

# Gliederung

## *Track A (Technik, montags):*

- A1. Eingabe- und Ausgabetechnik bei Rechnersystemen
- A2. Technik der digitalen Bildverarbeitung
- A3. Technik der digitalen Tonverarbeitung
- A4. Technik der digitalen Bewegtbildverarbeitung
- A5. Speichermedien
- A6. Digitale Schnittstellen und Vernetzung im Überblick

## *Track B (Programmierung, freitags)*

- B1. Ein-/Ausgabebetonte Programmierung (Swing)
- B2. 2D-Computergrafik (Java 2D)
- B3. Bildbearbeitung (Java Advanced Imaging)
- B4. Toneinbindung und Tonbearbeitung (Java Sound)
- B5. Frameworks zur Medieneinbindung (Java Media Framework)
- B6. 3D-Computergrafik (Java 3D)
- B7. Web-Programmierung (Applets, Servlets, Java Server Pages)

# B1. Ein-/Ausgabebetonte Programmierung

B1.1 Mensch-Maschine-Kommunikation



B1.2 Modell-Sicht-Paradigma

B1.3 Bausteine für grafische Oberflächen

B1.4 Ereignisgesteuerte Programme

In Track B generell  
gute Java-Kenntnisse  
vorausgesetzt !

# Mensch und Maschine

Sinnesorgane  
"Fürs Überleben ausgestattet"  
Soziales Wesen  
Assoziatives Denken



```
010100011000100010  
010000111001111001  
001100111000010100  
111001001001001000  
...
```

- "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, ...
- „*Ubiquitous Computing*“  
(„Allgegenwärtige“ Rechnerunterstützung):
  - Eingabe: Alltagsgegenstände, Anwesenheit, Bewegungen, ...
  - Ausgabe: Alltagsgegenstände, Beleuchtung, beliebige Geräte, ...
  - Zusätzlich Verwendung diverser mobiler Geräte  
(z.B. PDA, Mobiltelefon, digitaler Musikspieler)

# 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

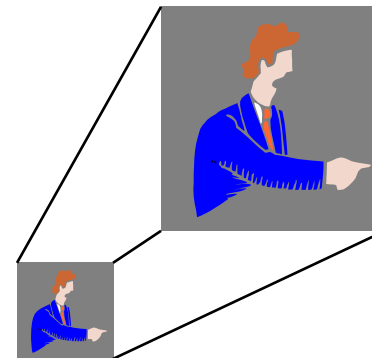
# Software-Ergonomie (*usability*)

- 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

# B1. Ein-/Ausgabebetonte Programmierung

B1.1 Mensch-Maschine-Kommunikation

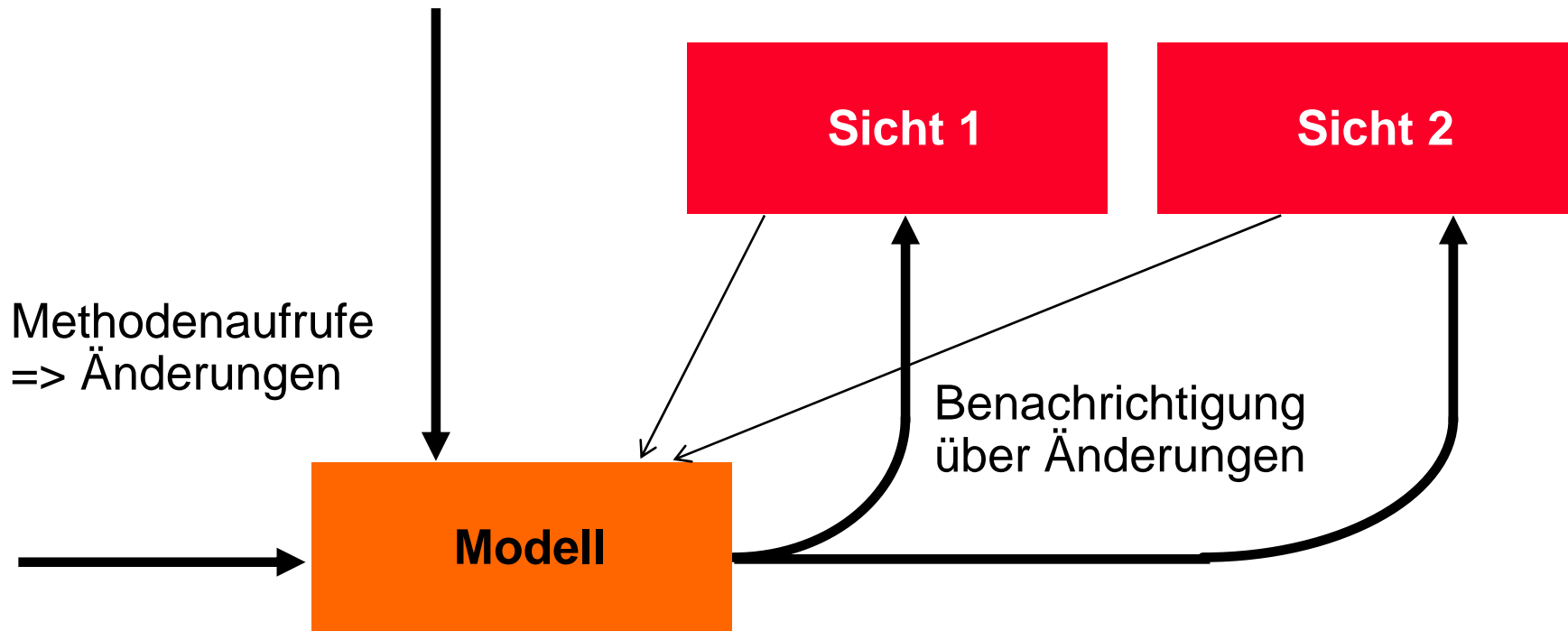
B1.2 Modell-Sicht-Paradigma 

B1.3 Bausteine für grafische Oberflächen

B1.4 Ereignisgesteuerte Programme



# Modell und Sicht

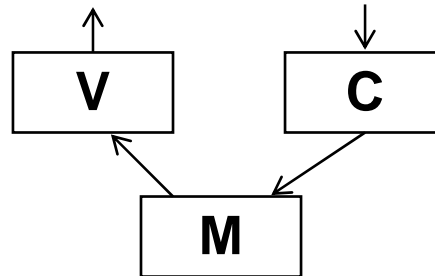


Beispiele: Verschiedene Dokumentenansichten, Statusanzeigen, Verfügbarkeit von Menüpunkten

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

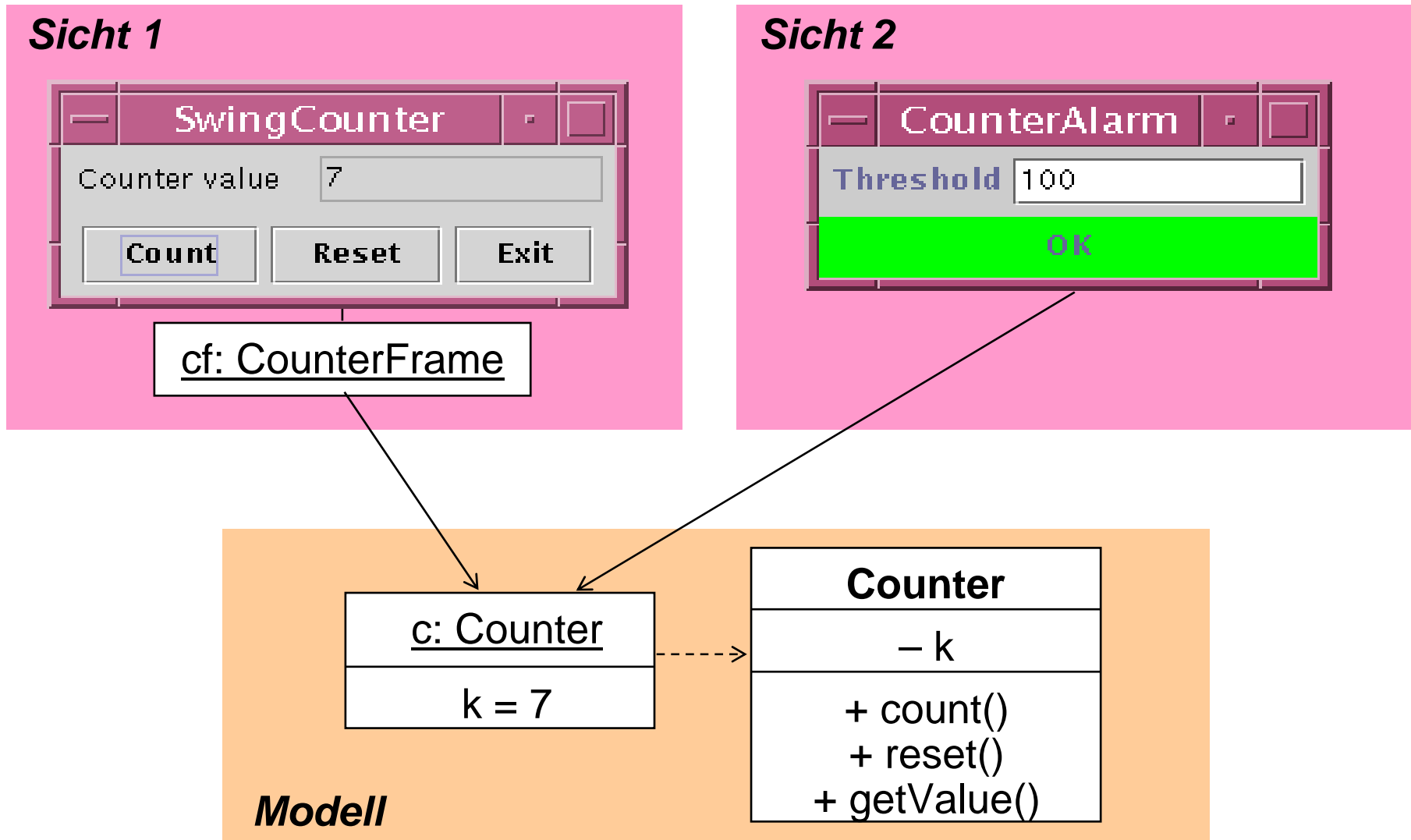
**Muster "Observer"**

# 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

# 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);  
}
```

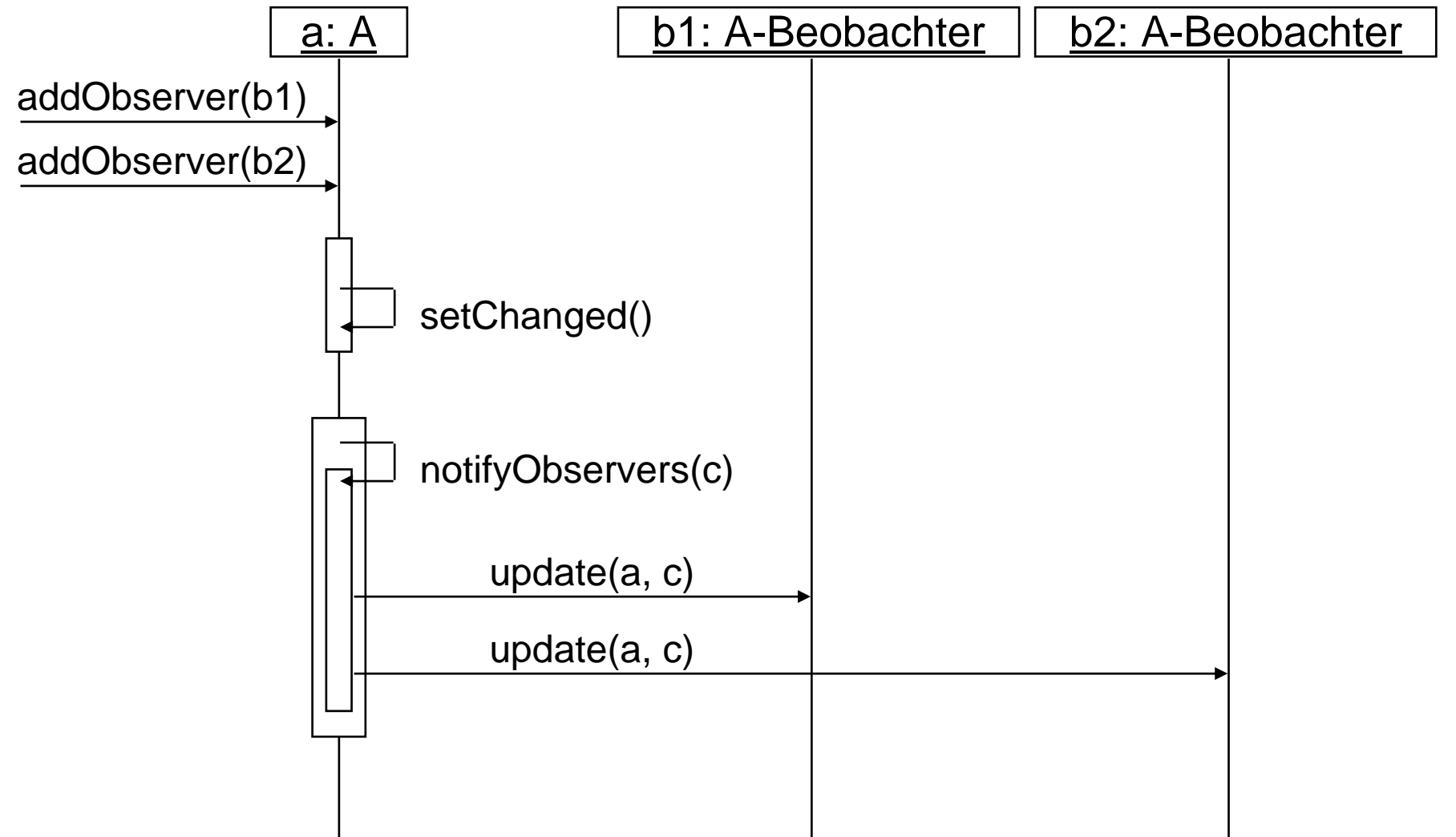
Argumente für notifyObservers():

meist nur Art der Änderung, nicht gesamte Zustandsinformation  
Beobachter können normale Methodenaufrufe nutzen, um sich näher zu informieren.

# Beispielablauf

a extends Observable;

b1, b2 implements Observer;



# B1. Ein-/Ausgabebetonte Programmierung

B1.1 Mensch-Maschine-Kommunikation

B1.2 Modell-Sicht-Paradigma

B1.3 Bausteine für grafische Oberflächen

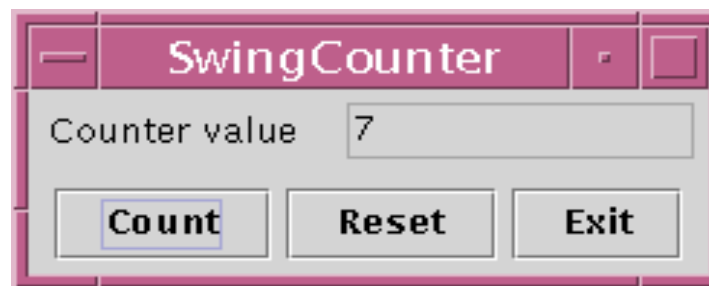


B1.4 Ereignisgesteuerte Programme



# 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)

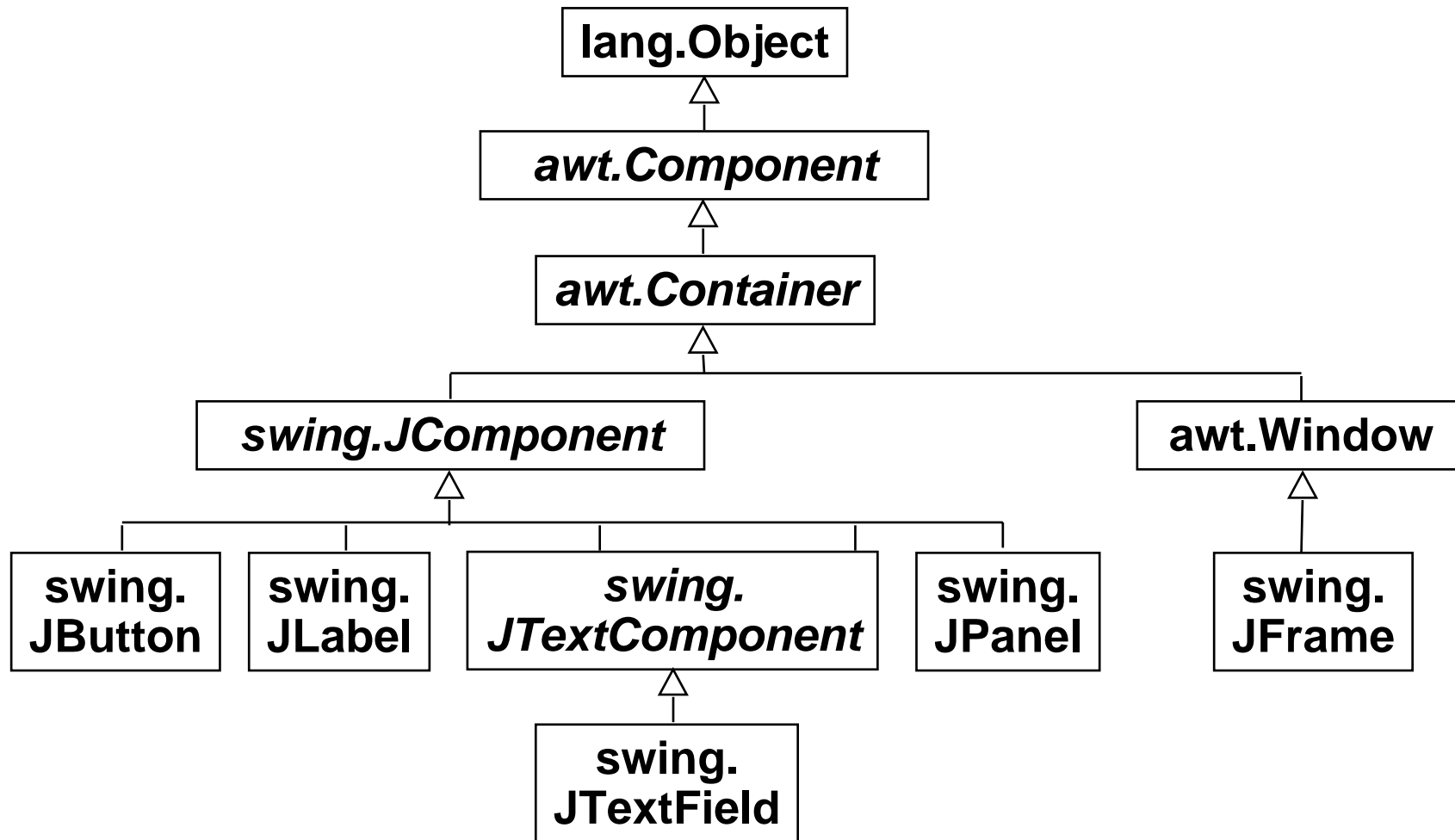


# Bibliotheken von AWT und Swing

- 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, JFrame, JPanel

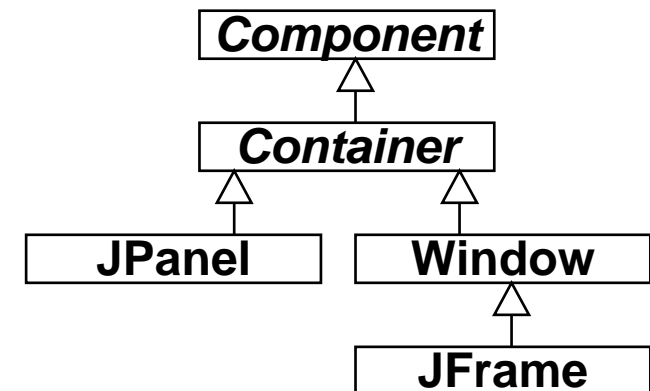
- **awt.Component** (abstrakt):
  - Oberklasse aller Bestandteile der Oberfläche

```
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

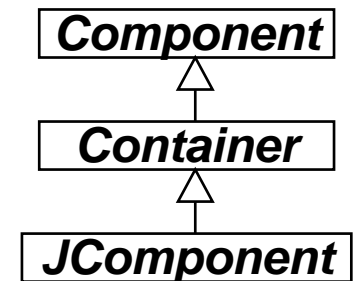
```
public void pack (); //Größe anpassen
```
- **swing.JFrame**
  - Größenveränderbares Fenster mit Titel

```
public void setTitle (String title);
```
- **swing.JPanel**
  - Zusammenfassung von Swing-Komponenten

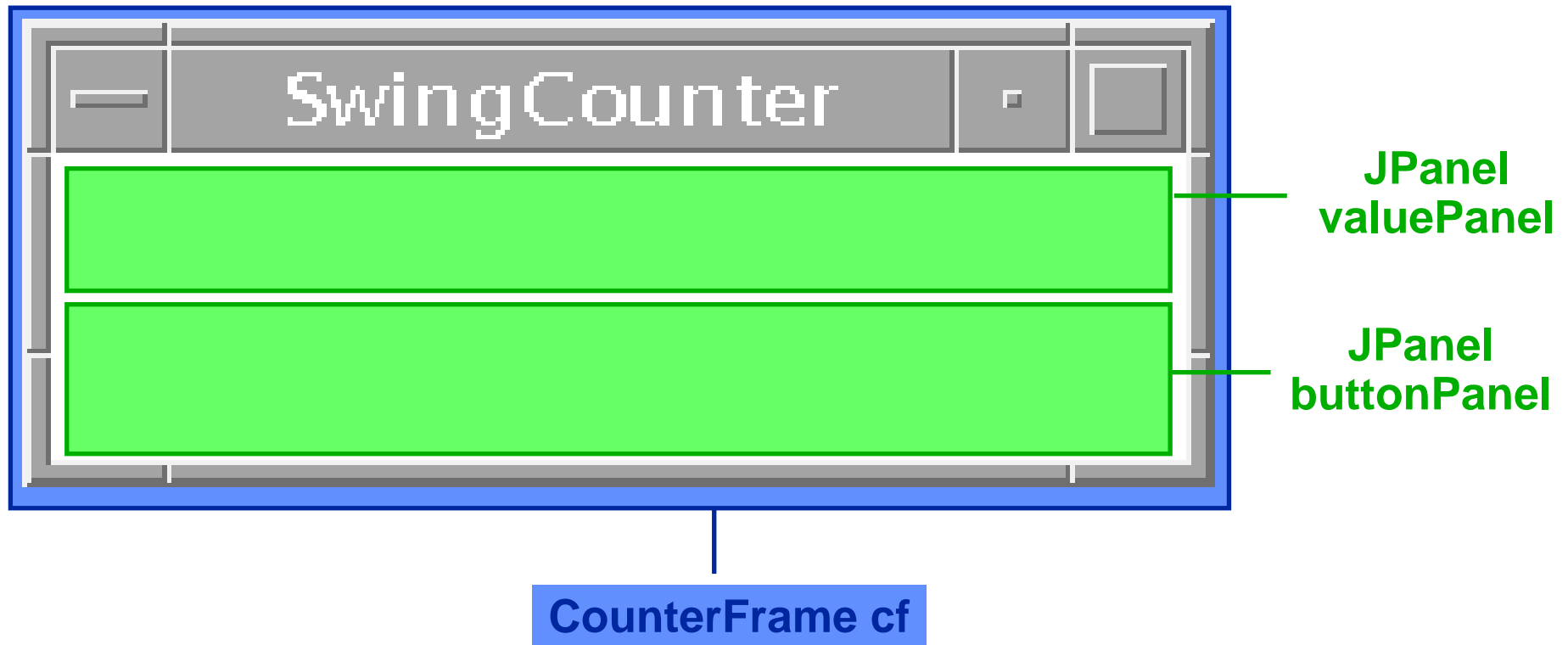


# JComponent

- 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();

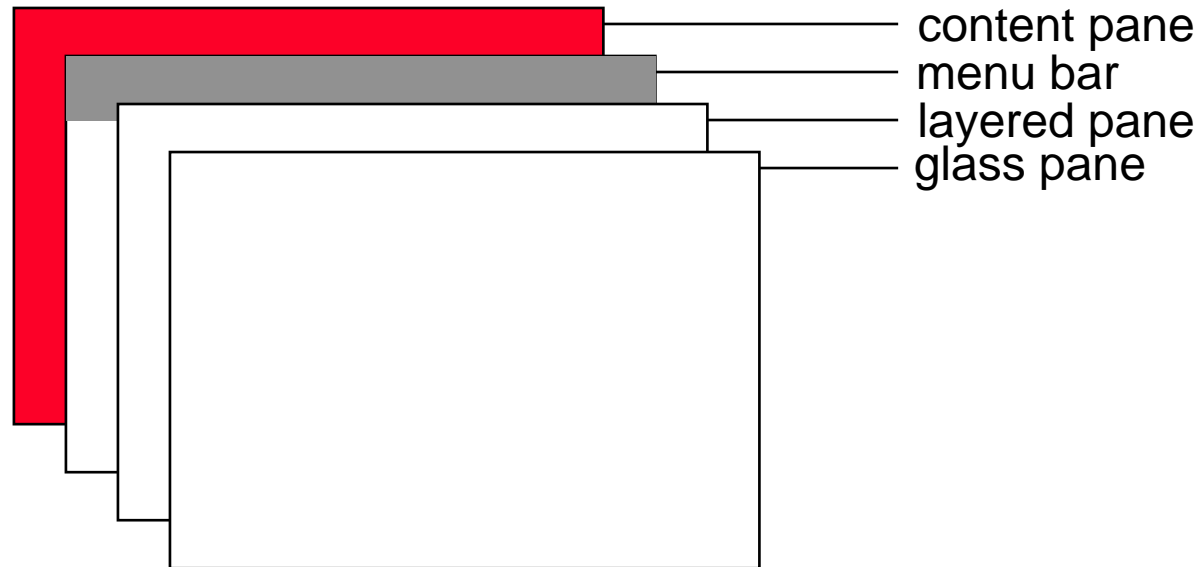
    public CounterFrame (Counter c) {
        setTitle("SwingCounter");

        ... valuePanel zu this hinzufügen

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

# Hinzufügen von Komponenten zu JFrame

- 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();

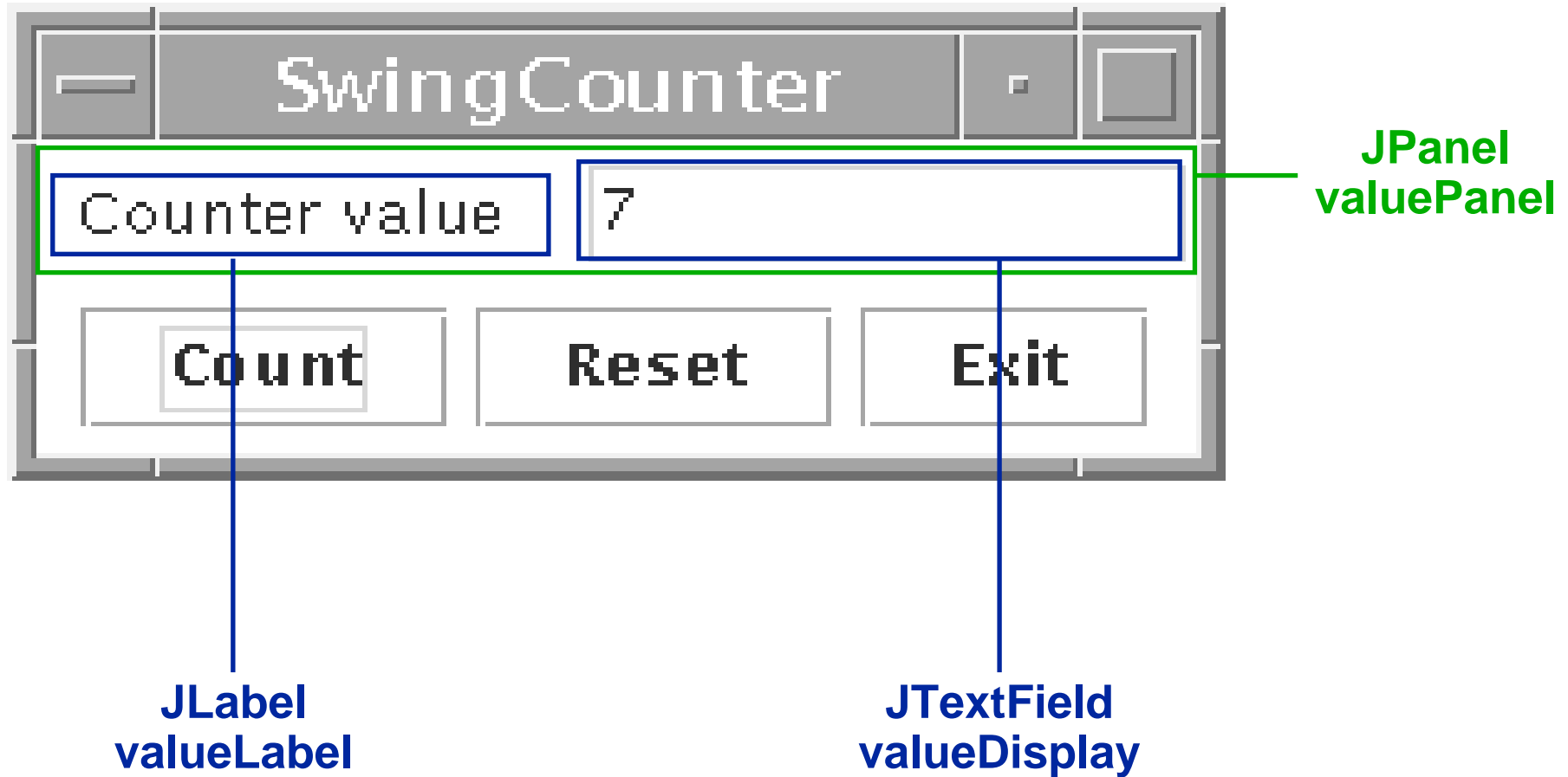
    JPanel buttonPanel = new JPanel();

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

        getContentPane().add(valuePanel);

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

# Zähler-Beispiel: Entwurf der Wertanzeige



# JTextComponent, JTextField, JLabel, JButton

- **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:**

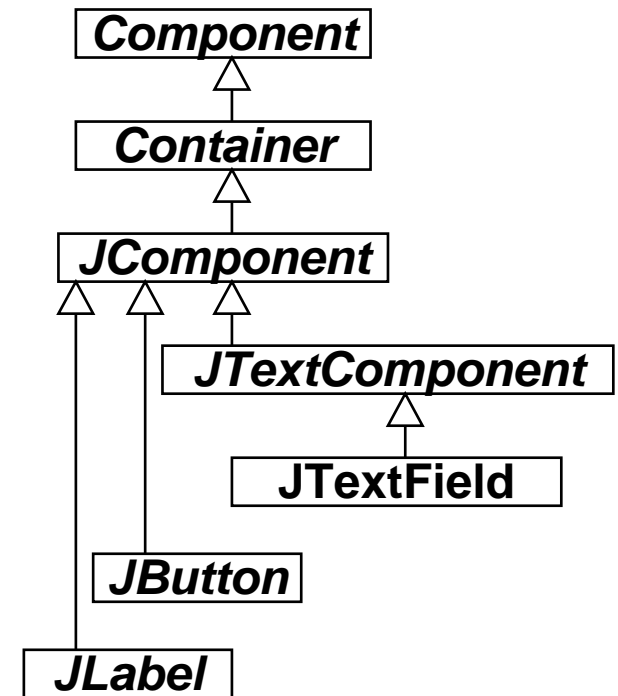
- Einzeiliger unveränderbarer Text

- ```
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);
    }
}
```