**Swing**

Abgrenzung von AWT und Swing

AWT (Abstract Window Toolkit) arbeitet mit „Heavyweight components

* Verwendung von plattformspezifischen AWT-Klassen (nicht in Java Implementiert)
* AWT-Komponenten besitzen einen Partner auf Betriebssystemseite (Peer), der Darstellung und Funktionalität steuert
* sehr schnell, da die Peer-Klassen im Code der Ausführungsplattform geschrieben sind

Swing arbeitet mit „Lightweight components“

* nur sehr wenige plattformspezifische GUI-Ressourcen verwendet
* lightweight components besitzen keinen Peer auf Betriebssystemseite
* besitzt zahlreiche zusätzliche GUI-Komponenten
* „bessere“ Plattformunabhängigkeit dafür langsamer

Ein Bild, das Text enthält.

Automatisch generierte BeschreibungJFrame

* Hauptkomponente ist Root Pane
* Darunter Panel Hierarchie
* neue Komponenten werden der Content Pane zugeordnet

Wichtige Methoden

überladener Konstruktor  
setDefaultCloseOperation(int i) legt fest, was beim Schließen des Fensters passiert

dabei mögliche Konstanten

WindowConstants.DO\_NOTHING\_ON\_CLOSE löst lediglich das Close-Event aus

WindowConstants.HIDE\_ON\_CLOSE versteckt das Fenster

WindowConstants.DISPOSE\_ON\_CLOSE zerstört den Frame

WindowConstants.EXIT\_ON\_CLOSE beendet die Applikatio

Getter- und Setter-Methoden für die Panels eines JFrames

Methoden aus der Klasse java.awt.Window

setBounds(int x, int y, int width, int heigth)

pack() passt die Fenstergröße an den Content an

setVisible(boolean b) aus der Klasse java.awt.Component

Layoutmanager

Anordnung der Elemente eines Containers nach bestimmten Verfahren

wesentliche Layout-Manager

* Flow-Layout ordnet seine Elemente von links nach rechts
* Border-Layout ermöglicht eine Anordnung in 5 verschiedenen Bereichen
* Grid-Layout ermöglicht die Anordnung der Komponenten in Zeilen und Spalten

mit der Methode setLayout(LayoutManager l) wird für ein JFrame der Layout-Manager gesetzt

Der Container JPanel

* weitere Container-Form
* ordnet mehrere Elemente unter der Kontrolle eines Layoutmanagers
* Layoutmanager und Komponenten werden direkt dem Panel zugewiesen
* bereits dem Konstruktor wird der Layoutmanager mitgegeben l über die add()- Methode werden die Komponenten dem Panel zugeordnet

Ein Bild, das Tisch enthält.

Automatisch generierte BeschreibungRahmen mit dem Border Interface

* sind über Klassen realisiert, die das Interface implementieren
* sollten nicht direkt über die Konstruktoren der Rahmen-Klassen sondern  
  über die Klassenmethoden der BorderFactory erzeugt werden
* jeder Swing-Komponente kann mit der Methode setBorder(Border b)   
  ein Rahmen zugewiesen werden
* einige Standardrahmen sind in Swing bereits implementiert

Beschriftungen und Grafikanzeige mit JLabel

* ermöglicht einfache Anzeige von Texten oder Grafiken
* zum Text können Icons angezeigt werden
* bietet Möglichkeit HTML-Tags anzuzeigen

Unterschiedliche Arten von Textfeldern

einfache Textfelder der Klasse JTextField

* überladener Konstruktor, um das Feld mit einem String vorzubelegen und/oder die Breite anzugeben
* Angabe der Schriftart über die Methode setFont()
* Auslesen des Inhalts über die Methode getText()

spezielle Felder für Passwörter der Klasse JPasswordfield

* Konstruktoren analog der Klasse JTextField
* Auslesen des Inhalts über die Methode getPassword()
* zwei boolsche Methoden cut() und copy(), die überprüfen, ob Werte mit cut(STRG+X) oder copy (STRG+C) aus dem Feld ausgelesen werden dürfen

mehrzeilige Textfelder der Klasse JTextArea

* Konstruktoren analog der Klasse JTextField (aber mit Höhe)
* Auslesen und ändern der Schriftart analog der Klasse JTextField
* Zeilenumbrüche werden bei getText() berücksichtigt

Ein Bild, das Tisch enthält.

Automatisch generierte BeschreibungValidierende Textfelder als spezielle Form

realisiert durch die Klasse JFormattedTextField

dem Konstruktor der Klasse wird das Format mitgegeben

mehrere Klassen stehen für die Maskierung zur Verfügung

* alle Objekte der Sub-Klassen der Klasse Format
* z.B. bei Drücken der Enter-Taste wird die Eingabe überprüft und ein mögliches ActionEvent ausgelöst

Objekte der Klasse MaskFormatter erlauben nur bestimmte Zeichen bei der Eingabe

Drop-Down-Listen über JComboBox

eine bestimmte Wertemenge wird zur Auswahl bereitgestellt

dem Konstruktor der Klasse JComboBox wird die Wertemenge als ein Array von Objekten der Klasse Object übergeben

wesentliche Methoden der Klasse JComboBox

* getSelectedItem() liefert den Wert des ausgewählten Elements zurück
* setSelectedItem(Object o) belegt das Feld mit dem Wert o vor, sofern dieser in dem  
   Array der Wertemenge vorhanden ist
* setEditable(boolean b) bestimmt, ob auch Werte außerhalb der Wertemenge erlaubt

Weitere Klassen aus dem Package Swing

JTable dient der Erstellung zweidimensionaler Tabellen  
JTree ermöglicht die Darstellung von Bäumen ähnlich dem Windows Explorer bestehend  
 aus Knoten und Blättern  
JToolBar dient der Erstellung von Symbolleisten analog den Microsoft Office  
JColorChooser dient der Erstellung eines Auswahldialogs zur Farbeinstellung  
JFileChooser dient der Erstellung eines Dialogs zur Auswahl einer Datei im FileSystem

**Das Event Konzept**

* UI Komponenten erzeugen Events   
  (Button ActionEvent, Auswahl ItemEvent, Fensterwechsel FocusEvent)
* Jeder Event-Typ hat ein Listener Interface um auf das Event reagieren zu können

(ActionEvent ActionListener, ItemEvent ItemListener, FocusEvent Focus Listener)

* Komponenten, welche Events erzeugen, können (mehrere) Implementierungen des jeweiligen Interfaces registrieren (z.b. addActionListener(ActionListener a))
* Tritt ein Event auf, wird die jeweilige Methode der registrierten Interface-Implementierung aufgerufen

Die Aufgaben des ItemListener

* als Interface implementiert
* gibt die abstrakte Methode itemStateChanged(ItemEvent e)
* wird von Objekten implementiert, die an einem Auswahlereignis interessiert sind
* Auswahlereignisse können von Objekten folgender Klassen ausgelöst werden: JComboBox, JCkeckBox, JList oder JCheckBoxMenuItem
* die Zuordnung zu einem ItemListener erfolgt über die jeweiligen Objekt-Methoden addItemListener() oder removeItemListener()
* wird ein Eintrag bei o.g. Objekten ausgewählt, wird implizit die Methode itemStateChanged(ItemEvent e) bei allen bei dem Objekt registrierten ItemListenern ausgeführt

Interaktion über Drucktasten

* überladener Konstruktor, der es ermöglicht Text und oder Graphik in Form eines Icons auf Button zu positionieren
* mit der Methode setText(String s) kann der Text nachträglich verändert werden
* wichtigste Methoden addActionListener() und removeActionListener()
* der ActionListener ist der Beobachter des Knopfes; ohne ActionListener kann dem Button keine Funktionalität zugewiesen werden
* sobald der Button gedrückt wird, wird ein ActionEvent ausgelöst, welches vom Beobachter abgefangen und ausgewertet wird

Die Aufgaben des Action Listener

* als Interface implementiert
* gibt die abstrakte Methode actionPerformed(ActionEvent e)
* wird implizit ausgeführt, sobald ein „abgehörtes“ Objekt ein ActionEvent auslöst
* die Klasse ActionEvent besteht aus drei Methoden
  + getActionCommand() liefert den String, der mit der Aktion verbunden ist
  + getModifiers() liefert einen Integer-Wert zurück, welche  
     Funktionstaste bei dem Ereignis gedrückt wurde
  + paramString() liefert einen Erkennungs-String, der mit  
     „ACTION\_PERFORMED“ oder „unknown type“  
     beginnt

Kontrollfelder mit JCheckBox

* kennen zwei Zustände: selektiert (true) und nicht selektiert (false)
* überladener Konstruktor, der es ermöglicht Text, Initialwert (true oder false) und Icon mitzugeben
* werden normalerweise als Kästchen mit einem Häkchen für den selektierten Zustand dargestellt
* Zustand kann über die Methode setSelected(boolean b) geändert werden
* kann allerdings nicht direkt über eine Getter-Methode ausgelesen werden
* bei der Änderung des Zustands durch den Anwender wird ein ItemEvent ausgelöst und an alle registrierten ItemListener weitergeleitet
* im ItemListener kann der Zustand des Kontrollfeldes ausgewertet und weiterverarbeitet werden

Optionsfelder mit JRadioButton & ButtonGroup

* bieten mehrere Auswahlmöglichkeiten an, wobei nur eine Option ausgewählt werden kann
* werden in einem Objekt der Klasse ButtonGroup zu einer Optionsfeldgruppe zusammengefasst
  + mit der Objektmethode add(AbstractButton b) der Klasse ButtonGroup wird ein Optionsfeld der Gruppe hinzugefügt
  + mit der Objektmethode remove(AbstractButton b) der Klasse ButtonGroup wird ein Optionsfeld aus der Gruppe entfernt
* überladener Konstruktor der Klasse JRadioButton analog der Klasse JCheckBox
* werden normalerweise als Kreis mit einem schwarzen Punktfür den selektierten Zustand dargestellt
* bei der Änderung des Zustands eines Optionsfeldes wird ein ActionEvent ausgelöst und an alle registrierten ActionListener weitergeleitet
* im ActionListener kann die Auswertung der Optionsfelder erfolgen

Erstellen von Menüs mit JMenuBar

* JMenuBar ist der Container für die einzelnen Menüs
  + mit der add(JMenu m) Methode wird dem Container ein Menü hinzugefügt
* Objekte der Klasse stellen die einzelnen Menüs dar und sind Container für konkrete Menüeinträge der Klasse JMenuItem
  + mit der add(JMenuItem i) Methode wird einem Menü ein konkreter Menüeintrag zugeordnet
* mit der Methode setJMenuBar(JMenubar m) wird einem Fenster eine Menüleiste zugeordnet
* um auf die Auswahl eines Menüeintrags zu reagieren, müssen die Menüeinträge einem ActionListener zugeordnet werden

Tooltips

* kleinere Hilfetexte, die beim längeren Verweilen auf einem GUI-Objektin einem kleinen PopUp-Fenster angezeigt werden
* ToolTips werden nicht direkt über den Konstruktor der Klasse JToolTip erzeugt, sondern über die Methode setToolTipText(String s) des GUI-Objektes