K07 JTable

- Inhalt
 - 1. JTable
 - 1. Übersicht
 - 2. Einfache Tabelle
 - 3. Konfiguration
 - 4. Selektion
 - 5. Zugriff auf Inhalt
 - 2. Modelle
 - 1. Tabellenmodell
 - 2. Spaltenmodell
 - 3. Renderer



Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2

K07 JTable Folie 1

Lernziele

- Sie kennen die Möglichkeiten von JTable und sind in der Lage diese in eigenen Applikationen einzusetzen
- Sie kennen die Modelle
 - Tabellenmodell
 - Spaltenmodell
- Sie können durch den Einsatz von Renderern das Erscheinungsbild von Zellen bestimmen

LUZERN

Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2

1.1 Übersicht (1)

- JTable
 - Eines der wichtigsten Dialogelemente ist eine Tabelle
 - · Sehr anspruchsvoll
 - JTable
 - Swing-Klasse, welche Programmierarbeit erleichtert
 - Darstellung von tabellarisch Daten
 - textuell
 - graphisch
 - Weitreichende Möglichkeiten
 - Konfiguration der Tabelle
 - Anpassen von Inhalten
 - · Reagieren auf Benutzerereignisse



Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2

K07 JTable

1.1 Übersicht (2)

Konstruktoren

public JTable(Object[][] rowData, Object[] columnNames)

- Daten in zweidimensionalem Array (Zeilen/Spalten)
- Array mit Bezeichnung der Spaltenköpfe

public JTable(Vector rowData, Vector columnNames)

- Datenvektor (Vektor mit Vektoren)
- Vektor mit Bezeichnung der Spaltenköpfe

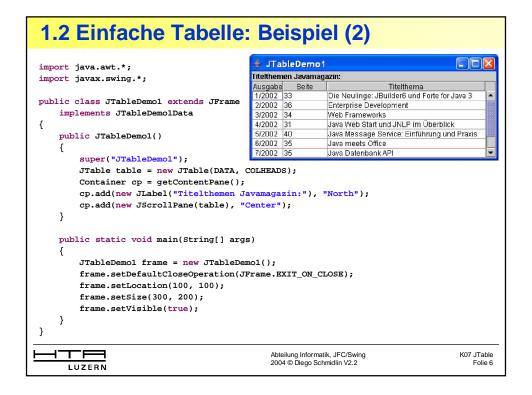
public JTable(TableModel dm, TableColumnModel cm, ListSelectionModel sm))

- TableModel stellt Daten zur Verfügung
- TableColumnModel definiert Spalten
- ListSelectionModel definiert Verhalten bei Selektion



Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2

1.2 Einfache Tabelle: Beispiel (1) Daten in zweidimensionalem Array public interface JTableDemolData public static final String[][] DATA = {" 1/2002", "33", "Die Neulinge: JBuilder6 und Forte for Java 3"}, {" 2/2002", "36", "Enterprise Development"}, {" 3/2002", "34", "Web Frameworks"}, {" 4/2002", "31", "Java Web Start und JNLP im Überblick"}, {" 5/2002", "40", "Java Message Service: Einführung und Praxis"}, {" 6/2002", "35", "Java meets Office"}, {" 7/2002", "35", "Java Datenbank API"}, {" 8/2002", "34", "Eigene JSP TagLibs entwickeln"}, {" 9/2002", "39", "Suchmaschinen entwickeln mit Apache und Lucene"} public static final String[] COLHEADS = { "Ausgabe", "Seite", "Titelthema" }; } Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2



1.3 Konfiguration (1)

• Wichtigste Methoden

Methode	Beschreibung
void setRowHeight(int newHeight)	Setzt Gesamthöhe der Zeile
void setRowMargin(int rowMargin)	Setzt freibleibender Raum oberhalb und unterhalb der Zeile
void setIntercellSpacing(Dimension newSpacing)	Setzt horizontalen Rand zwischen den Zellen
void setShowGrid(boolean b)	Schaltet alle Begrenzungslinien Ein/Aus
void setShowHorizontalLines(boolean b)	Schaltet horizontale Begrenzungslinien Ein/Aus
void setShowVerticalLines(boolean b)	Schaltet vertikale Begrenzungslinien Ein/Aus
void setGridColor(Color newColor)	Setzt Farbe der Begrenzungslinien
void setSelectionForeground(Color selectionForeground)	Setzt Vordergrundfarbe für selektierte Zellen
void setSelectionBackground(Color selectionBackground)	Setzt Hintergrundfarbe für selektierte Zellen
void setAutoResizeMode(int mode)	Bestimmt Verhalten der Tabelle bei Grössenänderungen

LUZERN

Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2

07 JTable Folie 7

1.3 Konfiguration (2)

• Konstanten für setAutoResizeMode

Methode	Beschreibung
AUTO_RESIZE_OFF	Keine automatische Grössenanpassung der übrigen Spalten. Wurde die Tabelle in JScrollPane verpackt, bekommt sie nötigenfalls einen horizontalen Schieberegler.
AUTO_RESIZE_LAST_COLUMN	Die letzte Spalte wird zum Grössenausgleich verwendet. Dadurch reduziert sich der Platz für die letzte Spalte, wenn eine andere Spalte vergrössert wird, und er erhöht sich, wenn sie verkleinert wird.
AUTO_RESIZE_NEXT_COLUMN	Die rechts neben der modifizierten Spalte liegende Spalte wird zum Grössenausgleich verwendet.
AUTO_RESIZE_SUBSEQUENT_COLUMNS	Die Grössenänderung wird gleichmässig auf alle nachfolgenden Spalten verteilt.
AUTO_RESIZE_ALL_COLUMNS	Die Größenänderung wird auf alle Spalten der Tabelle verteilt.

Abteilung Informatik, JFC/Swing K07 JTable
LUZERN 2004 © Diego Schmidlin V2.2 Folie 8

1.4 Selektion (1)

• Selektionsmodi

Methode	Beschreibung
void setRowSelectionAllowed(boolean flag)	Erlaubt Selektion von Zeilen
<pre>void setColumnSelectionAllowed(boolean flag)</pre>	Erlaubt Selektion von Spalten
<pre>void setSelectionMode(int selectionMode)</pre>	Setzt Modus für Zellenselektion (siehe JSDK Doku)
void setCellSelectionEnabled(boolean flag)	Erlaubt Selektion von Zellen

• Abfrage der Selektion

Methode	Beschreibung
<pre>int getSelectedRow()</pre>	Gibt selektierte Zeile zurück
<pre>int getSelectedColumn()</pre>	Gibt selektierte Spalte zurück
<pre>int[] getSelectedRows()</pre>	Gibt selektierte Zeilen in Array zurück
<pre>int[] getSelectedColumns()</pre>	Gibt selektierte Spalten in Array zurück

Abteilung Informatik, JFC/Swing K07 JTable 2004 © Diego Schmidlin V2.2 Folie 9

1.4 Selektion (2)

• Verändern der Selektion

Methode	Beschreibung
<pre>void selectAll()</pre>	Selektiert gesamte Tabelle
<pre>void setRowSelectionInterval(int index0,</pre>	Mehrere zusammenhängende Zeilen können selektiert werden
<pre>void addRowSelectionInterval(int index0,</pre>	Nimmt mehrere zusammenhängende Zeilen zur Selektion hinzu
<pre>void removeRowSelectionInterval(int index0,</pre>	Entfernt mehrere zusammenhängende Zeilen von der Selektion
<pre>void setColumnSelectionInterval(int index0,</pre>	Mehrere zusammenhängende Spalten können selektiert werden
<pre>void addColumnSelectionInterval(int index0,</pre>	Nimmt mehrere zusammenhängende Spalten zur Selektion hinzu
<pre>void removeColumnSelectionInterval(int index0,</pre>	Entfernt mehrere zusammenhängende Spalten von der Selektion

Abteilung Informatik, JFC/Swing K07 JTable 2004 © Diego Schmidlin V2.2 Folie 10

1.5 Zugriff auf Inhalt

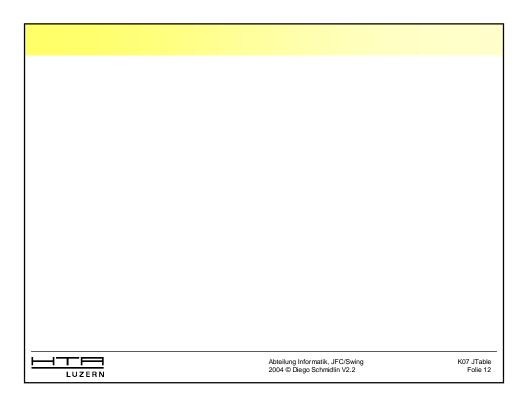
Zugriff auf Daten

Methode	Beschreibung
<pre>int getRowCount()</pre>	Gibt Anzahl Zeilen zurück
<pre>int getColumnCount()</pre>	Gibt Anzahl Spalten zurück
Object getValueAt(int row, int column)	Gibt Objekt an Position (raw, column) zurück
<pre>void setValueAt(Object aValue,</pre>	Setzt Objekt an Position (raw, column)

• Editieren von Daten

Methode	Beschreibung
boolean isEditing()	Gibt true zurück, wenn eine Zelle der Tabellen gerade editiert wird
<pre>int getEditingRow()</pre>	Gibt Zeile zurück, in der eine Zelle der Tabelle gerade editiert wird
<pre>int getEditingColumn()</pre>	Gibt Spalte zurück, in der eine Zelle der Tabelle gerade editiert wird
boolean editCellAt(int row, int column)	Programm erzwingt das Editieren einer Zelle

Abteilung Informatik, JFC/Swing K07 JTable 2004 © Diego Schmidlin V2.2 Folie 11



2.1 Tabellenmodell

- Tabellenmodell
 - Verantwortlich für die Beschaffung der Daten



- Default Tabellenmodell
 - DefaultTableModel
- Verwenden von eigenen *Tabellenmodellen*
 - Daten komplex strukturiert sind
 - Ein Array die Daten nicht verwalten kann
 - Daten an externe Quellen gebunden sind
- Erzeugen von eigenen Tabellenmodellen
 - Durch Implementation des Interface TableModel
 - TableModel aus javax.swing.table



Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2 K07 JTable Folie 13

2.1.1 TabelModel

• Die wichtigsten Methoden des Interface

Methode	Beschreibung
int getRowCount()	Gibt Anzahl Zeilen im Modell zurück
int getColumnCount()	Gibt Anzahl Spalten im Modell zurück
String getColumnName(int columnIndex)	Liefert Spaltenüberschrift
Class getColumnClass(int columnIndex)	Liefert Datentyp der Spalte zurück
boolean isCellEditable(int rowIndex, int columnIndex)	Gibt true zurück, wenn Wert editiert werden kann
Object getValueAt(int rowIndex, int columnIndex)	Erfragt einen bestimmten Wert im Modell
void setValueAt(Object aValue, int rowIndex, int columnIndex)	Schreibt einen geänderten Wert in das Modell zurück
void addTableModelListener(TableModelListener 1)	Registriert einen Listener beim Modell, der über Änderungen am Modell unterrichtet wird
void removeTableModelListener(TableModelListener 1)	Deregistriert einen Listener vom Modell

Abteilung Informatik, JFC/Swin 2004 © Diego Schmidlin V2.2

2.1.2 TabelModel: Beispiel (1)

- Es wird ein Modell entworfen, welches eine sehr grosse Tabelle repräsentiert (1000 * 1000 Elemente)
- Es sollen nur die tatsächlich belegten Elemente gespeichert werden
- Für die Verwaltung wird eine Hashtable verwendet
- Als Schlüssel wird ein String mit den Zellen-Koordinaten verwendet
- Der Zugriff auf die Daten im Modell erfolgt dann über diesen Schlüssel
- Um das Modell nicht von Grund auf zu implementieren, wird die abstrakte Klasse AbstractTableModel verwendet



Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2

K07 JTable Folie 15

2.1.2 TabelModel: Beispiel (2)

```
import java.util.*;
import javax.swing.table.*;
public class JTableDemo2Data extends AbstractTableModel {
    private int size;
    private Hashtable data;
    //Konstruktor
    public JTableDemo2Data(int size) {
        this.size = size:
        this.data = new Hashtable();
    //Methoden für das TableModel-Interface
    public int getRowCount() {
        return size;
    public int getColumnCount() {
        return size;
                                              Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2
      LUZERN
```

```
public String getColumnName(int columnIndex) {
    return "C" + columnIndex;
}

public Class getColumnClass(int columnIndex) {
    return String.class;
}

public boolean isCellEditable(int rowIndex, int columnIndex) {
    return rowIndex < size && columnIndex < size;
}

public Object getValueAt(int rowIndex, int columnIndex) {
    String key = "[" + rowIndex + "," + columnIndex + "]";
    String value = (String)data.get(key);
    return value == null ? "-" : value;
}

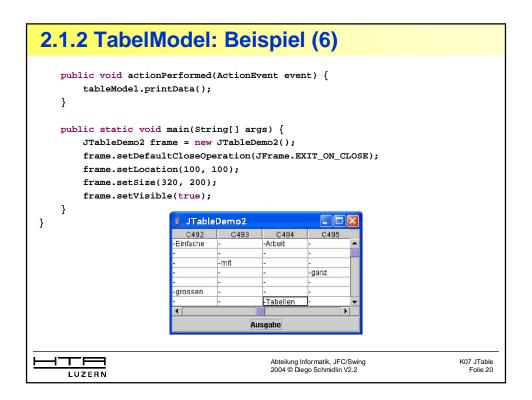
Abtellung Informatik, JFC/Swing 2004 © Dego Schmidlin V2.2</pre>

K07.JTable Foile 17
```

```
2.1.2 TabelModel: Beispiel (4)
    public void setValueAt(Object aValue, int rowIndex, int columnIndex) {
        String key = "[" + rowIndex + "," + columnIndex + "]";
        String value = (String)aValue;
        if (value.length() <= 0) {</pre>
            data.remove(key);
        } else {
            data.put(key, value);
        }
    //Zusätzliche Methoden
    public void printData() {
        Enumeration e = data.keys();
        while (e.hasMoreElements()) {
            String key = (String)e.nextElement();
            System.out.println("At " + key + ": " + (String)data.get(key));
    }
}
                                            Abteilung Informatik, JFC/Swing
2004 © Diego Schmidlin V2.2
     LUZERN
```

Hochschule Technik+Architektur Luzern Abteilung Informatik, Swing/JFC

```
2.1.2 TabelModel: Beispiel (5)
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class JTableDemo2 extends JFrame implements ActionListener {
   JTable table:
   JTableDemo2Data tableModel;
   public JTableDemo2() {
       super("JTableDemo2");
        tableModel = new JTableDemo2Data(1000);
        table = new JTable(tableModel, null);
        table.setAutoResizeMode(JTable.AUTO RESIZE OFF);
        table.setSelectionMode(ListSelectionModel.SINGLE_INTERVAL_SELECTION);
        table.setCellSelectionEnabled(true);
        Container cp = getContentPane();
        cp.add(new JScrollPane(table), "Center");
        JButton button = new JButton("Ausgabe");
        button.addActionListener(this);
        cp.add(button, "South");
                                           Abteilung Informatik, JFC/Swing
2004 © Diego Schmidlin V2.2
                                                                               K07 JTable
Folie 19
   \top
```



2.2 Spaltenmodell

- Spaltenmodell
 - Verantwortlich für die Eigenschaften der Spalten
- Default Spaltenmodell
 - DefaultTableColumnModel
- Verwenden von eigenen Spaltenmodellen
 - Kontrolle von Spalten bezüglich:
 - Auswahl, Hinzufügen, Entfernen, Bewegen
 - Editoren, Darstellung (Renderer)
- Erzeugen von eigenen Spaltenmodellen
 - Durch Implementation des Interface TableColumnModel
 - TableColumnModel aus javax.swing.table



Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2 K07 JTable Folie 21

2.2.1 TableColumnModel

• Die wichtigsten Methoden des Interface

Methode	Beschreibung
<pre>int getColumnCount()</pre>	Gibt Anzahl Spalten im Modell zurück
void addColumn(TableColumn aColumn)	Fügt eine Spalte am Ende des Spalten-Array hinzu
void removeColumn(TableColumn column)	Entfern die Spalte aus dem Spalten-Array
<pre>void moveColumn(int columnIndex, int newIndex)</pre>	Verschiebt Spalte an neue Position
boolean getColumnSelectionAllowed()	Gibt true zurück, wenn Spalte selektiert werden kann
void setColumnSelectionAllowed(boolean flag)	Setzt Selektierbarkeit der Spalte
void addColumnModelListener(TableColumnModelListener x)	Registriert einen Listener beim Modell, der über Änderungen am Modell unterrichtet wird
void removeColumnModelListener(TableColumnModelListener x)	Deregistriert einen Listener vom Modell

LUZERN

Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2

2.2.2 TableColumnModel: Beispiel (1)

- Es wird ein Modell entworfen, welches den Spalten eine Default Breite vorgibt
- Jede Spalte der Tabelle kann individuell angepasst werden
- Um das Modell nicht von Grund auf zu implementieren, wird die Klasse DefaultTableColumnModel verwendet
- Beim Aufruf des Konstruktors wird neben dem Spaltenmodell auch ein Tabellenmodell benötigt
 - Dieses leiten in einer lokale Klasse von AbstractTableModel ab



Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2

K07 JTable Folie 23

2.2.2 TableColumnModel: Beispiel (2)

```
import java.awt.*;
import javax.swing.*;
import javax.swing.table.*;

public class JTableDemo3 extends JFrame implements JTableDemo3Data
{
    public JTableDemo3()
    {
        super("JTableDemo3");

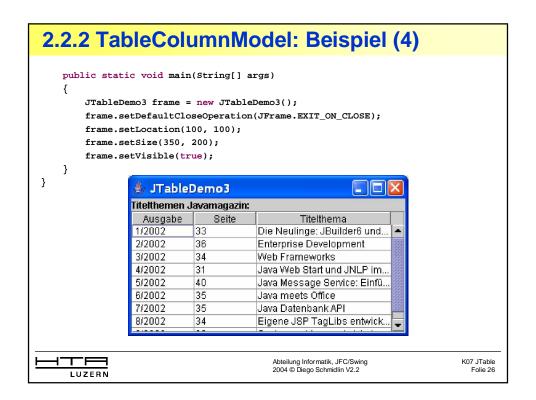
        //Spaltenmodell erzeugen
        DefaultTableColumnModel cm = new DefaultTableColumnModel();
        for (int i = 0; i < COLHEADS.length; ++i)
        {
            TableColumn col = new TableColumn(i, i == 2 ? 150 : 60);
            col.setHeaderValue(COLHEADS[i]);
            cm.addColumn(col);
        }
}</pre>
```

LUZERN

Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2

Hochschule Technik+Architektur Luzern Abteilung Informatik, Swing/JFC

```
2.2.2 TableColumnModel: Beispiel (3)
        //Tabellenmodell erzeugen
        TableModel tm = new AbstractTableModel()
            public int getRowCount() return DATA.length;
            public int getColumnCount() return DATA[0].length;
            public Object getValueAt(int row, int column)
                return DATA[row][column];
        };
        //Tabelle erzeugen und ContentPane füllen
       JTable table = new JTable(tm. cm):
        Container cp = getContentPane();
       cp.add(new JLabel("Titelthemen Javamagazin:"), "North");
        cp.add(new JScrollPane(table), "Center");
  Abteilung Informatik, JFC/Swing
2004 © Diego Schmidlin V2.2
                                                                              K07 JTable
Folie 25
```



2.3 Renderer

- Rendering
 - Vorgang, wie Zellen dargestellt werden
- Default Renderer
 - DefaultTableCellRenderer
- Verwenden von eigenen Renderer
 - Kontrolle von *Darstellung* der Zellendaten
 - · Schriftart, Farbe, Formatierung, Grafik
- Erzeugen von eigenen Renderer
 - Durch Implementation des Interface TableCellRenderer
 - TableCellRenderer aus javax.swing.table



Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2 K07 JTable Folie 27

2.3.1 TableCellRenderer

• Die einzige Methoden im Interface

Methode	Beschreibung
<pre>public Component getTableCellRendererComponent(</pre>	Referenz auf JTable, welche den Renderer aufruft Der Wert, der vom Renderer dargestellt werden soll true, wenn die Zelle selektiert ist true, wenn die Zelle den Fokus erhält Zeilen-Index Spalten-Index

- Arbeitet als Factory-Methode
- Wird immer dann aufgerufen, wenn zur Darstellung einer Zelle ein Renderer benötigt wird

LUZERN

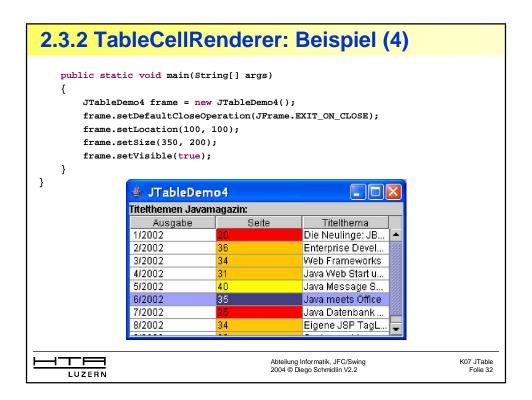
Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2

```
2.3.2 TableCellRenderer: Beispiel (1)
import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.table.*;
public class JTableDemo4CellRenderer implements TableCellRenderer
   private Color lightBlue = new Color(160, 160, 255);
   private Color darkBlue = new Color(64, 64, 128);
   public Component getTableCellRendererComponent(JTable table, Object value,
       boolean isSelected, boolean hasFocus, int row, int column)
        //Label erzeugen
       JLabel label = new JLabel((String)value);
       label.setOpaque(true):
        Border b = BorderFactory.createEmptyBorder(1, 1, 1, 1);
        label.setBorder(b);
        label.setFont(table.getFont());
        label.setForeground(table.getForeground());
        label.setBackground(table.getBackground());
  Abteilung Informatik, JFC/Swing
2004 © Diego Schmidlin V2.2
                                                                             K07 JTable
Folie 29
```

2.3.2 TableCellRenderer: Beispiel(2)

```
if (hasFocus) {
            label.setBackground(darkBlue);
            label.setForeground(Color.white);
        } else if (isSelected) {
            label.setBackground(lightBlue);
        } else {
            //Angezeigte Spalte in Modellspalte umwandeln
            column = table.convertColumnIndexToModel(column);
            if (column == 1) {
                int numpages = Integer.parseInt((String)value);
                if (numpages < 30) {</pre>
                    label.setBackground(Color.red);
                } else if (numpages < 40) {
                    label.setBackground(Color.orange);
                } else {
                    label.setBackground(Color.yellow);
            }
        }
        return label;
}
     LUZERN
```

Hochschule Technik+Architektur Luzern Abteilung Informatik, Swing/JFC



Zusammenfassung

- Eines der wichtigsten Dialogelemente in anspruchsvollen Programmen ist die *Tabelle*
- Mit JTable haben sie eine leistungsfähige Klasse für die tabellarische Darstellung von Daten
- Mit dem Tabellenmodel TableModel können die Daten einer JTable einfach verwaltet werden
- Mit dem Spaltenmodel TableColumnModel können die Eigenschaften der Spalten kontrolliert werden
- Mit der Klasse TableCellRenderer kann das Erscheinungsbild von Zellen verändert werden



Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2

K07 JTable Folie 33

@ Übungsaufgabe U07.1

· Gehen Sie im Internet auf die Seite

http://java.sun.com/docs/books/tutorial/uiswing/components/table.html

- Lesen Sie das Kapitel "How to Use Tables"
- Laden Sie die Beispiele herunter und spielen Sie mit den Möglichkeiten von Tabellen
- Versuchen Sie für den Address Manager eine Tabelleansicht mit Adressen zu implementieren

LUZERN

Abteilung Informatik, JFC/Swing 2004 © Diego Schmidlin V2.2