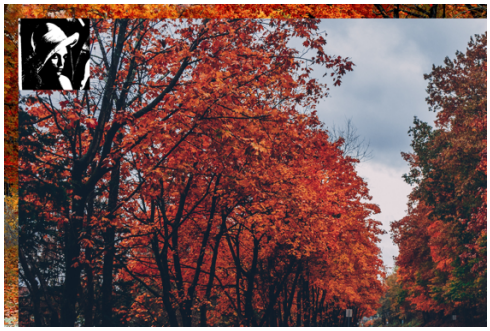
SOURCE CODE:

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

public class sef {
   public static void main(String[] args) {
      try {
         BufferedImage originalImage = ImageIO.read(new File("out.png"));
         BufferedImage backgroundModel = ImageIO.read(new File("Doo.png"));

         int threshold = 25;

         BufferedImage foregroundImage = createForegroundImage(originalImage,
backgroundModel, threshold);

         saveImageToFile(foregroundImage, "result_foreground.png");
      } catch (IOException e) {
         e.printStackTrace();
      }
   }

   private static BufferedImage createForegroundImage(BufferedImage originalImage,
BufferedImage backgroundModel, int threshold) {
      int width = originalImage.getWidth();
      int height = originalImage.getHeight();

      BufferedImage foregroundImage = new BufferedImage(width, height,
BufferedImage.TYPE_INT_ARGB);

      for (int y = 0; y < height; y++)
         for (int x = 0; x < width; x++) {
            int alpha = Math.abs(getGrayValue(originalImage, x, y) -
getGrayValue(backgroundModel, x, y)) > threshold ? 255 : 0;
            foregroundImage.setRGB(x, y, (alpha << 24) | (originalImage.getRGB(x, y) &
0x00FFFFFF));
         }

      return foregroundImage;
   }
```

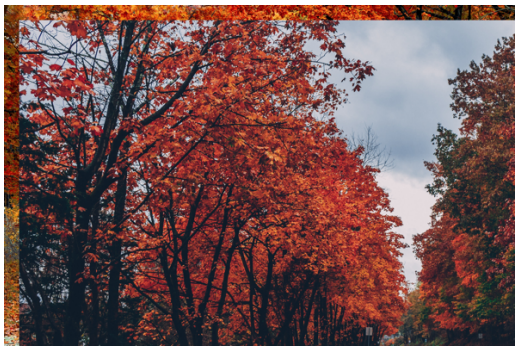
```
    private static int getGrayValue(BufferedImage image, int x, int y) {
        int rgb = image.getRGB(x, y);
        int red = (rgb >> 16) & 0xFF;
        int green = (rgb >> 8) & 0xFF;
        int blue = rgb & 0xFF;
        return (red + green + blue) / 3;
    }

    private static void saveImageToFile(BufferedImage image, String filePath) {
        try {
            ImageIO.write(image, "png", new File(filePath));
            System.out.println("Resulting foreground image saved to: " + filePath);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT:

Input image



Output image:



Result:
Thus the above program executed successfully .

EX. NO: 16	HUFFMAN CODING FOR STRING
DATE:06/11/2023	

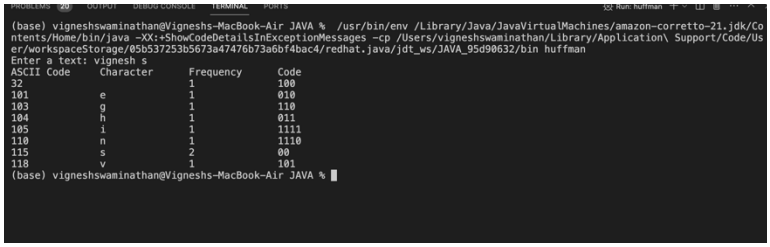
AIM:
To write a java program for Huffman encoding using java image class

SOURCE CODE:

```
import java.util.*;
public class huffman {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      System.out.print("Enter a text: ");
      String text = input.nextLine();
      int[] counts = new int[256];
      for (char c : text.toCharArray()) counts[(int)c]++;
      PriorityQueue<Tree> heap = new PriorityQueue<>();
      for (int i = 0; i < counts.length; i++) if (counts[i] > 0) heap.add(new Tree(counts[i],
(char)i));
      while (heap.size() > 1) heap.add(new Tree(heap.poll(), heap.poll()));
      String[] codes = new String[256];
      assignCode(heap.peek().root, "", codes);
      System.out.printf("%-15s%-15s%-15s%-15s\n", "ASCII Code", "Character",
"Frequency", "Code");
      for (int i = 0; i < codes.length; i++)
         if (counts[i] != 0)
            System.out.printf("%-15d%-15s%-15d%-15s\n", i, (char)i + "", counts[i], codes[i]);
   }
   public static void assignCode(Tree.Node root, String code, String[] codes) {
      if (root.left != null) {
         assignCode(root.left, code + "0", codes);
         assignCode(root.right, code + "1", codes);
      } else codes[(int)root.element] = root.code = code;
   }
   public static class Tree implements Comparable<Tree> {
      Node root;
      public Tree(Tree t1, Tree t2) {
         root = new Node();
         root.left = t1.root;
         root.right = t2.root;
         root.weight = t1.root.weight + t2.root.weight;
      }
      public Tree(int weight, char element) {
         root = new Node(weight, element);
```

```
        }
        public int compareTo(Tree t) {
            return Integer.compare(root.weight, t.root.weight);
        }
        public class Node {
            char element;
            int weight;
            Node left;
            Node right;
            String code = "";
            public Node() {}
            public Node(int weight, char element) {
                this.weight = weight;
                this.element = element;
            }
        }
    }
}
```

OUTPUT:



Result:
Thus the above program executed successfully.

EX. NO: 17	DISCRETE WAVELET TRANSFORMATION FOR STRING
DATE:13/11/2023	

AIM:
To write a java program for discrete wavelet transformation using java image class

SOURCE CODE:

```
import java.util.Arrays;
public class wave {
   public static void main(String[] args) {
      String text = "Hello, World!";
      int[] signal = textToNumericSignal(text);
      int[] dwtResult = discreteWaveletTransform(signal);
      System.out.println("DWT Result: " + Arrays.toString(dwtResult));
   }
   public static int[] textToNumericSignal(String text) {
      int[] signal = new int[text.length()];
      for (int i = 0; i < text.length(); i++) {
         signal[i] = (int) text.charAt(i);
      }
      return signal;
   }
   public static int[] discreteWaveletTransform(int[] signal) {
      int[] dwtResult = Arrays.copyOf(signal, signal.length);
      return dwtResult;
   }
}
```

OUTPUT:

Result:
Thus the above program executed successfully.