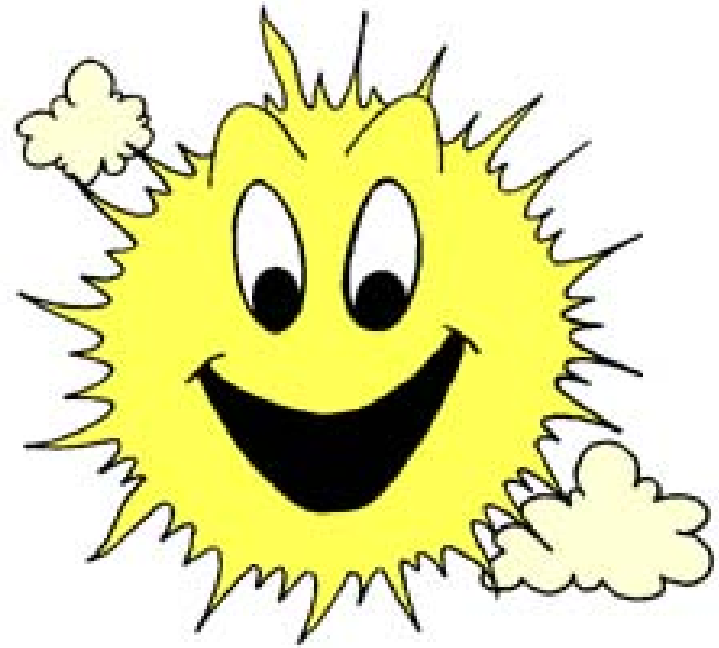
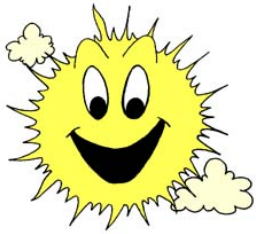
Wiederholung 8. Vorlesung



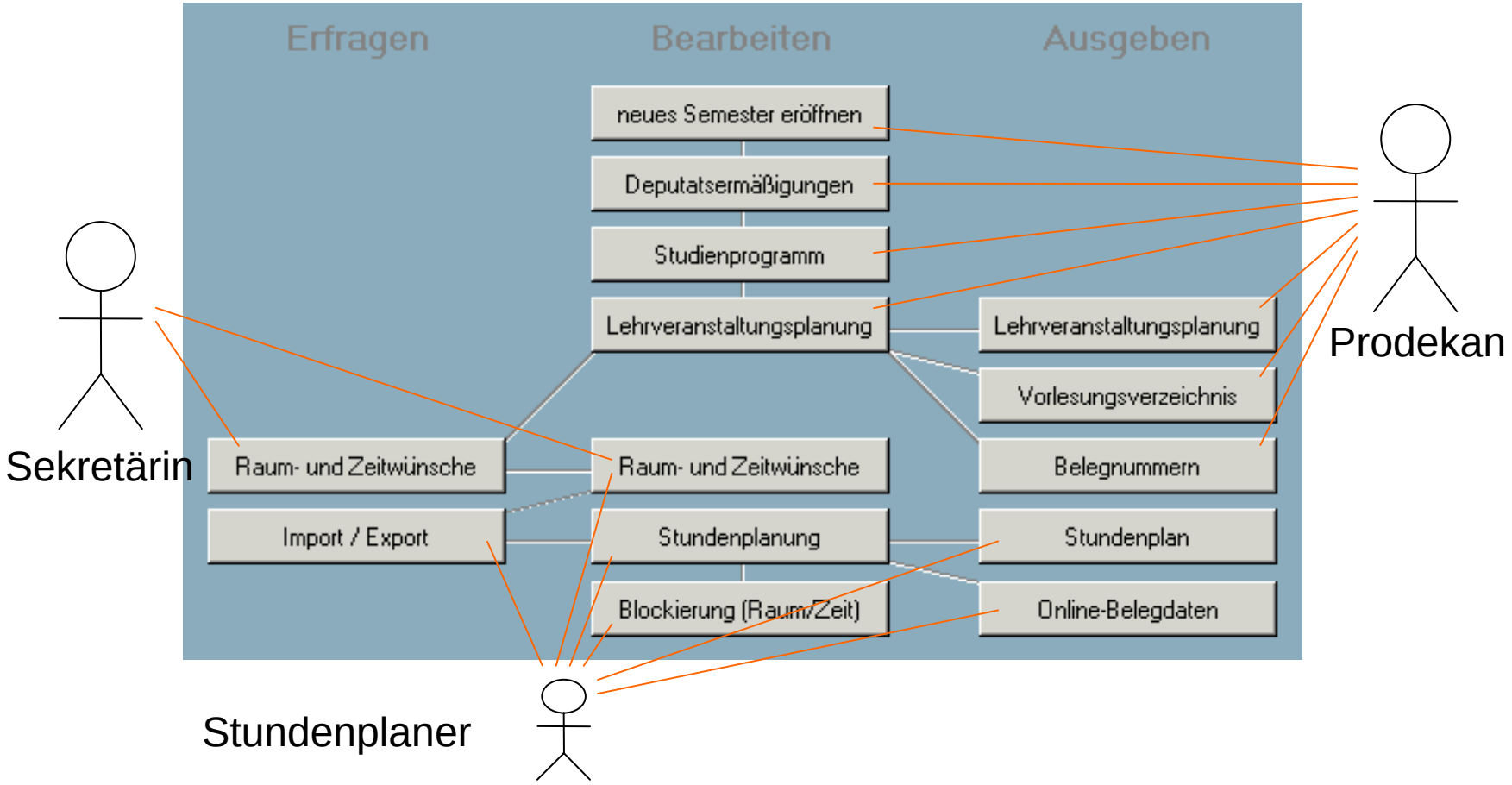
Ende der Wiederholung



7. Entwicklung von Anwendungen

- Anwendungsfälle(Use Cases)
- Workflow
- Masken

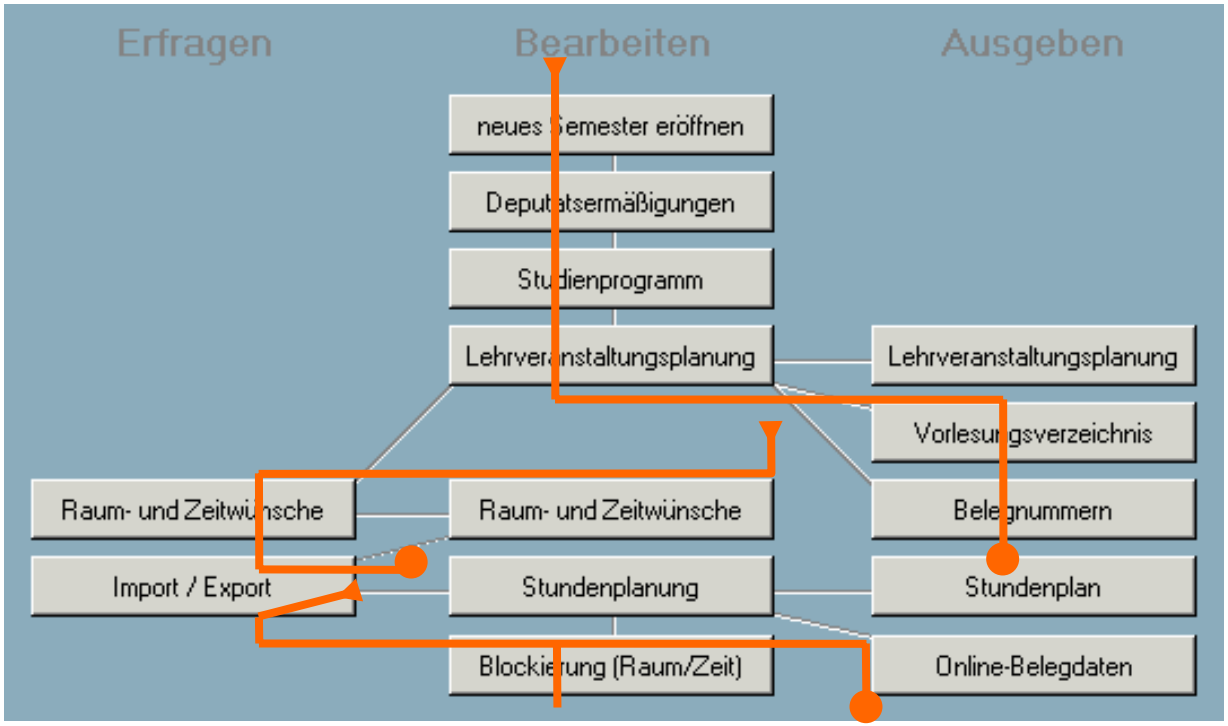
7. Entwicklung von Anwendungen



7. Entwicklung von Anwendungen: Masken

- Zu jedem **Use-Case** wird eine **Bildschirmmaske** entworfen
  - Titel des Screens meist gleich der Benennung des Use Case
  - In einen Screen gehören **nur Daten und Funktionen**, die **unmittelbar** mit **diesem Anwendungsfall** zu tun haben
- Gruppierung der Screens nach Benutzergruppen
  - besser: **Benutzer** durch Anmeldung **identifizieren**
  - nicht benötigte Use Cases ausblenden
  - Jeder Benutzer bekommt nur Screens seiner Gruppe zu sehen
- Das Hauptfenster enthält in geeigneter Form die Übersicht über die Funktionen (Use Cases)
  - ermöglicht Verzweigung dorthin

7. Workflow (High Level)



Sekretärin

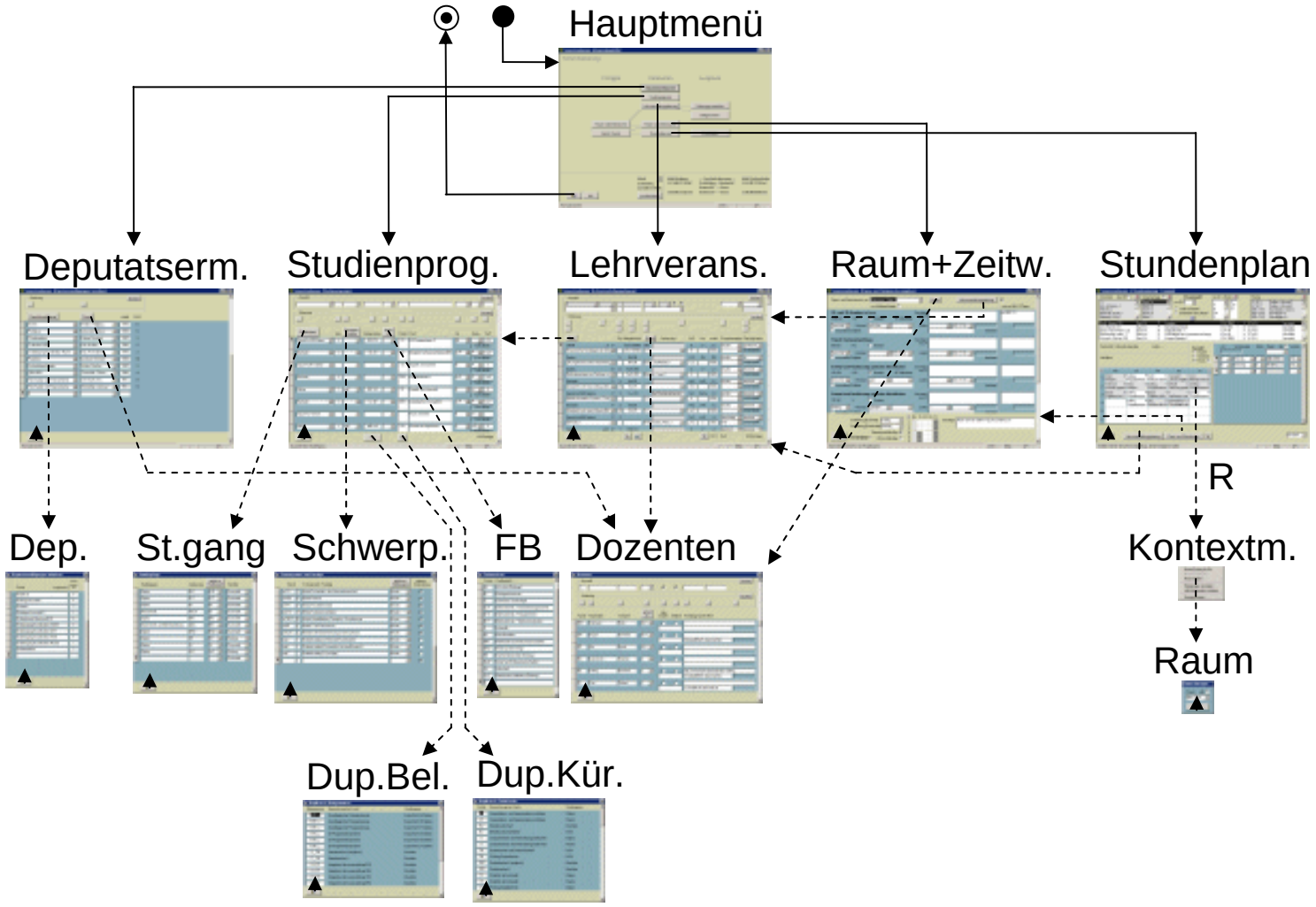
Prodekan

Stundenplaner

7. Szenarien beschreiben (Low Level)

- Ablauf hinter dem Anwendungsfall/Screen beschreiben
  - Verfeinerung des Anwendungsfalls
- verbale Beschreibung aus Sicht des Benutzers
  - jetzt wichtige systeminterne Objekte dabei berücksichtigen
  - systeminterne Objekte sollten dem mentalen Modell des Benutzers entsprechen
- formale Beschreibung mit Sequenzdiagrammen
  - Präzisierung der verbalen Beschreibung
  - Benutzerereignisse rufen Methoden der Fensterklasse auf

7. Navigationsübersicht



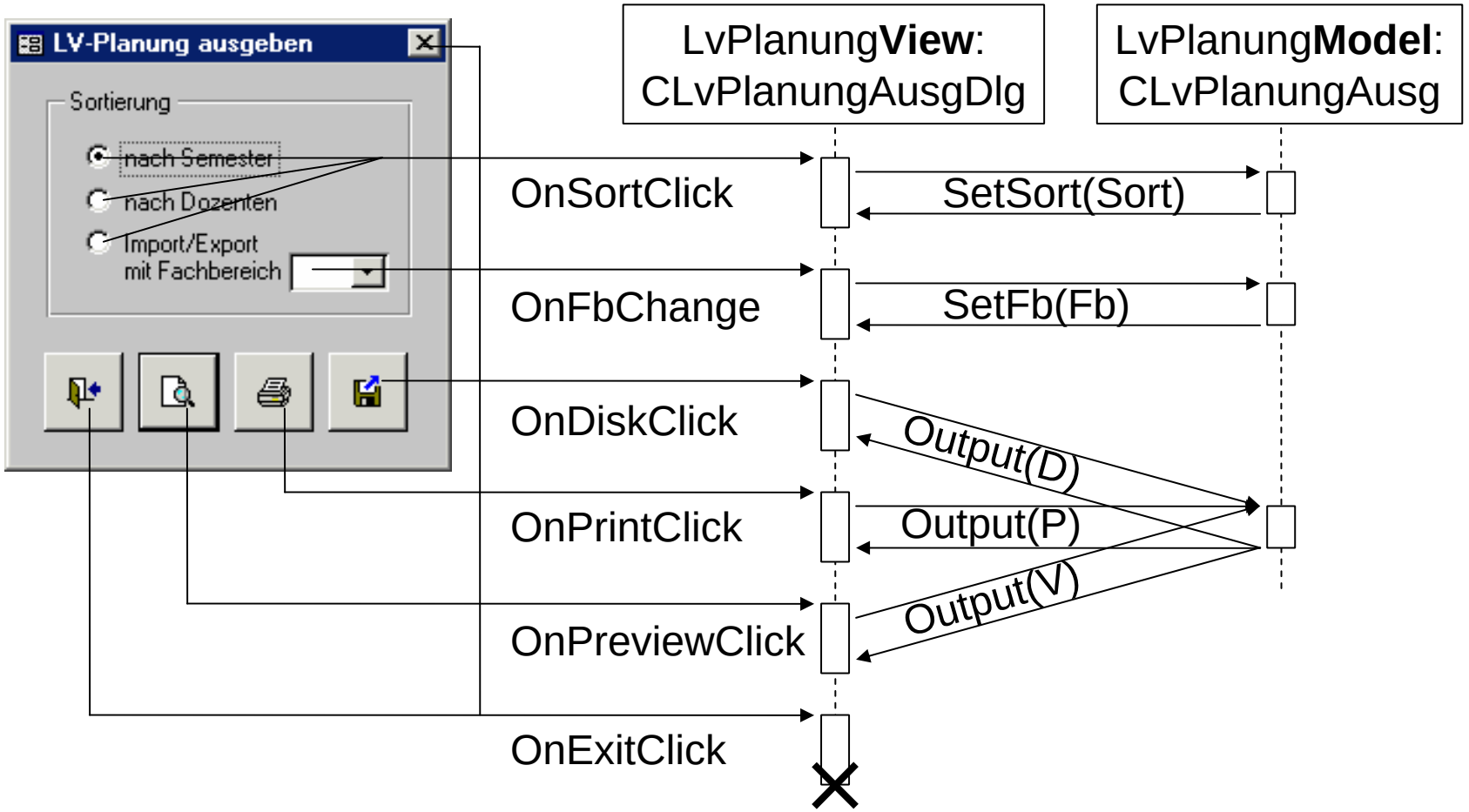
Übersicht mit Workflow

Use Cases

R: rechter Mausklick

Pop-Ups

7. Verbindung zum Sequenzdiagramm



7. Komplexität beherrschen

- Diagramme beanspruchen viel Platz
  - Problem beim Zeichnen
  - Problem beim Verstehen

gilt für alle, nicht nur für Zustandsdiagramme

- sinnvolle Zerlegung in Teildiagramme ist wichtig
  - insbesondere auch hierarchische Gliederung
  - Abstraktion
  - nicht einfach orientiert an der Papiergröße zerschneiden
- aussagekräftige Bezeichnung für jedes Teildiagramm
  - analog zu sinnvollen Klassen- und Funktionsnamen

8 Tooltip, Shortcuts, Accelerators, ...

- Tooltip
- Shortcut
- Accelerator
- Navigation mit Tastatur
  - Fokus
  - Default Button
  - Buttons mit Eingabe-Taste auslösen
- Verifikation von Eingaben
- Bestätigung der Eingabe

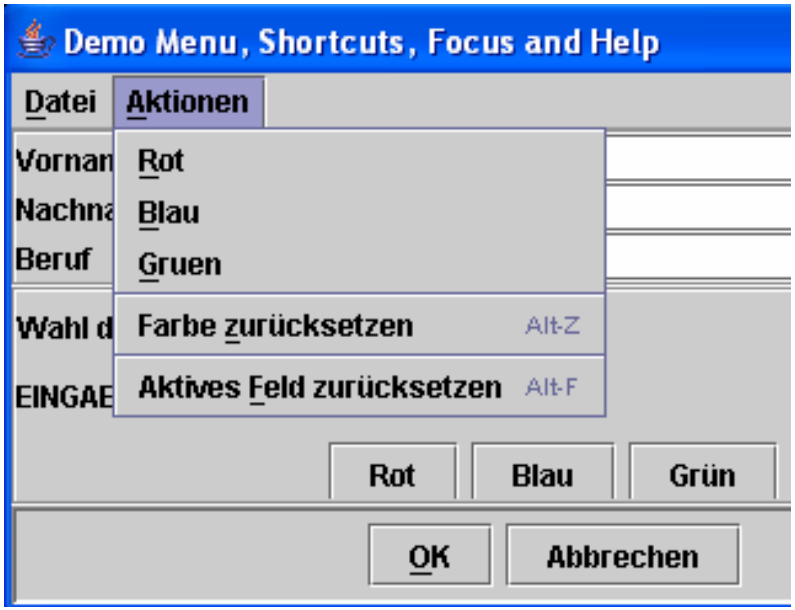
- Beispiel: „FrameNav.java“

8.1 Tooltip

- Tooltip sind kurze Texte, die für die jeweilige Komponente **component** angezeigt werden, wenn der Mauszeiger sich darüber befindet
- Tooltip-Texte werden mit der folgenden Methode gesetzt:
  component.setToolTipText(“Erläuterung …");

8.2 Shortcuts: Mnemonic

- Ist man mit einer Anwendung vertraut, kann man mit Tastenkombinationen häufig schneller navigieren, als mit dem Mauszeiger:



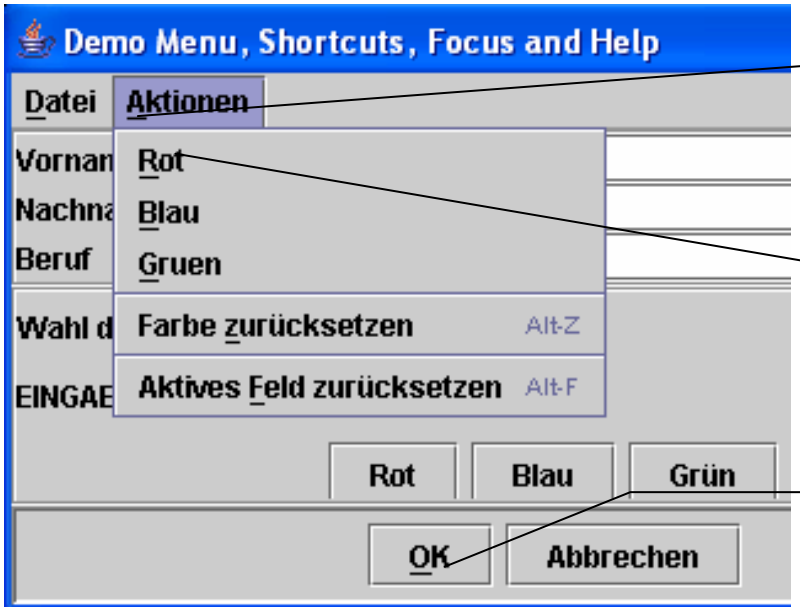
Maus: Klick auf „Aktionen“, dann z.B. „Gruen“

Tastatur: ALT+A

- ↓↑ Eingabe oder
- ALT+A, ALT+G

- Die erforderliche Tastenkombination wird durch Unterstreichung des Buchstabens angezeigt (Mnemonic)

8.2 Shortcuts: Mnemonic (2)



ALT+A öffnet Menu „**A**ktionen“

ALT+R selektiert Menu-Punkt „**R**ot“

Tastenkombination ALT+O betätigt **O**K-Button

Zwei Möglichkeiten zum Setzen von Mnemonics:

- im Konstruktor der Komponente (z.B. JMenuItem)
```
JMenuItem iRot= new JMenuItem("Rot",KeyEvent.VK_R);
```
- mit der Methode setMnemonic()
```
bOK.setMnemonic(KeyEvent.VK_O);
```

8.2 Shortcuts: Mnemonic (3)

Mnemonics auch für **CheckBox**es, **RadioButton**s etc:

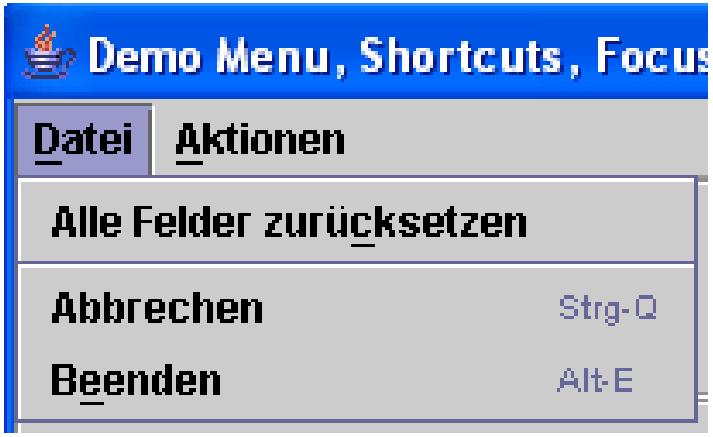
(*Übung zu Hause*: Erweitern Sie Beipiel „mvc2“ für bequeme Bedienung per Tastatur)

Traversierung erfolgt mit **TAB** und **Shift+TAB** (siehe 8.4)

CheckBoxes, **RadioButton**s können mit der **LEERTASTE** (Space) aktiviert/deaktiviert werden

8.3 Shortcuts: Accelerator

- Accelerators sind Tastenkombinationen, welche die **direkte** Aktivierung per Tastatur von Menu-Punkten ermöglicht.



Maus: Klick auf

- „Datei“ und „Beenden“
- „Datei“ und „Abbrechen“

Tastatur:

- ALT+E
- STRG+Q (CTRL+Q)

- Die erforderliche Tastenkombination wird durch die Tastenkombination (Accelerator) angezeigt

8.3 Shortcuts: Accelerator (2)

- Accelerators werden mit der Methode **`setAccelerator()`** gesetzt. Accelerators sind immer Tastenkombinationen.
- zum Beispiel für ALT+E:

```
itemExit.setAccelerator( KeyStroke.getKeyStroke(
    KeyEvent.VK_E, ActionEvent.ALT_MASK ) );
```

- für CTRL+Q:

```
itemQuit.setAccelerator( KeyStroke.getKeyStroke(
    KeyEvent.VK_Q, ActionEvent.CTRL_MASK) );
```

8.2-8.3 Konvention Shortcuts & Mnemonics

Aus „Java Look and Feel Design Guidelines“:

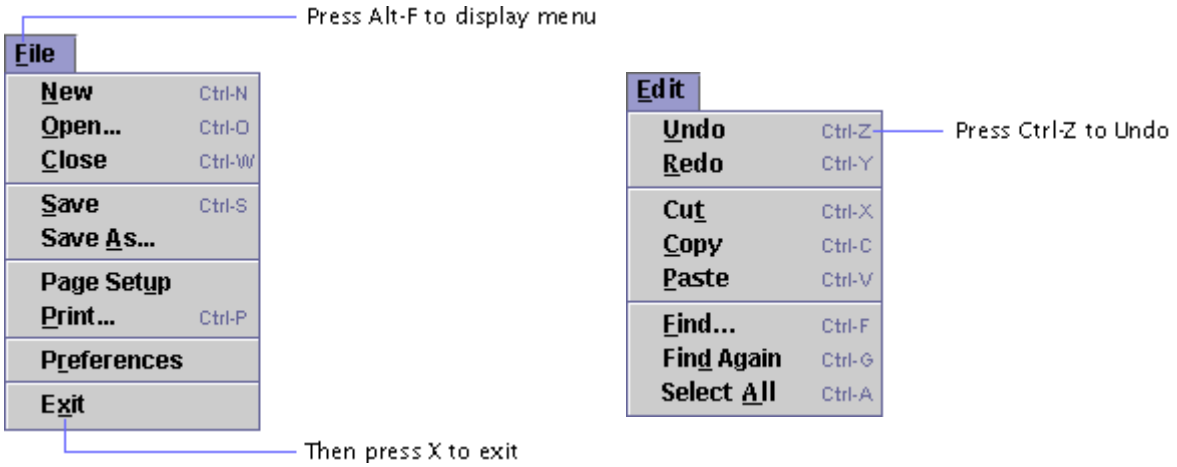
Navigation und Activation

Action	Keyboard Operation
Navigates in, navigates out	Tab[1]
Navigates out of a component that accepts tabs	Control-Tab[1]
Moves focus left one character or component within a group	Left arrow
Moves focus right one character or component within a group	Right arrow
Moves focus up one line or component within a group	Up arrow
Moves focus down one line or component within a group	Down arrow
Moves up one view	Page Up
Moves down one view	Page Down
Moves to the beginning of data; in a table, moves to the beginning of a line	Home
Moves to the end of data; in a table, moves to the last cell in a row	End
Activates the default command button	Enter or Return
Dismisses a menu or dialog box without changes	Escape
Activates or selects the component (with keyboard focus)	Spacebar

8.2-8.3 Konvention Shortcuts & Mnemonics

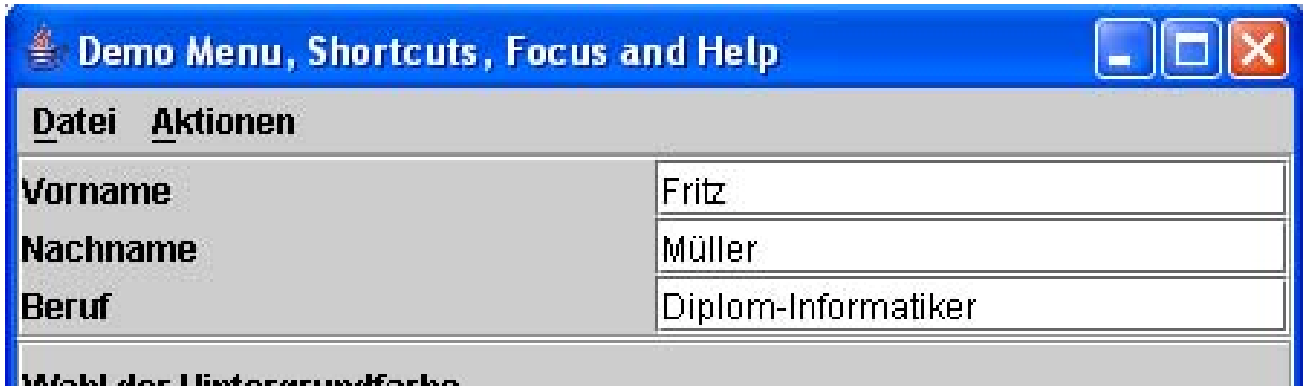
Aus „Java Look and Feel Design Guidelines“: **Menues**

Sequence	Equivalent
Ctrl-N	New (File menu)
Ctrl-O	Open (File menu)
Ctrl-S	Save (File menu)
Ctrl-P	Print (File menu)
Ctrl-W	Close (File menu)
Ctrl-Z	Undo (Edit menu)
Ctrl-Y	Redo (Edit menu)
Ctrl-X	Cut (Edit menu)
Ctrl-C	Copy (Edit menu)
Ctrl-V	Paste (Edit menu)
Ctrl-F	Find (Edit menu)
Ctrl-G	Find Again (Edit menu)
Ctrl-A	Select All (Edit menu)



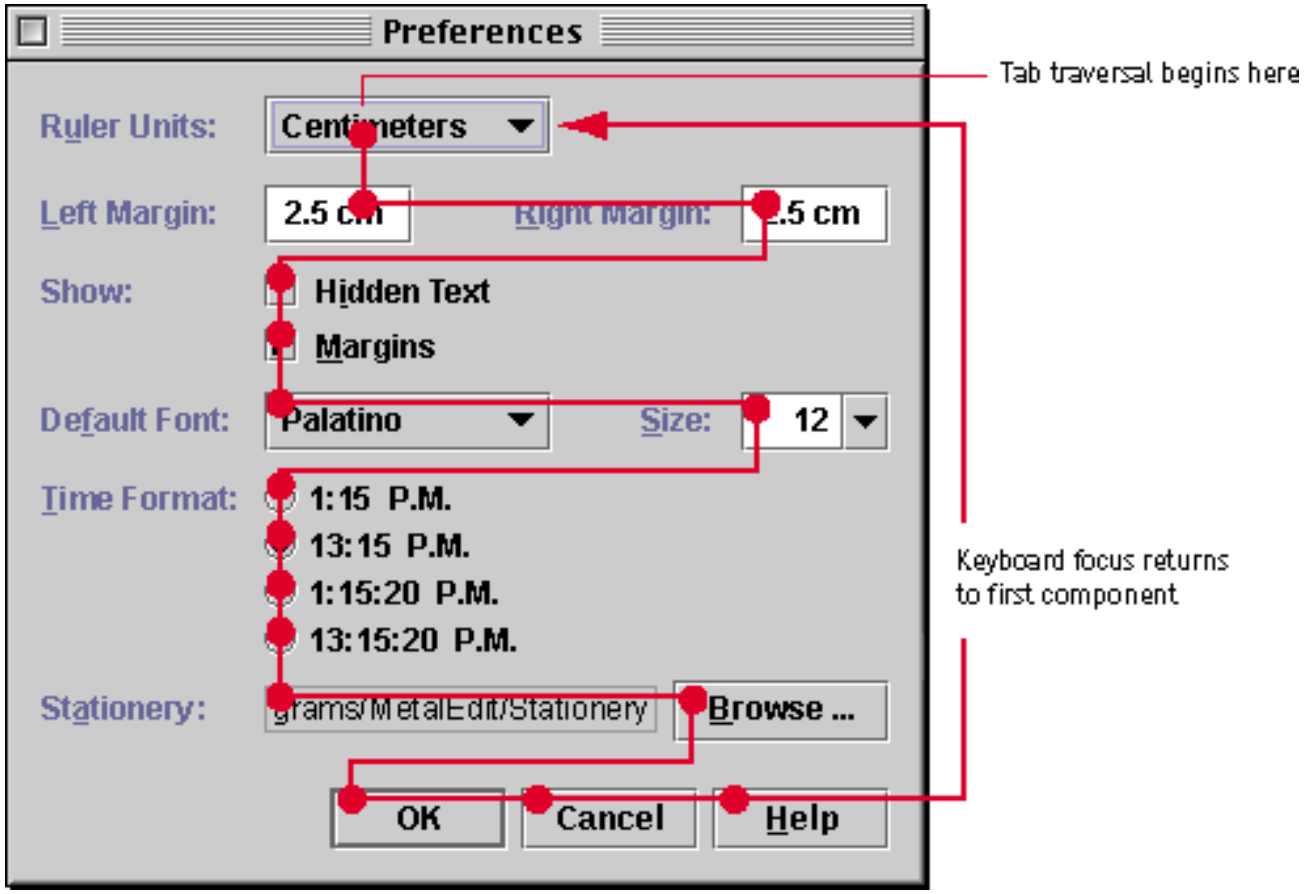
Menu Titles	Menu Items
File	New, Open, Close, Save, Save As, Page Setup, Print, Preferences, Exit
Edit	Undo, Redo, Cut, Copy, Paste, Find, Find Again, Select All
Help	Contents, Tutorial, Index, Search, About Application

8.4 Navigation durch Dialoge: Traversierung



- In Dialogen kann mit der Tastatur über die Komponenten navigiert werden:
  - **vorwärts**: mit Taste **TAB**
  - **rückwärts**: mit Taste **Umschalt + TAB (Shift + TAB)**
- Dies ermöglicht eine schnelle und komfortable Bedienung
- **Wechsel** zwischen Tastatur und Maus **kann** vollständig **vermieden werden**, besonders bei Eingabe in mehrere Textfelder

8.4 Navigation durch Dialoge (2)



- **Traversierung** erfolgt mit **TAB** und **Shift+TAB**
- **CheckBox**es, **RadioButton**s können mit der **LEERTASTE** (Space) aktiviert/deaktiviert werden

8.4.1 Fokus auf Komponenten

- Damit eine Komponente über TAB bzw. Shift+TAB den Fokus bekommen kann, muss sie sichtbar (visible), enabled und fokusierbar (focusable) sein
- Die Reihenfolge ist in der Regel im Dialog festgelegt.
- Die Reihenfolge kann verändert werden:
  - seit Java 1.4 durch Implementieren der **FocusTraversalPolicy**
  - mit Java 1.4 sind die bisherigen Methoden `setNextFocusableComponent`, `getNextFocusableComponent`, `requestDefaultFocus`, und `isManagingFocus` veraltet (deprecated)
- Man kann einer Komponente **`component`** beim Start des Dialogs den Fokus setzen:

```
component.requestFocusInWindow();
```

8.4.2 Default Button

- In vielen Dialogen gibt es eine Schaltfläche, deren Betätigung durch den Benutzer am wahrscheinlichsten ist
- Beispiel: In einem Anmeldedialog tippt der Benutzer
  - Namen (user id)
  - Passwort
- Statt von der Tastatur zur Maus zu greifen und den OK-Button zu betätigen, wäre es komfortabler die Eingabetaste zu drücken. ➔
- OK-Button als Default-Button setzen (**nach** Aufruf der Methode **`pack()`** !):

```
getRootPane().setDefaultButton( buttonOK );
```

8.4.3 Buttons über EINGABE-Taste auslösen

- Auch mehrere Schaltflächen können über die Eingabetaste betätigt werden, sobald sie den Fokus haben:
- Ein **KeyListener** verarbeitet Tastaturereignisse
- Den Schaltflächen wird die Implementierung des KeyListener bekannt gemacht.
- KeyListener (Interface) deklariert 3 Methoden:

```
public void keyPressed(KeyEvent e) {...}
public void keyReleased(KeyEvent e) {...}
public void keyTyped(KeyEvent e) {...}
```

- Details siehe Beispiel „FrameNav“

8.4.4 Fokus auf Labels

- Auch auf **Labels** kann mittels **Mnemonic** der Fokus über die Tastatur gesetzt werden
- Sinnvoll in großen Dialogen um schneller zu navigieren
- Der Fokus auf dem Label selbst macht wenig Sinn. Fokus kann **aber auf eine Komponente** wie TextField, CheckBox, RadioButton, ComboBox etc. **verweisen**.

- Mnemonic für ein Label `label` wird gesetzt mit:
  `label.setDisplayedMnemonic(KeyEvent.VK_S);`
- Verweis auf eine Komponente `component` erfolgt mit:
  `label.setLabelFor(component);`

8.5 Verifikation von Eingaben

Fehlervermeidung ist **besser als Korrektur**:

- Angenommen der Benutzer füllt ein Formular mit vielen Feldern aus und und soll dann mit einer Schaltfläche abschliessen
- weiterhin angenommen ein Feld (z.B. Datum) sei falsch ausgefüllt.

→ Möglichkeit der Rückmeldung:

1. Mit der Eingabebestätigung erfolgt Fehlermeldung
2. fehlerhaftes Feld erhält sofort wieder den Fokus und evtl. verbunden mit akustischer Rückmeldung

➔ 2. Möglichkeit ist besser!

In Java (ab 1.3) mit **`javax.swing.InputVerifier`**

Beispiel: *VerifierTest.java*