Experiment No: 2

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

Theory:

1. **Java.awt.Color class**

   The Color class is a part of the Java Abstract Window Toolkit(AWT) package. The Color class creates color by using the given RGBA values where RGBA stands for RED, GREEN, BLUE, ALPHA or using HSB values where HSB stands for HUE, SATURATION, BRIcomponents. The value for individual components RGBA ranges from 0 to 255 or 0.0 to 0.1. The value of alpha determines the opacity of the color, where 0 or 0.0 stands fully transparent and 255 or 1.0 stands opaque.

   1. Color(int r, int g, int b) :
      Creates a opaque color with specified RGB components(values are in range 0 – 255)
   2. getGreen() / getBlue() / getRed() :
      Returns the green / blue / red component respectively in the range 0-255 in the default sRGB space.

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.
   2. getRGB(int x, int y)
      It returns an integer pixel in the default RGB color model (TYPE_INT_ARGB) and default sRGB colorspace.
   3. setRGB (int x, int y, int rgb)
      Sets the pixel at position (x, y) with the rgb value provided

Algorithm:

1. Read the image and store it into some container to perform operations on it
2. Get rgb value of the pixel
3. Calculate new RGB values as shown :
   1. R = R * 0.30
   2. G = G * 0.59
   3. B = B * 0.11
4. Save the new RGB value in the pixel

5. Repeat step 2 - 4 for each pixel of the image
6. Choose a directory to store the new image
7. Store the negative image into the selected directory

Program:

Java Code:

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

public class Grayscale_Image {
public static void main(String[] args) {
int width = 1920;
int height = 1080;
BufferedImage image = null;
File f = null;
File f1 = null;
try {
//Read Image
f = new File("/run/media/deeprajb/HDD/Important
Photos/Wallpapers/wallpaperflare.com_wallpaper3.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
f1 = new File("./java_grayscale_output.jpg");
for (int i = 0; i < height; i++) {
for (int j = 0; j < width; j++) {
Color c = new Color(image.getRGB(j, i));
int red = (int)(c.getRed() * 0.299);
int green = (int)(c.getGreen() * 0.587);
int blue = (int)(c.getBlue() * 0.114);
Color newColor = new Color(red + green + blue, red + green + blue, red + green + blue);
image.setRGB(j, i, newColor.getRGB());
}
}
```

```
ImageIO.write(image,"jpg",f1);
System.out.println("Writing Complete");
}
catch(IOException e){
System.out.println("Error: "+e);
}
}
}
```

Python Code:

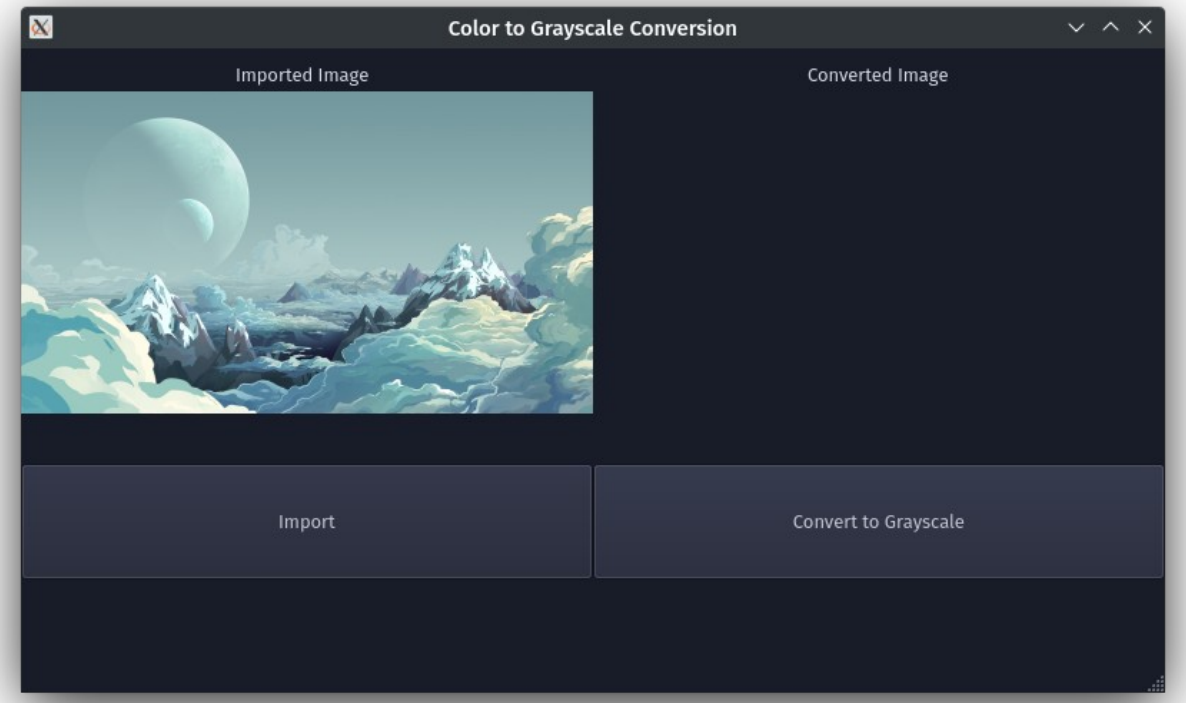
```
from PyQt5 import QtCore, QtGui, QtWidgets
import cv2


class Ui_MainWindow(object):
def setupUi(self, MainWindow):
MainWindow.setObjectName("MainWindow")
MainWindow.resize(800, 450)
self.centralwidget = QtWidgets.QWidget(MainWindow)
self.centralwidget.setObjectName("centralwidget")
self.imageinput = QtWidgets.QLabel(self.centralwidget)
self.imageinput.setGeometry(QtCore.QRect(0, 30, 400, 225))
self.imageinput.setText("")
self.imageinput.setPixmap(QtGui.QPixmap(""))
self.imageinput.setScaledContents(True)
self.imageinput.setObjectName("imageinput")
self.import_image = QtWidgets.QPushButton(self.centralwidget)
self.import_image.setGeometry(QtCore.QRect(0, 290, 400, 81))
self.import_image.setObjectName("import_image")
self.output = QtWidgets.QPushButton(self.centralwidget)
self.output.setGeometry(QtCore.QRect(400, 290, 400, 81))
self.output.setObjectName("output")
self.imageoutput = QtWidgets.QLabel(self.centralwidget)
self.imageoutput.setGeometry(QtCore.QRect(401, 30, 400, 225))
self.imageoutput.setText("")
self.imageoutput.setPixmap(QtGui.QPixmap(""))
self.imageoutput.setScaledContents(True)
self.imageoutput.setObjectName("imageoutput")
self.label = QtWidgets.QLabel(self.centralwidget)
self.label.setGeometry(QtCore.QRect(150, 10, 101, 17))
self.label.setObjectName("label")
self.label_2 = QtWidgets.QLabel(self.centralwidget)
self.label_2.setGeometry(QtCore.QRect(550, 10, 101, 17))
self.label_2.setObjectName("label_2")
MainWindow.setCentralWidget(self.centralwidget)
```

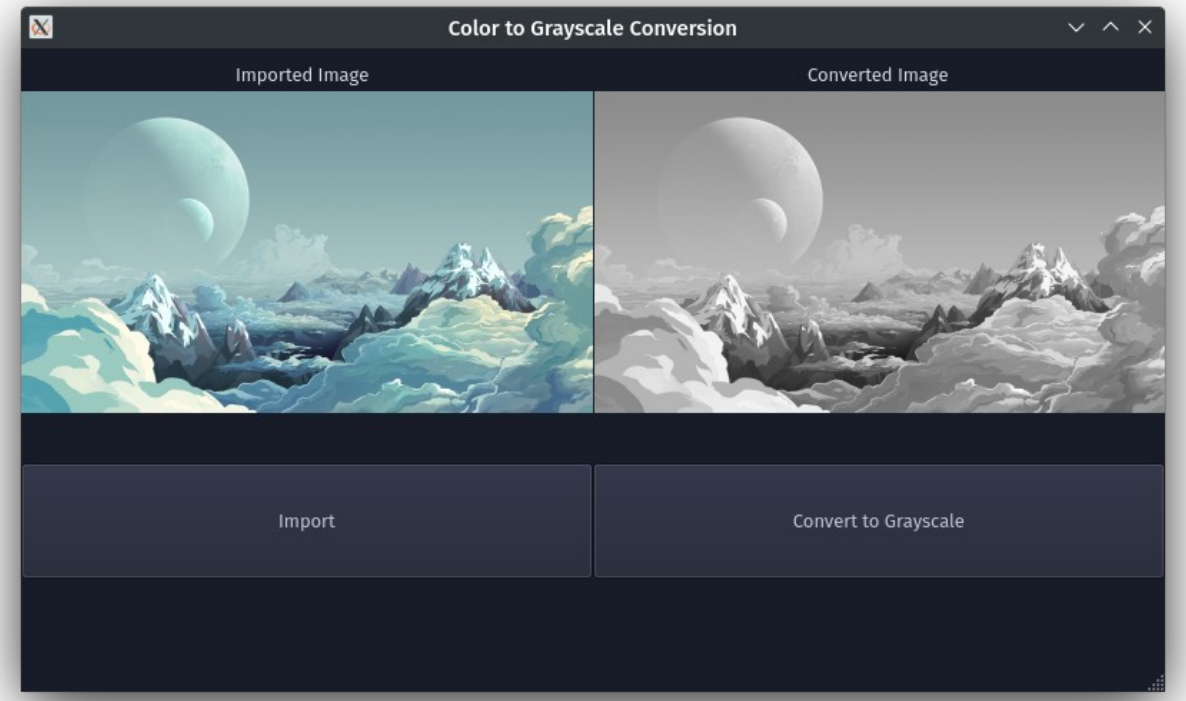
```
self.menubar = QtWidgets.QMenuBar(MainWindow)
self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 27))
self.menubar.setObjectName("menubar")
MainWindow.setMenuBar(self.menubar)
self.statusbar = QtWidgets.QStatusBar(MainWindow)
self.statusbar.setObjectName("statusbar")
MainWindow.setStatusBar(self.statusbar)

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.convert_to_grayscale)

def retranslateUi(self, MainWindow):
_translate = QtCore.QCoreApplication.translate
MainWindow.setWindowTitle(_translate("MainWindow", "Color to Grayscale Conversion"))
self.import_image.setText(_translate("MainWindow", "Import"))
self.output.setText(_translate("MainWindow", "Convert to Grayscale"))
self.label.setText(_translate("MainWindow", "Imported Image"))
self.label_2.setText(_translate("MainWindow", "Converted Image"))

def show_image(self):
file_filter = 'Image File (*.jpg *.png)'
fname = QtWidgets.QFileDialog.getOpenFileName(parent=self.centralwidget,
caption='Select an Image',
directory="/run/media/deeprajb/HDD/Important Photos/Wallpapers",
filter=file_filter)
self.img = cv2.imread(fname[0])
self.img1 = QtGui.QImage(self.img.data, self.img.shape[1], self.img.shape[0],
QtGui.QImage.Format_RGB888).rgbSwapped()
self.imageinput.setPixmap(QtGui.QPixmap.fromImage(self.img1))
def convert_to_grayscale(self):
(row, col) = self.img.shape[0:2]
for i in range(row):
for j in range(col):
# Find the average of the RGB pixel values
self.img[i, j] = sum(self.img[i, j]) * 0.33
cv2.imwrite('grayscaled_output.jpg',self.img)
self.imageoutput.setPixmap(QtGui.QPixmap("grayscaled_output.jpg"))


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:

Python GUI Output:





Conclusion: Program to read an image and convert it into grayscale was written and executed successfully.

Deepraj Bhosale
Batch-A
181105016