1.4. Lectura de imágenes mediante un lenguaje de programación

En este punto se muestra como acceder a una imagen RGB almacenada en un formato de compresión como jpg y en lenguaje Java orientado a objetos y empleando el patrón Modelo-Vista-Control. En la figura 1.14 se muestra el código de la clase LectorDeImagen, esta clase es parte del modelo del patrón; en el constructor líneas 30 y 31 es donde se lee la imagen que se almacena en un objeto BufferedImage. Si se necesita el conjunto de pixeles de la imagen en un arreglo matricial de tipo entero se tiene el método getImagenInt() que devuelve la matriz de pixeles de acuerdo con lo que usted necesite, devuelve el canal rojo, el canal verde, el canal azul, la imagen en niveles de gris o los pixeles en color. También puede devolver un objeto de tipo Image si lo que usted necesita es visualizarla en alguna ventana de Java, el método es getImage(), también tiene un método para convertir la matriz de imagen a un vector (convertirInt2DA1D()), también tiene algunos métodos getters() y setters() para devolver lo necesario si el usuario lo necesita. También se creó en la vista un panel que almacena la imagen en un objeto Image para poder visualizarla, esta clase se llama PanelDeImagen y se muestra en la figura 1.15, la cual tiene su constructor y el método paintComponent() para poder visualizar la imagen. Para crear una ventana en donde podamos incluir al panel donde se visualiza la imagen se tiene en la vista la clase FrameVisorImagen que se muestra en la figura 1.16, en esta clase se incluyen dos atributos, uno es el PanelDeImagen y otro es una clase del control (pertenece al patrón) denominada ControlImagen y se muestra en la figura 1.17. Por último se muestra la clase principal que contiene a la función main() donde comienza la ejecución del programa (figura 1.18). El resultado final se muestra en la figura 1.19.

```java
package unidaduno;

import java.awt.Color;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.awt.image.ColorModel;
import java.awt.image.MemoryImageSource;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import javax.imageio.ImageIO;
import javax.swing.JFrame;

/**
 * Clase lectora de imagen
 *
 * @author sdelaot
 */
public class LectorDeImagen {
    private ColorModel colorModel;
    private int tipo;
    private BufferedImage laImagen;
    private String path;
    private String nombreArchivo;
    public LectorDeImagen(String nombreArchivo) {
        this.nombreArchivo = nombreArchivo;
        path = "/Users/sdelaot/NetBeansProjects/ImageAnalysis/src/vista/" +
                this.nombreArchivo;
        System.out.println(path);
        try {
            FileInputStream input = new FileInputStream(new File(path));
            laImagen = ImageIO.read(input);
            colorModel = laImagen.getColorModel();
            tipo = laImagen.getType();
        } catch(IOException ioe) {
            System.err.println(ioe);
        }
    }
```

```java
    /**
     * Toma el bufer de la imagen
     *
     * @return devuelve el objeto de la imagen
     */
    public BufferedImage getBufferedImage() {
        return laImagen;
    }
    /**
     * Pone el path en caso de que se cambie
     *
     * @param path la nueva via de acceso
     */
    public void setPath(String path) {
        this.path = path;
    }
    /**
     * Devuelve la via de acceso de la imagen
     *
     * @return devuelve el path donde encuentra a la imagen
     */
    public String getPath() {
        return path;
    }
    /**
     * Devuelve el nombre del archivo de la imagen
     *
     * @return devuelve el nombre del archivo donde encuentra a la imagen
     */
    public String getNombreArchivo() {
        return nombreArchivo;
    }
    /**
     * Devuelve la imagen del color solicitado para procesarla
     *
     * @param queColor 1=rojo, 2=verde, 3=azul, 4=niveles de gris, 5=color<br>
     *
     * @return devuelve una imagen entera en dos dimensiones para ser procesada
     * como sea necesario
     */
    public int [][] getImagenInt(int queColor) {
        int alto = laImagen.getHeight();
        int ancho = laImagen.getWidth();
        int [][] imagenInt = new int[alto][ancho];
        int pixel;
        Color color;
        for(int y=0; y<alto; y++) {
            for(int x=0; x<ancho; x++) {
                switch(queColor) {
                    case 1: // rojo
                        pixel = laImagen.getRGB(x,y);
                        int R = (pixel & 0x00ff0000) >> 16;
                        color = new Color(R, 0, 0);
                        imagenInt[y][x] = color.getRGB();
                        break;
                    case 2: // verde
                        pixel = laImagen.getRGB(x,y);
                        int G = (pixel & 0x0000ff00) >> 8;
                        color = new Color(0, G, 0);
                        imagenInt[y][x] = color.getRGB();
                        break;
                    case 3: // azul
                        pixel = laImagen.getRGB(x,y);
                        int B = pixel & 0x000000ff;
                        color = new Color(0, 0, B);
                        imagenInt[y][x] = color.getRGB();
                        break;
```

```
                    case 4:
                        pixel = laImagen.getRGB(x,y);
                        int rojo  = (pixel & 0x00ff0000) >> 16;
                        int verde = (pixel & 0x0000ff00) >> 8;
                        int azul  =  pixel & 0x000000ff;
                        double gris = (double)(rojo+verde+azul) / 3.0;
                        color = new Color((int)gris, (int)gris, (int)gris);
                        imagenInt[y][x] = color.getRGB();
                        break;
                    case 5:
                        imagenInt[y][x] = laImagen.getRGB(x, y);
                        break;
                    }
                }
            }
        return imagenInt;
    }
    /**
     * Devuelve una imagen para ser vista en un frame, panel, frame interno,
     * etc.
     *
     * @param queColor 1=rojo, 2=verde, 3=azul, 4=niveles de gris, 5=color<br>
     *
     * @return Devuelve ob objeto de tipo image
     */
    public Image getImage(int queColor) {
        Image imagenLocal;
        int [][] imagenInt = getImagenInt(queColor);
        int alto = laImagen.getHeight();
        int ancho = laImagen.getWidth();

        JFrame frameTmp = new JFrame();
        imagenLocal = frameTmp.createImage(new MemoryImageSource(ancho,
                alto, convertirInt2DAInt1D(imagenInt, ancho, alto),
                0, ancho));

        return imagenLocal;
    }
    /**
     * Convierte un buffer de tipo double y de dos dimensiones de imagen a uno
     * de una dimension de tipo entero.
     *
     * @param matriz la matriz a convertir
     * @param ancho ancho de la imagen en pixeles
     * @param alto alto de la imagen en pixeles
     *
     * @return el vector de la image convertida
     */
    public int [] convertirInt2DAInt1D(int[][] matriz, int ancho, int alto) {
        int index = 0;
        int [] bufferInt = null;
        try {
            bufferInt = new int[ancho * alto];
            for(int y = 0; y < alto; y++) {
                for(int x=0; x < ancho; x++) {
                    bufferInt[index++] = matriz[y][x];
                    }
                }
        } catch (NegativeArraySizeException e) {
            System.out.println(" Error alto, ancho o ambos negativos"
                + " en  convierteInt2DAInt1D( double [][] ) "
                + e);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println(" Error desbordamiento en bufferInt"
                + " en  convierteInt2DAInt1D( double [][] ) "
                + e);
        } catch (NullPointerException e) {
            System.out.println(" Error bufferInt nulo"
                + " en  convierteInt2DAInt1D( double [][] ) "
                + e);
        }
        return bufferInt;
    }
```

```
179     /**
180      * Devuelve el alto en pixeles de la imagen que se aperturo
181      *
182      * @return devuelve la altura en pixeles de la imagen
183      */
184     public int getAlto() {
185         return laImagen.getHeight();
186     }
187     /**
188      * Devuelve el ancho en pixeles de la imagen que se aperturo
189      *
190      * @return devuelve el ancho en pixeles de la imagen
191      */
192     public int getAncho() {
193         return laImagen.getWidth();
194     }
195 }
```

Figura 1.14 Clase LectorDeImagen.

```
1   package vista;
2
3   /**
4    * Para esta clase se utilizan las librerias awt y swing con estas clases.
5    */
6   import java.awt.Graphics;
7   import java.awt.Image;
8   import javax.swing.JPanel;
9   /**
10   * Clase para crear el panel que despliega la imagen junto con el buffer de
11   * la imagen.
12   *
13   * @author  Saul De La O Torres
14   * @version 1.0 20/jun/2003
15   */
16  public class PanelDeImagen extends JPanel {
17      /**
18       * Imagen que debera ser desplegada en pantalla, sobre el escritorio de
19       * trabajo en un Frame Interno.
20       */
21      private Image imagen;
22      /**
23       * Constructor de la clase, recibe a la imagen, su buffer, el alto y el
24       * ancho de la misma.
25       *
26       * @param img la imagen a desplegar en el panel
27       */
28      public PanelDeImagen( Image img ) {
29          this.imagen = img;
30      }
31      /**
32       * Dibuja la imagen BMP en el panel del Frame interno.
33       *
34       * @param g el componente grafico !=null
35       */
36      @Override
37      public void paintComponent( Graphics g ) {
38          super.paintComponent( g ); //pinta el fondo
39          //Pone la imagen en su tamanio normal.
40          if( imagen!=null ) g.drawImage(imagen, 1, 1, this );
41      }
42  }
```

Figura 1.15 Clase PanelDeImagen.

```java
package vista;

import control.ControlImagen;
import java.awt.BorderLayout;
import java.awt.Container;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import javax.swing.JFrame;

/**
 *
 * @author sdelaot
 */
public class FrameVisorImagen extends JFrame {
    private PanelDeImagen panel;
    private ControlImagen controlImagen;
    public FrameVisorImagen(String nombreArchivo) {
        super("Visor de imagen");
        initComponents(nombreArchivo);
    }
    private void initComponents(String nombreArchivo) {
        Container contenedor = this.getContentPane();
        contenedor.setLayout(new BorderLayout());
        controlImagen = new ControlImagen(nombreArchivo);
        panel = new PanelDeImagen(controlImagen.getImagen());
        contenedor.add(panel, BorderLayout.CENTER);
        this.setSize(controlImagen.getAncho(), controlImagen.getAlto()+40);
        this.setVisible(true);
        this.addWindowListener(new SalidaFrame());
    }
    private class SalidaFrame extends Object implements WindowListener {
        @Override
        public void windowOpened(WindowEvent e) { }
        @Override
        public void windowClosing(WindowEvent e) {
            System.exit(0);
        }
        @Override
        public void windowClosed(WindowEvent e) {
            System.exit(0);
        }
        @Override
        public void windowIconified(WindowEvent e) { }
        @Override
        public void windowDeiconified(WindowEvent e) { }
        @Override
        public void windowActivated(WindowEvent e) { }
        @Override
        public void windowDeactivated(WindowEvent e) { }
    }
}
```

Figura 1.16 Clase FrameVisorImagen.

```java
package control;

import java.awt.Image;
import unidaduno.LectorDeImagen;

/**
 *
 * @author sdelaot
 */
public class ControlImagen {
    private LectorDeImagen lector;
    public ControlImagen(String nombreArchivo) {
        lector = new LectorDeImagen(nombreArchivo);
    }
    public Image getImagen(int cual) {
        return lector.getImage(cual);
    }
    public int getAlto() {
        return lector.getAlto();
    }
    public int getAncho() {
        return lector.getAncho();
    }
}
```

Figura 1.17 Clase ControlImagen.

```java
package imageanalysis;

import vista.FrameVisorImagen;

/**
 *
 * @author sdelaot
 */
public class ImageAnalysis {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        FrameVisorImagen visor = new FrameVisorImagen("aguila.jpg");
    }

}
```

Figura 1.18 Clase principal ImageAnalysis.

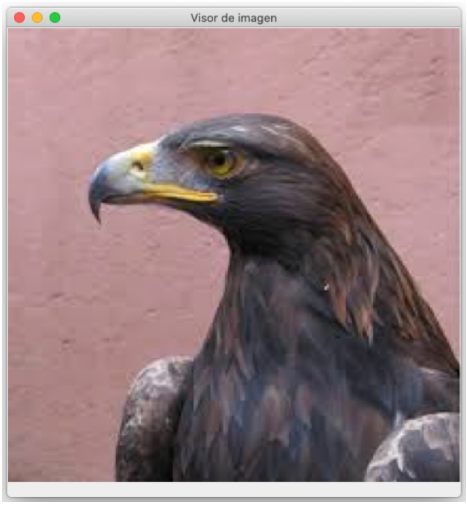

Figura 1.19 Resultado final del visualizador de imagen.