Swing-Programmierung II

Referent: Felix Vogel

Seminar: Medientechnik

Dozent: Prof. Dr. Hussmann

LMU-München / SS03 /14.04.2003

Inhalt

- Benutzeroberfläche mit Swing
  - Einstellungen von Look and Feel
- Ereignisse
  - Einführung
  - Ereignistypen und Beispiele
- Konzepte und Grundlagen
  - Komponententypen
  - Menüerweiterungen

Benutzeroberfläche mit Swing (1)

- Eigenschaften von Swing
  - Mit **Look and Feel** lässt sich das Erscheinungsbild von Swing-Anwendungen an unterschiedliche Betriebssystemoberflächen anpassen
    - (z.B. Microsoft Windows, UNIX, Apple Macintosh)
  - Das Umschalten erfolgt zur Laufzeit des Programms
  - Einstellungen von verschiedenen Arten von Look and Feel erfolgt durch die
    - **Klasse: UIManager und Methode:** *setLookAndFeel(String s)*

- Vier Gruppen von Look-and-Feel
  - **Metal**-Look-and-Feel (für alle Betriebssysteme)
  - **Windows**-Look-and-Feel (für Windows-Betriebssystem)
  - **Motif**-Look-and-Feel (für UNIX-Systeme wie Solaris)
  - **Macintosh**-Look-and-Feel (Apple Macintosh Betriebssystem)

Benutzeroberfläche mit Swing (2)

- **Metal**-Look-and-Feel
  - **Metal** ist das standardmäßige Look-and-Feel für alle Swing-Anwendungen
  - Metal gilt als Standard-Look-and-Feel für alle eingesetzten Betriebssysteme
  - Keine explizite Deklaration im Programm
  - Wenn doch, dann innerhalb der Übersetzungsmaske
  - Den String von Standard-Look-and-Feel von Java, „**Metal**“, liefert die Methode:

    → *getClassPlatformLookAndFeelClassName(this.getContentPane( ))*
  - Bei direkter Initialisierung von Metal-Look-and-Feel lautet der String für die Methode *setLookAndFeel(String s):*

    →„javax.swing.plaf.metal.MetalLookAndFeel“

    (ersetze die Zeile 5 aus der Folie Nr.: 5 mit diesem String)

Benutzeroberfläche mit Swing (3)

- Ausschnitt aus Metal-Look-and-Feel Java-Programm

```
1.  public static void main(String[ ] args)
2.  {
3.          try {
4.          UIManager.setLookAndFeel(
5.  →       UIManager.getCrossPlatformLookAndFeelClassName( ));
6.              }
7.          catch(Exception e) { }
8.          // Ab hier wird das User-Interface erzeugt,
9.          // auch Menues und andere Komponenten
10.}
```

Benuzeroberfläche mit Swing (4)

- **Windows**-Look-and-Feel
  - Das Erscheinungsbild und die Bedienelemente werden der Windows-Benutzeroberfläche nachempfunden
  - Explizite Deklaration erfolgt durch den String: (ersetze Zeile 5 aus Folie 5)
    → „com.sun.java.swing.plaf.windows.WindowsLookAndFeel“

- **Motif**-Look-and-Feel
  - Motif-Look-and-Feel stellt die graphische Benutzeroberfläche von Sun Solaris-Maschinen dar
  - Explizite Deklaration erfolgt durch den String: (ersetze Zeile 5 aus Folie 5)
    → „com.sun.java.swing.plaf.motif.MotifLookAndFeel“

- **Macintosh**-Look-and-Feel
  - Benutzeroberfläche von Apple Macintosh-Computer nachempfunden
  - Explizite Deklaration erfolgt durch den String: (ersetze Zeile 5 aus Folie 5)
    → „javax.swing.plaf.mac.MacLookAndFeel“

Ereignis-Bearbeitung (1)

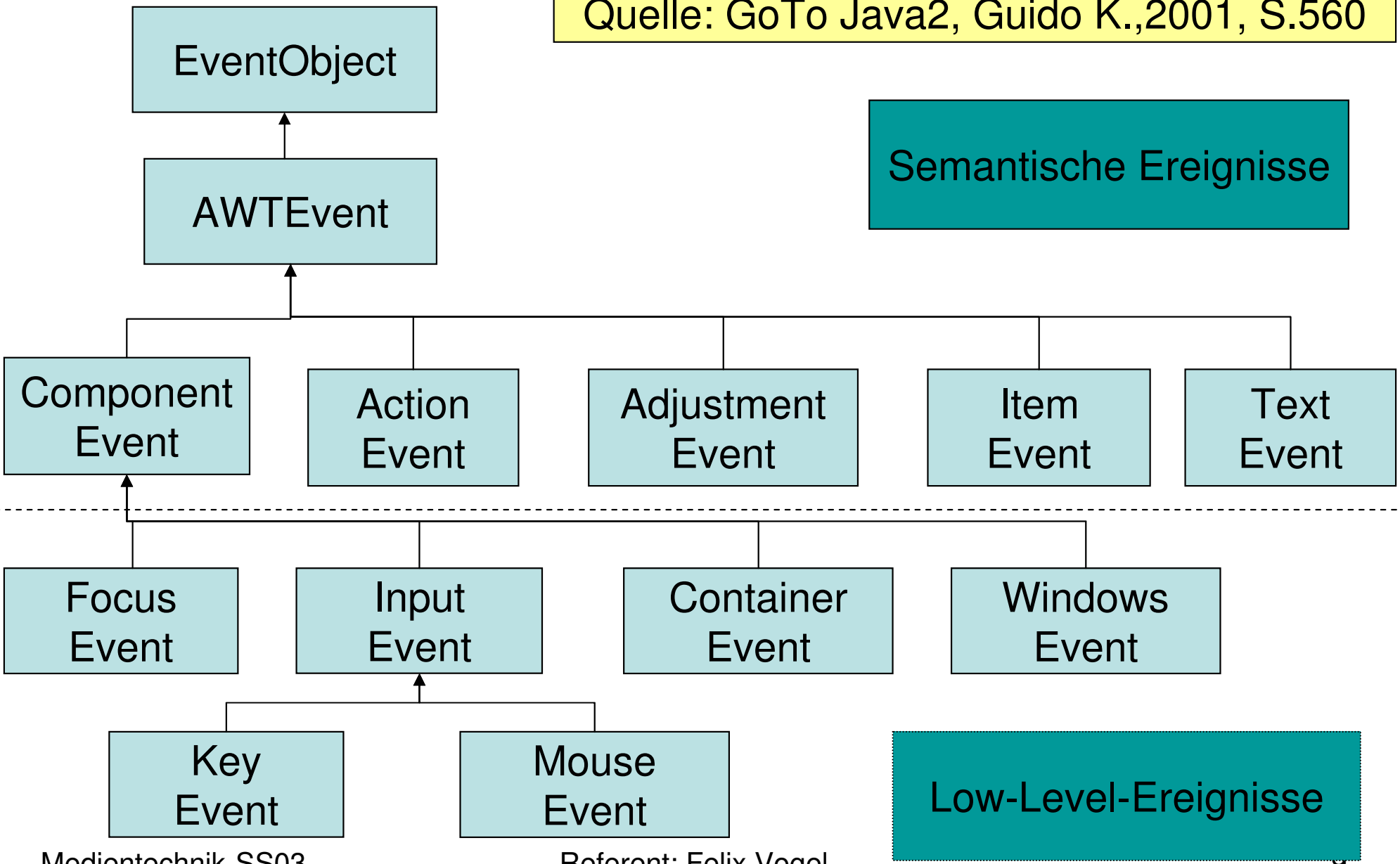
- Konzepte der Ereignis-Programmierung
  - Klare Trennung zwischen Ereignissen, Ereignisquellen und Ereignisempfängern
  - Das Modell der Kommunikation zwischen Ereignisquelle und Ereignisempfänger nennt man **Delegation Event Modell** (DEM ab JDK 1.1)
  - Vorteile der DEM
    - Klare Trennung zwischen Behandlung von Ereignissen und Darstellung der graphischen Oberfläche
    - Erleichtert die Erstellung von gut strukturierten Programm-Klassen
    - Verringerung des Nachrichtenverkehrs
    - Erhöhung der Performance des Nachrichtentransports
    - Erzeugung von robusten Codes
  - Nachteile der DEM
    - Komplexe GUI- Anwendungen sind zu langsam
    - Bei zunehmenden Ereignissen sind die Anwendungen schwerfällig

Ereignis –Bearbeitung (2)

- Ereignistypen
  - EventObject ist die Oberklasse aller Ereignisklassen
  - Ereignisklassen werden aus dem Paket *java.util.EventObject a*bgeleitet
  - Die Fähigkeit dieser Klasse ist: das Objekt zu speichern, das die Nachricht ausgelöst hat und durch den Aufruf der Methode getSource( ) anzuzeigen
  - AWT-spezifische Ereignisklassen sind aus der Klasse EventObject abgeleitet und im Paket *java.awt.events* zu finden
  - Low-Level-Ereignisse
    - Klassen für den Transfer von elementaren Nachrichten
      (z.B. Ausführen von Buttonereignissen, Setzen des Fokus)
  - Semantische Ereignisse
    - Klassen für die Übermittlung höherwertiger Ereignisse
      (z.B. Ausführen eines Kommandos, Ändern eines Zustandes)

Die Hierarchie der Ereignisklasse

Quelle: GoTo Java2, Guido K.,2001, S.560



Ereignis-Bearbeitung (3)

- Ereignisse
  - Ereignisse werden meist von Benutzer selbst ausgelöst
    (z.B. Mausklick, Mausbewegung, Tastatureingabe)
  - Zu jeder Art von Ereignis gibt es die zugehörige Ereignisklasse
    (z.B. Mausereignisse - Klasse MouseEvent)
  - Ereignisklassen sind im Paket *java.awt.events* enthalten
- Ereignisquellen
  - Ereignisquellen sind alle Komponenten einer graphischen Oberfläche
    (z.B. Buttons, Textfelder, oder Oberfläche selbst)
  - Jede Komponente muss explizit mit einem Ereignisempfänger verbunden sein
    (z.B. *Button.addMouseListener(new MouseAdapter( ) {.....})*

Ereignis-Bearbeitung (4)

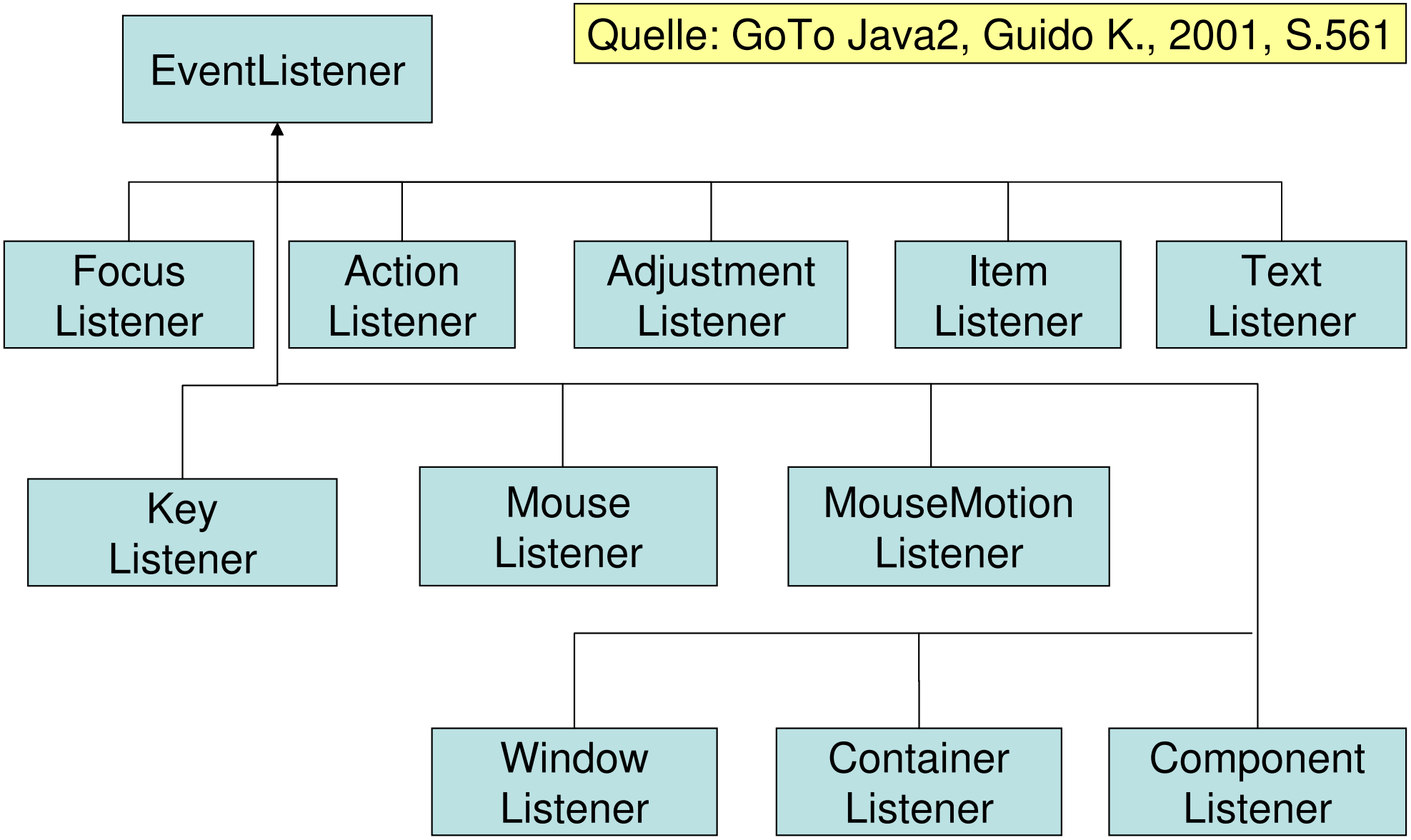
- Ereignisempfänger (auch „Ereigniswächter“)
  - Ereignisempfänger sind Instanzen, die über Methoden zur Behandlung von Ereignisse verfügen
  - Ereignisempfänger werden als Klassen erstellt: EventListener-Interface oder AdapterKlassen
  - Zu jeder Ereignisklasse gibt es das zugehörige EventListener-Interface

    (z.B. MouseEvent gehört zum Interface MouseListener)
  - EventListener-Interface
    - Definieren Methoden „Abhören“ von Ereignissen

      (z.B. *mouseClicked( )* gehört zum Interface MouseListener)
  - AdapterKlassen
    - Abstrakte Klassen mit leeren Methodenrümpfen

    (z.B. MouseAdapter ist Adapter-Klasse zum MouseListener-Interface)

Die Hierarchie der EventListener-Interface



Windows-Ereignisse

- Ereignsklasse ist WindowEvent
  - Zum Schließen eines Fensters
  - Zum Beenden einer Anwendung
- Dazugehörige Registrierungsmethode
  - *addWindowListener(WindowListener l)*
- Wichtigste Methode von WindowsListener-Interface
  - *windowClosing(WindowEvent e)*
- Abstrakte Klasse
  - WindowAdapter

Beispiel zur Windows-Ereignisse zum Schließen eines Frames

```
1.  // Einrichten der main-Methode
2.          public static void main(String[] args)
3.          {
4.          JFrame rahmen=new DesktopKlasse(); // Erstelle einen Frame
5.          // WindowsListener zum Schliessen des Fensters
6.          WindowListener w1=new WindowAdapter()
7.          {
8.                      public void windowClosing(WindowEvent we)
9.                              {  System.exit(0); }
10.         };
11.         // WindowListener zum Rahmen hinzufügen
12.         rahmen.addWindowListener(w1);
13.         rahmen.setSize(500,300);
14.         rahmen.setVisible(true);
15.         }
```

Action-Ereignisse

- Ereignisklasse ist ActionEvent
  - Für das Betätigen von Schaltflächen
  - Für das Drücken von Tasten
  - Für das Auswählen von Einträgen
- Dazugehörige Registrierungsmethode
  - *addActionListener(ActionListener l)*
- Wichtigste Methode von ActionListener-Interface
  - *actionPerformed(ActionEvent e)*
- Abstrakte Klasse
  - ActionAdapter
- Unterscheidung von Action-Ereignissen
  - *getSource( )*

Beispiel zur Action-Ereignisse zur Ermittlung von Ereignisquellen

```
1.  public void actionPerformed(ActionEvent ae)
2.          {
3.                  // Bestimme die Quelle von Ereignissen
4.                  Object quelle=ae.getSource();
5.                  if (quelle==mi1) createFrame("Winter.gif");
6.                  if (quelle==mi2) createFrame("Sonnenuntergang.gif");
7.                  if (quelle==mi3) createFrame("Wasserlilien.gif");
8.                  if (quelle==mi4) createFrame("Blaue Berge.gif");
9.          }
```

Maus-Ereignis

- Elemente zur Maus-Ereignisbehandlung
  - Ereignisklasse MouseEvent
    (Klasse zur Erzeugung von Mausereignissen)
  - Listener-Interface MouseListener
  - Adapterklasse MouseAdapter
    (Klasse stellt die erforderlichen Methoden für die Maus bereit)
  - Listenermethoden mouseClicked(MouseEvent me)
    mousePressed(MouseEvent me)
    mouseExited(MouseEvent me)
    mouseReleased(MouseEvent me)…
    →Methoden zur Verarbeitung der Ereignisse
  - Listener hinzufügen addMouseListener
    (Komponente wird mit dem Ereignisempfänger verbunden)

Beispiel zu Maus-Ereignissen (1)

```
1.  import java.awt.*;
2.  import java.awt.event.*;
3.  import javax.swing.*;

4.  public class ButtonFrame extends JFrame
5.  {       Adapter adapter=new Adapter(this); //s. nächste Folie
6.          JButton button=new JButton("Klicken");
7.          JTextField text=new JTextField("",10);
8.          public ButtonFrame()
9.          {           super("Einfache Ereignisse");
10.                     JPanel panel=new JPanel();
11.                     button.addMouseListener(adapter);
12.                     panel.add(button);
13.                     panel.add(text);
14.                     setContentPane(panel);
15.         }
16.}        // Hier fehlt noch die main-Methode für das JFrame
```

Beispiel zu Maus-Ereignissen (2)

```
1.  import java.awt.*;
2.  import javax.swing.*;
3.  import java.awt.event.*;

4.  public class Adapter extends MouseAdapter
5.  {
6.          ButtonFrame be; int m;
7.          Adapter(ButtonEreignis be)  {  this.be=be;  }
8.          public void mouseClicked(MouseEvent e)
9.          {
10.                 m++;
11.                 be.text.setText(m+". Mal geklickt");
12.         }
13. }
```

Grundkonzepte und Grundlagen (1)

- Komponenten
  - Schwergewichtige Komponenten
    - Komponenten werden in einem eigenen undurchsichtigen Fenster gerendert

      (Objekte vom Typ JFrame, JApplets, JDialog, JWindow)
  - Leichgewichtige Komponenten
    - Komponenten besitzen einen durchsichtigen Hintergrund

    (alle Swing-Komponenten außer den o. g.)

Grundkonzepte und Grundlagen (2)

- Typen von Conainern
  - Top-Level-Container
  - General-Purpose Container
  - Special-Purpose Container
- Typen von Komponenten
  - Basic Controls
  - Nicht editierbare Komponenten
  - Editierbare Komponenten

Grundkonzepte und Grundlagen (3)

- Top-Level-Container
  - Bilden den äußeren Rahmen eines Programms
  - Dazu zählen drei Komponenten: Applets, Dialoge und Rahmen
    - (Basisklassen: JApplet, JDialog und JFrame)
  - Unterschiede beim Einfügen von Unterkomponenten
    - AWT: Komponenten können direkt hinzugefügt werden
      - *applet.add(neueKomponente)*
    - Swing: Komponenten werden über eine Zwischenkomponente (**content pane**) hinzugefügt
      - *applet.getContentPane().add(neueKomponente)*

Grundkonzepte und Grundlagen (4)

- General-Purpose Container
  - Behälter für allgemeine bzw. universelle Anwendungen
  - Dazu zählen:
    - Panels (Behälter für weiteren Komponenten)
    - Scroll panes (Zeigen Ausschnitt größerer Komponenten)
    - Split panes (Verteilen von Komponenten in zwei Fenster)
    - Tabbed panes (Prinzip von Karteikarten)
    - Toolbars (Schaltflächen mit graphischen Elementen)

Grundkonzepte und Grundlagen (5)

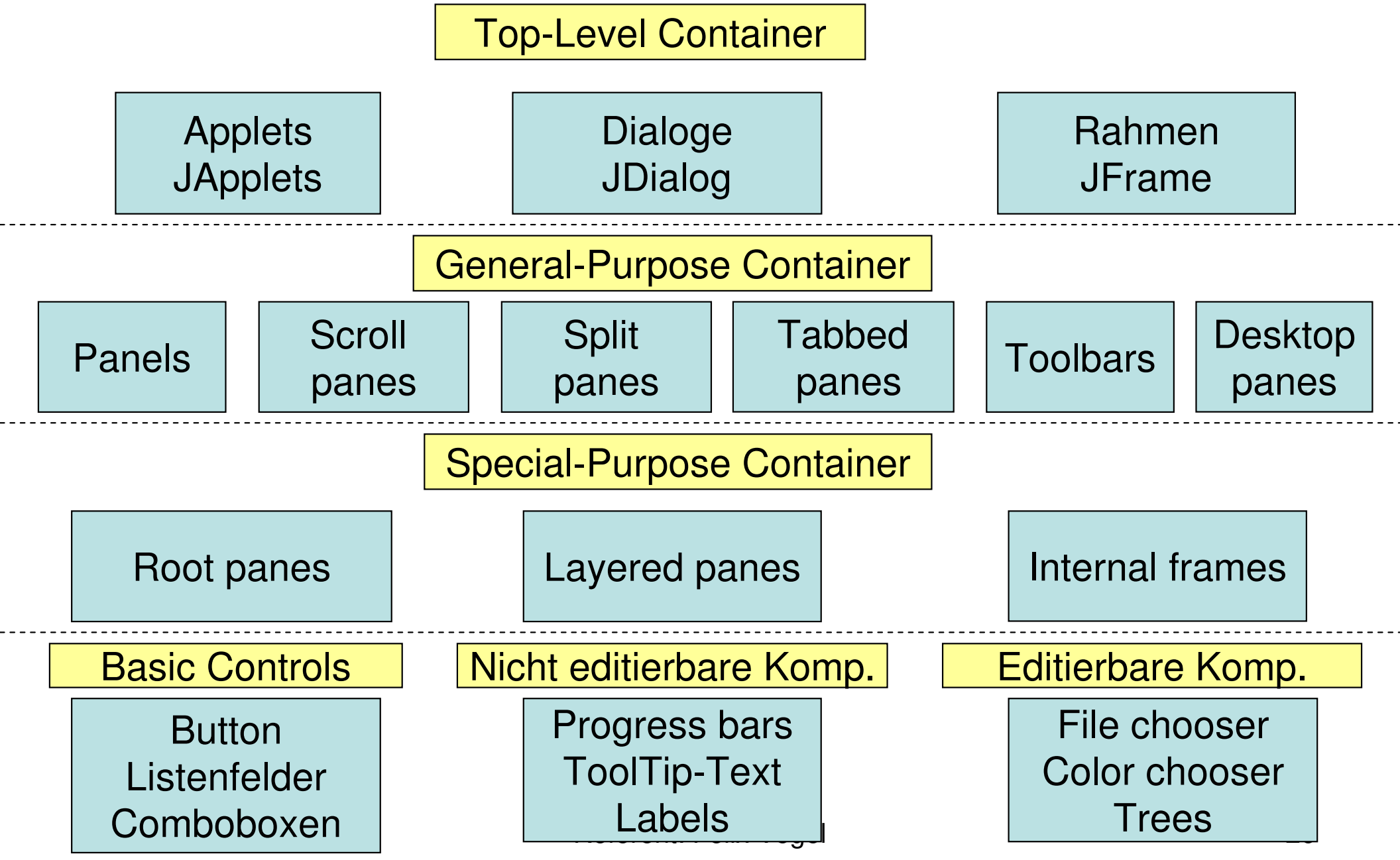
- Special-Purpose Container
  - Behälter für spezielle Einsatzgebiete
  - Dazu zählen:
    - Internal frames (Bildung von MDI-Anwendungen durch interne Rahmen)
    - Layered panes (Darstellung von überlappenden Komponenten)
    - Root panes (beinhalten

    Glass pane, MenuBar, Layered pane, Content pane)

Grundkonzepte und Grundlagen (6)

- Basic Controls
  - Gruppen von Komponenten für Eingabe und Anzeigen von Zuständen eines Programms
    - Z.B. Schaltflächen, Radiobuttons, Checkboxen
- Nicht editierbare Komponenten
  - Komponenten zur Ausgabe von nicht mehr editierbaren Informationen
    - Z.B. Hilfetexte, ToolTip-Text, Fortschrittsbalken, Labels
- Editierbare Komponenten
  - Komponenten zur Benutzereingabe
    - Z.B. Standarddialog, Color chooser, File chooser, Trees

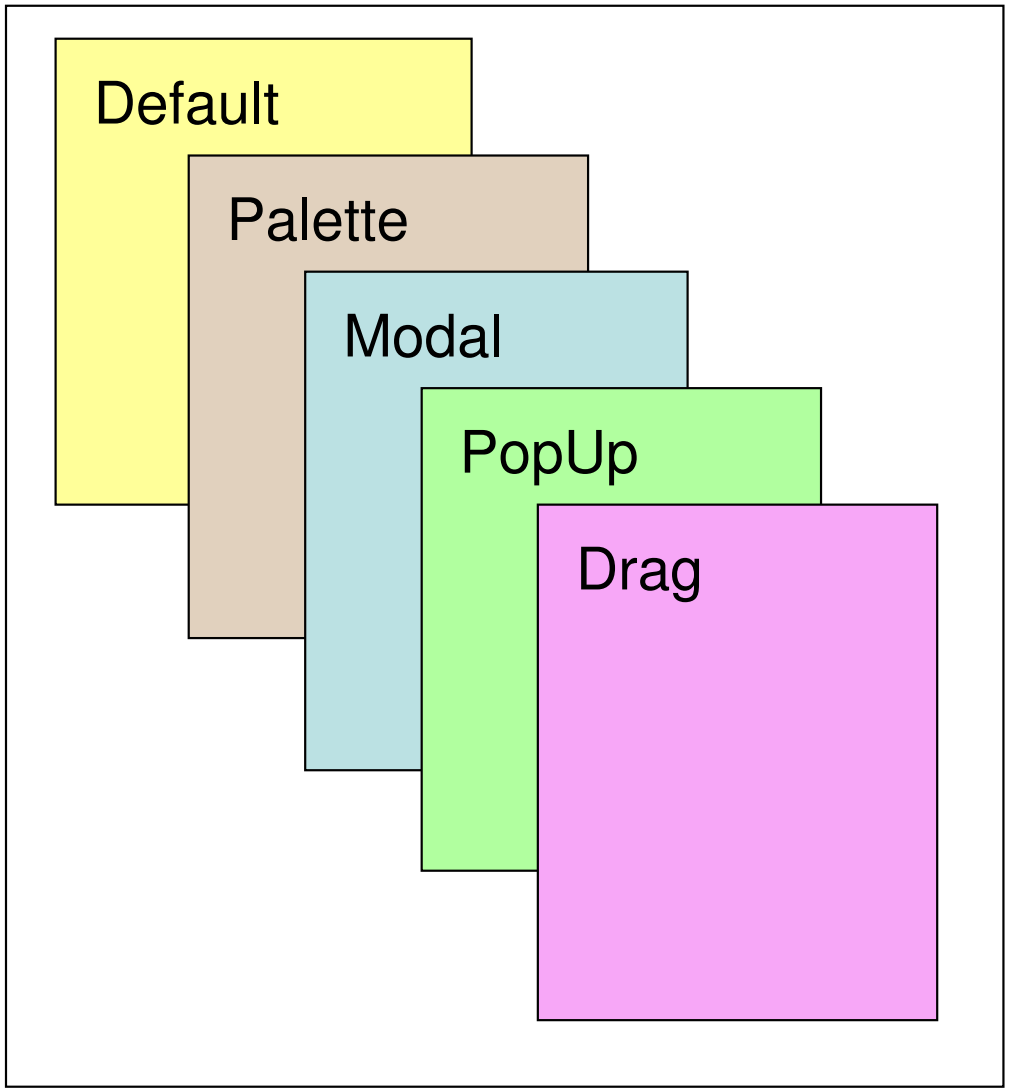
Grundkonzepte und Grundlagen (7)



Verschiedene Pane-Typen (1)

- JLayeredPane
  - Container mit mehreren Schichten
  - Teilebenen
    - DEFAULT_LAYER (Aufnahme der meisten Komponenten)
    - PALETTE_LAYER (Aufnahme beweglicher Komponenten)
    - MODAL_LAYER (Aufnahme modaler Dialoge)
    - POPUP_LAYER (Positionieren von Popup-Dialogen)
    - DRAG_LAYER (Aufnahme kurzfristiger verschiebbarer Komponenten)
  - Bewegen der Komponenten auf verschied. Ebenen
    - Methode: *moveToFront( )* und *moveToBack( );*
  - Direktes Anwählen eines Ziel-Layers
    - *layeredPane.add(child, JLayeredPane.DEFAULT_LAYER);*

Verschiedene Pane-Typen (2)



Anordnung der Teilebenen von JLayeredPane

Verschiedene Pane-Typen (3)

- JDesktopPane
  - Container zur Erstellung von virtuellen Bildschirmen (z.B. MDI-Anwendungen)
    - Objekte zur Darstellung von internen Frames
      - z.B. JInternalFrame jif = new JInternalFrame()
  - Erzeugen Objekten vom Typ JDesktopPane
    - JDesktopPane dp = new JDesktopPane( )
  - Nachteile von Internen Rahmen
    - Undurchsichtig
    - verdecken darunter liegende Rahmen

Verschiedene Pane-Typen (4)

- JViewport
  - Container zur Anzeige von Ausschnitten einer Komponente
  - Verschiebung des Viewports über der Komponente z. B. mit Rollbalken
  - Erzeugen Objekte von Typ JViewport ohne Argumente
    - Objekt: JViewport jvp = new JViewport( );
    - Methode: *setViewPosition(int a, int b);*
    - Methode: *getViewPosition( );*

Verschiedene Pane-Typen (5)

- JScrollPane
  - Container zur Anzeige von Ausschnitten einer Komponente mit Hilfe von horizontalen und vertikalen Rollbalken
  - JScrollPane-Objekte enthalten Viewports
    - Objekt: JScrollPane jsp = new JScrollPane(label)
  - Für zusätzliche Ansichten mit Linealen
    - Methode: *setColumnHeaderView(hLineal);*
    - Methode: *setRowHeaderView(vLineal);*

Verschiedene Pane-Typen (6)

- JSplitPane
  - Container zur Darstellung **zweier** Komponenten wahlweise übereinander oder nebeneinander
  - Komponenten werden durch Balken getrennt
  - Erzeugen Objekte von Typ JSplitPane
    - JSplitPane sp=new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, panel, jep)
  - Ausrichtung der Komponenten durch Konstanten
    - HORIZONTAL_SPLIT (nebeneinander positionieren)
    - VERTIKAL_SPLIT (untereinander positionieren)
  - Bereich der Abgrenzung verkleinern oder vergrößern
    - Methode: *setOneTouchExpandable(boolean newValue);*

Verschiedene Pane-Typen (6)

- JTabbedPane
  - Container zur Aufnahme unterschiedlicher Oberflächen
  - Normalisierte Panels selektiert in Form von Karteikarten über Reiter
  - Erzeugen Objekte von Typ JTabbedPane
    - JTabbedPane tp = new JTabbedPane( )
  - Reiter mit Text benennen
    - Methode: *add(„Text“, panel)*

Spezielle Swing-Komponenten (1)

- Tabellen mit JTable
  - Container zur Erstellung von Tabellen
  - Paket zur Erstellung von Tabellen
    - javax.swing.table.*
  - Anzeige von verdeckten Feldern der Tabelle erfordert ein JScrollPane-Objekt
    - JTable table = new JTable(daten, namen)
    - JScrollPane sp = new JScrollPane(table)
  - Definition der Tabelleninhalte
    - Object [ ] [ ] daten = {...} (Array für Datensätze)
    - Object [ ] namen= {...} (Array für Spaltenüberschriften)
  - JTable-Objekte dienen ausschließlich der Anzeige der Daten bzw. ihrer Bearbeitung

Spezielle Swing-Komponenten (2)

- Drei Typen von Tabellenmodellen
  - TableModel (Bearbeitung von Tabellen)
  - TableColumnModel (Definieren von Spalten)
  - ListSelectionModel (Selektieren von Tabellenelementen)
  - Konstruktor für die Tabellenmodelle
    - Public JTable(TableModel dm, TableColumnModel cm, ListSelectionModel sm)
  - Alle drei Modelle werden separat instantziert

Spezielle Swing-Komponenten (3)

- TableModel
  - Interface stellt die grundlegende Funktionalität von Datenmodellen für Tabellen auf
  - AbstractTableModel
    - Die abstrakte Klasse enthält Standardimplementierungen für die meisten TableModel-Methoden
    - Vereinfacht Generierung von TableModelEvents
  - DefaultTableModel
    - Unterstützt das Ändern und das Speichern von Daten
    - Stellt Methoden zur Modifizierung von Tabellen auf

Erzeugen von einfachen Menüs

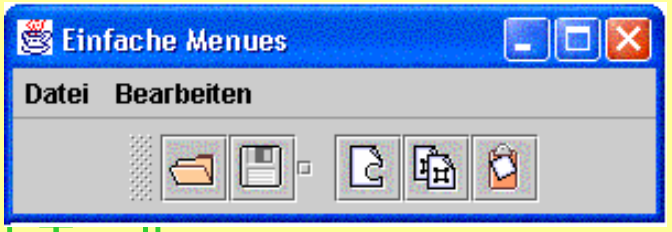
- Menüleisten
  - Optionale Bestandteile von Root panes
  - Ein abgeleitetes Objekt von der Klasse JMenuBar
    - JMenuBar mb = new JMenuBar( )
  - Einfügen von Objekten (Menüs) erfolgt durch die Methode:
    - *add( )*
  - Menüzeilen dienen als Container für die Menüs
- Menüs
  - Objekte der Klasse JMenu
    - JMenu m1 = new JMenu
  - Die Menüs sind hingegen Container für die Menüeinträge
- Menüeinträge
  - Stellen die Beziehung der auszuführenden Funktionen dar
    - JMenuItem mi1 = new JMenuItem(String)


```
1.  public class Menues extends JFrame
2.  {       public Menues()
3.          {   super("Einfache Menues");//Frametitel
4.          JMenuBar mb = new JMenuBar();
5.          JMenu    m1 = new JMenu("Datei");
6.          JMenu    m2 = new JMenu("Bearbeiten");
7.          JMenu    m3 = new JMenu("Einfügen");
8.          JMenuItem mi1=new JMenuItem("Öffnen");
9.          JMenuItem mi2=new JMenuItem("Speichern");
10.         JMenuItem mi3=new JMenuItem("Schließen");
11.         JMenuItem mi4=new JMenuItem("Ausschneiden");
12.         JMenuItem mi5=new JMenuItem("Kopieren");
13.         JMenuItem mi7=new JMenuItem("Alle");
14.         JMenuItem mi8=new JMenuItem("Markierte");
15.         mb.add(m1);      mb.add(m2);
16.         m1.add(mi1);     m1.add(mi2);     m1.add(mi3);
17.         m2.add(mi4);     m2.add(mi5);
18.         m2.add(m3);                  m3.add(mi7);     m3.add(mi8);
19.         setJMenuBar(mb);  }  // MenüBar dem Frame hinzufügen
20.         // Hier steht die main-Methode }
```

Erzeugung von einfachen Toolbars

- Toolbars
  - Darstellung u. a. von Menüeinträgen durch Symbole
  - Die Hauptklasse von Toolbars ist JToolBar
    - JToolBars tb = new JToolBar( )
  - Elemente von Toolbars sind Buttons mit Icons
    - JButton b = new JButton(new ImageIcon(„bild.gif“))
  - Einfügen von Objekten in Toolbars erfolgt durch die Methode: *add( )*
  - Toolbars dienen als Container für die Objekte

Beispiel für eine Menüleiste mit Symbole




```
1.  public class Toolbar
2.  {
3.          public Toolbar(JPanel pan)
4.          {JToolBar toolb=new JToolBar();  // Object-Toolbar erzeugen
5.          JButton btn1=new JButton(new ImageIcon("open.gif"));
6.          JButton btn2=new JButton(new ImageIcon("save.gif"));
7.          JButton btn3=new JButton(new ImageIcon("close.gif"));
8.          JButton btn4=new JButton(new ImageIcon("cut.gif"));
9.          JButton btn5=new JButton(new ImageIcon("copy.gif"));
10.         JButton btn6=new JButton(new ImageIcon("paste.gif"));
11.         toolb.add(btn1); toolb.add(btn2); toolb.add(btn3);
12.         toolb.addSeparator();  // schafft Zwischenraum zwischen Symbolen
13.         toolb.add(btn4); toolb.add(btn5); toolb.add(btn6);
14.         pan.add(toolb);  // Toolbar dem Panel hinzufügen
15.         }
16. }
```

Literatur

- **Guido Krüger**. (2000): GoTo Java2, Handbuch der Java-Programmierung, 2.Auflg. München.
- **Satyaraj Pantham** (1999): Pure JFC Swing, A Code-Intensive Premium Reference, 1.Auflg. ,SAMS, Indianapolis/USA.
- **Stefan Middendorf** [u.a.](1996): Java: Programmier-handbuch und Referenz.1.Auflg. Heidelberg.
- **Oliver Böhm** (2002): Java Software Engineering unter Linux,1.Auflg. Nürnberg.
- **Ralf Jesse** (2002): Das Einsteigseminar Java Swing. 1.Auflg. Bonn.

- Vielen Dank für Ihre Aufmerksamkeit