Javaによる色のプログラミング (第 3 回)

氏名　越智 秀次
クラス　理 科 一 類 11 組
学生証番号 J4-170235

▫課題3.1 - 3.2節 例 1: 光の三原色ーAdditiveColor.java

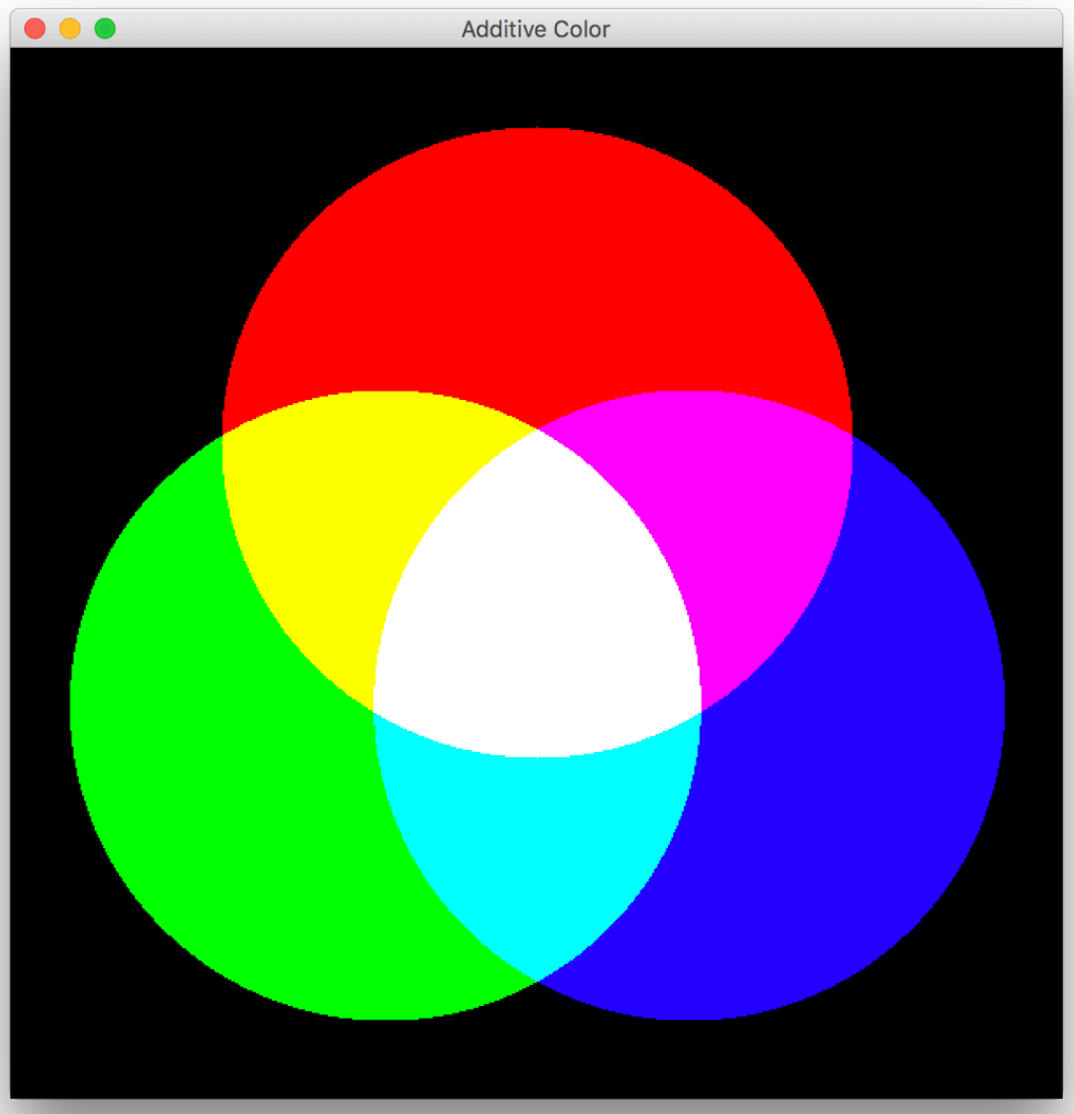
○プログラムリスト

略

○実行コマンド

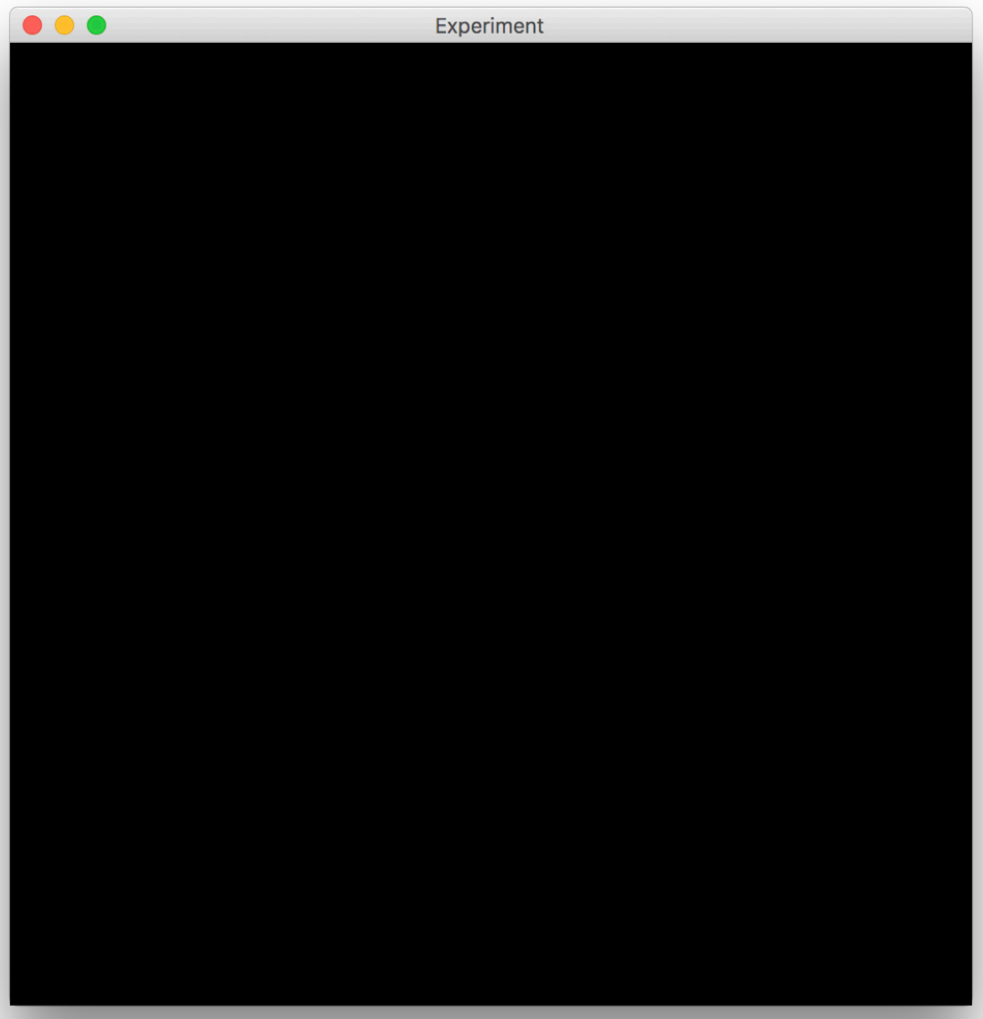
```
ochihidejinoMacBook-Pro:Chap03 ochihideji$ java AdditiveColor
```

○実行結果



○考察

本プログラミングのコンストラクタのRGB値は0.0以上1.0以下の数値で指定するので、プログラム中の配列colorは当然float型で定義されなければならない。例えば、これをint型で定義すると下のように真っ黒な画面に何も表示されない。



▫課題3.2 - 3.2節 例 2: 色相円ーColorRingRGB.java

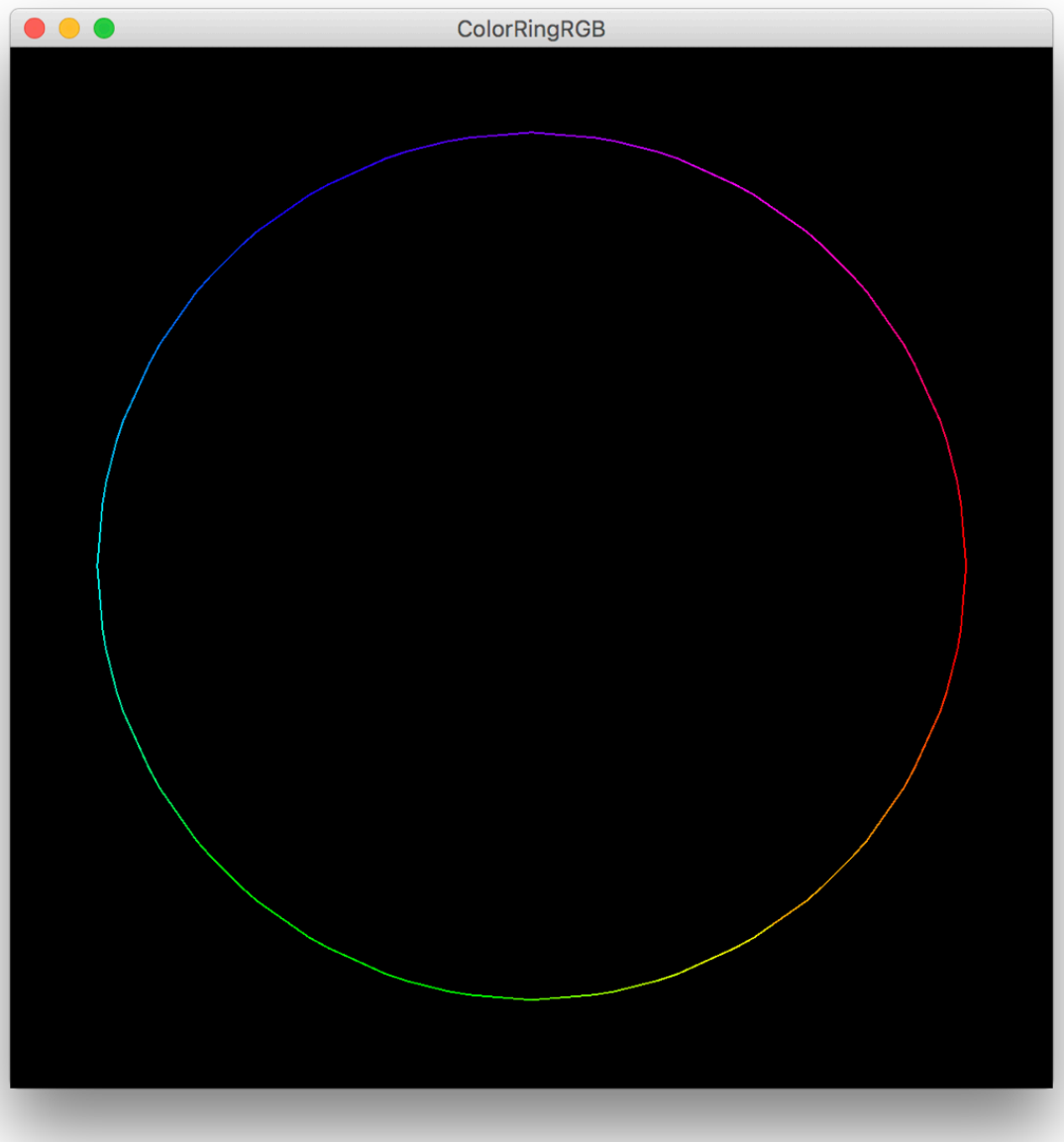
○プログラムリスト

略

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap03 ochihideji$ java ColorRingRGB 128
```

○実行結果



○考察

教科書とs同じプログラミングを書いているにも関わらず、異なる描画がなされてしまうことがあった。

▫課題3.3 - 3.2節 例 3: HSB表現と色円盤ーColorDisk.java

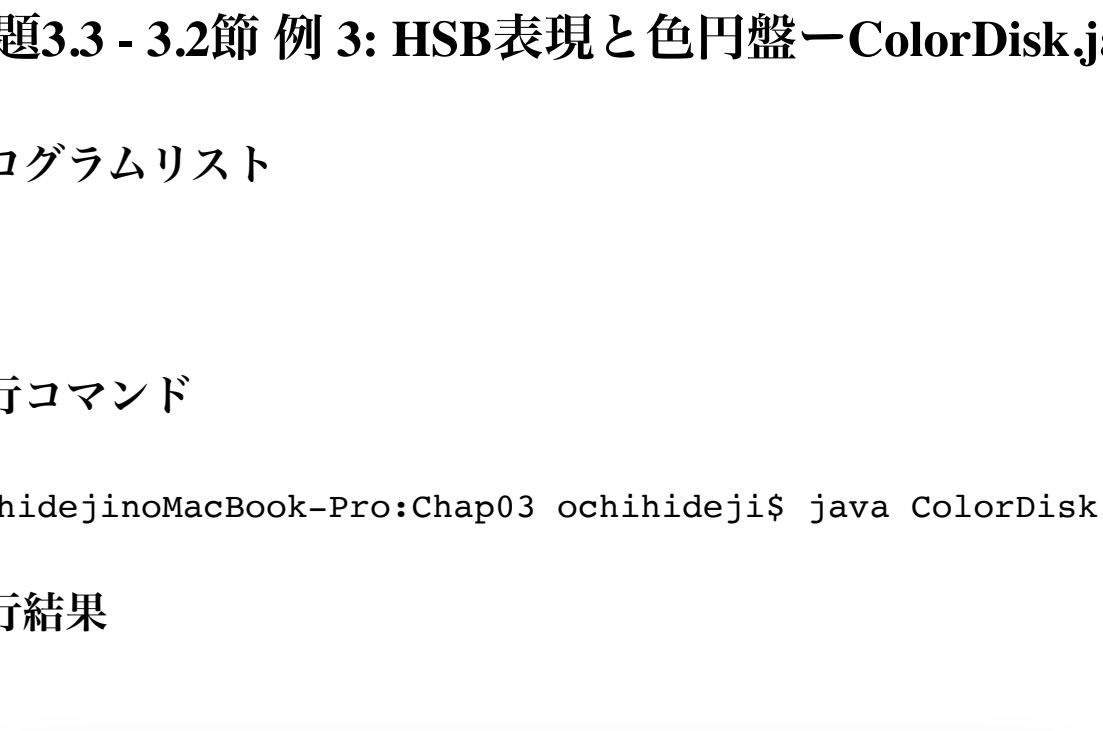
○プログラムリスト

略

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap03 ochihideji$ java ColorDisk
```

○実行結果



○考察

sとbは0と1の間の浮動小数点値にする必要がある。そのため、38行目にあるようにs＞1 となるような点ではs=0に固定しておく必要がある。

▫課題3.3 - 章末問題：色の三原色(減法混色/CMY表現)

○プログラムリスト

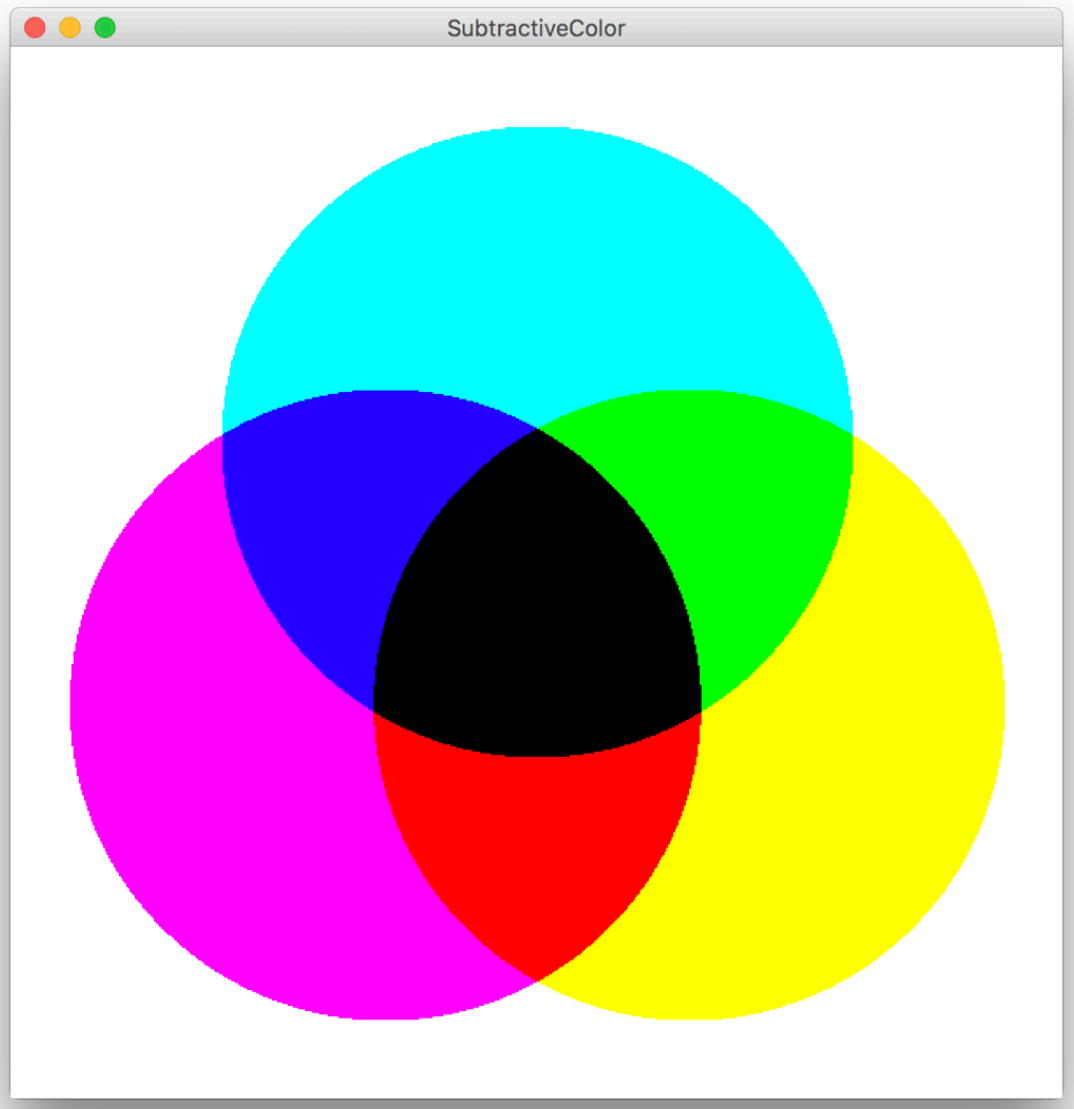
```
import java.awt.*;
import java.awt.event.*;

public class SubtractiveColor extends Canvas {
    public static void main(String[] args) {
        new SubtractiveColor("SubtractiveColor");
    }

    protected static final int width = 600;
    protected static final int height = 600;
    protected static final double radius2 = 180.0 * 180.0;

    protected double [][] centers = {{300.0,225.0}, {213.4, 375.0}, {386.6, 375.0}};

    protected SubtractiveColor(String name) {
        super();
        setSize(width, height);

        Frame f = new Frame(name);
        f.add(this);
        f.pack();
        f.addWindowListener(new WindowAdapter() {
            public void WindowClosing(WindowEvent e) {
```

```
                System.exit(0);
            }
        });
        f.setVisible(true);
    }

    public void paint(Graphics g) {
        for (int y = 0; y < height; y++) {
            for (int x = 0; x < width; x++) {
                float[] color = new float[3];
                for (int i = 0; i < 3; i++) {
                    double dist2 = (x - centers[i][0]) * (x - centers[i][0]) + (y - centers[i][1]) * (y - centers[i][1]);
                    color[i] = (dist2 < radius2) ? 0.0f : 1.0f;

                }
                g.setColor(new Color(color[0], color[1], color[2]));

                g.fillRect(x, y, 1, 1);
            }
        }
    }
}
```

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap03 ochihideji$ java SubtractiveColor
```

○実行結果



○考察

教科書16ページ (3.1)式を考慮すれば、減法混色のプログラミングは加法混色のプログラミングの36行目を変えるだけで良い。すなわち、各円盤に対応するRGBの値が円盤内部では0に、外部では1になるようにすれば良い。(加法混色の場合は逆であった)

□課題3.4 - 章末課題：HSB表現を用いた色相円

○プログラムリスト

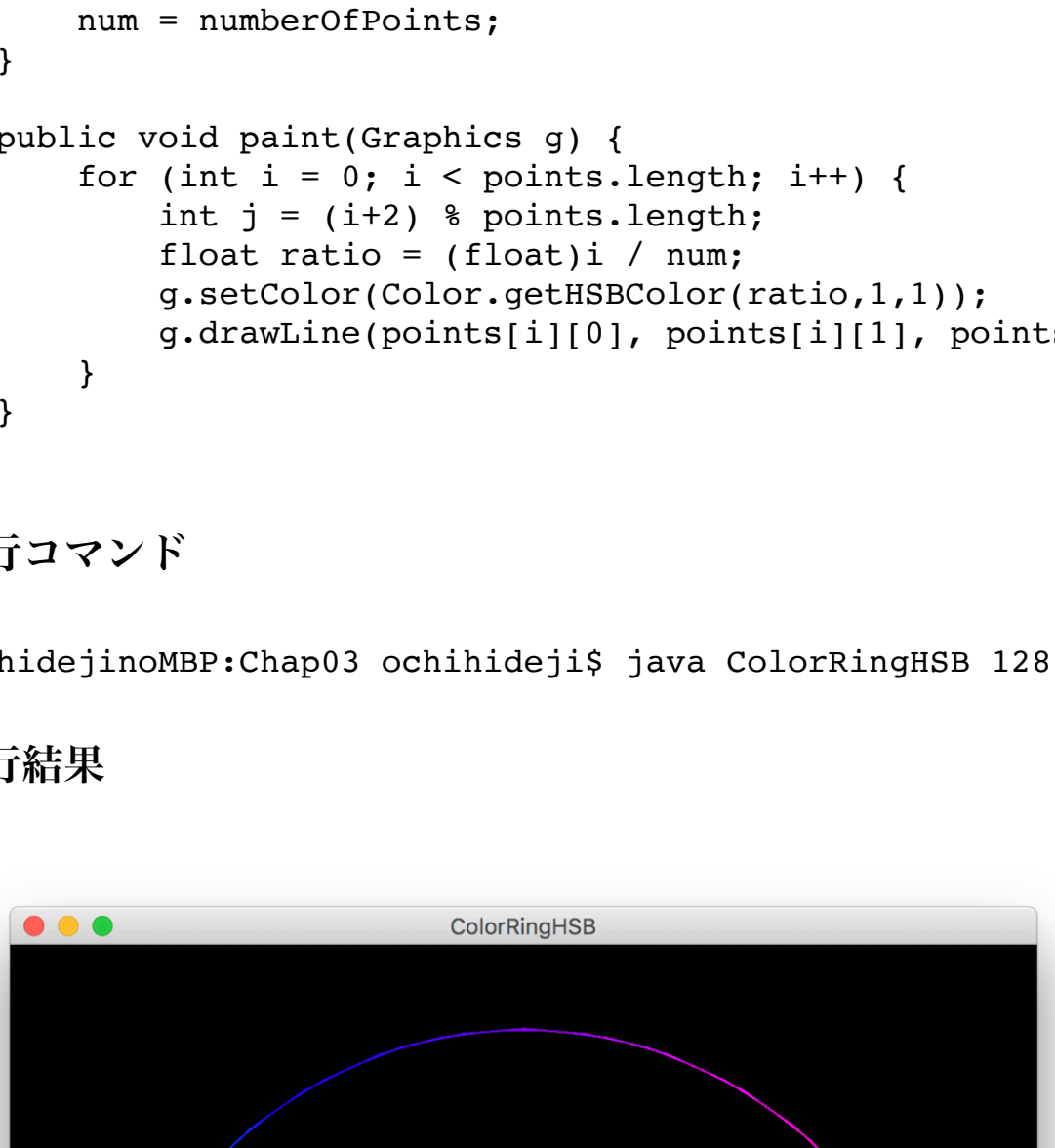
```
import java.awt.*;

public class ColorRingHSB extends Circle {
    public static void main(String[] args) {
        if (args.length == 0)
          System.err.println("Usage: java ColorRingHSB #");
        else
```

```
      new ColorRingHSB("ColorRingHSB", Integer.parseInt(args[0]));
}

protected Color[] colors;
private int num;

protected ColorRingHSB(String name, int numberOfPoints) {
    super(name, numberOfPoints);
    setBackground(Color.black);
    setColor(numberOfPoints);
}

protected void setColor(int numberOfPoints) {
    num = numberOfPoints;
}

public void paint(Graphics g) {
    for (int i = 0; i < points.length; i++) {
        int j = (i+2) % points.length;
        float ratio = (float)i / num;
        g.setColor(Color.getHSBColor(ratio,1,1));
        g.drawLine(points[i][0], points[i][1], points[j][0], points[j][1]);
    }
}
```

○実行コマンド

```
ochihidejinoMBP:Chap03 ochihideji$ java ColorRingHSB 128
```

○実行結果



○考察

同じ色相円を描くのにも、HSB表現を使えばよりプログラムが簡単に書けるのが面白いと思った。

□課題3.5 - 章末問題：明度を変化させた色円盤

○プログラムリスト

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

public class ColorDisk2 extends Canvas {
    public static void main(String[] args) {
        new ColorDisk2("ColorDisk2");
    }
```

```
protected static final int width = 600;
protected static final int height = 600;
protected static final int centerX = 300;
protected static final int centerY = 299;
protected static final int radius = 250;

protected ColorDisk2(String name) {
    super();
    setSize(width, height);

    Frame f = new Frame(name);
    f.add(this);
    f.pack();
    f.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            System.exit(0);
        }
    });
    f.setVisible(true);
}
    public void paint(Graphics g) {
        for (int y = 0; y < height; y++) {
            for (int x = 0; x < width; x++) {
                double dx = (double)(x - centerX);
                double dy = (double)(y - centerY);
                float h = (float)(Math.atan2(dy,dx) /( 2.0 * Math.PI));
                float b = (float)(Math.sqrt(dx*dx + dy*dy) / radius);
                if (b > 1.0f) b = 0.0f;
                float s = (b > 1.0f) ? 0.0f : 1.0f;
                g.setColor(Color.getHSBColor(h,s,b));
                g.fillRect(x, y, 1, 1);
            }
        }
    }
}
```

○実行コマンド

```
ochihidejinoMBP:Chap03 ochihideji$ java ColorDisk2
```

○実行結果

○考察

当初37行目の記述(円の外でb=0)を忘れていたため、円の外でも明度が変化する非常に不自然な図が描かれてしまった。

□課題3.6 - 自習課題：ColorConstancy.java

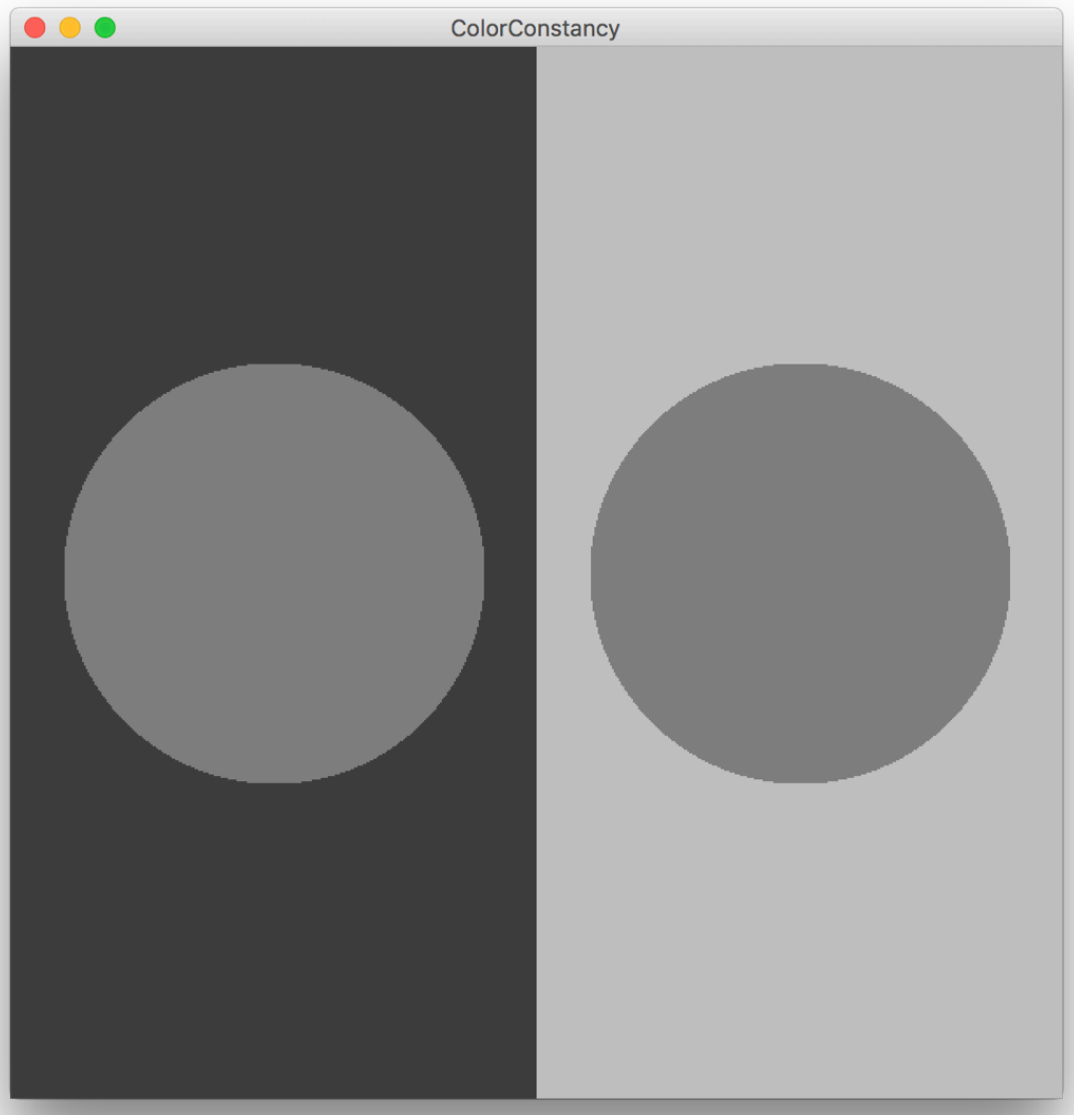
○プログラムリスト

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

public class ColorConstancy extends Canvas{
    public static void main(String args[]){
        new ColorConstancy("ColorConstancy");
    }

    private static final int width = 600;
    private static final int height = 600;
    private static int centerX1 = 150;
    private static int centerX2 = 450;
    private static int centerY = 300;
    private static int radius2 = 120 * 120;

    protected ColorConstancy(String name) {
        super();
        setSize(width, height);
        Frame f = new Frame(name);
        f.add(this);
        f.pack();
        f.addWindowListener(new WindowAdapter() {
            public void WindowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
        f.setVisible(true);
    }

    public void paint(Graphics g) {
        for (int y = 0; y < height; y++) {
            final int dy = (y - centerY) * (y - centerY);
            for (int x = 0; x < 300; x++) {
                final int dx = (x - centerX1) * (x - centerX1);
                if (dx + dy < radius2)
                  g.setColor(Color.gray);
                else
                  g.setColor(Color.darkGray);
                g.fillRect(x, y, 1, 1);
            }
            for (int x = 300; x < 600; x++){
                final int dx = (x - centerX2) * (x - centerX2);
                if (dx + dy < radius2)
                  g.setColor(Color.gray);
                else
                  g.setColor(Color.lightGray);
                g.fillRect(x, y, 1, 1);
            }
        }
    }
}
```

○実行コマンド

```
ochihidejinoMBP:Chap03 ochihideji$ java ColorConstancy
```

○実行結果



○考察

授業の最後で話があったcolor constancyについて再現してみようと試みた。２つの背景色の明度の差異や２つの円盤同士の距離などがうまく錯視されるか否かに関係していると考え、様々な組み合わせを試してみた。一番上手くいったものをここにあげている。

□課題や授業に関して

○レポート作成に要した時間

2時間程度

○特に苦労した点

「HSB表現を用いた色相円」がなかなかうまくいかず、苦労した。レポート作成に要した時間のほとんどがこの課題に費やされた。

○授業についての感想や希望

色と混色法については軽く図形科学Aで触れたことがありましたが、ここまで詳しくは勉強していなかったので、この章は非常に興味深かったです。