Projektwoche
Grafische Nutzerschnittstellen mit Java

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- Raum 2.202
- Tel. 03943 / 659 338

Inhalt

Benutzeroberflächen: Die Java Klassenbibliothek

- Regelmäßige Erweiterung und Veränderung mit den Versionen des JDK
- Teile der Bibliothek werden in der Dokumentation zum JDK als packages bezeichnet
- Objektorientierte Vorteile der Bibliothek
  - Verfeinerung der allgemeine Klassen durch Vererbung der Methoden innerhalb eines package an spezielle Klassen
  - Bereitstellung generischer Datenstrukturen (Templates)

Die Java Klassenbibliothek

- Beispiele
  - java.lang: Sprachunterstüzung (Threads, String)
  - java.util: Hilfeklassen, allgemeine Datenstrukturen (Stack, Vector)
  - java.io: Ein/Ausgabe-Klassen (Dateien, Streams)
  - java.net: Sockets, URL
  - java.applets, java.awt: Java Windows Toolkit
- Erweiterungen (Auswahl)
  - java.swing: AWT mit anpassbaren „Look & Feel“)
  - java.swt: Nachfolger von Swing (Eclipse)
  - speech: Sprachein- und Sprachausgabe
  - JDBC: Datenbankanbindung
  - Beans: wiederverwendbare Komponenten
  - JMF: Java Media Foundation

Klassen

- Eine Klasse ist eine Form für Bausteine.
- Man kann sie benutzen um die Form und Eigenschaften (Funktionen und Attribute) eines Bausteins zu definieren.
- Man muss eigene Bausteine mit Hilfe der Form erzeugen.
- Man kann die Form erweitern.

Klassen 1

Javaprogramm:

```
public class Klasse1 {

   public static void main(String[] args) {
      int summe;
      summe = 0;
      for (int  i=1; i<=100; i++) {
         summe = summe + i;
      }
      System.out.println(summe);
   }  // main

} // class
```

Eigenschaften:

- Daten sind in der Routine
- Berechnung und Ausgabe zusammen

Klassen 2

```
public class Klasse2 {
   static int summe;  //

   public static void calc() {
      summe = 0;
      for (int  i=1; i<=100; i++) {
         summe = summe + i;
      }
   }  // calc

   public static void print() {
      System.out.println(summe);
   }  // print

   public static void main(String[] args) {
      calc();
      print();
   }  // main

} // class
```

Eigenschaften:

- Daten sind in der Routine
- Berechnung und Ausgabe getrennt
- Modularer Aufbau

Klassen 3

```
public class Klasse3 {
   static int summe;  //

   public static void calc() {
      summe = 0;
      for (int  i=1; i<=100; i++) {
         summe = summe + i;
      }
   }  // calc

   public static void print() {
      System.out.println(summe);
   }  // print

   public static void main(String[] args) {
      calc();
      summe += 12;
      print();
   }  // main
} // class
```

Eigenschaften:

- Die Variable „summe“ wird mehrfach benutzt.
- Die Variable „summe“ wird in verschiedenen Kontexten benutzt.
- Berechnung und Ausgabe getrennt
- Modularer Aufbau
- Kein funktionaler Aufbau

Klassen 4

Eigenschaften:

- Berechnung und Ausgabe getrennt
- Modularer Aufbau
- Funktionaler Aufbau
- Schutz von Variablen
- Unabhängigkeit
- Fehlermeldung summe

```
public class klasse4 {

   public static int calc(int n) {
      int summe = 0;
      for (int  i=1; i<=n; i++) {
         summe = summe + i;
      }
      return summe;
   }  // calc

   public static void print(int summe) {
      System.out.println(summe);
   }  // print

   public static void main(String[] args) {
      int s;
      s = calc(100);
      summe += 12;
      print(s);
   }  // main
} // class
```

Klassen 4

Gesucht ein Konstrukt mit folgenden Eigenschaften:

- Modular
- Verstecken der Variablen
- Zugriff auf die Variablen nur über Funktionen
  - get
  - set
- Kommunikation nur über Funktionen
- Änderungen der internen Strukturen ohne Änderung der übergeordneten Programme

Klassen 5

Eine Klasse definiert eine Struktur von Attributen und Methoden.

- Die Instanzen einer Klasse heißen Objekte.
- Die Attribute und Funktionen einer Klasse nennt man ihre Komponenten oder Elemente.

- Attribute sind im wesentlichen Variablen, die zu einem Objekt gehören. Sie definieren Datenelemente, die in jeder Instanz der Klasse vorhanden sind.

- Eine Methode ist eine Prozedur oder Funktion, die zu einer bestimmten Klasse gehört. Die meisten Methoden führen Operationen mit Objekten (Instanzen) durch. Manche Methoden arbeiten jedoch mit den Klassentypen selbst (statisch).

- Eine Methode ist eine Schnittstelle zu den Daten eines Objekts (die oftmals in einem Attribut gespeichert sind). Eigenschaften verfügen über Zugriffsangaben, die bestimmen, wie ihre Daten gelesen und geändert werden. Sie erscheinen für die anderen Bestandteile eines Programms (außerhalb des Objekts) in vielerlei Hinsicht wie ein Attribut.

Klassen 6

- Objekte sind dynamisch zugewiesene Speicherblöcke, deren Struktur durch ihren Klassentyp festgelegt wird. Jedes Objekt verfügt über eine eigene Kopie der in der Klasse definierten Attribute.

- Die Methoden werden jedoch von allen Instanzen gemeinsam genutzt. Das Erstellen und Freigeben von Objekten erfolgt mit Hilfe spezieller Methoden, den Konstruktoren und Destruktoren.

- In Java gibt es keine Destruktoren, da der Garbage Colletor nicht benötigten Speicher automatisch entfernt.

Klassen 7

```
public class kreis
{
      // Attribute der Klasse
    int x;
    int y;
    int radius;

      // Construktor
    public kreis()
    {
        // Initialisierung
      x=1;
      y=1;
      radius=1;
    }

}  // class
```

Klassen 8

```
public class kreis
{
    // Attribute der Klasse
  int x;
  int y;
  int radius;


    // Construktor
  public kreis()
  {
    x=0;
    y=0;
    radius=0;
    System.out.println("kreis create");
  }  // kreis

    // main
  public static void main(String[] args) {
    kreis kr;     // Definition der Variablen kr
    kr = new kreis(); // Erzeugen von kr
  }  // main

}  // class
```

Klassen 9

```
public class kreis
{
    // Attribute der Klasse
  int x;
  int y;
  int radius;


    // Construktor
  public kreis()
  {
    x=0;
    y=0;
    radius=0;
    System.out.println("kreis create");
  }  // kreis

    // main
  public static void main(String[] args) {
    kreis kr;     // Definition der Variablen kr
    kr = new kreis(); // Erzeugen von kr
    kr.x = 2;    // Zugriff erlaubt
  }  // main

}  // class
```

Klassen 10

```
public class kreis
{
    // Attribute der Klasse
  private
    int x;
    int y;
    int radius;


    // Construktor
  public kreis()
  {
    x=0;
    y=0;
    radius=0;
    System.out.println("kreis create");
  }  // kreis
}
```

```
public class run {

    // main
  public static void main(String[] args) {
    kreis kr;     // Definition der Variablen kr
    kr = new kreis(); // Erzeugen von kr
    kr.x = 2;    // Zugriff verboten
  }  // main

}  // class run
```

Klassen 10

```
class kreis
{
    // Attribute der Klasse
  private
    int x;
    int y;
    int radius;


    // Construktor
  public kreis()
  {
    x=0;
    y=0;
    radius=0;
    System.out.println("kreis create");
  }  // kreis

  public void setx(int value) {
    x = value;
  }
}  // class kreis

public class run {

   // main
  public static void main(String[] args) {
    kreis kr;     // Definition der Variablen kr
    kr = new kreis(); // Erzeugen von kr
    kr.setx( 2 );
  }  // main

}  // class run
```

Aufbau einer Klasse: Definition eines Objektes

- public class Klassenname {
- Konstruktor
  public Klassenname () {
  }
- Attribute
  int r,
- Funktionen
  getRadius
  setRadius
  getFlaeche
  zeichneKreisAufLeinwand

Beispiel klasse5

```
class CSumme {
  int summe, n;

  public CSumme(int Anz) {
    n = Anz;
  }

  public int calc() {
     summe = 0;
    for (int  i=1; i<=n; i++) {
       summe = summe + i;
    }
   return summe;
 }  // calc

public void print() {
  System.out.println(summe);
}  // print

}
```

```
public static void main(String[] args) {
    CSumme csum;
    int s;
    csum  = new CSumme(100);
    s = csum.calc();
    csum.print();
}  // main
```

Dialogfenster in Java

Ein Dialogfenster besteht aus einem Frame und weiteren grafischen Benutzerelemente zur Eingabe und Ausgabe von Daten.

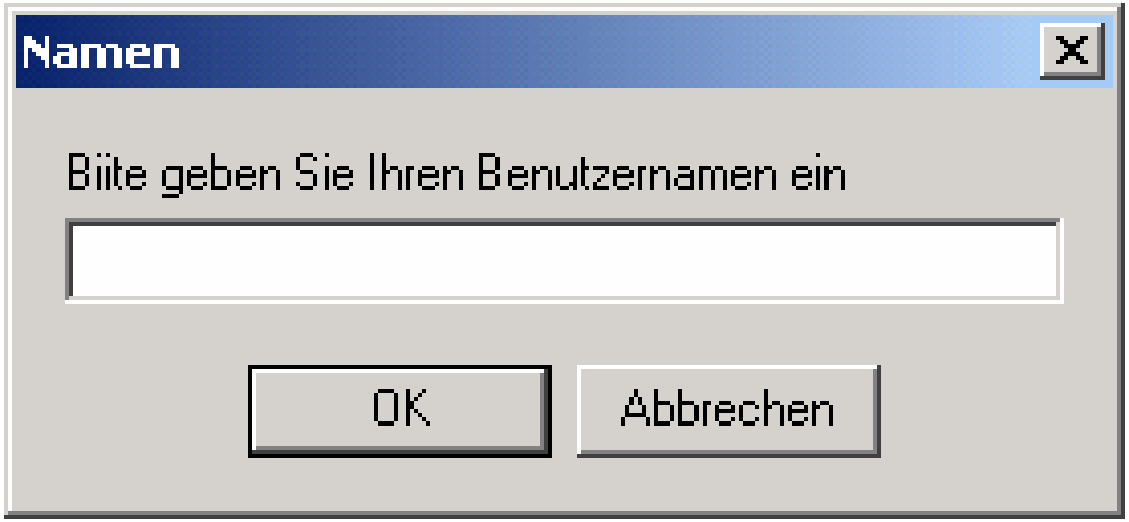
Zum Beenden des Fensters müssen die Schalter „Ok“ und „Abbruch“ vorhanden sein.

Dialogfenster

Ein Dialogfenster ist ein Fenster, in dem der Benutzer Daten eingeben kann. Das Fenster mindestens einen Schalter (Ok). Mit Betätigen wird eine Aktion ausgelöst.

Beispiel: Meldung

Beispiele für Dialogfenster



Eigenschaften:

- Titel
- Aktives Element
- Schalter Abbruch
- Schalter Ok

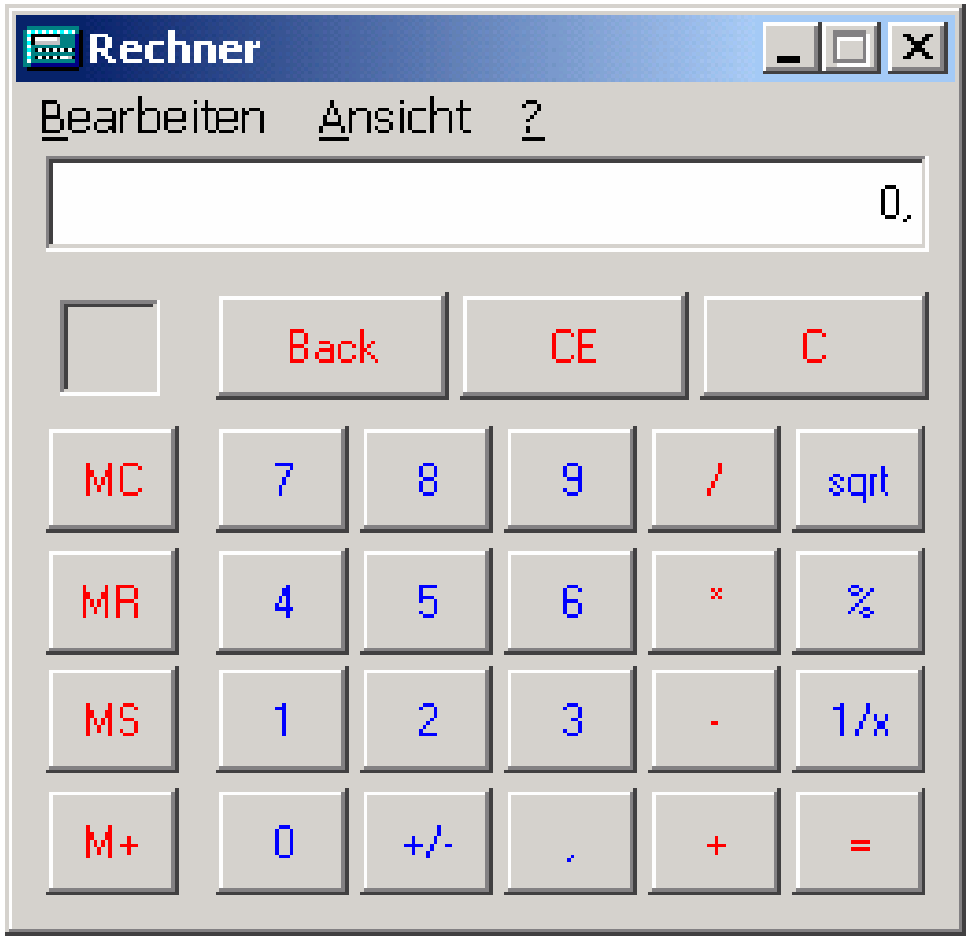
Mit Betätigen des Schalters „Ok“ wird eine weitere Aktion ausgeführt.

Single Dokument Interface

Beispiele:

Calculator	Einfaches Fenster
Notepad	Darstellung eines Textes
Explorer	Zweigeteiltes Fenster
Spiele	Komplexe Darstellung
Internetbrowser	Zweigeteiltes Fenster

Beispiele für SDI-Fenster



Eigenschaften:

- Titel
- Aktives Element
- Schalter Abbruch
- Mehrere Schalter
- Menüs
- Abbruch mit dem Schalter **x**

GUI-Elemente in Java (aktive Elemente)

JLabel	**Anzeigefeld**
JTextField	**einzeiliges Editorfeld**
JButton	**Schalter (Text, Bild)**
JToggleButton	Schalter (Text, Bild)
JRadioButton	**Element zur Auswahl, korrespondiert mit anderen**
JCheckBox	**Element zur Auswahl**
ButtonGroup	Umfasst Checkbox, Radiobuttons
JComboBox	Aufklappbare Liste
JList	**Liste**
JTextArea	**Editor, Scrolling, ASCII**
JTextPane	Editor, RTF, HTM
JEditorPane	Editor, RTF, HTM
JTable	Tabelle
JScrollBar	Scrollbalken
JTree	Anzeige der Elemente in einem Baum
JPane	**Container für weitere GUI-Elemente**

1. Aufbau eines Dialogfenster in Java

```
import javax.swing.JFrame;
// Erzeugt ein normales Fenster ohne Grafik-Elemente

public class Dialog01 extends JFrame {

  //  Haupteinstieg beim Erzeugen des Objektes
  public Dialog01() {

  }
    // hier wird das Objekt "Fenster" erzeugt
  public static void main(String[] args) {
    Dialog01 frame = new Dialog01();
    frame.setVisible(true);
  }
}  // Dialog1
```

Originalframe. Ohne weitere GUI-Elemente, „Keine Größe“

2. Aufbau eines Dialogfenster in Java

```
import javax.swing.JFrame;
// Erzeugt ein normales Fenster ohne Grafik-Elemente

public class Dialog02 extends JFrame {
  //  Haupteinstieg beim Erzeugen des Objektes
  public Dialog02() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(400,300);
    setTitle("Dialog2");
  }
    // hier wird das Objekt "Fenster" erzeugt
  public static void main(String[] args) {
    Dialog02 frame = new Dialog02();
    frame.setVisible(true);
  }
}  // Dialog2
```

Bildschirmgestaltung: Layout

- Die Bildschirmstruktur wird durch ein Containerobjekt oder Layoutverwalter realisiert.
- Darstellung ist unabhängig von der Größe des Fensters
- Layout stellt eine Beziehung zwischen den Elementen dar.
- Setzen des Layout-Verwalters mit setLayout().

Layout-Verwalter:

BorderLayout

GridLayout

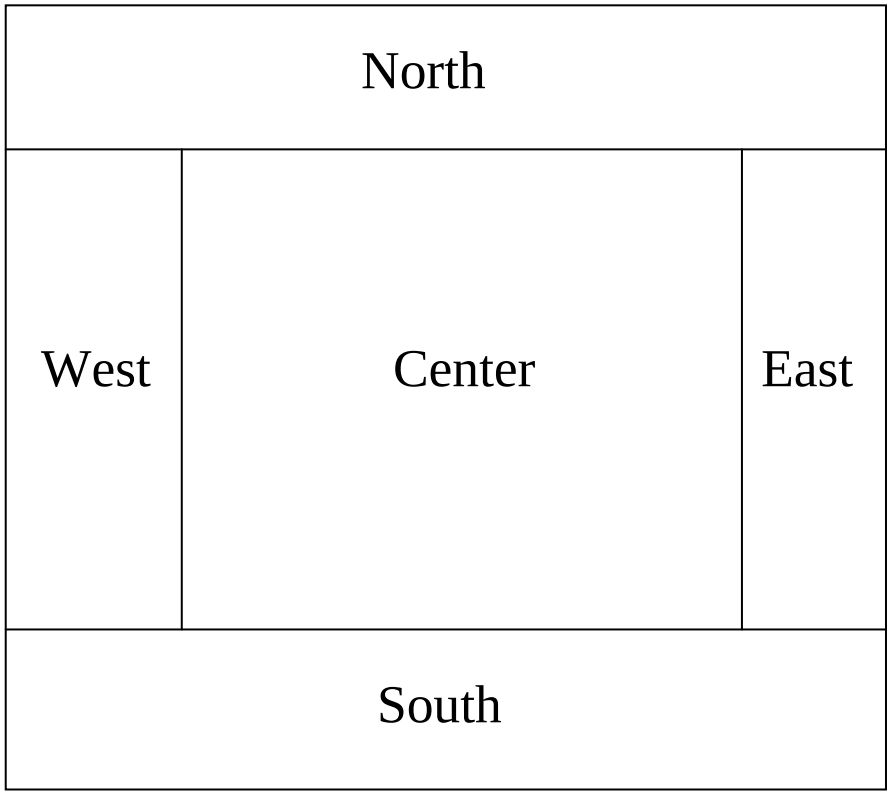
FlowLayout

CardLayout

BoxLayout

GridBagLayout

Border-Layout: Grundstruktur / Aufbau



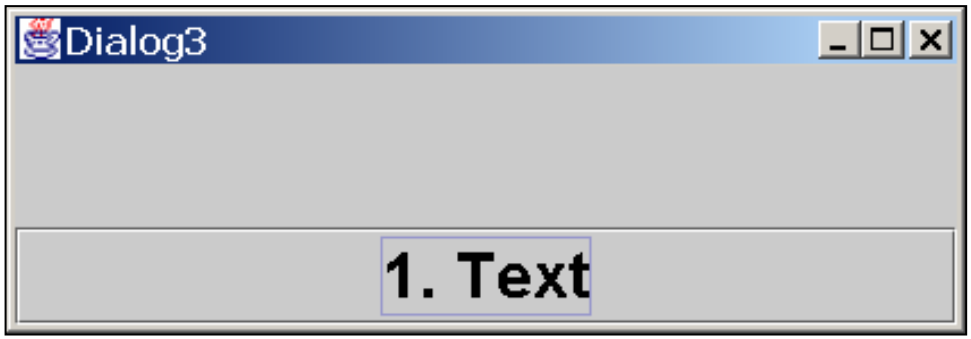
In jedem Feld kann genau ein Element eingefügt werden.

Der Standard-Layoutmanager von JFrame ist das Border-Layout.

Dialogfenster in Java, mit einem Schalter

Ergebnis: Schalter hat die gesamte Breite

```
public class Dialog03 extends JFrame {
  //  Haupteinstieg beim Erzeugen des Objektes
  public Dialog03() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(400,200);
    setTitle("Dialog3");
    setGUI();
  }

  public void setGUI() {
        JButton bn1;              // Schalter definieren
        bn1 = new JButton();    // Schalter erzeugen
        bn1.setText("1. Text");              // Text setzen
        getContentPane().add(bn1, BorderLayout.SOUTH);
  }
```



Dialogfenster in Java, mit einem Label

```
public class Dialog04 extends JFrame {
  public Dialog04() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(400,200);
    setTitle("Dialog4");
    setGUI();
  }
  public void setGUI() {
    JLabel label1;                          // Label Objekt definieren
    label1 = new JLabel();                  // Label erzeugen
    label1.setText("Dies ist ein Anzeigetext");//  Labeltext
    label1.setFont( new Font("Arial", Font.BOLD,28));
    this.getContentPane().add(label1, BorderLayout.NORTH);   // Element einfügen
  }
```

Dialogfenster in Java, JLabel Methoden

Einige Methoden:

void setHorizontalAlignment(int alignment)

Sets the alignment of the label's contents along the X axis.

void setHorizontalTextPosition(int textPosition)

Sets the horizontal position of the label's text, relative to its image.

void setText(String text)

void setVerticalAlignment(int alignment)

Sets the alignment of the label's contents along the Y axis.

void setVerticalTextPosition(int textPosition)

Sets the vertical position of the label's text, relative to its image.

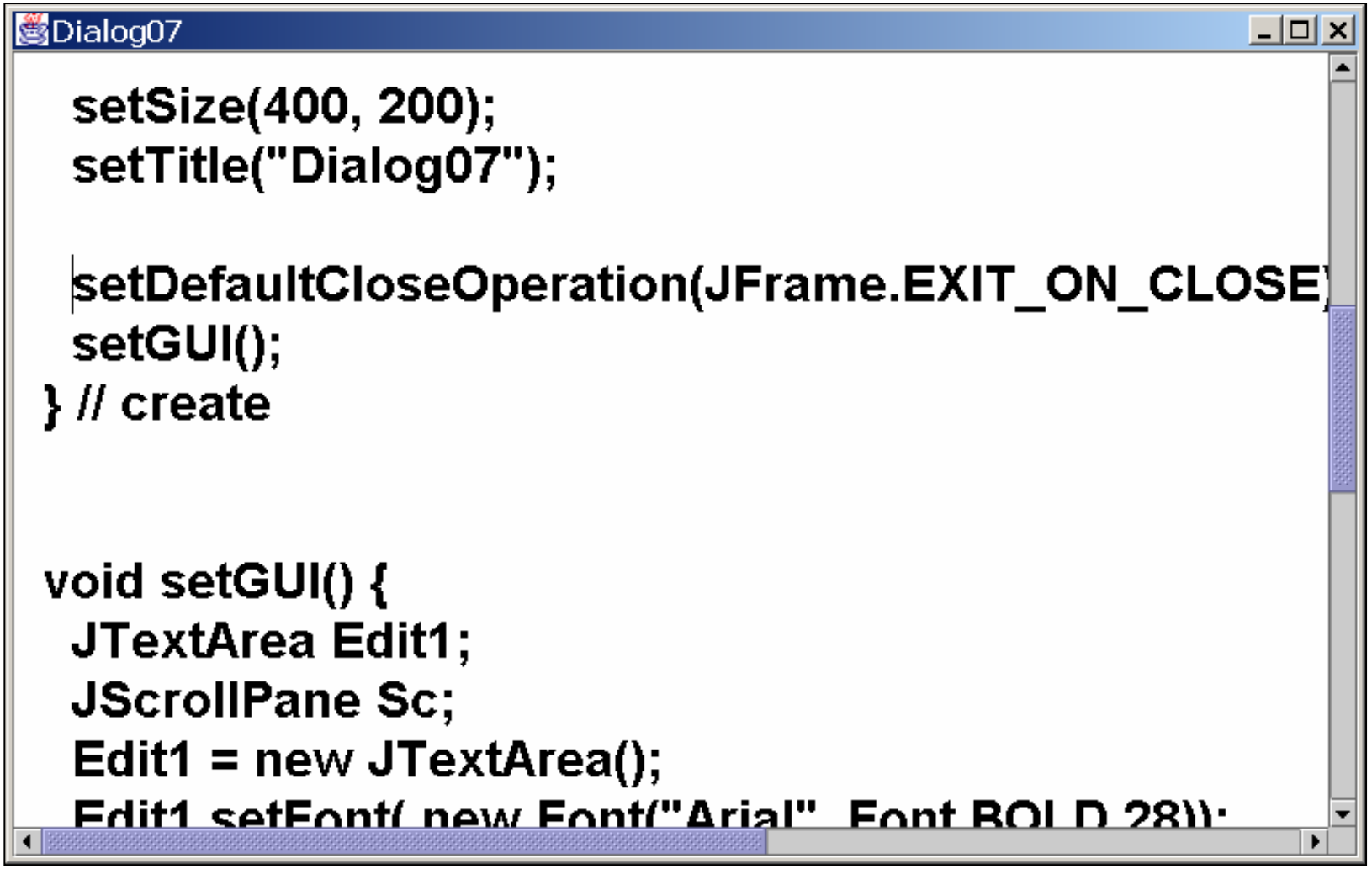
Dialogfenster in Java, mit einem Texteditor

```
public class Dialog05 extends JFrame {
   public Dialog05() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(400,200);
    setTitle("Dialog05");
    setGUI();
   }
   public void setGUI() {
    JTextField Edit1;
    Edit1 = new JTextField();
    Edit1.setFont( new Font("Arial", Font.BOLD,28));
    Edit1.setText("Hallo Studenten");
    getContentPane().add( Edit1, BorderLayout.NORTH);
   }
```

Dialogfenster in Java, mit einem Editor

```
public Dialog06() {
    setSize(400, 200);
    setTitle("Dialog06");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setGUI();
  } // create


  void setGUI() {
    JTextArea Edit2;
    Edit2 = new JTextArea();
    Edit2.setFont( new Font("Arial", Font.BOLD,28));
    Edit2.setText("Hallo Studenten");
    getContentPane().add( Edit2, BorderLayout.CENTER);
  }
```

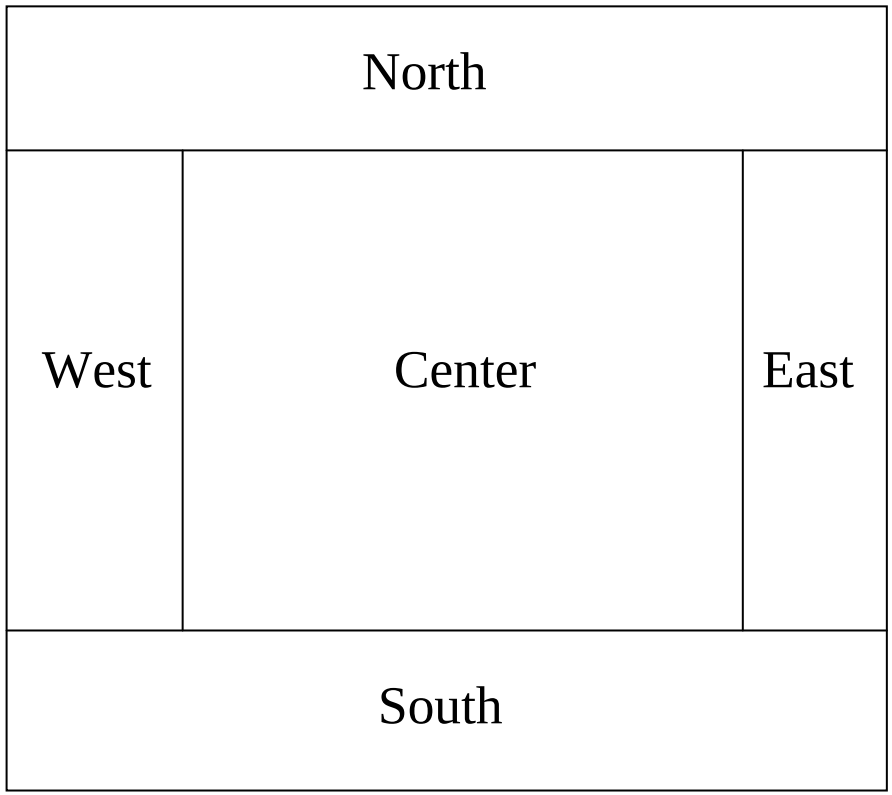
Beispielbild eines Editors mit Scrollbalken

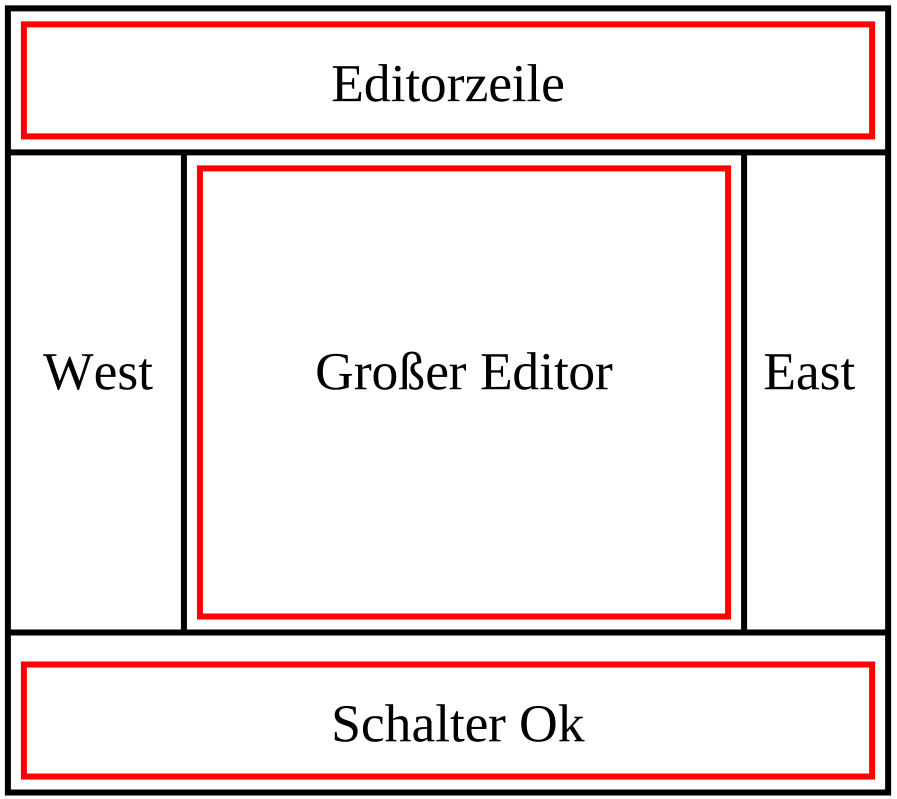


Editor mit Scrollbalken: Ausgangspunkt

```
public Dialog07() {
  JTextArea Edit1;
  setSize(400, 200);
  setTitle("Dialog07");
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  setGUI();
 } // create

 void setGUI() {
  Edit1 = new JTextArea();
  Edit1.setFont( new Font("Arial", Font.BOLD,28));
  Edit1.setText("Hallo Studenten");
  getContentPane().add( Edit1, BorderLayout.CENTER);
 }
```

Komponenten im Layout: Vollständig ?



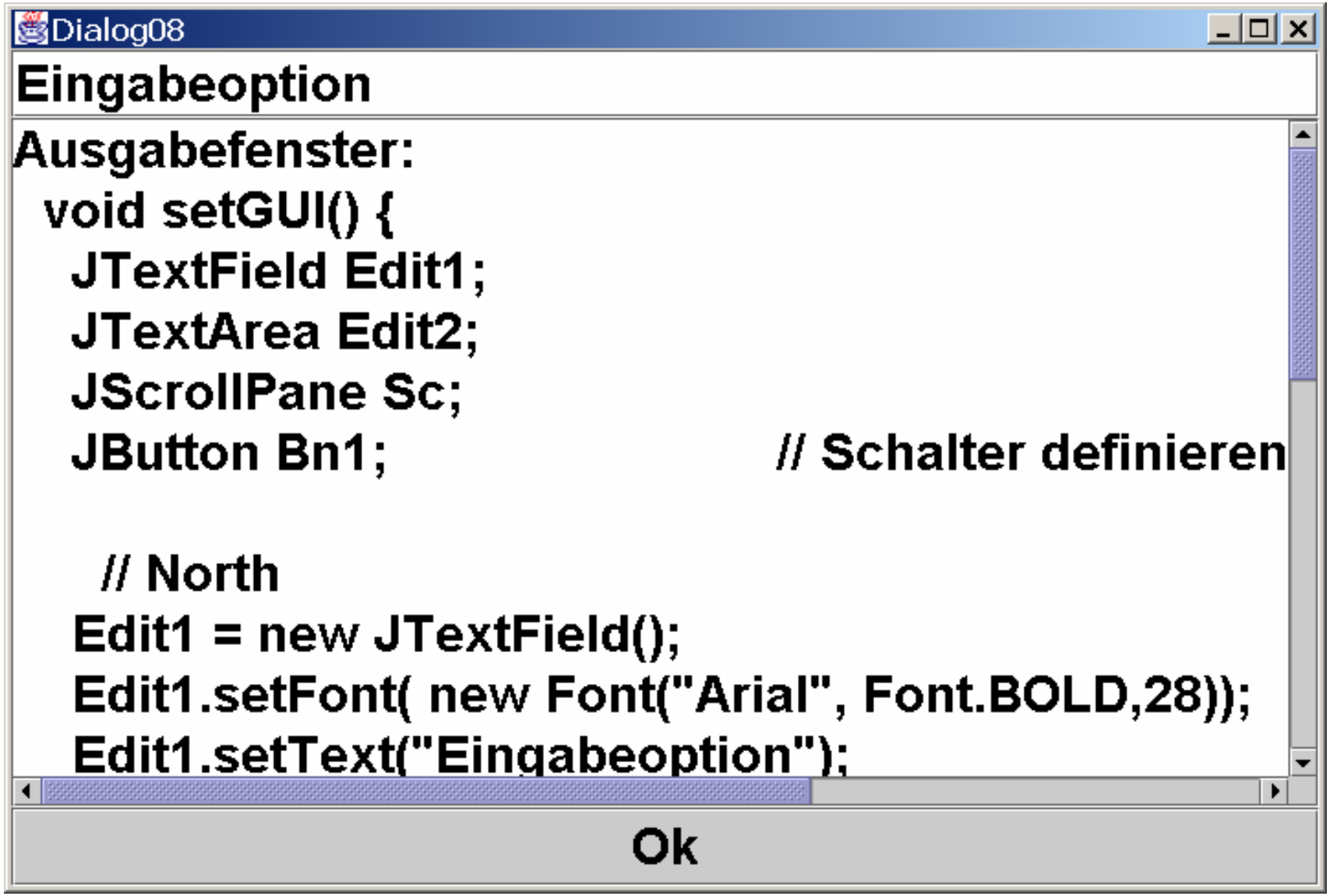


Vollständiges Beispiel: Quellcode

```
void setGUI() {
   JTextField Edit1;
   JTextArea Edit2;
   JScrollPane Sc;
   JButton Bn1;

    // North
   Edit1 = new JTextField();
   Edit1.setFont( new Font("Arial", Font.BOLD,28));
   Edit1.setText("Eingabeoption");
   getContentPane().add( Edit1, BorderLayout.NORTH);
```

Vollständiges Beispiel:

```
    // Center
  Edit2 = new JTextArea();
  Edit2.setFont( new Font("Arial", Font.BOLD,28));
  Edit2.setText("Ausgabefenster\nHier ist der Editor\nund dies ist die 3. Zeile");
  Sc = new JScrollPane(Edit2);
  getContentPane().add( Sc, BorderLayout.CENTER);

    // South
  Bn1 = new JButton();    // Schalter erzeugen
  Bn1.setText("Ok");      // Text setzen
  Bn1.setFont( new Font("Arial", Font.BOLD,28));
  this.getContentPane().add(Bn1, BorderLayout.SOUTH);

}
```


Dialog08
Eingabeoption
Ausgabefenster:
void setGUI() {
JTextField Edit1;
JTextArea Edit2;
JScrollPane Sc;
JButton Bn1; // Schalter definieren
// North
Edit1 = new JTextField();
Edit1.setFont(new Font("Arial", Font.BOLD,28));
Edit1.setText("Eingabeoption");
Ok

Eigenschaften dieser Lösung

- Die Editorzeile hat keine Bezeichnung
- Es gibt nur einen Schalter
- Besser wäre Ok und Abbruch
- Ein Abstand zwischen den Elementen wäre besser

Abhilfe:

- GridBagLayout, kompliziert aber bestens geeignet
- Vertikale Box
- Horizontale Box

Layout Box

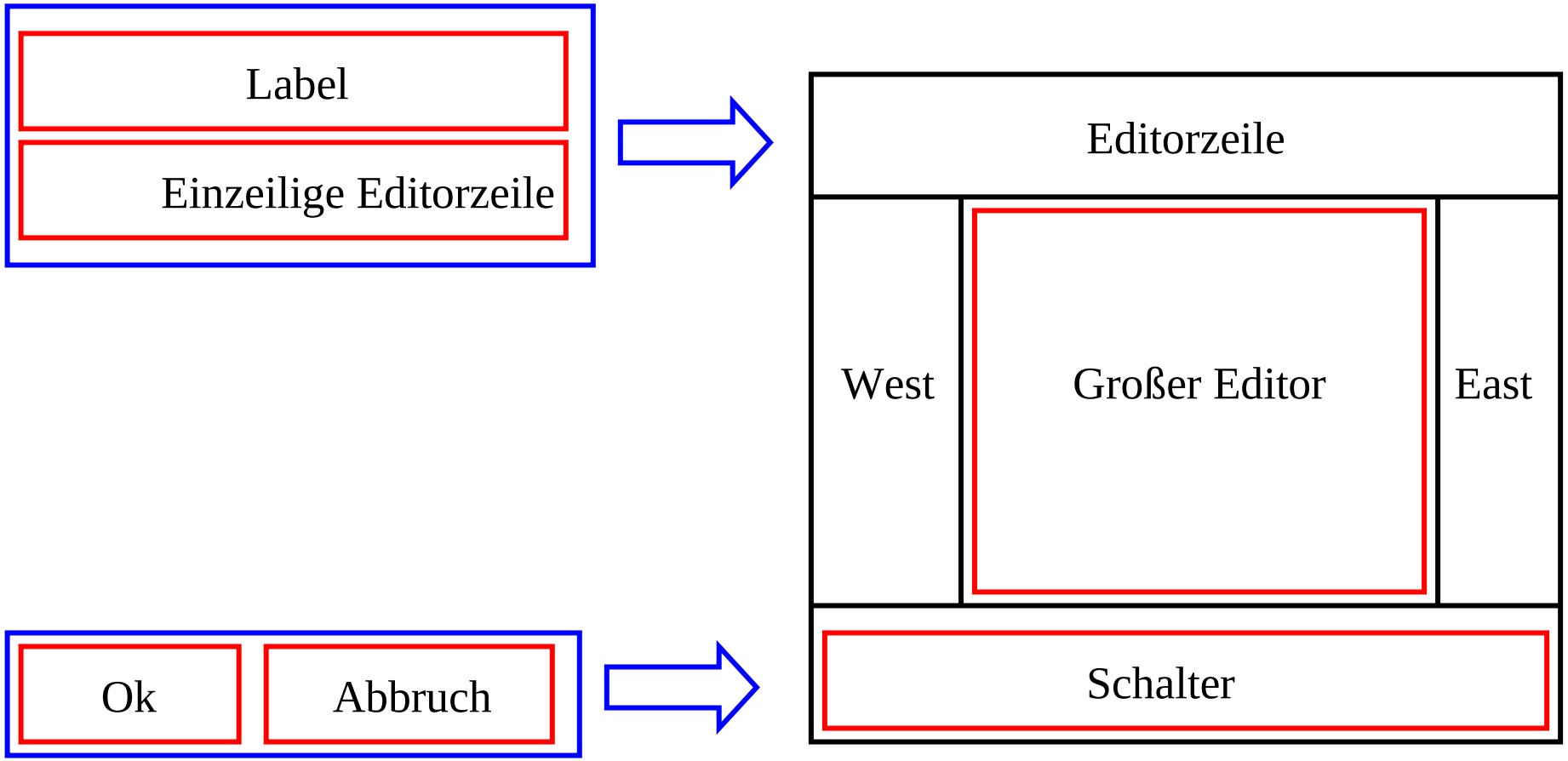
- Das Layout-Element Box kann mehrere Elemente darstellen.
- Alle Elemente haben die gleiche Größe
- Mögliche Ausrichtungen:
  - horizontal
  - vertikal

Vollständiges Beispiel:

```
void  setGUI() {
   Box box1;
   JTextField Edit1 = new JTextField();
   JTextField Edit2 = new JTextField();
   JTextField Edit3 = new JTextField();
   JTextField Edit4 = new JTextField();
   JLabel Label1 = new JLabel();
   box1 = Box.createVerticalBox();
   Edit1.setText("Müller");
   Edit1.setFont( new Font("Arial", Font.BOLD,28));
   Edit2.setText("Meyer");
   Edit2.setFont( new Font("Arial", Font.BOLD,28));
   Label1.setText("Hier ist ein Label");
   Label1.setFont( new Font("Arial", Font.BOLD,28));
   Edit3.setText("Schmidt");
   Edit3.setFont( new Font("Arial", Font.BOLD,28));
   Edit4.setText("Schulze");
   Edit4.setFont( new Font("Arial", Font.BOLD,28));
   box1.add(Edit1) ;
   box1.add(Edit2) ;
   box1.add(Label1) ;
   box1.add(Edit3) ;
   box1.add(Edit4) ;
   this.getContentPane().add(box1, BorderLayout.CENTER);
}
```

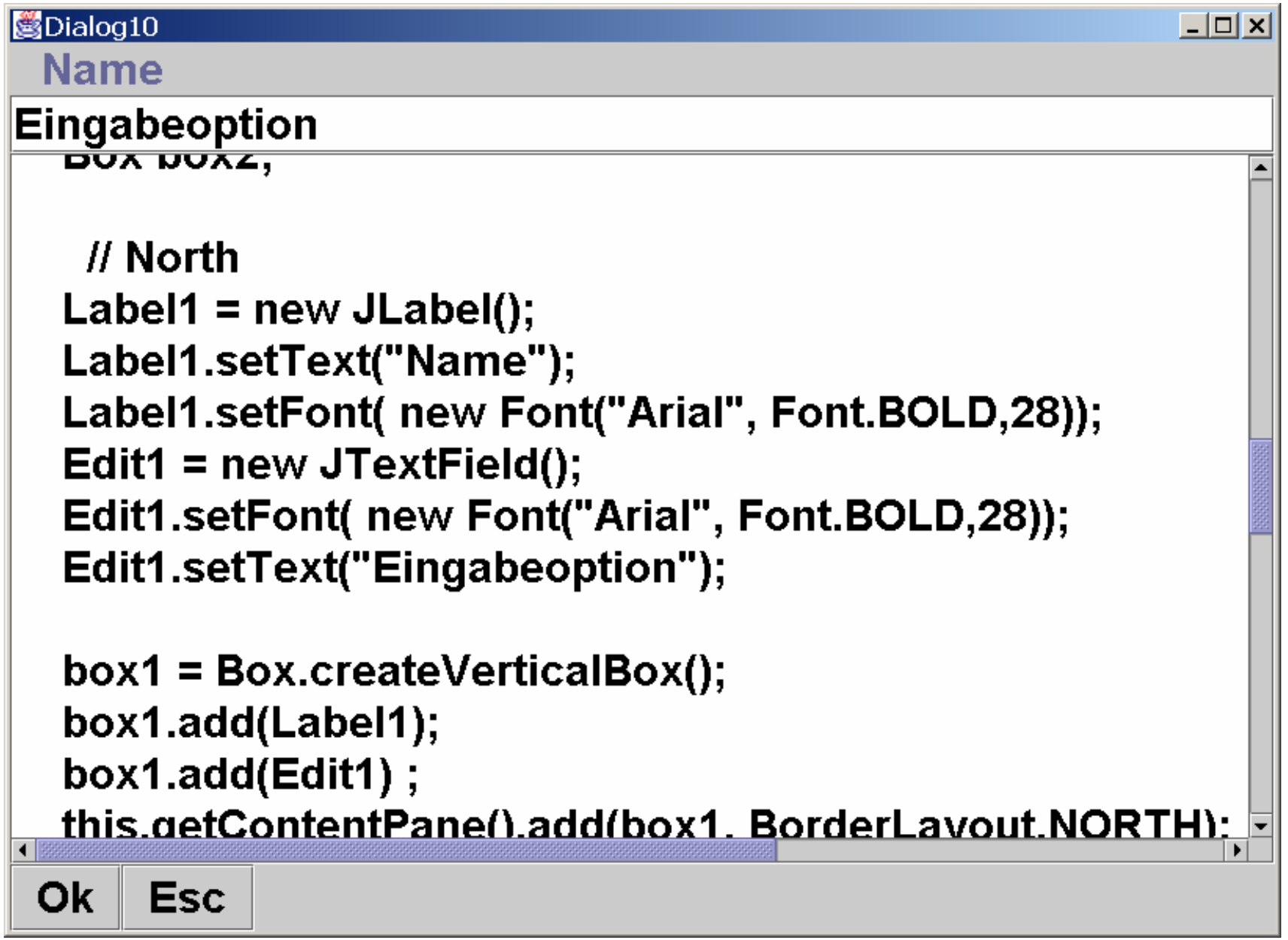
```
Box box1 = Box.createVerticalBox();
JLabel label1;                                  // Label Objekt definieren
    label1 = new JLabel();                              // Label erzeugen
    label1.setText("Dies ist ein Anzeigetext");//  Labeltext
    label1.setFont( new Font("Arial", Font.BOLD,28));


Edit1 = new JTextField();
    Edit1.setFont( new Font("Arial", Font.BOLD,28));
    Edit1.setText("Eingabeoption");

box1.add(Label1) ;
 box1.add(Edit1) ;

    this.getContentPane().add(box1, BorderLayout.NORTH);
  }
```

Vollständiges Beispiel:



```
void setGUI() {
   JLabel Label1;          JTextField Edit1;
   JTextArea Edit2;        JScrollPane Sc;
   JButton BnOk;           JButton BnEsc;
   Box box1;               Box box2;

    // North, 1. Definition
   Label1 = new JLabel();
   Label1.setText("Name");
   Label1.setFont( new Font("Arial", Font.BOLD,28));
   Edit1 = new JTextField();
   Edit1.setFont( new Font("Arial", Font.BOLD,28));
   Edit1.setText("Eingabeoption");
    // 2. Einfügen
   box1 = Box.createVerticalBox();
   box1.add(Label1);
   box1.add(Edit1) ;
   this.getContentPane().add(box1, BorderLayout.NORTH);
```

```
        // Center, bekannt
  Edit2 = new JTextArea();
  Edit2.setFont( new Font("Arial", Font.BOLD,28));
  Edit2.setText("Ausgabefenster\nHier ist der Editor\nund dies ist die 3. Zeile");
  Sc = new JScrollPane(Edit2);
  getContentPane().add( Sc, BorderLayout.CENTER);

   // South, 1. GUI erzeugen
  BnOk = new JButton();              // Schalter erzeugen
  BnOk.setText("Ok");     // Text setzen
  BnOk.setFont( new Font("Arial", Font.BOLD,28));
  BnEsc= new JButton();              // Schalter erzeugen
  BnEsc.setText("Esc");   // Text setzen
  BnEsc.setFont( new Font("Arial", Font.BOLD,28));
   // South, 2. GUI einfügen
  box2 = Box.createHorizontalBox();
  box2.add(BnOk);
  box2.add(BnEsc) ;
  this.getContentPane().add(box2, BorderLayout.SOUTH);
}
```


Dialog10
Name
Eingabeoption
Box box2;
// North
Label1 = new JLabel();
Label1.setText("Name");
Label1.setFont(new Font("Arial", Font.BOLD,28));
Edit1 = new JTextField();
Edit1.setFont(new Font("Arial", Font.BOLD,28));
Edit1.setText("Eingabeoption");
box1 = Box.createVerticalBox();
box1.add(Label1);
box1.add(Edit1) ;
this.getContentPane().add(box1, BorderLayout.NORTH);
Ok
Esc

Eigenschaften dieser Lösung

- Lösung fast optimal
- Man kann die Editorzeile beschriften
- Es gibt beliebig viele Schalter
- Es fehlt ein Abstand zwischen den Elementen

Abhilfe:

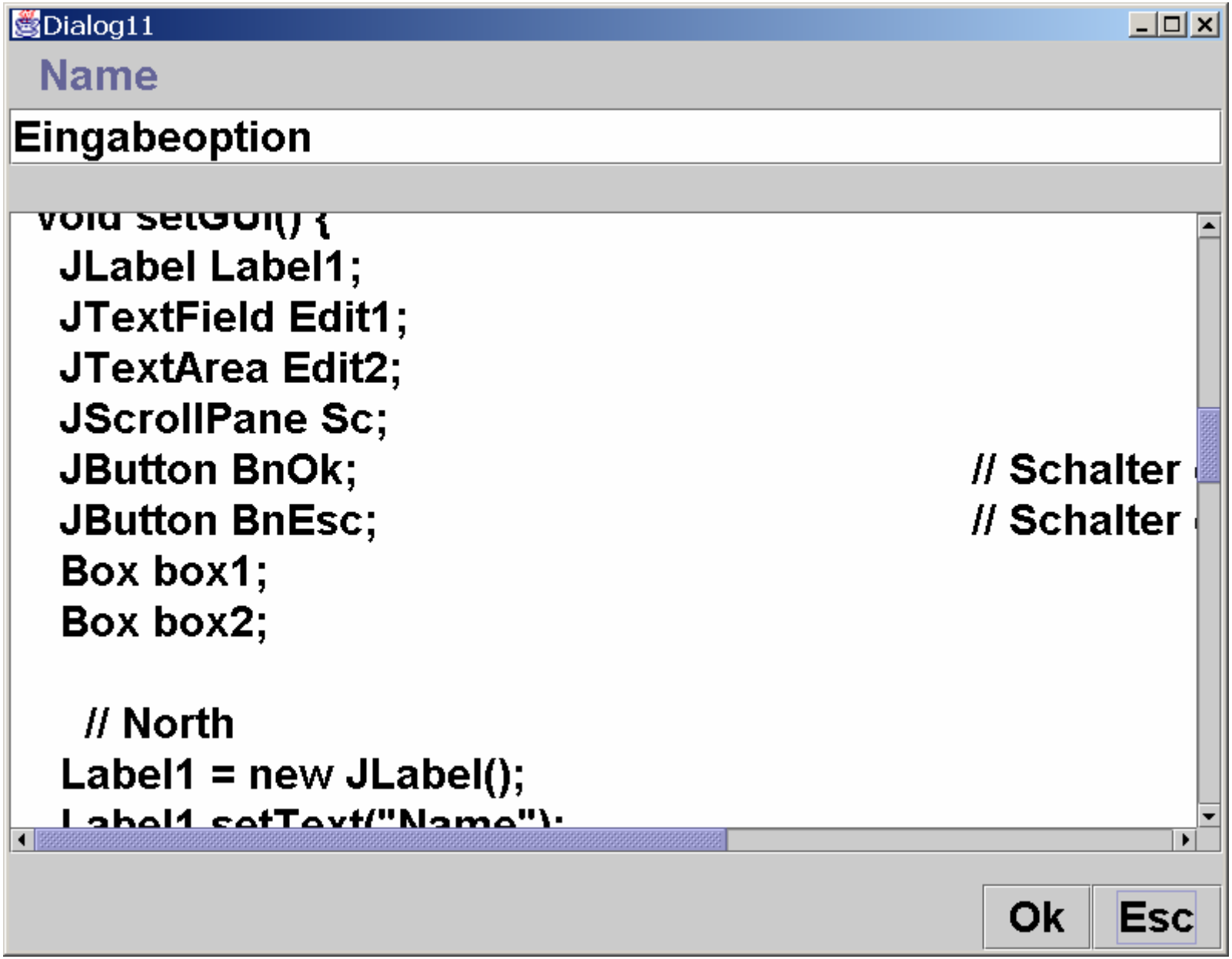
- Box.createHorizontalStrut(int width)
- Box.createVerticalStrut(int height)

Beide Methoden liefern ein nicht sichtbares Element

Man muss aber diesen Abstand nicht in die Box einfügen; der Bereich wird automatisch angefügt.

Statische Methoden

```
box1 = Box.createVerticalBox();
   box1.add(  Box.createVerticalStrut(5) );    // 5 Pixel
   box1.add(Label1);
   box1.add(  Box.createVerticalStrut(5) );
   box1.add(Edit1) ;
   box1.add(  Box.createVerticalStrut(30) );
   this.getContentPane().add(box1, BorderLayout.NORTH);

// South
   BnOk = new JButton();              // Schalter erzeugen
   BnOk.setText("Ok");     // Text setzen
   BnOk.setFont( new Font("Arial", Font.BOLD,28));
   BnEsc= new JButton();              // Schalter erzeugen
   BnEsc.setText("Esc");   // Text setzen
   BnEsc.setFont( new Font("Arial", Font.BOLD,28));
   box2 = Box.createHorizontalBox();
   box2.add(  Box.createVerticalStrut(40) );
   box2.add(BnOk);
   box2.add(BnEsc) ;
   this.getContentPane().add(box2, BorderLayout.SOUTH);
```


Dialog11
Name
Eingabeoption
void setGUI() {
JLabel Label1;
JTextField Edit1;
JTextArea Edit2;
JScrollPane Sc;
JButton BnOk; // Schalter
JButton BnEsc; // Schalter
Box box1;
Box box2;
// North
Label1 = new JLabel();
Ok
Esc

Ereignisse:

Ereignisse durch ein GUI-Element werden durch folgende Aktionen ausgelöst.

- Ein GUI-Element angeklickt
- Der Inhalt einer Textzeile wird verändert
- Die Bezeichnung eines JLabels wird verändert
- Mausklick auf einen Schalter
- Auswahl eines Menüs
- Auswahl eines Popupmenüs
- Maus über einem Element
- Maustaste wurde gedrückt
- Maustaste wurde losgelassen
- Taste wurde gedrückt
- Taste wurde losgelassen
- etc

Ereignisse:

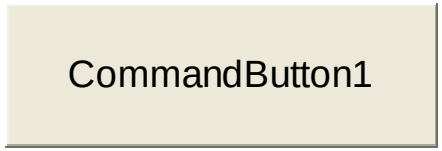
Klassen der verschiedenen Ereignisse: (Anzeige aus dem JBuilder)

- actionPerformed (CR)
- ancestorAdded
- ancestorMoved
- ancestorRemoved
- caretPositionChanged
- componentAdded
- componentHidden
- componentMoved
- componentRemoved
- componentResized
- componentShown
- focusGained
- focusLost
- inputMethodTextChanged
- itemStateChanged
- Keypressed
- keyReleased
- keyTyped
- mouseClicked
- mouseDragged
- mouseEntered
- mouseExited
- mouseMoved
- mousePressed
- mouseReleased
- propertyChange
- stateChange
- vetoableChange

Ereignisse:

Beispiel Schalter:

- Ein Schalter hat für jede „Aktion“ eine Liste mit möglichen Empfängern.
- Um ein bestimmtes Ereignis zu erhalten, muss man sich beim Schalter, der die Ereignisse überwacht, registrieren lassen (ActionListener).
- Der Schalter ruft nun durch die Definition des Listener die interne Funktion auf.
- Diese internen Funktionen werden nacheinander aufgerufen
- (Verkettete Liste).



Beispiel Schalter:

```
    Bn = new JButton( "Ok" );

    ActionListener al;
    al = new java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            Bn_click();
        }
    };
    Bn.addActionListener(al);
}  // setGUI

 void Bn_click() {
   setTitle("Der Schalter wurde gedrückt");
  }
```

Der blauer Text definiert die interne Funktion des Actionlisteners.

Eingabe- Ausgabe einer Zahl

- Die Editorzeile liest nur Zeichenketten ein.
- Deshalb müssen die Eingaben in eine Zahl konvertiert werden.
- Bei der Ausgabe muss die Zahl wieder in einen String transformiert werden.

Umwandlung Integerzahl in eine Zeichenkette:

```
public String IntToStr(int i) {
    return Integer.toString(i);
  }
```

```
// sehr vereinfacht
public long LVal(String sText) {
        long result =-999;
        try {
            result = Long.valueOf(sText).longValue();
        }
        catch (NumberFormatException e) {
          result = -999;
        }
        return result;
  } // LVal
```

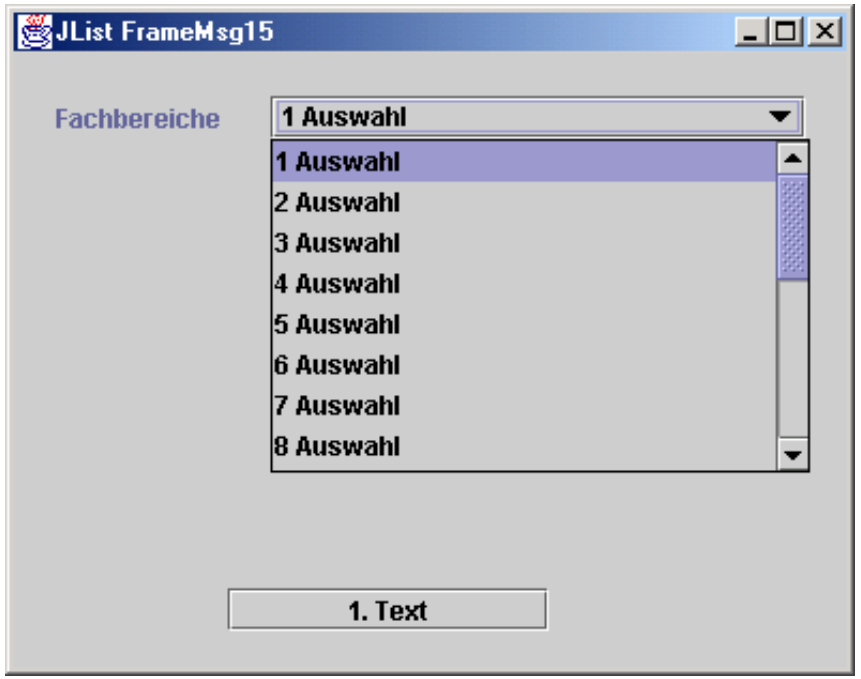
Vollständiges Beispiel mit Berechnung der Fakultät

- Problem verstehen
- Problem lösen, Algorithmus entwickeln
- Algorithmus in Java „programmieren“, erst mal im Kopf
- GUI entwickeln
  - Editorzeile enthält den Wert n
  - Editor zeigt alle Werte, mit Zwischenwert
  - Schalter „Ok“ startet die Berechnung
- Ereignisse „verdrahten“
- Algorithmus in BnOk_Click eintragen
- Editor löschen mit setText("");
- Ergebnisse mit Editor.append(...) eintragen

```
void BnOk_Click() {
   long i, n, p;
     // hole den Wert n
   n = LVal(Editzeile.getText() );
   Editor.setText("");  // Ausgabefenster, Editorinhalt löschen
   if ( (n != -999)&&(  (n>0) && (n<=20)  )     ) {
     p = 1;
     for (i=1; i<=n; i++) {
       p = p * i;
       Editor.append(i + " " + p + "\n");
     }
   }
   else {
     Editor.setText("Bitte eine gültige Zahl im Bereich 1 bis 20 eingeben");
   }
 }  // BnOk_Click
```

Beispiel: JComboBox

JComboBox ist eine Klasse, die in einer Liste Zeichenketten zur Auswahl anzeigt. Das Element klappt beim Anklicken auf. Man kann ein Element auswählen.



Beispiel: JComboBox

Methoden der Klasse JComboBox:

```
public JComboBox ();                        // leere Liste
public JComboBox (Object [] listData);      // Array mit Objekten
public JComboBox (Vector listData);         // Vector mit Objekten
```

setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
Nur ein Wert darf selektiert werden

setSelectionMode(ListSelectionModel.SINGLE_INTERVAL_SELECTION);
Ein Wert selektiert, oder ein Bereich

setSelectionMode(ListSelectionModel. MULTIPLE_INTERVAL_SELECTION);
Beliebiger Bereich darf selektiert werden

Methoden der Klasse JComboBox:

boolean isSelectionEmpty()
Wurde mindestens ein Wert selektiert ?

int getSelectedIndex() // Aktueller Index

Object getSelectedValue()
Liefert das erste aktuell ausgewählte Objekt oder null.

Object[] getSelectedValues()
Liefert alle ausgewählten Objekte als Object (cast)

void setSelectedIndex(int anIndex)
Setz den Index der Liste

void setSelectedItem(Object anObject)
Setzt den Eintrag über ein Objekt

Weitere Beispiele

- Summe von 1 bis N
- Quersumme
- Fibonacci, Leonardo von Pisa
- Berechnung der Primzahlen von 1 bis N
- Funktionsberechnung mit einer festen Funktion
  - $f(x) = 2*x^2 - 4*x + 44$
- Wertetabelle
- Wurzelalgorithmus (Prof. Zimmermann)

Java: Exception

```
public class exception1 {

  public static void main(String args[]) {
    int j,k;
    j=0;
    k=3 / j;
  }  // main
}
```

Java: Exception

```
D:\HS-Harz\java>java exception1
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at exception1.main(exception1.java:8)

D:\HS-Harz\java>
```

Java: Exception

Ein perfektes Programm mit perfekten Anwendern benötigt keine Fehlerbehandlung. Alle Daten werden korrekt eingegeben, jede Datei existiert und alle Module sind richtig programmiert.

Reale Welt:

Beim Absturz eines Programms entstehen:

- Datenverluste
- Unstimmigkeiten in einer Datenbank
- Neueingaben
- Ärger des Anwenders

Abhilfe:

- Benachrichtigung an den Anwender
- Sicherung der Daten
- Sicheres Beenden des Programms

Java: Exception

Definition:
public class Exception extends Throwable

The class Exception and its subclasses are a form of Throwable that indicates conditions that a reasonable application might want to catch.

Since:
JDK1.0

Java benutzt für die Fehlerbehandlung ein „error trapping“ bzw. „exception handling“ (Delphi, C++). Diese Fehlerbehandlung ist weit flexibler als die VB Basicvariante „On error goto“.

Java: Exception

Ursachen für eine Exception:

- Fehlerhafte Eingabe (Numerische Eingabe, URL)
- Gerätfehler (Drucker, Webseite, Diskette)
- Physikalische Grenzen (mangelnder Speicher, Freier Speicherplatz)
- Programmierfehler (Arrayindex, Stackfehler, Overflow, Underflow)

Exception:

Bei einer Exception wird

- die aktuelle Prozedur sofort verlassen
- es wird ein Exceptionobjekt erzeugt
- es wird kein Returncode erzeugt
- der Aufrufcode wird nicht weiterabgearbeitet
- es beginnt die Suche nach einem „exception handler“, der für diese Exception zuständig ist

Java: Exception-Arten

Standard Runtime Exceptions:

ArithmeticException:	An exceptional arithmetic situation has arisen, such as an integer division (§15.16.2) operation with a zero divisor.
ArrayStoreException:	An attempt has been made to store into an array component a value whose class is not assignment compatible with the component type of the array (§10.10, §15.25.1).
ClassCastException:	An attempt has been made to cast (§5.4, §15.15) a reference to an object to an inappropriate type.
IllegalArgumentException:	A method was passed an invalid or inappropriate argument or invoked on an inappropriate object. Subclasses of this class include:
IllegalThreadStateException:	A thread was not in an appropriate state for a requested operation.
NumberFormatException:	An attempt was made to convert a String to a value of a numeric type, but the String did not have an appropriate format.
IllegalMonitorStateException:	A thread has attempted to wait on (§20.1.6, §20.1.7, §20.1.8) or notify (§20.1.9, §20.1.10) other threads waiting on an object that it has not locked.
IndexOutOfBoundsException:	Either an index of some sort (such as to an array, a string, or a vector) or a subrange, specified either by two index values or by an index and a length, was out of range.

Java: Exception-Arten

Standard Runtime Exceptions:

NullPointerException:	An attempt was made to use a null reference in a case where an object reference was required.
SecurityException:	A security violation was detected (§20.17).

Java: Exception

Package java:

java.io.IOException:	A requested I/O operation could not be completed normally. Subclasses of this class include:
java.io.EOFException:	End of file has been encountered before normal completion of an input operation.
java.io.FileNotFoundException:	A file with the name specified by a file name string or path was not found within the file system.
java.io.InterruptedIOException:	The current thread was waiting for completion of an I/O operation, and another thread has interrupted the current thread, using the interrupt method of class Thread (§20.20.31).
java.io.UTFDataFormatException:	A requested conversion of a string to or from Java modified UTF-8 format could not be completed (§22.1.15, §22.2.14) because the string was too long or because the purported UTF-8 data was not the result of encoding a Unicode string into UTF-8.

Java: Exception (Anfangsbeispiel)

```
import java.io.*;

public class exception1 {

  public static void main(String argv[]) {
    int j,k;
    j=0;
    k=3 / j;
  }  // main
}
```

Java: Exception (interne Abfrage)

```
import java.io.*;

public class exception2 {

  public static void main(String argv[]) {
    int j,k;
    j=0;
    try {     // Excecption Block
      k=3 / j;
    }
    catch (ArithmeticException f) {
        System.err.println(" ArithmeticException : " + f);
    }
  }  // main
}
```

exception2

Ein- und Ausgabe in eine Datei

- Ausgabe von Text in eine ASCII-Datei
- Ausgabe von Zahlen in eine ASCII-Datei
- Ausgabe von Fließkommazahlen in eine ASCII-Datei
- Ausgabe von Zeichen, Zahlen in eine Binär-Datei

- Einlesen einer ASCII-Datei, zeilenweise
- Einlesen einer Binärdatei (Byte)
- Einlesen einer Binärdatei (Zahlen, Zeichen)

Java: Ausgabe in eine Datei

```
import java.io.*;
public class write1 {
  public static void main(String argv[]) {
                // Stream
      // Beispiel mit Buffer Funktion
    try {
        FileOutputStream Fout = new FileOutputStream("c:\\1.dat");
                // Druckausgabe
          PrintStream p = new PrintStream(Fout);
         p.println("Hallo");
            p.println("Hallo Studenten");
         p.close();
       }
       catch (IOException ee) {
          System.err.println("IOException: " + ee);
       }
  }  // main
}
```

Java: Ausgabe in eine Datei (int)

```
public static void main(String argv[]) {
   int i;
   try {
       FileOutputStream Fout = new FileOutputStream("1.int");
       DataOutputStream Dout = new DataOutputStream(Fout);
       i = 1234;
         Dout.writeInt(i);
       i = -1234;
         Dout.writeInt(i);
         Dout.writeInt(-1234);
       Dout.close();
      }
      catch (IOException ee) {
         System.err.println("IOException: " + ee);
      }
 }  // main
```

Java: Ausgabe in eine Datei (Double)

```
import java.io.*;
public class write5 {
       // Ausgabe ohne Buffer,  Ausgabe von Double Zahlen
  public static void main(String argv[]) {
     double d;
    try {
        FileOutputStream Fout = new FileOutputStream("c:\\1.dbl");
        DataOutputStream DataOut = new DataOutputStream(Fout);
        d = 1234.0;
          DataOut.writeDouble(d);
        d = 1234.11111;
          DataOut.writeDouble(d);
          DataOut.writeDouble(-1234.0);
          DataOut.close();
       }
       catch (IOException ee) {
          System.err.println("IOException: " + ee);
       }
  }  // main
}
```

Java: Ausgabe in eine Datei (Charakter)

```
public static void main(String argv[]) {
   char ch;
   try {
       FileOutputStream Fout = new FileOutputStream("1.ch");
       DataOutputStream Dout = new DataOutputStream(Fout);
       ch =  '\u0065';
       Dout.writeChar(ch);  // jedes Character hat zwei Zeichen
       ch =  '\u0066';
       Dout.writeChar(ch);
       Dout.close();
      }
      catch (IOException ee) {
          System.err.println("IOException: " + ee);
      }
    System.out.println("Es wurde die Datei 1.ch erzeugt");
}  // main
```

Java: Methoden der Klasse DataOutputStream

DataOutputStream:

writeBoolean
writeByte (Schreiben einer 8-Bit Vorzeichenzahl)
writeChar (Schreiben einer 16-Bit Vorzeichenlosen Zahl)
writeDouble (Schreiben einer Double-Zahl)
writeFloat (Schreiben einer Single-Zahl)
writeInt (Schreiben einer 32-Bit Vorzeichenzahl)
writeLong (Schreiben einer 64-Bit Vorzeichenzahl)
writeShort (Schreiben einer 16-Bit Vorzeichenzahl)

Int-Datentypen in Java

byte

from -128 to 127

short

from -32768 to 32767

int

from -2147483648 to 2147483647

long

from -9223372036854775808 to 9223372036854775807

char

from '\u0000' to '\uffff‘, =>, from 0 to 65535

Java: Lesen einer ASCII-Datei

```
public static void main(String argv[])  throws IOException {
   FileInputStream fin;
   DataInputStream din;
   char ch;
   byte b;
   try {
     fin = new FileInputStream("read1.java");
     din = new DataInputStream(fin);
     while  (true) {
       b = din.readByte();
       ch = (char) (b);
       System.out.print(ch);
     }
   }
   catch (IOException ee) {
         System.err.println("IOException: " + ee);
   }
}
```

Java: Lesen einer ASCII-Datei

```
import java.io.*;
import java.net.*;

public class read2 {
      // Exception wird hier in die Definition eingebaut !!!!!
      // WICHTIG: die Methode readLine ist veraltet,    siehe read3
  public static void main(String argv[])  throws IOException {
    FileInputStream fin;
    DataInputStream din;
    String s;

    fin = new FileInputStream("c:\\1.dat");
    din = new DataInputStream(fin);
    while (  (s=din.readLine()) != null) {
      System.out.println(s);
    }
  }  // main
}
```

Java: Lesen einer ASCII-Datei (neu)

```
public static void main(String argv[])  throws IOException {
   FileInputStream fin;
   InputStreamReader iin;
   LineNumberReader din;
            // aktuelle Version zum Einlesen einer ASCII-Datei
   String sLine;
   try {
     fin = new FileInputStream("read3.java");
     iin = new InputStreamReader(fin);
     din = new LineNumberReader(iin);
     while  ( din.ready()  ) {
       sLine = din.readLine();
       System.out.println(sLine);
     }
   }
   catch (IOException ee) {
         System.err.println("IOException: " + ee);
   }
}
```

Java: Lesen einer Binärdatei: Nur Bytes

```
import java.io.*;
public static void main(String argv[])  throws IOException {
   FileInputStream fin;
   DataInputStream din;
   char ch;
   byte b;
   try {
     fin = new FileInputStream("1.dat");
     din = new DataInputStream(fin);

     while  (din.available()>0    ) {
       b = din.readByte();
       ch = (char) (b);
       System.out.println(ch);
     }
   }
   catch (IOException ee) {
         System.err.println("IOException: " + ee);
   }
```

Java: Lesen einer Binärdatei

```
import java.io.*;
// Einlesen aus einer binären Datei, in der int-Werte gespeichert wurden
public class read5 {
  private static void read(String sFilename){
      int i;
      try {
          FileInputStream Fin = new FileInputStream(sFilename);
          DataInputStream DataIn = new DataInputStream(Fin);
          i = DataIn.readInt();
          System.out.println(i);
          i = DataIn.readInt();
          System.out.println(i);
          i = DataIn.readInt();
          System.out.println(i);
          DataIn.close();
      }
      catch (IOException ee) { }
  }
```

```
      // Einlesen von int Zahlen
      // Einlesen der Zahlen
  public static void main(String argv[]) {
    read("1.int");
    pause();
  }  // main
}  // class
```

Java: Schreiben und Lesen einer Binärdatei

```
import java.io.*;
    // Ausgabe ohne Buffer, Ausgabe von int Zahlen, Einlesen der Zahlen

 public class readwrite1 {

 public static void main(String argv[]) {
    int i;
    try {
      FileOutputStream Fout = new FileOutputStream("rw1.int");
      DataOutputStream DataOut = new DataOutputStream(Fout);
      i = 1234;
        DataOut.writeInt(i);
      i = 4321;
        DataOut.writeInt(i);
        DataOut.writeInt(-1234);
       DataOut.close();
```

Java: Schreiben und Lesen einer Binärdatei

```
      // weiter im Sourcecode
     FileInputStream Fin = new FileInputStream("rw1.int");
            // Einlesen der Dateien
     DataInputStream DataIn = new DataInputStream(Fin);
     i = DataIn.readInt();
      System.out.println(i);
     i = DataIn.readInt();
      System.out.println(i);
     i = DataIn.readInt();
      System.out.println(i);
      DataIn.close();
    }
    catch (IOException ee) {
       System.err.println("IOException: " + ee);
    }
  System.out.println("Es wurde in die Datei rw1.int geschrieben und gelesen");
  System.out.println("Bitte Taste druecken");
   pause();
 }  // main
}
```

Java: Methoden der Klasse DataInputStream

DataInputStream:

readBoolean
readByte (Einlesen 8-Bit Vorzeichenzahl)
readChar (Einlesen 16-Bit Vorzeichenzahl)
readDouble (Einlesen einer Double-Zahl)
readFloat (Einlesen Single-Zahl)
readInt (Einlesen 32-Bit Vorzeichenzahl)
readLong (Einlesen 64-Bit Vorzeichenzahl)
readShort (Einlesen 16-Bit Vorzeichenzahl)

Einfügen der Methode WriteString (Erweitern)

```
class MyDataOutputStream extends DataOutputStream {
    public MyDataOutputStream(OutputStream out) {
        super(out);
    }

    public void writeString(String s) throws IOException{
        int n = s.length();
        System.out.println("String:" + s);
        System.out.println("StringNumber of characters:" + n);
        writeInt(n);
        for(int i= 0; i < n; i++)
          writeChar(s.charAt(i));
    }

}  // Ende MyDataOutputStream
```

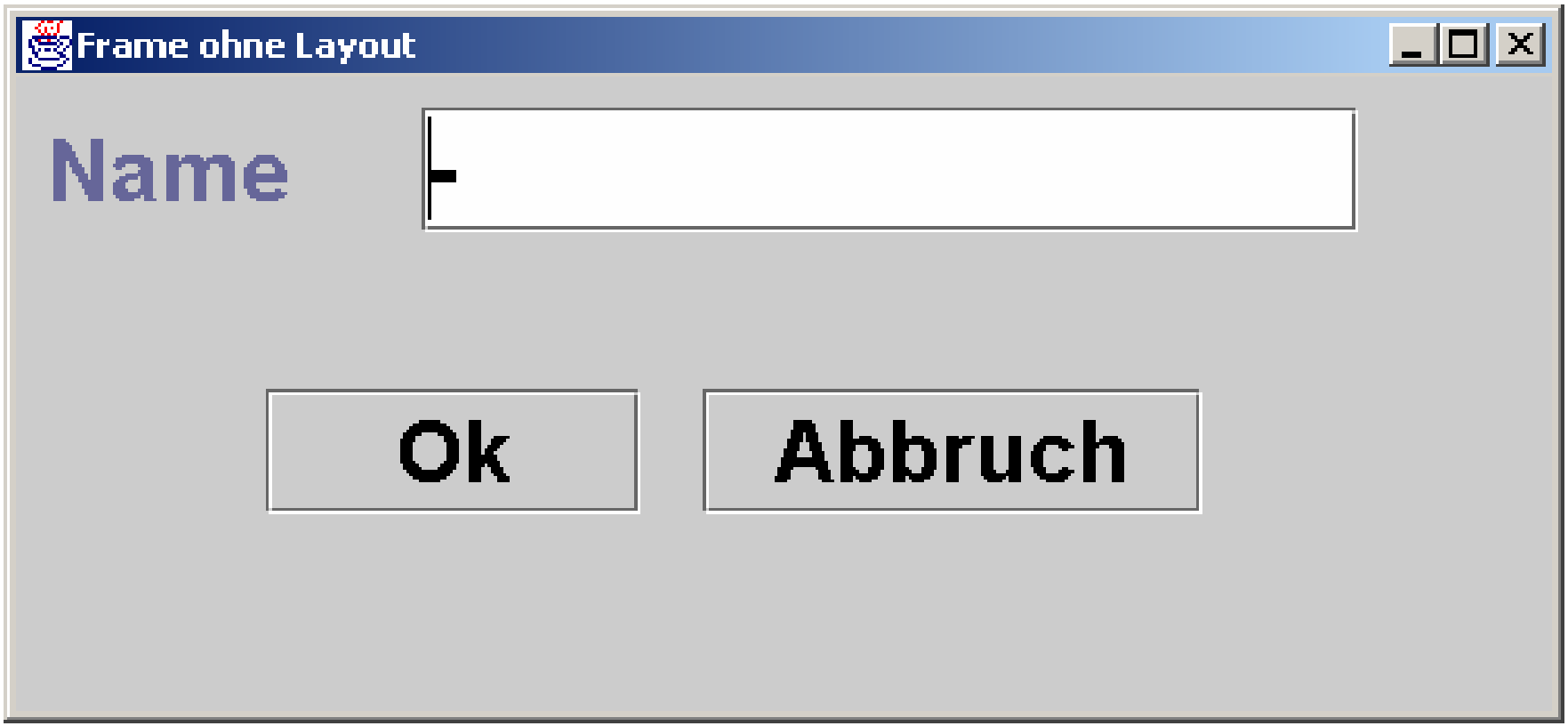
Einfügen der Methode ReadString (Erweitern)

```
class MyDataInputStream extends DataInputStream {
    public MyDataInputStream(InputStream in){
        super(in);
    }

    public String readString() throws IOException {
        int n = readInt();
        String s = new String("");
        for(int i = 0; i < n; i++)
            s += readChar();
      return s;
    }

}  // Ende MyDataInputStream
```

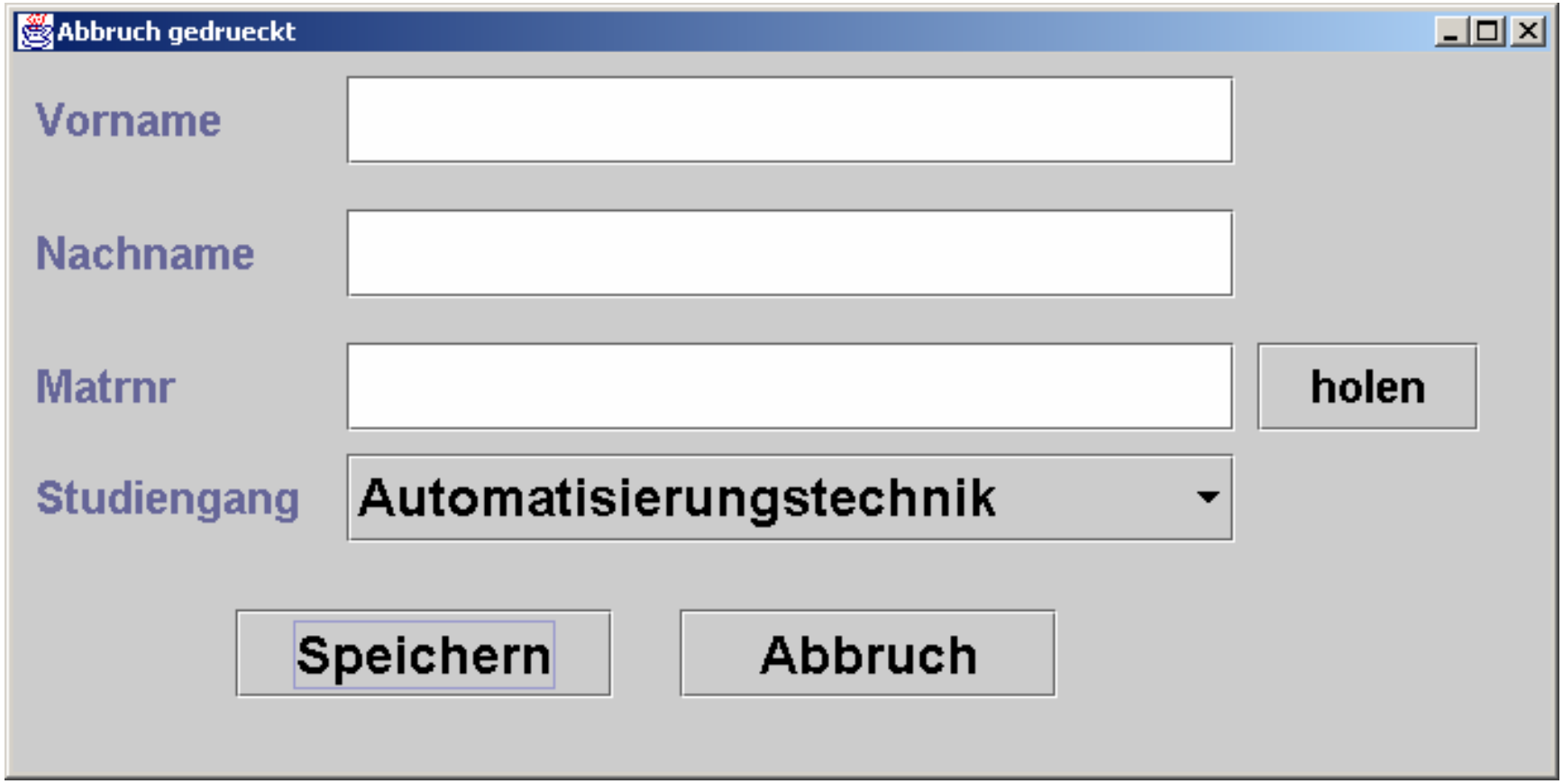
Layout „Null-Layout“

```
private void setGUI() {
    setSize(500, 230);  // width Height
    setTitle("FRame ohne Layout");
    this.getContentPane().setLayout( null );

    Label1.setText("Name");
    this.getContentPane().add(Label1);
    Label1.setBounds(10, 10, 80, 40);           // x/y   width height
    Label1.setFont( new Font("Arial", Font.BOLD,28));

    TName.setText("-");
    this.getContentPane().add(TName);
    TName.setBounds(130, 10, 300, 40);       // x/y   width height
    TName.setFont( new Font("Arial", Font.BOLD,28));
 }  // setGUI
```

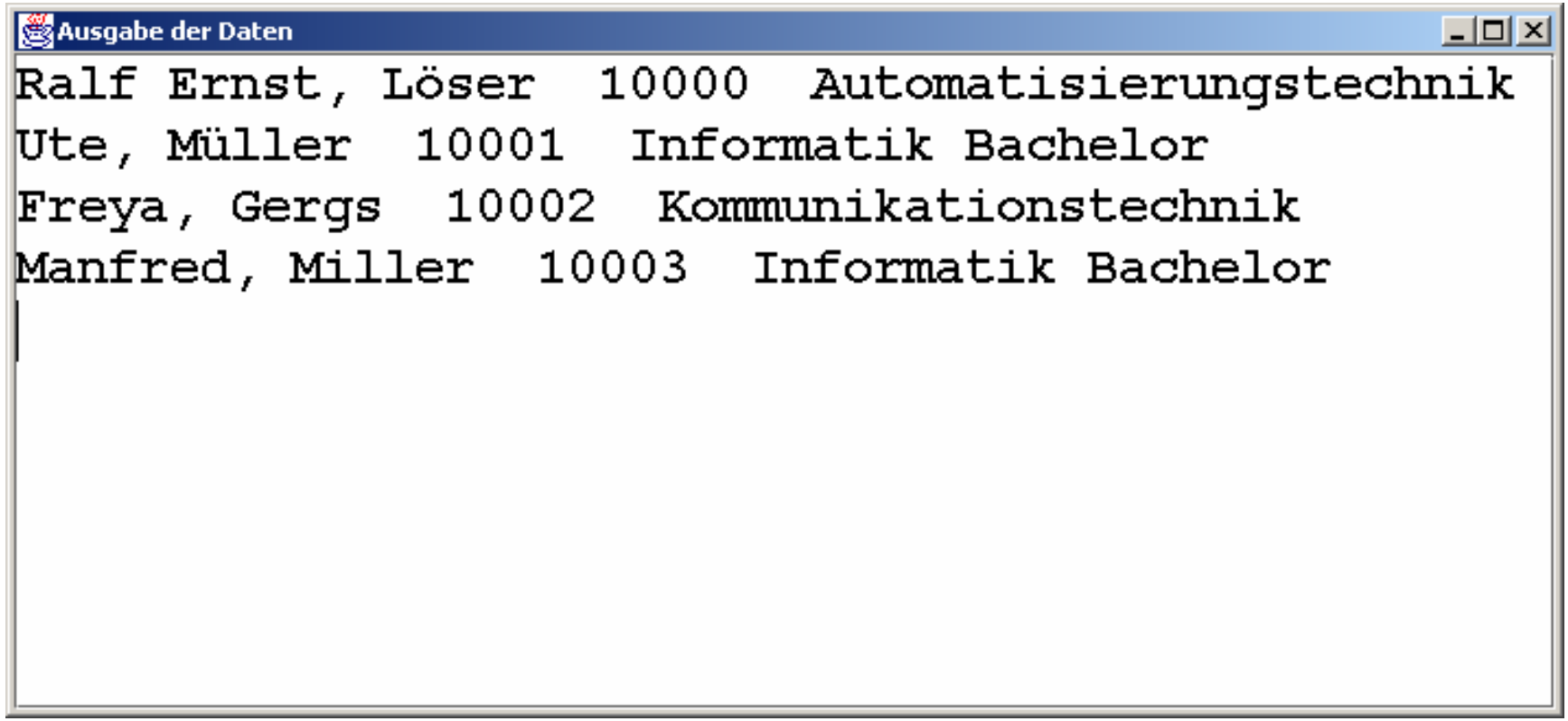
Layout „Null-Layout“



2. Aufgabe: Studenten-Datenbank



2. Aufgabe: Studenten-Datenbank: Listenfenster



2. Aufgabe: Studenten-Datenbank

Klasse bsp02 von JFrame
Zwei Schalter rufen die Fenster auf

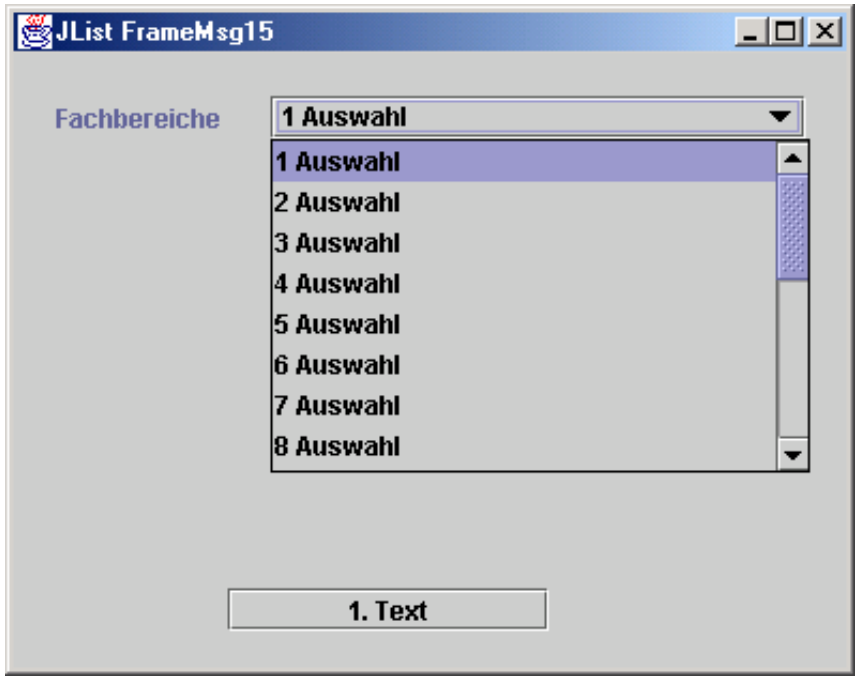
Klasse Window_Input von JFrame
Eingabe von:
Vorname
Nachname
Matrikelnummer
Studiengang

Speicherung in eine Datei

Klasse Window_Liste von JFrame
Einlesen der Datei:
Ausgabe in einem Editor

Beispiel: JComboBox

JComboBox ist eine Klasse, die in einer Liste Zeichenketten zur Auswahl anzeigt. Das Element klappt beim Anklicken auf. Man kann ein Element auswählen.



Object cCombo.getSelectedItem() // Returns the currently selected item.

JComboBox ohne Layout, setBounds

```
public ComboBox1() {
  panel = new JPanel();
  panel.setLayout ( null );                  // Layout ausgeschaltet !!!!
  bn1 = new JButton("1. Text");  // Label erzeugen
  panel.add(bn1);
  bn1.setBounds(100, 250, 150, 20); // x/y   width height

  items = new String[20];
  for (int i=0; i<20; i++) {
    items[i] = Integer.toString(i+1)+" Auswahl";
  }

  liste = new JComboBox(items);
  liste.setBounds(20, 20, 350, 200); // x/y   width height
  panel.add(liste);

  this.getContentPane().add(panel, BorderLayout.CENTER);
} // create
```

Aufgaben

- Hauptfenster:
  - Erstellen des Hauptfensters mit zwei Schaltern
  - Klick-Events „verdrahten“
  - Fensterklassen definieren
  - Beide Unterfenster anzeigen
- Eingabenfenster
  - No-Layout einstellen
  - GUI-Elemente erstellen (3×Editzeilen, 1×ComboBox, 2×Schalter
  - Klick-Events „verdrahten“ (MatrNr, Save)
  - Speichern der aktuellen Daten (Student, MatrNr), löschen der Eingabe
- Listenfenster
  - GUI-Elemente erstellen (1×Editor)
  - Einlesen der Studenten-Datei
  - Anzeige im Editor (setText)

Weitere Hinweise:

Klasse Database:

•Konstruktor (sFilename)
•Methode addStudent

```
public void addStudent(  String sVorname, String sNachname,
      long Matrnr, String sStudiengang) {

  int i, Anz;
  FileOutputStream Fout;
  MyDataOutputStream Dout;
  File file;
  try {
     file = new File(_sFilename);
     if (file.exists() )
        Fout = new FileOutputStream(_sFilename, true); // Append
     else
        Fout = new FileOutputStream(_sFilename);
  }
  catch (IOException ee) {
      System.err.println("IOException: " + ee);
  }
}
```

Weitere Hinweise:

- Setzen des Focus
  - EVorname.requestFocus();

- Listenfenster
  - Einlesen der Studenten-Datei mittels der Methode readListe() oder mit einer Methode der Klasse Database

Weitere Hinweise:

- Existiert eine Datei ?

```
File file;
try {
      file = new File(sFilename);
      if (file.exists() ) {
        try {
   // Öffnen der Datei und lesen der Blöcke
        }
        catch (IOException e2) {
       }
                        finally {
         Fin.close();
       }
    }  // try
```