10-2

Swing-Komponenten

10.2.1 Typische Komponenten für Benutzerschnittstellen ☺

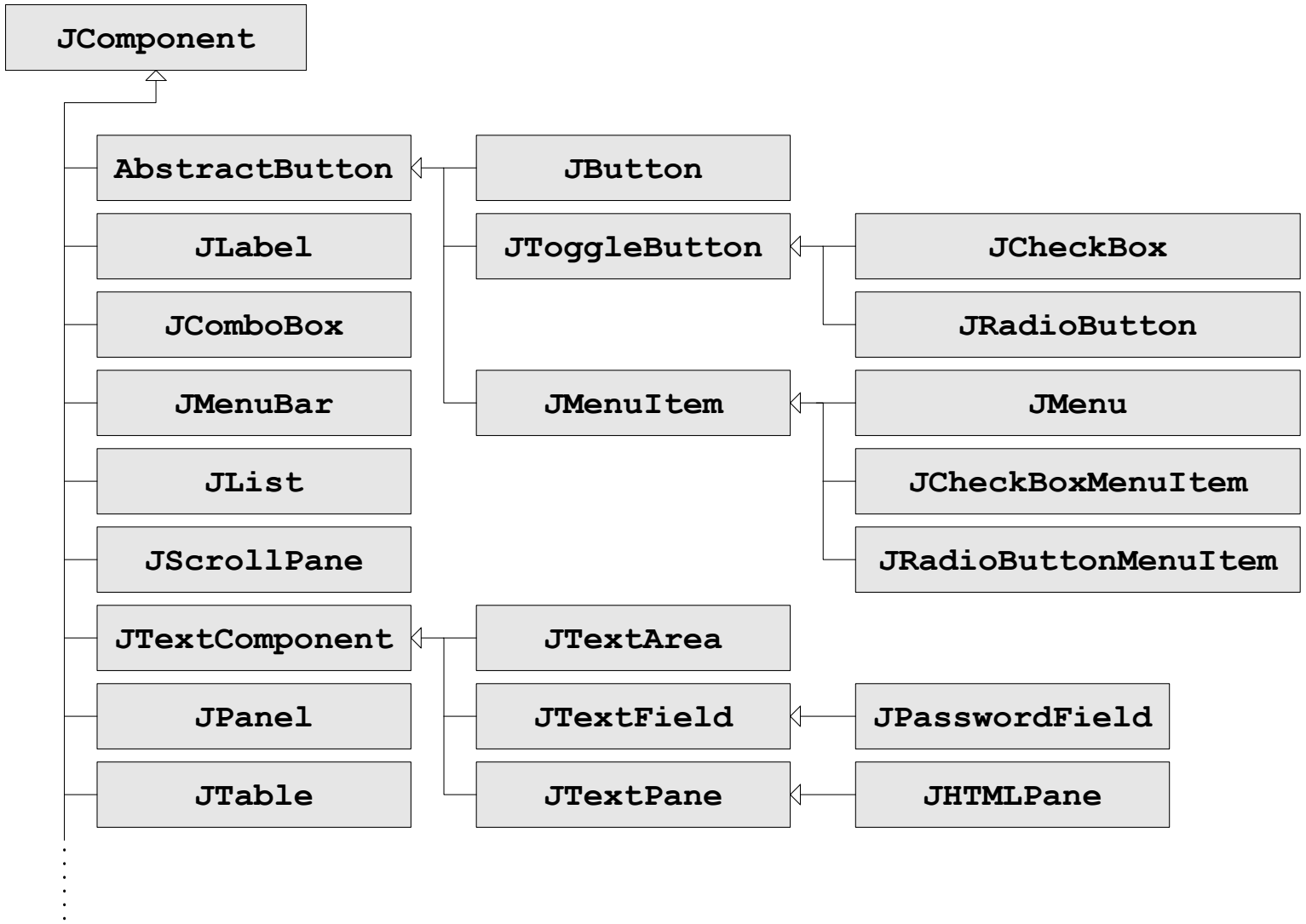
- Das **Abstract Window Toolkit** (AWT) liefert bereits viele Möglichkeiten für Komponenten für graphische Oberflächen
- **Komponenten** im Paket `java.awt` (z. B.):
  - Schalter, Druckknöpfe — Klasse `Button`
  - Häkchen-Kästchen — Klasse `Checkbox`
  - Einzeilige Textfelder — Klasse `TextField`
  - Mehrzeilige Textfelder — Klasse `TextArea`
  - Beschriftungen — Klasse `Label`
  - Listen — Klasse `List`
  - Pop-up Auswahllisten (Klapptafeln) — Klasse `Choice`
  - Schieber und Rollbalken — Klasse `Scrollbar`
  - Menüs — Klassen `Menu, MenuItem, ...`
  - Container — Klassen `Panel, Window, ...`

Typische Komponenten für Benutzerschnittstellen

- **Swing**
  - Erweiterung des AWT um neuen Satz von GUI-Komponenten mit Betriebssystem-unabhängigem aber auch anpassbaren "Look and feel"
  - vollständig in 100% purem Java implementiert, zusätzliche Features
- **Komponenten** im Paket `javax.swing` (z. B.)
  - Schalter, Druckknöpfe — Klasse `JButton`
  - Häkchen-Kästchen — Klasse `JCheckbox`
  - Einzeilige Textfelder — Klasse `JTextField`
  - Mehrzeilige Textfelder — Klasse `JTextArea`
  - Beschriftungen — Klasse `JLabel`
  - Listen — Klasse `JList`
  - Pop-up Auswahllisten (Klapptafeln) — Klasse `JComboBox`
  - Schieber und Rollbalken — Klasse `JScrollPane`
  - Menüs — Klassen `JMenu, JMenuItem, ...`
  - Container — Klassen `JPanel, JTable`
  - `...`

Typische Komponenten für Benutzerschnittstellen

- Auszug aus der Swing-Klassen-Hierarchie

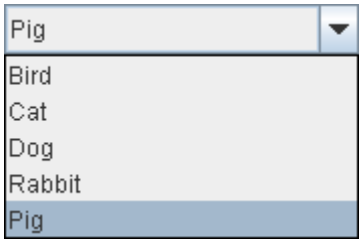


Swing-Komponenten: basic controls

JButton

JCheckBox

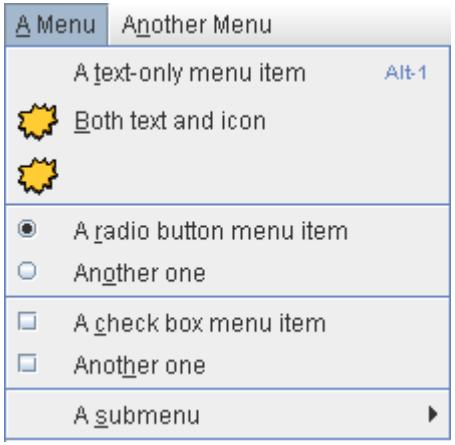
JComboBox



JList

JRadioButton

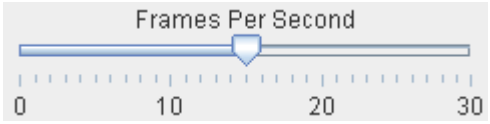
JMenu



JTextField

JPasswortField

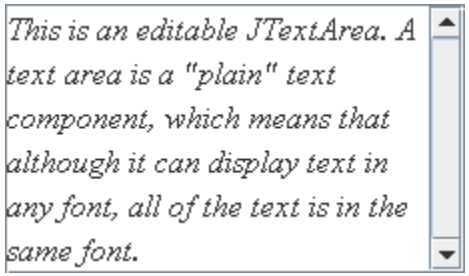
JSlider



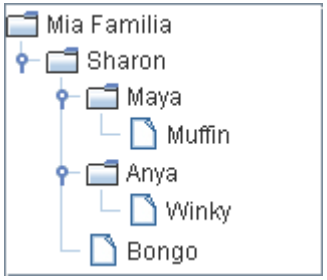
JSpinner

Swing-Komponenten: Complex Controls

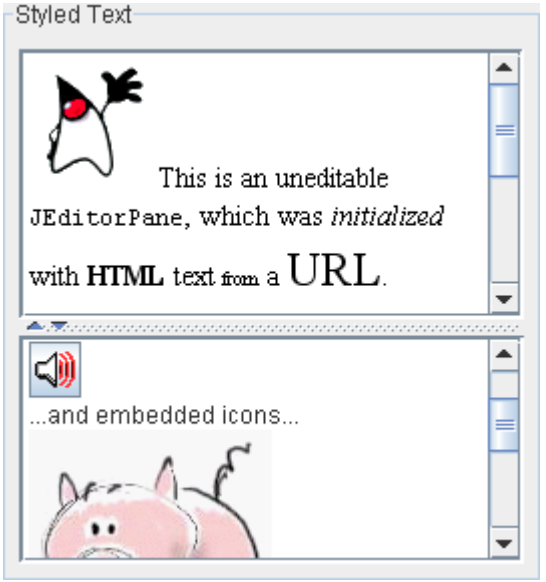
JTextArea



JTree



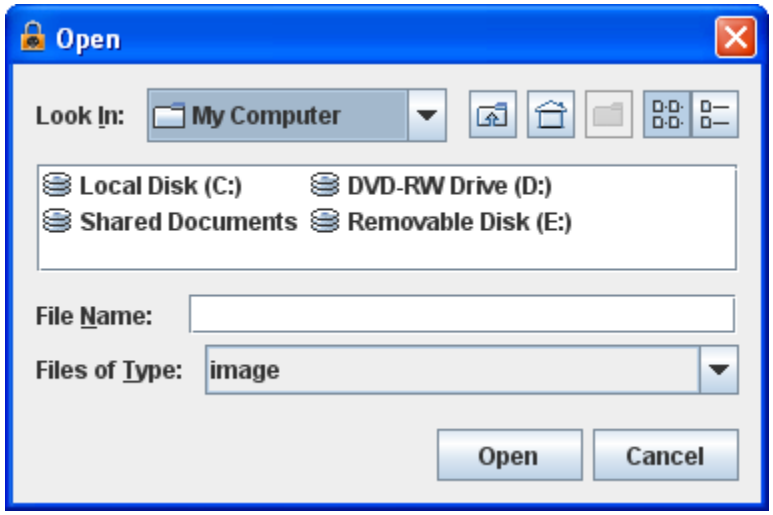
JEditorPane / JTextPane



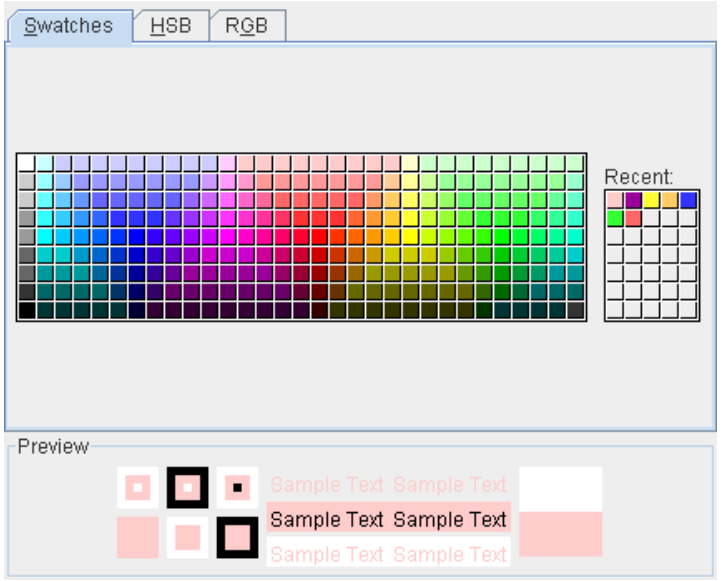
JTable

Host	User	Password	Last Modified
Biocca Games	Freddy	!#asf6Awwzb	Mar 16, 2006
zabble	ichabod	Tazb!34$fZ	Mar 6, 2006
Sun Developer	fraz@hotmail.co...	AasW541!fbZ	Feb 22, 2006
Heirloom Seeds	shams@gmail....	bkz[ADF78!	Jul 29, 2005
Pacific Zoo Shop	seal@hotmail.c...	vbAf124%z	Feb 22, 2006

JFileChooser

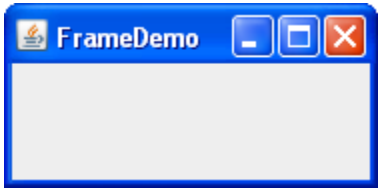


JColorChooser



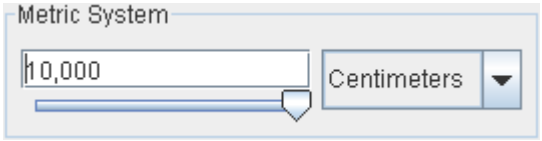
Swing-Komponenten: Top Level Container

JFrame



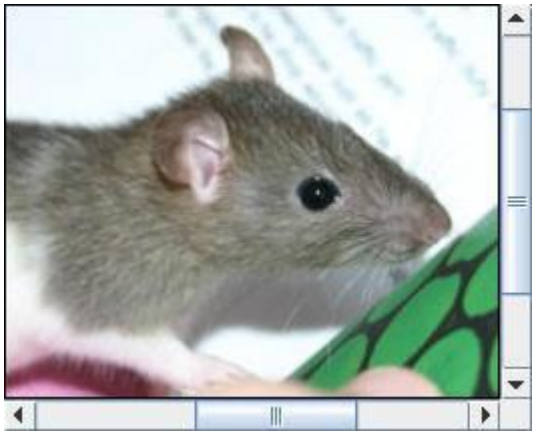
JApplet

JPanel

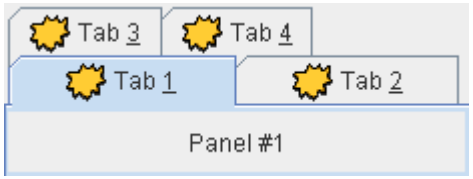


Box

JScrollPane



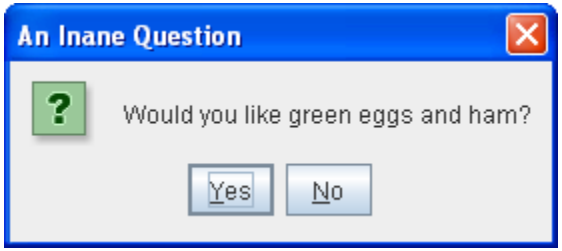
JTabbedPane



JSplitPane

JToolBar

JDialog



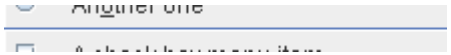
Swing Komponenten: Uneditable

JLabel

JToolTip

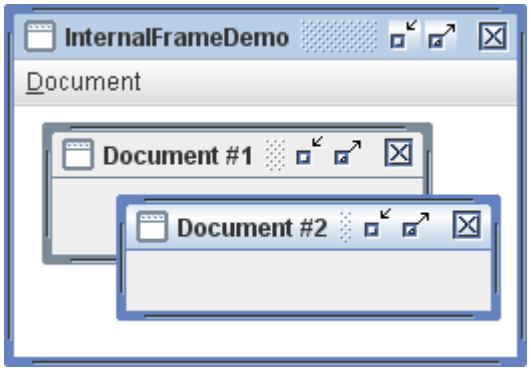
JProgressBar

JSeparator

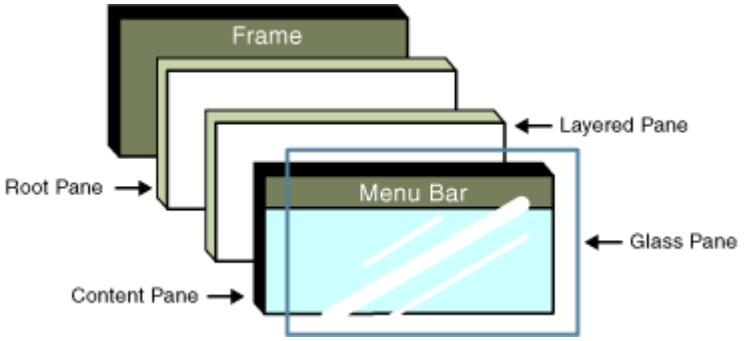


Swing Komponenten: Special Purpose Container

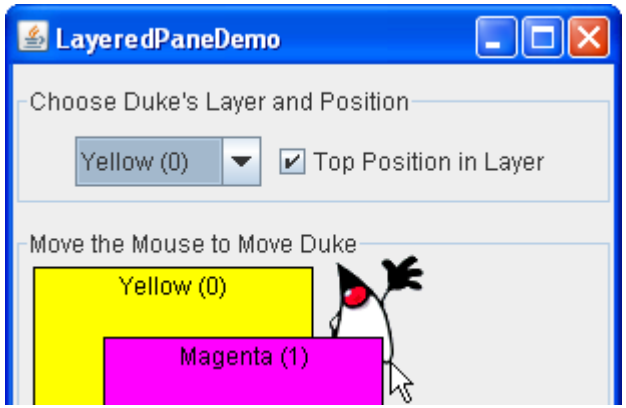
JInternalFrame



JRootPane



JLayeredPane



Typische Komponenten für Benutzerschnittstellen

- Übliche Standard-Ausstattung der Komponenten-Klassen:
  - Mehrfach überladene Konstruktoren ohne und mit diversen Parametern für Text- und Bildbeschriftungen und ihre horizontale und vertikale Ausrichtung
  - Klassenkonstanten wie **`LEFT, RIGHT, CENTER, TOP, BOTTOM, usw.`** vom Typ **`int`** zur Spezifikation der horizontalen und vertikalen Ausrichtung
  - Instanz-Methoden zum Auslesen und Setzen der dargestellten Texte und Bilder (**`getText()`**, **`setText(…)`**, **`getIcon()`**, **`setIcon(…)`**)
  - Instanz-Methoden für die Festlegung der Position und der horizontalen und vertikalen Ausrichtung der dargestellten Texte und Bilder
  - Je nach Funktionalität der Komponenten weitere Instanz-Methoden zum Aktivieren und Deaktivieren von Einstellungen bzw. Zuständen (z. B. "selektiert", "editierbar", …) oder die Beeinflussung des Tastaturfokus
- Erzeugung eines Bildobjekts (ein Objekt einer Klasse, die das Interface **`Icon`** implementiert) meistens durch ein Objekt der Klasse **`ImageIcon`** mit dem Konstruktor
  - **`public ImageIcon(String filename)`**
    erzeugt ein **`ImageIcon`**-Objekt aus dem Bild in der Datei **`filename`**.

10.2.2 Die Klasse JLabel

- Beispiel

```
import java.awt.*;
import javax.swing.*;
public class FrameMitBild extends JFrame {
  Container c;
  JLabel lab;
  public FrameMitBild() {
    c = getContentPane();
    c.setLayout(new FlowLayout());
    Icon bild = new ImageIcon("babycat.jpg");
    lab = new JLabel("Spotty", bild, JLabel.CENTER);
    lab.setHorizontalTextPosition(JLabel.CENTER);
    lab.setVerticalTextPosition(JLabel.BOTTOM);
    c.add(lab);
  }
}
```

10.2.3 Die abstrakte Klasse `AbstractButton`

- Alle Arten von Buttons (Knöpfen) können mit Text, mit einem Bild oder mit Text *und* Bild beschriftet werden.
- Die Basisfunktionalitäten für alle Arten von Buttons sind in der abstrakten Klasse **`AbstractButton`** bereitgestellt, von der alle anderen Button-Klassen erben.
- Methoden
  - **`public boolean isSelected()`**
    liefert **`true`**, wenn der Button selektiert ist, andernfalls **`false`**.
  - **`public void setSelected(boolean b)`**
    setzt den Zustand des Buttons auf ”selektiert”, falls `b` den Wert **`true`** hat

10.2.4 Die Klasse `JButton`

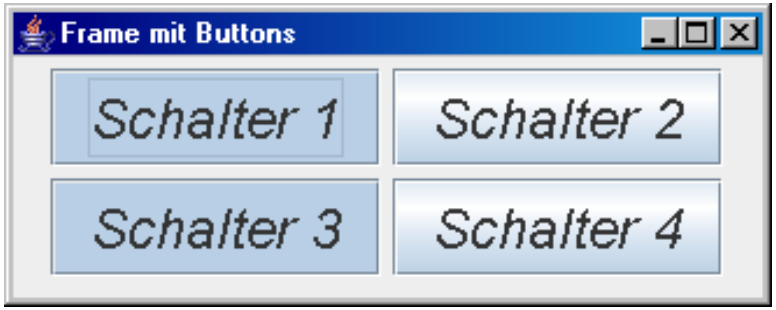
- Mit Text und/oder Bild beschriftete Schaltflächen (Knöpfe, Tasten).
- Beispiel

```
import java.awt.*;
import javax.swing.*;
public class FrameMitButtons extends JFrame {
  Container c;
  JButton[] b = new JButton[4];
  public FrameMitButtons() {
    c = getContentPane();
    c.setLayout(new FlowLayout());
    for (int i = 0; i < 4; i++) {
      b[i] = new JButton("Taste " + (i+1));
      b[i].setFont(new Font("SansSerif",Font.ITALIC,24));
      c.add(b[i]);
    }
  }
}
```

10.2.5 Die Klasse `JToggleButton`

- Mit Text und/oder Bild beschriftete, ”echte” Schalter, die sich ihren Zustand (an/aus bzw. selektiert/nicht selektiert) merken können.
- Beispiel

```
import java.awt.*;
import javax.swing.*;
public class FrameMitToggleButtons extends JFrame {
  Container c;
  JToggleButton[] b = new JToggleButton[4];
  public FrameMitToggleButtons() {
    c = getContentPane();
    c.setLayout(new FlowLayout());
    for (int i = 0; i < 4; i++) {
      b[i] = new JToggleButton("Schalter " + (i+1));
      b[i].setFont(new Font("SansSerif",Font.ITALIC,24));
      c.add(b[i]);
    }
    b[0].setSelected(true);
    b[2].setSelected(true);
  }
}
```



10.2.6 Die Klasse JCheckBox

- Kästchen, die man mit der Maus oder mit der Leertaste ”ankreuzen” und in den Zustand ”selektiert” bringen kann.
- Der ”selektiert”-Zustand wird durch ein kleines Häkchen gekennzeichnet.
- Beispiel

```
import java.awt.*;
import javax.swing.*;
public class FrameMitCheckBoxes extends JFrame {
  Container c;
  JCheckBox[] cb = new JCheckBox[4];
  public FrameMitCheckBoxes() {
    c = getContentPane();
    c.setLayout(new FlowLayout());
    for (int i = 0; i < 4; i++) {
      cb[i] = new JCheckBox("Box " + (i+1));
      c.add(cb[i]);
    }
    cb[0].setSelected(true);
    cb[2].setSelected(true);
  }
}
```

10.2.7 Die Klassen `JRadioButton` und `ButtonGroup`

- Objekte der Klasse **`JRadioButton`** werden als Kreise dargestellt, die man mit der Maus oder mit der Leertaste ”ankreuzen” und in den Zustand ”selektiert” bringen kann.
- Der ”selektiert”-Zustand wird durch einen Punkt (einen ausgefüllten kleinen Kreis) gekennzeichnet.
- In der Regel werden **`JRadioButton`**-Objekte ***gruppiert***, so dass stets *höchstens eine* Markierung pro Gruppe aktiviert sein kann.
- Dazu wird ein Objekt der Klasse **`ButtonGroup`** eingesetzt.
- Konstruktor
  - **`public ButtonGroup()`**
- Methoden
  - **`public void add(AbstractButton b)`**
    fügt **`b`** der Gruppierung hinzu.
  - **`public void remove(AbstractButton b)`**
    entfernt **`b`** aus der Gruppierung.
- Gruppierung sind also mit allen Arten von **`AbstractButton`**-Objekten möglich, aber eigentlich nur mit **`JToggleButton`**-Objekten sinnvoll.

Die Klassen JRadioButton und ButtonGroup

- Beispiel

```
import java.awt.*;
import javax.swing.*;
public class FrameMitRadioButtons extends JFrame {
  Container c;
  JRadioButton[] rb = new JRadioButton[4];
  public FrameMitRadioButtons() {
    c = getContentPane();
    c.setLayout(new FlowLayout());
    ButtonGroup bg = new ButtonGroup();
    for (int i = 0; i < 4; i++) {
      rb[i] = new JRadioButton("Box " + (i+1));
      bg.add(rb[i]); // der Gruppe hinzufuegen
      c.add(rb[i]);  // dem Frame hinzufuegen
    }
  }
}
```

10.2.8 Die Klasse `JComboBox<E>`

- Auswahlliste, die man mit Hilfe der Maus oder der Tastatur aufklappen und in der man einen Eintrag auswählen kann.
- Angezeigt wird dabei jeweils der ausgewählte Eintrag und ein Pfeil nach unten, der andeutet, dass es sich um eine aufklappbare Liste handelt.
- Konstruktoren
  - **`public JComboBox()`**
    erzeugt ein **`JComboBox`**-Objekt ohne Einträge.
  - **`public JComboBox(Object[] items)`**
    erzeugt ein **`JComboBox`**-Objekt, dessen Einträge durch die Komponenten des Feldes **`items`** festgelegt sind.
- Methoden zum Auf- und Abbau der Auswahlliste
  - **`public void addItem(Object item)`**
    fügt dem **`JComboBox`**-Objekt den Eintrag **`item`** (am Ende) hinzu.
  - **`public void removeItem(Object item)`**
    entfernt den Eintrag **`item`** aus dem **`JComboBox`**-Objekt.

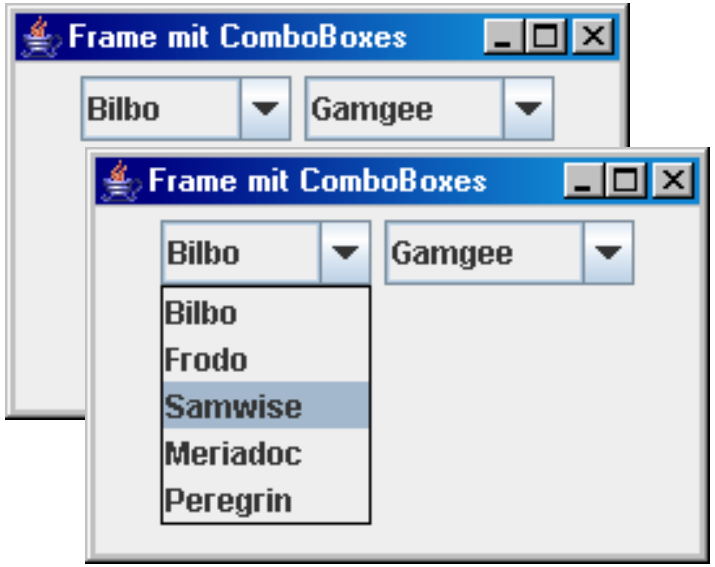
Die Klasse `JComboBox<E>`

- Methoden zum Zugriff auf die Einträge
  - **`public Object getItemAt(int index)`**
    liefert den Listen-Eintrag an der Position **`index`**.
  - **`public int getItemCount()`**
    liefert die Anzahl der Einträge in der Liste.
  - **`public int getSelectedIndex()`**
    liefert den Index (die Position) des gerade ausgewählten Eintrags.
  - **`public Object getSelectedItem()`**
    liefert den gerade ausgewählten Eintrag.
  - **`public void setSelectedIndex(int index)`**
    legt den Eintrag unter dem Index (der Position) **`index`** als gerade ausgewählten Eintrag des **`JComboBox`**-Objekts fest.
  - **`public void setSelectedItem(Object item)`**
    legt den Eintrag **`item`** als gerade ausgewählten Eintrag des **`JComboBox`**-Objekts fest.
- **`JList`** als Variante der **`JComboBox`** erlaubt Selektion von *mehreren* Einträgen

Die Klasse JComboBox

- Beispiel

```
import java.awt.*;
import javax.swing.*;
public class FrameMitComboBoxes extends JFrame {
  Container c;
  JComboBox<String> vornamen, nachnamen;
  public FrameMitComboBoxes() {
    c = getContentPane();
    c.setLayout(new FlowLayout());
    String[] namen = new String[]
      { "Bilbo", "Frodo", "Samwise", "Meriadoc", "Peregrin" };
    vornamen = new JComboBox<String> (namen);
    nachnamen = new JComboBox<String>
    nachnamen.addItem("Baggins");
    nachnamen.addItem("Brandybuck");
    nachnamen.addItem("Gamgee");
    nachnamen.addItem("Took");
    nachnamen.setSelectedIndex(2);
    c.add(vornamen);
    c.add(nachnamen);
  }
}
```



10.2.9 Die abstrakte Klasse `JTextComponent`

- Swing stellt verschiedene Klassen zur Eingabe und Anzeige von Texten bereit
  - **`JTextField`**, **`JPasswordField`** einzeilige Texteingaben
  - **`JTextArea`** mehrzeilige Texteingaben
  - **`JEditorPane`**, **`JTextPane`** formatierte Texte (z. B. HTML)
- Die Basis-Funktionalitäten aller Text-Komponenten werden durch die Methoden der abstrakten Klasse **`JTextComponent`** (aus dem Paket **`javax.swing.text`**) bereitgestellt, von der alle anderen Text-Komponenten erben.
- Methoden zur Kommunikation mit dem Betriebssystem
  - **`public void copy()`**
    kopiert den gerade markierten Textteil in die Zwischenablage.
  - **`public void cut()`**
    löscht den gerade markierten Textteil und kopiert ihn in die Zwischenablage.
  - **`public void paste()`**
    fügt den Text aus der Zwischenablage in die Text-Komponente ein.

Die abstrakte Klasse `JTextComponent`

- Methoden für den Zugriff auf den Inhalt einer Text-Komponente
  - **`public String getSelectedText()`**
    liefert den *gerade markierten* Textteil.
  - **`public String getText()`**
    liefert den *kompletten* Text der Text-Komponente.
  - **`public boolean isEditable()`**
    liefert **`true`**, falls die Text-Komponente editierbar ist, oder andernfalls **`false`**.
  - **`public void setEditable(boolean b)`**
    setzt die Text-Komponente in den Modus ”editierbar” bzw. ”nicht editierbar”.
  - **`public void setText(String t)`**
    setzt den Text der Text-Komponente auf den Inhalt von **`t`**.

10.2.10 Die Klassen `JTextField` und `JPasswordField`

- Felder zur Eingabe und die Bearbeitung einer einzelnen Textzeile.
  - **`JTextField`**-Objekt stellt die Textzeile lesbar dar.
  - **`JPasswordField`**-Objekt stellt die Textzeile unlesbar dar (mittels einer entsprechenden Anzahl von Ersatz-Zeichen, den ”Echo-Zeichen”).
- Die Klasse **`JPasswordField`** überschreibt die Methoden **`copy()`** und **`cut()`** so, dass sie lediglich ein Fehler-Signal erzeugen, da die Operation unzulässig ist, um sicherzustellen, dass ein Text, der in ein **`JPasswordField`**-Objekt eingegeben wurde, nicht in die Zwischenablage kopiert werden kann!

Die Klassen `JTextField` und `JPasswordField`

- Beispiel

```
import java.awt.*;
import javax.swing.*;
public class FrameMitTextFeldern extends JFrame {
  Container c;
  public FrameMitTextFeldern() {
    c = getContentPane();
    c.setLayout(new GridLayout(2,2));
    JLabel name = new JLabel("Name:",JLabel.RIGHT);
    JLabel passwd = new JLabel("Passwort:",JLabel.RIGHT);
    JTextField tf = new JTextField();
    JPasswordField pf = new JPasswordField();
    c.add(name);
    c.add(tf);
    c.add(passwd);
    c.add(pf);
  }
}
```

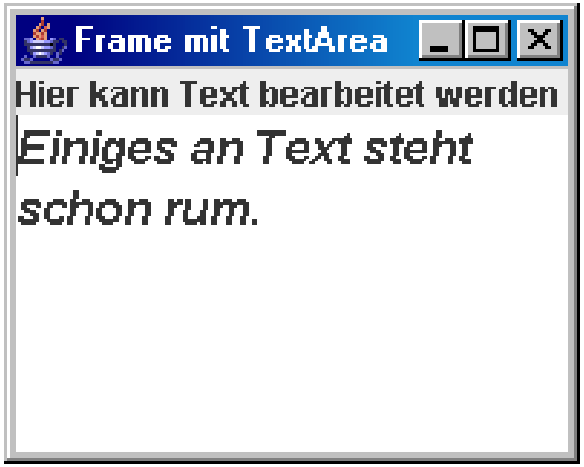
10.2.11 Die Klasse `JTextArea`

- Text-Komponente für die Eingabe und die Bearbeitung mehrzeiliger Texte.
- Konstruktoren
  - **`public JTextArea()`**
    erzeugt ein leeres **`JTextArea`**-Objekt.
  - **`public JTextArea(String text)`**
    erzeugt ein **`JTextArea`**-Objekt, das den Text **`text`** enthält.
- Methoden
  - **`public int getLineCount()`**
    liefert die Anzahl der Zeilen.
  - **`public boolean getLineWrap()`**
    liefert **`true`**, wenn der automatische Zeilenumbruch aktiviert ist, andernfalls **`false`**.
  - **`public void setLineWrap(boolean wrap)`**
    aktiviert bzw. deaktiviert den automatischen Zeilenumbruch.
  - **`public boolean getWrapStyleWord()`**
    liefert **`true`**, wenn wortweiser Zeilenumbruch aktiviert ist, andernfalls **`false`**.
  - **`public void setWrapStyleWord(boolean word)`**
    aktiviert bzw. deaktiviert den wortweisen Zeilenumbruch.

Die Klasse JTextArea

- Beispiel

```
import java.awt.*;
import javax.swing.*;
public class FrameMitTextArea extends JFrame {
  Container c;
  JLabel info;
  JTextArea ta;
  Font f = new Font("SansSerif",Font.BOLD+Font.ITALIC,16);
  public FrameMitTextArea() {
    c = getContentPane();
    info = new JLabel("Hier kann Text bearbeitet werden");
    ta = new JTextArea("Einiges an Text steht schon rum.");
    ta.setFont(f);
    ta.setLineWrap(true);
    ta.setWrapStyleWord(true);
    c.add(info,BorderLayout.NORTH);
    c.add(ta);
  }
}
```



10.2.12 Die Klasse `JScrollPane`

- Einbettung anderer Komponenten in einen Darstellungsbereich, der mit horizontalen und vertikalen Bildlaufleisten ausgestattet ist.
- Ausschnittsweise Sichten auf eine Komponente können mit Schiebereglern (englisch: scrollbars) bestimmt werden.
- Konstruktoren
  - **`JScrollPane(Component view)`**
    erzeugt ein **`JScrollPane`**-Objekt, das den Inhalt der Komponente **`view`** anzeigt. Vertikale und/oder horizontale Schieberegler erscheinen erst, wenn der Inhalt der Komponente zu groß f ür die Darstellung wird.
  - **`public JScrollPane(Component view, int vScr, int hScr)`**
    erzeugt ein JScrollPane-Objekt, das den Inhalt der Komponente view anzeigt. Die Parameter **`vScr`** und **`hScr`** legen fest, wann vertikale und horizontale Schieberegler erscheinen.
- Für **`vScr`** und **`hScr`** können wie üblich vordefinierte Konstanten aus der Klasse **`JScrollPane`** verwendet werden.

Die Klasse JScrollPane

- Beispiel

```
import java.awt.*;
import javax.swing.*;
public class FrameMitScrollText extends JFrame {
  Container c;
  JLabel info;
  JTextArea ta;
  JScrollPane sp;
  Font f = new Font("SansSerif",Font.BOLD+Font.ITALIC,16);
  public FrameMitScrollText() {
    c = getContentPane();
    info = new JLabel("Hier kann Text bearbeitet werden");
    ta = new JTextArea("Einiges an Text steht schon rum.");
    ta.setFont(f);
    ta.setLineWrap(true);
    ta.setWrapStyleWord(true);
    sp = new JScrollPane(ta);
    c.add(info,BorderLayout.NORTH);
    c.add(sp);
  }
```

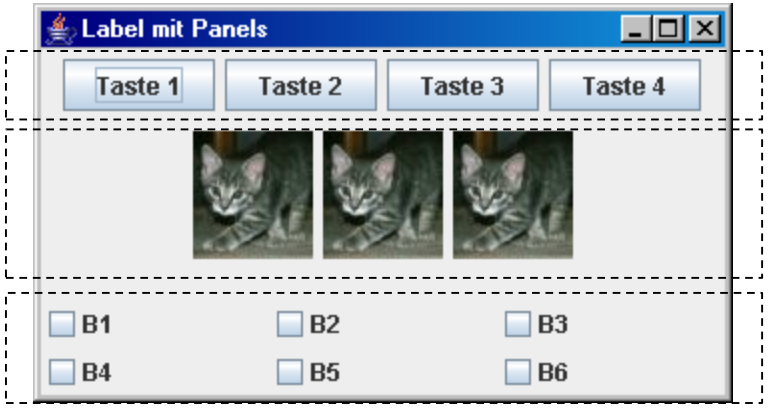
10.2.13 Die Klasse `JPanel`

- Eigentlich keine echte Grundkomponente, sondern ein Container
- Kann selbst wieder Komponenten enthalten und dient hauptsächlich der Strukturierung von Oberflächen
- Im Gegensatz zu den Top-Level-Containern ist **`JPanel`** aber lightweight!
- Voreingestelltes Layout ist **`Flow-Layout`**
- Konstruktoren
  - **`public JPanel()`**
    erzeugt einen leeren Container.
  - **`public JPanel(LayoutManager layout)`**
    erzeugt einen leeren Container mit dem angegebenen Layout.

Die Klasse JPanel

- Beispiel

```
import java.awt.*;
import javax.swing.*;
public class FrameMitPanels extends JFrame {
  Container c;
  public FrameMitPanels() {
    c = getContentPane();
    JPanel jp1 = new JPanel(), jp2 = new JPanel(),
           jp3 = new JPanel(new GridLayout(2,3));
    for (int i=1; i<=4; i++)
      jp1.add(new JButton("Taste " + i));
    Icon bild = new ImageIcon("babycatSmall.jpg");
    for (int i=1; i<=3; i++)
      jp2.add(new JLabel(bild));
    for (int i=1; i<=6; i++)
      jp3.add(new JCheckBox("B"+i));
    c.add(jp1,BorderLayout.NORTH);
    c.add(jp2,BorderLayout.CENTER);
    c.add(jp3,BorderLayout.SOUTH);
  }
```



10.2.14 Spezielle Top-Level-Container

- **`JFrame`** Fenster mit Rahmen
- **`JWindow`** Fenster ohne Rahmen
- **`JDialog`** Fenster zur Abwicklung eines Dialogs
- Standard-Methoden für Top-Level-Container
  - **`getContentPane()`, `setDefaultCloseOperation(…)`, `setTitle(…)`** wie bereits mehrfach für Frames eingesetzt
  - **`public void pack()`**
    passt die Größe des Fensters so an, dass gerade noch alle darin platzierten Komponenten Platz finden.
  - **`public void setJMenuBar(JMenuBar menubar)`**
    setzt die Menüleiste des aufrufenden Fenster-Objekts.
- **`JDialog`**-Objekte können ***modal*** gestaltet werden, d. h. das übergeordnete Fenster ("Owner", Besitzer des Dialogs) kann gesperrt werden, bis das Dialogfenster abgearbeitet ist.
  - **`public JDialog(Frame owner, boolean modal)`**
    erzeugt ein modales (falls **`modal`** den Wert **`true`** hat) oder nicht-modales (andernfalls) Dialog-Fenster, das dem **`Frame`**-Objekt **`owner`** gehört.

Spezielle Top-Level-Container

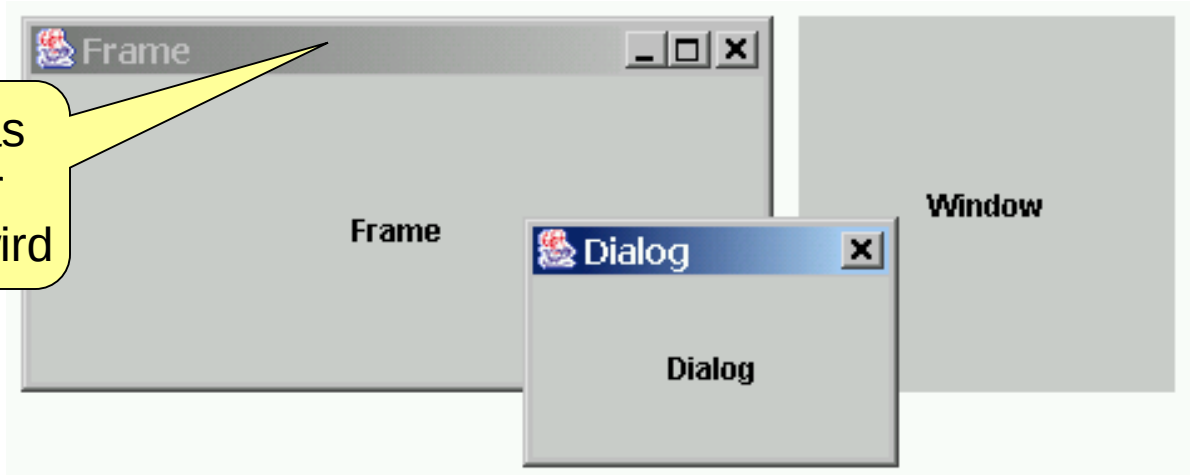
- Beispiel

```
import javax.swing.*;
public class TopLevelContainer {
  public static void main(String[] args) {
    // Hauptfenster erzeugen und beschriften
    JFrame f = new JFrame("Frame");
    f.getContentPane().add(new JLabel("Frame",JLabel.CENTER));
    f.setSize(300,150);
    f.setLocation(100,100);
    f.setVisible(true);
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    // Unterfenster (Window) erzeugen und beschriften
    JWindow w = new JWindow(f);
    w.getContentPane().add(new JLabel("Window",JLabel.CENTER));
    w.setSize(150,150);
    w.setLocation(410,100);
    w.setVisible(true);
```

Spezielle Top-Level-Container

- Beispiel (Fortsetzung)

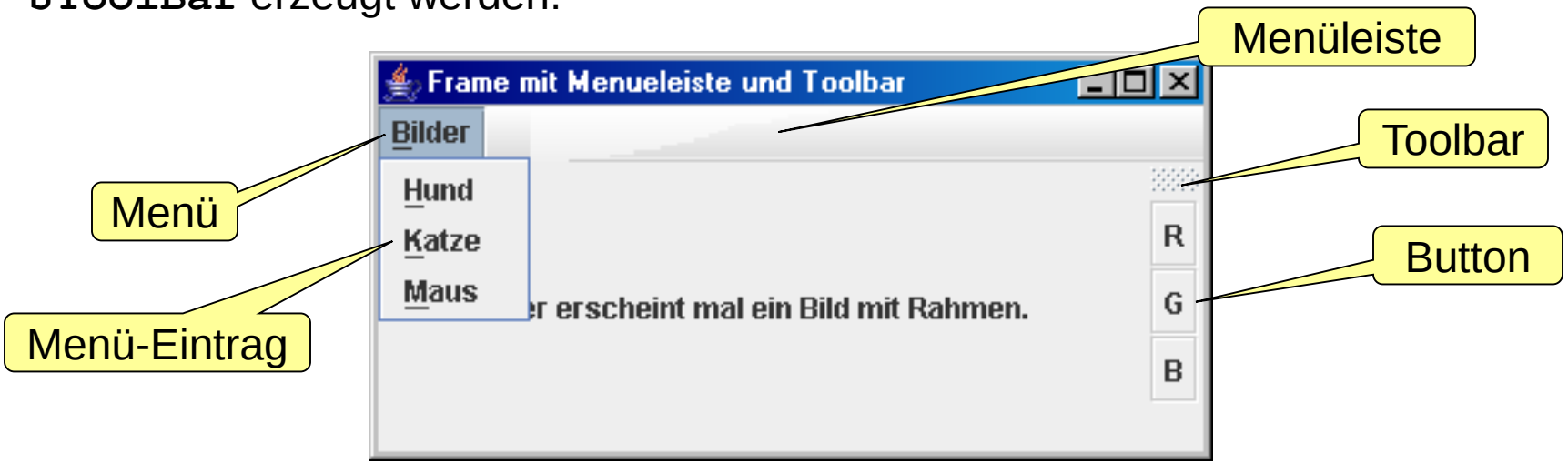
```
    // Modales Unterfenster (Dialog) erzeugen und beschriften
    JDialog d = new JDialog(f,true);
    d.getContentPane().add(new JLabel("Dialog",JLabel.CENTER));
    d.setTitle("Dialog");
    d.setSize(150,100);
    d.setLocation(300,180);
    d.setVisible(true);
    d.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  }
}
```



inaktiv, bis das Dialogfenster geschlossen wird

10.2.15 Menüs und Toolbars

- ***Menüleisten*** in einem Fenster können als Objekte der Klasse **`JMenuBar`** erzeugt werden.
- Die einzelnen ***Menüs*** (***Pulldown-Menüs***) der Menüleiste können durch Objekte der Klasse **`JMenu`** erzeugt werden.
- Die ***Einträge*** eines Menüs können durch Objekte der Klasse **`JMenuItem`** erzeugt werden.
  - Da **`JMenu`** eine Unterklasse von **`JMenuItem`** ist, können die Menü-Einträge selbst wieder Menüs sein!
- Verschiebbare ***Werkzeugleisten*** (***Toolbars***) können als Objekte der Klasse **`JToolBar`** erzeugt werden.



Menüs und Toolbars

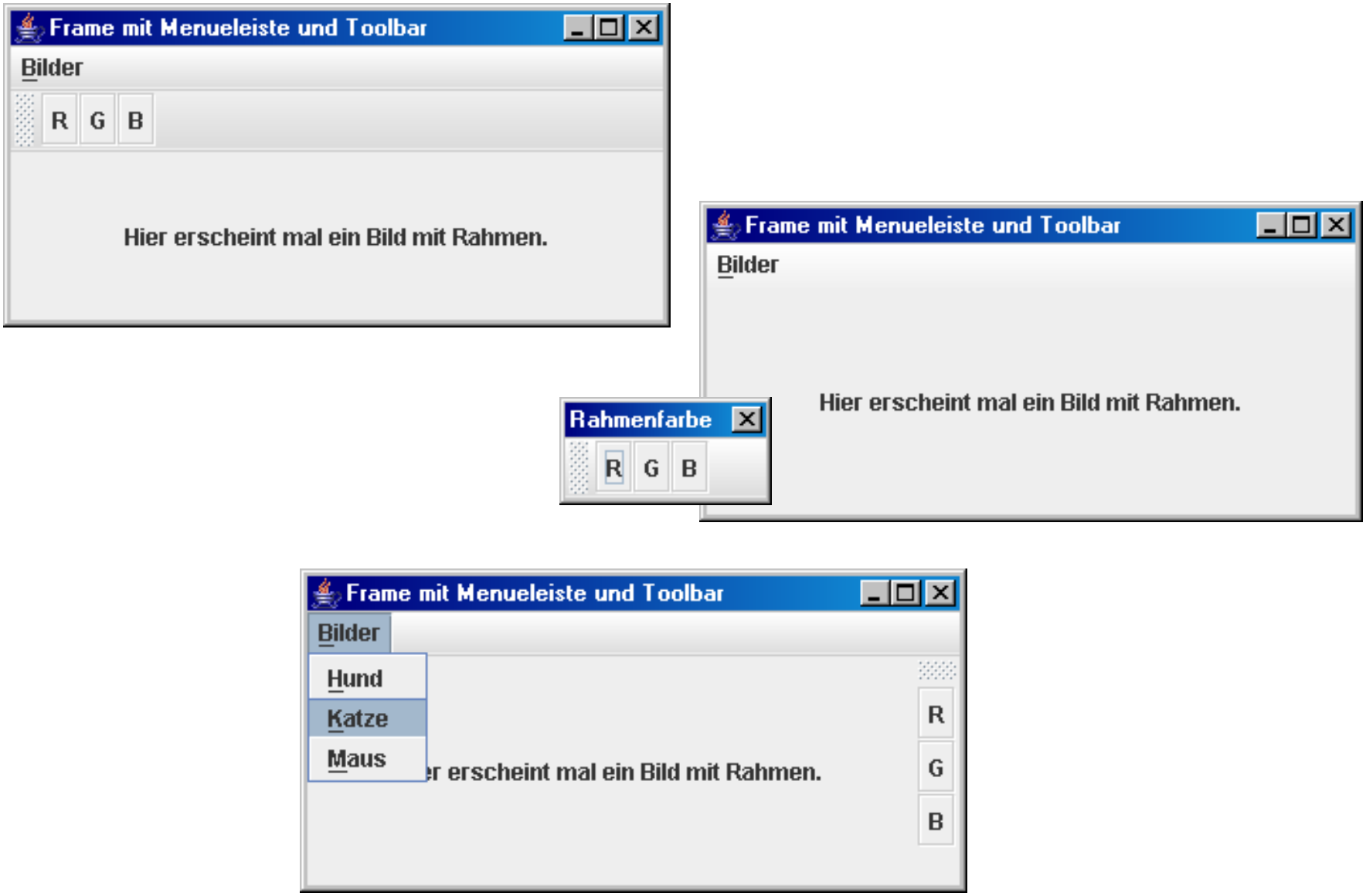
- Beispiele

```
JMenuBar menuBar = new JMenuBar();
JMenu menu = new JMenu("Bilder");
menu.setMnemonic(java.awt.event.KeyEvent.VK_B);
JMenuItem menuItem = new JMenuItem("Hund");
menuItem.setMnemonic(java.awt.event.KeyEvent.VK_H);
menu.add(menuItem);
...
menuBar.add(menu);
setJMenuBar(menuBar);
```

```
...
```

```
JToolBar toolBar = new JToolBar("Rahmenfarbe");
JButton button = new JButton("R");
button.setToolTipText("roter Rahmen");
toolBar.add(button);
...
c.add(toolBar, BorderLayout.NORTH);
```

Menüs und Toolbars



10.2.16 Vordefinierte, anpassbare Dialoge

- Die Klasse JOptionPane stellt verschiedene vordefinierte und meist modale Dialoge zur Verfügung
  - Bestätigung einer Informationsmeldung
  - Einlesen von Werten

- Diese Dialoge werden meist dynamisch während eines Programmlaufs erzeugt

- Beispiel: (Aufruf aus einer inneren Klasse der Klasse **`TestFrame`**)

```
JOptionPane.showConfirmDialog(this, "Ist das o.k. ?");
```