

**IMAGE PROCESSING**

**In Partial Fulfillment**

**Of Requirements for the CIS 043 Course**

**Summer 2015**

**By**

**Namrata Joglekar**

**Eric Shen**

**Table of Contents**

Chapter 1: Introduction

1.1 Problem Statement………………………………….........................03

1.2 Techniques covered… ...………………………...………………….03

Chapter 2: Project design and implementation

2.1 Source Code…………………………..………………………...…...04

2.2 Code explanation………………………………...…..………………35

2.3 Filter description……………………………………………………..37

Chapter 3: Results for Filter

3.1 Gray Scale ……………………………………………………...……40

3.2 Add Noise………………………………….……..…………...……..40

3.3 Blur…………………………………………………………………...41

3.4 Color……………………...………………………………..………....41

3.5 Double………………………………………………………………..42

3.6 Skew………………………………………………………………….43

3.7 Stretch………………………………………………………………...43

3.8 3D…………………………………………………………………….44

3.9 Flip……………………………………………………………………44

3.10 Glare……………………………………………..………………….45

3.11 Shadow ………………………………………..…………………. 45

3.12 Chrome ………………………………………..…………………. . 46

3.13 Mirror…………………………………………..…………………... 47

3.14 Water……………………………….…………..…………………... 47

Chapter 4: Conclusion……………………………………………………………...48

Chapter 5: Appendix………………………………………………..........................49

Chapter 6: References……………………………………………………………...50

**Chapter 1: Introduction**

* 1. **Problem Statement**
* To create an application that will allow the user to select an image from his/her computer and apply filters to change the image.
* The application will create a pop up from Eclipse, and will contain the menus ‘File’ and ‘Filters’. The ‘File’ menu will have the menu item ‘Open’ which will open an interface that will allow the user to select an image from his or her computer and then open that image in another pop up. The ‘Filters’ menu will have a menu item for each filter which will allow the user to apply filters onto the selected image**.**
  1. **Abstract:**

This application allows the user to select an image saved on his or her computer or on a removable disk and edit that image by applying filters. Filters can overlap on top of each other with enables to user to apply several filters on the same image for further effect. The user has a wide variety of filters, some cause shifts in the RGB value of the pixels, some adjust the RGB values and some add to the image in some way. Some filters are further customizable with user input, allowing the user to select the color they want the image scaled to or the extent of the effect of a filter. The application then allows the user to save his or her edited image.

* 1. **Techniques Covered**

Course Concept

* + - Object creation
    - For loops
    - Try catch
    - If else statements
    - GUI
    - Classes and Methods
    - Constructors
    - Java API
    - Two dimensional Arrays
    - Interface
    - Inheritance
    - Exception Handling

New concepts

* + - JMenu
    - JMenuItems
    - JMenuBar
    - BufferedImage
    - Image Filters

**Chapter 2: Project Design and Implementation**

**2.1 Source Code**

The code contains the following files:

1. /src/ImageLab/BlueGlare.java
2. /src/ImageLab/Blues.java
3. /src/ImageLab/BlueShadow.java
4. /src/ImageLab/Blur.java
5. /src/ImageLab/Cyan.java
6. /src/ImageLab/Double.java
7. /src/ImageLab/FileSelect.java
8. /src/ImageLab/Filters.java
9. /src/ImageLab/Flip.java
10. /src/ImageLab/Glare.java
11. /src/ImageLab/GrayScale.java
12. /src/ImageLab/GreenGlare.java
13. /src/ImageLab/Greens.java
14. /src/ImageLab/GreenShadow.java
15. /src/ImageLab/Handler.java
16. **/src/ImageLab/ImageLab.java**
17. /src/ImageLab/Menu.java
18. /src/ImageLab/Noise.java
19. /src/ImageLab/Noise1.java
20. /src/ImageLab/PurpleScale.java
21. /src/ImageLab/RedGlare.java
22. /src/ImageLab/Reds.java
23. /src/ImageLab/RedShadow.java
24. /src/ImageLab/Scaling.java
25. /src/ImageLab/Shadow.java
26. /src/ImageLab/Skew.java
27. /src/ImageLab/Stretch.java
28. /src/ImageLab/Test.java
29. /src/ImageLab/ThreeD.java
30. /src/ImageLab/Yellow.java

The code is as follows:

**package** ImageLab;

//import java.awt.\*;

**import** java.awt.BorderLayout;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferByte;

**import** javax.swing.\*;

**public** **class** Filters **extends** JMenuItem {

**public** **static** JMenuItem *fil*;

**protected** **static** ImageIcon *icon*;

**public** Filters(String str){

**super**(str);

}

**public** **static** **void** setIcon(ImageIcon i){

*icon* = i;

}

**public** **static** ImageIcon icon(){

**return** *icon*;

}

**public** Filters(String str,String des)

{

*fil* = **new** JMenuItem(str);

*fil*.getAccessibleContext().setAccessibleDescription(

des);

*fil*.getAccessibleContext().setAccessibleName(

str);

Menu.*filters*.add(*fil*);

}

**public** **static** **int**[][] getRGB(BufferedImage img){

**int** [][] arr = **new** **int**[img.getWidth()][img.getHeight()];

**for**(**int** i = 0; i < img.getWidth(); i++)

{

**for**(**int** j = 0; j < img.getHeight(); j++)

{

arr[i][j] = img.getRGB(i,j);

}

}

**return** arr;

}

**public** **static** **void** makeImage(BufferedImage img){

*icon* = **new** ImageIcon(img);

JLabel L1 = **new** JLabel("",*icon*,JLabel.***CENTER***);

JFrame frame1 = **new** JFrame("Image");

frame1.setDefaultCloseOperation(JFrame.***DISPOSE\_ON\_CLOSE***);

frame1.getContentPane().add(L1, BorderLayout.***WEST***);

frame1.pack();

frame1.setVisible(**true**);

}

}

package ImageLab;

import java.io.File;

import javax.swing.JFileChooser;

public class FileSelect {

private static File file;

public void filePath(){

JFileChooser jfc = new JFileChooser();

jfc.showDialog(null,"Select");

jfc.setVisible(true);

File filename = jfc.getSelectedFile();

file = filename;

}

public static File getFile(){

return file;

}

}

package ImageLab;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowEvent;

public class Handler implements ActionListener{

public void actionPerformed(ActionEvent e){

Scaling.removeFrame();

if(e.getSource() == Scaling.getB1())

Reds.act();

else if(e.getSource()==Scaling.getB2())

Greens.act();

else if(e.getSource() == Scaling.getB3())

Blues.act();

else if(e.getSource() == Scaling.getB4())

PurpleScale.act();

else if(e.getSource() == Scaling.getB5())

Yellow.act();

else if(e.getSource() == Scaling.getB6())

Cyan.act();

}

}

**package** ImageLab;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** java.awt.image.IndexColorModel;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenuItem;

**import** com.jhlabs.image.BoxBlurFilter;

**public** **class** Blur **extends** Filters

{

**public** Blur(String str,String des)

{

**super**(str,des);

*fil*.addActionListener(**new** handler());

}

**private** **class** handler **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**if** (img != **null**)

{

BoxBlurFilter bf = **new** BoxBlurFilter();

bf.setRadius(10);

BufferedImage dest = img; //copy constructor

bf.filter(img, dest);

img=dest;

}

*makeImage*(img);

}

}

}

**package** ImageLab;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** java.awt.image.IndexColorModel;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenuItem;

**import** com.jhlabs.image.BoxBlurFilter;

**import** com.jhlabs.image.FlipFilter;

**public** **class** Flip **extends** Filters

{

**public** Flip(String str,String des)

{

**super**(str,des);

*fil*.addActionListener(**new** handler());

}

**private** **class** handler **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**if** (img != **null**)

{

FlipFilter ff = **new** FlipFilter();

ff.setOperation(FlipFilter.***FLIP\_HV***);

BufferedImage dest = **new** BufferedImage(img.getHeight(), img.getWidth(), BufferedImage.***TYPE\_INT\_RGB***);

ff.filter(img, dest);

img=dest;

}

*makeImage*(img);

}

}

}

**package** ImageLab;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.security.SecureRandom;

**import** javax.swing.JOptionPane;

**public** **class** Noise **extends** Filters {

**public** Noise(String str,String des)

{

**super**(str,des);

*fil*.addActionListener(**new** handler());

}

**private** **class** handler **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

**int** num = *userInput*();

SecureRandom ran = **new** SecureRandom();

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**for**(**int** i = 0; i <img.getWidth();i++){

**for**(**int** j = 0;j < img.getHeight();j++){

**int** random = ran.nextInt(101);

**if**(random<=num){

Color col = **new** Color(ran.nextInt(256),ran.nextInt(256),ran.nextInt(256));

img.setRGB(i,j,col.getRGB());

}**else**{

Color c = **new** Color(img.getRGB(i,j), **true**);

Color c1 = **new** Color(c.getRed(),c.getGreen(),c.getBlue());

img.setRGB(i,j,c1.getRGB());

}

}

}

*makeImage*(img);

}

}

**public** **static** **int** userInput(){

**int** num = 50;

**try**{

String answer = JOptionPane.*showInputDialog*("Enter desired amount of noise");

num = Integer.*parseInt*(answer);

}**catch**(NumberFormatException e){

*userInput*();

}

**return** num;

}

}

**package** ImageLab;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** java.awt.image.IndexColorModel;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenuItem;

**import** com.jhlabs.image.BoxBlurFilter;

**import** com.jhlabs.image.NoiseFilter;

**public** **class** Noise1 **extends** Filters

{

**public** Noise1(String str,String des)

{

**super**(str,des);

*fil*.addActionListener(**new** handler());

}

**private** **class** handler **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**if** (img != **null**)

{

NoiseFilter nf = **new** NoiseFilter();

BufferedImage dest = img; //copy constructor

nf.filter(img, dest);

img=dest;

}

*makeImage*(img);

}

}

}

**package** ImageLab;

**import** com.jhlabs.image.\*;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** java.awt.image.IndexColorModel;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenuItem;

**import** com.jhlabs.image.BoxBlurFilter;

**import** com.jhlabs.image.ChromeFilter;

**public** **class** Chrome **extends** Filters

{

**public** Chrome(String str,String des)

{

**super**(str,des);

*fil*.addActionListener(**new** handler());

}

**private** **class** handler **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**if** (img != **null**)

{

ChromeFilter cf = **new** ChromeFilter();

BufferedImage dest = img; //copy constructor

cf.filter(img, dest);

img=dest;

}

*makeImage*(img);

}

}

}

**package** ImageLab;

**import** com.jhlabs.image.\*;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** java.awt.image.IndexColorModel;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenuItem;

**import** com.jhlabs.image.BoxBlurFilter;

**import** com.jhlabs.image.ChromeFilter;

**public** **class** Water **extends** Filters

{

**public** Water(String str,String des)

{

**super**(str,des);

*fil*.addActionListener(**new** handler());

}

**private** **class** handler **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**if** (img != **null**)

{

WaterFilter w = **new** WaterFilter();

BufferedImage dest = img; //copy constructor

w.filter(img, dest);

img=dest;

}

*makeImage*(img);

}

}

}

**package** ImageLab;

**import** com.jhlabs.image.\*;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** java.awt.image.IndexColorModel;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenuItem;

**import** com.jhlabs.image.BoxBlurFilter;

**import** com.jhlabs.image.ChromeFilter;

**import** com.jhlabs.image.MirrorFilter;

**public** **class** Mirror **extends** Filters

{

**public** Mirror(String str,String des)

{

**super**(str,des);

*fil*.addActionListener(**new** handler());

}

**private** **class** handler **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**if** (img != **null**)

{

MirrorFilter m = **new** MirrorFilter();

BufferedImage dest = img; //copy constructor

m.filter(img, dest);

img=dest;

}

*makeImage*(img);

}

}

}

**package** ImageLab;

**import** java.awt.GridLayout;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.WindowEvent;

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JPanel;

**public** **class** Glare **extends** Filters {

**private** **static** JButton *B1*;

**private** **static** JButton *B2*;

**private** **static** JButton *B3*;

**private** **static** JFrame *frame*;

**public** Glare(String str,String des)

{

**super**(str,des);

*fil*.addActionListener(**new** handler());

}

**private** **class** handler **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

**if**(e.getSource()==*B1*){

*removeFrame*();

RedGlare.*act*();

}**else** **if**(e.getSource()==*B2*){

*removeFrame*();

GreenGlare.*act*();

}**else** **if**(e.getSource() == *B3*){

*removeFrame*();

BlueGlare.*act*();

}**else**{

*frame* = **new** JFrame("Color Select");

*frame*.setDefaultCloseOperation(JFrame.***DISPOSE\_ON\_CLOSE***);

GridLayout grid = **new** GridLayout(3,1);

*B1* = **new** JButton("Red Glare");

*B1*.addActionListener(**new** handler());

*B2* = **new** JButton("Green Glare");

*B2*.addActionListener(**new** handler());

*B3* = **new** JButton("Blue Glare");

*B3*.addActionListener(**new** handler());

JPanel pan = **new** JPanel(grid);

pan.add(*B1*);

pan.add(*B2*);

pan.add(*B3*);

JLabel L1 = **new** JLabel("Select the color you want the glare to be");

*frame*.setLayout(grid);

*frame*.getContentPane().add(L1);

*frame*.add(pan);

*frame*.pack();

*frame*.setVisible(**true**);

}

}

}

**public** **static** **void** removeFrame(){

*frame*.dispatchEvent(**new** WindowEvent(*frame*, WindowEvent.***WINDOW\_CLOSING***));

}

**public** **static** JButton getB1(){

**return** *B1*;

}

**public** **static** JButton getB2(){

**return** *B2*;

}

**public** **static** JButton getB3(){

**return** *B3*;

}

}

**package** ImageLab;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** java.awt.image.IndexColorModel;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenuItem;

**import** javax.swing.JOptionPane;

**public** **class** GreenGlare **extends** Filters {

**public** GreenGlare(String str,String des)

{

**super**(str,des);

}

**public** **static** **void** act(){

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**for**(**int** i = 0; i <img.getWidth();i++){

**for**(**int** j = 0;j < img.getHeight();j++){

Color c = **new** Color(img.getRGB(i,j), **true**);

Color cG = c;

**if**(i>= 1)

cG = **new** Color(img.getRGB(i-1,j),**true**);

Color c1 = **new** Color(c.getRed(),cG.getGreen(),c.getBlue());

img.setRGB(i,j,c1.getRGB());

}

}

*makeImage*(img);

}

}

**package** ImageLab;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** java.awt.image.IndexColorModel;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenuItem;

**import** javax.swing.JOptionPane;

**public** **class** RedGlare **extends** Filters {

**public** RedGlare(String str,String des)

{

**super**(str,des);

}

**public** **static** **void** act(){

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**for**(**int** i = 0; i <img.getWidth();i++){

**for**(**int** j = 0;j < img.getHeight();j++){

Color c = **new** Color(img.getRGB(i,j), **true**);

Color cR = c;

**if**(i>= 1)

cR = **new** Color(img.getRGB(i-1,j),**true**);

Color c1 = **new** Color(cR.getRed(),c.getGreen(),c.getBlue());

img.setRGB(i,j,c1.getRGB());

}

}

*makeImage*(img);

}

}

**package** ImageLab;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** java.awt.image.IndexColorModel;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenuItem;

**import** javax.swing.JOptionPane;

**public** **class** BlueGlare **extends** Filters {

**public** BlueGlare(String str,String des)

{

**super**(str,des);

}

**public** **static** **void** act(){

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**for**(**int** i = 0; i <img.getWidth();i++){

**for**(**int** j = 0;j < img.getHeight();j++){

Color c = **new** Color(img.getRGB(i,j), **true**);

Color cB = c;

**if**(i>= 1)

cB = **new** Color(img.getRGB(i-1,j),**true**);

Color c1 = **new** Color(c.getRed(),c.getGreen(),cB.getBlue());

img.setRGB(i,j,c1.getRGB());

}

}

*makeImage*(img);

}

}

**package** ImageLab;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** java.awt.image.IndexColorModel;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenuItem;

**public** **class** PurpleScale **extends** Filters {

**public** PurpleScale(String str,String des)

{

**super**(str,des);

}

**public** **static** **void** act(){

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**for**(**int** i = 0; i <img.getWidth();i++){

**for**(**int** j = 0;j < img.getHeight();j++){

Color c = **new** Color(img.getRGB(i,j), **true**);

**int** avg = (c.getRed()+c.getBlue())/2;

Color c1 = **new** Color(avg,0,avg);

img.setRGB(i,j,c1.getRGB());

}

}

*makeImage*(img);

}

}

**package** ImageLab;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** java.awt.image.IndexColorModel;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenuItem;

**public** **class** GrayScale **extends** Filters {

**public** GrayScale(String str,String des)

{

**super**(str,des);

*fil*.addActionListener(**new** handler());

}

**private** **class** handler **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**for**(**int** i = 0; i <img.getWidth();i++){

**for**(**int** j = 0;j < img.getHeight();j++){

Color c = **new** Color(img.getRGB(i,j), **true**);

**int** num = average(c.getRed(),c.getGreen(),c.getBlue());

Color c1 = **new** Color(num,num,num);

img.setRGB(i,j,c1.getRGB());

}

}

*makeImage*(img);

}

}

**private** **int** max(**int** i,**int** j, **int** k){

**int** max = 1;

**if**(j>i)

max=j;

**if**(k>max)

max=k;

**return** max;

}

**private** **int** average(**int** i,**int** j, **int** k){

**return** (i+j+k)/3;

}

**private** **int** min(**int** i,**int** j, **int** k){

**int** min = 1;

**if**(j<i)

min=j;

**if**(k<min)

min=k;

**return** min;

}

}

**package** ImageLab;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** java.awt.image.IndexColorModel;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenuItem;

**public** **class** PurpleScale **extends** Filters {

**public** PurpleScale(String str,String des)

{

**super**(str,des);

}

**public** **static** **void** act(){

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**for**(**int** i = 0; i <img.getWidth();i++){

**for**(**int** j = 0;j < img.getHeight();j++){

Color c = **new** Color(img.getRGB(i,j), **true**);

**int** avg = (c.getRed()+c.getBlue())/2;

Color c1 = **new** Color(avg,0,avg);

img.setRGB(i,j,c1.getRGB());

}

}

*makeImage*(img);

}

}

**package** ImageLab;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** java.awt.image.IndexColorModel;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenuItem;

**public** **class** Yellow **extends** Filters {

**public** Yellow(String str,String des)

{

**super**(str,des);

}

**public** **static** **void** act(){

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**for**(**int** i = 0; i <img.getWidth();i++){

**for**(**int** j = 0;j < img.getHeight();j++){

Color c = **new** Color(img.getRGB(i,j), **true**);

**int** avg = (c.getRed()+c.getGreen())/2;

Color c1 = **new** Color(avg,avg,0);

img.setRGB(i,j,c1.getRGB());

}

}

*makeImage*(img);

}

}

**package** ImageLab;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** java.awt.image.IndexColorModel;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenuItem;

**public** **class** Blues **extends** Filters {

**public** Blues(String str,String des)

{

**super**(str,des);

}

**public** **static** **void** act(){

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**for**(**int** i = 0; i <img.getWidth();i++){

**for**(**int** j = 0;j < img.getHeight();j++){

Color c = **new** Color(img.getRGB(i,j), **true**);

Color c1 = **new** Color(0,0,c.getBlue());

img.setRGB(i,j,c1.getRGB());

}

}

*makeImage*(img);

}

}

**package** ImageLab;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** java.awt.image.IndexColorModel;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenuItem;

**public** **class** Cyan **extends** Filters {

**public** Cyan(String str,String des)

{

**super**(str,des);

}

**public** **static** **void** act(){

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**for**(**int** i = 0; i <img.getWidth();i++){

**for**(**int** j = 0;j < img.getHeight();j++){

Color c = **new** Color(img.getRGB(i,j), **true**);

**int** avg = (c.getGreen()+c.getBlue())/2;

Color c1 = **new** Color(0,avg,avg);

img.setRGB(i,j,c1.getRGB());

}

}

*makeImage*(img);

}

}

**package** ImageLab;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** java.awt.image.IndexColorModel;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenuItem;

**public** **class** Reds **extends** Filters {

**public** Reds(String str,String des)

{

**super**(str,des);

}

**public** **static** **void** act(){

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_ARGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**for**(**int** i = 0; i <img.getWidth();i++){

**for**(**int** j = 0;j < img.getHeight();j++){

Color c = **new** Color(img.getRGB(i,j), **true**);

Color c1 = **new** Color(c.getRed(),0,0);

img.setRGB(i,j,c1.getRGB());

}

}

*makeImage*(img);

}

}

**package** ImageLab;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**public** **class** ThreeD **extends** Filters {

**public** ThreeD(String str,String des)

{

**super**(str,des);

*fil*.addActionListener(**new** handler());

}

**private** **class** handler **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

//icon = new ImageIcon(""+FileSelect.getFile(),"");

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

BufferedImage im = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

**for**(**int** i = 0; i <img.getWidth();i++){

**for**(**int** j = 0;j < img.getHeight();j++){

Color c = **new** Color(img.getRGB(i,j), **true**);

Color cR= c;

Color cB= c;

**if**(i<img.getWidth()/20\*19)

{

cB = **new** Color(img.getRGB(i+img.getWidth()/20, j),**true**);

//System.out.println("B"+(i+img.getWidth()/20));

}

**if**(i>img.getWidth()/20)

{

cR = **new** Color(img.getRGB(i-img.getWidth()/20,j),**true**);

//System.out.println("r"+(i-img.getWidth()/20));

}

Color c1 = **new** Color(cR.getRed(),c.getGreen(),cB.getBlue());

im.setRGB(i,j,c1.getRGB());

}

/\*

System.out.print("\*\*\*\*\*\*\*\*\* " + i + " " + img.getWidth()/20);

System.out.println("\*\*\*\*\*\*\*\*\* " + (i-img.getWidth()/20));\*/

}

*makeImage*(im);

}

}

}

**package** ImageLab;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**public** **class** Stretch **extends** Filters {

**public** Stretch (String str,String des)

{

**super**(str,des);

*fil*.addActionListener(**new** handler());

}

**private** **class** handler **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

//icon = new ImageIcon(""+FileSelect.getFile(),"");

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

BufferedImage im = **new** BufferedImage(

*icon*.getIconWidth()\*2,

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

**for**(**int** i = 0; i <img.getWidth()\*2;i++){

**for**(**int** j = 0;j < img.getHeight();j++){

Color c = **new** Color(img.getRGB(i/2,j), **true**);

Color c1 = **new** Color(c.getRed(),c.getGreen(),c.getBlue());

im.setRGB(i,j,c1.getRGB());

im.setRGB(i+1,j,c1.getRGB());

}

i++;

}

*makeImage*(im);

}

}

}

**package** ImageLab;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.image.BufferedImage;

**import** java.security.SecureRandom;

**import** javax.swing.JOptionPane;

**public** **class** Skew **extends** Filters {

**public** Skew(String str,String des)

{

**super**(str,des);

*fil*.addActionListener(**new** handler());

}

**private** **class** handler **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

**int** num = *userInput*();

**double** rad = num\*Math.***PI***/180;

BufferedImage img = **new** BufferedImage(

*icon*.getIconWidth(),

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

*icon*.paintIcon(**null**, g, 0,0);

g.dispose();

**double** scale = Math.*tan*(rad);

BufferedImage im = **new** BufferedImage(

*icon*.getIconWidth()+(**int**)(*icon*.getIconHeight()\*scale)+1,

*icon*.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

**for**(**int** i = 0; i <img.getWidth();i++){

**for**(**int** j = 0;j < img.getHeight();j++){

Color c = **new** Color(img.getRGB(i,j), **true**);

Color c1 = **new** Color(c.getRed(),c.getGreen(),c.getBlue());

im.setRGB(i+(**int**)((img.getHeight()-j)\*scale),j,c1.getRGB());

}

}

*makeImage*(im);

}

}

**public** **static** **int** userInput(){

**int** num = 0;

**try**{

String answer = JOptionPane.*showInputDialog*("Enter angle of bottom left corner");

num = Integer.*parseInt*(answer);

}**catch**(NumberFormatException e){

*userInput*();

}

**if**(num>85)

num = 85;

**if**(num<5)

num = 5;

**return** num;

}

}

**package** ImageLab;

**import** java.awt.GridLayout;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.WindowEvent;

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JPanel;

**public** **class** Shadow **extends** Filters {

**private** **static** JButton *B1*;

**private** **static** JButton *B2*;

**private** **static** JButton *B3*;

**private** **static** JFrame *frame*;

**public** Shadow(String str,String des)

{

**super**(str,des);

*fil*.addActionListener(**new** handler());

}

**private** **class** handler **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

**if**(e.getSource()==*B1*){

*removeFrame*();

RedShadow.*act*();

}**else** **if**(e.getSource()==*B2*){

*removeFrame*();

GreenShadow.*act*();

}**else** **if**(e.getSource() == *B3*){

*removeFrame*();

BlueShadow.*act*();

}**else**{

*frame* = **new** JFrame("Color Select");

*frame*.setDefaultCloseOperation(JFrame.***DISPOSE\_ON\_CLOSE***);

GridLayout grid = **new** GridLayout(3,1);

*B1* = **new** JButton("Red Shadow");

*B1*.addActionListener(**new** handler());

*B2* = **new** JButton("Green Shadow");

*B2*.addActionListener(**new** handler());

*B3* = **new** JButton("Blue Shadow");

*B3*.addActionListener(**new** handler());

JPanel pan = **new** JPanel(grid);

pan.add(*B1*);

pan.add(*B2*);

pan.add(*B3*);

JLabel L1 = **new** JLabel("Select the color you want the shadow to be");

*frame*.setLayout(grid);

*frame*.getContentPane().add(L1);

*frame*.add(pan);

*frame*.pack();

*frame*.setVisible(**true**);

}

}

}

**public** **static** **void** removeFrame(){

*frame*.dispatchEvent(**new** WindowEvent(*frame*, WindowEvent.***WINDOW\_CLOSING***));

}

**public** **static** JButton getB1(){

**return** *B1*;

}

**public** **static** JButton getB2(){

**return** *B2*;

}

**public** **static** JButton getB3(){

**return** *B3*;

}

}

package ImageLab;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowEvent;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

public class Scaling extends Filters {

private static JButton B1;

private static JButton B2;

private static JButton B3;

private static JButton B4;

private static JButton B5;

private static JButton B6;

private static JFrame frame;

public Scaling(String str,String des)

{

super(str,des);

fil.addActionListener(new handler());

}

private class handler implements ActionListener{

public void actionPerformed(ActionEvent e){

frame = new JFrame("Color Select");

frame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

GridLayout grid = new GridLayout(6,1);

B1 = new JButton("Red");

B1.addActionListener(new Handler());

B2 = new JButton("Green");

B2.addActionListener(new Handler());

B3 = new JButton("Blue");

B3.addActionListener(new Handler());

B4 = new JButton("Purple");

B4.addActionListener(new Handler());

B5 = new JButton("Yellow");

B5.addActionListener(new Handler());

B6 = new JButton("Cyan");

B6.addActionListener(new Handler());

JPanel pan = new JPanel(grid);

pan.add(B1);

pan.add(B2);

pan.add(B3);

pan.add(B4);

pan.add(B5);

pan.add(B6);

JLabel L1 = new JLabel("Select the color you want your image to be scaled to");

frame.setLayout(grid);

frame.getContentPane().add(L1);

frame.add(pan);

frame.pack();

frame.setVisible(true);

}

}

public static void removeFrame(){

frame.dispatchEvent(new WindowEvent(frame, WindowEvent.WINDOW\_CLOSING));

}

public static JButton getB1(){

return B1;

}

public static JButton getB2(){

return B2;

}

public static JButton getB3(){

return B3;

}

public static JButton getB4(){

return B4;

}

public static JButton getB5(){

return B5;

}

public static JButton getB6(){

return B6;

}

}

**package** ImageLab;

**import** com.jhlabs.image.MirrorFilter;

**import** com.jhlabs.image.\*;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Dimension;

**import** java.awt.Graphics;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.KeyEvent;

**import** java.awt.event.WindowEvent;

**import** java.awt.image.BufferedImage;

**import** java.io.File;

**import** java.io.IOException;

**import** javax.imageio.ImageIO;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenu;

**import** javax.swing.JMenuBar;

**import** javax.swing.JMenuItem;

**import** javax.swing.KeyStroke;

**import** com.jhlabs.image.CrystallizeFilter;

**import** com.jhlabs.image.DissolveFilter;

**public** **class** Menu{

JFrame frame;

**static** JMenu *filters*;

**private** JMenuItem open;

**private** JMenuItem save;

**public** Menu() {

frame = **new** JFrame("Image Processor");

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

JLabel whiteLabel = **new** JLabel();

whiteLabel.setOpaque(**true**);

whiteLabel.setBackground(**new** Color(255, 255, 255));

whiteLabel.setPreferredSize(**new** Dimension(300, 280));

JMenuBar MenuBar = **new** JMenuBar();

MenuBar.setOpaque(**false**);

MenuBar.setPreferredSize(**new** Dimension(300, 30));

frame.getContentPane().add(whiteLabel, BorderLayout.***CENTER***);

frame.setJMenuBar(MenuBar);

// creates the file menu

JMenu file = **new** JMenu("File");

file.setMnemonic(KeyEvent.***VK\_A***);

file.getAccessibleContext().setAccessibleDescription(

"Open an image to edit");

MenuBar.add(file);

// creates the open menu item in the file menu

open = **new** JMenuItem("Open",

KeyEvent.***VK\_T***);

open.setAccelerator(KeyStroke.*getKeyStroke*(

KeyEvent.***VK\_1***, ActionEvent.***ALT\_MASK***));

open.getAccessibleContext().setAccessibleDescription(

"This opens an image");

open.addActionListener(**new** handler());

file.add(open);

save = **new** JMenuItem("Save",

KeyEvent.***VK\_2***);

save.setAccelerator(KeyStroke.*getKeyStroke*(

KeyEvent.***VK\_2***, ActionEvent.***ALT\_MASK***));

save.getAccessibleContext().setAccessibleDescription(

"This saves an image");

save.addActionListener(**new** handler());

file.add(save);

// creates the filters menu

*filters* = **new** JMenu("Filters");

*filters*.getAccessibleContext().setAccessibleDescription(

"Apply a filter to your image");

MenuBar.add(*filters*);

//Size the frame.

frame.pack();

//Show it.

frame.setVisible(**true**);

}

**private** **class** handler **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e) {

**if**(e.getSource()==save){

*save*(Filters.*icon*(),"JPG");

}**else** **if**(e.getSource()==open){

FileSelect file = **new** FileSelect();

file.filePath();

Filters.*setIcon*(**new** ImageIcon(""+file.*getFile*(),""));

JLabel whiteLabel = **new** JLabel("",Filters.*icon*(),JLabel.***CENTER***);

JFrame frame1 = **new** JFrame("Image");

frame1.setDefaultCloseOperation(JFrame.***DISPOSE\_ON\_CLOSE***);

frame1.getContentPane().add(whiteLabel, BorderLayout.***CENTER***);

frame1.pack();

frame1.setVisible(**true**);

**if**(whiteLabel.getHeight()==0)

frame1.dispatchEvent(**new** WindowEvent(frame, WindowEvent.***WINDOW\_CLOSING***));

}

}

}

**private** **static** **void** save(ImageIcon image, String ext) {

String fileName = "Edited Image";

BufferedImage img = **new** BufferedImage(

image.getIconWidth(),

image.getIconHeight(),

BufferedImage.***TYPE\_INT\_RGB***);

Graphics g = img.createGraphics();

// paint the Icon to the BufferedImage.

image.paintIcon(**null**, g, 0,0);

g.dispose();

File file = **new** File(fileName + "." + ext);

**try** {

ImageIO.*write*(img, ext, file); // ignore returned boolean

} **catch**(IOException e) {

System.***out***.println("Write error for " + file.getPath() +

": " + e.getMessage());

}

}

**public** **void** addFilters()

{ Blur blr = **new** Blur("Blur","This filter blurs the image");

Flip f = **new** Flip ("Flip","This filter will flip the image");

Noise n = **new** Noise("Add Noise","This adds a user defined amount of noise");

Noise1 n1 = **new** Noise1("Noise","This filter will add noise to the image");

Chrome cf = **new** Chrome (" Chrome", " This filter simulate colors");

Mirror m = **new** Mirror (" Mirror", " This filter mirrors an image");

Water w = **new** Water (" Water", " This filter produces a water ripple effects on an image");

// Circle ci = new Circle (" Mirror", " This filter mirrors an image");

Double d = **new** Double("Double","This filter doubles the image");

GrayScale gs = **new** GrayScale("Gray Scale","This filter converts an image to black and white");

Scaling c = **new** Scaling("Color Scale","This filter scales the image to an user defined color");

ThreeD three = **new** ThreeD("3D","Adds a 3D effect");

Stretch s = **new** Stretch("Stretch","This stretches the image");

Shadow shadow = **new** Shadow("Shadow","Adds shadows to the image");

Glare g = **new** Glare("Glare","This adds glare to an image");

Skew sk = **new** Skew("Skew","Skews an image a user definied amount");

}

}

**package** ImageLab;

**public** **class** ImageLab {

**public** **static** **void** main(String[] args) {

Menu menu = **new** Menu();

menu.addFilters();

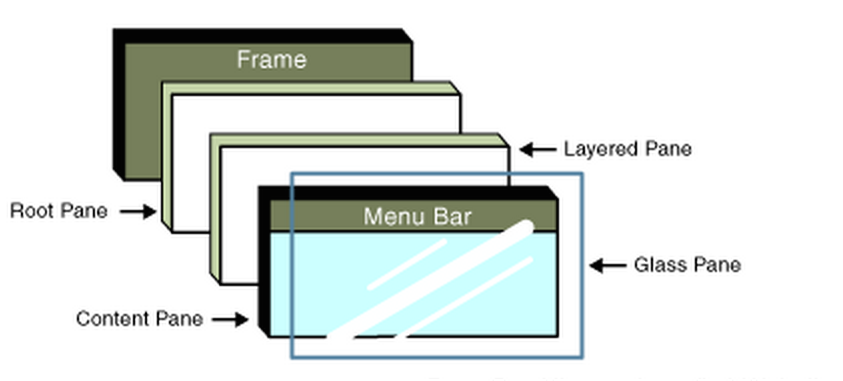
}

}

**2.2 Code Explanation**

1. ImageLab class contains the main function.
2. In the main function, we create an object of the menu Class using a constructor of Menu with no arguments. ( Menu menu = new Menu(); )

Menu class is the heart of the program. It creates the GUI of the program using by creating “GUI objects” and displaying them on a Jframe. JFrame is described as a 'Top-Level' container and is split into 'panes' - like panes of glass - that are positioned on top of one another. There is a Root Pane, a Layered Pane, a Content Pane and a Glass Pane as shown below. Each pane carries out a different function. The Content Pane is where we put all the text, buttons and other graphics we need for our program. We can access the Content Pane directly which we will use. It is possible to also use a JPanel as a layer on top of the Content Pane. [Ref: http://www.macs.hw.ac.uk/cs/java-swing-guidebook/?name=JPanel&page=1]



In our Class Menu, we create a JFrame object as,

JFrame frame = new JFrame("Image Processor");

To this frame, we add JMenuBar and JMenu (for File, and Filters), and JLabel objects. In the JMenu objects, we add JMenuItems such as Open and Save). The JMenuItem class represents the actual item in a menu. To each of the JMenuItems’s addActionListener function, we pass an object of a class that implements the interface ActionListener. When the Save menuitem is clicked, the “actionPerformed” function is called back, which will handle the Save, or Open functions. Save will write a file called EditedImage.JPG and open will create an object of type FileSelect, which we use to “select” the file to be opened, and this is stored as an ImageIcon object " protected static ImageIcon icon;”. Inside this ImageIcon object, the image that was just loaded, is stored in the form of a BufferedImage object (See Ref for description of the BufferedImage class).

1. The Main Function then calls menu.addFilters() function. The addFilters method creates an object of every filter needed, and thus calls the necessary constructors for each of those Filter classes. E.g.
   1. Double d = new Double("Double","This filter doubles the image");

GrayScale gs = new GrayScale("Gray Scale","This filter converts an image to black and white");

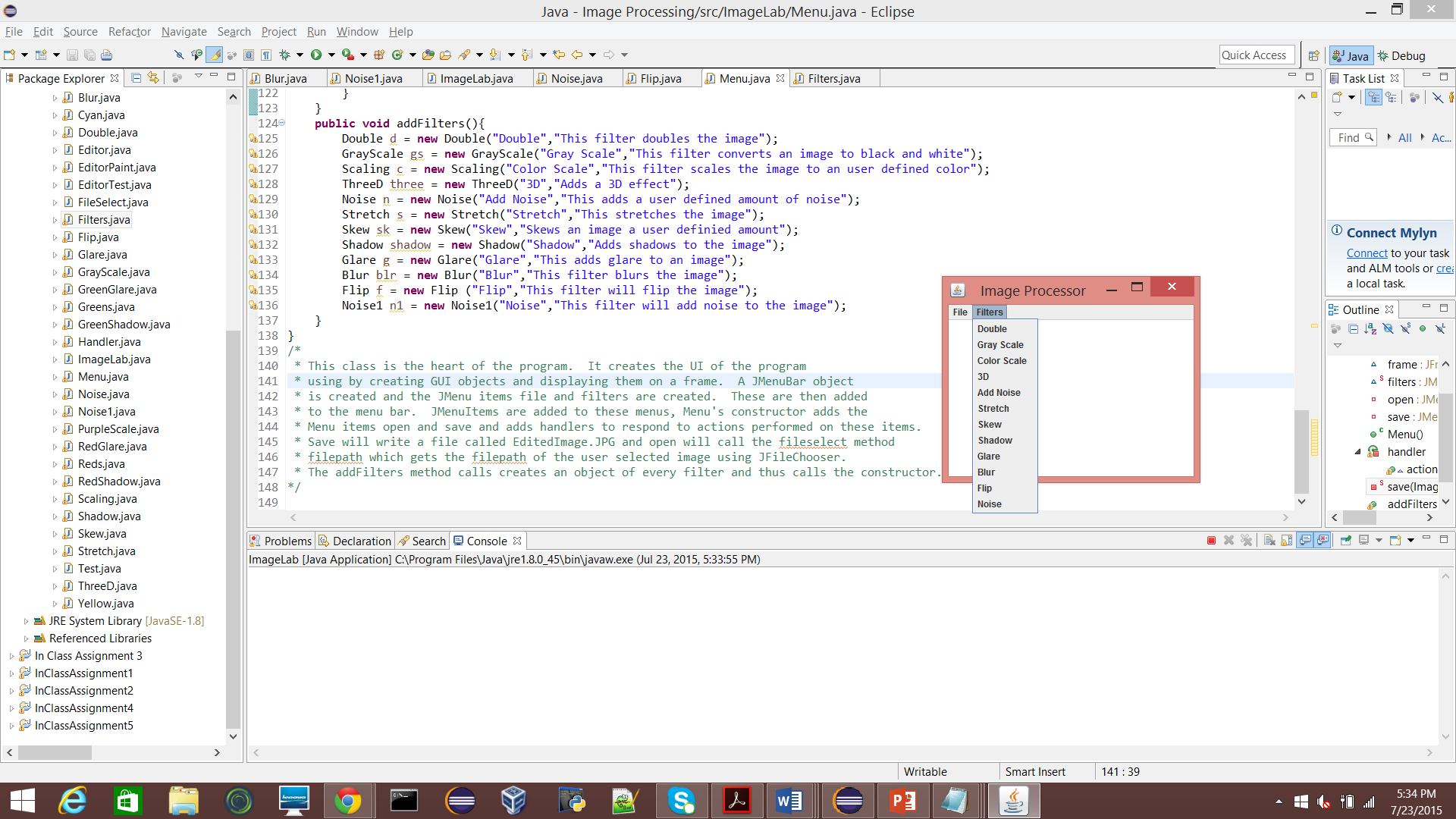
Scaling c = new Scaling("Color Scale","This filter scales the image to an user defined color");

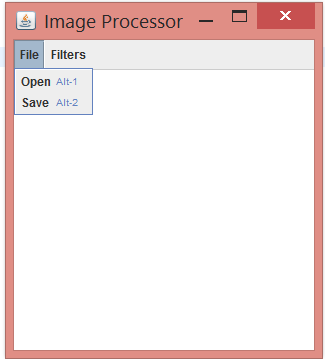
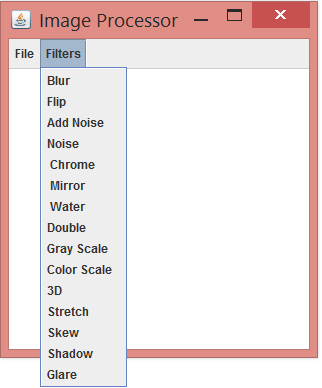
ThreeD three = new ThreeD("3D","Adds a 3D effect"). The entire list of Filter classes is as follows:

1. Class "Blur", "This filter blurs the image"
2. Class "Flip", "This filter will flip the image”
3. Class "Add Noise", "This adds a user defined amount of noise"
4. Class "Noise", "This filter will add default noise to the image”
5. Class "Chrome", "This filter simulate colors”
6. Class "Mirror", "This filter mirrors an image"
7. Class "Water", "This filter produces a water ripple effects on an image");
8. Class "Double", "This filter doubles the image"
9. Class “Gray Scale", "This filter converts an image to black and white");
10. Class "Color Scale", "This filter scales the image to an user defined color");
11. Class "3D","Adds a 3D effect"
12. Class "Stretch", "This stretches the image"
13. Class “Shadow", "Adds shadows to the image"
14. Class "Glare", "This adds glare to an image"
15. Class "Skew", "Skews an image a user defined amount”
    1. Each of these classes, viz; Double, GrayScale, Scaling, ThreeD, Noise, Blur etc, are derived classes from the base Class Filter. This class Filter is basically extended from MenuItem, and as such is a Menuitem. A few special filter classes will create additional GUI windows to allow for user input. All filter classes call the constructor of the filters class which creates a JMenuItem in the Filters menu based on the two strings passed as parameters. All constructors of subclasses of filter will then add a actionlistener and have a unique handler. The handler of every filter will take the image, process it through for loops, iterating though every pixel and changing it according to some pattern and then calls the method makeImage() from the filter class. This MakeImage() function is defined in Filter.java (base class), and will grab the BufferedImage and display it in the frame. The variable icon in this class is also set to the edited image so for the case where a user wants to apply several filters on the same image.
    2. **Filter Description:**

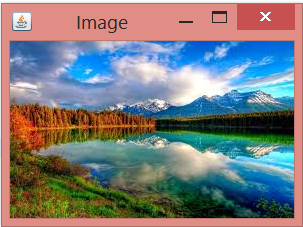
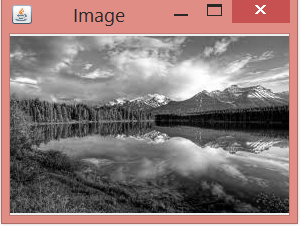
* **ChromeFilter** (low pass filter) is a variant of Light Filter which simulates chrome by applying a wavy transfer function to the result. So we can see image in black. It can create a variety of metallic effects and an even wider variety of very dull effects so experimentation is encouraged.
* **WaterFilter** will produce a water ripple effect on your image. You can change the wavelength, phase and amplitude of the ripples and turn antialiasing on or off. It's fun to animate the phase of the ripples to get the effect of the ripples moving out from the centre. Water filter considers the wavelength, amplitude, X and Y components from which we need to process the image and it should look like wave appearance to human eye, radius from the center etc. Wavelength, phase and amplitude should be set in such a way that wave should appear which can be done if parameters are set to sinusoidal formation. Pixel\*1.2 + pixel\*2.0 + pixel 2.5 for e.g and then divide with same value so that it will appear like sinusoidal wave.
* **GrayScale**: This filter converts an image to a grayscale image. To do this it finds the brightness of each pixel and sets the red, green and blue of the output to the brightness value. But what is the brightness? The simplest answer might be that it is the average of the RGB components, but that neglects the way in which the human eye works. The eye is much more sensitive to green and red than it is to blue, and so we need to take less acount of the blue and more account of the green. The weighting used by GrayscaleFilter is: luma = 77R + 151G + 28B . There are no parameters to this filter.
* **FlipFilter**: The FlipFilter does simple reflections and rotations of an image. You can rotate the image by 90, 180 or 270 degrees, flip it horizontally, vertically or about its leading diagonal. This can be useful for producing tiled patterns or variants of game sprites. Parameters: int Operation - One of FLIP\_H, FLIP\_V, FLIP\_HV, FLIP\_90CW, FLIP\_90CCW, FLIP\_180.
* **NoiseFilter**: NoiseFilter adds random noise to an image. You can specify the amount of noise, whether it has a uniform or Gaussian distribution and whether the noise is monochrome or coloured. This filter is useful for making images look 'less perfect' or as a starting point for producing textures.
* **MirrorFilter**: This filter mirrors an image vertically with a graduated transparency. It was written for the single purpose of producing an effect like that shown in the image below, so I make no excuses about its lack of generality.
* **BoxBlurFilter**: This filter blurs an image with a box blur algorithm. This is much faster than Gaussian blur, but doesn't look as good. However, if you iterate a box blur, it approximates a Gaussian blur so this may be significantly faster for a given blur radius. You can also use the box blur to do horizontal or vertical motion blur.

**Chapter 3: Results of Filter Appication**

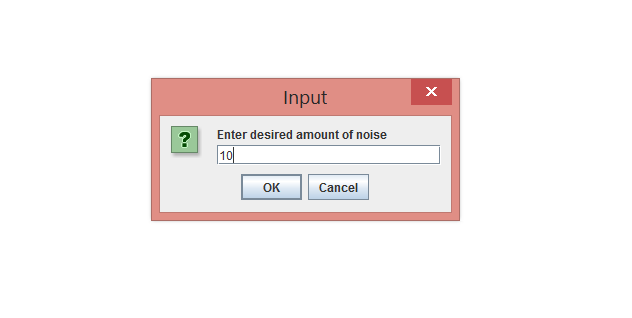


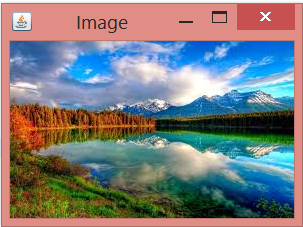
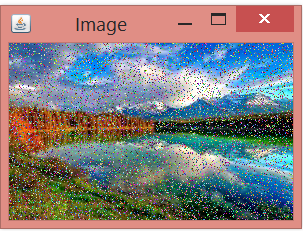
 

**3.1 Gray Scale:**

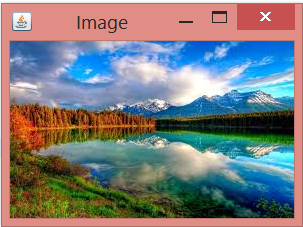
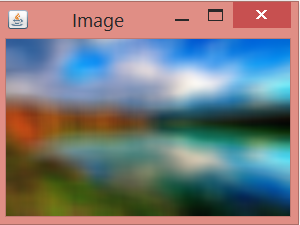
** **

**3.2 Add Noise**

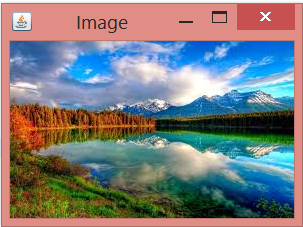
****

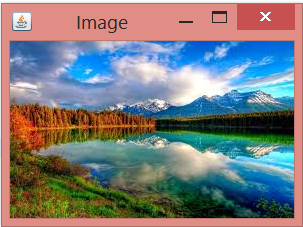
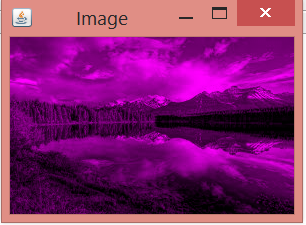
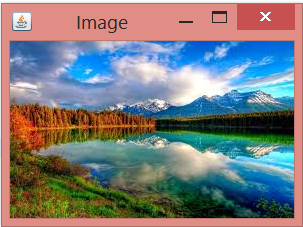
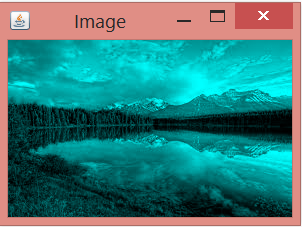
** **

**3.3 Blur**

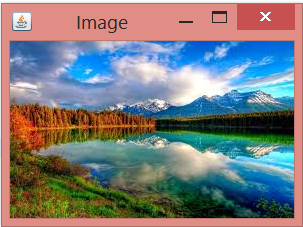
** **

**3.4 Color Scale**

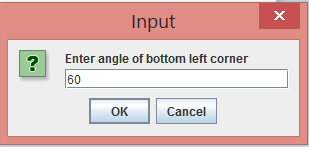
** **

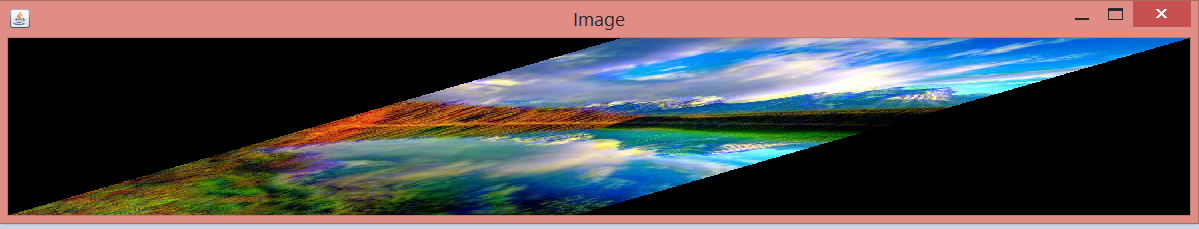
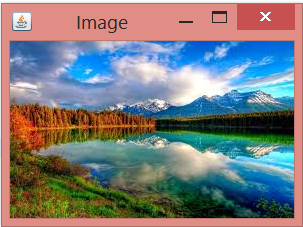
**   **

**3.5 Double**

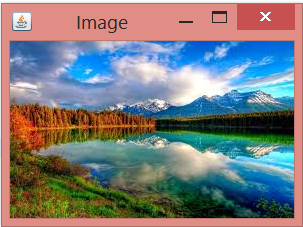
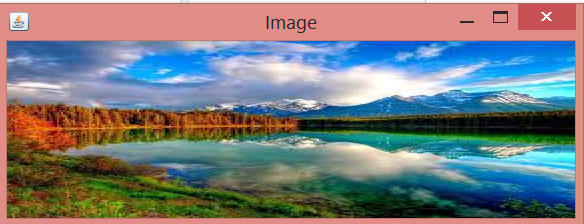
** **

**3.6 Skew**

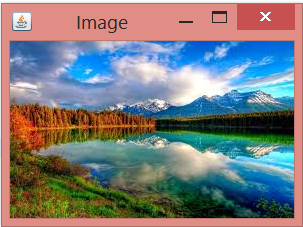
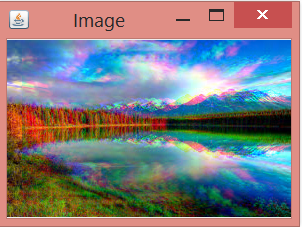
****

****

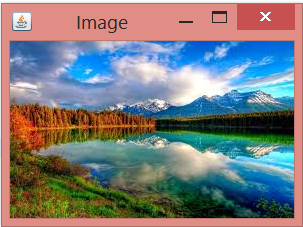
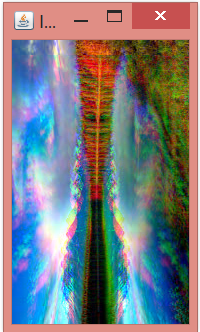
**3.7 Stretch**

** **

**3.8 3D**

** **

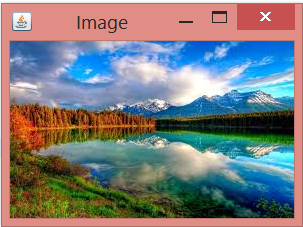
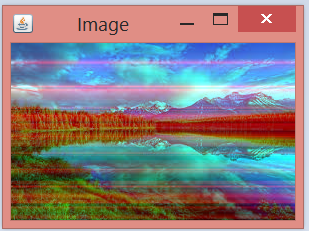
**3.9 Flip**

** **

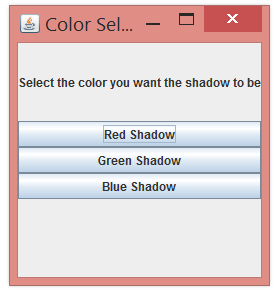
**3.10 Glare**

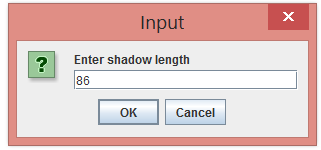
****

**Red Glare**

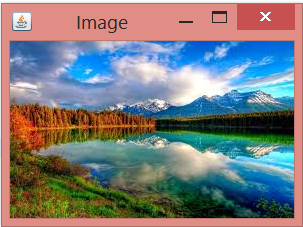
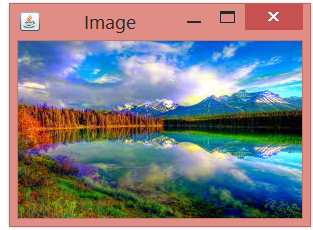
** **

**3.11 Shadow**

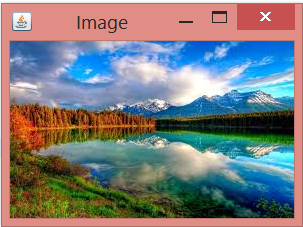
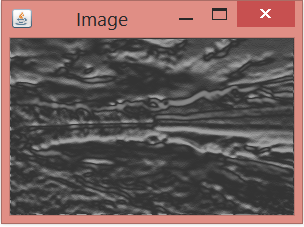
****

****

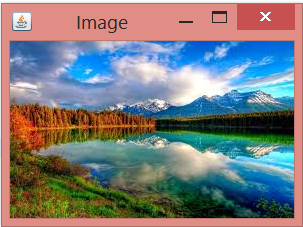
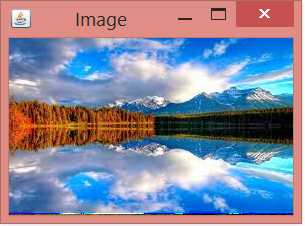
**Blue Shadow**

** **

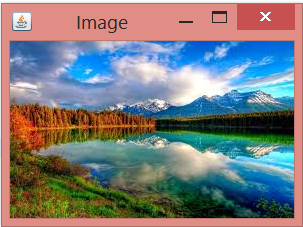
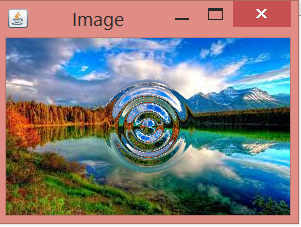
**3.12 Chrome**

** **

**3.13 Mirror**

** **

**3.14 Water**

** **

**Chapter 4: Conclusion**

* This course work and project taught us how **object oriented programming** works and what are the other important components need to take in consideration when doing object oriented programming.
* We have implemented image processing functionality using BufferedImage set/get RGB for each pixel. Displayed using JFrame, Jmenu, JMenuItem
* We learned how to apply ‘in class’ concepts ( inheritance, 2D array, interface, Base class, derived class, GUI etc. ) discussed in class.
* Also we learned some new concepts like JMenu, BufferedImage etc. which are very helpful in image processing.
* We learned how to apply different filters to image and which filter does what changes to image.
* Now we can successfully create functional GUI where user can choose and click buttons to open and save the image. Also can apply a filter of choice and observe the changes.

**Chapter 5: Appendix**

**Pixel shifting:**

Returns an integer pixel in the default RGB color model (TYPE\_INT\_ARGB) and default sRGB colorspace. Color conversion takes place if this default model does not match the image ColorModel. There are only 8-bits of precision for each color component in the returned data when using this method.

**Class BufferedImage:**

The BufferedImage subclass describes an [Image](http://docs.oracle.com/javase/7/docs/api/java/awt/Image.html) with an accessible buffer of image data. A BufferedImage is comprised of a [ColorModel](http://docs.oracle.com/javase/7/docs/api/java/awt/image/ColorModel.html) and a [Raster](http://docs.oracle.com/javase/7/docs/api/java/awt/image/Raster.html) of image data. The number and types of bands in the [SampleModel](http://docs.oracle.com/javase/7/docs/api/java/awt/image/SampleModel.html) of the Raster must match the number and types required by the ColorModel to represent its color and alpha components. AllBufferedImage objects have an upper left corner coordinate of (0, 0). Any Raster used to construct a BufferedImage must therefore have minX=0 and minY=0.

This class relies on the data fetching and setting methods of Raster, and on the color characterization methods of ColorModel.

**Chapter 6: References**

* Filters based on

[http](file:///C:\Users\Erc\Downloads\http)[://www.jhlabs.com/ip/filters/index.html](http://www.jhlabs.com/ip/filters/index.html)

* BufferedImage Wiki

<http://docs.oracle.com/javase/7/docs/api/java/awt/image/BufferedImage.html>

* Jmenu Swing Tutorial

<http://www.java2s.com/Tutorial/Java/0240__Swing/0420__JMenuItem.htm>

* Pixel shifting

<http://stackoverflow.com/questions/10726594/bufferedimage-getrgbx-y-does-not-yield-alpha>

* Text

Java how to program (Early object ) 10 th edition