DIP_ ASSIGNMENT 1

SUBMISSION DATE: 13-02-2022

NAME OF STUDENT: Vishal Dange

BRANCH: Information Technology

ID: 191080020

AIM:

To Write a program to read and display your own photo on screen. Do not use any inbuilt function to read and write/display images.

THEORY:

Image:

- An image is an array, or a matrix, of square pixels (picture elements) arranged in columns and rows. In a (8-bit) greyscale image each picture element has an assigned intensity that ranges from 0 to 255.
- A grey scale image is what people normally call a black and white image, but the name emphasizes that such an image will also include many shades of grey.
- A normal greyscale image has 8 bit colour depth = 256 greyscales. A “true colour” image has 24 bit colour depth = 8 x 8 x 8 bits = 256 x 256 x 256 colours = ~16 million
colours.

PPM :

- The PPM format is a lowest common denominator color image file format.
- The name "PPM" is an acronym derived from "Portable Pixel Map." Images in this format (or a precursor of it) were once also called "portable pixmaps."
- It should also be noted that files often conform to this format in every respect except the precise semantics of the sample values. These files are useful because of the way PPM is used as an intermediary format. They are informally called PPM files, but to be absolutely precise, you should indicate the variation from true PPM.

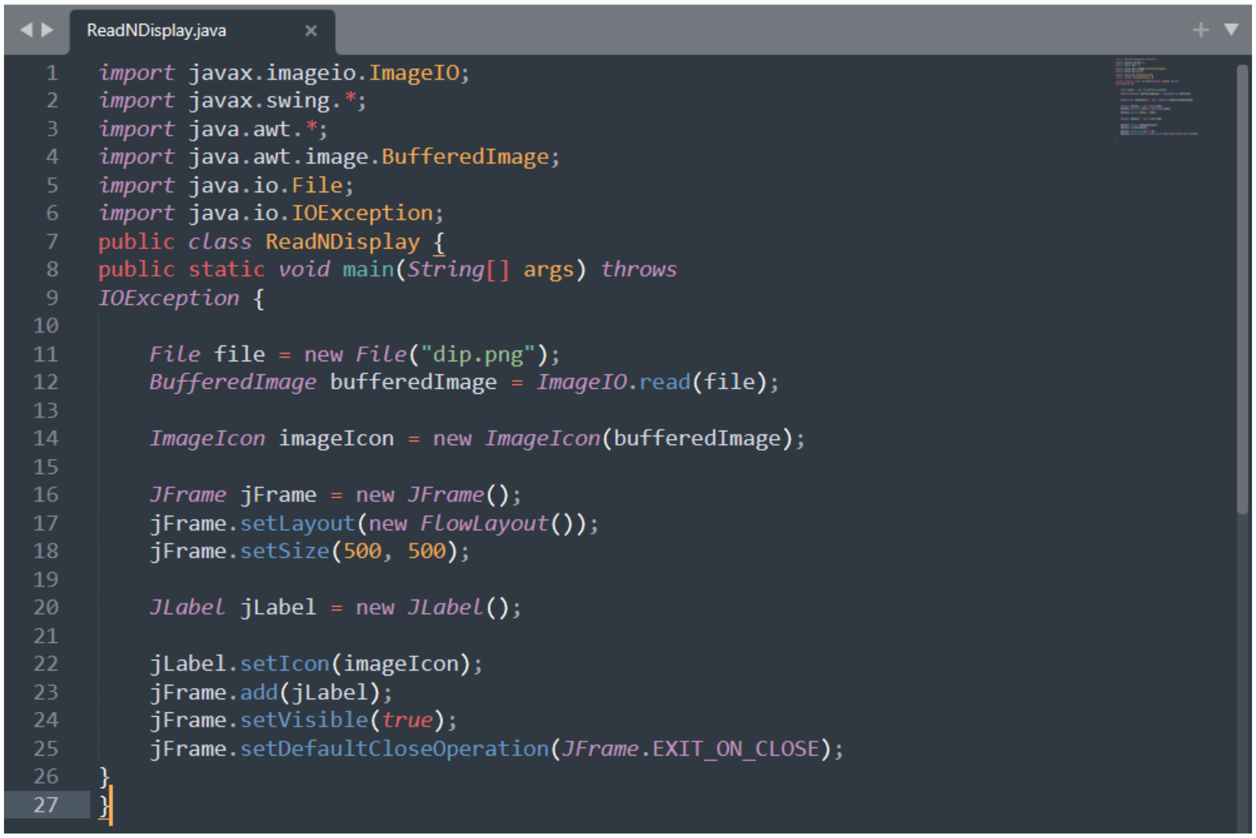
For example, "PPM using the red, green, and blue colors that the scanner in question uses."

- It should be noted that this format is egregiously inefficient. It is highly redundant, while containing a lot of information that the human eye can't even discern. Furthermore, the format allows very little information about the image besides basic color, which means you may have to couple a file in this format with other independent information to get any decent use out of it. However, it is very easy to write and analyze programs to process this format, and that is the point.

Implementation Steps :

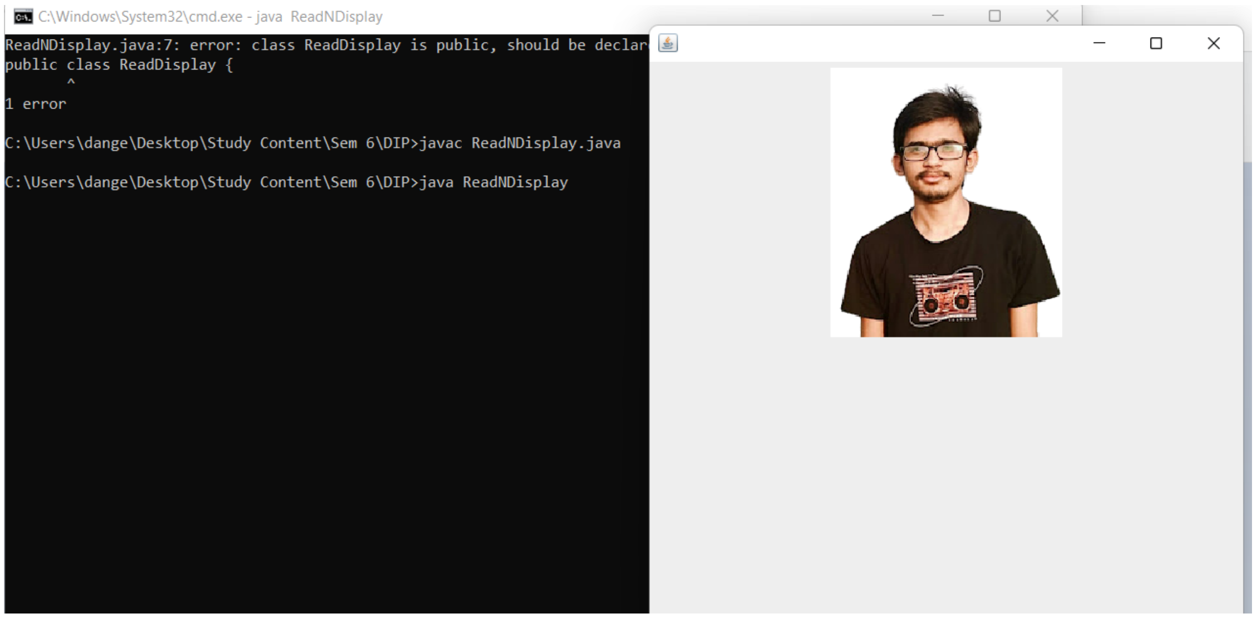
- . To read the image file, we use the File class and pass the path of the image. Next we convert the image to a BufferedImage object using ImageIO.read(). Now we create an icon to be shown in the JLabel.
- To show the label icon, we need a JFrame object with a FlowLayout and a size of 500 x 500. The size can be adjusted according to our needs. Now we create a JLabel object and set its icon using JLabel.setIcon() function. Then we add the jLabel component to jFrame and set the visibility of the frame as true

Code :



```java
import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
public class ReadNDisplay {
public static void main(String[] args) throws
IOException {

    File file = new File("dip.png");
    BufferedImage bufferedImage = ImageIO.read(file);

    ImageIcon imageIcon = new ImageIcon(bufferedImage);

    JFrame jFrame = new JFrame();
    jFrame.setLayout(new FlowLayout());
    jFrame.setSize(500, 500);

    JLabel jLabel = new JLabel();

    jLabel.setIcon(imageIcon);
    jFrame.add(jLabel);
    jFrame.setVisible(true);
    jFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}
```

Output :



```
ReadNDisplay.java:7: error: class ReadDisplay is public, should be declar
public class ReadDisplay {
       ^
1 error

C:\Users\dange\Desktop\Study Content\Sem 6\DIP>javac ReadNDisplay.java

C:\Users\dange\Desktop\Study Content\Sem 6\DIP>java ReadNDisplay
```

CONCLUSION:

- In this Experiment, We wrote a program to read and display your own photo on screen.

*********End of the Assignment*********