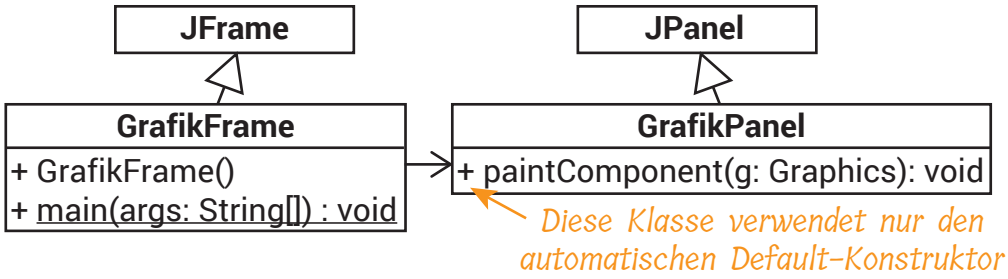
Grafiken

In Java kann auf jeder beliebigen GUI-Komponente (eine GUI-Komponente ist ein Objekt, dessen Klasse von `javax.swing.Component` erbt) gezeichnet werden. Damit die Grafik auch in einem Layout eingebunden werden kann, ist es sinnvoll die Grafik auf einer eigenen GUI-Komponente zu erstellen, die nur dafür verwendet wird. Als Basis dafür eignet sich die Klasse `JPanel`.

Grundstruktur

Für eine GUI-Oberfläche, die eine eigene Grafik einbindet, ergibt sich also folgende Grundstruktur



Implementierung der Grafik

Ausgangspunkt ist also eine Klasse, die von der Klasse `JPanel` erbt. Zur Umsetzung der eigentlichen Grafik muss die `paintComponent`-Methode überschrieben werden. Diese Methode wird von Java automatisch aufgerufen, wenn eine GUI-Komponente angezeigt wird. Damit müssen zur Umsetzung der Zeichnung folgende Schritte durchgeführt werden:

1. @Override verwenden: Damit wird Java mitgeteilt, dass ich eine Methode überschreiben möchte (Verringertes Risiko durch Tippfehler die Methode gar nicht zu überschreiben).
2. Die `paintComponent`-Methode der Superklasse aufrufen, um einmal die Basis-Darstellung der Komponente zu erhalten.
3. Das von Java an `paintComponent` als Parameter übergebene Graphics-Objekt verwenden, um darauf zu zeichnen.

Diese Schritte sind im unteren Beispiel-Quellcode ersichtlich:

```
import javax.swing.JPanel;
import java.awt.*;

public class GrafikPanel extends JPanel {

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

    g.setColor(Color.RED);
    g.fillRect(10, 10, 300, 100);

    Color c = new Color(0,0,150);
    g.setColor(c);
    g.fillOval(20, 20, 280, 80);

    g.setColor(Color.WHITE);
```

Damit die Grafik später in die GUI eingebettet werden kann, muss als Super-Klasse eine GUI-Komponente verwendet werden, z.B: JPanel.

Die Methode paintComponent wird überschrieben und mit @Override annotiert (d.h. gekennzeichnet)

Die Methode paintComponent der Super-Klasse wird hier aufgerufen, um die Basis zu zeichnen

Hier wird die Zeichen-Farbe mit Hilfe einer Color-Konstanten auf Rot gesetzt.

Das Rechteck wird hier in Rot auf das Graphics-Objekt gezeichnet

Hier wird mit einem eigens erstellten Color-Objekt eine selbst gewählte Farbe gesetzt

```
        Font f = new Font(Font.SANS_SERIF, Font.BOLD, 24);
        g.setFont(f);
        g.drawString("Ich mag SEW", 80, 67);
        g.drawLine(80, 70, 235, 70);
    }
}
```

Durch setzen eines neuen Font-Objekts wird für das gesamte Graphics-Objekt eine neue Schriftart festgelegt

Die Klasse `Graphics`

Das `Graphics`-Objekt, das der Methode `paintComponent` als Parameter mitgegeben wird, enthält Informationen über folgende Punkte

- Den Zeichenbereich
- Welche Farbe gerade aktuell ist
- Welche Schriftart gerade aktuell ist

Beim Aufruf der Zeichen-Methoden wird immer die gerade aktuelle Einstellung übernommen. Das bedeutet z.B. eine Farbe wird solange verwendet, bis eine andere Farbe gesetzt wird. So wird im gezeigten Beispiel sowohl die `drawString`-Methode als auch die darauffolgende `drawLine`-Methode mit der gleichen Farbe ausgeführt. Gleiches gilt für die aktuelle Schriftart. Des Weiteren überlagern die Zeichen-Aufrufe einander. Damit wird das zuletzt gezeichnete Element zuoberst dargestellt und zuvor gezeichnete Elemente unter Umständen überzeichnet.

Die Klasse `Graphics` besitzt viele unterschiedliche Zeichen-Methoden. Die meisten davon existieren in einer `draw`*`Xyz`* oder einer `fill`*`Xyz`*-Variante. Die `draw`-Methoden zeichnen nur die Umrisslinie der ausgewählten Figur, während die `fill`-Methoden zusätzlich noch die Fläche mit der aktuellen Farbe ausfüllen, z.B.

`void`	`drawOval(int x, int y, int width, int height)` Draws the outline of an oval.
`void`	`fillOval(int x, int y, int width, int height)` Fills an oval bounded by the specified rectangle with the current color.

Weitere praktische Zeichen-Methoden, die hier nur in der `draw`-Variante angegeben sind, aber auch in der `fill`-Variante existieren:

`void`	`drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)` Draws the outline of a circular or elliptical arc covering the specified rectangle.
`void`	`drawPolygon(int[] xPoints, int[] yPoints, int nPoints)` Draws a closed polygon defined by arrays of x and y coordinates.
`void`	`drawRect(int x, int y, int width, int height)` Draws the outline of the specified rectangle..
`void`	`drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)` Draws an outlined round-cornered rectangle using this graphics context's current color.

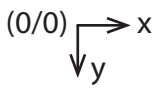
Von den folgenden 2 Methoden gibt es nur die `draw`-Variante:

`void`	`drawLine(int x1, int y1, int x2, int y2)` Draws a line, using the current color, between the points (x1, y1) and (x2, y2) in this graphics context's coordinate system.
`void`	`drawString(String str, int x, int y)` Draws the text given by the specified string, using this graphics context's current font and color.

Einheiten und Koordinatensystem

Das Java Graphics-Objekt verwendet ein eigenes Koordinaten-System namens User Space, das aber im Allgemeinen 1:1 auf das Koordinatensystem des Anzeigegerätes (z.B. Monitor) umgesetzt wird. Damit entspricht eine 1 Einheit in Java meistens einem Pixel.

Das Koordinatensystem beginnt in Java in der linken oberen Ecke mit den Koordinaten (0/0) und hat dann nach rechts die positive x-Richtung und nach unten die positive y-Richtung. Damit können negative Koordinaten zwar angegeben werden, liegen aber auf jeden Fall außerhalb des dargestellten Bereiches.



Farben in Java

Die Darstellung von Farben in Java wird über die Klasse `Color` aus dem Package `java.awt` geregelt. Damit können Farben selbst definiert werden oder vordefinierte Farben verwendet werden. Die vordefinierten Farben sind in der folgenden Tabelle mit ihren RGB-Werten angegeben.

Farbname	Rot	Grün	Blau
WHITE	255	255	255
BLACK	0	0	0
RED	255	0	0
GREEN	0	255	0
BLUE	0	0	255
YELLOW	255	255	0
MAGENTA	255	0	255
CYAN	0	255	255
PINK	255	175	175
ORANGE	255	200	0
LIGHT_GRAY	192	192	192
DARK_GRAY	64	64	64

Die angegebenen Farbnamen sind statische Konstanten (Klassenkonstanten) in der Klasse `Color`. Wie bei Methoden bedeutet das `static`, dass sie über den Klassennamen aufgerufen werden, also mit *Klassennamen.Konstantennamen*. Um z.B. die Zeichenfarbe des `Graphics`-Objektes `g` auf gelb zu setzen, kann man folgende Code-Zeile schreiben:

```
g.setColor(Color.YELLOW);
```

Eigene Farben

Man kann auch eigene Farben erzeugen. Dazu muss man eigene `Color`-Objekte erzeugen. Zu diesem Zweck stellt die Klasse `Color` u.a. folgende Konstruktoren zur Verfügung.

`Color(float r, float g, float b)` Creates an opaque sRGB color with the specified red, green, and blue values in the range (0.0 - 1.0).
`Color(int r, int g, int b)` Creates an opaque sRGB color with the specified red, green, and blue values in the range (0 - 255).

Beim erste Konstruktor können die Farbanteile für Rot, Grün und Blau in Prozenten (0.9 = 90%) angegeben werden. Beim zweiten kann man hingegen die Farbanteile, wie bei vielen Grafikprogrammen üblich, in Werten von 0-255 (hexadezimal 0x00 bis 0xFF) übergeben. Folgende Code-Zeilen setzen z.B. die Farbe des Graphics-Objektes g auf ein helles Blau:

```
Color hbl = new Color(200, 200, 255);
g.setColor(hbl);
```

bzw. in Hexadezimal-Darstellung:

```
Color hbl = new Color(0xC8, 0xC8, 0xFF);
```

oder mit Float-(Prozent-)Werten:

```
Color hbl = new Color(0.8f, 0.8f, 1.0f);
```

Anzeigen der Grafik in einem JFrame

Die Grafik muss nun für die Anzeige in einen **Top-Level-Container** eingebettet werden

```
import javax.swing.*;

public class GrafikFrame extends JFrame {
  public GrafikFrame() {
    super("Grafik-Beispiel");
    GrafikPanel gp = new GrafikPanel();
    this.add(gp);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setBounds(100, 200, 400, 200);
    this.setVisible(true);
  }
  public static void main(String[] args) {
    new GrafikFrame();
  }
}
```

Instanzieren der selbst erstellten Grafik-Komponente

Nachdem nur die Grafik-Komponente als Inhalt angezeigt wird, wird diese direkt zur Content-Pane hinzugefügt

Größe angeben! `preferredSize` ist beim GrafikPanel-Objekt nicht gesetzt

Sichtbar machen (sonst wird nichts angezeigt)

Das Resultat des dargestellten Programmes sieht dann folgendermaßen aus:

Anpassbare Grafiken:

Die obige Grafik arbeitet mit festen Koordinaten. Das bedeutet, dass sie sich nicht an die Größe der Grafik-Komponente anpassen kann. Zu diesem Zweck muss man aber zunächst diese Größe erfragen. Dazu kann man die Methoden `getWidth` und `getHeight` verwenden. Diese werden aus der Klasse `JComponent` geerbt und stehen damit jeder GUI-Komponente zur Verfügung.

`int`	`getHeight()` Returns the current height of this component.
`int`	`getWidth()` Returns the current width of this component.

Die Breite und Höhe können dann dazu verwendet, um die Grafik an die Größe anzupassen, oder das Zeichenelement in die Mitte zu rücken oder auch um vielleicht bei einem Linienraster mehr oder weniger Linien zu zeichnen. Im unten gegebenen Beispiel wird das Rechteck an die Größe des Panels angepasst und die weiteren Elemente (ungefähr) zentriert, wobei die Abmessungen der Schrift geschätzt wurden.

```java
import javax.swing.JPanel;
import java.awt.*;

public class GrafikPanel extends JPanel {

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

    int mitteX = super.getWidth()/2;
    int mitteY = super.getHeight()/2;

    g.setColor(Color.RED);
    g.fillRect(10, 10, super.getWidth()-20, super.getHeight()-20);

    Color c = new Color(0,0,150);
    g.setColor(c);
    g.fillOval(mitteX-140, mitteY-40, 280, 80);

    g.setColor(Color.WHITE);
    Font f = new Font(Font.SANS_SERIF, Font.BOLD, 24);
    g.setFont(f);
    g.drawString("Ich mag SEW", mitteX-78, mitteY+7);
    g.drawLine(mitteX-80, mitteY+15, mitteX+80, mitteY+15);
  }
}
```

Berechnen der horizontalen Mitte mit Hilfe der Methode getWidth() der Super-Klasse

Berechnen der vertikalen Mitte mit Hilfe der Methode getHeight() der Super-Klasse

Das Rechteck an die Panel-Größe anpassen mit jeweils 10 px Rand an jeder Seite

Das Oval zentrieren

Den Text zentrieren

Die Linie mittig unter den Text platzieren.

Wenn diese Grafik geladen wird, dann ist sie zentriert und das Rechteck füllt die Fläche aus:

Auch wenn die Größe des Fensters geändert wird, bleibt die Zentrierung erhalten:

Habe ich es verstanden:

- Ich kann eine eigene GUI-Komponente basierend auf einem `JPanel` erstellen
- Ich kann `paintCompontent` überschreiben, um mit Hilfe von `draw` und `fill`-Methoden Grafiken auf die Oberfläche einer GUI-Komponente zeichnen
- Ich kann mit `Color`-Objekten arbeiten
- Ich kann mit `Font`-Objekten arbeiten
- Ich kann mit Hilfe von `getWidth` und `getHeight` Grafiken an die Fenstergröße anpassen.
- Ich kann die Grafik-Komponente in einem `JFrame` anzeigen.