Bryan Kline

CS326

Homework 8

LaTex (TeXstudio)

12/08/2016

1.
Write a Java application that implements a color sampler, each color is described by its name and its red, green and blue components. Colors are read from a file, displayed in the sampler, with a list for each color, and the sampler can change the colors with increment and decrement buttons. It should also be able to save the color values back to the file and reset the color values to a previously saved state.

```
import java.io.*;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.event.*;

//ColorSampler class
public class ColorSampler extends JFrame
{
    //variable declarations
    int colorRow = -1;
    int redCounter = 0;
    int greenCounter = 0;
    int blueCounter = 0;
    static int[][] colors = new int[11][3];
    static int[][] fileColors = new int[11][3];
    static protected String fileName;
    String colorListData[] = "Red", "Green", "Blue",
                            Yellow", "Cyan", "Magenta", "Orange",
                                    "Pink", "Grey", "Black", "White";

    //button declarations
    protected JButton save;
    protected JButton reset;
    protected JButton redMinus;
    protected JButton greenMinus;
    protected JButton blueMinus;
    protected JButton redPlus;
    protected JButton greenPlus;
    protected JButton bluePlus;

    //panel declarations
    protected JPanel blank1, blank2;
    protected JPanel mainPanel;
    protected JPanel listPanel;
    protected JPanel button;
    protected JPanel colorPanel;
    protected JPanel colorButtonPanel;
    protected JPanel controlButtonPanel;

    //text field declarations
    protected JTextField red;
    protected JTextField green;
    protected JTextField blue;
protected JTextField myTitle;

//label declarations
protected JLabel redLabel;
protected JLabel greenLabel;
protected JLabel blueLabel;

//list declaration
protected JList<String> colorList;

//graphic declaration
protected Drawing drawBox;

//main, throws IO exception, takes in command line argument
public static void main (String argv[]) throws IOException
{
    int row, column;
    fileName = argv[0];

    //class constructor call
    new ColorSampler("Color Sampler");

    //input stream opened
    FileInputStream inStream = new FileInputStream(fileName);
    InputStreamReader streamReader = new
                                InputStreamReader(inStream);
    StreamTokenizer streamToken = new StreamTokenizer(streamReader);

    //reading in from the file into arrays
    streamToken.nextToken();
    for(row = 0; row < 11; row++)
    {
        streamToken.nextToken();
        for(column = 0; column < 3; column++)
        {
            fileColors[row][column] = (int) streamToken.nval;
            colors[row][column] = fileColors[row][column];
            streamToken.nextToken();
        }
    }

    //input stream closed
    inStream.close();
    }

    //class constructor
    public ColorSampler(String title)
    {
        //title set, bounds set, destructor created
        super(title);
        setBounds(100, 100, 500, 350);
        addWindowListener(new WindowDestroyer());

        //layouts created
```

```
getContentPane().setLayout(new GridLayout(1, 2, 10, 10));
mainPanel = new JPanel(new GridLayout(2, 1, 10, 10));
button = new JPanel(new GridLayout(2, 1, 10, 10));
listPanel = new JPanel(new FlowLayout());
colorPanel = new JPanel(new GridLayout(1, 1, 10, 10));
colorButtonPanel = new JPanel(new GridLayout(3, 4, 10, 10));
controlButtonPanel = new JPanel(new FlowLayout());
blank1 = new JPanel(new GridLayout(1, 1, 10, 10));
blank2 = new JPanel(new GridLayout(1, 1, 10, 10));

//buttons, text fields, labels, list, and graphic created
save = new JButton("Save");
reset = new JButton("Reset");
redMinus = new JButton("-");
greenMinus = new JButton("-");
blueMinus = new JButton("-");
redPlus = new JButton("+");
greenPlus = new JButton("+");
bluePlus = new JButton("+");
red = new JTextField("" + redCounter);
green = new JTextField("" + greenCounter);
blue = new JTextField("" + blueCounter);
redLabel = new JLabel(" Red:");
greenLabel = new JLabel(" Green:");
blueLabel = new JLabel(" Blue:");
colorList = new JList<String>();
drawBox = new Drawing();

//panels, buttons, and graphic added to pane
getContentPane().add(mainPanel);
getContentPane().add(listPanel);
mainPanel.add(colorPanel);
mainPanel.add(button);
button.add(colorButtonPanel);
button.add(controlButtonPanel);
colorPanel.add(drawBox);
colorButtonPanel.add(redLabel);
colorButtonPanel.add(red);
colorButtonPanel.add(redMinus);
colorButtonPanel.add(redPlus);
colorButtonPanel.add(greenLabel);
colorButtonPanel.add(green);
colorButtonPanel.add(greenMinus);
colorButtonPanel.add(greenPlus);
colorButtonPanel.add(blueLabel);
colorButtonPanel.add(blue);
colorButtonPanel.add(blueMinus);
colorButtonPanel.add(bluePlus);
controlButtonPanel.add(blank1);
```

```
        controlButtonPanel.add(save);
        controlButtonPanel.add(reset);
        controlButtonPanel.add(blank2);

        //list added to pane, list size set, list populated with
        //data
        colorList.setFixedCellWidth(200);
        colorList.setFixedCellHeight(20);
        listPanel.add(colorList);
        colorList.setListData(colorListData);

        //action handler listeners set for buttons and the list
        redMinus.addActionListener(new ActionHandler());
        redPlus.addActionListener(new ActionHandler());
        red.addKeyListener(new KeyPress());
        greenMinus.addActionListener(new ActionHandler());
        greenPlus.addActionListener(new ActionHandler());
        green.addKeyListener(new KeyPress());
        blueMinus.addActionListener(new ActionHandler());
        bluePlus.addActionListener(new ActionHandler());
        blue.addKeyListener(new KeyPress());
        save.addActionListener(new ActionHandler());
        reset.addActionListener(new ActionHandler());
        colorList.addListSelectionListener(new ListHandler());

        //pane set to visible
        setVisible(true);
    }

    //action handler class
    private class ActionHandler implements ActionListener
    {
        boolean cleared = false;
        boolean saved = false;

        //function to determine which action occured
        public void actionPerformed(ActionEvent button)
        {
            //if source is a plus or minus button for a color,
            //increment the corresponding color
            if(button.getSource() == redMinus)
            {
                if(redCounter > 0)
                {
                    redCounter = redCounter - 5;
                }
            }
            else if(button.getSource() == greenMinus)
            {
                if(greenCounter > 0)
    {
        greenCounter = greenCounter - 5;
    }
}
else if(button.getSource() == blueMinus)
{
    if(blueCounter > 0)
    {
        blueCounter = blueCounter - 5;
    }
}
else if(button.getSource() == redPlus)
{
    if(redCounter < 255)
    {
        redCounter = redCounter + 5;
    }
}
else if(button.getSource() == greenPlus)
{
    if(greenCounter < 255)
    {
        greenCounter = greenCounter + 5;
    }
}
else if(button.getSource() == bluePlus)
{
    if(blueCounter < 255)
    {
        blueCounter = blueCounter + 5;
    }
}
//if the source is the save button, call arraySave()
else if(button.getSource() == save)
{
    arraySave();
    setTitle("Color Sampler");
    saved = true;
}
//if the source is the reset button, call arrayReset()
else if(button.getSource() == reset)
{
    arrayReset();
    cleared = true;

    redCounter = colors[colorRow][0];
    greenCounter = colors[colorRow][1];
    blueCounter = colors[colorRow][2];
}
```

```
        if(!cleared)
        {
            colors[colorRow][0] = redCounter;
            colors[colorRow][1] = greenCounter;
            colors[colorRow][2] = blueCounter;
        }
        //update text fields
        red.setText("" + redCounter);
        green.setText("" + greenCounter);
        blue.setText("" + blueCounter);
        //if it hasn't been saved, change title
        if(!saved)
        {
            setTitle("Color Sampler*");
        }
    }
    //function to reset the array holding color values back to
    //what was last saved
    public void arrayReset()
    {
        int row, column;
        for(row = 0; row < 11; row++)
        {
            for(column = 0; column < 3; column++)
            {
                colors[row][column] = fileColors[row][column];
            }
        }
    }
    //function to save the current colors in the array into the
    //array that hold saved values
    public void arraySave()
    {
            int row, column;
            for(row = 0; row < 11; row++)
            {
                for(column = 0; column < 3; column++)
                {
                    fileColors[row][column] =
                                        colors[row][column];
                }
            }
    }
}
//list handler class
```

```
private class ListHandler implements ListSelectionListener
{
    String colorValue;

    //function to change color values based on the list
    //selection
    public void valueChanged(ListSelectionEvent event)
    {
        if(event.getSource() == colorList)
        {
            //if the list isn’t adjusting, get the string from
            //the list and then call colorProcess() on that
            //string to select the color to display
            if(!event.getValueIsAdjusting())
            {
                colorValue = (String)
                                colorList.getSelectedValue();
                colorProcess(colorValue);
            }
        }
    }

    //function to display the color corresponding to the list
    //selection
    public void colorProcess(String colorString)
    {

        //check the string passed in, set the colors values to
        //those corresponding to that color
        if(colorString == "Red")
        {
            redCounter = colors[0][0];
            greenCounter = colors[0][1];
            blueCounter = colors[0][2];
            colorRow = 0;
            }
            else if(colorString == "Green")
            {
                redCounter = colors[1][0];
                greenCounter = colors[1][1];
                blueCounter = colors[1][2];
                colorRow = 1;
            }
            else if(colorString == "Blue")
            {
                redCounter = colors[2][0];
                greenCounter = colors[2][1];
                blueCounter = colors[2][2];
                colorRow = 2;
            }
```

```
else if(colorString == "Yellow")
{
    redCounter = colors[3][0];
    greenCounter = colors[3][1];
    blueCounter = colors[3][2];
    colorRow = 3;
}
else if(colorString == "Cyan")
{
    redCounter = colors[4][0];
    greenCounter = colors[4][1];
    blueCounter = colors[4][2];
    colorRow = 4;
}
else if(colorString == "Magenta")
{
    redCounter = colors[5][0];
    greenCounter = colors[5][1];
    blueCounter = colors[5][2];
    colorRow = 5;
}
else if(colorString == "Orange")
{
    redCounter = colors[6][0];
    greenCounter = colors[6][1];
    blueCounter = colors[6][2];
    colorRow = 6;
}
else if(colorString == "Pink")
{
    redCounter = colors[7][0];
    greenCounter = colors[7][1];
    blueCounter = colors[7][2];
    colorRow = 7;
}
else if(colorString == "Grey")
{
    redCounter = colors[8][0];
    greenCounter = colors[8][1];
    blueCounter = colors[8][2];
    colorRow = 8;
}
else if(colorString == "Black")
{
    redCounter = colors[9][0];
    greenCounter = colors[9][1];
    blueCounter = colors[9][2];
    colorRow = 9;
```

```
            }
            else if(colorString == "White")
            {
                redCounter = colors[10][0];
                greenCounter = colors[10][1];
                blueCounter = colors[10][2];
                colorRow = 10;
            }
            //update text fields
            red.setText("" + redCounter);
            green.setText("" + greenCounter);
            blue.setText("" + blueCounter);
            //change the title
            setTitle("Color Sampler*");
        }
    }
}
//class to draw the graphic
private class Drawing extends JComponent
{
    //function that draws the rectangle
    public void paint(Graphics box)
    {
        //creates a rectangle and fills it with the color
        //values and draws it
        Dimension size = getSize();
        box.setColor(new Color(redCounter,
                                            greenCounter, blueCounter));
        box.fillRect(1, 1, size.width, size.height);
        repaint();
    }
}
//key press listener class
private class KeyPress implements KeyListener
{
    int r, g, b;
    //function that listens for the enter key to be pressed
    public void keyPressed(KeyEvent key)
    {
        //if the key entered is the constant for enter, get
        //the values in the text fields and if the values in
        //the text fields are valid, then update the list
        //array and color values
        if(key.getKeyCode() == KeyEvent.VK_ENTER)
        {
            r = Integer.valueOf(red.getText());
```

```
                g = Integer.valueOf(green.getText());
                b = Integer.valueOf(blue.getText());
                if(r >= 0 && r <= 255)
                {
                    redCounter = r;
                }
                if(g >= 0 && g <= 255)
                {
                    greenCounter = g;
                }
                if(b >= 0 && b <= 255)
                {
                    blueCounter = b;
                }
                if(colorRow > -1)
                {
                    colors[colorRow][0] = redCounter;
                    colors[colorRow][1] = greenCounter;
                    colors[colorRow][2] = blueCounter;
                }
                //update title
                setTitle("Color Sampler*");
            }
        }
        //empty interface methods
        public void keyTyped(KeyEvent e)
        public void keyReleased(KeyEvent e)
    }
    //class destructor
    private class WindowDestroyer extends WindowAdapter
    {
        //function that tries to write to the input file by
        //calling fileWriter(), throws an exception if it
        //it's not possible and ends the program upon
        //closure of the window
        public void windowClosing(WindowEvent event)
        {
            try
            {
                fileWriter();
            }
            catch(Exception e)
            {
                System.out.println("Couldn't save
                                            the colors to a file.");
            }
            //ends program
            System.exit(0);
        }
        //function which writes the saved color array back
        //out to the input file, throws io exception
        public void fileWriter() throws IOException
        {
            int row, column;
            String colorLabel = " ";
            //output stream opened
            FileOutputStream outStream = new
                                        FileOutputStream(fileName);
            PrintWriter streamWriter = new PrintWriter(outStream);
            //writes the appropriate color labels out to the
            //file then writes the current color values
            //corresponding to that color in the array
            for(row = 0; row < 11; row++)
            {
                switch(row)
                {
                    case 0:
                        colorLabel = "Red";
                    break;
                    case 1:
                        colorLabel = "Green";
                    break;
                    case 2:
                        colorLabel = "Blue";
                    break;
                    case 3:
                        colorLabel = "Yellow";
                    break;
                    case 4:
                        colorLabel = "Cyan";
                    break;
                    case 5:
                        colorLabel = "Magenta";
                    break;
                    case 6:
                        colorLabel = "Orange";
                    break;
                    case 7:
                        colorLabel = "Pink";
                    break;
```

```
                        case 8:
                            colorLabel = "Grey";
                        break;

                        case 9:
                            colorLabel = "Black";
                        break;

                        case 10:
                            colorLabel = "White";
                        break;
                    }
                    streamWriter.print(colorLabel + "");

                    for(column = 0; column < 3; column++)
                    {
                      streamWriter.print(fileColors[row][column] + " ");
                    }
                    streamWriter.println("");
                }
                //stream written and closed
                streamWriter.flush();
                outStream.close();
            }
        }
    }
}
```