LABORATORIO #5 GRAFICACIÓN EN COMPONENTES CANVAS

GRUPO #13

INTEGRANTES:

Añez Vladmirovna Leonardo Henry	217002498
Caricari Torrejón Pedro Luis	217009514
Mercado Oudalova Danilo Anatoli	214016668
Mollinedo Franco Milena	216182425
Oliva Rojas Gerson	216036127

MATERIA: **Interacción Hombre-Computador**

FECHA: 04/02/20

PORCENTAJE COMPLETADO: 100%

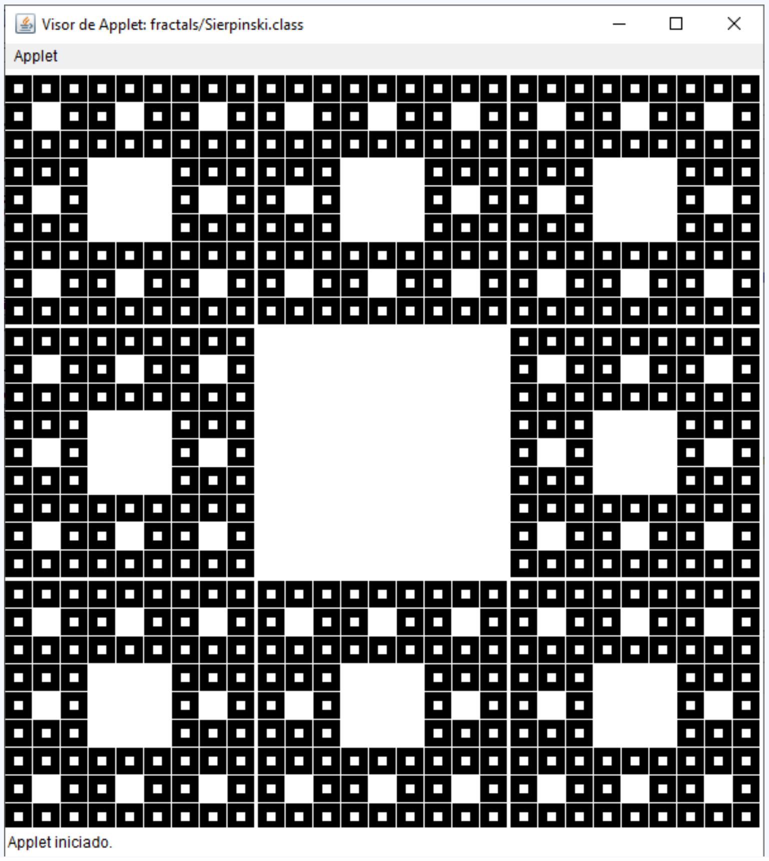
COMENTARIO:

En la práctica de hoy hemos aprendido los fractales y cómo graficarlos en un canvas, actualmente ya sabemos que, para dibujar un fractal, debemos realizar un algoritmo recursivo. También para poder realizar fractales más coloridos y más animados tenemos que aprender a usar la librería de Graphics g, que es un componente fundamental a la hora de dibujar en el lienzo canvas.

CÓDIGO:

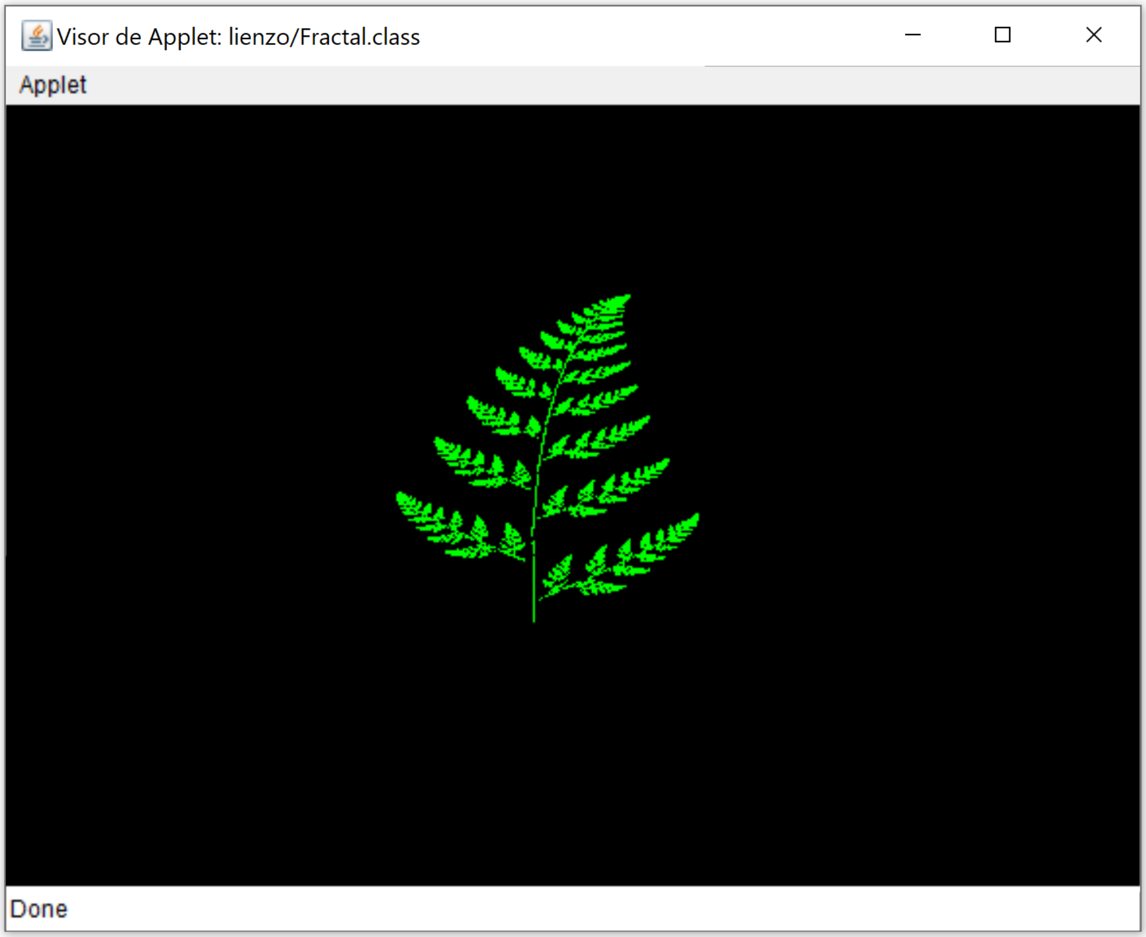
EJERCICIO #1

```java
package fractals;

import java.awt.*;

public class Lienzo extends Canvas {
    private int m, n, nivel;

    public Lienzo(int m, int n, int nivel) {
        this.m = m;
        this.n = n;
        this.nivel = nivel;
        this.setSize(n, m);
    }

    @Override
    public void paint(Graphics g){
        dibujar(0, 0, m, n, nivel, g);
    }

    public void dibujar(int x, int y, int m, int n, int nivel, Graphics
g){
        if (nivel == 0) {
            g.fillRect(x, y, n, m);
            return;
        }
        int ancho = m/3, alto = n/3;
        dibujar(x + 0*ancho, y + 0*alto, ancho, alto, nivel-1, g);
        dibujar(x + 1*ancho, y + 0*alto, ancho, alto, nivel-1, g);
        dibujar(x + 2*ancho, y + 0*alto, ancho, alto, nivel-1, g);
        dibujar(x + 0*ancho, y + 1*alto, ancho, alto, nivel-1, g);
        dibujar(x + 0*ancho, y + 2*alto, ancho, alto, nivel-1, g);
        dibujar(x + 1*ancho, y + 2*alto, ancho, alto, nivel-1, g);
        dibujar(x + 2*ancho, y + 1*alto, ancho, alto, nivel-1, g);
        dibujar(x + 2*ancho, y + 2*alto, ancho, alto, nivel-1, g);
    }
}

package fractals;

import java.applet.Applet;

public class Sierpinski extends Applet {
    private Lienzo lz;
    int m = 600;
    int n = 600;

    @Override
    public void init() {
        this.setSize(m, n);
        this.lz = new Lienzo(m, n, 4);
        this.add(lz);
    }
}
```



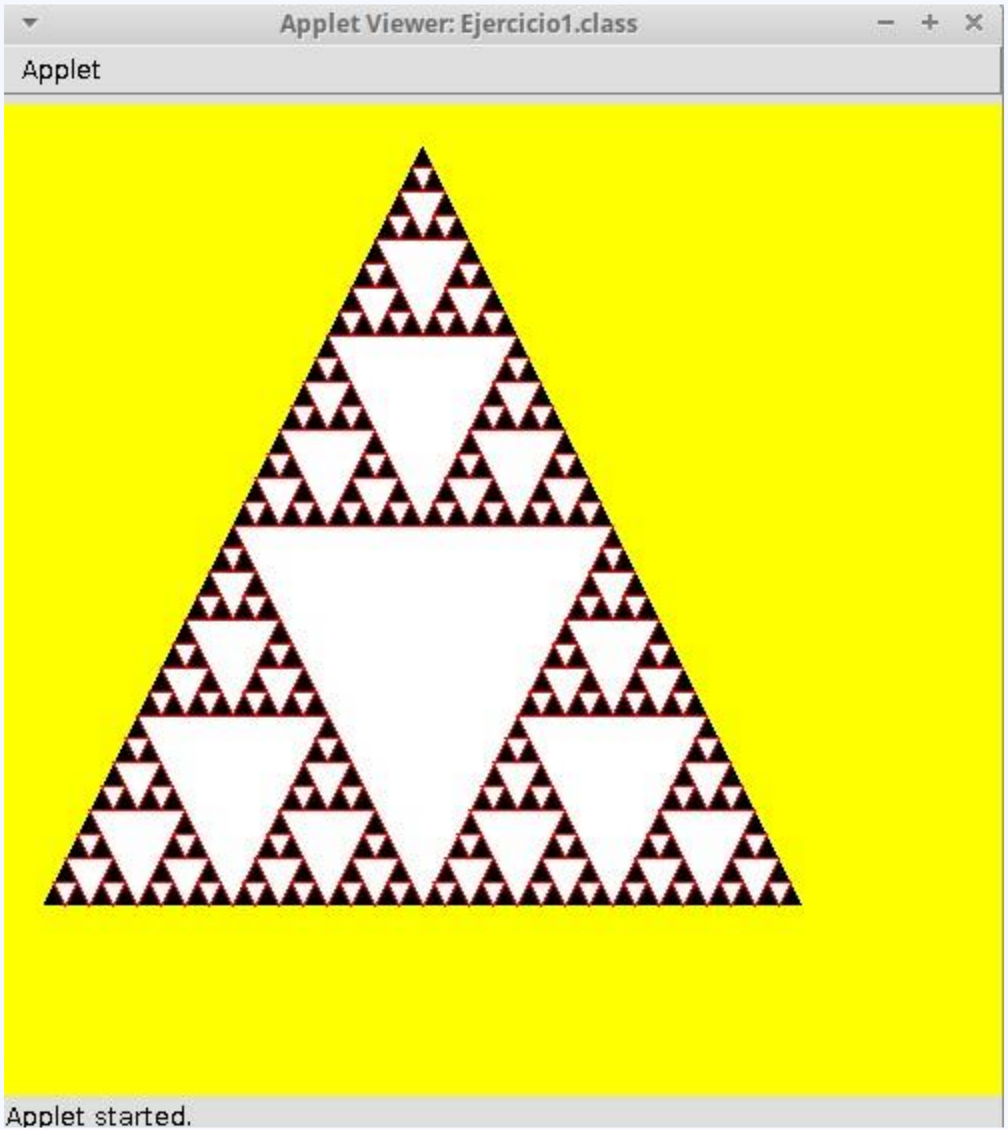
EJERCICIO #2

```java
package lienzo;

import java.awt.Canvas;
import java.awt.Color;
import java.awt.Graphics;

public class Fern extends Canvas {

    double mxx, myy, bxx, byy;

    public Fern() {

    }

    public void paint(Graphics g) {
        dibujar(g);
    }

    public void dibujar(Graphics g) {
        int i;
        double x, y, xn, yn, r;
        int pex, pey;
        int max, seed;
        max = 15000;
        seed = 68111;
        x = 0.5;
        y = 0.0;
        convert(0.0, 1.0, 1.0, -0.5);
        setBackground(Color.black);
        g.setColor(Color.green);
        for (i = 1; i <= max; i++) {
            r = Math.random();
            if (r <= 0.02) {
                xn = 0.5;
                yn = 0.27 * y;
            } else if ((r > 0.02) && (r <= 0.17)) {
                xn = -0.139 * x + 0.263 * y + 0.57;
                yn = 0.246 * x + 0.224 * y - 0.036;
            } else if ((r > 0.17) && (r <= 0.3)) {
                xn = 0.17 * x - 0.215 * y + 0.408;
                yn = 0.222 * x + 0.176 * y + 0.0893;
            } else {
                xn = 0.781 * x + 0.034 * y + 0.1075;
                yn = -0.032 * x + 0.739 * y + 0.27;
            }
            x = xn;
            y = yn;
            pex = (int) (mxx * x + bxx);
            pey = (int) (myy * y + byy);
            g.drawLine(pex, pey, pex, pey);
        }
    }

    void convert(double xiz, double ysu, double xde, double yinf) {
        double maxx, maxy, xxfin, xxcom, yyin, yysu;
        maxx = 600;
        maxy = 450;
        xxcom = 0.15 * maxx;
        xxfin = 0.75 * maxx;
        yyin = 0.8 * maxy;
        yysu = 0.2 * maxy;
        mxx = (xxfin - xxcom) / (xde - xiz);
        bxx = 0.5 * (xxcom + xxfin - mxx * (xiz + xde));
        myy = (yyin - yysu) / (yinf - ysu);
        byy = 0.5 * (yysu + yyin - myy * (yinf + ysu));
    }

}
```



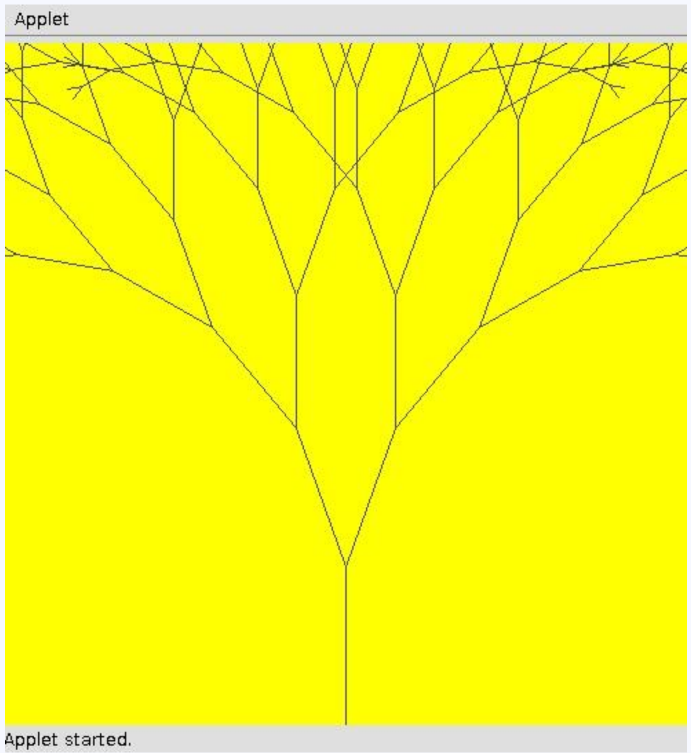
EJERCICIO #3

```
import java.awt.*;

public class Lienzo extends Canvas {

    int n, m;

    Lienzo(int n, int m) {
        this.n = n;
        this.m = m;
        setBackground(Color.yellow);
        setSize(n, m);
    }

    public void paint(Graphics g) {
        int px[] = {20, 400, 210};
        int py[] = {400, 400, 20};

        g.setColor(Color.black);
        g.fillPolygon(px, py, 3);

        paintTriangle(g, new Point(20,400),new Point(400,400),new
Point(210,20), 5);
    }

    public void paintTriangle(Graphics g, Point a, Point b, Point c, int
lvl) {

        Point a1,b1,c1, a2,b2,c2, a3,b3,c3;

        if (lvl==0) return;

        lvl -= 1;

        int px[] = {c.x, (c.x+b.x)/2, (a.x+c.x)/2};
        int py[] = {b.y, (c.y+a.y)/2, (c.y+a.y)/2};

        g.setColor(Color.white);
        g.fillPolygon(px, py, 3);
        g.setColor(Color.red);
        g.drawPolygon(px, py, 3);

        a1 = a;
        b1 = new Point(c.x, b.y);
        c1 = new Point((a.x+c.x)/2, (c.y+a.y)/2);
        paintTriangle(g, a1, b1, c1, lvl);

        a2 = new Point(c.x, b.y);
        b2 = b;
        c2 = new Point((c.x+b.x)/2, (c.y+a.y)/2);
        paintTriangle(g, a2, b2, c2, lvl);

        a3 = new Point((a.x+c.x)/2, (c.y+a.y)/2);
        b3 = new Point((c.x+b.x)/2, (c.y+a.y)/2);
        c3 = c;
        paintTriangle(g, a3, b3, c3, lvl);
    }
}
```


Applet Viewer: Ejercicio1.class
Applet
Applet started.

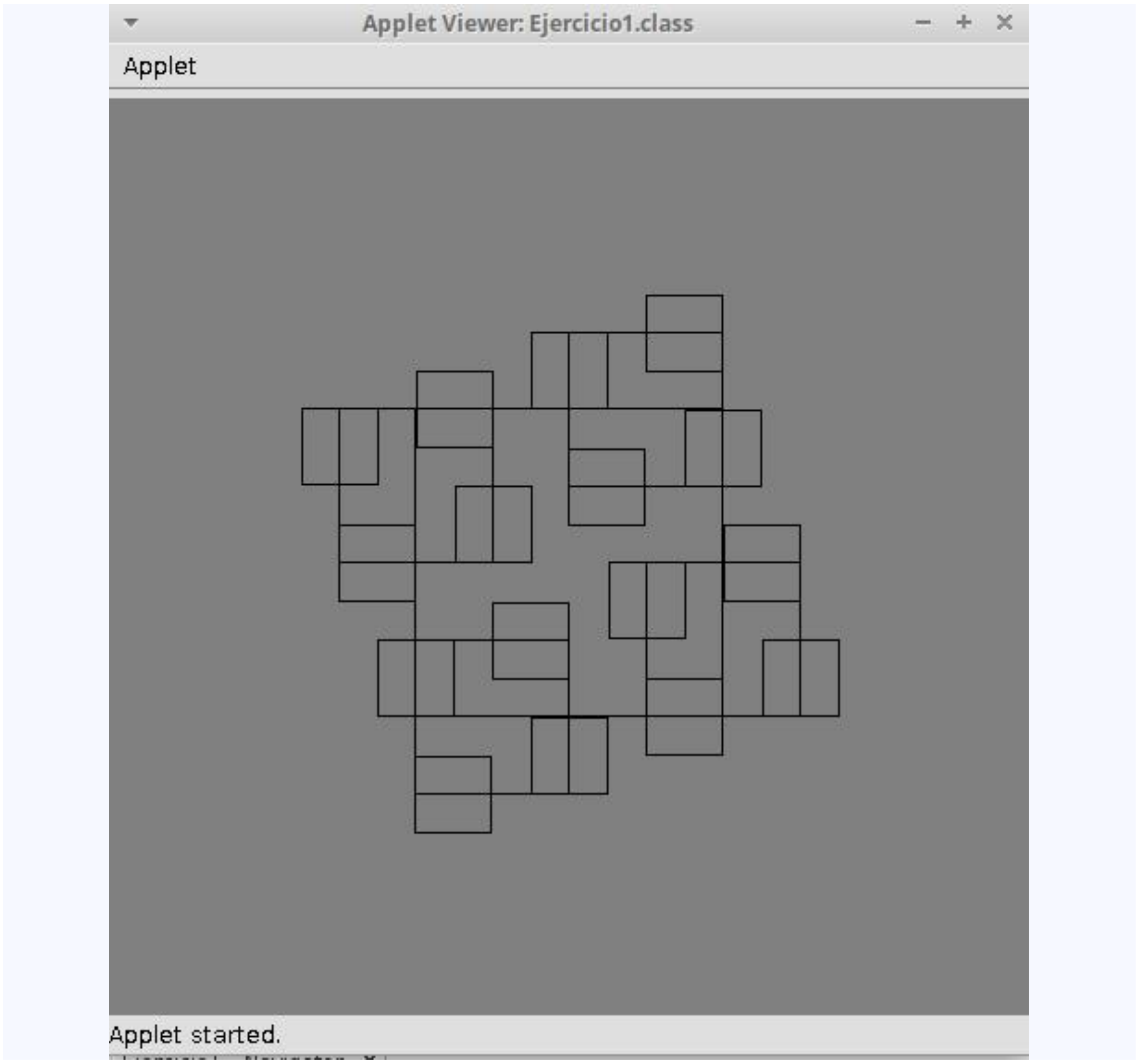
EJERCICIO #4

```
import java.awt.*;

public class Lienzo extends Canvas {

    int n, m;

    Lienzo(int n, int m) {
        this.n = n;
        this.m = m;
        setBackground(Color.yellow);
        setSize(n, m);
    }

    public void paint(Graphics g){
        g.setColor(Color.DARK_GRAY);
        drawTree(g, getWidth() / 2, getHeight(), -90, 10);
    }

    private void drawTree(Graphics g, int x1, int y1, double angle, int
depth) {
        if (depth == 0)
            return;
        int x2 = x1 + (int) (Math.cos(Math.toRadians(angle)) * depth *
12.0);
        int y2 = y1 + (int) (Math.sin(Math.toRadians(angle)) * depth *
12.0);
        g.drawLine(x1, y1, x2, y2);
        drawTree(g, x2, y2, angle - 20, depth - 1);
        drawTree(g, x2, y2, angle + 20, depth - 1);
    }
}
```



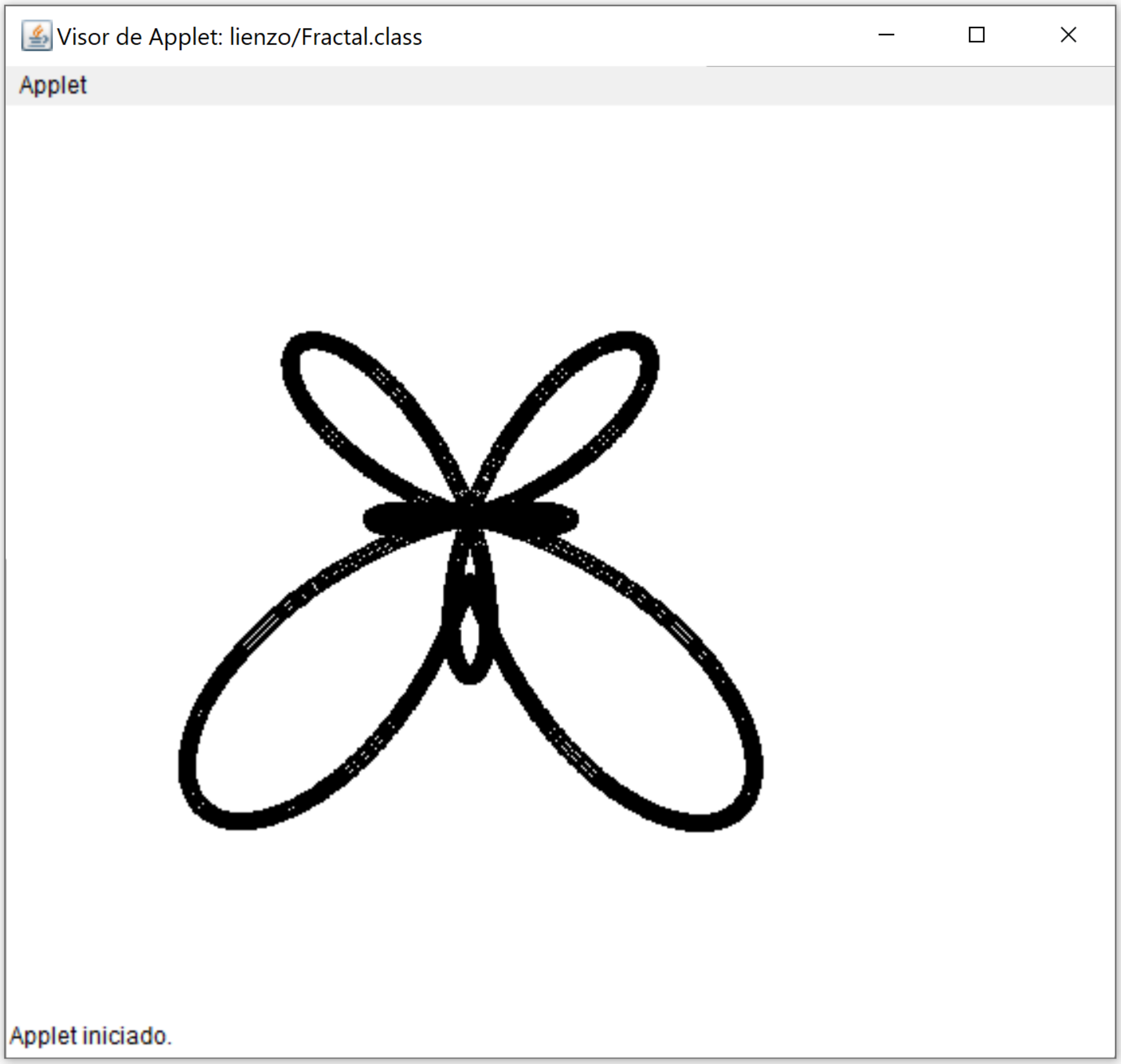
EJERCICIO #5

```
import java.awt.*;
import java.util.ArrayList;
import java.util.Collections;

public class Lienzo extends Canvas {

    int n, m;
    int count = 2;


    Lienzo(int n, int m) {
        this.n = n;
        this.m = m;
        setBackground(Color.GRAY);
        setSize(n, m);

    }
    public void paint(Graphics g) {
                if(++count > 12)
                        count--;
                int width  = n;
                int height = m;
                int size   = width > height ? height / 3 : width / 3;

                fractal(g, new Rectangle(width / 2 - size / 2, height / 2
- size / 2, size, size), count - 1);
        }

        private void fractal(Graphics g, Rectangle r, int count) {
                g.drawRect(r.x, r.y, r.width, r.height);

                if(count < 1)
                        return;

                int newSize = r.width / 2;

                fractal(g, new Rectangle(r.x - newSize / 2, r.y, newSize,
newSize), count - 1);
                fractal(g, new Rectangle(r.x + r.width - newSize, r.y -
newSize / 2, newSize, newSize), count - 1);
                fractal(g, new Rectangle(r.x + r.width - newSize / 2, r.y
+ r.height - newSize, newSize, newSize), count - 1);
                fractal(g, new Rectangle(r.x, r.y + r.height - newSize /
2, newSize, newSize), count - 1);
        }


}
```


Applet Viewer: Ejercicio1.class
Applet
Applet started.

EJERCICIO #6

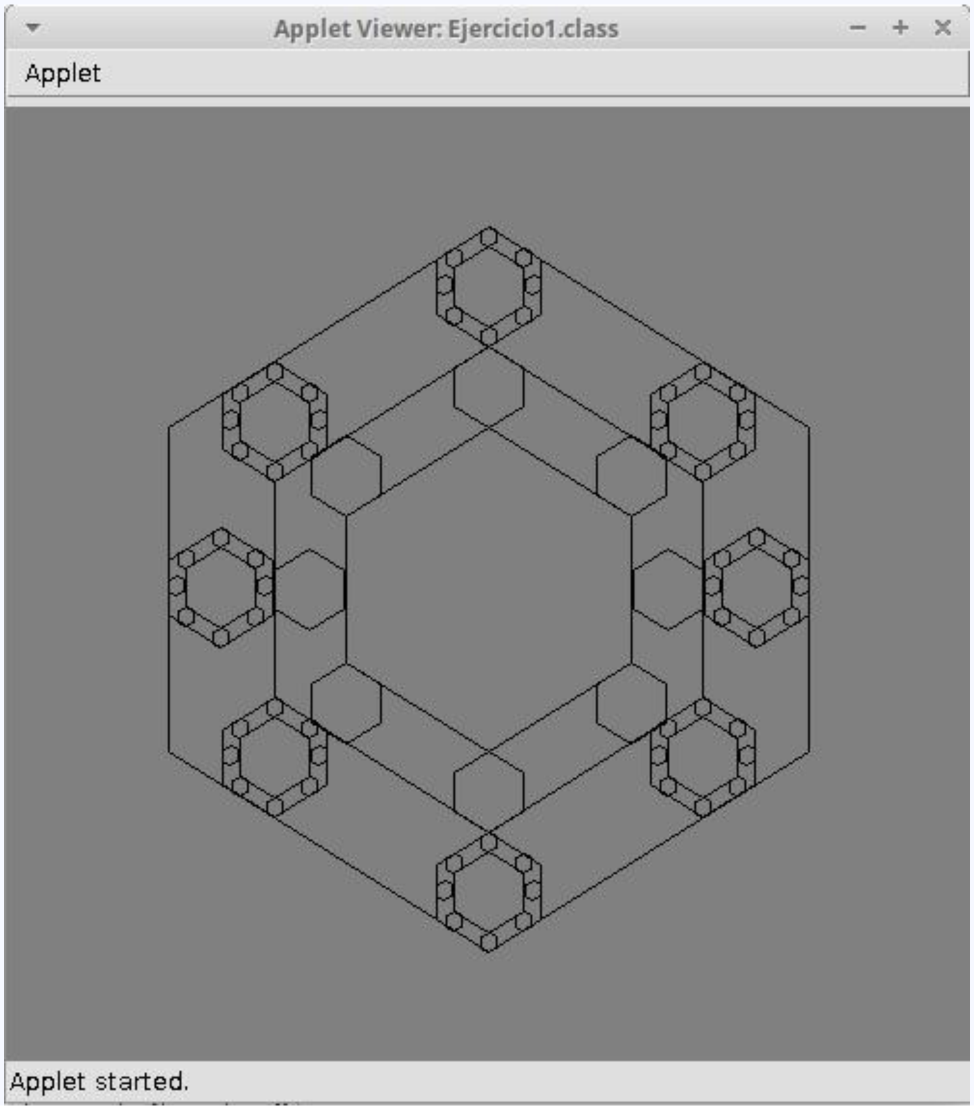
```java
package lienzo;

import java.awt.Canvas;
import java.awt.Graphics;

public class ButterflyCurve extends Canvas{

    public ButterflyCurve(int n,int m){
        this.setSize(n,m);
    }



    public void paint(Graphics g) {
        dibujar(g);
    }

    public void dibujar(Graphics g) {
        g.translate(200, 200);
        double max = Math.PI*2;
        double iter = 2000;
        double d = max/iter;
        double t = 0;

        for(int i=0;i<iter;++i){
            double x = (50)*Math.sin(t) * (Math.pow(Math.E, Math.cos(t))
- 2 *Math.cos(4*t)-Math.pow(t/12, 5)) ;
            double y = (50)*Math.cos(t) * (Math.pow(Math.E, Math.cos(t))
- 2 *Math.cos(4*t)-Math.pow(t/12, 5)) ;
            t+=d;
            g.drawOval((int)x,(int)y, 8, 8);
        }
    }

}
```


Visor de Applet: lienzo/Fractal.class
Applet
Applet iniciado.

EJERCICIO #7

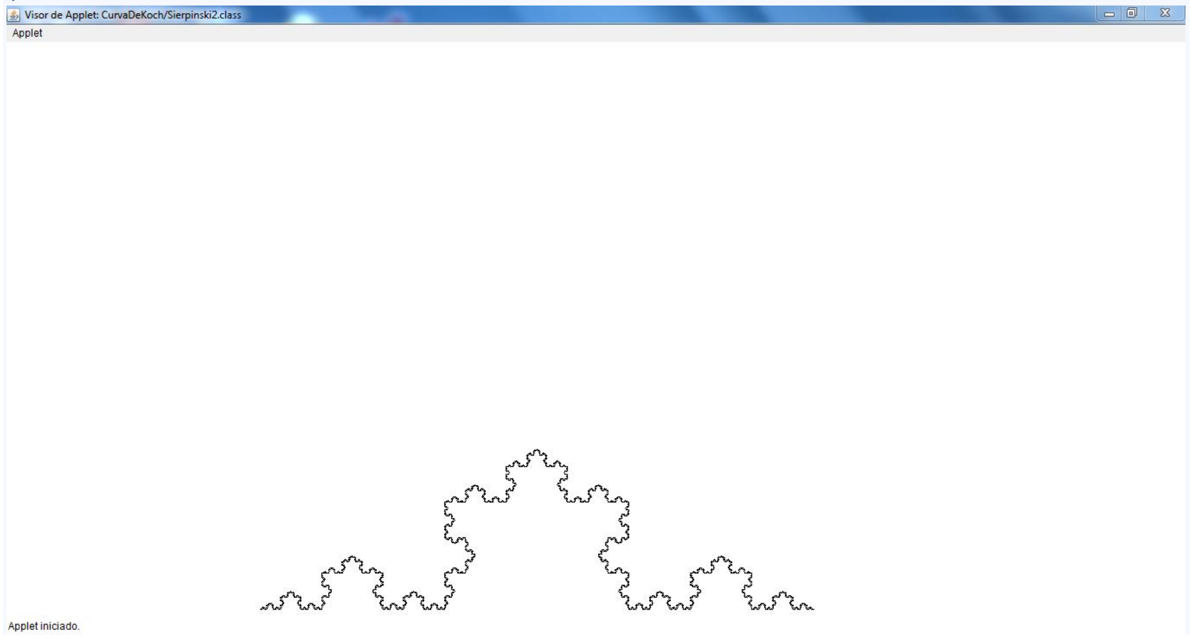
```
import java.awt.*;
import java.util.ArrayList;
import java.util.Collections;

public class Lienzo extends Canvas {

    int n, m;
    int count = 2;

    Lienzo(int n, int m) {
        this.n = n;
        this.m = m;
        setBackground(Color.GRAY);
        setSize(n, m);

    }

    public void paint(Graphics g) {
                if(++count > 10)
                        count--;
                int width  = n;
                int height = m;

                fractal(g, new Point(width / 2, height / 8),
                                new Point(width / 2 - height / 3, height /
3),
                                new Point(width / 2 - height / 3, height -
height / 3),
                                new Point(width / 2, height - height / 8),
                                new Point(width / 2 + height / 3, height -
height / 3),
                                new Point(width / 2 + height / 3, height /
3),
                                count - 1);
        }

        private void fractal(Graphics g, Point p1, Point p2, Point p3,
Point p4, Point p5, Point p6, int count) {
                g.drawPolygon(new int[] {p1.x, p2.x, p3.x, p4.x, p5.x,
p6.x}, new int[] {p1.y, p2.y, p3.y, p4.y, p5.y, p6.y}, 6);

                if(count < 1)
                        return;

                int w  = p6.x - p2.x;
                int h  = p4.y - p1.y;
                int h1 = p3.y - p2.y;
                int w2 = w / 6;
                int h2 = h / 6;
                int h3 = p2.y - p1.y;

                Point np1 = new Point(p1.x, p1.y + h2);
                Point np2 = new Point(p2.x + w2, p2.y + h1 / 6);
                Point np3 = new Point(p3.x + w2, p3.y - h1 / 6);
                Point np4 = new Point(p4.x, p4.y - h2);
                Point np5 = new Point(p5.x - w2, p5.y - h1 / 6);
        Point np6 = new Point(p6.x - w2, p6.y + h1 / 6);

        fractal(g, np1, np2, np3, np4, np5, np6, count - 1);

        np1 = new Point(p1.x, p1.y);
        np2 = new Point(p1.x - w2 / 2, p1.y + h3 / 6);
        np3 = new Point(p1.x - w2 / 2, p1.y + h2 - h3 / 6);
        np4 = new Point(p1.x, p1.y + h2);
        np5 = new Point(p1.x + w2 / 2, p1.y + h2 - h3 / 6);
        np6 = new Point(p1.x + w2 / 2, p1.y + h3 / 6);

        fractal(g, np1, np2, np3, np4, np5, np6, count - 1);

        np1 = new Point(p4.x, p4.y - h2);
        np2 = new Point(np1.x - w2 / 2, np1.y + h3 / 6);
        np3 = new Point(np1.x - w2 / 2, np1.y + h2 - h3 / 6);
        np4 = new Point(np1.x, np1.y + h2);
        np5 = new Point(np1.x + w2 / 2, np1.y + h2 - h3 / 6);
        np6 = new Point(np1.x + w2 / 2, np1.y + h3 / 6);

        fractal(g, np1, np2, np3, np4, np5, np6, count - 1);

        np1 = new Point(p2.x + w2 / 2, p2.y + h3 / 2);
        np2 = new Point(np1.x - w2 / 2, np1.y + h3 / 6);
        np3 = new Point(np1.x - w2 / 2, np1.y + h2 - h3 / 6);
        np4 = new Point(np1.x, np1.y + h2);
        np5 = new Point(np1.x + w2 / 2, np1.y + h2 - h3 / 6);
        np6 = new Point(np1.x + w2 / 2, np1.y + h3 / 6);

        fractal(g, np1, np2, np3, np4, np5, np6, count - 1);

        np1 = new Point(p6.x - w2 / 2, p6.y + h3 / 2);
        np2 = new Point(np1.x - w2 / 2, np1.y + h3 / 6);
        np3 = new Point(np1.x - w2 / 2, np1.y + h2 - h3 / 6);
        np4 = new Point(np1.x, np1.y + h2);
        np5 = new Point(np1.x + w2 / 2, np1.y + h2 - h3 / 6);
        np6 = new Point(np1.x + w2 / 2, np1.y + h3 / 6);

        fractal(g, np1, np2, np3, np4, np5, np6, count - 1);

        np1 = new Point(p3.x + w2, p3.y - h1 / 6);
        np2 = new Point(np1.x - w2 / 2, np1.y + h3 / 6);
        np3 = new Point(np1.x - w2 / 2, np1.y + h2 - h3 / 6);
        np4 = new Point(np1.x, np1.y + h2);
        np5 = new Point(np1.x + w2 / 2, np1.y + h2 - h3 / 6);
        np6 = new Point(np1.x + w2 / 2, np1.y + h3 / 6);

        fractal(g, np1, np2, np3, np4, np5, np6, count - 1);

        np1 = new Point(p5.x - w2, p5.y - h1 / 6);
        np2 = new Point(np1.x - w2 / 2, np1.y + h3 / 6);
        np3 = new Point(np1.x - w2 / 2, np1.y + h2 - h3 / 6);
        np4 = new Point(np1.x, np1.y + h2);
        np5 = new Point(np1.x + w2 / 2, np1.y + h2 - h3 / 6);
        np6 = new Point(np1.x + w2 / 2, np1.y + h3 / 6);

        fractal(g, np1, np2, np3, np4, np5, np6, count - 1);
```

```
                np1 = new Point(p2.x + w2, p2.y + h1 / 6 - h2);
                np2 = new Point(np1.x - w2 / 2, np1.y + h3 / 6);
                np3 = new Point(np1.x - w2 / 2, np1.y + h2 - h3 / 6);
                np4 = new Point(np1.x, np1.y + h2);
                np5 = new Point(np1.x + w2 / 2, np1.y + h2 - h3 / 6);
                np6 = new Point(np1.x + w2 / 2, np1.y + h3 / 6);

                fractal(g, np1, np2, np3, np4, np5, np6, count - 1);

                np1 = new Point(p6.x - w2, p6.y + h1 / 6 - h2);
                np2 = new Point(np1.x - w2 / 2, np1.y + h3 / 6);
                np3 = new Point(np1.x - w2 / 2, np1.y + h2 - h3 / 6);
                np4 = new Point(np1.x, np1.y + h2);
                np5 = new Point(np1.x + w2 / 2, np1.y + h2 - h3 / 6);
                np6 = new Point(np1.x + w2 / 2, np1.y + h3 / 6);

                fractal(g, np1, np2, np3, np4, np5, np6, count - 1);
        }
}
```



EJERCICIO #8

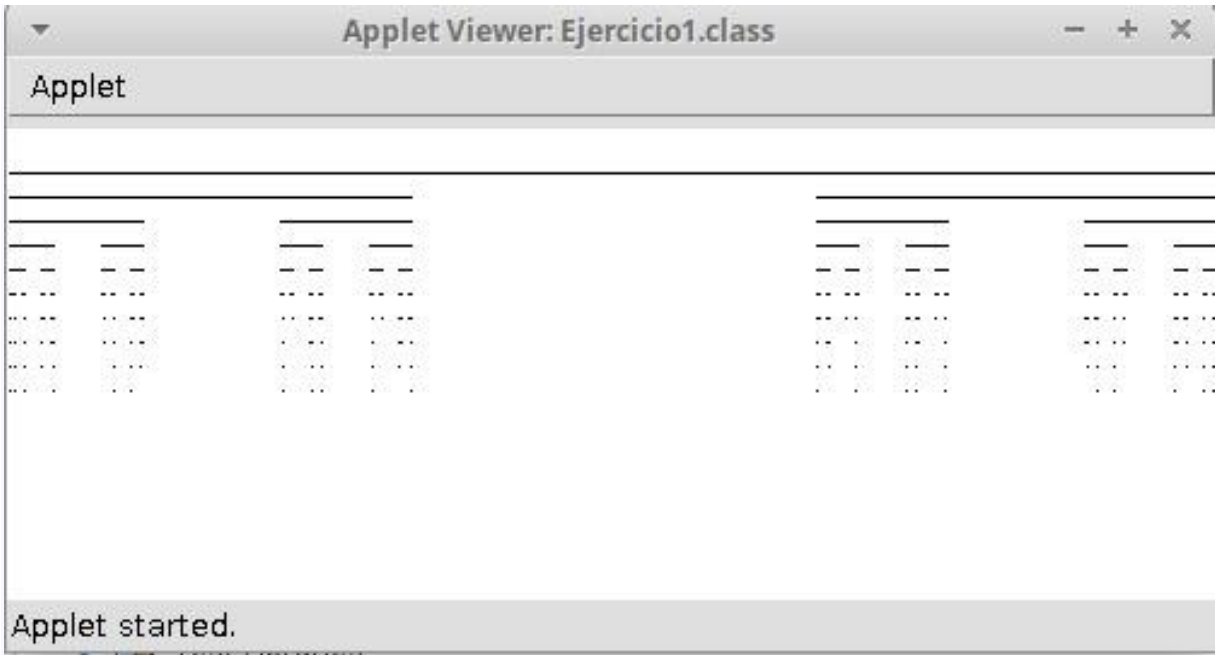
```java
public class Lienzo2 extends Canvas {
    double xp1 = 650;
    double yp1 = 650;
    double xp2 = 10;
    double yp2 = 650;
    double sin60 = Math.sin(3.14 / 3.);
    int nivel_de_recursividad = 6;
    public Lienzo2(int n, int m) {
        this.setSize(n, m);
    }
    public void paint(Graphics g) {

               paintRecursivo(g,nivel_de_recursividad,xp1,yp1,xp2,yp2);
    }
    private void paintRecursivo(Graphics g, int i, double xp12, double 
yp12, double xp22, double yp22) {
        double dx = (xp22 - xp12) / 3.;
        double dy = (yp22 - yp12) / 3.;
        double xx = xp12 + 3 * dx / 2. - dy * sin60;
        double yy = yp12 + 3 * dy / 2. + dx * sin60;
        if (i <= 0) {
            g.drawLine((int) xp12, (int) yp12, (int) xp22, (int) yp22);
        } else {
            paintRecursivo(g, i - 1, xp12, yp12, xp12 + dx, yp12 + dy);
            paintRecursivo(g, i - 1, xp12 + dx, yp12 + dy, xx, yy);
            paintRecursivo(g, i - 1, xx, yy, xp22 - dx, yp22 - dy);
            paintRecursivo(g, i - 1, xp22 - dx, yp22 - dy, xp22, yp22);
        }
    }
}
```



Applet iniciado.

EJERCICIO #9

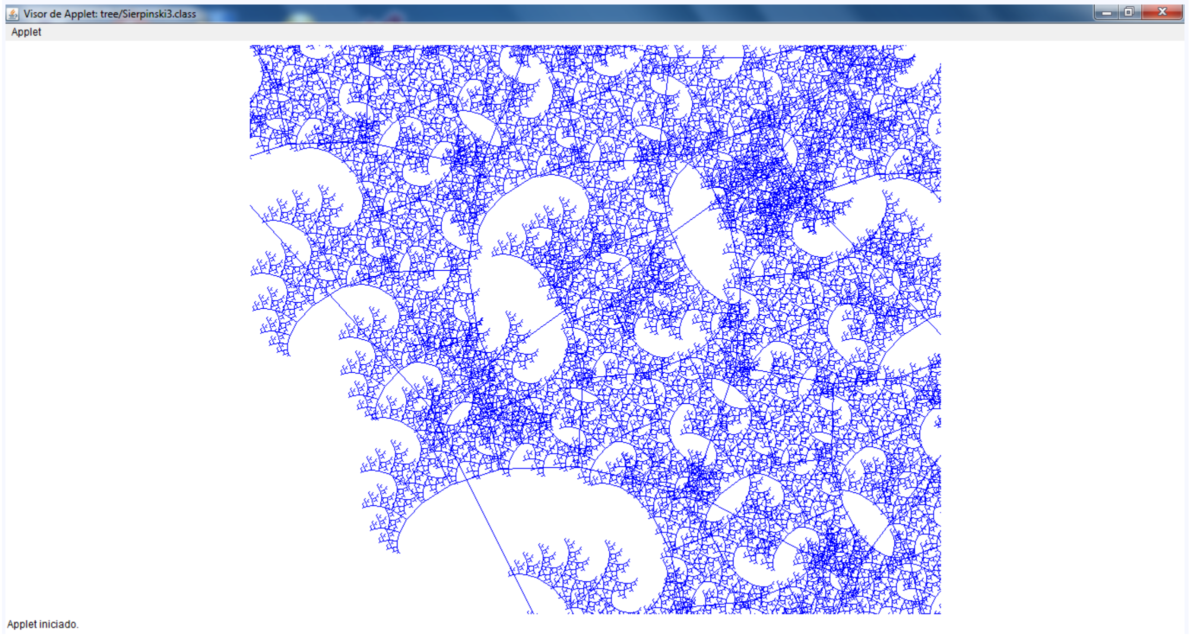
```
import java.awt.*;
import java.util.ArrayList;
import java.util.Collections;

public class Lienzo extends Canvas {

    int n, m;
    int maxIter = 10;

    Lienzo(int n, int m) {
        this.n = n;
        this.m = m;
        setBackground(Color.WHITE);
        setSize(n, m);

    }

   public void paint(Graphics g) {

        int y = 17;

        for (int m = 1; m < maxIter + 1; m++) {

            for (int x = 0; x < n; x++) {

                if (belongsToSet(m,x) == true) g.drawOval(x,y,1,1);
            }
            y = y + 10;
        }
    }

    public boolean belongsToSet(int m, double x) {

        double line = n;
        double x1,x2;

        for(int mm = 1; mm < m ; mm++) {

            for (int k = 0; k < (int)((Math.pow(3,(mm-1))) - 1 + 1); k++)
{

                x1 = (3.0*k + 1) / Math.pow(3.0,mm);
                x2 = (3.0*k + 2) / Math.pow(3.0,mm);

                if ((x > x1 * line) && (x < x2 * line)) return false;

            }

        }
        return true;
    }
}
```



```
Applet started.
```

EJERCICIO #10

```
public class Lienzo3 extends Canvas {

    public Lienzo3(int n, int m) {
        this.setSize(n, m);
    }

    public void paint(Graphics g) {
        drawTree(500, 1000, 650, 90, g);
    }

    public void drawTree(double x0, double y0, double len, double angle,
Graphics g) {
        if (len > 2) {
            int x1 = (int) (x0 + len * Math.cos(angle));
            int y1 = (int) (y0 - len * Math.sin(angle));
            g.setColor(Color.blue);
            g.drawLine((int) x0, (int) y0, x1, y1);
            drawTree(x1, y1, len * 0.75, angle + 30, g);
            drawTree(x1, y1, len * 0.66, angle - 50, g);
        }
    }
}
```


Visor de Applet: tree/Sierpinski3.class
Applet
Applet iniciado.