Experiment No: 1

Aim: Write a Java program to read an image from a file system and write the image back with a different name.

Theory:

1. **Java.io.File class**

   The File class is an abstract representation of file and directory pathnames.

   1. File(String pathname)

      This method creates a new File instance by converting the given pathname string into an abstract pathname.

2. **Java.awt.image.BufferedImage class**

   Java BufferedImage class is a subclass of Image class. It is used to handle and manipulate the image data. A BufferedImage is made of ColorModel of image data. All BufferedImage objects have an upper left corner coordinate of (0, 0).

   1. BufferedImage(int width, int height, int ImageType)

      The constructor constructs a BufferedImage of one of the predefined image types.

3. **Java.imageio.ImageIO class**

   A class containing static convenience methods for locating ImageReaders and ImageWriters, and performing simple encoding and decoding.

   1. ImageIO.read(File input)

      Returns a BufferedImage as the result of decoding a supplied File with an ImageReader chosen automatically from among those currently registered. The File is wrapped in an ImageInputStream. If no registered ImageReader claims to be able to read the resulting stream, null is returned.

   2. ImageIO.write(RenderedImage im, String FormatName, File output)

      Writes an image using an arbitrary ImageWriter that supports the given format to a File. If there is already a File present, its contents are discarded.

Algorithm:

1. Choose an image you want to read.
2. Read/Store the image into a container
3. Choose a directory to save the image
4. Write the image into the directory chosen by changing its name.

Program:

Java Code:

```
package Exp1;

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

public class Read_Write_File {
public static void main(String[] args) {
int width = 963;
int height = 640;
BufferedImage image = null;
File f = null;
try {
//Read Image
f = new File("./Exp1/wallpaper.jpg");
image = new BufferedImage(width,height,BufferedImage.TYPE_INT_ARGB);
image = ImageIO.read(f);
System.out.println("Reading Complete");
}
catch(IOException e){
System.out.println("Error: "+e);
}
try {
//Write Image
f = new File("./Exp1/wallpaper_output.jpg");
ImageIO.write(image,"jpg",f);
System.out.println("Writing Complete");
}
catch(IOException e){
System.out.println("Error: "+e);
}
}
}
```

Python Code:

```
from PyQt5 import QtCore, QtGui, QtWidgets
import cv2


class Ui_MainWindow(object):
def setupUi(self, MainWindow):
MainWindow.setObjectName("MainWindow")
MainWindow.resize(800, 600)
```

```
self.centralwidget = QtWidgets.QWidget(MainWindow)
self.centralwidget.setObjectName("centralwidget")
self.image = QtWidgets.QLabel(self.centralwidget)
self.image.setGeometry(QtCore.QRect(0, 0, 801, 451))
self.image.setText("")
self.image.setPixmap(QtGui.QPixmap(""))
self.image.setScaledContents(True)
self.image.setObjectName("image")
self.import_image = QtWidgets.QPushButton(self.centralwidget)
self.import_image.setGeometry(QtCore.QRect(0, 450, 261, 81))
self.import_image.setObjectName("import_image")
self.output = QtWidgets.QPushButton(self.centralwidget)
self.output.setGeometry(QtCore.QRect(260, 450, 281, 81))
self.output.setObjectName("output")
self.clear = QtWidgets.QPushButton(self.centralwidget)
self.clear.setGeometry(QtCore.QRect(540, 450, 261, 81))
self.clear.setObjectName("clear")
MainWindow.setCentralWidget(self.centralwidget)
self.menubar = QtWidgets.QMenuBar(MainWindow)
self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 27))
self.menubar.setObjectName("menubar")
MainWindow.setMenuBar(self.menubar)
self.statusbar = QtWidgets.QStatusBar(MainWindow)
self.statusbar.setObjectName("statusbar")
MainWindow.setStatusBar(self.statusbar)

self.retranslateUi(MainWindow)
QtCore.QMetaObject.connectSlotsByName(MainWindow)

#Importing Image
self.import_image.clicked.connect(self.show_image)

#Checking Output Image
self.output.clicked.connect(self.show_output)

#Clearing Image
self.clear.clicked.connect(self.clear_image)

def retranslateUi(self, MainWindow):
_translate = QtCore.QCoreApplication.translate
MainWindow.setWindowTitle(_translate("MainWindow", "Read and Write an Image"))
self.import_image.setText(_translate("MainWindow", "Import"))
self.output.setText(_translate("MainWindow", "Output"))
self.clear.setText(_translate("MainWindow", "Clear"))

def show_image(self):
self.img = cv2.imread('killer.jpg')
```

```
self.img1 = QtGui.QImage(self.img.data, self.img.shape[1], self.img.shape[0],
QtGui.QImage.Format_RGB888).rgbSwapped()
self.image.setPixmap(QtGui.QPixmap.fromImage(self.img1))
def show_output(self):
cv2.imwrite('output.jpg',self.img)
self.image.setPixmap(QtGui.QPixmap("output.jpg"))
def clear_image(self):
self.image.clear()


if __name__ == "__main__":
import sys
app = QtWidgets.QApplication(sys.argv)
MainWindow = QtWidgets.QMainWindow()
ui = Ui_MainWindow()
ui.setupUi(MainWindow)
MainWindow.show()
sys.exit(app.exec_())
```

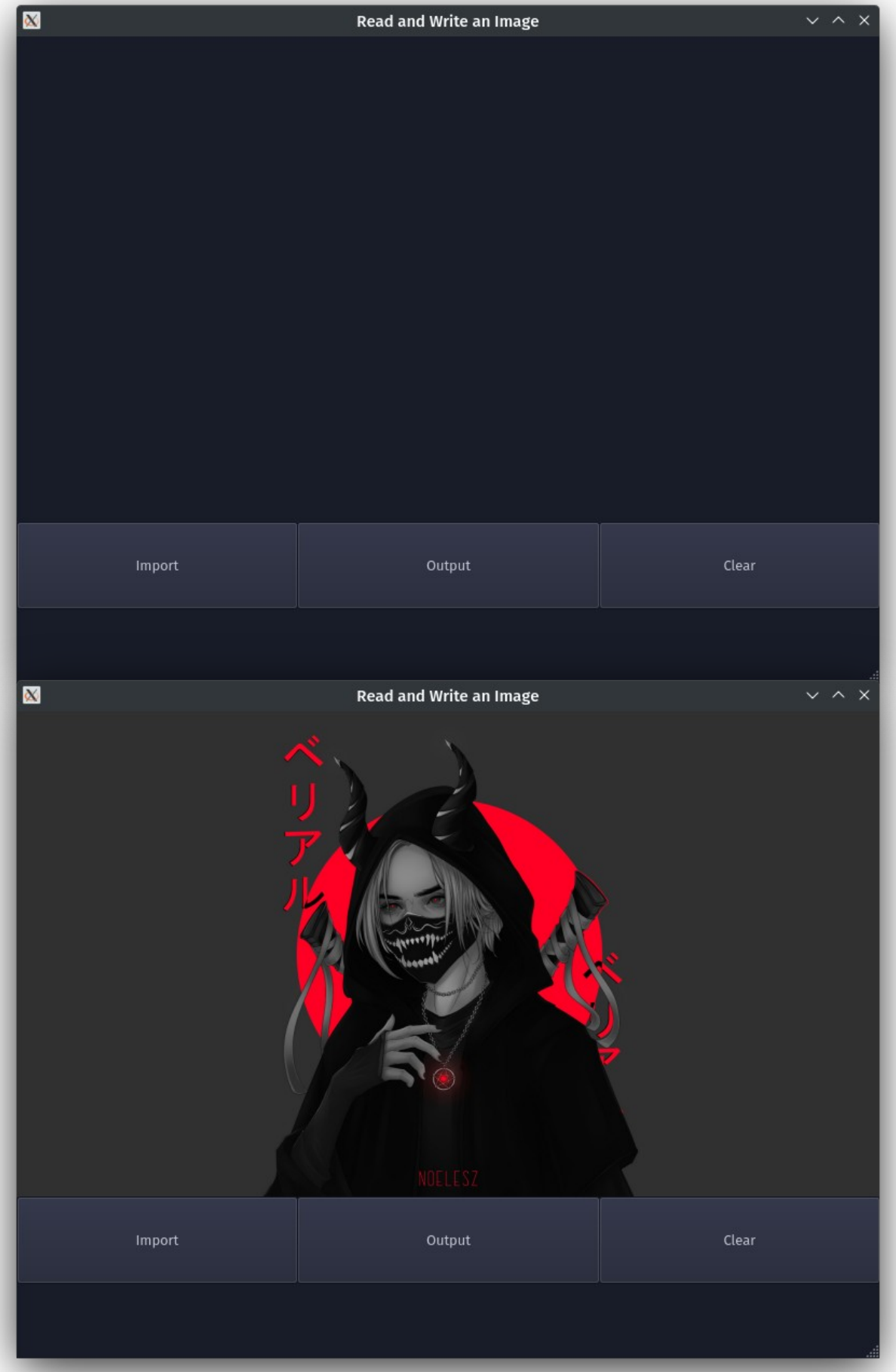
Output:

Java Output:

Reading complete.

Writing complete.

Python GUI Output:


Read and Write an Image
Import
Output
Clear
Read and Write an Image
NOELESZ
Import
Output
Clear

Conclusion: Program to read and write an image was written and executed.

Deepraj Bhosale
Batch-A
181105016