```java
/**
 * This application allows the user to generate different colors using sliders
 * representing Red, Green, and Blue values and outputs the color with a circle
 * in the middle of the window as well as the numeric representation and a bar
 * graph visually representing the ratio of component primary colors. The
 * numeric representations can be either decimal, binary, octal, or hexadecimal.
 *
 * @name    ColorFactory (Homework Assignment 06)
 * @author  Ravi S. Ramphal
 * @class   CCSF CS111B
 * @date    2017.07.27
 * @version 1.0
 */

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

class Circle extends Canvas
{
    int circleSize;

    // ================================ paint() ================================

    public void paint (Graphics graphicsContext)
    {
        int x = (getWidth()  / 2) - (circleSize / 2);
        int y = (getHeight() / 2) - (circleSize / 2);
        graphicsContext.fillOval(x, y, circleSize, circleSize);
    }

    // =============================== Circle() ===============================

    public Circle (int size)
    {
        circleSize = size;
    }
}

class Bars extends Canvas
{
    final double SCALE = 0.4;

    final int BAR_WIDTH   = (int) (20.0 * SCALE); // relative to 256 then scaled
    final int BAR_GAP     = (int) (10.0 * SCALE); // relative to 256 then scaled

    public int redValue   = 0;
    public int blueValue  = 0;
    public int greenValue = 0;

    int horizontalOffset = 100;
    int verticalOffset = 10;

    // ================================ paint() ================================

    public void paint (Graphics graphicsContext)
    {
        graphicsContext.setColor(Color.RED);
        graphicsContext.fillRect(
            horizontalOffset + BAR_GAP,
            verticalOffset + (int) (255.0 * SCALE) - (int) (redValue * SCALE),
            BAR_WIDTH,
            (int) (redValue * SCALE)
        );

        graphicsContext.setColor(Color.GREEN);
        graphicsContext.fillRect(
            horizontalOffset + (2 * BAR_GAP) + BAR_WIDTH,
            verticalOffset + (int) (255.0 * SCALE) - (int) (greenValue * SCALE),
            BAR_WIDTH,
            (int) (greenValue * SCALE)
        );

        graphicsContext.setColor(Color.BLUE);
        graphicsContext.fillRect(
            horizontalOffset + (3 * BAR_GAP) + (2 * BAR_WIDTH),
            verticalOffset + (int) (255.0 * SCALE) - (int) (blueValue * SCALE),
            BAR_WIDTH,
            (int) (blueValue * SCALE)
        );
    }

    // ============================= updateBars() =============================

    public void updateBars (int red, int green, int blue)
    {
        redValue   = red;
        greenValue = green;
        blueValue  = blue;
        repaint();
    }
}

public class ColorFactory extends Frame implements AdjustmentListener,
                                                   ItemListener
{
    final String TITLE = "Color Factory";
    final int DIRECTION = Scrollbar.HORIZONTAL;
    final int CIRCLE_SIZE = 200;

    Dimension screenSize;
    Dimension windowSize;

    int redValue   = 0;
    int greenValue = 0;
    int blueValue  = 0;

    Label header = new Label(TITLE, Label.CENTER);

    Panel columns = new Panel();

    Panel         outputPanel    = new Panel();
    CheckboxGroup outputGroup    = new CheckboxGroup();
    Checkbox      decimal        = new Checkbox("Decimal", outputGroup, true );
    Checkbox      binary         = new Checkbox("Binary",  outputGroup, false);
    Checkbox      octal          = new Checkbox("Octal",   outputGroup, false);
    Checkbox      hex            = new Checkbox("Hex",     outputGroup, false);
    Label         output         = new Label(getOutput(), Label.CENTER);

    Circle display = new Circle(CIRCLE_SIZE);

    Panel rgbPanel = new Panel();

    Panel sliderPanel = new Panel();

    Label     redLabel  = new Label("Red  ", Label.RIGHT);
    Scrollbar redSlider = new Scrollbar(DIRECTION, redValue, 1, 0, 256);
    Label     redNumber = new Label("  " + redValue, Label.LEFT);

    Label     greenLabel  = new Label("Green  ", Label.RIGHT);
    Scrollbar greenSlider = new Scrollbar(DIRECTION, greenValue, 1, 0, 256);
    Label     greenNumber = new Label("  " + greenValue, Label.LEFT);

    Label     blueLabel  = new Label("Blue  ", Label.RIGHT);
    Scrollbar blueSlider = new Scrollbar(DIRECTION, blueValue, 1, 0, 256);
    Label     blueNumber = new Label("  " + blueValue, Label.LEFT);

    Panel barsPanel = new Panel();
    Bars  bars      = new Bars();

    // ============================== leftPad() ================================

    private static String leftPad (String input, int width, char padder)
    {
        // cache number of characters in input
        int inputWidth = input.length();

        // if user desires a width shorter than input width, return unchanged
        if (inputWidth >= width) return input;

        // initialize a StringBuilder with capacity set to desired width
        StringBuilder output = new StringBuilder(width);

        // repeat characters to fill missing width
        for (int i = 0; i < (width - inputWidth); i++)
        {
            output.append(padder);
        }

        // append original input to end of repeated characters and return String
        return output.append(input).toString();
    }

    // ============================ setWindowSize() ============================

    private void setWindowSize ()
    {
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        screenSize = toolkit.getScreenSize();
        windowSize = new Dimension(screenSize.width / 2, screenSize.height / 3);
        setSize(windowSize);
    }

    // ============================= setupOutput() =============================

    private void setupOutput ()
    {
        decimal.addItemListener(this);
        binary.addItemListener(this);
        octal.addItemListener(this);
        hex.addItemListener(this);

        output.setFont(new Font("Dialog", Font.BOLD, 12));

        outputPanel.setLayout(new GridLayout(5, 1, 0, 0));

        outputPanel.add(decimal);
        outputPanel.add(binary);
        outputPanel.add(octal);
        outputPanel.add(hex);
        outputPanel.add(output);

        columns.add(outputPanel);
    }

    // ============================ setupDisplay() ============================

    private void setupDisplay ()
    {
        columns.add(display);
    }

    // ============================== setupRGB() ==============================

    private void setupRGB ()
    {
        Font rgbFont = new Font("Dialog", Font.BOLD, 12);

        sliderPanel.setLayout(new GridLayout(3, 3));

        redLabel.setFont(rgbFont);
        redSlider.addAdjustmentListener(this);
        redNumber.setFont(rgbFont);
        sliderPanel.add(redLabel);
        sliderPanel.add(redSlider);
        sliderPanel.add(redNumber);

        greenLabel.setFont(rgbFont);
        greenSlider.addAdjustmentListener(this);
        greenNumber.setFont(rgbFont);
        sliderPanel.add(greenLabel);
        sliderPanel.add(greenSlider);
        sliderPanel.add(greenNumber);

        blueLabel.setFont(rgbFont);
        blueSlider.addAdjustmentListener(this);
        blueNumber.setFont(rgbFont);
        sliderPanel.add(blueLabel);
        sliderPanel.add(blueSlider);
        sliderPanel.add(blueNumber);

        rgbPanel.setLayout(new GridLayout(2, 1, 0, 0));

        rgbPanel.add(sliderPanel);
        rgbPanel.add(bars);

        columns.add(rgbPanel);
    }

    // ========================== updateColorValues() ==========================

    public void updateColorValues ()
    {
        redValue   = redSlider.getValue();
        greenValue = greenSlider.getValue();
        blueValue  = blueSlider.getValue();
    }

    // ============================ updateNumbers() ============================

    public void updateNumbers ()
    {
        redNumber.setText("" + redValue);
        greenNumber.setText("" + greenValue);
        blueNumber.setText("" + blueValue);
    }

    // ========================= updateDisplayColor() =========================

    public void updateDisplayColor ()
    {
        display.setForeground(new Color(redValue, greenValue, blueValue));
    }

    // ============================= updateBars() =============================

    public void updateBars ()
    {
        bars.updateBars(redValue, greenValue, blueValue);
    }

    // ============================= getOutput() =============================

```

```java
    public String getOutput ()
    {
        Checkbox selectedOutput = outputGroup.getSelectedCheckbox();

        if (selectedOutput == decimal)
        {
            return (
                redValue   + ", " +
                greenValue + ", " +
                blueValue
            );
        }
        else if (selectedOutput == binary)
        {
            return (
                leftPad(Integer.toBinaryString(redValue),   8, '0') + ", " +
                leftPad(Integer.toBinaryString(greenValue), 8, '0') + ", " +
                leftPad(Integer.toBinaryString(blueValue),  8, '0')
            );
        }
        else if (selectedOutput == octal)
        {
            return (
                Integer.toOctalString(redValue)   + ", " +
                Integer.toOctalString(greenValue) + ", " +
                Integer.toOctalString(blueValue)
            );
        }
        else if (selectedOutput == hex)
        {
            return (
                "#" +
                leftPad(Integer.toHexString(redValue),   2, '0') +
                leftPad(Integer.toHexString(greenValue), 2, '0') +
                leftPad(Integer.toHexString(blueValue),  2, '0')
            ).toUpperCase();
        }

        return "";
    }

    // ============================ updateOutput() ============================

    public void updateOutput ()
    {
        output.setText(getOutput());
    }

    // ======================= adjustmentValueChanged() =======================

    public void adjustmentValueChanged (AdjustmentEvent event)
    {
        updateColorValues();
        updateOutput();
        updateDisplayColor();
        updateNumbers();
        updateBars();
    }

    // ========================== itemStateChanged() ==========================

    public void itemStateChanged (ItemEvent event)
    {
        updateOutput();
    }

    // ============================ ColorFactory() ============================

    public ColorFactory ()
    {
        setWindowSize();
        setTitle(TITLE);
        setLayout(new BorderLayout(0, 0));

        header.setFont(new Font("Dialog", Font.BOLD, 24));
        add(header, BorderLayout.PAGE_START);

        columns.setLayout(new GridLayout(1, 3));
        add(columns, BorderLayout.CENTER);

        setupOutput();
        setupDisplay();
        setupRGB();

        setVisible(true);
    }

    // ============================= closeFrame() =============================

    public static void closeFrame (Frame frame)
    {
        frame.addWindowListener(new WindowAdapter ()
        {
            public void windowClosing (WindowEvent event)
            {
                System.exit(0);
            }
        });
    }

    // ================================ main() ================================

    public static void main (String ... args) {
        ColorFactory colorFactory = new ColorFactory();
        closeFrame(colorFactory);
    }
}
```