/\*\*

\* PictureDemo.java

\* Hunter Damron

\* Honor Code: I ain't a cheater

\* Purpose: Creates picture based on user choice, then gives the user

\* the option to make a border around the picture, scale down to

\* half the size, or scale up to twice the size, respectively.

\*/

import javax.swing.JOptionPane;

public class PictureDemo {

public static void main(String[] args) {

String input; //Reads user input

Picture queenmary = new Picture();

/\*

\* Loads picture based on user input

\*/

input = JOptionPane.showInputDialog(null, "Which picture would you like to load?\n"

+ "'A' for NM.jpg\n'B' for queen-mary.png\nAny other option will default to B");

if(input.matches("A")) {

queenmary.load("NM.jpg");

}

else {

queenmary.load("queen-mary.png");

}

/\*

\* Makes border around picture

\*/

input = JOptionPane.showInputDialog("Would you like a border around your picture?");

if(input.matches("yes")) {

makeBorder(queenmary);

}

else {

if(!(input.matches("no"))) {

JOptionPane.showMessageDialog(null, "Are you dumb?! This is a yes or no question!\n"

+ "Here it is without a border, idiot.");

}

}

/\*

\* Scale picture down by 1/2

\*/

input = JOptionPane.showInputDialog("Would you like to scale your picture to one half its size?");

if(input.matches("yes")) {

makeHalf(queenmary);

}

else {

if(!(input.matches("no"))) {

JOptionPane.showMessageDialog(null, "Mamma always said \"Stupid is as stupid does.\""

+ " So you must be stupid or something. It is yes or no.");

}

}

/\*

\* Scales picture up by 2

\*/

input = JOptionPane.showInputDialog("Would you like to scale your picture to one twice its size?");

if(input.matches("42")) {

JOptionPane.showMessageDialog(null, "What is the answer to the ultimate question of life,"

+ " the universe, and everything?");

}

if(input.matches("yes")) {

makeTwice(queenmary);

}

else {

if(!(input.matches("no"))) {

JOptionPane.showMessageDialog(null, "My vocabulary is very limited. I understand yes, no, and 42.");

}

}

}

public static void makeBorder(Picture pic) {

int width = pic.getWidth();

width = width - 30;

int height = pic.getHeight();

height = height - 30;

pic.scale(width, height);

pic.move(15, 15);

pic.border(10);

}

public static void makeHalf(Picture pic) {

int width = pic.getWidth();

int topx = width / 4;

width = width / 2;

int height = pic.getHeight();

int topy = height / 4;

height = height / 2;

pic.scale(width, height);

pic.move(topx, topy);

}

public static void makeTwice(Picture pic) {

int width = pic.getWidth();

int topx = -1 \* width / 4;

width = width \* 2;

int height = pic.getHeight();

int topy = -1 \* height / 4;

height = height \* 2;

pic.move(topx, topy);

pic.scale(width, height);

}

}

import java.awt.Color;

import java.io.File;

import java.net.URL;

import java.awt.geom.AffineTransform;

import java.awt.image.AffineTransformOp;

import java.awt.image.BufferedImage;

import java.awt.image.BufferedImageOp;

import java.awt.image.ColorModel;

import java.awt.image.Raster;

import java.awt.image.WritableRaster;

import javax.imageio.ImageIO;

import javax.swing.ImageIcon;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

import javax.swing.JLabel;

/\*\*

\* This class allows you to view and edit pictures.

\*/

public class Picture

{

private String source;

private JFrame frame;

private JLabel label;

private BufferedImage image;

/\*\*

Constructs a blank picture.

\*/

public Picture()

{

frame = new JFrame();

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

label = new JLabel("(No image)");

frame.add(label);

frame.pack();

frame.setVisible(true);

}

/\*\*

Gets the width of this picture.

@return the width

\*/

public int getWidth() { return image.getWidth(); }

/\*\*

Gets the height of this picture.

@return the height

\*/

public int getHeight() { return image.getHeight(); }

/\*\*

Loads a picture from a given source.

@param source the image source. If the source starts

with http://, it is a URL, otherwise, a filename.

\*/

public void load(String source)

{

try

{

this.source = source;

BufferedImage img;

if (source.startsWith("http://"))

img = ImageIO.read(new URL(source).openStream());

else

img = ImageIO.read(new File(source));

setImage(img);

}

catch (Exception ex)

{

this.source = null;

ex.printStackTrace();

}

}

/\*\*

Reloads this picture, undoing any manipulations.

\*/

public void reload()

{

load(source);

}

/\*\*

Displays a file chooser for picking a picture.

\*/

public void pick()

{

JFileChooser chooser = new JFileChooser(".");

if (chooser.showOpenDialog(null) == JFileChooser.APPROVE\_OPTION)

{

load(chooser.getSelectedFile().getAbsolutePath());

}

}

/\*\*

Moves this picture by the given amount in x- and y-direction.

@param dx the offset in the x-direction

@param dy the offset in the y-direction

\*/

public void move(int dx, int dy)

{

BufferedImageOp op = new AffineTransformOp(

AffineTransform.getTranslateInstance(dx, dy),

AffineTransformOp.TYPE\_BILINEAR);

BufferedImage filteredImage

= new BufferedImage(image.getWidth(), image.getHeight(),

BufferedImage.TYPE\_INT\_ARGB);

op.filter(image, filteredImage);

setImage(filteredImage);

}

/\*\*

Scales this picture to a new size. If the new size is smaller

than the old size, the remainder is filled with transparent

pixels. If it is larger, it is clipped.

@param newWidth the new width of the picture

@param newHeight the new height of the picture

\*/

public void scale(int newWidth, int newHeight)

{

double dx = newWidth \* 1.0 / image.getWidth();

double dy = newHeight \* 1.0 / image.getHeight();

BufferedImageOp op = new AffineTransformOp(

AffineTransform.getScaleInstance(dx, dy),

AffineTransformOp.TYPE\_BILINEAR);

BufferedImage filteredImage

= new BufferedImage(image.getWidth(), image.getHeight(),

BufferedImage.TYPE\_INT\_ARGB);

op.filter(image, filteredImage);

setImage(filteredImage);

}

/\*\*

Adds a black border to the image.

@param width the border width

\*/

public void border(int width)

{

for (int x = 0; x < width; x++)

{

for (int y = 0; y < image.getHeight(); y++)

{

setColorAt(x, y, Color.BLACK);

setColorAt(image.getWidth() - 1 - x, y, Color.BLACK);

}

}

for (int y = 0; y < width; y++)

{

for (int x = width; x < image.getWidth() - width; x++)

{

setColorAt(x, y, Color.BLACK);

setColorAt(x, image.getHeight() - 1 - y, Color.BLACK);

}

}

}

/\*\*

Gets the color of a pixel.

@param x the column index (between 0 and getWidth() - 1)

@param y the row index (between 0 and getHeight() - 1)

@return the color of the pixel at position (x, y)

\*/

public Color getColorAt(int x, int y)

{

Raster raster = image.getRaster();

ColorModel model = image.getColorModel();

int argb = model.getRGB(raster.getDataElements(x, y, null));

return new Color(argb, true);

}

/\*\*

Sets the color of a pixel.

@param x the column index (between 0 and getWidth() - 1)

@param y the row index (between 0 and getHeight() - 1)

@param c the color for the pixel at position (x, y)

\*/

public void setColorAt(int x, int y, Color c)

{

WritableRaster raster = image.getRaster();

ColorModel model = image.getColorModel();

Object colorData = model.getDataElements(c.getRGB(), null);

raster.setDataElements(x, y, colorData);

label.repaint();

}

private void setImage(BufferedImage image)

{

this.image = image;

label.setIcon(new ImageIcon(image));

label.setText("");

label.setSize(image.getWidth(), image.getHeight());

frame.pack();

}

}