```java
// Spektrum der Wissenschaft Juni 2003
// Kolam-Zeichnungen - durch formale Sprachen reproduziert
// 2003.05.31; Rolf Krüger
// Überarbeutet am 23.10.2007

import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;


public class Schlangen extends Applet implements ActionListener  {
    private Button butZeichne;          // Starte Neuzeichnen
    private int iWidth;                 // Die Weite des Appletbereichs
    private int iHeight;                // Die Höhe des Appletbereichs
    private int iRek;                   // Die gewählte Rekursionsstufe

    private double dStartX;             // Linien-Startwert X
    private double dStartY;             // Linien-Startwert Y
    private double dEndX;               // Linien-Endwert X
    private double dEndY;               // Linien-Endwert Y
    private double dLinie;              // Linienlänge
    private double dWinkel = 0.0;       // Aktueller Zeichnungswinkel

    // Die Strings für die Regeln
    // Die Startregel
    StringBuffer initRegel = new StringBuffer();
    String zeichenRegel;                // Die Zeichenregel
    String ersatzRegel;                 // Die Ersetzungsregel
    // Der String für den Titeltext
    String nString;

    // Die darzustellende Regel
    StringBuffer dRegel = new StringBuffer();


  public void init() {
    // Die Parameter des Appletbereichs werden eingelesen.
    // Sind die Parameter nicht gesetzt, werden Standardwerte gesetzt

    // Lese die Weite und Höhe des Applets ein
    if((nString = getParameter("WIDTH")) != null)
      iWidth = Integer.parseInt(nString);
    else
        iWidth = 320;
    if((nString = getParameter("HEIGHT")) != null)
      iHeight = Integer.parseInt(nString);
    else
        iHeight = 330;

    // Lese die Rekursionstiefe ein
    if((nString = getParameter("Rekursion")) != null)
      iRek = Integer.parseInt(nString);
    else iRek = 3;

    // Lese die Ausgangsregel ein
    if((nString = getParameter("InitRegel")) != null)
      initRegel.append(nString);
    else
      initRegel.append("B--F--B--F");

    // Lese die Zeichenregel ein
    if((nString = getParameter("ZeichenRegel")) != null)
      zeichenRegel = nString;
    else
      zeichenRegel = "F+F+F--F--F+F+F";

    // Lese die Ersetzungsregel ein
    if((nString = getParameter("ErsatzRegel")) != null)
      ersatzRegel = nString;
    else
      ersatzRegel = "B+F+B--F--B+F+B";

    // Lese die Appletüberschrift ein
    if((nString = getParameter("AppletText")) != null)
      ;
    else
      nString = "kein Titeltext angegeben";

    // Den Zeichnungsbutton darstellen
    butZeichne = new Button("Zeichne");
    add(butZeichne);
    butZeichne.addActionListener(this);
  }

    private StringBuffer neueRegel(StringBuffer aStr, String eStr, int iRek) {
        // Durchsuche den Ausgangsstring nach "B" und ersetze "B" durch eStr
```

```
87          // Das Ergebnis wird in neue Regel abgelegt
88          // Wenn die Rekursionstiefe auf 0 gesetzt wird, erfolgt trotzdem
89          // eine Ersetzung
90          StringBuffer hStr = new StringBuffer();
91          for(int i=0; i<aStr.length(); i++)
92          {
93                  int iZeichen = (int) aStr.charAt(i);
94                  if(iZeichen == (int) &apos;B&apos;)
95                          hStr.append(eStr);
96                  else hStr.append((char) iZeichen);
97          }
98          if(iRek >0)
99                  hStr = neueRegel(hStr, eStr, iRek -1);
100         return hStr;
101     }
102
103
104     public void actionPerformed(ActionEvent event) {
105         // Eine Mehrfach-Ersetzung
106         dRegel = neueRegel(initRegel, ersatzRegel, iRek-1);
107         // Vollständige Ersetzung durch die Zeichenregeln
108         dRegel = neueRegel(dRegel, zeichenRegel, 0);
109         repaint();
110     }
111
112     public void paint(Graphics g) {
113         g.setColor(Color.black);
114         setBackground(new Color(255, 247, 247));
115         dStartX = 30;
116         dStartY = iHeight/2.0;
117         dEndX = dStartX;
118         dEndY = dStartY;
119         dLinie = 5.0;
120         dWinkel=0;
121
122         // Interpretiere den darzustellenden String
123         for(int i=0; i<dRegel.length(); i++)
124         {
125             int iZeichen = (int) dRegel.charAt(i);
126
127             switch(iZeichen){
128                 case (int) &apos;F&apos;:
129                     // Langsam Zeichnen
130                     try {
131                         Thread.sleep(1);
132                     }
133                     catch(InterruptedException e) {
134                         // tue aber nichts um einen Programmabbruch zu vermeiden
135                     }
136                     // Zeichne eine Linie der Länge iLinie in die Richtung
137                     // iWinkel
138                     dEndX = dStartX + dLinie* Math.cos(dWinkel);
139                     dEndY = dStartY + dLinie* Math.sin(dWinkel);
140                     g.drawLine((int) dStartX, (int) dStartY,
141                         (int) (dEndX), (int) (dEndY));
142                     dStartX = dEndX; dStartY = dEndY;
143                     break;
144
145                 case (int) &apos;-&apos;:
146                     dWinkel += Math.PI/4.0;
147                     break;
148
149                 case (int) &apos;+&apos;:
150                     dWinkel -= Math.PI/4.0;
151                     break;
152             }
153         }
154     }
155 }
```