```
//
//  Biegeschwingungen: Kragarm mit Zusatzkoerper
//
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class Kragarm extends JFrame
                          implements ActionListener {
        private static final long serialVersionUID = 2018;
        JTextField  eingabeStartWert, ausgabeEigenWert,
        eingabeP1, eingabeP2;
        double l1 = 0., l2 = 0., ew = 0., ratio;
        int bc;
        Color fbgColor, bgColor;
        private Curve diagram;
        JButton eigenWert;

        // Konstruktor
        public Kragarm() {
            Container fenster = getContentPane();
            fbgColor = new Color(178,214,252);
            fenster.setBackground(fbgColor);
            BorderLayout h1 = new BorderLayout();
            fenster.setLayout(h1);
            JLabel p1Wert = new JLabel("m/"+"\u03C1"+"AL : ",
                                       JLabel.RIGHT);
            eingabeP1 = new JTextField(12);

eingabeP1.setBorder(BorderFactory.createLoweredBevelBorder());
            JLabel p2Wert = new
JLabel("J/"+"\u03C1"+"AL"+"\u00B3"+" : ",
                                       JLabel.RIGHT);
            eingabeP2 = new JTextField(12);

eingabeP2.setBorder(BorderFactory.createLoweredBevelBorder());
            JLabel startWert = new JLabel("Sch"+"\u00E4"+"tzwert
[1/L] : ", JLabel.RIGHT);
            eingabeStartWert = new JTextField(12);
eingabeStartWert.setBorder(BorderFactory.createLoweredBevelBorder(
));
            bgColor = new Color(140,181,222);
            eigenWert = new JButton("Eigenwert");
            eigenWert.setBackground(bgColor);
            ausgabeEigenWert = new JTextField(12);
            ausgabeEigenWert.setBackground(fbgColor);

ausgabeEigenWert.setBorder(BorderFactory.createLoweredBevelBorder(
));
            JPanel h2d = new JPanel();

h2d.setBorder(BorderFactory.createLineBorder(Color.darkGray));
            h2d.setBackground(bgColor);
```

```
            h2d.setLayout(new GridLayout(4,2,5,5));
            h2d.add(p1Wert);
            h2d.add(eingabeP1);
            h2d.add(p2Wert);
            h2d.add(eingabeP2);
            h2d.add(startWert);
            h2d.add(eingabeStartWert);
            h2d.add(eigenWert);
            eigenWert.addActionListener(this);
            h2d.add(ausgabeEigenWert);
            JLabel titel = new JLabel
                ("Biegeschwingungen: Kragarm mit
Zusatzk"+"\u00F6"+"rper",
                 JLabel.CENTER);
            fenster.add(titel, BorderLayout.NORTH);
            fenster.add(h2d, BorderLayout.SOUTH );
            // Grafikkomponente
            diagram = new Curve();

diagram.setBorder(BorderFactory.createLoweredBevelBorder());
            fenster.add(diagram, BorderLayout.CENTER);
            // Erscheinungsbild: Nimbus
            try {
UIManager.setLookAndFeel("javax.swing.plaf.nimbus.NimbusLookAndFee
l");
               }
               catch (InstantiationException e) {
               }
               catch (ClassNotFoundException e) {
               }
               catch (UnsupportedLookAndFeelException e) {
               }
               catch (IllegalAccessException e) {
               }
               SwingUtilities.updateComponentTreeUI(fenster);
            fenster.setVisible(true);

        }

     // Initialisierung
     public static void main(String[] args) {
          int xPos,yPos;
          JFrame frame = new Kragarm();
          ExitWindow abbrechen = new ExitWindow();
          frame.addWindowListener(abbrechen);
          // Abfrage Bildschirmabmessungen
          Dimension dim =
Toolkit.getDefaultToolkit().getScreenSize();
          // Abmessungen des Applikationsfensters
          frame.setSize(640,320);
          // Positionierung des Applikationsfensters auf dem
Bildschirm
```

```
      xPos = (dim.width-640)/2;
      yPos = (dim.height-320)/2;
      frame.setLocation(xPos,yPos);
      // Anzeige des Rahmenfensters auf dem Desktop
      frame.setVisible(true);
}

    public void actionPerformed(ActionEvent event) {
    double x, g1, g2, g3, g4, g5;
    int iew;
    // Eigenwert
      if (event.getSource() == eigenWert) {
            l1 = Double.parseDouble(eingabeP1.getText());
            if(l1 < 0.) {
                l1 = Math.abs(l1);
                eingabeP1.setText("" + l1);
            }
            l2 = Double.parseDouble(eingabeP2.getText());
            if(l2 < 0.) {
                l2 = Math.abs(l2);
                eingabeP2.setText("" + l2);
            }
            if(l1 == 0.) {
              l2 = 0.;
              eingabeP2.setText("" + l2);
            }
        x = Double.parseDouble(eingabeStartWert.getText());
        // Newtonsches Naeherungsverfahren
        double inkrement = cf(x)/cfs(x);
        while (Math.abs(inkrement/x) > 1.e-7) {
          x = x - inkrement;
          inkrement = cf(x)/cfs(x);
        }
        ew = x - inkrement;
        iew = (int) Math.round(ew*10000f);
        ew = iew/10000f;
        ausgabeEigenWert.setText((float) ew + " / " + "L");
        g1 = Math.exp(ew);
        g2 = Math.sin(ew);
        g3 = Math.cos(ew);
        g4 = 0.5*(g1-1./g1);
        g5 = 0.5*(g1+1./g1);
        ratio = (g2-g4+l1*ew*(g3-g5))/(g3+g5+l1*ew*(g4-g2));
        repaint();
      }
    }

        double cf(double x) {
          // Charakteristische Eigenwertgleichung
          double e1, e2, e3, sh, ch, f1, f2, f3, px;
          e1 = Math.exp(x);
          e2 = Math.sin(x);
```

```
              e3 = Math.cos(x);
              // Hyperbelfunktionen
              sh = 0.5*(e1-1./e1);
              ch = 0.5*(e1+1./e1);
              f1 = e3*sh;
              f2 = e2*ch;
              f3 = e3*ch;
              px = Math.pow(x,3.);
              return 1.+f3+l1*x*(f1-
f2)+px*l2*((f1+f2)+l1*x*(f3-1.));
            }

            double cfs(double x) {
              // Ableitung der Eigenwertgleichung
              double e1, e2, e3, sh, ch, f1, f2, f3, f4, px;
              e1 = Math.exp(x);
              e2 = Math.sin(x);
              e3 = Math.cos(x);
              // Hyperbelfunktionen
              sh = 0.5*(e1-1./e1);
              ch = 0.5*(e1+1./e1);
              f1 = e3*sh;
              f2 = e2*ch;
              f3 = e3*ch;
              f4 = e2*sh;
              px = Math.pow(x,3.);
              return (1.+l1)*(f1-f2)-
                   x*(2.*l1*f4+3.*l2*x*(f1+f2))
+px*l2*(2.*f3+4.*l1*((f3-1.)+
                   x*(f1-f2)));
             }

    class Curve extends JComponent {
    double esfmax, esfmin, dif, tr;
    Color cColor;
        public void paintComponent(Graphics gc) {
          Graphics2D gc2D = (Graphics2D) gc;
          super.paintComponent(gc2D);
            // Abmessungen Grafikobjekt
            int hc = getSize().height -30;
            bc = getSize().width - 20;
            // maximale, minimale Kurvenwerte
            esfmax = esf(0, ew);
            esfmin = esfmax;
            float strichDicke = 2.0f;
            BasicStroke stroke = new BasicStroke(strichDicke);
            gc2D.setStroke(stroke);
            cColor = new Color(255,130,38);
            for (int x = 0 ; x < bc ; x++) {
                double fr = esf(x + 1, ew);
                if (esfmax <  fr) {
                    esfmax = fr;
```

```
                }
                if (esfmin > fr) {
                   esfmin = fr;
                }
            }
            // Achsenhoehe Ordinate
            dif = esfmax - esfmin;
            // Verschiebung Null-Linie
            tr = hc * Math.abs(esfmin) / dif;
            gc2D.setColor(cColor);
            if(ew <= 0.) {
                tr = hc / 2;
                    if(ew < 0.) {
                        ausgabeEigenWert.setBackground(Color.RED);

ausgabeEigenWert.setText("Sch"+"\u00E4"+"tzwert unpassend");
                    }
            } else {
                // Zeichnen der normierten Kurve
                ausgabeEigenWert.setBackground(fbgColor);
                for (int ix = 0 ; ix < bc ; ix++) {
                gc2D.drawLine(ix + 10, (int) (hc * esf(ix, ew) /
dif + tr)
                + 10, ix + 11, (int) (hc * esf(ix + 1, ew) / dif +
tr)
                + 10);
                }
            }
            gc2D.setColor(Color.darkGray);
            // Einspannung
            gc2D.drawLine(10, (int) tr, 10, (int) tr + 20);
            // Null-Linie
            gc2D.drawLine(10, (int) tr + 10, bc + 10, (int) tr
+10);
            gc2D.drawString("\u00a9" + "pwill", 15, hc + 25);
        }

        double esf(int i, double e) {
            // Eigenschwingform
            double g1, g2, g3, g4, g5;
            double z = (double) i;
            // Normierung Abszisse
            z = z / bc;
            g1 = Math.exp(z*e);
            g2 = Math.sin(z*e);
            g3 = Math.cos(z*e);
            // Hyperbelfunktionen
            g4 = 0.5*(g1-1./g1);
            g5 = 0.5*(g1+1./g1);
            return (g3-g5+ratio*(g2-g4));
        }
    }
```

```
}

class ExitWindow extends WindowAdapter {
        public void windowClosing(WindowEvent e) {
          System.exit(0);
        }
        // Aufruf leerer WindowListener-Methoden
        public void windowIconified(WindowEvent we) {
        }
        public void windowOpened(WindowEvent we) {
        }
        public void windowClosed(WindowEvent we) {
        }
        public void windowDeiconified(WindowEvent we) {
        }
        public void windowActivated(WindowEvent we) {
        }
}
```