Medientechnik

Heinrich Hußmann
Ludwig-Maximilians-Universität München
Wintersemester 2003/2004

Lehr- und Forschungseinheit Medieninformatik

Prof. Dr. Heinrich Hußmann
Amalienstr. 17, 5. OG, Raum 508
Email hussmann@informatik.uni-muenchen.de

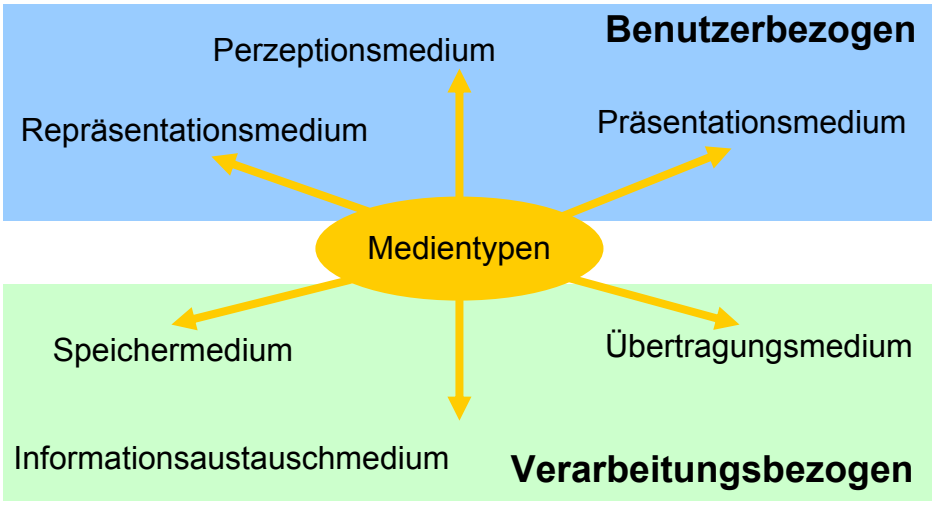
Übungsleitung:
Arnd Vitzthum, Raum 501

Wichtigste Informationsquelle:
http://mimuc.de/mt
(mimuc = MedienInformatik in MUC;
mt = Medientechnik)

Inhalt

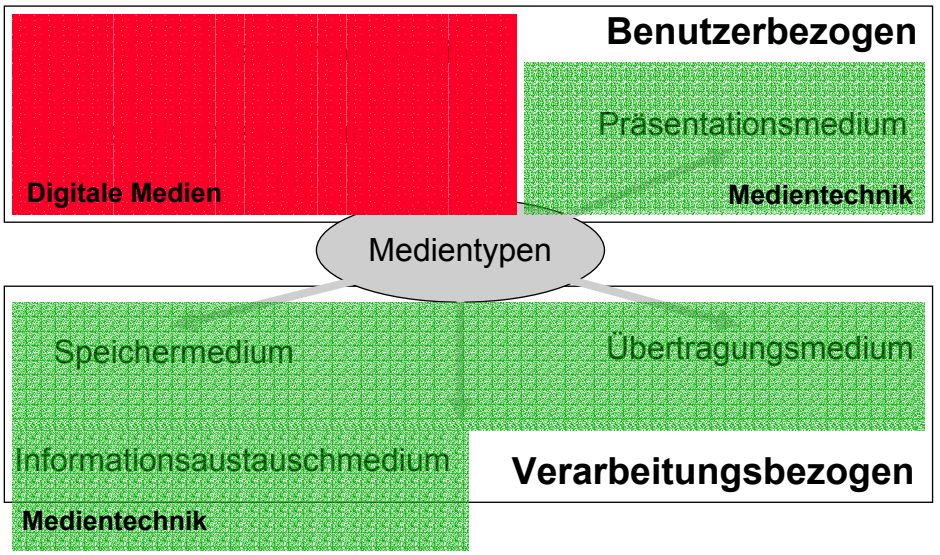
- Diese Vorlesung: Ergänzendes Wissen zu digitalen Medien
  - Teilweise aufbauend auf "Digitale Medien"
  - Schwerpunkt 1: Hardwarefragen, Audio-, Foto- und Video-Technik
  - Schwerpunkt 2: Medienbezogene Programmierung in Java
- Themen dieser Vorlesung:
  - Programmierung grafischer Benutzungsoberflächen (Bsp. Java Swing)
  - Grundlagen der 2D-Computergrafik
  - Klassische Ein- und Ausgabegeräte
  - Überblick zu Speichermedien, digitalen Schnittstellen, Netztechnologien
  - Digitale Bildverarbeitung (Fototechnik, Programmierschnittstellen)
  - Digitale Tonverarbeitung (Aufnahme- und Wiedergabetechnik, Programmierschnittstellen)
  - Digitale Filmverarbeitung (TV-, Video- und Filmtechnik, digitaler Filmschnitt, Programmierschnittstellen)
  - Techniken zur digitalen Verarbeitung dreidimensionaler Bildinformation
  - WWW-basierte interaktive Medien (Beispiele Java Applets/Servlets, JSP)

Typen von technischen „Medien“



Alle Medientypen gehören zum (weiteren) Gebiet der Medieninformatik;
im engeren Sinne konzentrieren wir uns auf benutzerbezogene Medientypen

Abdeckung der „technischen Medien“ in den LV



Begleitende Literatur

Zu dieser Vorlesung empfohlen:

- Peter A. Henning: Taschenbuch Multimedia, 2. Auflage, Fachbuchverlag Leipzig/Carl Hanser 2001
- Andreas Holzinger: Basiswissen Multimedia, Band 1: Technik, Vogel Verlag, 2000
- Ralf Steinmetz: Multimedia-Technologie. Grundlagen, Komponenten und Systeme, Springer, 2000

Weiterführende Literatur:

- siehe die Web-Seiten zur Vorlesung !

Vorlesung und Übungen

- Vorlesung "Medientechnik":
  - Konzepte, Überblickswissen
  - ***Keine*** vollständigen Listen von Schnittstellenmethoden, Parametern etc.
  - Keine Bedienungsanleitungen für Softwaresysteme
- Übungen "Medientechnik":
  - Praktische Anwendung und Ergänzung des Vorlesungsstoffs
  - Zum Themenschwerpunkt Hardware/Technik:
    - » Angeleitete Laborexperimente in kleinen Gruppen; Hausaufgaben
  - Zum Themenschwerpunkt Programmierung:
    - » Übungsgruppen, Klausur
  - Erste Übungsstunde (Einschreibung):
    Mittwoch, 29. Oktober, 16 Uhr c.t., The/E45
    Donnerstag, 30. Oktober, 16 Uhr c.t., The/112
- Erwerb des Leistungsnachweises:
  - Klausur nach Abschluss der Vorlesung
  - Hausaufgaben zum Erwerb der Teilnahmeberechtigung an der Klausur

Spezialisten gesucht!

- Gibt es Hörer, die aus Hobby oder Beruf Spezialkenntnisse in Medientechnik haben?

  Z.B. in:
  - Digitaler Musikproduktion, Studiotechnik
  - Filmproduktion
  - Anwendungen von 3D-Datenverarbeitung
  - ...
- Angebot:
  - Gestaltung eines Vortrags oder einer praktischen Vorführung in Ergänzung zur Vorlesung
  - Gegenleistung: reduzierte Hausaufgabenpflicht
- Bei Interesse bitte am Lehrstuhl melden!

Gliederung

1. Konventionelle Ein-/Ausgabebetonte Programmierung
2. Konventionelle Eingabetechnik
3. Konventionelle Ausgabetechnik
4. Speichermedien
5. Digitale Schnittstellen und Vernetzung im Überblick
6. Techniken zur digitalen Bildverarbeitung
7. Techniken zur digitalen Tonverarbeitung
8. Techniken zur digitalen Filmverarbeitung
9. Techniken zur digitalen Verarbeitung 3-dimensionaler Darstellungen
10. WWW-basierte interaktive Medien

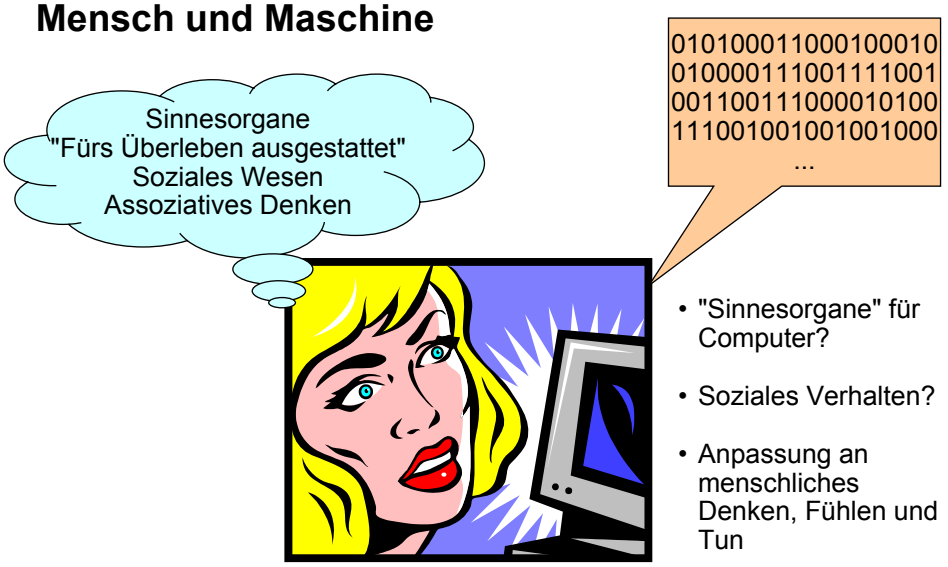
1. Konventionelle Ein-/Ausgabebetonte Programmierung

1.1 Mensch-Maschine-Kommunikation 

1.2 Realisierung grafischer Benutzungsoberflächen
– Beispiel Java AWT und Swing

1.3 Grundlagen der 2D-Computergrafik
– Beispiel Java-Grafikprogrammierung, Java 2D

Mensch und Maschine



- "Sinnesorgane" für Computer?
- Soziales Verhalten?
- Anpassung an menschliches Denken, Fühlen und Tun

"Ohren und Mund" des Computers

- Konventionell:
  - Eingabe: Tastatur, Zeigegeräte
  - Ausgabe: Bildschirmanzeige, Drucker
- Multimedial:
  - Eingabe: Kameras, Scanner, Mikrofone, Musikinstrumente, ...
  - Ausgabe: Fernsehgeräte, Lautsprecher, ...
- "*Ubiquitious Computing*" ("Allgegenwärtige" Rechnerunterstützung):
  - Eingabe: Alltagsgegenstände, Anwesenheit, Bewegungen, ...
  - Ausgabe: Allltagsgegenstände, Beleuchtung, beliebige Geräte, ...

Kommunikationsqualität bei konventioneller Ein-/Ausgabe

- Ausgabe ("Mund"):
  - Bildschirmanzeige erlaubt hochwertige Bilder (Farbtiefe besser als wahrnehmbar, Bildfrequenz angemessen, Auflösung akzeptabel)
  - Tonausgabe mit gleicher Qualität möglich wie bei Musikwiedergabegeräten
- Eingabe ("Ohren"):
  - Tastatur/Maus wesentlich langsamer als vom Menschen generierte Informationsraten
  - Spracherkennung, Handschrifterkennung etc. qualitativ noch unzureichend
- Computer können sich gut ausdrücken, aber dem Menschen nur sehr schlecht zuhören!

  (nach: Chris Crawford, The Art of Interactive Design)

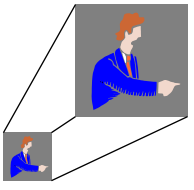
- Konsequenzen:
  - Eingabekanäle auf guten Durchsatz optimieren
  - Ausgabekanäle auf Übersichtlichkeit optimieren

Software-Ergonomie *(usability)*

- Grenzgebiet zur Psychologie
- Gestaltung von Software unter dem Aspekt der Benutzbarkeit

Angemessen zur Lösung der Aufgabe

Flexibel für verschiedene Arbeitsweisen und Zugänge



Erlaubt Weiterentwicklung: Lernen während der Arbeit

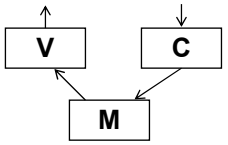
Benutzungsoberflächen

- Technische Realisierungen:
  - Stapelverarbeitungssprache *(batch control, job control)*
  - Zeilenorientierte interaktive Kommandosprache
    - » Beispiele: Kommandosprachen von MS-DOS, UNIX
  - Skriptsprache
  - Bildschirm- und maskenorientierter Dialog
    - » Beispiele: Dialogoberfläche von MVS, VM/CMS
  - **Graphische Benutzungsoberfläche *(graphical user interface, GUI)***
  - Multimedia-Benutzungsoberfläche
  - Virtuelle Welt

- Tendenz:
  - Bessere Anpassung an menschliche Kommunikation
  - Weg von sequentieller Organisation hin zu freier Interaktionsgestaltung

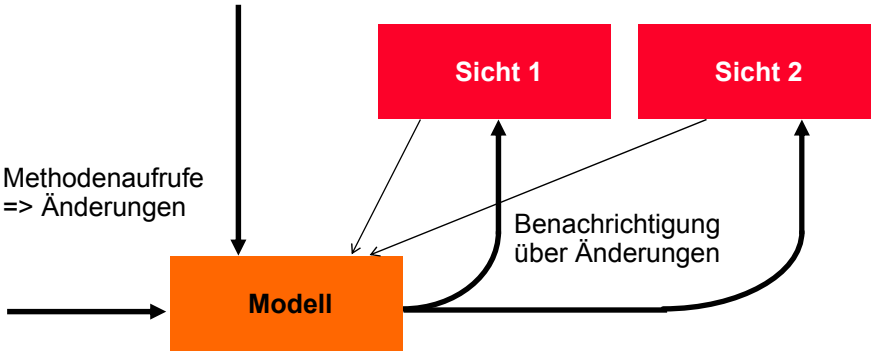
1. Konventionelle Ein-/Ausgabebetonte Programmierung

Model-View-Controller-Architektur (MVC)



- Model:
  - Fachliches Modell, weitestgehend unabhängig von Oberfläche
  - Beobachtbar *(observable)*
- View:
  - Repräsentation auf Benutzungsoberfläche
  - Beobachter des Modells
  - Erfragt beim "update" ggf. notwendige Daten beim Modell
- Controller:
  - Modifiziert Werte im Modell
  - Ist an bestimmte Elemente der "View" (z.B. Buttons) gekoppelt
  - Reagiert auf Ereignisse und setzt sie um in Methodenaufrufe

Modell und Sicht

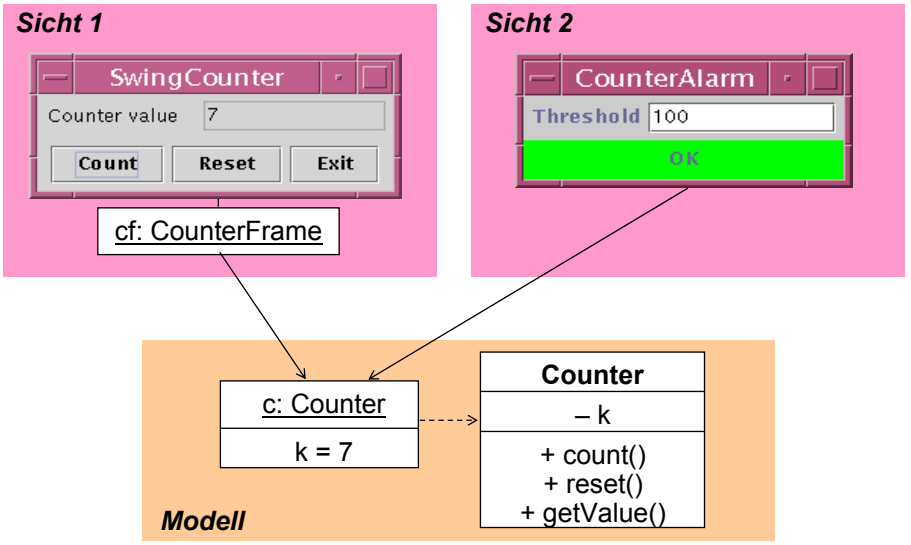


Beispiele: Verschiedene Dokumentenansichten, Statusanzeigen, Verfügbarkeit von Menüpunkten

Frage: *Wie hält man das Modell unabhängig von den einzelnen Sichten darauf ?*

Muster "Observer"

Sichten: Motivierendes Beispiel



Ein Zähler (Beispiel fachliches Modell)

```
class Counter                          {
  private int k = 0;
  public void count () {
      k++;


  }
  public void reset () {
      k = 0;


  }
  public int getValue () {
      return k;
  }
}
```

Beobachtbares Modell *(Model)*

```
class Counter extends Observable {
  private int k = 0;
  public void count () {
      k++;
      setChanged();
      notifyObservers();
  }
  public void reset () {
      k = 0;
      setChanged();
      notifyObservers();
  }
  public int getValue () {
      return k;
  }
}
```

- Das fachliche Modell enthält keinerlei Bezug auf die Benutzungsoberfläche !

java.util.Observable, java.util.Observer

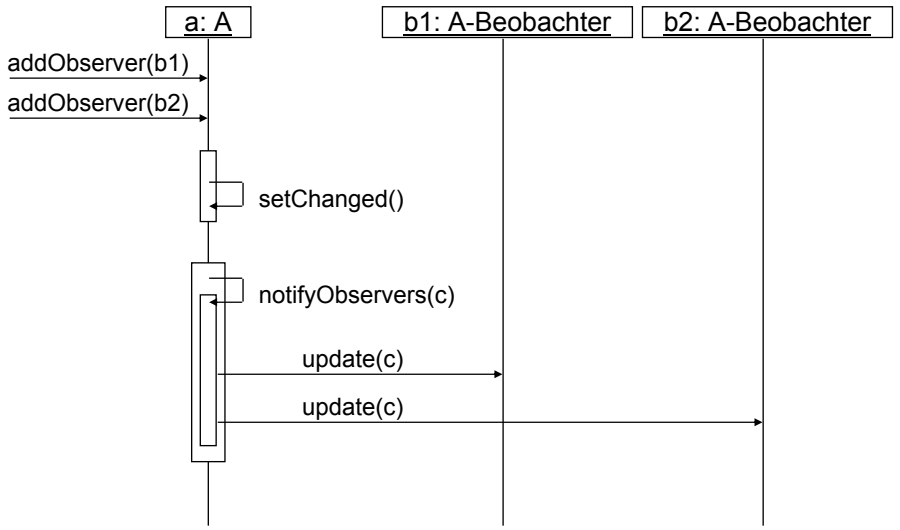
```
public class Observable {

  public void addObserver (Observer o);
  public void deleteObserver (Observer o);

  protected void setChanged();
  public void notifyObservers ();
  public void notifyObservers (Object arg);
}

public interface Observer {

  public void update (Observable o, Object arg);
}
```

Argumente für notifyObservers():

– meist nur Art der Änderung, nicht gesamte Zustandsinformation

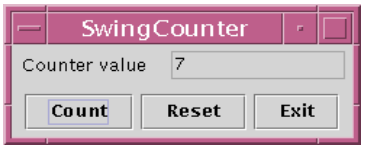
– Beobachter können normale Methodenaufrufe nutzen, um sich näher zu informieren.

Beispielablauf



Graphische Benutzungsoberflächen

- 1980: Smalltalk-80-Oberfläche (Xerox)
- 1983/84: Lisa/Macintosh-Oberfläche (Apple)
- 1988: NextStep (Next)
- 1989: OpenLook (Sun)
- 1989: Motif (Open Software Foundation)
- 1987/91: OS/2 Presentation Manager (IBM)
- 1990: Windows 3.0 (Microsoft)
- 1995-2001: Windows 95/NT/98/2000/ME/XP (Microsoft)
- 1995: Java ***Abstract Window Toolkit AWT*** (SunSoft)
- 1997: ***Swing*** Components for Java (SunSoft)



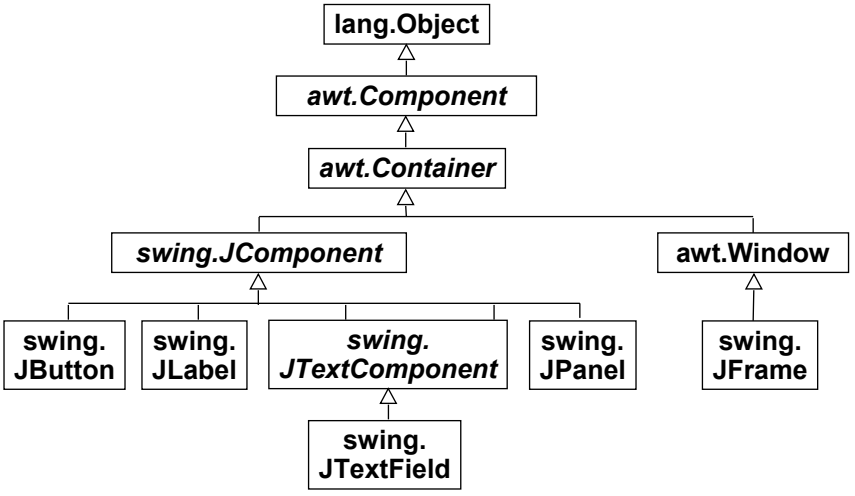
Bibliotheken von AWT und Swing

- Wichtigste AWT-Pakete:
  - **java.awt**: u.a. Grafik, Oberflächenkomponenten, Layout-Manager
  - **java.awt.event**: Ereignisbehandlung
  - Andere Pakete für weitere Spezialzwecke
- Wichtigstes Swing-Paket:
  - **javax.swing**: Oberflächenkomponenten
  - Andere Pakete für Spezialzwecke
- Viele AWT-Klassen werden auch in Swing verwendet!
- Standard-Vorspann:

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

- (Naiver) Unterschied zwischen AWT- und Swing-Komponenten:
  - AWT: Button, Frame, Menu, ...
  - Swing: JButton, JFrame, JMenu, ...

AWT/Swing-Klassenhierarchie (Ausschnitt)



- Dies ist nur ein sehr kleiner Ausschnitt!
- Präfixe "java." und "javax." hier weggelassen.

Component, Container, Window, Frame, Panel

- ***awt.Component*** (abstrakt):
  - Oberklasse aller Bestandteile der Oberfläche

```
public void setSize (int width, int height);
public void setVisible (boolean b);
```

- ***awt.Container*** (abstrakt):
  - Oberklasse aller Komponenten, die andere Komponenten enthalten

```
public void add (Component comp);
public void setLayout (LayoutManager mgr);
```

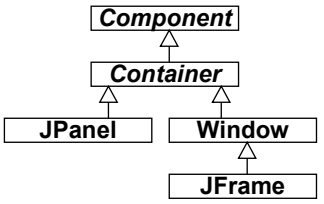
- **awt.Window**
  - Fenster ohne Rahmen oder Menüs

```
public void pack (); //Größe anpassen
```

- **swing.JFrame**
  - Größenveränderbares Fenster mit Titel

```
public void setTitle (String title);
```

- **swing.JPanel**
  - Zusammenfassung von Swing-Komponenten



JComponent

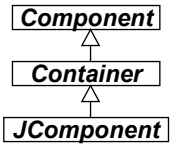
- Oberklasse aller in der Swing-Bibliothek neu implementierten, verbesserten Oberflächenkomponenten. Eigenschaften u.a.:
  - Einstellbares "Look-and-Feel" (sh. später)
  - Komponenten kombinierbar und erweiterbar
  - Rahmen für Komponenten

```
void setBorder (Border border);
```

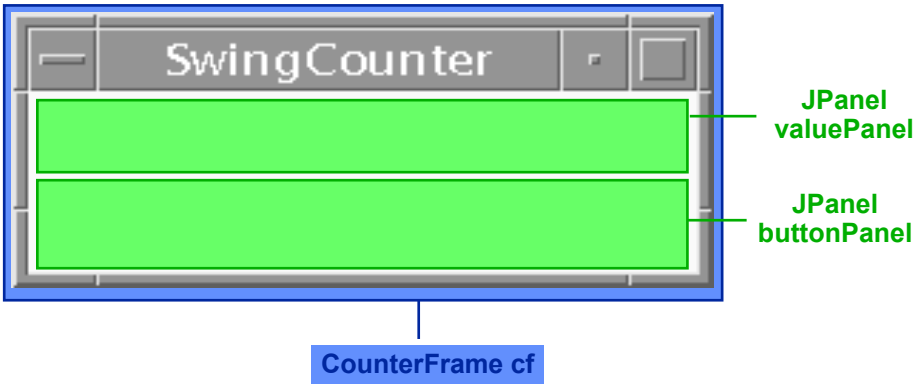
    (Border-Objekte mit `BorderFactory` erzeugbar)
  - ToolTips -- Kurzbeschreibungen, die auftauchen, wenn der Cursor über der Komponente liegt

```
void setToolTipText (String text);
```

  - Automatisches Scrolling
- Beispiele für weitere Unterklassen von JComponent:
  - JList: Auswahlliste
  - JComboBox: "Drop-Down"-Auswahlliste mit Texteingabemöglichkeit
  - JPopUpMenu: "Pop-Up"-Menü
  - JFileChooser: Dateiauswahl



Zähler-Beispiel: Grobentwurf der Oberfläche

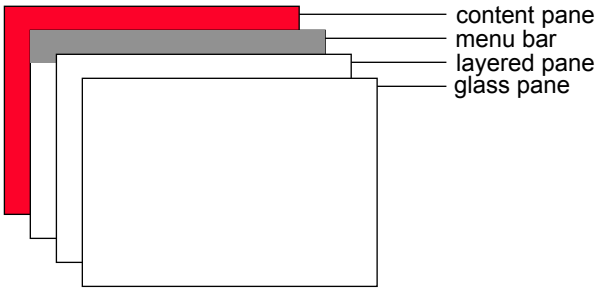


Die Sicht *(View):* Gliederung, 1. Versuch

```
class CounterFrame extends JFrame {
  JPanel valuePanel = new JPanel();

  JPanel buttonPanel = new JPanel();



  public CounterFrame (Counter c) {
    setTitle("SwingCounter");


    ... valuePanel zu this hinzufügen


    ... buttonPanel zu this hinzufügen
    pack();
    setVisible(true);
  }
}
```

Hinzufügen von Komponenten zu JFrames

- Ein JFrame ist ein "Container", d.h. dient zur Aufnahme weiterer Elemente.
- Ein JFrame ist intern in verschiedene "Scheiben" *(panes)* organisiert. Die wichtigste ist die *content pane*.

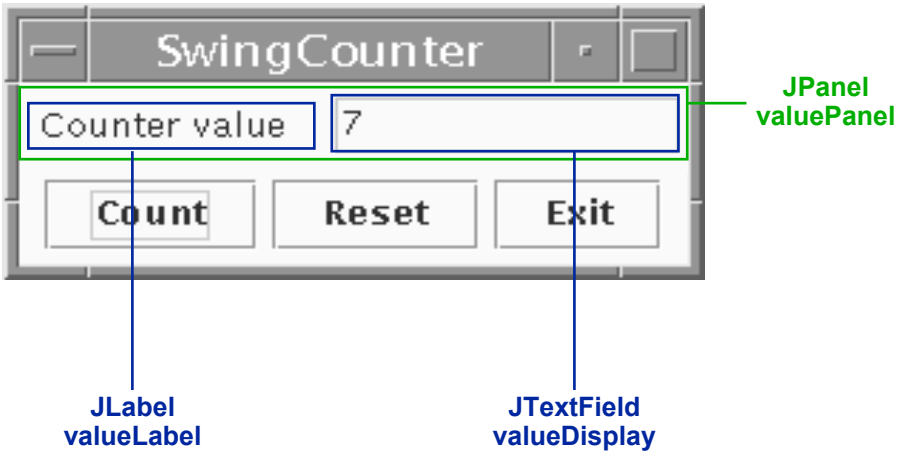


- In JFrame ist definiert:
  `Container getContentPane();`

Die Sicht *(View):* Gliederung, 2. Versuch

```
class CounterFrame extends JFrame {
  JPanel valuePanel = new JPanel();

  JPanel buttonPanel = new JPanel();



  public CounterFrame (Counter c) {
    setTitle("SwingCounter");


    getContentPane().add(valuePanel);


    getContentPane().add(buttonPanel);
    pack();
    setVisible(true);
  }
}
```

Zähler-Beispiel: Entwurf der Wertanzeige



TextComponent, TextField, Label, Button

- ***JTextComponent:***
  - Oberklasse von JTextField und JTextArea

```
public void setText (String t);
public String getText ();
public void setEditable (boolean b);
```

- **JTextField:**
  - Textfeld mit einer Zeile

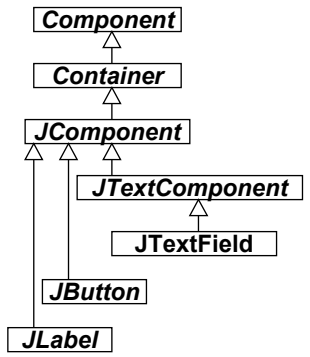
```
public JTextField (int length);
```

- **JLabel:**
  - Einzeiliger unveränderbarer Text

```
public JLabel (String text);
```

- **JButton:**
  - Druckknopf mit Textbeschriftung

```
public JButton (String label);
```



Die Sicht *(View):* Elemente der Wertanzeige

```
class CounterFrame extends JFrame {
  JPanel valuePanel = new JPanel();
  JTextField valueDisplay = new JTextField(10);
  JPanel buttonPanel = new JPanel();



  public CounterFrame (Counter c) {
    setTitle("SwingCounter");
    valuePanel.add(new JLabel("Counter value"));
    valuePanel.add(valueDisplay);
    valueDisplay.setEditable(false);
    getContentPane().add(valuePanel);



    getContentPane().add(buttonPanel);
    pack();
    setVisible(true);
  }
}
```