```
// www.jjam.de - Lindenmayer System 2 - Version 21.03.2003
//Ergänzung Haftendorn: Anzeige der Stufen, ausgelöst durch Javascript 2.5.05
import java.awt.*;
import java.applet.*;

public class Lindenmayer2 extends Applet {

    Point a, b;                // Verbindungspunkte eines Einzelschritts


    int stufe=7;               //  Stufe  max 7  //Ha
    int j;                     //Zähler//Ha
    int k=2;                  //Streckfaktor klein auf groß pro Stufe//Ha
    int lengthF = 4;           // Schrittlänge
    double direction;          // Richtung in Grad
    double rotation = 37;      // Drehung in Grad

    Graphics g;
    Graphics2D g2;

    public void init() {
        setBackground(new Color(255,255,200));//Ha vari
    }

    public void paint(Graphics g) {
        g2 = (Graphics2D) g; // Anti-Aliasing
        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

        a = new Point(90,430); // Start-Punkt
        direction = -80;         // Start-Richtung in Grad
        lengthF=k*k*k*k*k*k*k*k;          //Ha
        for (j=1;j<stufe+1;j++){       // Schrittlänge    Ha
            lengthF= lengthF/k;
        }



        turtleGraphic(g2, "T", stufe);  // Axiom und Iterationstiefe

    }

    public void setStufe(int stufeNr)                          //Ha eingefügt
    /** soll im JavaSrcipt verwendet werden **/
        {
            stufe=stufeNr;
            repaint(50L);  //L fuer Long, 50 Millisek
        }

    public void turtleGraphic(Graphics g2, String instruction, int depth) {

        if (depth==0) return;
        depth -= 1;

        Point aMark = new Point(0,0);
        double directionMark = 0;
        // Dummy-Werte

        int i;
        char c;

        for (i=0;i<instruction.length();i++) {

            c = instruction.charAt(i);

            // Produktionsregel iterieren, solange Tiefe nicht erreicht ist
```

```
            if (c=='T') turtleGraphic(g2, "R-[T]++[++L]R[--L]+[T]--T", depth);
            else if (c=='R') turtleGraphic(g2, "F[++L][--L]F", depth);
            else if (c=='L') turtleGraphic(g2, "[+FX-FX-FX+|+FX-FX-FX|+FXFX]", depth);

            // Schritt Vorwärts
            else if (c=='F') {

                if (i+1<instruction.length() && instruction.charAt(i+1)=='X')
turtleGraphic(g2, "FX", depth);
                else turtleGraphic(g2, "FF", depth);

                // Zeichnen: Ab 'a' in Richtung 'direction' einen Schritt der Länge
'lengthF'
                if (depth==0) {
                    double rad = 2*Math.PI/360 * direction; // Grad -> Radiant

                    int p = (int) (lengthF * Math.cos(rad));
                    int q = (int) (lengthF * Math.sin(rad));

                    b = new Point(a.x+p, a.y+q);

                    // Farben Blätter und Stamm
                    if (i+1<instruction.length() && instruction.charAt(i+1)=='X')
g2.setColor(new Color(154,205,50));
                    else g2.setColor(new Color(120,150,100));

                    g2.drawLine(a.x, a.y, b.x ,b.y);

                    a = b; // Neuer Startpunkt
                }
            }

            // Drehung links herum
            else if (c=='+') direction += rotation;

            // Drehung rechts herum
            else if (c=='-') direction -= rotation;

            // Drehung um 180 Grad
            else if (c=='|') direction += 180;

            // Position und Richtung speichern
            else if (c=='[') {
                aMark = a;
                directionMark = direction;
            }

            // Zurück zu gespeicherter Position und Richtung
            else if (c==']') {
                a = aMark;
                direction = directionMark;
            }
        }
    }
}
```