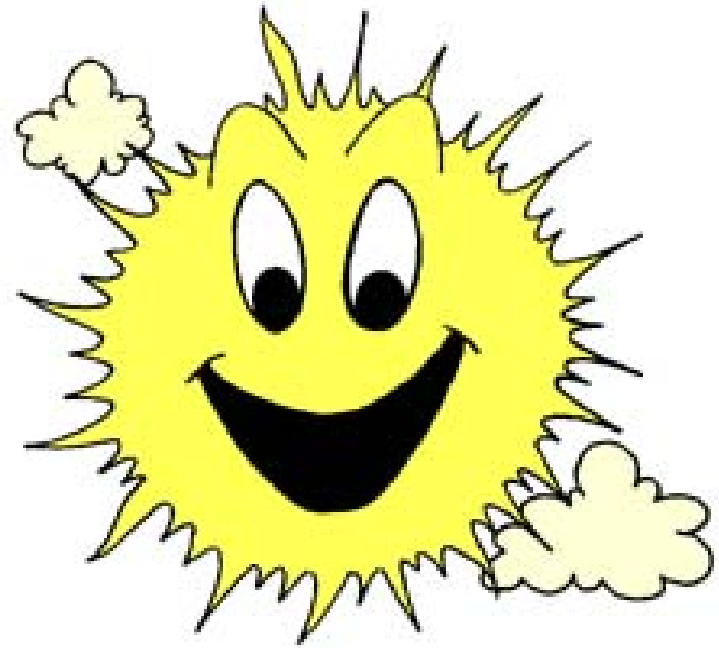
Wiederholung 5. Vorlesung



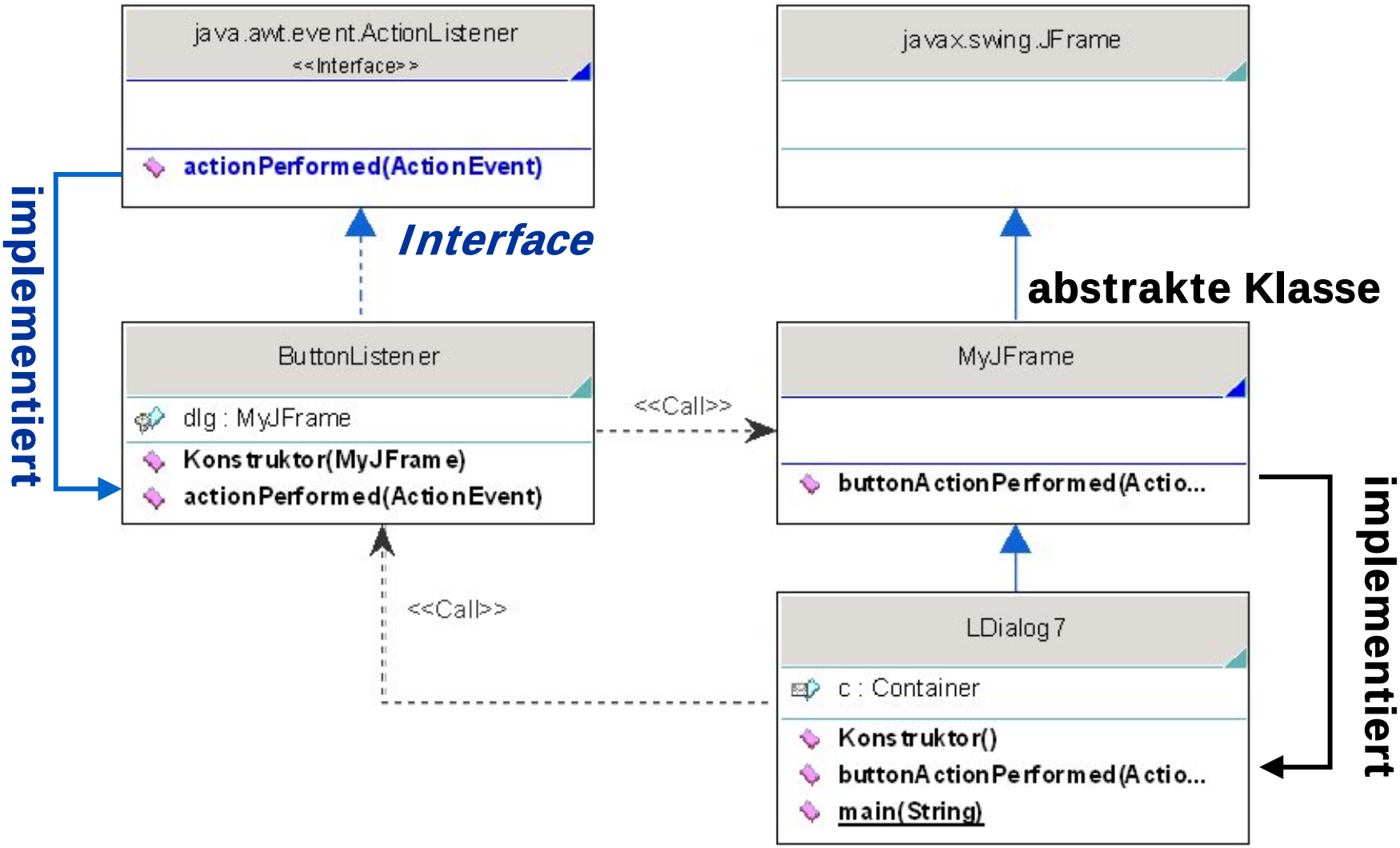
4.3 Varianten der Ereignisverabeitung

Informatik

Bei der Ereignisverarbeitung kann man prinzipiell vier Varianten unterscheiden:

- Listener-Klasse als innere Klasse (Bsp: LDialog)
- Listener-Klasse als anonyme Klasse (Bsp: LDialog3)
- Container-Klasse wird selbst zur Listener-Klasse (Bsp: LDialog2)
- Listener-Klasse wird als separate Klasse realisiert (Bsp: LDialog4)

4.3.2 Bsp: Interface und abstrakte Klasse



4.3.3 Optimierung Listener

- mit *ActionEvent.**getSource()*** kann der Auslöser des Ereignisses bestimmt werden.
- Wir nutzen dies in den Bsp: LDialog – LDialog4
- Sollen **mehrere Schaltflächen** die **gleiche Aktion** auslösen, ist dieses Vorgehen **nicht optimal**, da alle Quellen abgefragt werden müssen.

4.3.3 Optimierung Listener (2) (LDialog5 u. 6)

Informatik

Lösung:

- Man kann Schaltflächen ein Kommando zuweisen, welches diese dann mit dem ActionEvent weitergeben:
- In der Dialogklasse:

```
JButton but = new Button( “Button Name“ );
but.setActionCommand(“tue was!“ );
```

- Mehreren Buttons oder Menu-Punkten kann gleiches Kommando zugewiesen werden
- Im ActionListener oder der entsprechenden Funktion erhält man aus dem `ActionEvent e` als `String` das Kommando mit `e.getActionCommand()` , das entsprechend ausgewertet werden kann, z.B:

```
if(e.getActionCommand().equals((“tue was!“))
    ...
```

4.3.3 Optimierung Listener (LDialog5)

```
public LDialog5() // Konstruktor
{
    // Buttons ein Komando zuweisen
    bRot.setActionCommand( "wechsle Rot!" );
    bBlau.setActionCommand( "jetzt Blau!" );

    ...
}

public void buttonActionPerformed( ActionEvent e )
{
  if (e.getActionCommand().equals("wechsle Rot!"))
      pCenter.setBackground( Color.RED );
  else if
    (e.getActionCommand().equals("jetzt Blau!“))
      pCenter.setBackground( Color.BLUE );
}
```

4.3.4 Menüs in Java (LDialog 6)

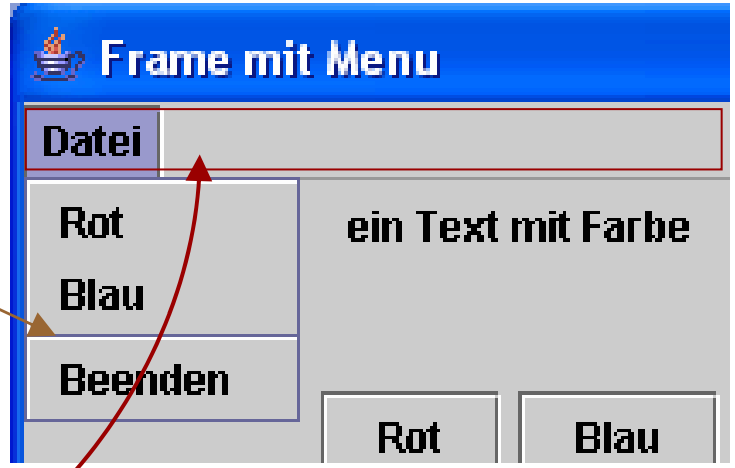
```
public class LDialog6 extends JFrame
{
   ...
   JMenuBar jMenuBar =
      new JMenuBar();
   JMenu jMenuFile =
      new JMenu( "Datei" );
   JMenuItem itemRot =
      new JMenuItem( "Rot" );
   JMenuItem itemBlau =
      new JMenuItem( "Blau" );
   JMenuItem itemQuit =
      new JMenuItem( "Beenden" );
```

4.3.4 Menüs in Java (2) (LDialog 6)

```
public LDialog6() { //Konstruktor
   ...
   // Menuleiste aufbauen
   jMenuFile.add(itemRot);
   jMenuFile.add(itemBlau);
   jMenuFile.addSeparator();
   jMenuFile.add(itemQuit);

   jMenuBar.add(jMenuFile);

   setJMenuBar(jMenuBar);
```



4.3.4 Menüs in Java (3) (LDialog 6)

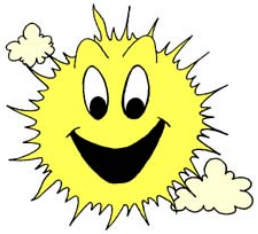
```
public LDialog6() { //Konstruktor (Fortsetzung)
  ...
  // Buttons/Menus ein Komando zuweisen
   bRot.setActionCommand( "wechsle Rot!" );
   itemRot.setActionCommand( "wechsle Rot!" );
   bBlau.setActionCommand( "jetzt Blau!" );
   itemBlau.setActionCommand( "jetzt Blau!" );
   itemQuit.setActionCommand( "Fertig" );

   // ActionListener erzeugen
   ButtonListener bl = new ButtonListener();
   // Den Schaltflächen den Listener bekannt machen.
   bRot.addActionListener( bl );
   bBlau.addActionListener( bl );
   itemRot.addActionListener( bl );
   itemBlau.addActionListener( bl );
   itemQuit.addActionListener( bl );
```

4.3.4 Menüs in Java (4) (LDialog 6)

```
class ButtonListener implements ActionListener
{
   /**
    * einzige Methode von ActionListener
    * @param e ActionEvent. wird von Komponente an
    * den Listener gesendet.
    */
   public void actionPerformed( ActionEvent e ) {
     if (e.getActionCommand().equals("wechsle Rot!“))
         pCenter.setBackground( Color.RED );
     else if
       (e.getActionCommand().equals("jetzt Blau!"))
         pCenter.setBackground( Color.BLUE );
     else if (e.getActionCommand().equals("Fertig"))
         System.exit(0); // Anwendung beenden
   }
}
```

Ende der Wiederholung



4.3.5 Java List-Boxen (Beispiel mvc1, DlgLook&Feel)

Informatik

```
String[] listItems = {
    "GTK", "Metal", "Motif",  "Windows" };
JList jList = new JList();

public Konstruktor() {
  ...
  jList.setListData( listItems ); // Einträge setzen
  // Einzelauswahl setzen (oder andere Einstellung)
  jList.setSelectionMode(
        ListSelectionModel.SINGLE_SELECTION );
  jList.setSelectedIndex(1); // default setzen (Metal)

  MyListListener lsl = new MyListListener();//Listener
  jList.addListSelectionListener( lsl );
  ...
}
```

4.3.5 List-Boxen (2) (Beispiel mvc1, DlgLook&Feel)

```
class ListSelListener implements
    ListSelectionListener
{
  public void valueChanged(ListSelectionEvent e) {
    /* Hier Aktionen implementieren
     * z.B ListSelectionEvent e enthält Infos
     */

    // oder jList kann über Auswahl abgefragt werden
    int sel = jList.getSelectedIndex();
  }
}
```

Siehe API Dokumentation und Literatur, auch für weitere Stichwörter wie ListSelectionModel

5. Layout Manager in Java

- Flow Layout
- Border Layout
- Box Layout
- Grid Layout
- Grid Bag Layout
- ...

5.1 Flow-Layout

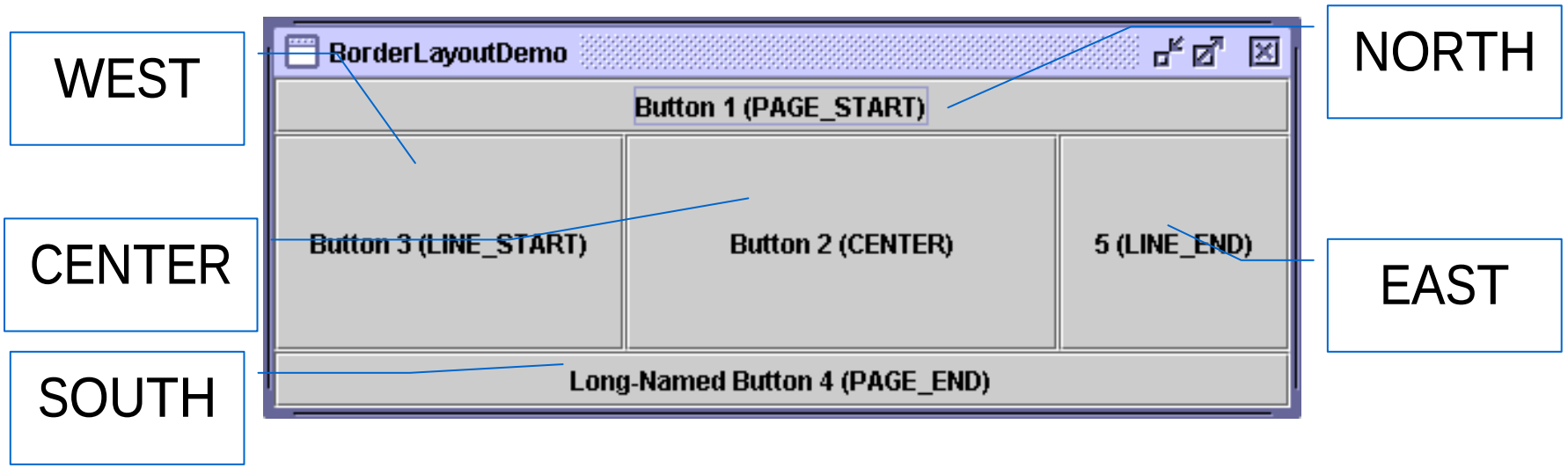
Layout setzen:

```
pane.setLayout( new FlowLayout() );
```

Komponente Component bx wird dem Container pane nacheinander in der gewünschten Reihenfolge hinzugefügt:

```
pane.add( button1 );
pane.add( button2 );
…
```

5.2 Border-Layout



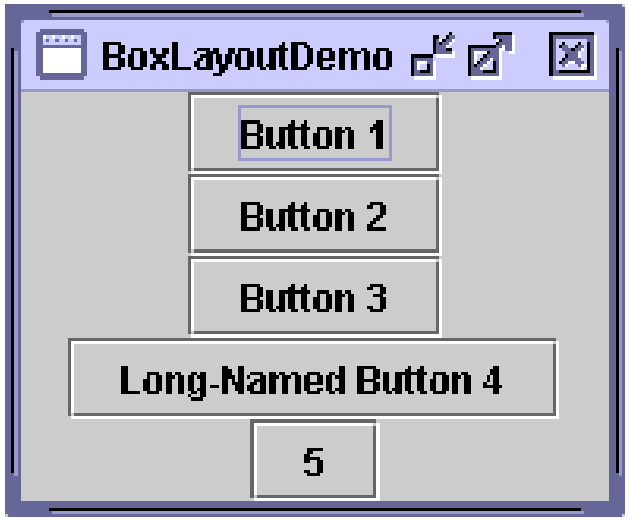
Layout setzen:

```
pane.setLayout( new BorderLayout() );
```

Komponente Component bx wird dem Container pane mit einem *„Constraint“* hinzugefügt:

```
pane.add( b2, BorderLayout.CENTER );
pane.add( b1, BorderLayout.NORTH );
…
```

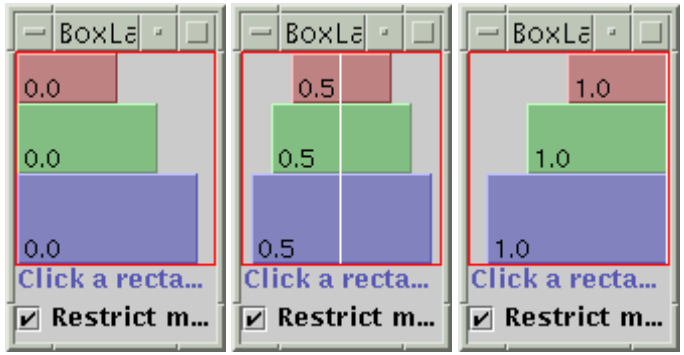
5.3 Box-Layout



Layout setzen:

```
pane.setLayout(
   new BoxLayout(
      pane, BoxLayout.PAGE_AXIS));

oder BoxLayout.LINE_AXIS
```



Ausrichtung setzen (hier CENTER):

```
b1.setAlignmentX(
    Component.CENTER_ALIGNMENT);
...
```

Komponente Component bx wird dem Container pane nacheinander in der gewünschten Reihenfolge hinzugefügt:

```
pane.add( b1 );
...
```

5.4 Grid-Layout

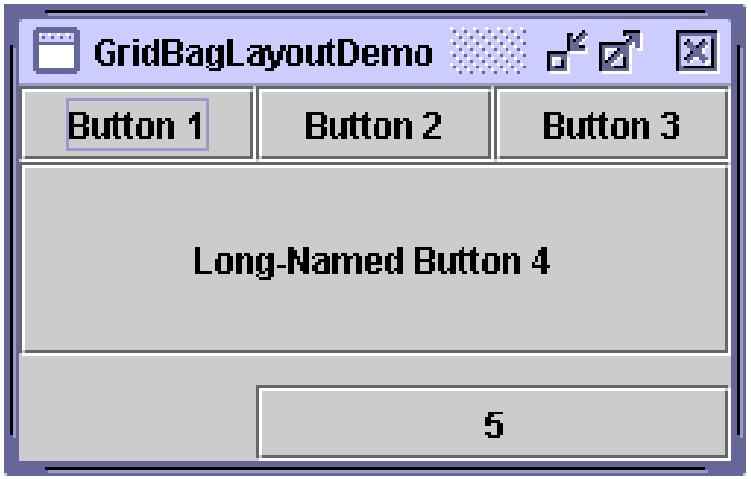
Layout setzen (Zeilen, Spalten) explizit oder **0**: automatisch :

```
pane.setLayout( new GridLayout(0, 2) );
```

Komponente Component bx wird dem Container pane hinzugefügt:

```
pane.add( button1 );
pane.add( button2 );
pane.add( button3 );
pane.add( longButton4 );
...
```

5.5 Grid-Bag-Layout



Dies ist nicht ganz trivial!

→

GUI-Builder wie beispielsweise der JBuilder unterstützen daher den Benutzer (Design-Ansicht) bei der Aufgabe ...

Layout setzen:

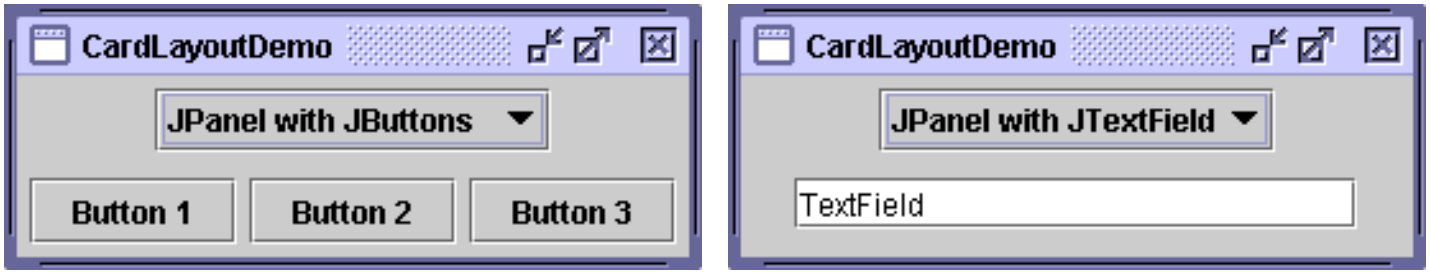
```
JPanel pane = new JPanel(new GridBagLayout());
GridBagConstraints c = new GridBagConstraints();
```

Constraint **c** wird für jede Komponente konfiguriert:

```
c.weightx = 0.5;      c.gridx = 0;      c.gridy = 0;
pane.add( button1, c ); // Hinzufügen zum Container

c.gridx = 1;      c.gridy = 0;
pane.add( button2, c ); // Hinzufügen zum Container
...
```

5.6 weitere Layout-Manager



CardLayout

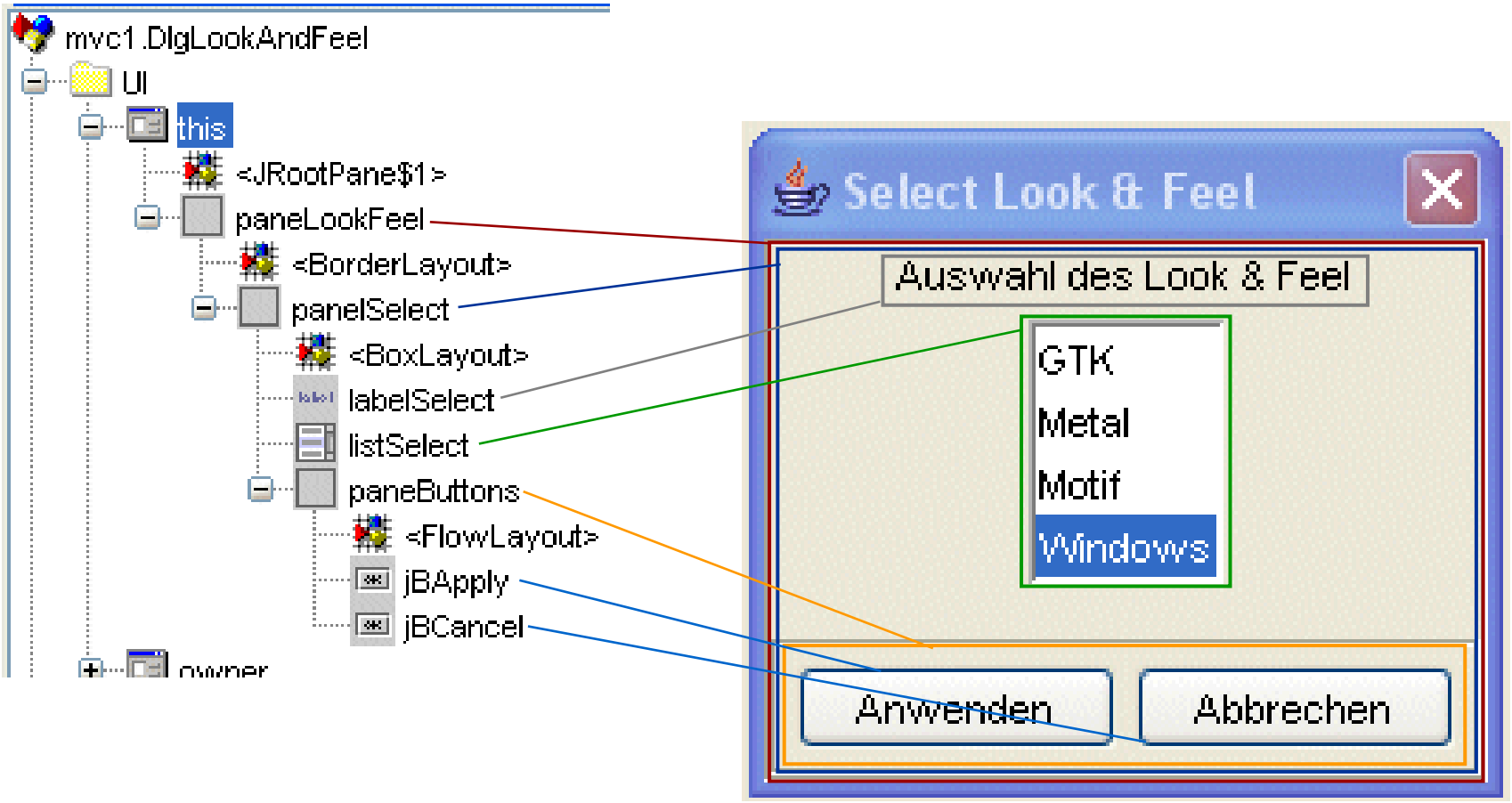
TabbedPane

Alle Details hierzu und zu weiteren Layout-Managern können in der Literatur, entsprechenden Tutorials und den API-Referenzen bei Bedarf nachgelesen werden

6. Beispiel einer Anwendung (techn. Sicht)

- Anwendung „mvc1“
  - Look & Feel in Java
  - Aufbau einer Oberfläche mit Layout Managern
  - Handhabung einer größeren Anzahl von Schaltflächen
  - Ereignis-Listener für Anwendungen und
  - Ereignisbearbeitung
- Einführung des MVC-Musters
- Erweiterung des ersten Beispiels
  - Observable – Observer Muster
  - Ein einfaches Model für das Zahlenkonversions-Beispiel
  - Erweiterung der Dialoge durch das Observer-Interface

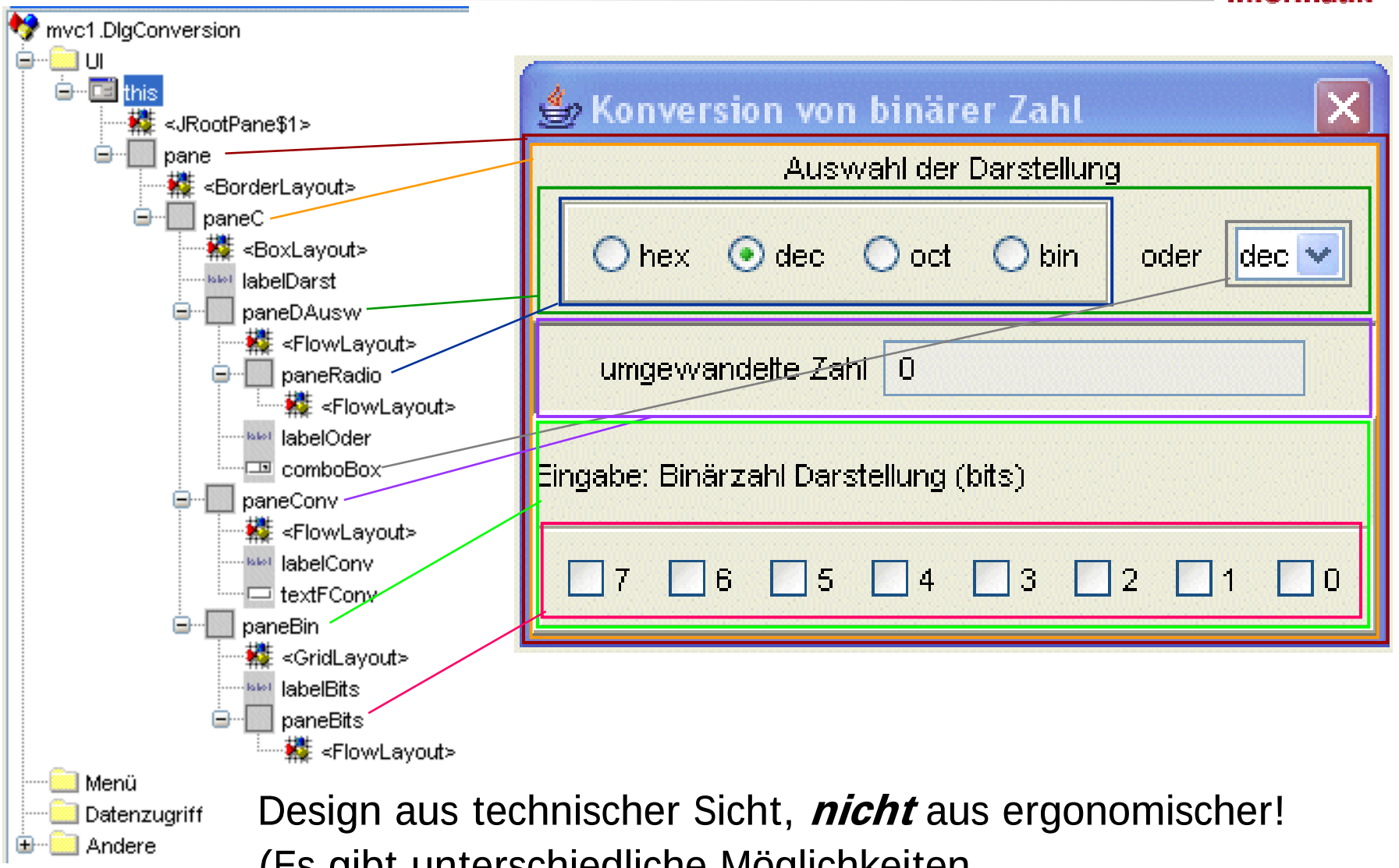
6.1 Beispiel „mvc1“ techn. Design (DlgL&F)



Design aus technischer Sicht, ***nicht*** aus ergonomischer!

(Es gibt unterschiedliche Möglichkeiten,
ein solches Fenster zu realisieren)

6.1 Beispiel „mvc1“ techn. Design (DlgConv)



Design aus technischer Sicht, ***nicht*** aus ergonomischer!

(Es gibt unterschiedliche Möglichkeiten,
ein solches Fenster zu realisieren)