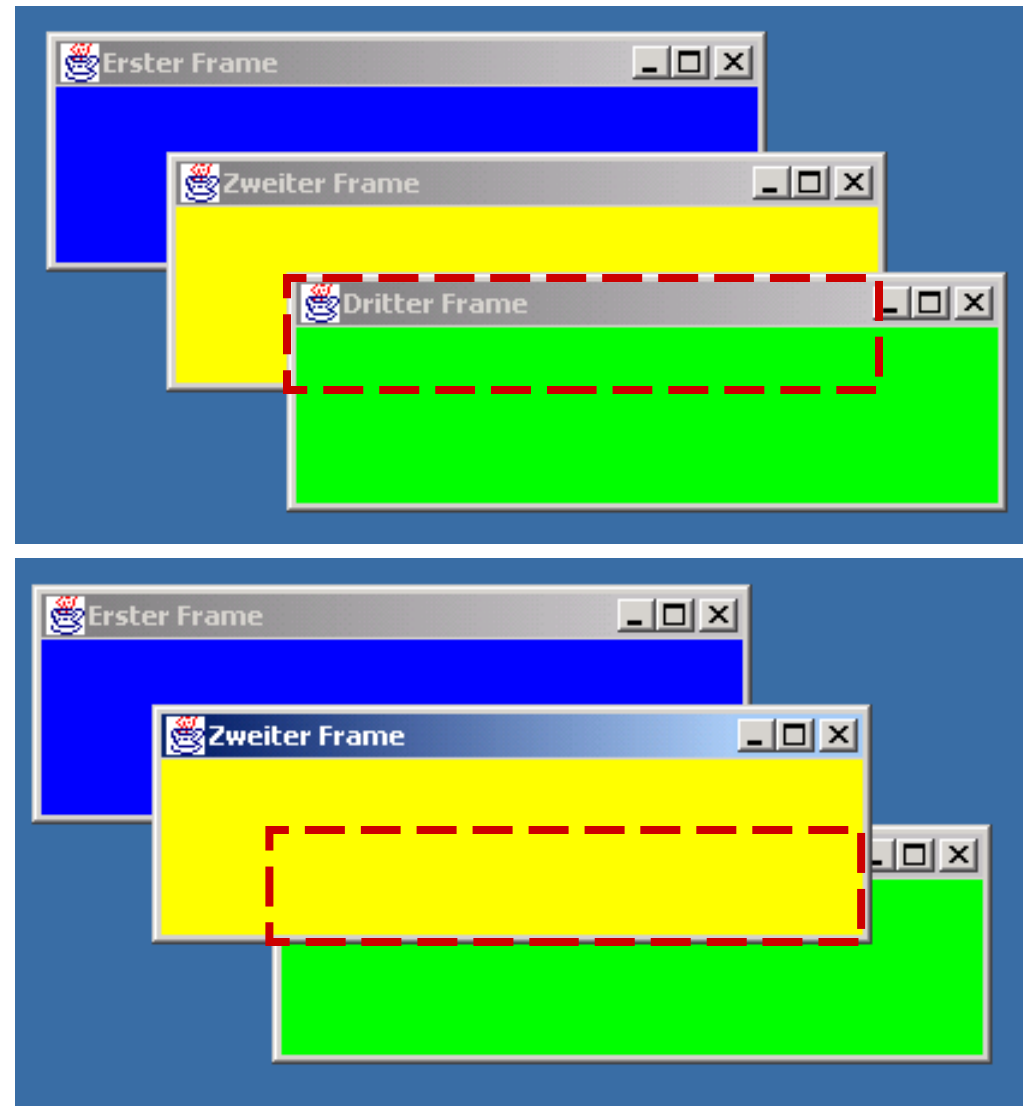


3. Graphikausgabe

3.1 Das Paint-Ereignis

Vorbemerkungen

- In diesem Kapitel wird ein grober Überblick über die Graphik-Ausgabe in AWT gegeben
- Insbesondere wird das **paint-Ereignis** und dessen Behandlung als erster Ereignistyp besprochen



3.1 Das Paint-Ereignis

Paint

- Ein Paint-Ereignis wird dann ausgelöst
 - wenn das Fenster zum ersten Mal auf den Bildschirm kommt
 - wenn ein Teil des Fensters aufgrund voriger Überdeckung durch andere Fenster repariert werden muß; das kann vorkommen durch
 - Window-Manager-Aktionen
 - Änderung der Window-Hierarchie oder der Window-Geometrie
- Das Paint-Ereignis wird in AWT an die Methode **paint()** ausgeliefert. Diese ist bei **Component** deklariert
- Jede Komponente, die sich zeichnen will, überlädt die paint-Methode

3.1 Das Paint-Ereignis

Beispielprogramm: PaintDemo1

```
public class PaintDemo1 extends Frame {  
    private int paintCount=0;  
  
    public PaintDemo1() {  
        setTitle("PaintDemo1"); setSize(200,200);  
        setLocation(200,200); setBackground(Color.green);  
    }  
  
    public void paint (Graphics g) {  
        paintCount++;  
        System.out.println("Paint: " + paintCount);  
    }  
main() siehe Demoprogramm  
}
```

3.1 Das Paint-Ereignis

Fälle

- Überdecken durch ein anderes Fenster, dann Wegbewegen des Fensters
- Aufziehen des Fensters (Vergrößerung)
- Ändern der Einstellungen im Window-System auf nicht permanente Aktualisierung

Unter WinXX: Systemsteuerung → Anzeige → Effekte → „Fensterinhalt beim Ziehen anzeigen“

3.2 Graphik-Primitiven

Graphik-Ausgabe-Operationen

- beinhalten immer drei Parameter
 - das **Window**, in das gezeichnet wird, denn davon hängt das zugrunde liegende Koordinatensystem ab
 - die **Primitive(n)**, d.h. die geometrische Figur
 - die **graphischen Attribute**, d.h. die Optik der Ausgabe

Allgemeine Ausgabe-Operation (Pseudocode)

```
draw ( Window w, Prim[] p, Attr[] a );
```

3.2 Graphik-Primitiven

Allgemeine Ausgabe-Operation (Pseudocode)

```
draw ( Window w, Prim[] p, Attr[] a );
```

oder

```
draw_Prim1 ( Window w, Geom[] g, Attr[] a );
```

```
draw_Prim2 ( Window w, Geom[] g, Attr[] a );
```

```
...
```

```
draw_PrimN ( Window w, Geom[] g, Attr[] a );
```

Prim	Primitiven-Klassen
------	--------------------

Geom	Geometrie-Klassen
------	-------------------

Window	Window-Klasse
--------	---------------

Attr	Graphische Attribute
------	----------------------

3.2 Graphik-Primitiven

Beispiele (Pseudocode)

Primitiven-Klassen

Point

Line

Polyline

Rectangle

Arc

Text

Geometrie-Klassen

Point

Line

Polyline

Graphische Attribute

Color

Line Width

Line Style

Line Color

Font Family

Font Size

Font Style

Fill Pattern

Mode

3.2 Graphik-Primitiven

Beispiele (Pseudocode)

Operationen

```
drawPoint ( Window w, Geom[] g, Attr[] a );  
drawLine ( Window w, Geom[] g, Attr[] a );  
drawPolyline ( Window w, Geom[] g, Attr[] a );  
drawRectangle ( Window w, Geom[] g, Attr[] a );  
drawArc ( Window w, Geom[] g, Attr[] a );  
drawText ( Window w, Geom[] g, Attr[] a );  
fillPolyline ( Window w, Geom[] g, Attr[] a );  
fillRectangle ( Window w, Geom[] g, Attr[] a );  
fillArc ( Window w, Geom[] g, Attr[] a );
```

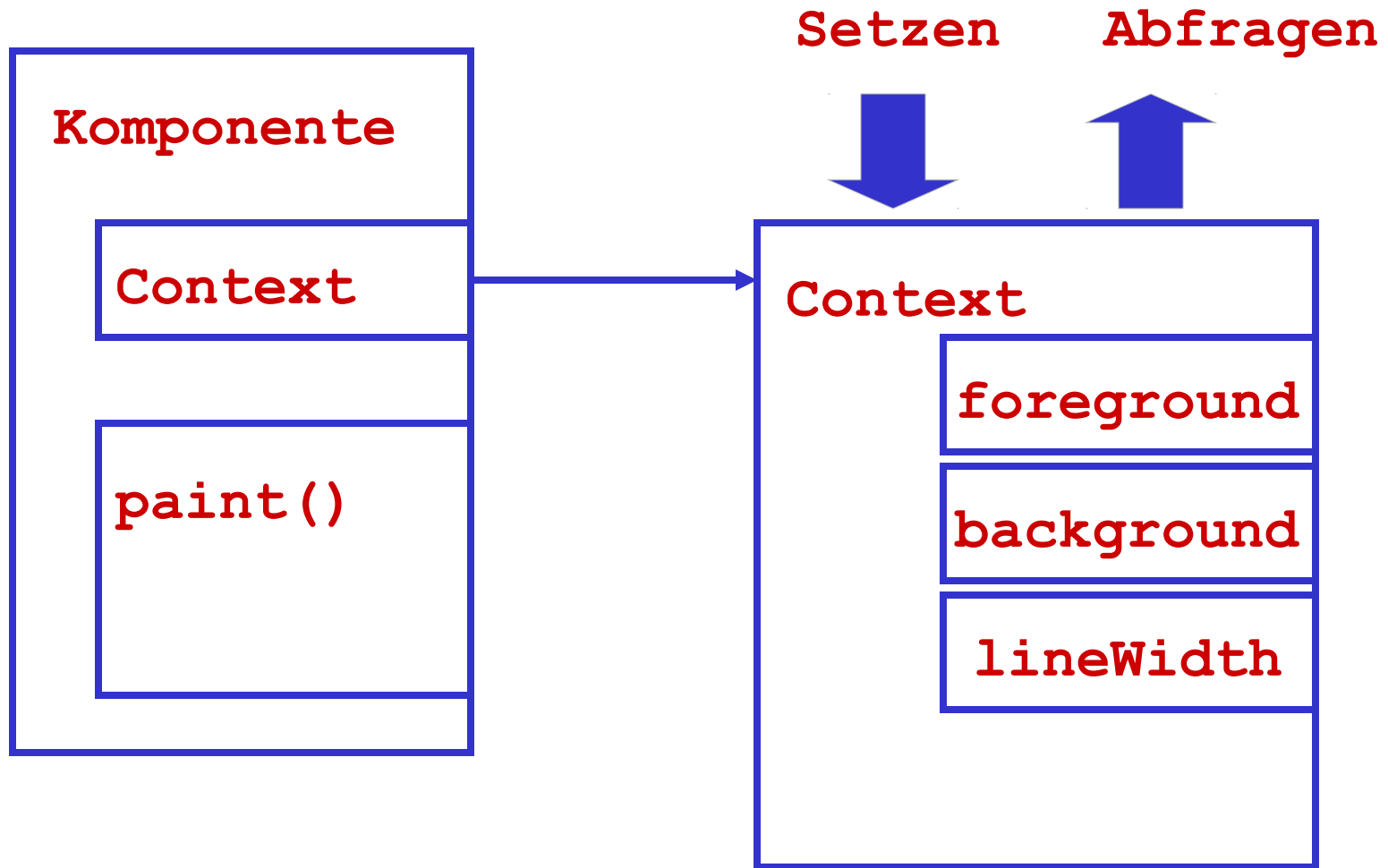
3.3 Der Graphik-Kontext

Da die Liste der Parameter immer sehr lang ist und es oft vorkommt, daß man viele draw-Operationen mit den gleichen Parameter macht, geht man zu dem Konzept des Graphik Kontext über. Dieses Konzept existiert in allen Graphik-APIs.

Grundzüge

- jedes Window besitzt einen **Graphik-Kontext**, der den aktuellen Zustand der graphischen Einstellungen während des gesamten Programms speichert, also Foreground, Background, Line Width, Line Style, ...
- jede Graphik-Operationen wird mit den Parametern dieses Graphik-Kontextes durchgeführt
- will man diese Graphik-Attribute ändern, so ändert man den Zustand des Kontext-Objektes

3.3 Der Graphik-Kontext



3.3 Der Graphik-Kontext

Ausgabe-Operationen (Pseudocode)

<u>Operation</u>	<u>Window</u>	<u>Kontext</u>	<u>Geometrie</u>
drawPoint	(Window w,	Context c,	Point p);
drawLine	(Window w,	Context c,	Point p1, Point p2);
drawPolyline	(Window w,	Context c,	Point[] points);
drawRectangle	(Window w,	Context c,	Point p1, Point p2);
drawArc	(Window w,	Context c,	Point p, int r);
u.s.w.			

3.4 Graphik-Ausgabe in AWT

In AWT sind die beiden Konzepte

- Graphik-Ausgabe-Operationen und
- Graphik-Attribute

in der Graphik-Kontext-Klasse `java.awt.Graphics` gebündelt


Die `paint()`-Methode hat als einzigen Parameter diesen Graphik-Kontext.

```
void paint ( Graphics g )  
    {  
        Kontext modifizieren  
        Graphik-Operation ausführen  
    }
```

3.4 Graphik-Ausgabe in AWT

Ausgabe mehrerer Primitiven (Pseudocode)

```
void paint ( Graphics g ) {  
    Primitive 1:  
        Kontext modifizieren  
        Graphik-Operation-1 ausführen unten  
    Primitive 2:  
        Kontext modifizieren  
        Graphik-Operation-2 ausführen  
    Primitiven 3-12:  
        Kontext modifizieren  
        Graphik-Operation-3 ausführen  
        ...  
        Graphik-Operation-12 ausführen  
}
```



oben

3.4.1 Class java.awt.Graphics

Beschreibung:

The Graphics class is the abstract base class for all graphics contexts that allow an application to draw onto components that are realized on various devices, as well as onto off screen images. A Graphics object encapsulates state information needed for the basic rendering operations that Java supports. This state information includes the following properties:

- The Component object on which to draw.
- A translation origin for rendering and clipping.
- The current clip.
- The current color.
- The current font. (...)

3.4.1 Class java.awt.Graphics

Set-Methoden

```
void setColor ( java.awt.Color c );  
void setFont ( java.awt.Font font );  
void setXORMode ( java.awt.Color c1 );
```

Get-Methoden

```
java.awt.Color setColor();  
java.awt.Font getFont();
```


3.4.1 Class java.awt.Graphics

Draw-Methoden

```
void drawLine ( int x1, int y1, int x2, int y2 );
void drawPolyline ( int[] xPoints, int[] yPoints,
                    int nPoints );
void drawPolygon ( int[] xPoints, int[] yPoints,
                  int nPoints );
void drawRect (int x1, int y1, int width, int height );
void drawRoundRect ( int x1, int y1,
                    int width, int height,
                    int arcWidth, int arcHeight );
void drawArc ( int x, int y, int width, int height,
              int startAngle, int arcAngle );
void drawOval ( int x, int y, int width, int height );
void drawString ( String str, int x, int y );
```

3.4.1 Class java.awt.Graphics

Fill-Methoden

```
void fillPolygon ( int[] xPoints, int[] yPoints,  
                  int nPoints );  
void fillRect (int x1, int y1, int width, int height );  
void fillRoundRect ( int x1, int y1,  
                    int width, int height,  
                    int arcWidth, int arcHeight );  
void fillArc ( int x, int y, int width, int height,  
              int startAngle, int arcAngle );  
void fillOval ( int x, int y, int width, int height );
```

3.4.1 Class java.awt.Graphics

Clear-Methode

```
void clearRect ( int x, int y, int width, int height );
```

„Clear“ bedeutet füllen mit der Hintergrund-Farbe

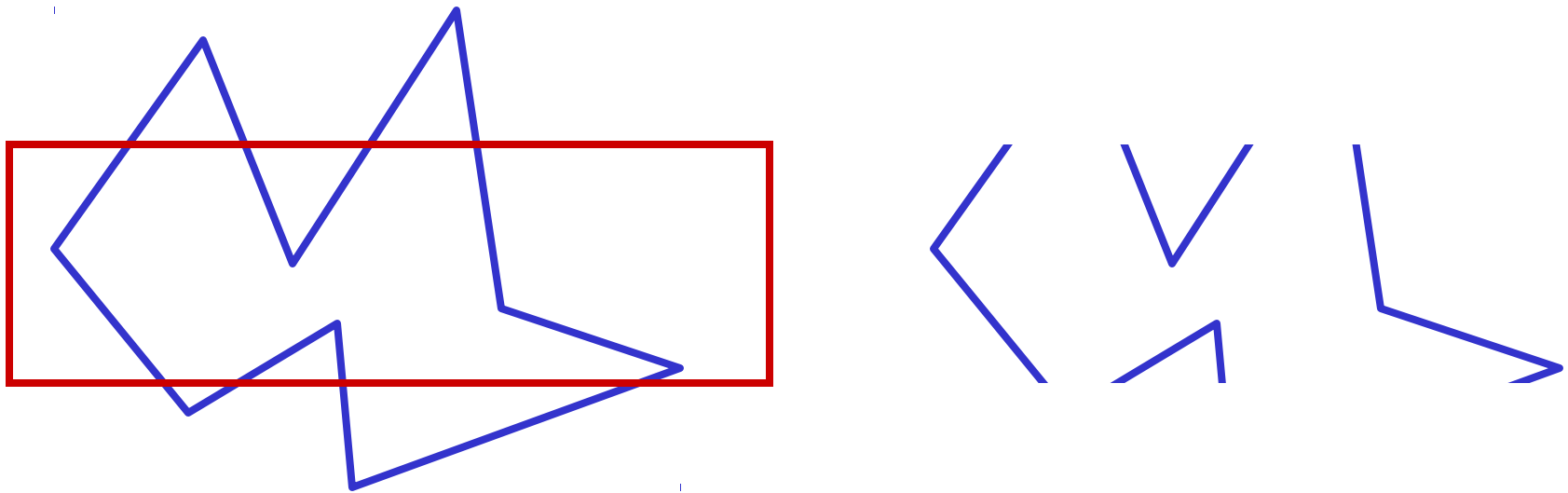
Image-Methoden

```
boolean drawImage ( java.awt.Image img, int x, int y,  
                    ImageObserver observer );
```

3.4.1 Class java.awt.Graphics

Clipping

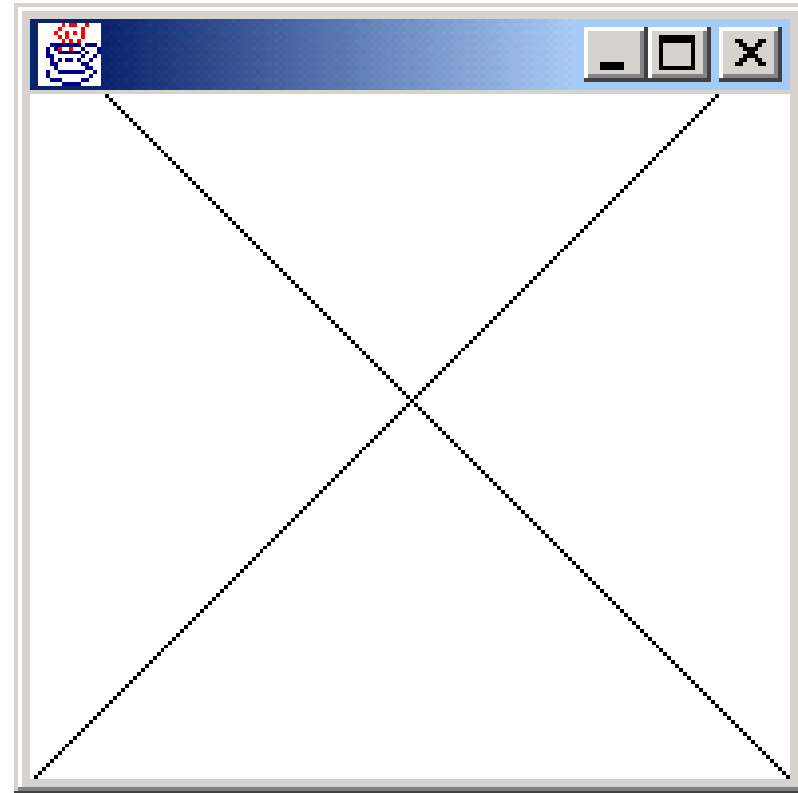
- Zusätzlich kann man die Ausgabe noch durch ein **Clip Rectangle** (oder einen beliebige **Clip Shape**) begrenzen



3.4.2 Beispiele

Beispiel: PaintDemo2

```
public class PaintDemo2 extends Frame {  
    public void paint ( Graphics g ) {  
        int width = getWidth();  
        int height = getHeight();  
        g.clearRect(0,0,width,height);  
        g.drawLine(0,0,width,height);  
        g.drawLine(0,height,width,0);  
    }  
  
    public PaintDemo2() {  
        setSize(200,200);  
        setLocation(100,100);  
    }  
}
```



3.4.2 Beispiele

Beispiel: `Frame2`

Demo: `Frame2Demo1`

Paint-Methode der Klasse `Frame2`

```
public void paint (Graphics g) {  
    int width = getWidth();  
    int height = getHeight();  
    boolean blue=true;  
  
    for (int i=0;i<width;i+=20) {  
        if (blue) g.setColor(Color.blue);  
        else g.setColor(Color.yellow);  
        blue = !blue;  
        g.fillRect(i,0,20,height);  
    }  
}
```

