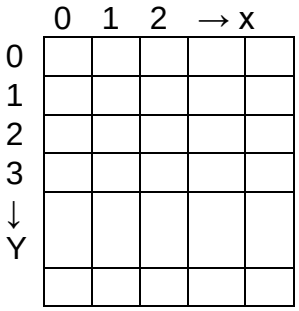
1 Grundlagen

Grundlegende 2D-Grafikfunktionen werden mit der Klasse Graphics zur Verfügung gestellt. Sie bietet Methoden zum Zeichnen von Linien, Kreisen, Rechtecken, Texten etc. Flächen können gefüllt werden, Farben und Schriftarten können eingestellt werden. Auf diese Weise können einfache 2-dimensionale Zeichnungen erstellt werden.

Ein `Graphics`-Objekt stellt in Java ein universelles Ausgabegerät dar. Die Ausgabe erfolgt in einem Koordinatensystem, dessen Ursprung in der linken oberen Ecke liegt.
Die x-Achse verläuft nach rechts, die y-Achse verläuft nach unten.

Seine Einheit ist ein Pixel.



Die Ausgabe der Grafik erfolgt durch Überlagerung der geerbten Methode `paint()`. Die Methode `paint()` wird automatisch aufgerufen, sobald das Fenster – z.B. durch Größenänderungen des Fensters – neu gezeichnet werden muss.
Soll das Fenster neu gezeichnet werden, ohne dass die Größe geändert wurde, so kann `repaint()` verwendet werden.

1.1 Programmierung

1.1.1 Einfaches Beispiel

```
public class GraphicsDemo
{
	private JFrame frame;

	public static void main(String[] args)
	{
		GraphicsDemo gd = new GraphicsDemo();
	}

	public GraphicsDemo()
	{
		frame = new JFrame(); // Fensterobjekt erzeugen
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		MyPanel pnlGraphik = new MyPanel(); // Objekt der inneren Klasse
		frame.getContentPane().add(pnlGraphik, BorderLayout.CENTER);
		frame.pack();
		frame.setSize(300, 200);
		frame.setVisible(true);
	}
```

```
	// Innere Klasse als Kindklasse von JPanel
	// geerbte Methode paint() wird überschrieben
	private class MyPanel extends JPanel
	{
		public void paint(Graphics g)
		{
			g.setFont(new Font("Courier New", Font.BOLD, 30));
			g.drawString("Hallo Welt", 50, 100);
			g.drawOval(45, 65, 200, 50);
		}
	}
}
```

1.1.2 Beispiel 2: Benutzung der Klasse `Graphics` als Teil einer GUI gemeinsam mit weiteren Komponenten

Soll eine 2D-Grafik zusammen mit anderen Komponenten in einer GUI verwendet werden, so kann die von `JPanel` abgeleitete Klasse wie gewohnt mit weiteren GUI Komponenten kombiniert werden. Im folgenden Beispiel wird die Klasse nicht als innere Klasse, sondern als eigenständige Klasse angelegt.

Beispiel:

```
public class MalPanel extends JPanel
{
	private String sText = "Hallo Welt";

	public void setsText(String sText)
	{
		this.sText = sText;
	}

	public void paint(Graphics g)
	{
		super.paint(g);   // Löscht den alten Inhalt in der Grafikebene
		g.setFont(new Font("Courier New",Font.BOLD,30));
		g.drawString(sText, 50, 100);
		g.drawOval(45, 65, 200, 50);
	}
}
```

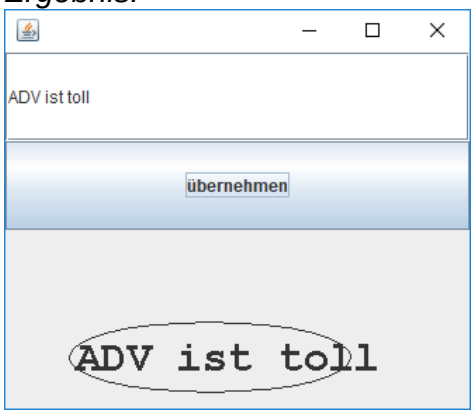
Verwendung von MalPanel in der GUI:

```
public class GraphicsDemo2 extends MouseAdapter
{
	private JFrame frame;
	private JTextField txtEingabe = new JTextField("ADV ist ");
	private MalPanel pnlMalen = new MalPanel();
	public GraphicsDemo2()
	{
		frame = new JFrame();
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		Container c = frame.getContentPane();
		c.setLayout(new GridLayout(2,1));
		JPanel pnlEingabe = new JPanel();
		JButton btnUebernehmen = new JButton ("übernehmen");
		pnlEingabe.setLayout(new GridLayout(2,1));
		pnlEingabe.add(txtEingabe);
		pnlEingabe.add(btnUebernehmen);
		btnUebernehmen.addMouseListener(this);
		c.add(pnlEingabe);
		c.add(pnlMalen);
	}

	public void mouseClicked(MouseEvent e)
	{
		pnlMalen.setsText(txtEingabe.getText());
		// Expliziter Aufruf der paint() Methode
		pnlMalen.paint(pnlMalen.getGraphics());
	}

	public static void main(String[] args)
	{
		GraphicsDemo2 gd = new GraphicsDemo2();
		gd.frame.pack();
		gd.frame.setSize(300, 400);
		gd.frame.setVisible(true);
	}
}
```

Ergebnis:

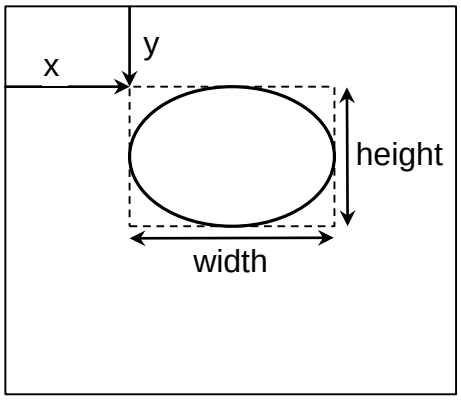


2 Zeichnen einfacher geometrischer Figuren

Die Graphics Klasse der AWT Bibliothek stellt eine Reihe von Zeichenmethoden zum Zeichnen von allgemein gebräuchlichen geometrischen Figuren zur Verfügung.

Viele dieser Zeichenmethoden benötigen als Übergabeparameter die x- und y-Koordinaten der linken oberen Ecke und die Angabe der Breite und Höhe, um den entsprechenden Zeichenbereich zu bestimmen. Beispiel für Kreise (bzw. Ovale):

Zeichenfläche z.B. `JPanel`



Die Parameter werden jeweils in Pixel angegeben.

```
void drawOval(int x, int y, int width,
              int height)
```

Zeichnet den Umriss eines Kreises oder Ovals in der voreingestellten Farbe.

```
void fillOval(int x, int y, int width,
              int height)
```

Zeichnet einen gefüllten Kreis/Oval in der voreingestellten Farbe.

Zum Zeichnen von Rechtecken bzw. Quadraten verwendet man:

```
void drawRect(int x, int y, int width, int height)
void fillRect(int x, int y, int width, int height)
```

Ein Dreieck kann man folgendermaßen zeichnen:

```
Graphics g = getGraphics();
int[] x_Polygon = { 50, 40, 60, 50 }; // 'array' für x-Koordinaten eines
                                      // Polygons
int[] y_Polygon = { 50, 70, 70, 50 }; // 'array' für y-Koordinaten eines
                                      // Polygons
g.setColor(Color.red); // Farbe: rot

g.fillPolygon(x_Polygon, y_Polygon, 4); // Dreieck zeichnen
```

Festlegung der Zeichenfarbe:

Wie man in dem Beispiel sieht kann man mit Hilfe der Methode
`public void setColor(Color c)`

die Zeichenfarbe festlegen.

Mit Hilfe der Klasse `Color` können Farben definiert werden. Die Klasse enthält eine Reihe von symoblischen Konstanten für gängige Farben wie `Color.red, Color.green, ...`
Ansonsten kann man mit Hilfe der Konstruktoren der Klasse `Color` beliebige Farben aus dem Farbraum definieren.

Am gebräuchlichsten ist der Konstruktor, der drei Werte im Bereich 0..255 als Integer entgegen nimmt:

```
Color(int r, int g, int b)
```

Die Parameter stehen für die Grundfarben der additiven Farbmischung: Rot, Grün und Blau.

Mit einem vierten Parameter kann man noch den Deckungsgrad im Bereich 0 .. 255 festlegen:

```
Color(int r, int g, int b, int a)
```

Bsp.

```
// Grau mit 80% Deckungsgrad
g.setColor(new Color(127, 127, 127, (int)(0.8 * 255)));
```