フラクタル図形 (第 6 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6.1 - 6.2節 例 1: フラクタル図形ーFractal.java

○プログラムリスト

略

○考察

今まで抽象クラスの意味や使う意義がよくわからなかったが、このFractal.javaによって理解することができたのでよかった。

▫課題6.2 - 6.2節 例 2: コッホ曲線ーKoch.java

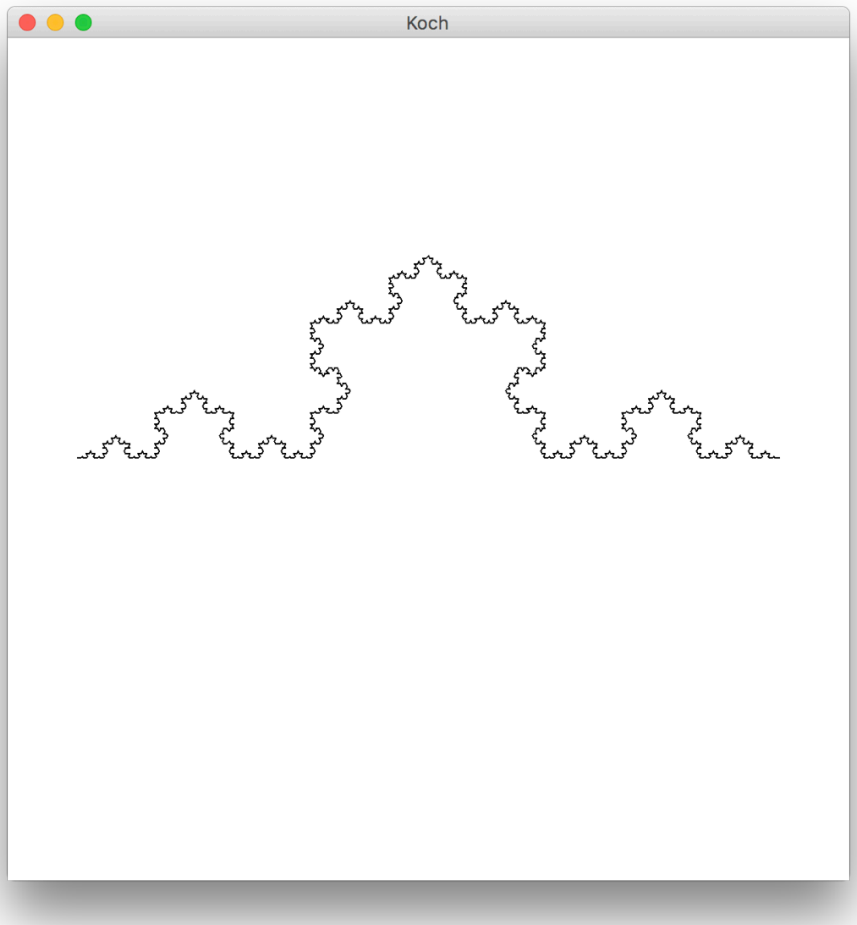
○プログラムリスト

略

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap06 ochihideji$ java Koch 5
```

○実行結果



○考察

コッホ雪片の場合、曲線の長さは無限大に発散するが、曲線によって囲まれる部分の面積は$2*3^{1/2}/5$に収束する。これは簡単な無限級数の計算によって求められる。(大学入試の数学でも往々に扱われるトピックである。)

▫課題6.3 - 6.2節 例 3: シェルピンスキー三角形ーSierpinski.java

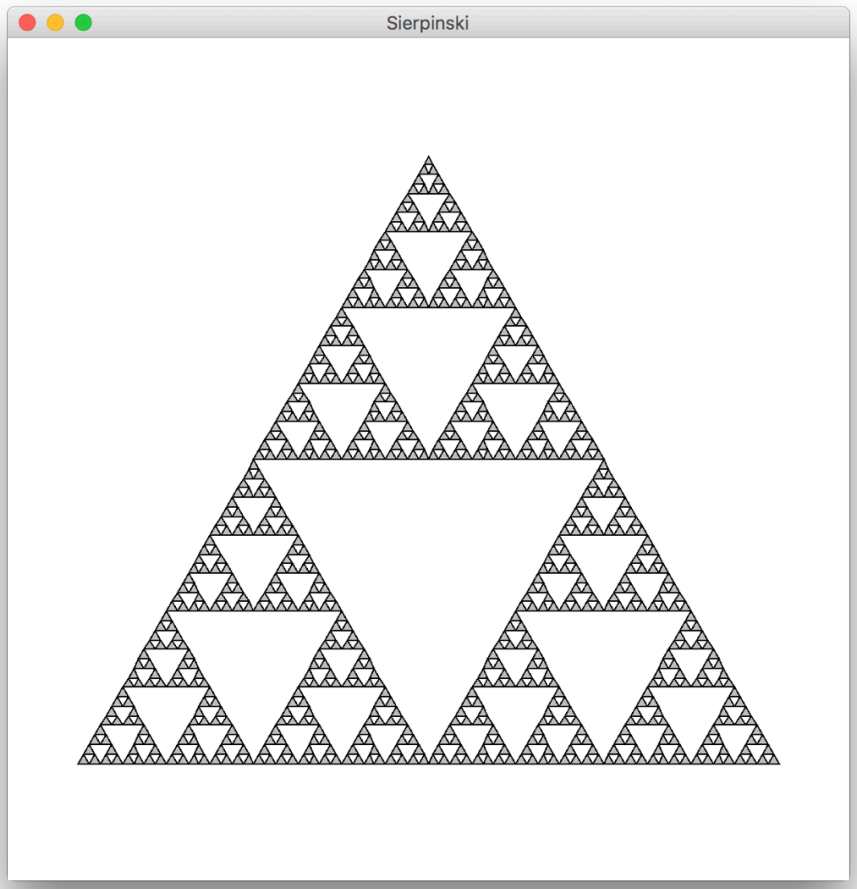
○プログラムリスト

略

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap06 ochihideji$ java Sierpinski 6
```

○実行結果



○考察

1回のステップによって、最も小さい三角形の一片の長さは1/2倍になり要素数は3倍になるので、ハウスドルフ次元は$\log_2 3$=1.584962501となる。

□課題6.4 - 章末課題：ドラゴン曲線ーDragon.java

○プログラムリスト

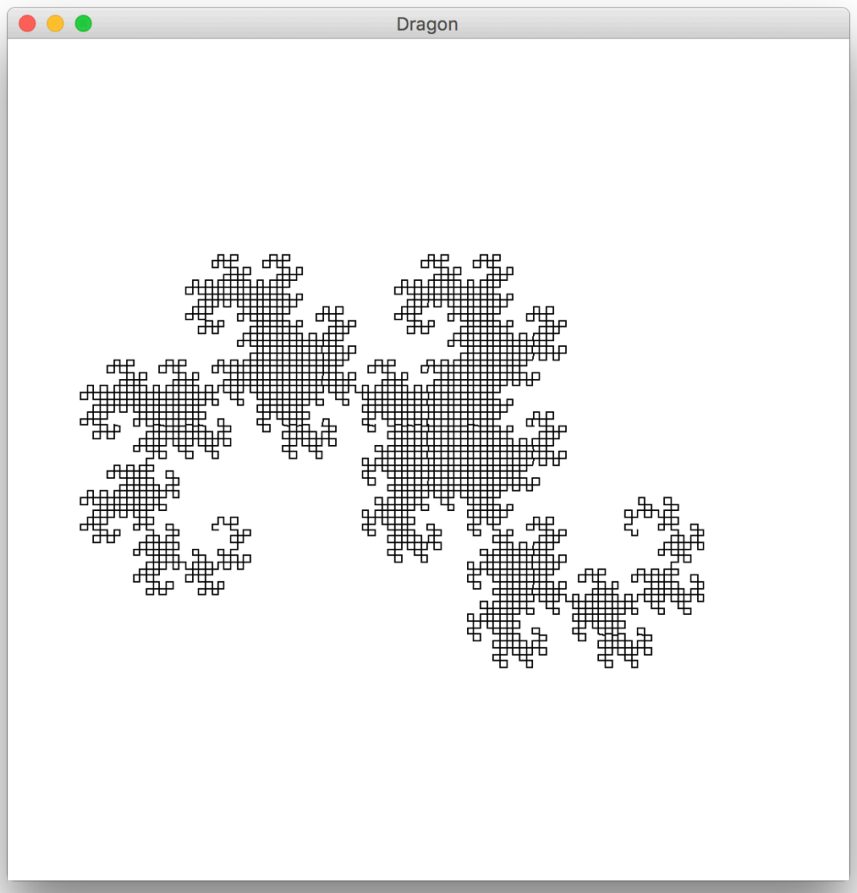
```
import java.awt.*;

public class Dragon extends Fractal {
    public static void main(String[] args) {
        if (args.length != 1)
          System.err.println("Usage: Dragon #iteration");
        else
          (new Dragon("Dragon", Integer.parseInt(args[0]))).showFrame();
    }

    protected Dragon(String name, int times) {
        super(name, times);
    }

    protected void setData() {
        setOrigin(150, 350);
        setRange(2.0);

        initPoints = new Vector2D[2];
        initPoints[0] = new Vector2D(0.0, 0.0);
        initPoints[1] = new Vector2D(1.0, 0.0);

        m = new Matrix2X2[2];
        v = new Vector2D[2];
        m[0] = Matrix2X2.scale(1.0 / Math.sqrt(2.0)).multiply(Matrix2X2.rotate(Math.PI / 4));
        m[1] = Matrix2X2.scale(1.0 / Math.sqrt(2.0)).multiply(Matrix2X2.rotate(3 * Math.PI / 4));
        v[0] = initPoints[0];
        v[1] = m[0].apply(initPoints[1]).subtract(m[1].apply(initPoints[1]));
    }
```

```
    protected void drawObject(Graphics g, Vector2D[] points) {
        drawPolyline(g, points);
    }
}
```

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap06 ochihideji$ java Dragon 12
```

○実行結果



○考察

1回のステップで線分の長さは1/sqrt(2)倍に、要素の数は2倍になるので、ハウスドルフ次元は $\log_{2^{1/2}}2=2$ これは曲面(二次元)のハウスドルフ次元と等しい、すなわち曲面(二次元)と同じ程度の「複雑さ」を持つと言える。

□課題6.5 - 章末課題：パスカル三角形(mod 4)ーPascalMod4.java

○プログラムリスト

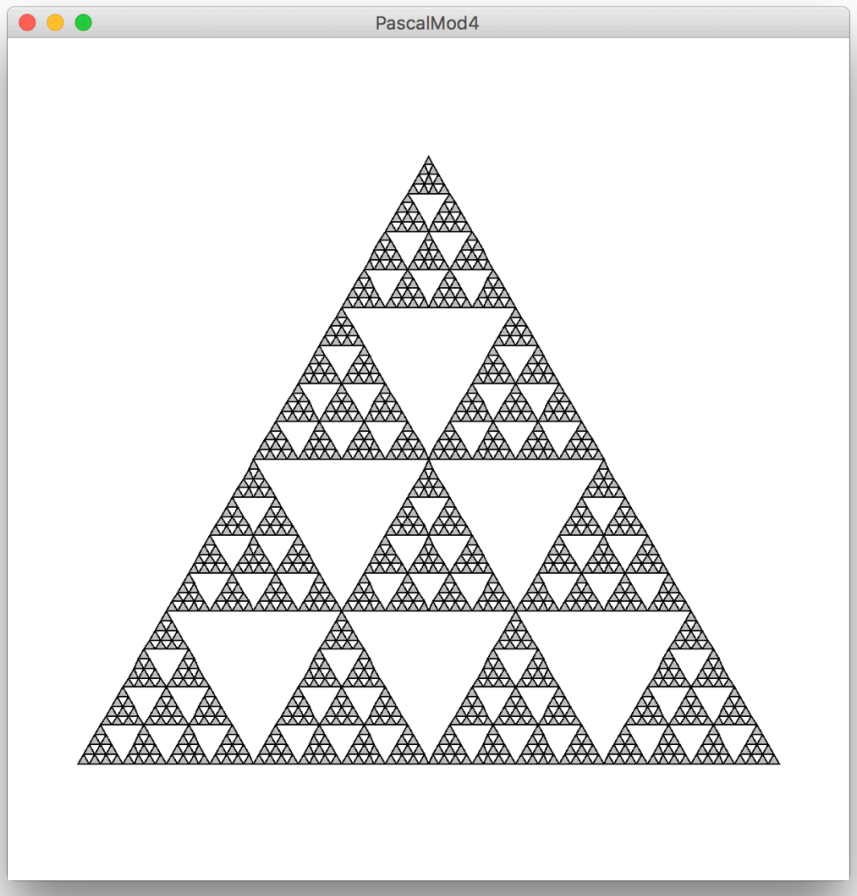
```
import java.awt.*;

public class PascalMod4 extends Fractal {
    public static void main(String[] args) {
        if (args.length != 1)
          System.err.println("Usage: PsacalMod4 #iteration");
        else
          (new PascalMod4("PascalMod4", Integer.parseInt(args[0]))).showFrame();
    }

    protected PascalMod4(String name, int times) {
        super(name, times);
    }
```

```
    protected void setData() {
        setOrigin(300, 84);
        setRange(2.4);

        initPoints = new Vector2D[3];
        initPoints[0] = new Vector2D(0.0, 0.0);
        initPoints[1] = new Vector2D(-1.0, -Math.sqrt(3.0));
        initPoints[2] = new Vector2D(1.0, -Math.sqrt(3.0));

        m = new Matrix2X2[10];
        v = new Vector2D[10];
        m[0] = Matrix2X2.scale(1.0 / 4.0);
        m[1] = Matrix2X2.scale(1.0 / 4.0);
        m[2] = Matrix2X2.scale(1.0 / 4.0);
        m[3] = Matrix2X2.scale(1.0 / 4.0);
        m[4] = Matrix2X2.scale(1.0 / 4.0);
        m[5] = Matrix2X2.scale(1.0 / 4.0);
        m[6] = Matrix2X2.scale(1.0 / 4.0);
        m[7] = Matrix2X2.scale(1.0 / 4.0);
        m[8] = Matrix2X2.scale(1.0 / 4.0);
        m[9] = Matrix2X2.scale(1.0 / 4.0);
        v[0] = initPoints[0];
        v[1] = m[1].apply(initPoints[1]).add(v[0]);
        v[2] = m[2].apply(initPoints[2]).add(v[0]);
        v[3] = m[3].apply(initPoints[1]).add(v[1]);
        v[4] = m[4].apply(initPoints[2]).add(v[2]);
        v[5] = m[5].apply(initPoints[1]).add(v[3]);
        v[6] = m[6].apply(initPoints[2]).add(v[4]);
        v[7] = m[7].apply(initPoints[2]).add(v[1]);
        v[8] = m[8].apply(initPoints[2]).add(v[3]);
        v[9] = m[9].apply(initPoints[1]).add(v[4]);
    }

    protected void drawObject(Graphics g, Vector2D[] points) {
        g.setColor(Color.lightGray);
        fillPolygon(g, points);
        g.setColor(Color.black);
        drawPolygon(g, points);
    }
}
```

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap06 ochihideji$ open PascalMod4.java
```

○実行結果



○考察

1回のステップで全体を1/4にした図形が10個できるので、ハウスドルフ次元は$\log_4 10$=1.660964047となる。直感的にはシェルピンスキー三角形(パスカルの三角形のMod2の場合)よりもこちらの図形の方が複雑であると感じられるが、実際にハウスドルフ次元を比べると、シェルピンスキー三角形は1.584962501(<1.660964047)なのでこちらの図形の方が複雑であることが確かめられる。

□課題6.6 - 章末課題：シェルピンスキーカーペットーSKarpet.java

○プログラムリスト

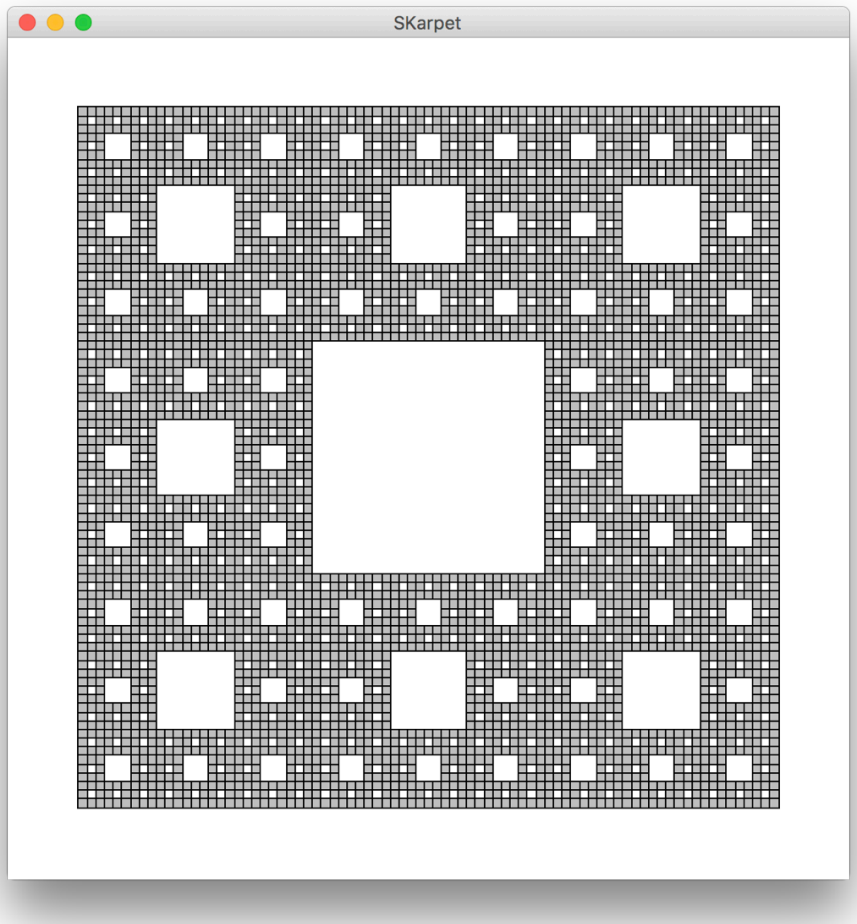
```
import java.awt.*;

public class SKarpet extends Fractal {
    public static void main(String[] args) {
        if (args.length != 1)
          System.err.println("Usage: java SKarpet #iteration");
        else
          (new SKarpet("SKarpet", Integer.parseInt(args[0]))).showFrame();
    }

    protected SKarpet(String name, int times) {
        super(name, times);
    }

    protected void setData() {
        setOrigin(300, 299);
        setRange(1.2);

        initPoints = new Vector2D[4];
        initPoints[0] = new Vector2D( 0.5,  0.5);
        initPoints[1] = new Vector2D(-0.5,  0.5);
        initPoints[2] = new Vector2D(-0.5, -0.5);
        initPoints[3] = new Vector2D( 0.5, -0.5);

        m = new Matrix2X2[8];
```

```
        v = new Vector2D[8];
        m[0] = Matrix2X2.scale(1.0 / 3.0);
        m[1] = Matrix2X2.scale(1.0 / 3.0);
        m[2] = Matrix2X2.scale(1.0 / 3.0);
        m[3] = Matrix2X2.scale(1.0 / 3.0);
        m[4] = Matrix2X2.scale(1.0 / 3.0);
        m[5] = Matrix2X2.scale(1.0 / 3.0);
        m[6] = Matrix2X2.scale(1.0 / 3.0);
        m[7] = Matrix2X2.scale(1.0 / 3.0);
        v[0] = m[0].multiply(Matrix2X2.scale(2.0)).apply(initPoints[0]);
        v[1] = m[1].apply(initPoints[1].subtract(initPoints[0])).add(v[0]);
        v[2] = m[2].multiply(Matrix2X2.scale(2.0)).apply(initPoints[1]);
        v[3] = m[3].apply(initPoints[2].subtract(initPoints[1])).add(v[2]);
        v[4] = m[4].multiply(Matrix2X2.scale(2.0)).apply(initPoints[2]);
        v[5] = m[5].apply(initPoints[3].subtract(initPoints[2])).add(v[4]);
        v[6] = m[6].multiply(Matrix2X2.scale(2.0)).apply(initPoints[3]);
        v[7] = m[7].apply(initPoints[0].subtract(initPoints[3])).add(v[6]);

    }

    protected void drawObject(Graphics g, Vector2D[] points) {
        g.setColor(Color.lightGray);
        fillPolygon(g, points);
        g.setColor(Color.black);
        drawPolygon(g, points);
    }
}
```

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap06 ochihideji$ java SKarpet 4
```

○実行結果



○考察

1回のステップで全体を1/3倍にした図形が8個できるので、ハウスドルフ次元は$\log_3 8=1.892789261$となる。

▫課題6.7 - 章末課題：五角形片ーPentaflake.java

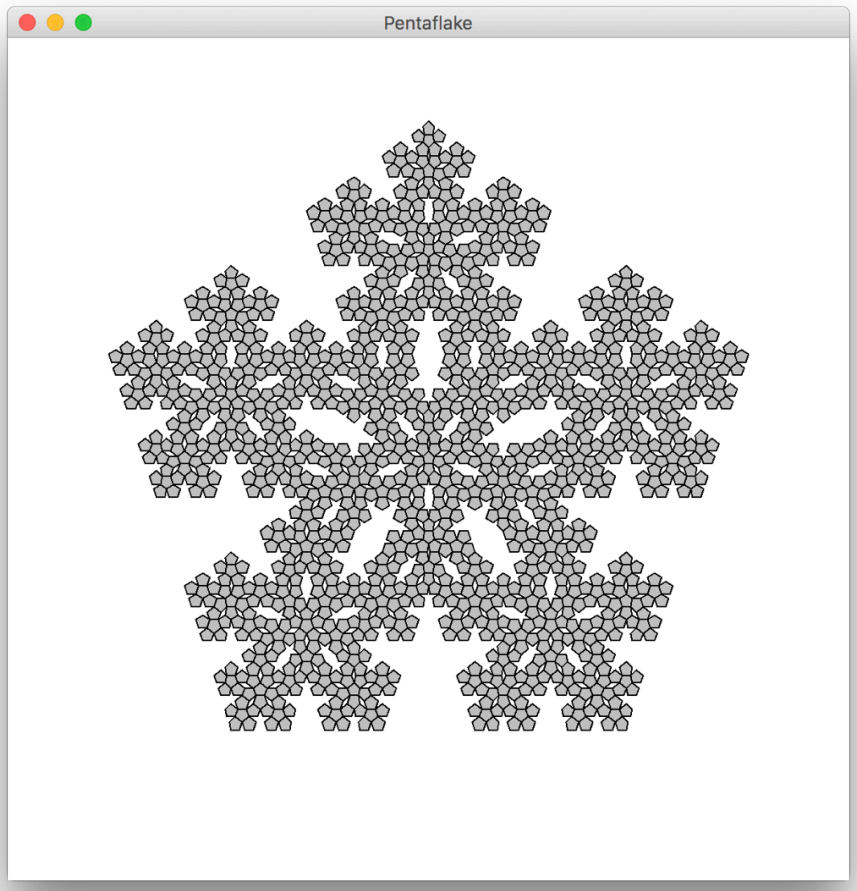
○プログラムリスト

```
import java.awt.*;

public class Pentaflake extends Fractal {
    public static void main(String[] args) {
        if (args.length != 1)
          System.err.println("Usage: java Pentaflake #iteration");
        else
          (new Pentaflake("Pentaflake", Integer.parseInt(args[0]))).showFrame();
    }

    protected Pentaflake(String name, int times) {
        super(name, times);
    }

    protected void setData() {
        setOrigin(300, 299);
        setRange(2.5);

        initPoints = new Vector2D[5];
        initPoints[0] = new Vector2D(Math.cos(Math.PI / 10), Math.sin(Math.PI / 10));
        initPoints[1] = new Vector2D(0.0, 1.0);
        initPoints[2] = new Vector2D(Math.cos(9 * Math.PI / 10), Math.sin(9 * Math.PI / 10));
        initPoints[3] = new Vector2D(Math.cos(13 * Math.PI / 10), Math.sin(13 * Math.PI / 10));
        initPoints[4] = new Vector2D(Math.cos(17 * Math.PI / 10), Math.sin(17 * Math.PI / 10));

        m = new Matrix2X2[6];
        v = new Vector2D[6];
        m[0] = Matrix2X2.scale((3 - Math.sqrt(5)) / 2);
        m[1] = Matrix2X2.scale((3 - Math.sqrt(5)) / 2);
        m[2] = Matrix2X2.scale((3 - Math.sqrt(5)) / 2);
        m[3] = Matrix2X2.scale((3 - Math.sqrt(5)) / 2);
        m[4] = Matrix2X2.scale((3 - Math.sqrt(5)) / 2);
        m[5] = Matrix2X2.scale((3 - Math.sqrt(5)) / 2).multiply(Matrix2X2.rotate(Math.PI));
        v[0] = initPoints[0].subtract(m[0].apply(initPoints[0]));
        v[1] = initPoints[1].subtract(m[2].apply(initPoints[1]));
        v[2] = initPoints[2].subtract(m[3].apply(initPoints[2]));
        v[3] = initPoints[3].subtract(m[4].apply(initPoints[3]));
        v[4] = initPoints[4].subtract(m[4].apply(initPoints[4]));
        v[5] = new Vector2D(0, 0);
    }

    protected void drawObject(Graphics g, Vector2D[] points) {
        g.setColor(Color.lightGray);
        fillPolygon(g, points);
        g.setColor(Color.black);
        drawPolygon(g, points);
    }
}
```

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap06 ochihideji$ java Pentaflake 4
```

○実行結果



○考察

1回のステップで全体を$1/\phi^2$(=nとおく)倍したものが6個できるので、ハウスドルフ次元は$\log_n 6$=1.86171596となる。

□課題6.8 - 章末課題：六角形片ーHexaflake.java

○プログラムリスト

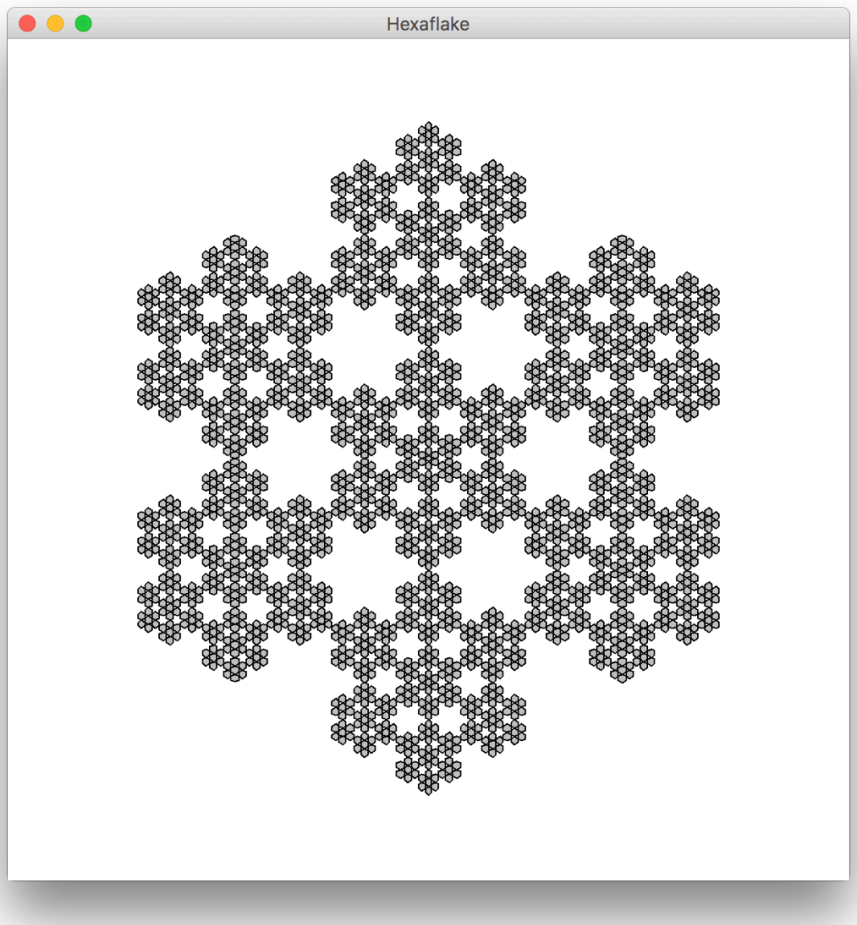
```
import java.awt.*;

public class Hexaflake extends Fractal {
    public static void main(String[] args) {
        if (args.length != 1)
          System.err.println("Usage: java Hexaflake #iteration");
        else
          (new Hexaflake("Hexaflakes", Integer.parseInt(args[0]))).showFrame();
    }

    protected Hexaflake(String name, int times) {
        super(name, times);
    }

    protected void setData() {
        setOrigin(300, 299);
        setRange(2.5);

        initPoints = new Vector2D[6];
        initPoints[0] = new Vector2D(Math.cos(Math.PI / 6), Math.sin(Math.PI / 6));
        initPoints[1] = new Vector2D(Math.cos(Math.PI / 2), Math.sin(Math.PI / 2));
        initPoints[2] = new Vector2D(Math.cos(5 * Math.PI / 6), Math.sin(5 * Math.PI / 6));
        initPoints[3] = new Vector2D(Math.cos(7 * Math.PI / 6), Math.sin(7 * Math.PI / 6));
        initPoints[4] = new Vector2D(Math.cos(3 * Math.PI / 2), Math.sin(3 * Math.PI / 2));
        initPoints[5] = new Vector2D(Math.cos(11 * Math.PI / 6), Math.sin(11 * Math.PI / 6));

        m = new Matrix2X2[7];
        v = new Vector2D[7];
        m[0] = Matrix2X2.scale(1.0 / 3.0);
        m[1] = Matrix2X2.scale(1.0 / 3.0);
```

```
        m[2] = Matrix2X2.scale(1.0 / 3.0);
        m[3] = Matrix2X2.scale(1.0 / 3.0);
        m[4] = Matrix2X2.scale(1.0 / 3.0);
        m[5] = Matrix2X2.scale(1.0 / 3.0);
        m[6] = Matrix2X2.scale(1.0 / 3.0);
        v[0] = m[0].multiply(Matrix2X2.scale(2.0)).apply(initPoints[0]);
        v[1] = m[1].multiply(Matrix2X2.scale(2.0)).apply(initPoints[1]);
        v[2] = m[2].multiply(Matrix2X2.scale(2.0)).apply(initPoints[2]);
        v[3] = m[3].multiply(Matrix2X2.scale(2.0)).apply(initPoints[3]);
        v[4] = m[4].multiply(Matrix2X2.scale(2.0)).apply(initPoints[4]);
        v[5] = m[5].multiply(Matrix2X2.scale(2.0)).apply(initPoints[5]);
        v[6] = new Vector2D(0, 0);
    }

    protected void drawObject(Graphics g, Vector2D[] points) {
        g.setColor(Color.lightGray);
        fillPolygon(g, points);
        g.setColor(Color.black);
        drawPolygon(g, points);
    }
}
```

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap06 ochihideji$ java Hexaflake 4
```

○実行結果



○考察

1回のステップで全体を1/3にした図形が7個できるのでハウスドルフ次元は$\log_3 7=1.771243749$となる。

□課題6.9 - 章末課題：パスカルの三角形(mod 3)ーPascalMod3.java

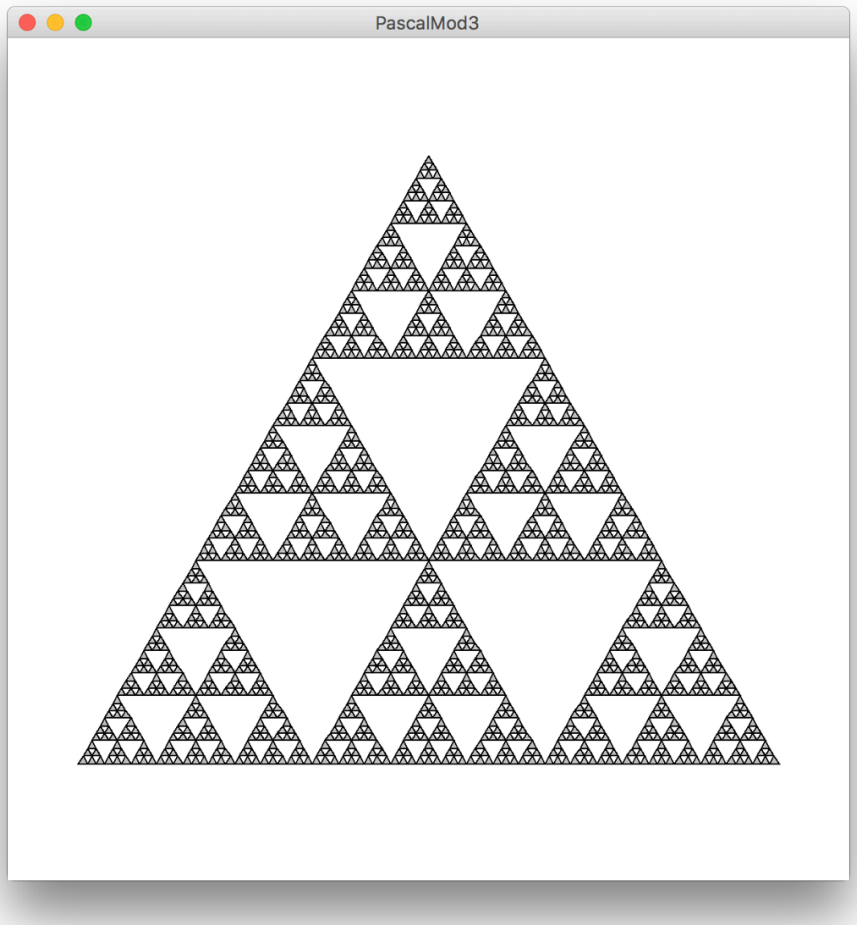
○プログラムリスト

```
import java.awt.*;

public class PascalMod3 extends Fractal {
    public static void main(String[] args) {
        if (args.length != 1)
          System.err.println("Usage: PsacalMod3 #iteration");
        else
          (new PascalMod3("PascalMod3", Integer.parseInt(args[0]))).showFrame();
    }

    protected PascalMod3(String name, int times) {
        super(name, times);
    }

    protected void setData() {
        setOrigin(300, 84);
        setRange(2.4);

        initPoints = new Vector2D[3];
        initPoints[0] = new Vector2D(0.0, 0.0);
        initPoints[1] = new Vector2D(-1.0, -Math.sqrt(3.0));
        initPoints[2] = new Vector2D(1.0, -Math.sqrt(3.0));

        m = new Matrix2X2[6];
        v = new Vector2D[6];
        m[0] = Matrix2X2.scale(1.0 / 3.0);
        m[1] = Matrix2X2.scale(1.0 / 3.0);
        m[2] = Matrix2X2.scale(1.0 / 3.0);
        m[3] = Matrix2X2.scale(1.0 / 3.0);
        m[4] = Matrix2X2.scale(1.0 / 3.0);
        m[5] = Matrix2X2.scale(1.0 / 3.0);
        v[0] = initPoints[0];
        v[1] = m[1].apply(initPoints[1]).add(v[0]);
        v[2] = m[2].apply(initPoints[2]).add(v[0]);
        v[3] = m[3].apply(initPoints[1]).add(v[1]);
        v[4] = m[4].apply(initPoints[2]).add(v[2]);
        v[5] = m[5].apply(initPoints[1]).add(v[2]);
    }

    protected void drawObject(Graphics g, Vector2D[] points) {
        g.setColor(Color.lightGray);
        fillPolygon(g, points);
        g.setColor(Color.black);
        drawPolygon(g, points);
    }
}
```

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap06 ochihideji$ java PascalMod3 4
```

○実行結果



○考察

1回のステップで全体を1/3倍した図形が6個できるので、ハウスドルフ次元は$\log_3 6=1.630929754$となる。これはシェルピンスキー三角形のハウスドルフ次元よりも大きく、Mod=4のパスカルの三角形のハウスドルフ次元よりも小さい。 一般にパスカルの三角形のModをnとしたとき、そのハウスドルフ次元は$\log_n n(n+1)/2=1+\log_n (n+1)/2$となる。

▫課題6.10 - 章末課題：Lシステムのプログラムー LSystem.java

○プログラムリスト

```
import java.awt.*;

public abstract class LSystem extends CGCanvas {
    protected int times;
    protected Vector2D[] initPoints;
    protected Matrix2X2[] m;
    protected Vector2D[] v;

    protected LSystem(String name, int times) {
        super(name);
        this.times = times;
        setData();
    }

    protected abstract void setData();

    public void paint(Graphics g) {
        drawObject(g, initPoints, times);
    }

    protected void drawObject(Graphics g, Vector2D[] points, int l) {
        if (l > 0) {
            for (int i = 0; i < m.length; i++) {
                Vector2D[] newPoints = new Vector2D[points.length];
                for (int j = 0; j < points.length; j++) {
                    newPoints[j] = m[i].apply(points[j]).add(v[i]);
```

```
                }
                drawObject(g, newPoints, l-1);
                drawObject(g, points);
            }
        }
    }

    protected abstract void drawObject(Graphics g, Vector2D[] points);
}
```

○考察

forによる繰り返しの記述の中に図を実際に描画する記述を入れることによって、前の描画を残しつつ新しい描画を書き加えることができる。else文は不要なので消去した。

▫課題6.11 - 章末課題：木の成長ーTree.java

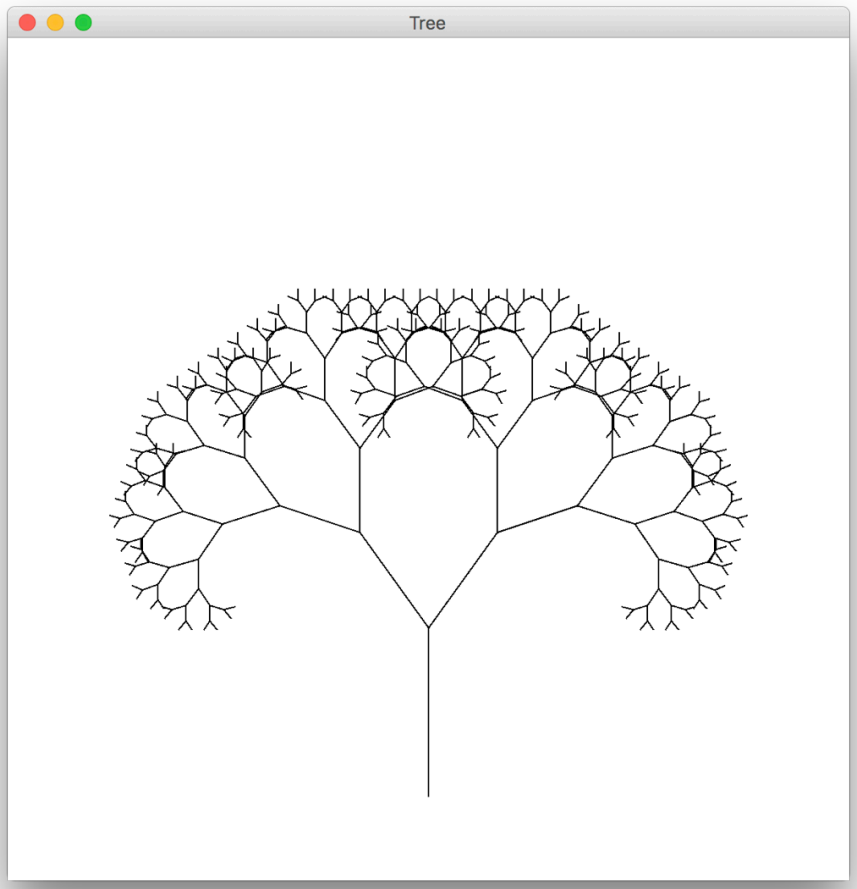
○プログラムリスト

```
import java.awt.*;

public class Tree extends LSystem {
    public static void main(String[] args) {
        if (args.length != 1)
          System.err.println("Usage: java Tree #iterarion");
        else
          (new Tree("Tree", Integer.parseInt(args[0]))).showFrame();
    }

    protected Tree(String name,int times) {
        super(name, times);
    }

    protected void setData() {
        setOrigin(300, 540);
        setRange(5.0);
        initPoints = new Vector2D[2];
        initPoints[0] = new Vector2D(0.0, 0.0);
        initPoints[1] = new Vector2D(0.0, 1.0);

        m = new Matrix2X2[2];
        v = new Vector2D[2];

        m[0] = Matrix2X2.scale(1/Math.sqrt(2)).multiply(Matrix2X2.rotate(-Math.PI / 5));
        m[1] = Matrix2X2.scale(1/Math.sqrt(2)).multiply(Matrix2X2.rotate(Math.PI / 5));
        v[0] = initPoints[1];
        v[1] = initPoints[1];
    }

    protected void drawObject(Graphics g, Vector2D[] points) {
        drawPolyline(g, points);
    }
}
```

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap06 ochihideji$ java Tree 9
```

○実行結果



○考察

FractalではなくLSystemを拡張する。原点の位置やrangeの設定、枝の分岐の角度などに試行錯誤した。

□課題や授業に関して

○レポート作成に要した時間

3時間程度

○特に苦労した点

アフィン変換のベクトルを求めるのが大変だった(特にシェルピンスキーカーペット)。 Webでフラクタルに関する情報を調べたが、専門的な数学に関する話題が多く、理解するのに苦労した。

○授業についての感想や希望

マンデルブロ集合やジュリア集合についてさらに調べてみようと思いました。(Webで調べるとかなり奇妙な図がたくさん出てきて面白いと思いました。)