

## Java Foundation Classes (JFC)

- Abstract Window Toolkit (awt)
- swing-Komponenten
- Java-2D-API

**Autor:** Dipl.-Math.  
E. Engelhardt

**Stand:** 02. Juni 2009



# Inhalt

## 1 Einleitung

- 1.1 Architektur des jdk 1.5
- 1.2 Standardpakete des jdk 1.5
- 1.3 Bestandteile des JFC
- 1.4 Weitere Pakete des JFC

## 2 Das awt

- 2.1 Packages des awt
- 2.2 Plattformunabhängigkeit durch Peers
- 2.3 Hauptklassen des awt
- 2.4 Komponenten des awt
  - 2.4.1 Hierarchie der awt-Komponenten
  - 2.4.2 Grundfunktionen von Komponenten
- 2.5 Ereignisbehandlung
  - 2.5.1 Ereignisbehandlung ab jdk 1.1
  - 2.5.2 Events und Listener
  - 2.5.3 Hierarchie der Ereignisklassen
  - 2.5.4 Tabelle
- 2.6 Layoutmanager
  - 2.6.1 Sinn von Layoutmanagern

- 2.6.2 Border-Layout
- 2.6.3 Flow-Layout
- 2.6.4 Grid-Layout
- 2.6.5 GridBag-Layout
- 2.6.6 Card-Layout
- 2.7 Nachteile des awt

## 3 swing-Komponenten

- 3.1 Heavyweight und Lightweight
- 3.2 zusätzliche swing-Komponenten
- 3.3 Plugable Look & Feel
- 3.4 Das klassische MVC-Konzept
- 3.5 Das MVC-Konzept von swing

## 4 Java-2D-API

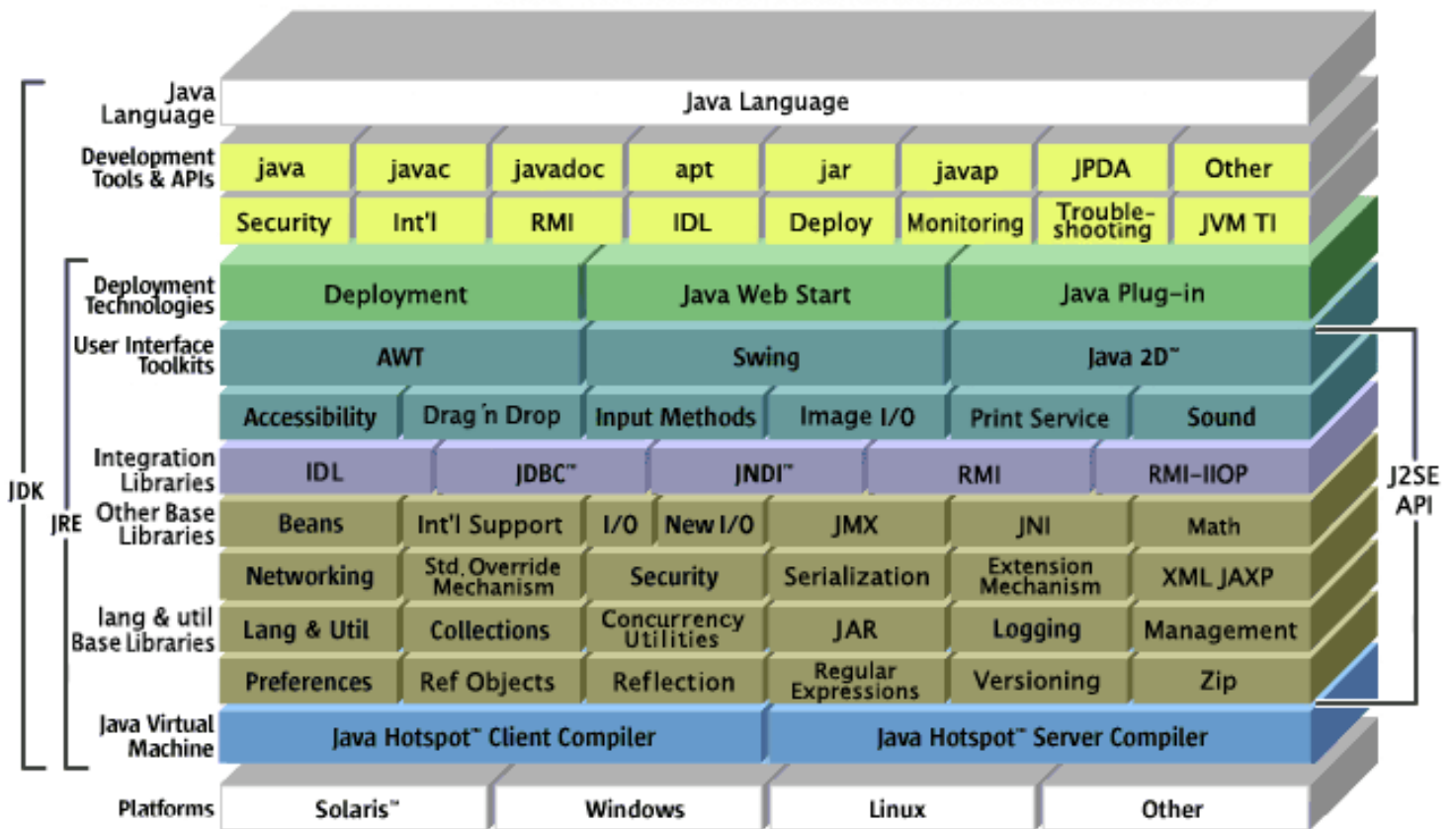
## 5 Accessibility-API

# Abkürzungen

- ◆ API Application Programming Interface
- ◆ AWT Abstract Window Toolkit
- ◆ GUI Gaphical User Interface
- ◆ IDE Integrated Development Environment
- ◆ IO Input Output (Ein- Ausgabeströme)
- ◆ JDK Java Development Kit
- ◆ JFC Java Foundation Classes
- ◆ JNI Java Native Interface
- ◆ JRE Java Runtime Environment
- ◆ JVM Java Virtual Machine
- ◆ MVC Model-View-Control-Konzept
- ◆ RMI Remote Method Interface
- ◆ SDK Software Development Kit
- ◆ SQL Structured Query Language
- ◆ XML Extended Markup Language

# 1.1 Architektur des jdk 1.5 „J2SE 5.0“

Java™ 2 Platform Standard Edition 5.0



# 1.2 Standardpakete des jdk 1.5

- ◆ java.applet                    Applets
- ◆ java.awt                        Abstract Windowing Toolkit
- ◆ java.beans                     Java Bean
- ◆ java.io                         Datei-I/O
- ◆ java.lang                      elementare Sprachunterstützung
- ◆ java.lang.ref                 Referenz-Objekte
- ◆ java.lang.reflect             Reflection-API
- ◆ java.math                      mathematische Funktionen
- ◆ java.net                        Netzwerkunterstützung
- ◆ java.nio                        new I/O (seit Java 5.0)
- ◆ java.rmi                        verteilte Anwendungen (RMI)
- ◆ java.security                 Security-Dienste
- ◆ java.sql                        Datenbankzugriff (JDBC)
- ◆ java.text                      Internationalisierung
- ◆ java.util                       Utilities, Collections

## 1.3 Bestandteile der JFC

- ◆ Entwicklung grafischer Benutzerschnittstellen
- ◆ JFC bestehen aus drei User Interface Toolkits:
  - Abstract Window Toolkit (awt)
  - Swing-Komponenten und
  - Java-2D-API
- ◆ Mit den Packages u.a.
  - Accessibility, Drag `n Drop, Input Methods, Image I/O, Print Service und Sound
  - Einige APIs verfügen über mehrere Packages
  - Das Java-2D-API enthält Klassen in „java.awt“ und in „java.awt.image“

## 1.4 Weitere Packages des JFC

- |                       |                                     |
|-----------------------|-------------------------------------|
| ◆ javax.accessibility | E/A-Geräte (Braille-Zeile)          |
| ◆ javax.crypto        | Kryptographische Erweiterungen      |
| ◆ javax.imageio       | Lesen und Schreiben von Bilddateien |
| ◆ javax.naming        | Zugriff auf Namens-Services         |
| ◆ javax.print         | Unterstützung zum Drucken           |
| ◆ javax.security.auth | Authentifizierung und Autorisierung |
| ◆ javax.sound         | Sound-API                           |
| ◆ javax.swing         | swing-Toolkit                       |
| ◆ javax.xml           | XML-Dateien                         |

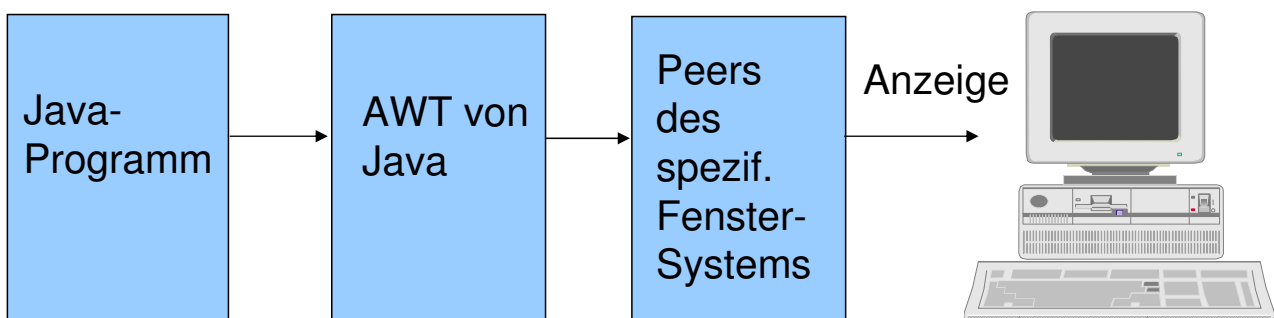
## 2 Das awt

### 2.1 Packages des awt

- ◆ java.awt Grundfunktionen
- ◆ java.awt.accessibility Unterstützende Verfahren
- ◆ java.awt.color Farben und Farbdefinitionen
- ◆ java.awt.datatransfer Zwischenablage
- ◆ java.awt.dnd Drag & Drop
- ◆ java.awt.event Ereignisklassen und Listener
- ◆ java.awt.font Font-Package von 2D-API
- ◆ java.awt.geom Zeichen-Package von 2D-API
- ◆ java.awt.im Eingabemethoden
- ◆ java.awt.image Bildbearbeitung
- ◆ java.awt.peer Peer-Schnittstellen
- ◆ java.awt.print Druck-Unterstützung von 2D-API
- ◆ java.awt.swing swing-Komponenten
- ◆ java.awt.test einzelnes Applet, das awt-Funktionen testet

### 2.2 Plattformunabhängigkeit durch Peers

- ◆ awt-Komponenten delegieren Funktionen an die Peer des entsprechenden Systems
- ◆ Peers sind die spezifischen Komponenten der jeweiligen grafischen Oberfläche



# Vorteile und Nachteile von Peers

## ◆ Vorteile:

- Schnelle Entwicklung des awt

## ◆ Nachteile:

- Komponenten haben kein einheitliches Aussehen unter verschiedenen Plattformen
- awt-Heavyweight-Komponenten besitzen auch eventuell ungünstige Eigenschaften der Peers (unterschiedliche Zeilen-Begrenzer, Sichtbarkeit)
- eingeschränkter Umfang an grafischen Elementen

## 2.3 Hauptklassen des awt

### ◆ Component

- abstrakte Grundklasse für Menüs, Schaltflächen, Beschriftungen, Listen, usw.

### ◆ Container

- abstrakte Grundklasse, die Klasse „Component“ erweitert
- abgeleitete Klassen sind u.a. Panel, Applet, Window, Dialog und Frame

### ◆ LayoutManager

- Positionierung und Größe von Komponenten innerhalb eines Behälters

### ◆ Graphics

- abstrakte Klasse für grafische Operationen

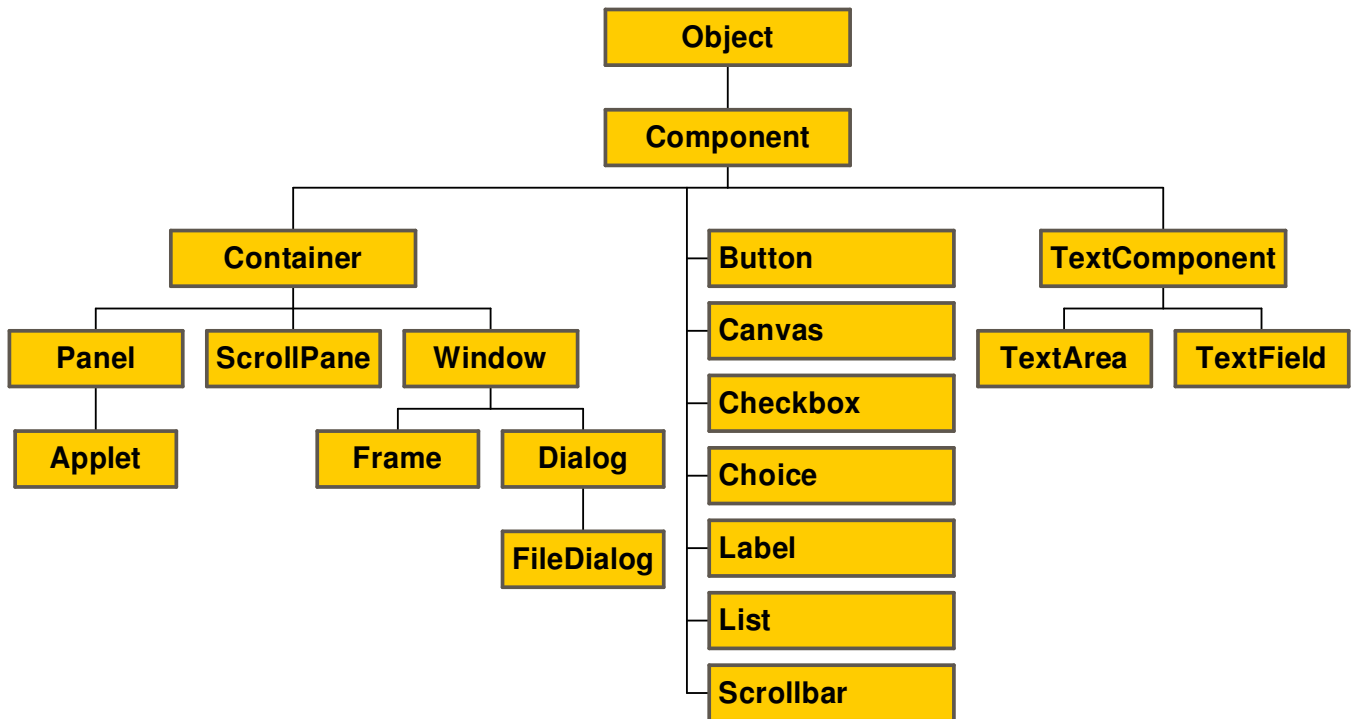
## 2.4 Komponenten des awt

<b>Name</b>	<b>übergeordnete Klasse</b>	<b>Beschreibung</b>
◆ Applet	Panel	Basisklasse für Applets
◆ Button	Component	Textschaltfläche
◆ Canvas	Component	Leinwand für Grafiken
◆ Checkbox	Component	markierbare boolsche Komponente
◆ Choice	Component	Pop-down-Menü mit Texteinträgen
◆ Dialog	Window	modal oder nichtmodal
◆ FileDialog	Dialog	Dateiauswahl
◆ Frame	Window	Fenster mit Titelleiste und Menüs
◆ Label	Component	Anzeige einer Zeichenkette
◆ List	Component	Bildlaufliste mit Texteinträgen
◆ Panel	Container	Basisbehälter für Komponenten

## Komponenten des awt

<b>Name</b>	<b>übergeordnete Klasse</b>	<b>Beschreibung</b>
◆ Scrollbar	Component	veränderliche Komponente zum Bildlauf von Elementen
◆ ScrollPane	Container	Behälter mit Rollbalken
◆ TextArea	TextComponent	mehrzeiliges, scrollbares Textfeld
◆ TextComponent	Component	Superklasse von „TextArea“ und „TextField“
◆ TextField	TextComponent	einzeilige Komponente zur Texteingabe
◆ Window	Container	randloses Fenster ohne Titel

## 2.4.1 Hierarchie der awt-Komponenten



## 2.4.2 Grundfunktionen von Komponenten

- ◆ Ein Graphics-Objekt
- ◆ Position und Größe
- ◆ Spezifisches Peer
- ◆ Übergeordneter Behälter
- ◆ Schriften und Schriftgrößen
- ◆ Vorder- und Hintergrundfarben
- ◆ Lokale Bezeichner
- ◆ minimale, maximale und bevorzugte Größen

## 2.5 Ereignisbehandlung

- ◆ Ereignisse sind Vorgänge, die sich bei Veränderungen von Objekten ereignen (Klick auf einen Knopf, Mausklick, Mausbewegung, Tastendruck, ...)
- ◆ Die Objekte sind die Ereignisquellen (Event-Source), sie können Meldungen verschicken (Events)
- ◆ In Java 1.0 wurden Ereignismeldungen an alle Elemente, in denen die Ereignisquelle eingebettet ist, geschickt und in „**handleEvent**“ behandelt.
- ◆ Diese Methode ist ineffizient und unflexibel und sollte nicht mehr verwendet werden

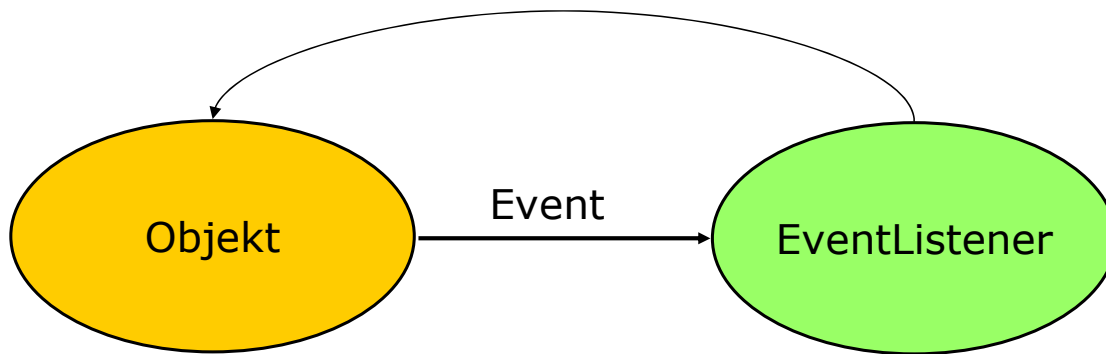
### 2.5.1 Ereignisbehandlung ab jdk 1.1

- ◆ Ab Java 1.1 müssen sich alle Elemente bei den Ereignisquellen als Listener (Lauscher) registrieren, wenn sie die Ereignisse verarbeiten wollen
- ◆ Die Meldungen werden dann nur an die registrierten Listener geschickt
- ◆ Die Ereignisquellen verwalten eine Liste von Listnern
- ◆ Diese Form des Eventhandling wird als Delegation bezeichnet
- ◆ Folgende Schritte sind durchzuführen:
  - Anlegen der Listener durch entsprechende Klassen bzw. Schnittstellen
  - Registrieren der Listener („xxx.addXxxListener“)
  - Implementieren der entsprechenden Methoden („actionPerformed“, „mousePressed“, ...)

## 2.5.2 Events und Listener

- ◆ LowLevel-Events z.B. Mausaktivitäten
- ◆ HighLevel-Events z.B. Menüeinträge, Listboxen

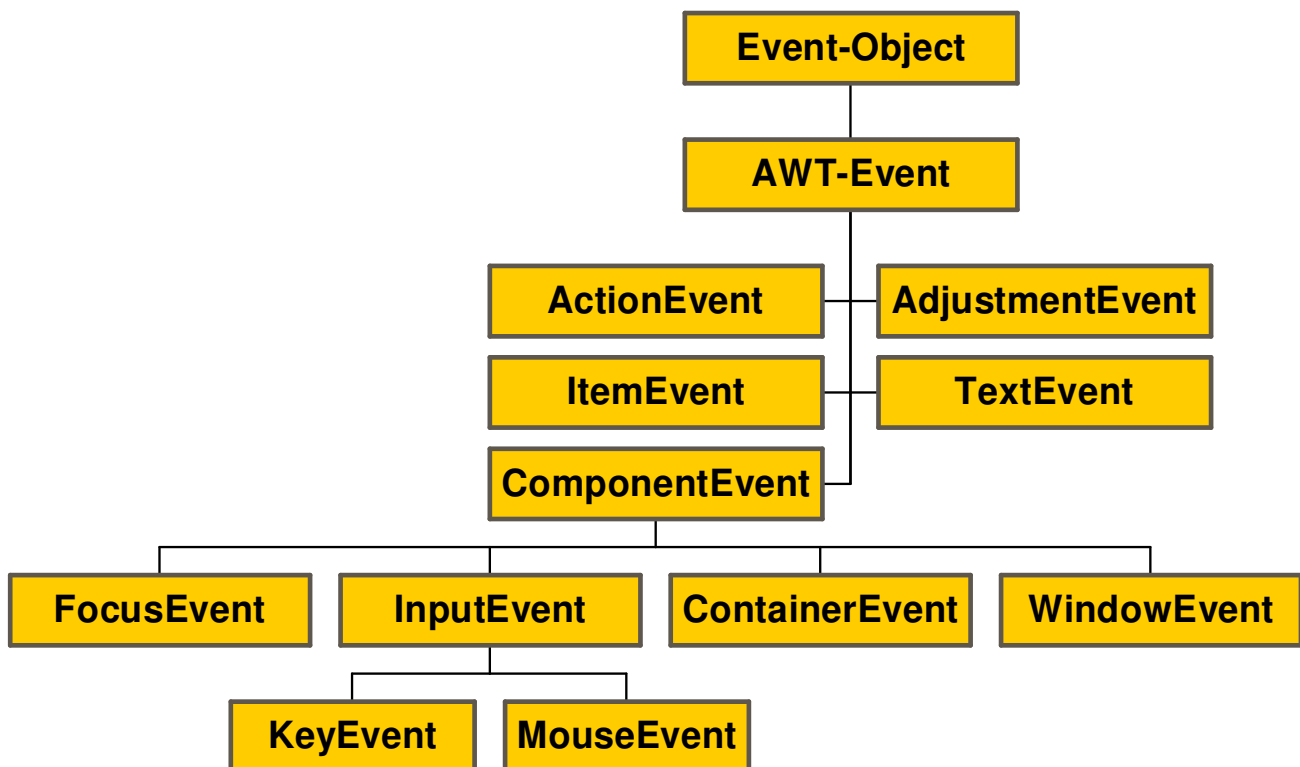
Interesse für Event (Aufruf einer Registrierungsmethode)  
addXxxListener und removeXxxListener



Event-Source

Methode für Events  
abgeleitet aus Schnittstellen  
java.util.EventListener

## 2.5.3 Hierarchie der Ereignisklassen



## 2.5.4 Tabelle

Compo- nente / Listener	Button	Canvas,Label Component	Checkbox	Checkbox- MenuItem	Choice	Container, Panel	Dialog, Frame	List	MenuItem	Scrollbar	ScrollPane	TextArea, -Component	TextField	Window
Action	✓							✓	✓				✓	
Adjustment										✓				
Component	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Container						✓	✓				✓			✓
Focus	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Item			✓	✓	✓									
Key	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Mouse	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Text												✓	✓	
Window							✓							✓

Folie 23 von 38

JFC

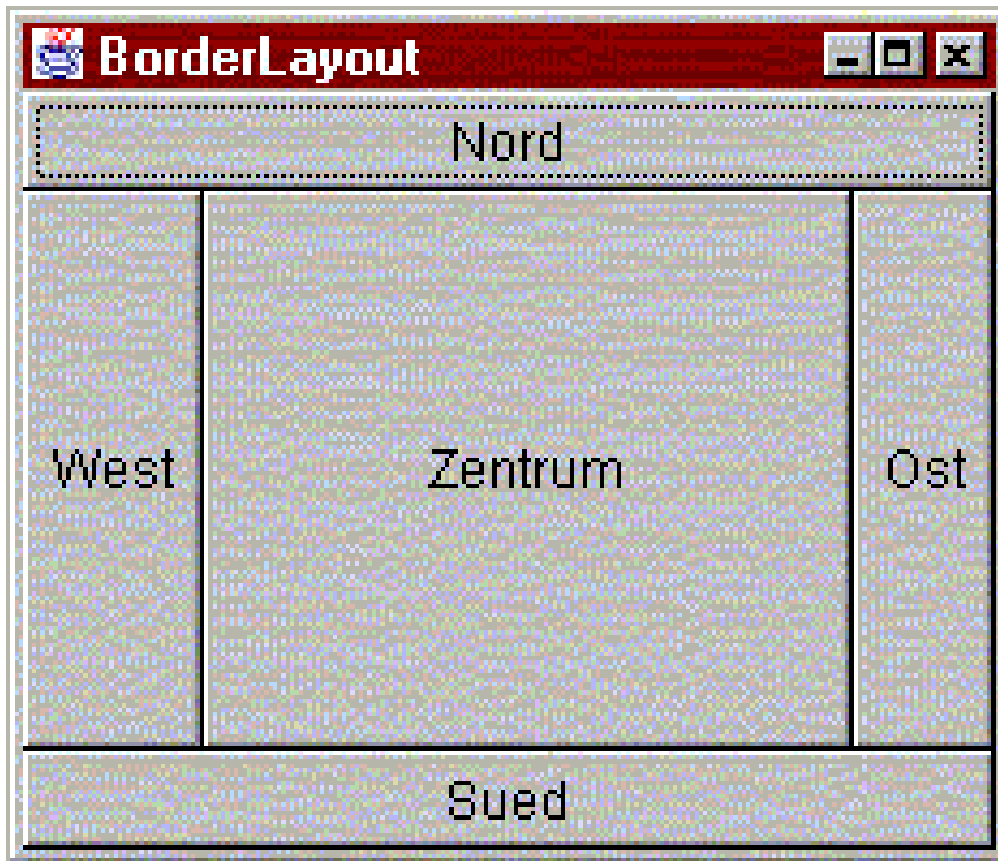
## 2.6 Layoutmanager 2.6.1 Sinn

- ◆ Automatische Anordnung von Komponenten entsprechend des gewählten Layout-Managers
- ◆ Anordnung auch bei veränderter Größe der Darstellungsfläche
- ◆ einfaches Hinzufügen neuer Komponenten
  
- ◆ Es besteht die Möglichkeit, den Layoutmanager aus-  
zuschalten und die Komponenten manuell zu platzieren  
durch:
  - **„setLayout(null);“**
  - **...**
  - **„xxx.setLocation(PixelX, PixelY);“**
  - **„xxx.setSize(BreitePixel, HoehePixel);“** (xxx ist der Name der entsprechenden Komponente)

Folie 24 von 38

JFC

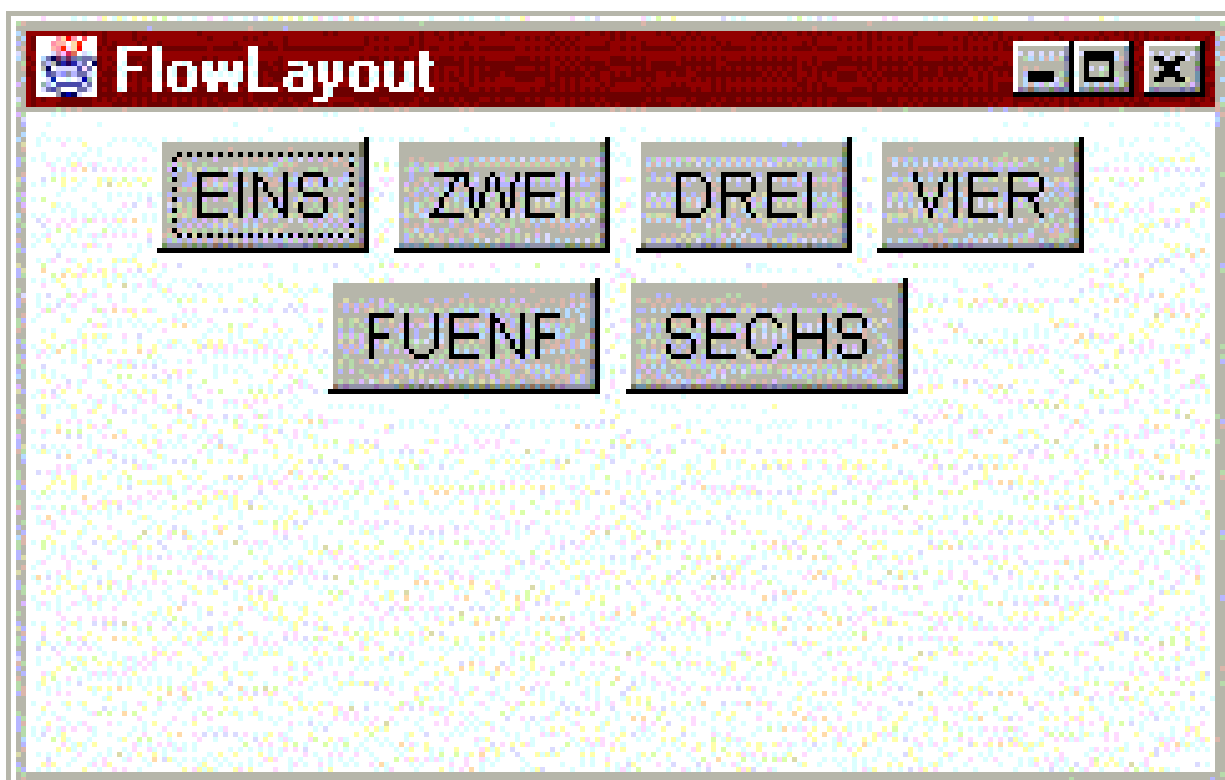
## 2.6.2 Border-Layout



Folie 25 von 38

JFC

## 2.6.3 Flow-Layout



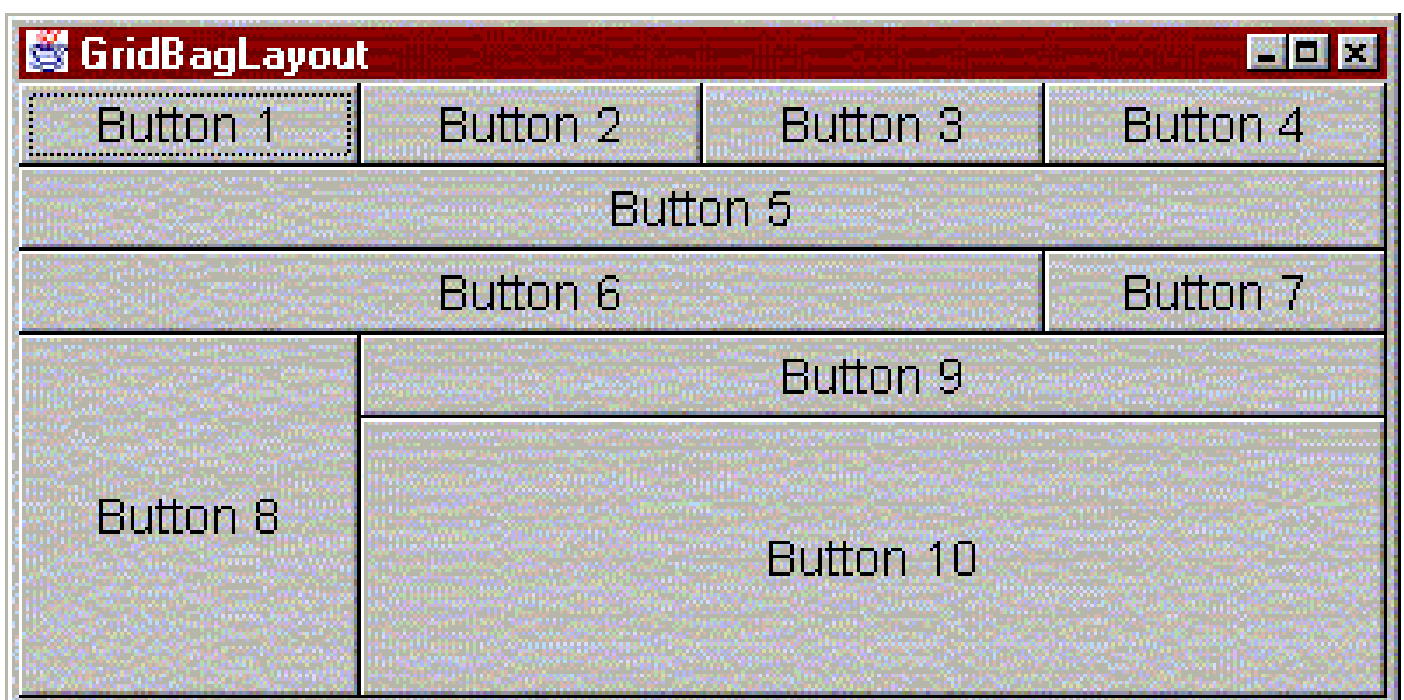
Folie 26 von 38

JFC

## 2.6.4 Grid-Layout



## 2.6.5 GridBag-Layout



## 2.6.6 Card-Layout



Folie 29 von 38

JFC

## 2.7 Nachteile des awt

- ◆ Nur einfache grafische Benutzeroberflächen möglich
- ◆ Größter gemeinsamer Nenner der Betriebssysteme
- ◆ Fehlen von Funktionen (Zwischenablage, Drucken, Pop-up-Menüs, Bildlaufleiste, ...)
- ◆ Peer-basierte Architektur, dadurch:
  - Hülle um komplexe, native Peers
  - Jeder Peer in einem eigenen nativen Fenster
  - Peer-Fehler und Kompatibilitätsprobleme (zwischen verschiedenen Betriebssystemen)
- ◆ nicht solide objektorientiert
- ◆ nicht auf dieser Basis erweiterbar
- ◆ Eigene Werkzeugsammlungen von Drittanbietern

Folie 30 von 38

JFC

## 3.1 Heavyweight und Lightweight

- ◆ Heavyweight-Komponenten sind mit einer Peer-Komponente verbunden und werden in einem eigenen *undurchsichtigen* Fenster dargestellt
- ◆ Lightweight-Komponenten haben kein natives Peer und werden im Fenster eines Heavyweight-Behälters dargestellt
  - sie können durchsichtigen Hintergrund haben
  - dadurch nicht notwendig rechteckig, obwohl mit rechteckiger Begrenzung
  - swing-Behälter sind Heavyweight-Komponenten (Frames, Windows, Applets, Dialoge)
- ◆ Für alle awt-Komponenten gibt es entsprechende swing-Komponenten

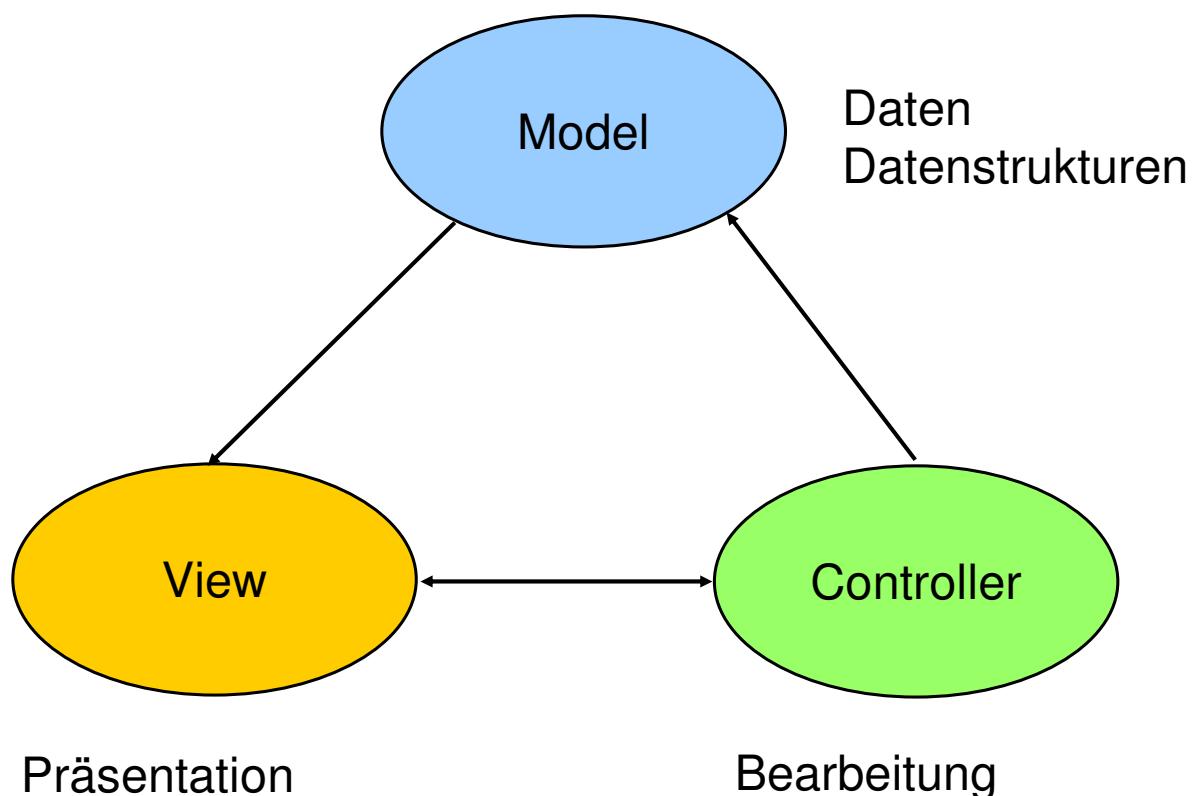
## 3.2 Zusätzliche swing-Komponenten

- |                              |                          |
|------------------------------|--------------------------|
| ◆ JDesktopPane, InternalPane | für MDI-Interface        |
| ◆ JEditorPane, JTextPane     | editierbar               |
| ◆ JOptionPane                | Popup-Fenster            |
| ◆ JPasswordField             | Passwortfeld             |
| ◆ JProgressBar               | Fortschrittsbalken       |
| ◆ JRadioButtonMenuItem       | in Menü integrierbar     |
| ◆ JSeparator                 | Menü-Trennbalken         |
| ◆ JSlider                    | erweiterter Scrollbar    |
| ◆ JSplitPane                 | Container                |
| ◆ JTable                     | zweidimensionale Tabelle |
| ◆ JToggleButton              | bleibt gedrückt          |
| ◆ JToolBar                   | verschiebbar für Fenster |
| ◆ JToolTip                   | Popup-Text               |
| ◆ JTree                      | Anzeige von Hierarchien  |

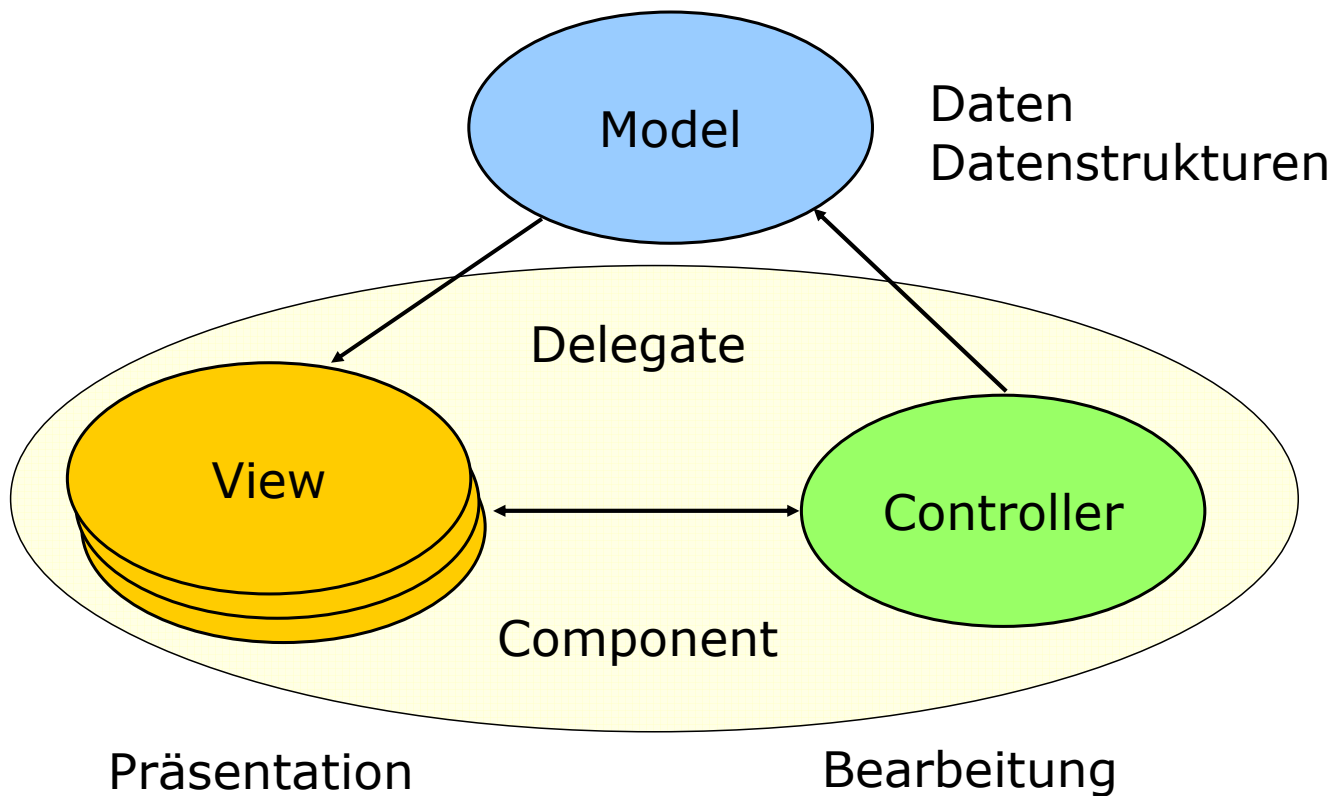
## 3.3 Plugabble Look & Feel

- ◆ **Metal**                      java-eigenes Look & Feel
  - Standard-Layout
  - auf allen Plattformen verfügbar
- ◆ **Motif**                      UNIX-Oberflächen
  - Motif ist Aufsatz auf X-Windows
  - X-Windows ist Grafikerweiterung von UNIX-Systemen
- ◆ **Windows**                  Microsoft-Windows
  - nur auf Windows-Plattformen verfügbar
- ◆ **Mac**                         Apple Macintosh
  - nur auf Macintosh verfügbar

## 3.4 Das klassische MVC-Konzept



## 3.5 Das MVC-Konzept von swing



## 4 Java-2D-API

- ◆ Starke Erweiterung der Möglichkeiten zur Bildbearbeitung und der Zeichenfunktionen
- ◆ Attribute des Grafikkontextes
  - Color Farben
  - Font Schriftarten
  - Stroke Linienbreite
  - Transform Drehen, Verschieben
  - Composite Darstellung auf Ausgabegerät
  - Clip Clipping

# 5 Accessibility-API

- ◆ Anpassungsfähigkeit der Benutzerschnittstelle
- ◆ z.B. Einbindung von E/A-Geräten für Behinderte
  - Schriften zur besseren Lesbarkeit
  - Knöpfe zur einfachen Aktivierung vergrößert darstellen
  - Braillezeile
  - Sprach-Ein- und Ausgabe

## Literaturangaben

- [1] Java 2 SDK v 1.2.2, *Grundlagen Programmierung*  
HERDT-Verlag für Bildungsmedien GmbH, Nackenheim
- [2] Guido Krüger: *Handbuch der Java-Programmierung*,  
Addison-Wesley Verlag,  
ISBN 3-8273-2201-4
- [3] Brit Schröter: *KompaktReferenz Java2*,  
DATA BECKER GmbH & Co. KG,  
ISBN 3-8158-1477-4
- [4] *Internet HTML-Seiten*,  
<http://java.sun.com>