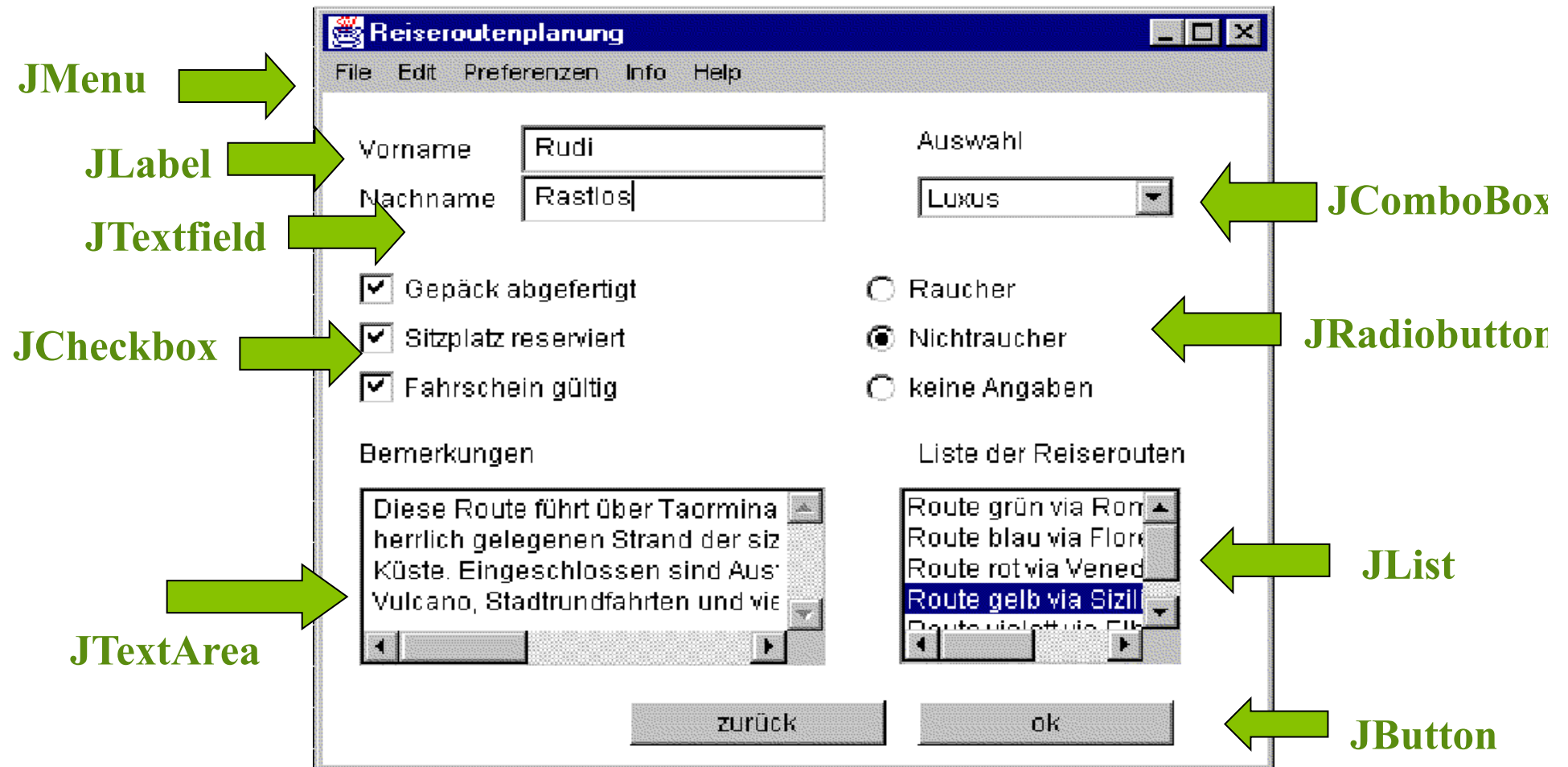


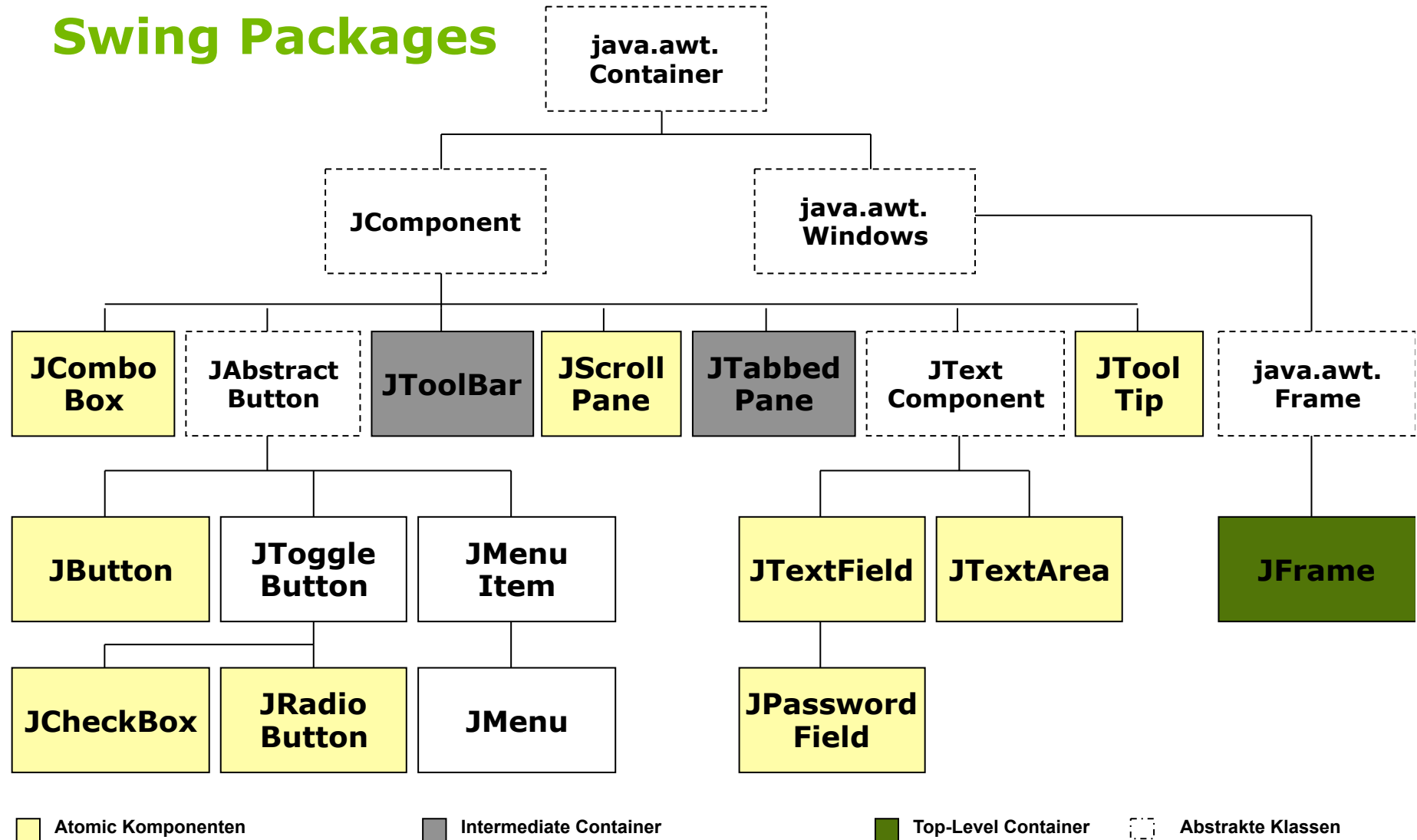
SWING

Übersicht der Komponenten

GUI (Grafical User Interface)



Swing Packages



JButton



- Ein JButton implementiert eine drückbare Schaltfläche.
- kann einen Text und/oder eine Grafik beinhalten
- Schaltfläche für auslösende Ereignisse (Funktionen oder Informationen)
- springt in seinen Ursprungsstatus zurück, nachdem die Maus ihn betätigt hat



JButton



Konstruktoeren

- *JButton();*
Erstellt Button ohne Text oder Grafik.
- *JButton(Icon icon);*
Erstellt Button mit angegebener Grafik.
- *JButton(String text);*
Erstellt Button mit angegebenem Text.
- *JButton(String text, Icon icon);*
Erstellt Button mit angegebenem Text und Grafik.

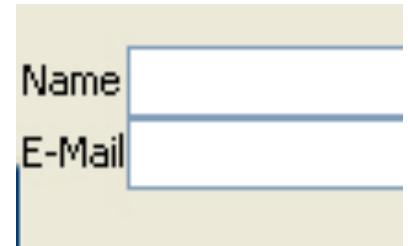
JButton



wichtige Methoden

- *addActionListener(ActionListener l);*
 - fügt den Listener für die Komponente hinzu
- *setBorder(Border border);*
 - setzt den Rahmen der Komponente
- *setBounds(int x,int y,int width,int height);*
 - setzt Position & Abmessungen der Komponente auf übergebene Werte

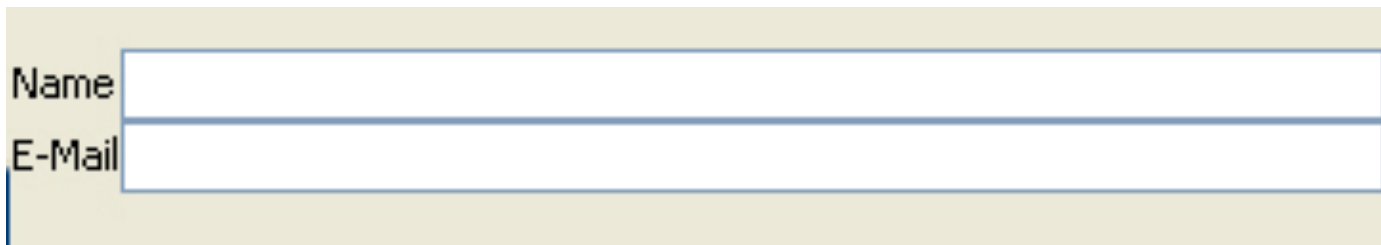
JTextField



Name

E-Mail

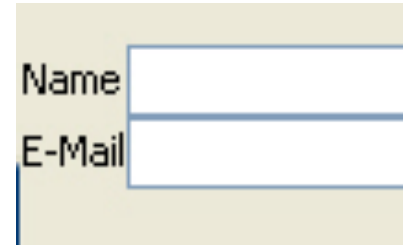
- ein einfaches, einzelzeiliges Textfeld als Texteingabekomponente
- direkt abgeleitet → **JPasswordField** als Textfeld für die Eingabe eines Passwortes. Keine Darstellung des eigentlich eingegebenen Zeichens, sondern eines einstellbaren Char (default → „ * “)



Name

E-Mail

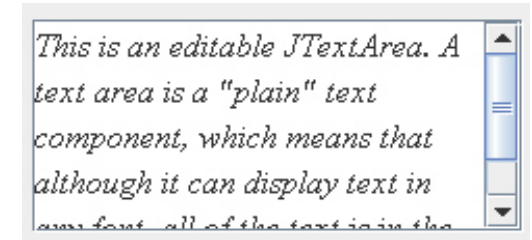
JTextField



Konstrukturen

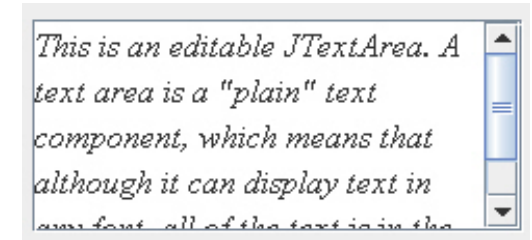
- *JTextField ()*;
Erzeugt ein neues Textfeld ohne Werte.
- *JTextField (String text)*;
Erzeugt ein Textfeld mit dem angegebenen Text.
- *JTextField (String text, int columns)*;
Erzeugt zusätzlich Anzahl der Spalten
- *JPasswordField()*;
Erzeugt ein Passwortfeld.
- *JPasswordField(String text)*;
Erzeugt ein Passwortfeld mit „text“ als Passwort.

JTextArea



- Eine mehrzeilige Textkomponente, über welche reiner Text editiert werden kann.
- Typischerweise wird **JTextArea** in eine **JScrollPane** integriert.
- JScrollPane liefert eine scrollbare Sicht auf die integrierte Komponente.
- Bei der JTextArea können automatische Zeilenumbrüche eingestellt werden.

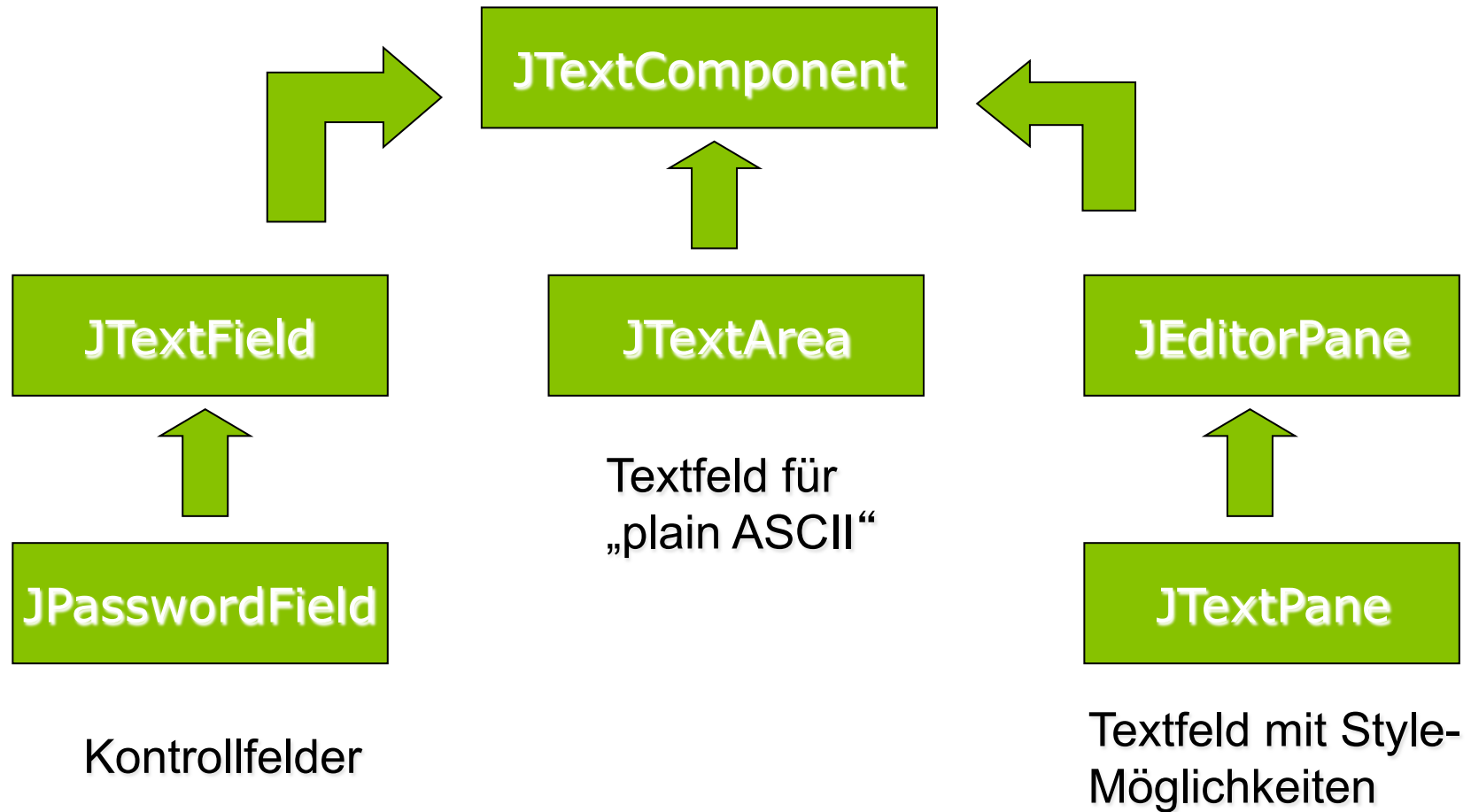
JTextArea



Konstruktoren

- *JTextArea();*
Erzeugt eine JTextArea ohne Werte.
- *JTextArea (String text);*
Erzeugt eine JTextArea mit Texteintrag.
- *JTextArea (int rows, int columns);*
Erzeugt JTextArea mit den angegebenen Werten.
- *JScrollPane (Component comp);*
Erstellt ein JScrollPane für die angegebene Komponente.

JTextComponent



JCheckBox



- Über eine JCheckBox kann der Benutzer einen Wert selektieren oder deselektieren.
- Eine JCheckBox verhält sich wie ein JToggleButton, d.h. er kann zwei Zustände haben: gedrückt oder nicht.
- Besitzt im Gegensatz zum JRadioButton normalerweise keine Abhängigkeiten (ButtonGroup) zu anderen Buttons.

JCheckBox



Konstruktoren

- *JCheckBox();*
Erstellt eine unselektierte JCheckBox.
- *JCheckBox (String text, boolean selected);*
Erstellt selektierte JCheckBox mit dem angegebenen Text, wenn boolean „true“ ist.
- *JCheckBox (Icon icon, boolean selected);*
Erstellt eine JCheckBox mit dem angegebenen Bild.
- *JCheckBox (String text, Icon icon, boolean selected);*
Kombination aus den beiden vorherigen Konstruktoren, default-Wert für boolean ist „false“ = unselektiert.

JCheckBox



wichtige Methoden

- *addItemListener (ItemListener l);*
fragt den Schaltzustand der JCheckBox ab und ermöglicht die Reaktion auf Schaltereignisse
- *setBackground (Color color);*
setzt die Hintergrundfarbe, JComponent-Methode
- *setBounds (int x,int y,int width,int height);*
setzt Position & Abmessungen der Komponente auf übergebene Werte

JRadioButton



- stellt Button dar, der wahlweise an/abgeschaltet werden kann
- Auswahl nur eines Button möglich
- in sinnvollem Kontext gruppiert



JRadioButton

Konstrukturen

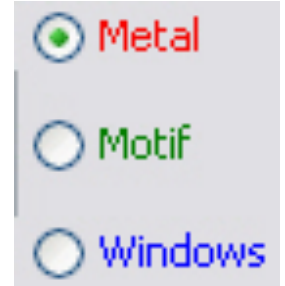
- *JRadioButton();*
erzeugt leeren RadioButton
- *JRadioButton(icon);*
erzeugt RadioButton mit Bild
- *JRadioButton(String);*
erzeugt RadioButton mit Text
- *JRadioButton(String, boolean);*
erzeugt RadioButton mit Text und Auswahlstatus

JRadioButton



Besonderheit von JRadioButton

- Zusammenfassung in **ButtonGroup**, da sonst mehr als 1 Eintrag gewählt werden kann
- Ereignisbehandlung mit *Interface Buttonmodel*
- Eventverarbeitung mit *getActionCommand()*



JRadioButton

Methoden

- *add(AbstractButton)*
RadioButton zu Gruppe hinzufügen
- *getSelection()*
liefert gewählten Eintrag
- *setSelection()*
setzt Eintrag als gewählt

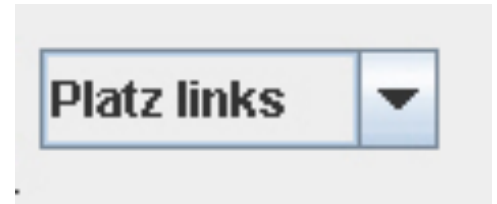
JRadioButton



Eventhandling

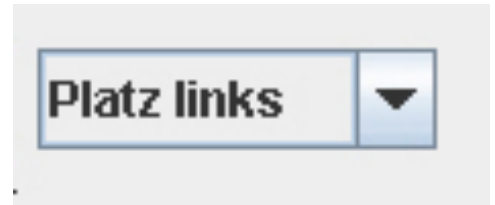
- `itemStateChanged (ItemEvent)`
beim Wechsel eines Radiobutton
- `actionPerformed (ActionEvent)`
zur späteren Auswertung des Knopfdrucks

JComboBox



- Kombination zwischen Textfeld und Liste
- Liste erst beim Auswählen sichtbar
- Nutzer soll definierten Wert auswählen

JComboBox



Konstruktoren

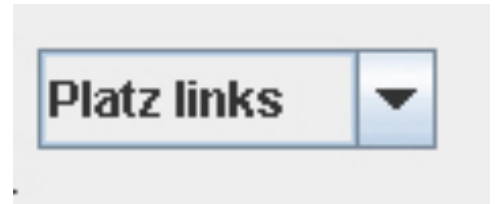
- *JComboBox();*
erzeugt leere ComboBox
- *JComboBox(Object[] Item);*
erzeugt ComboBox und füllt diese mit den Werten aus dem Array

JComboBox

Methoden

- *addItem(Object);*
Eintrag hinzufügen
- *removeItemAt(int);*
löscht Eintrag an Position
- *removeAllItems();*
löscht alle Einträge
- *Object getSelectedItem();*
liefert gewählten Eintrag

JComboBox



Besonderheit

Elemente der Liste zur Laufzeit dynamisch

Eventhandling

actionPerformed (ActionListener)

bei Selektion von Hand bzw. Texteingabe

itemStateChanged (ItemListener)

bei Wechsel von Element

JMenu



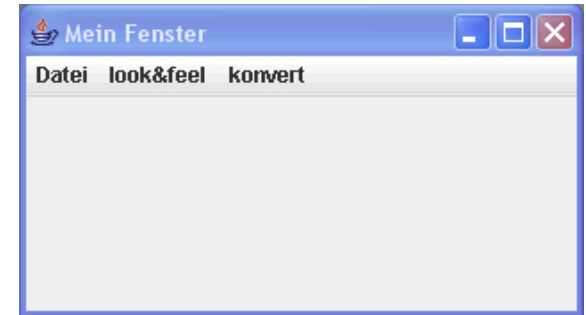
- Umsetzung von Menüs im Frame
- enthält Menüeinträge
- kann beliebig verschachtelt werden
- Tastenkürzel können hinterlegt werden

Menüklassen

Basisklassen

Verwendung von 3 Klassen erforderlich

- *JMenuBar*
Menüleiste auf Fenster
- *JMenu*
repräsentiert Menü selbst
- *JMenuItem*
repräsentiert Menüeinträge



JMenuBar

Konstrukturen

- *JMenuBar();*
erzeugt Menüleiste

Methoden

- *add(new JMenu("..."));*
fügt Menü zu Menüleiste hinzu
- *setJMenuBar(new MenuLeiste());*
legt Menüleiste auf das Fenster



JMenu

Konstruktor

- *JMenu(String);*
erzeugt Menü

Methoden

- *add(String);*
fügt Eintrag in Menü hinzu
- *add(MenuItem);*
fügt Untermenü zum Untermenü hinzu
- *addSeparator();*
fügt optische Trennlinie ein



JMenuItem

Konstruktor

- *JMenuItem(String);*
erzeugt Menüeintrag

Methoden

- *setMnemonic(char);*
setzt Tastaturkürzel Alt + Parameter
- *setEnabled(boolean);*
aktiviert bzw. deaktiviert Eintrag
- *addActionListener(ActionListener)*
implementiert ActionListener



JTabbedPane



- spezielles Panel
- Dialog mit einer Reihe von Registern
- können über Register ausgewählt werden

JTabbedPane

Konstruktor

- *JTabbedPane()*;
erzeugt leeres Register
- *JTabbedPane(int)*;
erzeugt Register mit Ausrichtung
Standardausrichtung: *JTabbedPane.TOP*



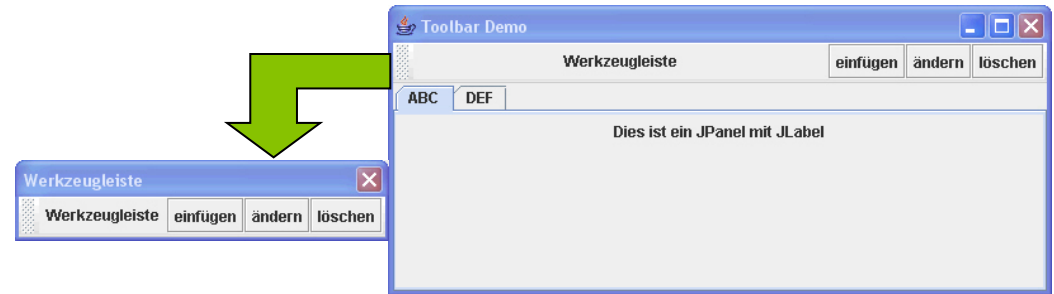
JTabbedPane

Methoden

- *addTab(String, Component);*
fügt Registerkarte hinzu
- *getTabCount();*
liefert Anzahl der Register
- *getSelectedComponent();*
gibt gewählte Komponente zurück

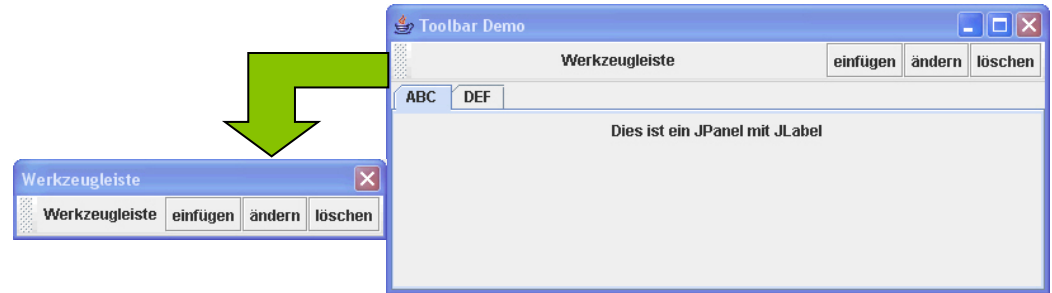


JToolBar



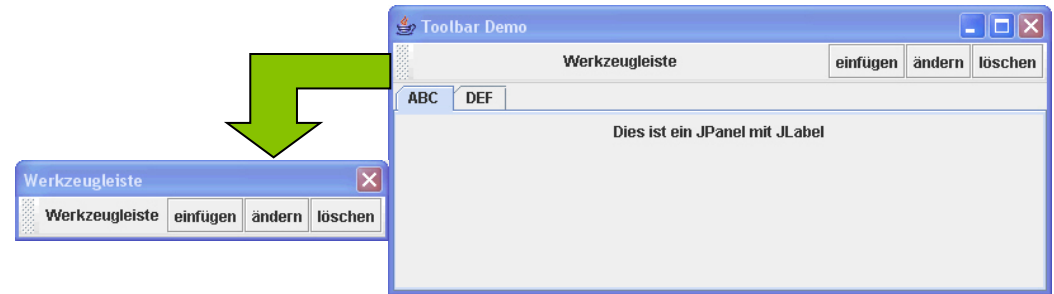
- frei schwebender, dockbarer Panel als Container für verschiedene Elemente zur Steuerung von Anwendungen
- kann bei den meisten Varianten des Look-and-Feel verschoben werden, so dass die ToolBar in einem neuen Fenster erscheint

JToolBar Konstruktoren



- *JToolBar ();*
Erstellt eine neue horizontale Toolbar.
- *JToolBar (int orientation);*
Erstellt eine neue Toolbar mit horizontaler oder vertikaler Ausrichtung.
- *JToolBar (String name);*
Erstellt eine horizontale Toolbar mit dem Titel für das externe Fenster.
- *JToolBar (String name, int orientation);*
Kombination der beiden vorherigen Konstruktoren.

JToolBar

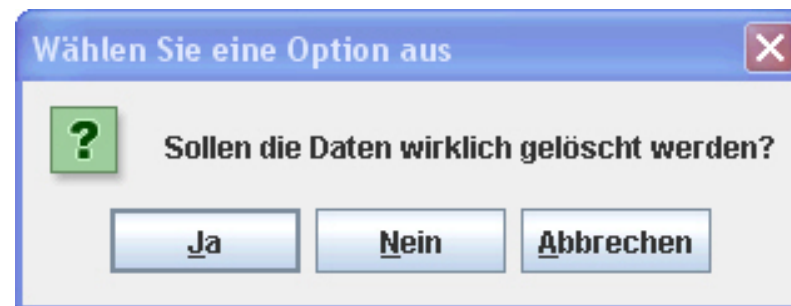


wichtige Methoden

- *add (Component);*
 - fügt Komponente ein
- *setFloatable (boolean b);*
 - mit Wert „true“ kann die Leiste verschoben werden
- *addSeparator();*
 - fügt Separator an oder zwischen Komponenten ein

JOptionPane

- spezielles Panel
- Dialog mit Schaltflächen und eigenen Icons
- Buttons schon aktionsfähig



JOptionPane



Methoden

- *JOptionPane.showConfirmDialog(fenster, message);*
zeigt Panel mit Auswahl- Button an
- *JOptionPane.showMessageDialog(fenster, message);*
zeigt Panel mit ok-Button an
- *JOptionPane.showInputDialog(komponente);*
zeigt Panel an, um Daten einzulesen

JOptionPane

Bsp: SourceCode

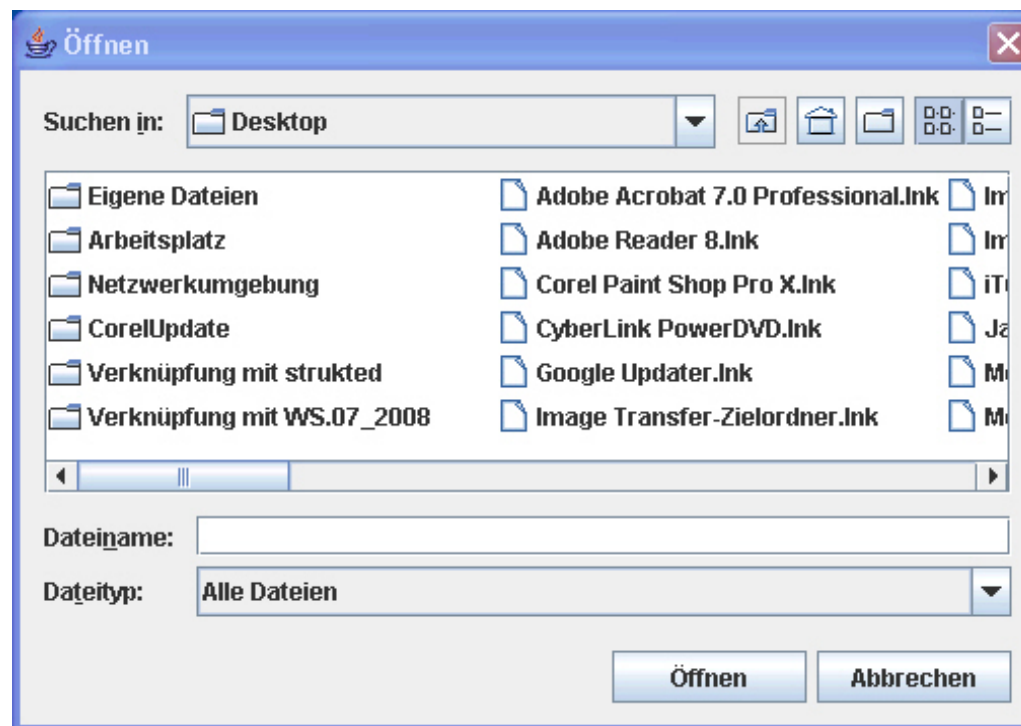
```
message = new String("Sollen die Daten wirklich gelöscht werden?");
int antwort = JOptionPane.showConfirmDialog(fenster, message);

if( antwort == JOptionPane.YES_OPTION){
    // Daten löschen
    message = new String("Daten wurden gelöscht");
    JOptionPane.showMessageDialog(fenster, message);
}
else if( antwort == JOptionPane.NO_OPTION){
    message = new String("Daten werden nicht gelöscht");
    JOptionPane.showMessageDialog(fenster, message);
}
else if( antwort == JOptionPane.CANCEL_OPTION){
    message = new String("Aktion abgebrochen");
    JOptionPane.showMessageDialog(fenster, message);
}
```



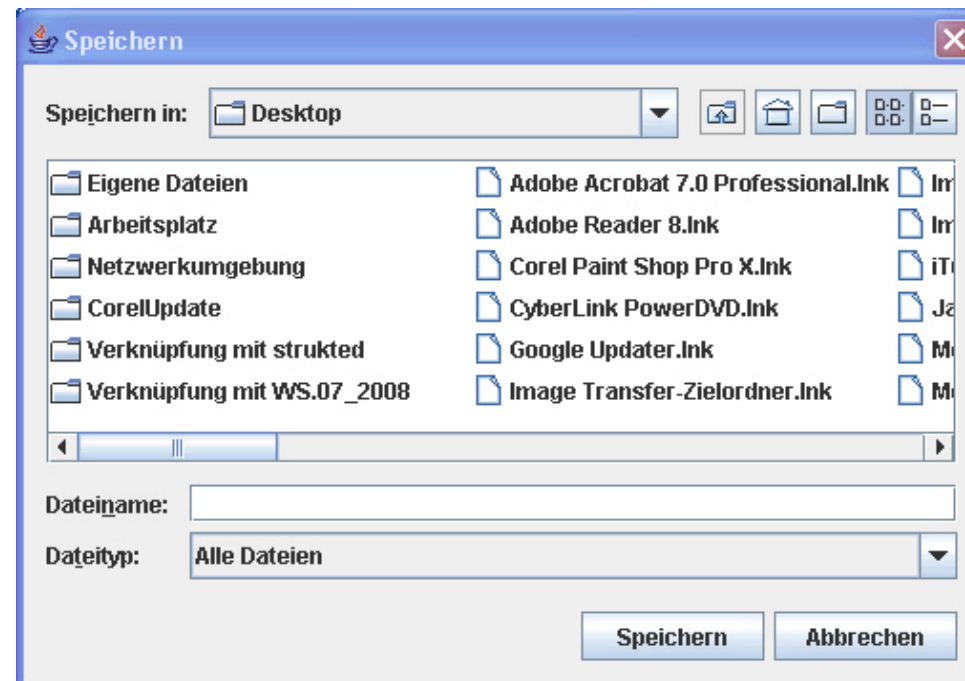
JFileChooser

- spezielles Panel mit „viel dahinter“, um Dateien zu öffnen. Fenster automatisch modal!



JFileChooser

- Auch zum Speichern von Dateien geeignet – Aufruf ähnlich.



JFileChooser

Bsp: SourceCode

```
File datei = null;
JFileChooser fc = new JFileChooser();

fc.setFileSelectionMode(JFileChooser.FILES_ONLY);

int returnVal = fc.showOpenDialog( fenster );
//int returnVal = fc.showSaveDialog( fenster );

if (returnVal == JFileChooser.APPROVE_OPTION) {
    datei = fc.getSelectedFile();
}

if(datei != null) {
    String dateiname = datei.getName(); // Dateinamen
    String dateipfad = datei.getPath();    // inkl. Name

    // Datei einlesen bzw Datei speichern
}
```