Bereich: Zeichnen

Wahl	Musterlösung
Package: `de.dhbwka.java.exercise.ui.paint`	**Klasse:** `Election`

```java
package de.dhbwka.java.exercise.ui.paint;

import java.awt.Color;
import java.awt.Font;
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) 2016-2021 by T. Schlachter, C. Schmitt, W. Suess, N. Doms
 *
 * @author DHBW lecturer
 * @version 1.3
 */
@SuppressWarnings( "serial" )
public class Election extends JComponent {

   private final static int MARGIN_LEFT = 10;
   private final static int MARGIN_RIGHT = 10;
   private final static int MARGIN_TOP = 50;
   private final static Color TOP_COL = new Color( 0, 71, 129 );
   private final static Color BG_COL = new Color( 121, 170, 211 );
   private final static Color RULER_COL = new Color( 62, 133, 192 );
   private final static Font FONT = new Font( Font.SANS_SERIF, Font.BOLD, 16 );
   private final static String INFO_TEXT = "in %";

   private final Party[] parties;
   private final String electionName;

   private double maxPercentageOfParties = 0.0f;

   public Election( String electionName, Party[] parties ) {
      this.electionName = electionName;
      this.parties = parties;

      // find maximum percentage of all parties
      for ( Party p : parties ) {
         this.maxPercentageOfParties =
               Math.max( p.getPercentage(), this.maxPercentageOfParties );
      }
   }

   // Continued on next page
```

```java
@Override
public void paintComponent( Graphics g ) {
   final int width = this.getWidth();
   final int height = this.getHeight();

   int maxBarWidth = (width - Election.MARGIN_LEFT - Election.MARGIN_RIGHT)
         / this.parties.length;

   // Max height of bar = 2/3 of component height
   int totalBarHeight = height * 2 / 3;
   // height for 1 percent point
   double heightPerPercent = totalBarHeight / this.maxPercentageOfParties;
   // After calculating height for 1 percent, add margin top to total height
   totalBarHeight += Election.MARGIN_TOP;

   g.setFont( Election.FONT );

   g.setColor( Election.BG_COL ); // Background color
   g.fillRect( 0, 0, width, height ); // Background rectangle

   // Top background bar to write title on
   g.setColor( Election.TOP_COL );
   g.fillRect( 0, 0, width, 30 );

   g.setColor( Election.RULER_COL ); // set line/ruler color

   g.fillRect( 0, 30, width, 2 ); // line below top background bar
   // Thick line above bottom white bar
   g.fillRect( 0, totalBarHeight, width, 10 );
   // line below bottom white bar
   g.fillRect( 0, totalBarHeight + 40, width, 2 );
   // line below percentage of parties
   g.fillRect( 0, totalBarHeight + 65, width, 2 );

   g.setColor( Color.WHITE );

   // Bottom white bar
   g.fillRect( 0, totalBarHeight + 10, width, 30 );
   // Top title on the left
   g.drawString( this.electionName.toUpperCase(), Election.MARGIN_LEFT, 22 );
   // Top info on the right
   g.drawString( Election.INFO_TEXT, width - 40, 22 );

   // thin white helper lines (10% intervals)
   for ( int i = 1; i <= (int) (this.maxPercentageOfParties / 10); i++ ) {
      g.drawLine( 0, (int) (totalBarHeight - i * 10 * heightPerPercent),
            width, (int) (totalBarHeight - i * 10 * heightPerPercent) );
   }
   // draw parties with helper function
   for ( int i = 0; i < this.parties.length; i++ ) {
      int partyHeight =
            (int) (this.parties[i].getPercentage() * heightPerPercent);
      this.drawBar( g, Election.MARGIN_LEFT + i * maxBarWidth,
            totalBarHeight - partyHeight, maxBarWidth,
            partyHeight, this.parties[i] );
   }
}
```

```
   /**
    * Draw bar for party
    *
    * @param g
    *           graphics
    * @param x
    *           x (left) to draw from
    * @param y
    *           y (top) to draw from
    * @param width
    *           width of bar
    * @param height
    *           height of bar
    * @param party
    *           party
    */
   public void drawBar( Graphics g, int x, int y, int width, int height,
         Party party ) {
      int margin = 10;
      int shadow = 2;

      // draw shadow rectangle
      g.setColor( java.awt.Color.GRAY );
      g.fillRect( x + margin + shadow, y - shadow + 1, width - 2 * margin,
            height );
      g.setColor( party.getColor() ); // draw actual colored bar over shadow
      // above line
      g.fillRect( x + margin, y, width - 2 * margin, height );
      // below line
      g.fillRect( x + margin, y + height + 10, width - 2 * margin, 10 );

      g.setColor( java.awt.Color.BLACK );

      // Draw name of party
      g.drawString( party.getName(), x + margin, y + height + 36 );

      // Draw percentage value, but replace decimal '.' with ','
      g.drawString( Float.toString( party.getPercentage() ).replace( '.', ',' ),
            x + margin + 5, y + height + 60 );
   }
   public static void main( String[] args ) {
      JFrame frame = new JFrame( "Wahl" );
      frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      frame.setSize( 640, 480 );
      frame.add( new Election( "Bundestagswahl 2021", new Party[] {
            new Party( 16.4f, "SPD", Color.RED ),
            new Party( 28.6f, "Union", Color.BLACK ),
            new Party( 11.6f, "Gr\u00FCne", Color.GREEN ),
            new Party( 4.3f, "FDP", Color.YELLOW ),
            new Party( 20.8f, "AfD", Color.BLUE ),
            new Party( 8.8f, "Linke", Color.MAGENTA ),
            new Party( 0.2f, "SSW", new Color(0,73,167) ),
            new Party( 9.4f, "Andere", Color.DARK_GRAY )
      } ) );
      frame.setVisible( true );
   }
}
```

```java
package de.dhbwka.java.exercise.ui.paint;

import java.awt.Color;

/**
 * Part of lectures on 'Programming in Java'. Baden-Wuerttemberg
 * Cooperative State University.
 *
 * (C) 2016 by W. Geiger, T. Schlachter, C. Schmitt, W. Suess
 *
 * @author DHBW lecturer
 * @version 1.3
 */
public class Party {

   private float percentage;
   private String name;
   private Color color;

   // Extension for seats
   private int seats;

   public Party( float percentage, String name, Color color ) {
      this.percentage = percentage;
      this.name = name;
      this.color = color;
   }

   // Extension for seats
   public Party( float percentage, String name, Color color, int seats ) {
      this( percentage, name, color );
      this.seats = seats;
   }

   public float getPercentage() {
      return this.percentage;
   }

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

   public Color getColor() {
      return this.color;
   }

   // Extension for seats
   public int getSeats() {
      return this.seats;
   }

}
```

Sternchenaufgabe / Sitzverteilung

```java
package de.dhbwka.java.exercise.ui.paint;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.GridLayout;
import java.util.ArrayList;
import java.util.List;

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

@SuppressWarnings( "serial" )
public class ElectionSeats extends JComponent {

   private final static int MARGIN_LEFT = 10;
   private final static int MARGIN_TOP = 50;
   private final static Color TOP_COL = new Color( 0, 71, 129 );
   private final static Color BG_COL = new Color( 121, 170, 211 );
   private final static Color RULER_COL = new Color( 62, 133, 192 );
   private final static Font FONT =
         new Font( Font.SANS_SERIF, Font.BOLD, 16 );

   // filtered party list (only with seats!)
   private List<Party> parties;
   private String title;
   private int totalSeats;

   public ElectionSeats( String electionName, Party[] parties ) {
      this.title = electionName;
      this.parties = new ArrayList<>();

      // calculate total count of seats
      for ( Party p : parties ) {
         this.totalSeats += p.getSeats();
         // and add party only if it has seats!
         if ( p.getSeats() > 0 ) {
            this.parties.add( p );
         }
      }
   }

   public static void main( String[] args ) {
      JFrame frame = new JFrame( "Wahl" );
      frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      frame.setSize( 1280, 480 );
      frame.setLayout( new GridLayout( 1, 2 ) );

      // use additional constructor with seat count
      Party[] parties = new Party[] {
            new Party( 28.6f, "Union", Color.BLACK, 208 ),
            new Party( 16.4f, "SPD", Color.RED, 120 ),
            new Party( 20.8f, "AfD", Color.BLUE, 152 ),
            new Party( 8.8f, "Linke", Color.MAGENTA, 64 ),
            new Party( 11.6f, "Gr\u00FCne", Color.GREEN, 85 ),
            new Party( 0.2f, "SSW", new Color(0,73,167), 1 )
      };

      // show election and seat view side by side
      Election election = new Election( "Bundestagswahl 2025", parties );
      ElectionSeats seats = new ElectionSeats( "Sitzverteilung", parties );

      frame.add( election );
      frame.add( seats );
      frame.setVisible( true );
   }

   @Override
   public void paintComponent( Graphics g ) {
      // For a better understanding the drawing is divided in multiple sub-parts
      // => some values are calculated multiple times and should be shared
      // across the methods in a non-demo environment

      this.drawBackground( g );

      this.drawPieChart( g );

      this.drawPartyInfo( g );
   }

   private void drawBackground( Graphics g ) {
      final int width = this.getWidth();
      final int height = this.getHeight();

      // Max height of bar = 2/3 of component height
      int baselineDiagramY = height * 2 / 3 + ElectionSeats.MARGIN_TOP;

      g.setFont( ElectionSeats.FONT );

      // Background
      g.setColor( ElectionSeats.BG_COL );
      g.fillRect( 0, 0, width, height );

      // Top background bar to write title on
      g.setColor( ElectionSeats.TOP_COL );
      g.fillRect( 0, 0, width, 30 );

      // set line/ruler color
      g.setColor( ElectionSeats.RULER_COL );

      // line below top background bar
      g.fillRect( 0, 30, width, 2 );
      // Thick line above bottom white bar
      g.fillRect( 0, baselineDiagramY, width, 10 );
      // line below bottom white bar
      g.fillRect( 0, baselineDiagramY + 40, width, 2 );
      // line below percentage of parties
      g.fillRect( 0, baselineDiagramY + 65, width, 2 );

      g.setColor( Color.WHITE );

      // Bottom white bar
```

```java
      g.fillRect( 0, baselineDiagramY + 10, width, 30 );
      // Top title on the left
      g.drawString( this.title.toUpperCase(), ElectionSeats.MARGIN_LEFT, 22 );
      // Top info on the right
      g.drawString( Integer.toString( this.totalSeats ), width - 40, 22 );
   }

   private void drawPieChart( Graphics g ) {
      // seats distribution is usually shown with a pie chart
      // based on a semicircle
      // ==> seats must be distributed on 180 degrees

      // Important: avoid integer division!
      float degreePerSeat = 180f / this.totalSeats;

      // width of circle, use 90% of component
      int cirleWidth = (int) (this.getWidth() * 0.9f);

      // height of circle, use 100% of height to
      // get 50% pie chart (remember: semicircle!)
      int circleHeight = this.getHeight();

      // X value of circle is at 5% of width
      // (90% circle with means 5% padding on each side)
      int cirlceX = (int) (this.getWidth() * 0.05f);

      // circle y (same as baselineDiagramY from background)
      int circleY = this.getHeight() * 2 / 3 + ElectionSeats.MARGIN_TOP;

      // current arc start angle, since we want to
      // start on the left, its's 180°
      int arcStart = 180;

      // let's draw the parties
      for ( Party party : this.parties ) {
         // First: part of the pie chart
         int angle = Math.round( party.getSeats() * degreePerSeat );

         g.setColor( party.getColor() );
         g.fillArc( cirlceX, circleY - circleHeight / 2, cirleWidth,
               circleHeight, arcStart - angle, angle );
         arcStart -= angle;
      }
   }

   private void drawPartyInfo( Graphics g ) {
      int width = this.getWidth();
      int height = this.getHeight();
      int count = this.parties.size();
      int gap = 20;

      // Diagram y, see background
      int baseY = height * 2 / 3 + ElectionSeats.MARGIN_TOP;



      // party color rectangle with:
```

```java
      // total area to draw: 90% (like circle)
      // minus all gaps (gap count: party count -1)
      int colorRectWidth = (int) ((width * 0.9f - (count - 1) * gap) / count);

      // align start x to circle
      int partyInfoX = (int) (this.getWidth() * 0.05f);

      // set font
      g.setFont( ElectionSeats.FONT );
      for ( Party party : this.parties ) {
         g.setColor( party.getColor() );
         g.fillRect( partyInfoX, baseY + 10, colorRectWidth, 10 );

         g.setColor( Color.BLACK );

         // draw party name
         g.drawString( party.getName(), partyInfoX, baseY + 36 );

         // draw seat count
         g.drawString( Integer.toString( party.getSeats() ), partyInfoX,
               baseY + 60 );

         // offset x by rectangle with + gap
         partyInfoX += colorRectWidth + gap;
      }

   }

}
```