Bereich: Threads (1)

Tanzende Schrift	**Musterlösung**
Package: `de.dhbwka.java.exercise.threads`	**Klasse:** `DancingText`

```java
package de.dhbwka.java.exercise.threads;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.util.Random;

import javax.swing.JComponent;
import javax.swing.JFrame;

/**
 * Part of lectures on 'Programming in Java'. Baden-Wuerttemberg
 * Cooperative State University.
 *
 * (C) 2016-2018 by W. Geiger, T. Schlachter, C. Schmitt, W. Suess
 *
 * @author DHBW lecturer
 * @version 1.1
 */
@SuppressWarnings( "serial" )
public class DancingText extends JComponent implements Runnable {

   private final static int XBASE = 30;
   private final static int XSTEP = 36;
   private final static int YBASE = 150;
   private final static Random RANDOM = new Random();

   private String text;
   private final long delay;

   private int colR = 0; // Color-Channel: red
   private int colG = 90; // Color-Channel: green
   private int colB = 180; // Color-Channel: blue
   private int yOffset = 0;

   public DancingText( String text, long delay ) {
      this.text = text;
      this.delay = delay;
      new Thread( this ).start(); // Start Thread
   }



   // Continued on next page
```

```
    /**
     * Paint dancing text on Graphics
     *
     * @param g
     *           graphics to use
     */
    @Override
    public void paintComponent( Graphics g ) {
       super.paintComponent( g );
       g.setFont( new Font( "Helvetica", Font.BOLD, 48 ) );
       for ( int i = 0; i < this.text.length(); i++ ) {
          char c = this.text.charAt( i );

          this.colR = (this.colR + 4 + DancingText.RANDOM.nextInt( 4 )) % 256;
          this.colG = (this.colG + 4 + DancingText.RANDOM.nextInt( 4 )) % 256;
          this.colB = (this.colB + 4 + DancingText.RANDOM.nextInt( 4 )) % 256;

          this.yOffset = DancingText.RANDOM.nextInt( 30 );

          g.setColor( new Color( this.colR, this.colG, this.colB ) );
          g.drawString( "" + c, DancingText.XBASE + i * DancingText.XSTEP,
                DancingText.YBASE - this.yOffset );
       }
    }

    @Override
    public void run() {
       while ( true ) {
          // Repaint and wait for delay
          this.repaint();
          try {
             Thread.sleep( this.delay );
          } catch ( InterruptedException e ) {
             System.err.println( "Interrupted!" );
          }
       }
    }

    public static void main( String[] args ) {
       // Create frame and add DancingText component
       JFrame f = new JFrame( "Dancing Text" );
       f.add( new DancingText( "Dancing Text :-)", 200 ) );
       f.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
       f.setSize( 640, 280 );
       f.setVisible( true );
    }

}
```

Bereich: Threads (1)

Ampel	**Musterlösung**
Package: `de.dhbwka.java.exercise.threads`	**Klasse:** `TrafficLight`

```java
package de.dhbwka.java.exercise.threads;

/**
 * Part of lectures on 'Programming in Java'. Baden-Wuerttemberg
 * Cooperative State University.
 *
 * (C) 2016-2018 by W. Geiger, T. Schlachter, C. Schmitt, W. Suess
 *
 * @author DHBW lecturer
 * @version 1.1
 */
public class LightPhase {

   private String name;
   private boolean red;
   private boolean yellow;
   private boolean green;
   private int duration;
   private LightPhase next;

   public LightPhase( String name, boolean red, boolean yellow, boolean green,
         int duration, LightPhase next ) {
      this.name = name;
      this.red = red;
      this.yellow = yellow;
      this.green = green;
      this.duration = duration;
      this.next = next;
   }

   public LightPhase( String name, boolean red, boolean yellow, boolean green,
         int duration ) {
      this( name, red, yellow, green, duration, null );
   }

   public String getName() {
      return this.name;
   }

   public void setName( String name ) {
      this.name = name;
   }

   public boolean isRed() {
      return this.red;
   }

   public void setRed( boolean red ) {
      this.red = red;
   }
```

```
    public boolean isYellow() {
        return this.yellow;
    }

    public void setYellow( boolean yellow ) {
        this.yellow = yellow;
    }

    public boolean isGreen() {
        return this.green;
    }

    public void setGreen( boolean green ) {
        this.green = green;
    }

    public int getDuration() {
        return this.duration;
    }

    public void setDuration( int duration ) {
        this.duration = duration;
    }

    public LightPhase getNext() {
        return this.next;
    }

    public void setNext( LightPhase next ) {
        this.next = next;
    }

}

// Continued on next page
```

```java
package de.dhbwka.java.exercise.threads;

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

import javax.swing.JComponent;
import javax.swing.JFrame;

/**
 * Part of lectures on 'Programming in Java'. Baden-Wuerttemberg
 * Cooperative State University.
 *
 * (C) 2016-2018 by W. Geiger, T. Schlachter, C. Schmitt, W. Suess
 *
 * @author DHBW lecturer
 * @version 1.1
 */
@SuppressWarnings( "serial" )
public class TrafficLight extends JComponent implements Runnable {

   private final static long DELAY = 500;

   private LightPhase currentPhase;
   // Possible alternative: list of phases and current phase index
   // private List<LightPhase> phases = new ArrayList<>();
   // private int currentPhase = 0;

   public TrafficLight() {
      // LightPhase stores successor, like an endless, round robin, linked list
      LightPhase redYellow = new LightPhase( "Rotgelb", true, true, false, 1 );
      LightPhase red =
            new LightPhase( "Rot", true, false, false, 10, redYellow );
      LightPhase yellow = new LightPhase( "Gelb", false, true, false, 2, red );
      LightPhase green =
            new LightPhase( "Gr\u00FCn", false, false, true, 10, yellow );
      redYellow.setNext( green );

      this.currentPhase = green; // start with green
      new Thread( this ).start();
   }

   @Override
   public void run() {
      while ( true ) {
         try {
            this.repaint();
            Thread.sleep(
                  this.currentPhase.getDuration() * TrafficLight.DELAY );

            // get next phase via getNext
            this.currentPhase = this.currentPhase.getNext();
         } catch ( InterruptedException ex ) {
         }
      }
   }
```

```
   @Override
   public void paintComponent( Graphics g ) {
      super.paintComponent( g );

      // traffic light box
      g.setColor( Color.BLACK );
      g.fillRect( 10, 10, 80, 195 );

      // 3 x empty light
      g.setColor( Color.WHITE );
      g.fillOval( 23, 23, 54, 54 );
      g.fillOval( 23, 83, 54, 54 );
      g.fillOval( 23, 143, 54, 54 );

      // draw colored lights, if active
      if ( this.currentPhase.isRed() ) {
         g.setColor( Color.RED );
         g.fillOval( 25, 25, 50, 50 );
      }
      if ( this.currentPhase.isYellow() ) {
         g.setColor( Color.YELLOW );
         g.fillOval( 25, 85, 50, 50 );
      }
      if ( this.currentPhase.isGreen() ) {
         g.setColor( Color.GREEN );
         g.fillOval( 25, 145, 50, 50 );
      }
   }

   public static void main( String[] args ) {
      // Create frame and add TrafficLight component
      JFrame f = new JFrame( "Traffic Light" );
      f.add( new TrafficLight() );
      f.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      f.setSize( 100, 260 );
      f.setVisible( true );
   }

}
```