```
// Stuetzlinien
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Stuetzlinie extends JFrame implements ActionListener
{
     private static final long serialVersionUID = 0220;
     JTextField  ausgabeHDKn, eingabeP1, eingabeP2;
     JRadioButton sepBogen;
     double qqmin, qq = 0., hb = 0., lb = 0., hdkn = 0.;
     int bc, qqint;
     boolean flag;
     Color fbgColor, bgColor;
     private Curve diagram;
     JButton sl;
     // Konstruktor
     public Stuetzlinie() {
         Container fenster = getContentPane();
         fbgColor = new Color(178,214,252);
         fenster.setBackground(fbgColor);
         BorderLayout h1 = new BorderLayout();
         fenster.setLayout(h1);
         JLabel sepB = new JLabel ("Bogen", JLabel.RIGHT);
         sepBogen = new JRadioButton("separat",false);
         JLabel p1Wert = new JLabel("<html>q<sub>+</sub>/q<sub>o</
sub> : <html>", JLabel.RIGHT);
         eingabeP1 = new JTextField(12);
eingabeP1.setBorder(BorderFactory.createLoweredBevelBorder());
         JLabel p2Wert = new JLabel("h/b : ", JLabel.RIGHT);
         eingabeP2 = new JTextField(12);
eingabeP2.setBorder(BorderFactory.createLoweredBevelBorder());
         bgColor = new Color(140,181,222);
         sl = new JButton("<html>F<sub>H</sub>/q<sub>o</sub>b</
html>");
         sl.setBackground(bgColor);
         ausgabeHDKn = new JTextField(12);
         ausgabeHDKn.setBackground(fbgColor);
ausgabeHDKn.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(sepB);
         h2d.add(sepBogen);
         h2d.add(p1Wert);
         h2d.add(eingabeP1);
         h2d.add(p2Wert);
         h2d.add(eingabeP2);
         h2d.add(sl);
         sl.addActionListener(this);
         h2d.add(ausgabeHDKn);
```

```
         JLabel titel = new JLabel("St"+"\u00FC"+"tzlinie (Achsen
normiert)",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 Stuetzlinie();
          ExitWindow abbrechen = new ExitWindow();
          frame.addWindowListener(abbrechen);
          // Abfrage Bildschirmabmessungen
          Dimension dim =
Toolkit.getDefaultToolkit().getScreenSize();
          // Abmessungen des Applikationsfensters
          frame.setSize(360,384);
          // Positionierung des Applikationsfensters auf dem
Bildschirm
          xPos = (dim.width-360)/2;
          yPos = (dim.height-346)/2;
          frame.setLocation(xPos,yPos);
          // Anzeige des Rahmenfensters auf dem Desktop
          frame.setVisible(true);
     }

     public void actionPerformed(ActionEvent event) {
      if (event.getSource() == sl) {
           flag=true;
           qq = Double.parseDouble(eingabeP1.getText());
             if(qq < 0.) {
                 qq = Math.abs(qq);
                 eingabeP1.setText("" + qq);
```

```
            }
            hb = Double.parseDouble(eingabeP2.getText());
            if(hb < 0.) {
                hb = Math.abs(hb);
                eingabeP2.setText("" + hb);
            }
            if(qq < 0.00001) {
              qq = 0.000000000001;
              eingabeP1.setText("0" );
            }
          if(sepBogen.isSelected()) {
            // Newtonsches Naeherungsverfahren
            qqmin = hb;
            double inkrement = ns(qqmin)/nss(qqmin);
            while (Math.abs(inkrement/qqmin) > 1.e-7) {
              qqmin = qqmin - inkrement;
              inkrement = ns(qqmin)/nss(qqmin);
            }
            qqmin = qqmin - inkrement;
            qqint = (int) Math.round(qqmin*1000f);
            qqmin = qqint/1000f;
            if(qq < qqmin) {
               eingabeP1.setForeground(Color.RED);
               eingabeP1.setText("Fehler: Argument " + "\u2264" +
" " + (float) qqmin);
               flag=false;
               ausgabeHDKn.setText(" ");
            }
          }
      }
      if (flag) {
           eingabeP1.setForeground(Color.BLACK);
           lb = arch(1.+qq);
           // normierte, konstante, horizontale Komponente der
Druckkraft
           int ihdkn;
           hdkn = qq/(hb*lb*lb);
           ihdkn = (int) Math.round(1000.* hdkn);
           hdkn = ihdkn/1000.;
           ausgabeHDKn.setText((float) hdkn + " ");
           repaint();
      }
     }

     double arch(double x) {
       // Areakosinus-Hyperbolicus
       double e1, e2;
       e1 = x+Math.sqrt(x*x-1.);
       e2 = Math.log(e1);
       return e2;
     }
```

```
    double ns(double x) {
        // Nullstellengleichung
        double e1;
        e1 = x-hb*arch(1.+x);
        return e1;
    }

    double nss(double x) {
        // Ableitung Nullstellengleichung
        double e2;
        e2 = 1.-hb/Math.sqrt(x*(x+2.));
        return e2;
    }

 class Curve extends JComponent {
    private static final long serialVersionUID = 2020;
    double slfmax, slfmin, dif, tr;
    Color cColor;
    public void paintComponent(Graphics gc) {
         Graphics2D gc2D = (Graphics2D) gc;
 gc2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
         super.paintComponent(gc2D);
        // Abmessungen Grafikobjekt
        int hc = getSize().height -30;
        bc = getSize().width/2 - 10;
        // maximale, minimale Kurvenwerte
        slfmax = fsl(0, lb);
        slfmin = slfmax;
        float strichDicke = 2.0f;
        BasicStroke stroke = new BasicStroke(strichDicke);
        gc2D.setStroke(stroke);
        cColor = new Color(255,130,38);
        for (int x = -bc ; x < bc ; x++) {
            double fr = fsl(x + 1, lb);
            if (slfmax <  fr) {
                slfmax = fr;
            }
            if (slfmin > fr) {
               slfmin = fr;
            }
        }
        // Achsenhoehe Ordinate
        dif = slfmax - slfmin;
        tr = hc * Math.abs(slfmax) / dif;
        gc2D.setColor(cColor);
        // Zeichnen der normierten Kurve
        ausgabeHDKn.setBackground(fbgColor);
        for (int ix = -bc ; ix < bc ; ix++) {
            gc2D.drawLine(ix + 10 + bc, (int) (tr-hc * fsl(ix,
lb) / dif )
```

```
            + 10, ix + 11 + bc, (int) (tr-hc * fsl(ix + 1, lb) /
dif )+ 10);
          }
          gc2D.setColor(Color.darkGray);
          // Null-Linie
          gc2D.drawLine(10,hc+10, 2*(bc+5), hc+10);
          gc2D.drawString("\u00a9" + "pwill", 10, hc + 25);
          // Mittellinie
          gc2D.drawLine(10+bc,10, 10+bc, hc+20);
      }

      double fsl(int i, double lb) {
          double z = (double) i;
          // Normierung Abszisse
          z = z / bc;
          double e1, e2;
          e1 = Math.exp(lb*z);
          // Kosinus-Hyperbolicus
          e2 = 0.5*(e1+1./e1);
          // Stuetzlinie
          e2 = hb*(1.-e2)/qq+hb;
          return e2;
      }
  }
}

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) {
        }
}
```