

Ereignisse

Graphikausgabe in AWT und Swing



Ereignisse

Kommunikation mit anderen Beans

- Zustandsänderung wird allen Interessierten mitgeteilt
- Lose gekoppelte Kommunikation

Quelle

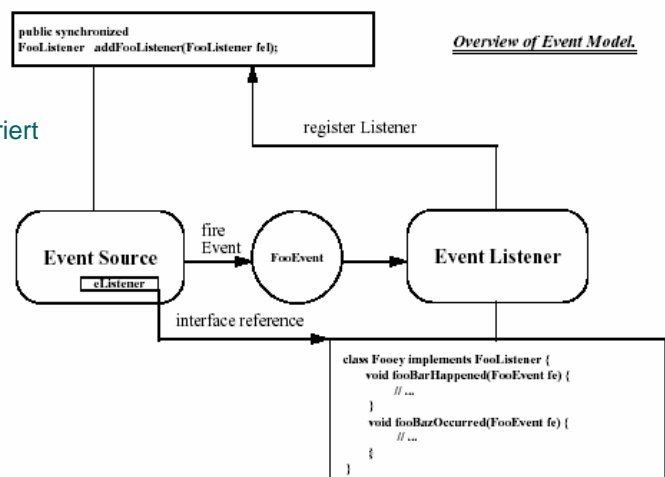
- Auslöser eines Ereignisses
- Benachrichtigt Interessierte

Interessierter (Listener)

- Reagiert auf Ereignis
- Wird bei Quelle registriert und deregistriert

Ereignisobjekt (EventObject)

- Informationen über Ereignis



Design Pattern für Ereignisse

Listener

- Interface welche Ereignismethoden implementiert
- *kann* `java.util.EventListener` erweitern

```
public interface TimerListener extends java.util.EventListener {  
    public void timerAction(TimerEvent e);  
}
```

Quelle

- implementiert Methoden für das Anfügen und Entfernen von Listener
`public void add<ListenerType>(<ListenerType> listener)`
`public void remove<ListenerType>(<ListenerType> listener);`

```
public class TimerBean {  
    public void addTimerListener(TimerListener l) { ... }  
    public void removeTimerListener(TimerListener l) { ... }  
  
    private void fireTimerEvent(TimerEvent e) { ... }  
    ...  
}
```



Design Pattern für Ereignisse (Fortsetzung)

Ereignisobjekt

- *erweitert* `java.util.EventObject`
- *kommuniziert* Informationen über Ereignis

```
public class TimerEvent extends java.util.EventObject {  
    private long time;  
  
    public TimerEvent(Object source) {  
        super(source);  
        time = System.currentTimeMillis();  
    }  
  
    public long getTime() {  
        return time;  
    }  
}
```



Ereignisse

Graphikausgabe in AWT und Swing



Grundlagen der Graphikausgabe

Components zeichnen sich mit Methode

```
public void paint(Graphics g)
```

wobei der Graphics-Parameter die Zeichenfläche für die Component darstellt

Anwenderprogramm darf paint nicht aufrufen,
sondern Aufruf der paint-Methoden erfolgt im *AWT-Loop*

Anwenderprogramm kann durch Aufruf von

```
public void repaint()  
public void repaint(int x, int y, int width, int height)
```

das Neuzeichnen der Komponente (eines Bereichs der Komponente) „fordern“



AWT-Loop

Anwenderprogramm und AWT-Loop laufen in unterschiedlichen Threads

Anwenderprogramm

- führt Änderungen in den Daten durch, die Änderungen in der GUI bewirken sollen
- ruft `repaint` der betroffenen Component auf

AWT-Loop

- durch `repaint` weiss AWT, dass Component neu gezeichnet werden muss
- AWT-Loop arbeitet die Zeichenanforderungen ab
- bewirkt Neuzeichnen der Components



AWT-Loop

Neuzeichnen:

- AWT-Loop ruft Methode `update` bei der Component auf
- `update` ruft `paint` auf

```
public void update(Graphics g) {  
    paint(g);  
}
```

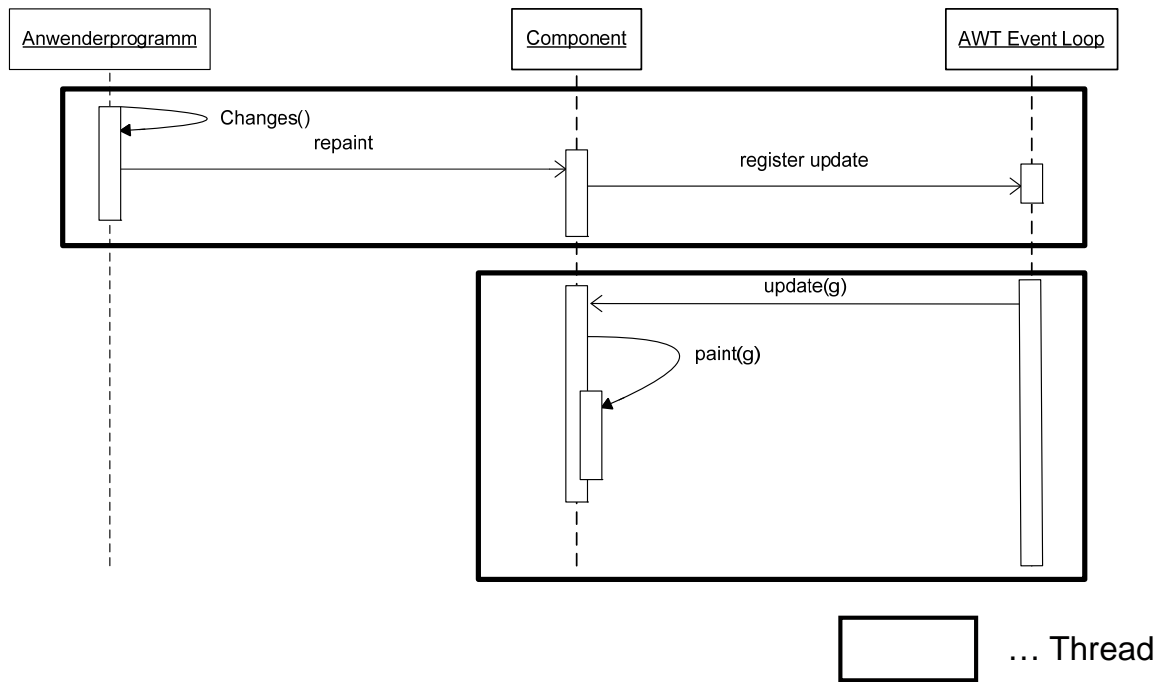
- `paint` führt Zeichnen der Component durch

```
public void paint(Graphics g) {  
    ...  
}
```

- `update` wird verwendet, um in Painting grundsätzlich einzugreifen (z.B.: DoubleBuffering)
- `paint` wird überschrieben, um bei Components spezielle Darstellungen zu realisieren



AWT-Loop



Beachte: update und paint laufen im AWT-Thread
→ lang laufende paint-Methoden blockieren AWT-Thread



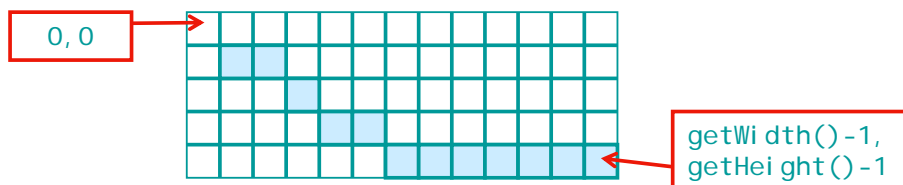
Graphikkontext Graphics

In Methode paint wird als Parameter ein Graphics-Objekt übergeben

```
public void paint(Graphics g)
```

Stellt virtuelle Zeichenfläche für die Component dar (genannt *Graphikkontext*)

- hat Größe der Component mit 0/0-Koordinate im linken, oberen Eck der Component



- ermöglicht Zeichnen von Linien, Figuren, Text und Bildern, ...
- hat eine Reihe von Zuständen, die die Grafikausgabe beeinflussen
 - Color: gegenwärtige Zeichenfarbe
 - Font: gegenwärtige Schriftart für Text
 - Clip: Zeichenoperationen wirken sich nur innerhalb des Clip aus
 - XORMode: logische Pixeloperation (XOR oder Paint)



Zeichenoperationen der Klasse Graphics

```
clearRect(int x, int y, int w, int h)
copyArea(x, y, w, h, dx, dy)
draw3DRect(x, y, w, h, boolean raised)
drawArc(x, y, w, h, int startAngle, int arcAngle)
drawLine(x1, y1, x2, y2)
drawOval(x, y, w, h)
drawPolygon(x[], y[], int nPoints)
drawPolyLine(x[], y[], int nPoints)
drawRect(x, y, w, h)
drawRoundRect(x, y, w, h, int arcWidth, int arcHeight)
drawString(String str, x, y)
fill3DRect(x, y, w, h, boolean raised)
fillArc(...)
fillOval(...)
fillPolygon(...)
fillRoundRect(...)
drawImage(Image img, x, y, ImageObserver obs)

....
```



Weitere Methoden der Klasse Graphics

```
translate(int x, int y) // Verschieben des Koordinatenursprungs
setPaintMode() // normalen Zeichenmodus setzen (default)
setXORMode(Color c1) // XOR-Modus momentane Farbe xor c1
setFont(Font f) // Schriftart setzen
setColor(Color c) // Zeichenfarbe setzen
setClip(x,y,w,h) // setzt neues Clip (Pixel nur innerhalb betroffen)
FontMetrics getFontMetrics() // holt Metrik-informationen zur Font
// zB. wieviele Pixel breit/hoch ist "1. Object"
Font getFont() // aktuelle Schriftart
Color getColor()
Rectangle getClipBounds() // Rechteckigen Clip-Bereich holen; Auch wenn
// eine Linie über diesen hinausgeht, wird nur
// innerhalb gezeichnet.
....
```



Grafikausgabe

Vorgehen:

- Klasse von Canvas ableiten
- paint-Methode überschreiben
- Canvas-Objekt in Frame einfügen

```
import java.awt.*;
import java.awt.event.*;

public class PaintTestCanvas extends Canvas {
    public void paint(Graphics g) {
        super.paint(g); // Löschen
        for (int x = 0; x <= getWidth()-1; x++) {
            g.setColor(new Color(0, (int)(255.0/getWidth()*x),
                (int)(255.0/getWidth()*x)));
            g.drawLine(x, 0, getWidth()-x, getHeight()-x);
        }
    }

    public static void main(String[] args) {
        Frame frame = new Frame("Paint test");
        frame.add(new PaintTestCanvas());
        frame.setSize(new Dimension(200, 200));
        frame.setVisible(true);
    }
}
```



Painting in Swing

Painting bei Swing in AWT-Painting eingebettet: Aufruf von

```
public void update(Graphics g)
```

und

```
public void paint(Graphics g)
```

Aber:

paint bei JComponent folgend implementiert

```
public void paint(Graphics g) {
    ...
    paintComponent(co);
    paintBorder(co);
    ...
    paintChildren(co);
    ...
}
```

Zeichnen der Komponente
Zeichnen eines Rands

Zeichnen der Kinder

Üblicherweise wird nur paintComponent(Graphics g) überschrieben

```
public class MyJComponent
    protected void paintComponent (Graphics g) {
        // my custom painting ...
    }
}
```



Painting in Swing

Graphi cs2D

- ist speziellens Graphi cs-Klasse
- mächtige Graphikfunktionen

Plus viele weitere Features:

- Double-Buffering bereits implementiert und bei jeder JComponent einschaltbar

```
publ i c voi d setDoubl eBuffered(bool ean db)
```

- Verschiedene Borders

```
voi d setBorder(Border b)
```

- TooltipText

```
voi d setTool ti pText(Stri ng text)
```

- Transparenter Hintergrund

```
voi d setOpaque(bool ean opaque)
```

- ...



Beispiel Painting in Swing

```
cl ass GridJComponent extends JComponent {  
  
    publ ic GridJComponent() {  
        setLayout(new FlowLayout());  
        JButton b1 = new JButton("Button 1");  
        JButton b2 = new JButton("Button 2");  
        setBorder(BorderFactory.createLoweredBevel Border());  
        add(b1);  
        add(b2);  
    }  
  
    protecte d voi d pai ntComponent(Graphi cs g) {  
        i nt wi dth = getSi ze(). wi dth;  
        i nt hei ght = getSi ze(). hei ght;  
        g. setCol or(Col or. gray);  
        for (i nt i = 0; i < wi dth; i += 10) {  
            g. drawLi ne(i, 0, i, hei ght);  
        }  
        for (i nt i = 0; i < hei ght; i += 10) {  
            g. drawLi ne(0, i, wi dth, i);  
        }  
    }  
}
```

