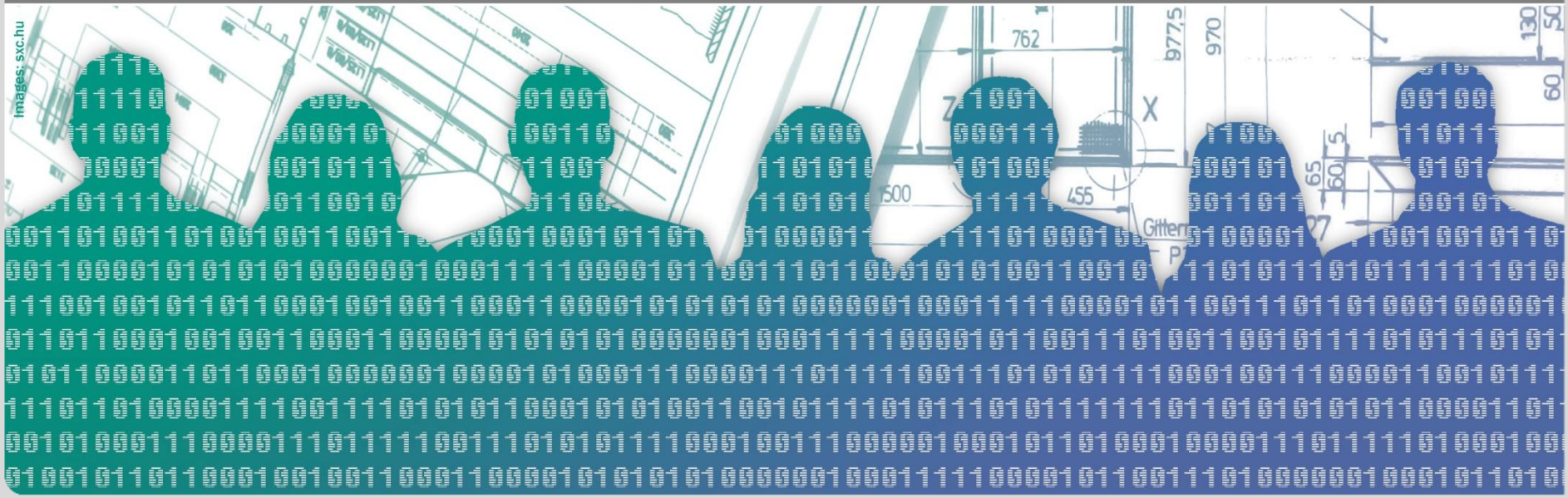
Kapitel X - Grafische Benutzeroberflächen mit Java und Swing

SWT I – Sommersemester 2010

Walter F. Tichy, Andreas Höfer, Korbinian Molitorisz

IPD Tichy, Fakultät für Informatik



Lernziele

- Großen Bibliotheken einsetzen können.
- Grafische Benutzeroberflächen (GBO, engl. Graphical User Interface, GUI) gestalten und bauen können.
- Ereignisgetriebene Software entwickeln können.
- Die Vorlesung liefert dazu die Konzepte und Hilfe zur Selbsthilfe.
- Merke: Dieses Instrument erfordert Übung!

Vorrausetzungen

- Grundvoraussetzung:
  - Java SE Development Kit (JDK) in einer aktuellen Version (JDK 6 Update 20): http://java.sun.com/javase/downloads/index.jsp
  - Alle für die Erstellung von Swing-Anwendungen benötigten Pakete werden darin mitgeliefert.
- Optional kann eine Entwicklungsumgebung verwendet werden.
  - Empfehlung – Eclipse (Vers. 3.5.2) for Java Developers (oder Classic): http://www.eclipse.org/downloads/
  - Alternativ – Netbeans (Vers. 6.8): http://www.netbeans.org/
- Verwenden Sie für das Erlernen von Swing in Ihrem eigenen Interesse **nicht** den Oberflächen-Designer von Netbeans.

Literatur

- Informationen zu Swing finden Sie in der aktuellen Java Dokumentation unter: http://java.sun.com/javase/6/docs/api/javax/swing/package-summary.html
- Einen Lehrgang zu Swing finden Sie unter: http://java.sun.com/docs/books/tutorial/uiswing/
- Die Java-Dokumentation können Sie auch herunterladen (Abschnitt „Additional Resources“): http://java.sun.com/javase/downloads/index.jsp
- Viele hilfreiche Code-Beispiele finden Sie unter: http://www.java2s.com/

Hilfe zur Selbsthilfe

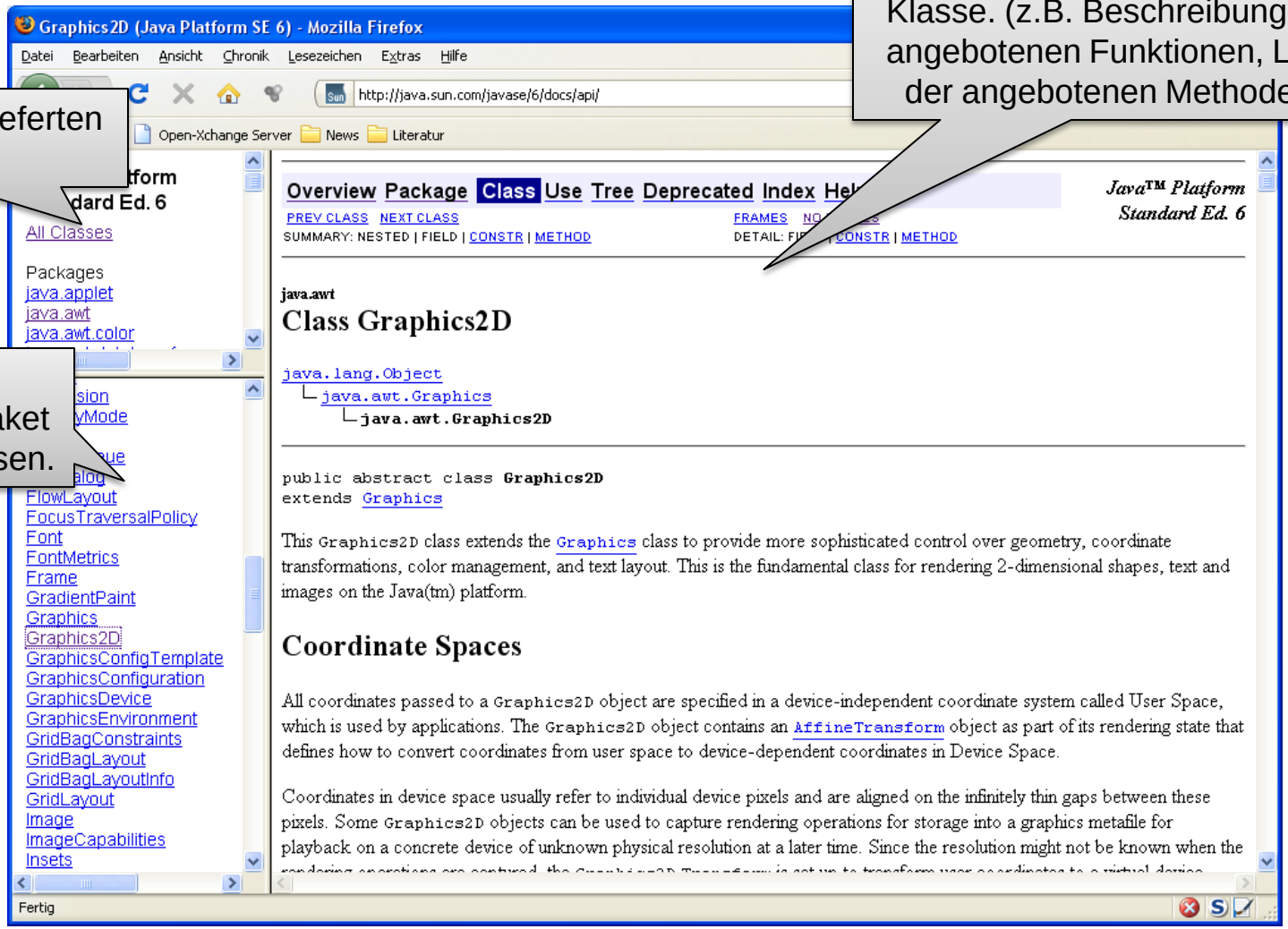
- Bei Problemen mit der Java API ist die Dokumentation von Sun sehr hilfreich:
  http://java.sun.com/javase/6/docs/api/
- Die Seite ist in drei Bereiche unterteilt:
  - Im linken oberen Bereich finden Sie alle Pakete, die mit der Java-Laufzeitumgebung mitgeliefert werden.
  - Im linken unteren Bereich sehen Sie alle Klassen, die zum ausgewählten Paket gehören.
  - Im rechten Bereich sehen Sie alle Details zur ausgewählten Klasse.

Hilfe zur Selbsthilfe

Liste aller mitgelieferten Pakete.

Liste aller im ausgewählten Paket enthaltenen Klassen.

Details zur ausgewählten Klasse. (z.B. Beschreibung der angebotenen Funktionen, Liste der angebotenen Methoden)



Hilfe zur Selbsthilfe

- Die Details zur Klasse enthalten unter Anderem folgende Informationen:
  - Beschreibung der angebotenen Funktionalität (ggf. mit Code-Beispielen).
  - Übersicht über alle angebotenen Konstanten, Konstruktoren und Methoden.
  - Beschreibung der Konstanten, Konstruktoren und Methoden, sowie deren Parameter und Rückgabewerte.

Hilfe zur Selbsthilfe

- Folgende Abschnitte der Dokumentation könnten für Sie interessant sein:
  - `java.awt.Graphics2D`
  - `java.awt.ActionEvent`, `java.awt.ActionListener`
  - `java.awt.BorderLayout`, `java.awt.FlowLayout`,...
  - `javax.swing.JPanel`, `javax.swing.JButton`,...
- Informativ ist auch folgende Einführung: http://java.sun.com/docs/books/tutorial/2d/index.html
- Code-Beispiele liefern oft hilfreiche Informationen, wenn die Java-Dokumentation einmal nicht weiterhilft.

Grafische Benutzeroberflächen in Java

- Es existieren verschiedene Schnittstellen für Java zur Erstellung von GBOs:
  - In Java integrierte (mitgelieferte) Schnittstellen:
    - **A**bstract **W**indow **T**oolkit (AWT)
    - Swing
  - Alternative Schnittstellen (nicht in Java integriert):
    - **S**tandard **W**idget **T**oolkit (SWT)
    - JFace (Erweiterung von SWT)

Leicht- vs. schwergewichtige Komponenten

Leichtgewichtige Komponenten

- Sind nicht an plattformabhängige Komponente gebunden
- Müssen letztendlich auf schwergew. Komponente gezeichnet werden.
- Sehen auf allen Plattformen gleich aus.
- Aussehen der Zielplattform zu emulieren ist aufwändig.

Schwergewichtige Komponenten

- Sind an eine plattformabhängige Komponente gebunden
- Auf Plattform nicht angebotene Komponenten müssen „von Hand“ nachgebaut werden.

Abstract Window Toolkit (schwergewichtig)

- Verwendet die von der zugrundeliegenden Plattform angebotenen Steuerelemente.
- Es gibt nur Steuerelemente, die auf allen von AWT unterstützen Plattformen existieren.
- Erstellung von komplexen GBO sehr aufwändig: Steuerelemente, wie z.B. Fortschrittsbalken, müssen von Hand erstellt werden.
- Aber: (Fast) genauso schnelle GBO wie bei nativen Anwendungen.

Swing (leichtgewichtig)

- Verwendet nur Fenster und Zeichenoperationen der zugrundeliegenden Plattform.
- Swing-Anwendungen sind ressourcenhungrig und oft langsamer in der Bedienung als AWT- oder andere plattformabhängige Anwendungen.
  - Diese Problem wird oft durch ungeschickte Programmierung verstärkt:
    - z.B. `JFileChooser`: Jedesmal neu erzeugen statt nur neu konfigurieren

Standard WidgetToolkit (schwergewichtig)

- Verwendet Steuerelemente von der zugrunde liegenden Plattform.
- Versucht, das Beste von AWT und Swing zu verbinden.
- Geschwindigkeit und Aussehen der Oberfläche wie bei nativen Anwendungen.
- Effizienzprobleme auf Nicht-Windows-Plattformen wegen fehlender Funktionen.
  Beispiel: Z-Ordering bei GTK+
- SWT-Bibliothek muss mit der Anwendung mitgeliefert werden.

JFace

- Baut auf SWT auf und setzt aus den SWT-Steuerelementen komplexere Steuerelemente zusammen.
- Verwendet das Architekturmuster „Model/View/Controller“ für fast alle Steuerelemente.
- JFace Bibliothek muss zusammen mit der Anwendung mitgeliefert werden.
- Aktuelle Versionen sind von zusätzlichen Eclipse-Bibliotheken abhängig, so dass weitere Dateien mitgeliefert werden müssen.

Exkurs: Model/View/Controller

- MVC ist eine Klassenkombination zur Konstruktion von Benutzerschnittstellen (entstanden zuerst in Smalltalk).
  - Modell (Model):
    Anwendungsobjekt
  - Sicht (View):
    Darstellung des Modells auf dem Bildschirm (evtl. mehrfach)
  - Steuerung (Controller):
    Definiert Reaktion der Benutzerschnittstelle auf Eingaben
- Mehr dazu im Kapitel „Entwurfsmuster“

Swing – Beispielanwendung (1)

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

public class HalloWelt extends JFrame {

	public HalloWelt() {
		super("Hallo Welt!");

		JButton schaltflaeche = new JButton("01 + 01 = ?");

		schaltflaeche.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e) {
		   JOptionPane.showMessageDialog(null, "01 + 01 = 10", "Antwort:",
		             JOptionPane.QUESTION_MESSAGE);
		   }
		});

		this.add(schaltflaeche);
		this.setDefaultCloseOperation(EXIT_ON_CLOSE);
		this.pack();
	}

	public static void main(String[] args) {
		HalloWelt fenster = new HalloWelt();
		fenster.setVisible(true);
	}
}
```

Importieren der benötigten Pakete

Erben von der Hauptfensterklasse

Konstruktor von JFrame mit Titel des Fensters aufrufen.

Erstellen einer Schaltfläche mit der angegebenen Beschriftung.

Angeben, was bei einer Aktion auf der Schaltfläche passieren soll.

Schaltfläche zum Fenster hinzufügen.

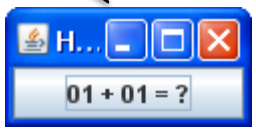
Aktion angeben, die beim Schließen des Fensters ausgelöst werden soll.

Fenstergröße anpassen, Anordnung der Komponenten an Fenstergröße anpassen.

Zeigt das erstellte Fenster an.

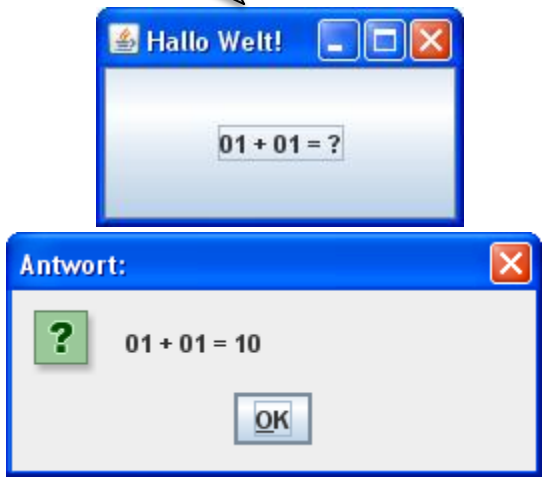
Swing – Beispielanwendung (2)

Nach dem Start.



Nach manueller Vergrößerung.

Nach Klick auf die Schaltfläche.



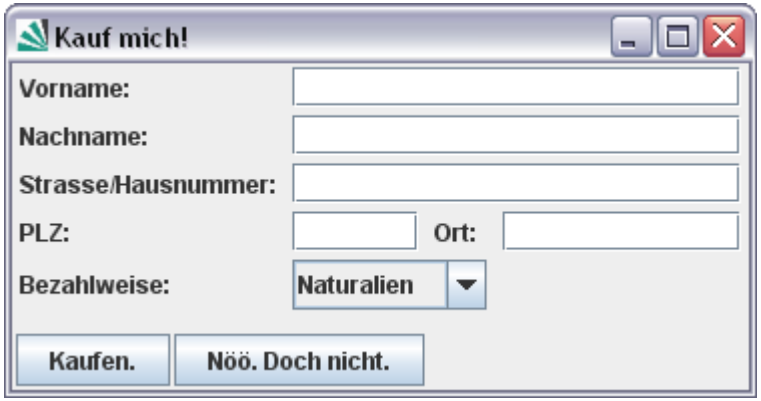
Erstellen von Fenstern: JFrame

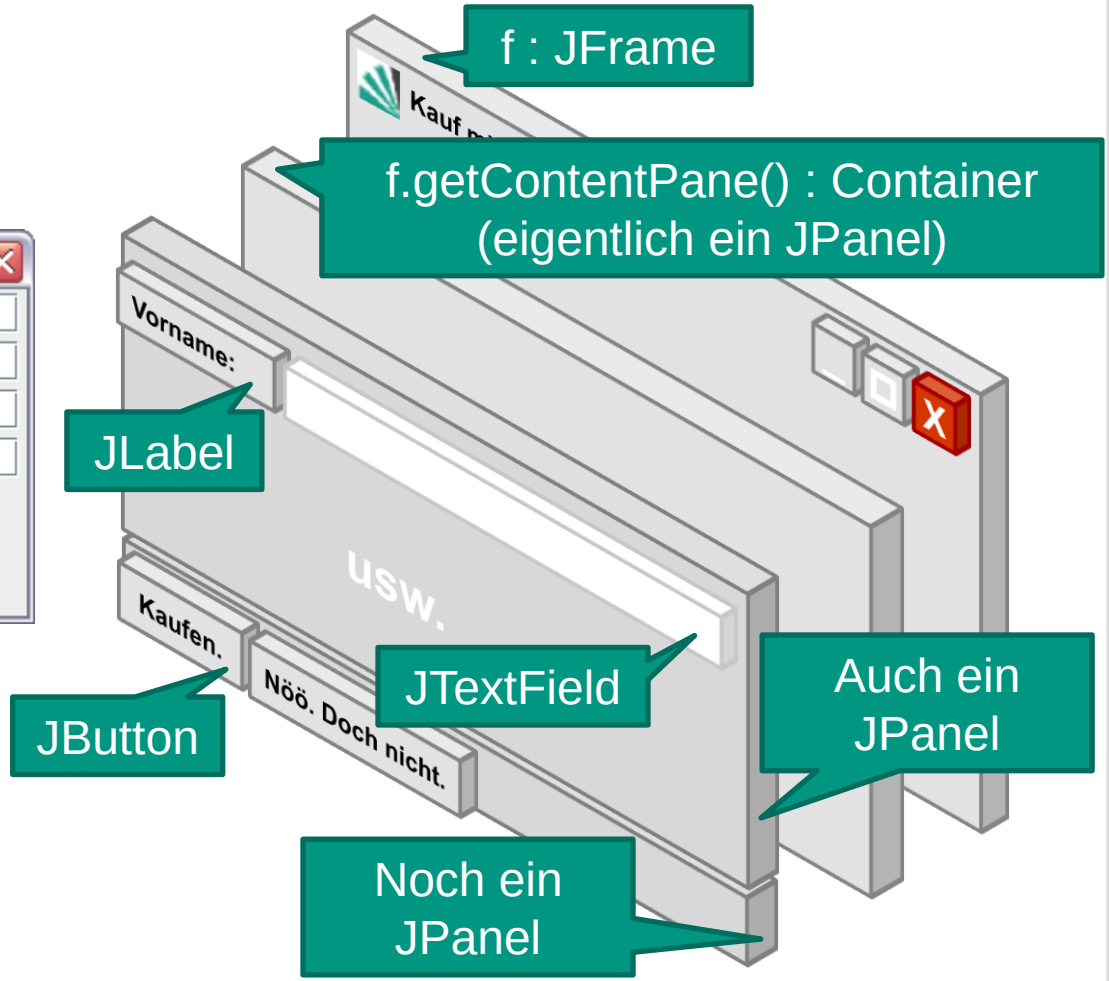
- Fenster werden in Swing i.d.R. mit der Klasse `JFrame` erstellt.
- Sie können entweder
  - eine eigene Klasse erstellen, welche von `JFrame` erbt oder
  - selbst ein `JFrame` Objekt erstellen und mit diesem Arbeiten.
- Danach können Sie weitere Eigenschaften des Fensters festlegen sowie Steuerelemente platzieren.
- Der Aufruf von `pack()` nach dem Hinzufügen der Steuerelemente ist optional.

Erstellen von Fenstern: JFrame

- Bei einem Klick auf das Schließen-Symbol in der Titelleiste eines `JFrame`-Fensters wird das Fenster standardmäßig nur versteckt (`setVisible(false)`).
- Soll stattdessen die Anwendung beendet werden, müssen Sie dies explizit angeben:
  `setDefaultCloseOperation(EXIT_ON_CLOSE);`
- `EXIT_ON_CLOSE` ist eine Konstante in `JFrame`.

Aufbau von Dialogen





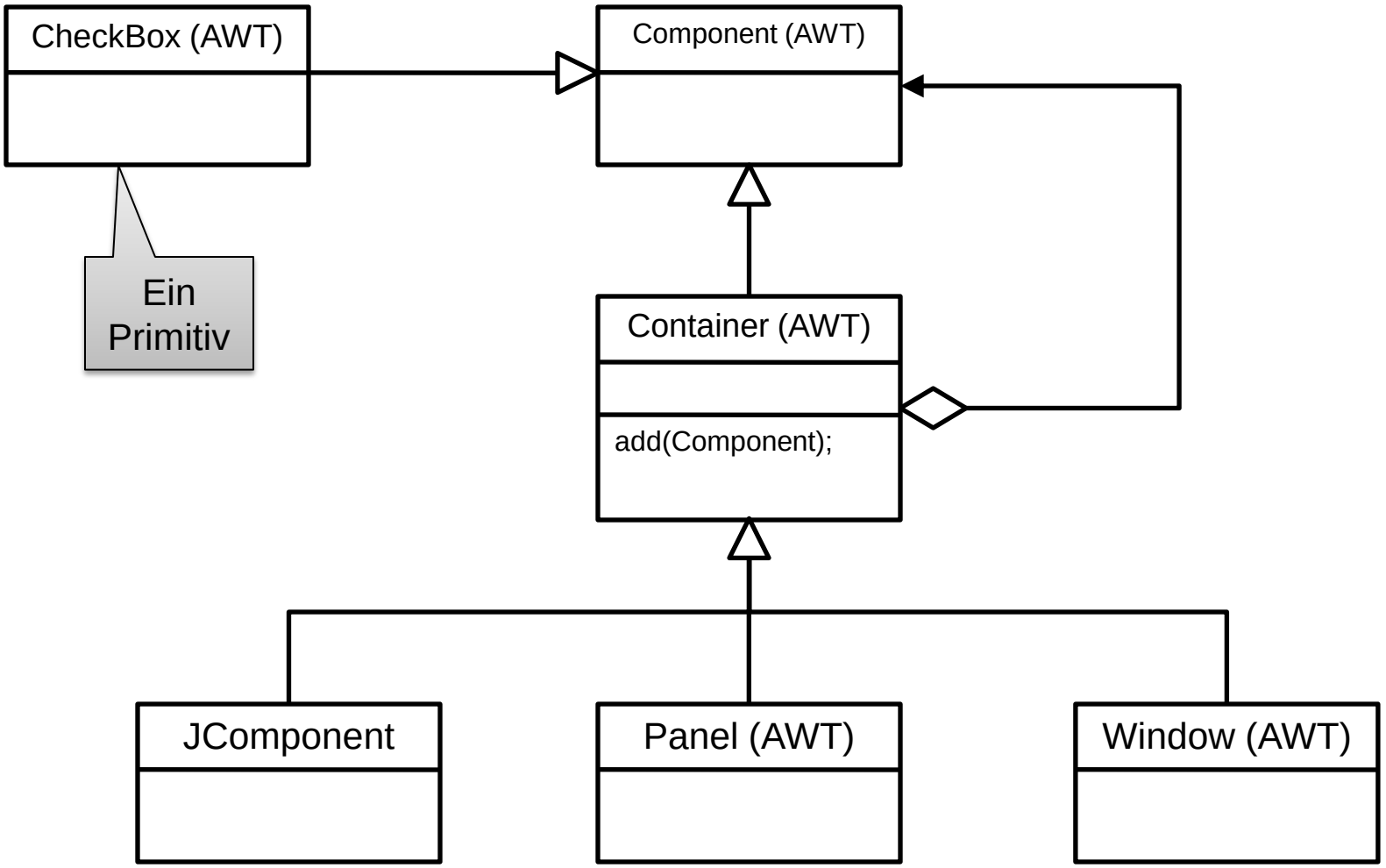
Alles ist ein Container

- Alle Steuerelemente von Swing erben von der AWT-Klasse `Container`.
- Aus diesem Grund ist es möglich, in jedes Swing-Steuerelement weitere Steuerelemente hinzuzufügen.
- **Beispiel:** Textfeld in einer Schaltfläche
- Sonderfall: `JFrame.add` delegiert an das enthaltene `JPanel` genannt `ContentPane`.

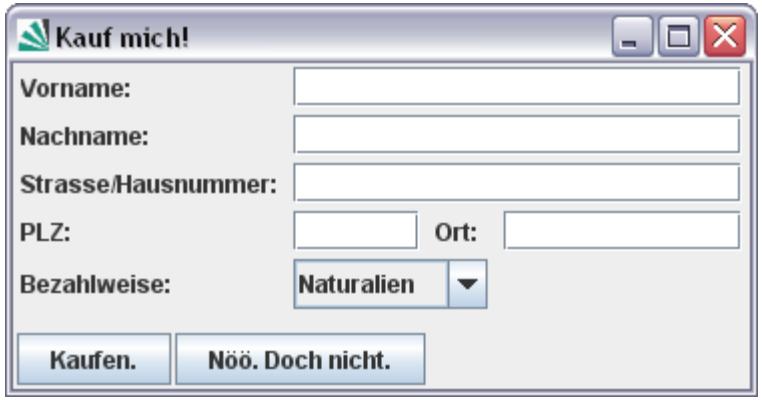
Entwurfsmuster „Kompositum“

- Swing verwendet zur Komposition von Oberflächen das Entwurfsmuster „Kompositum“.
- Das Entwurfsmuster „Kompositum“ wird dazu verwendet, Objekte zu Baumstrukturen zusammenzufügen.
- Es ermöglicht die einheitliche Behandlung von Primitiven (bspw. `java.awt.CheckBox`) und deren Kompositionen.
- Es können leicht weitere Primitive hinzugefügt werden.
- **Hinweis:** Entwurfsmuster werden in einer späteren Vorlesung genauer behandelt.

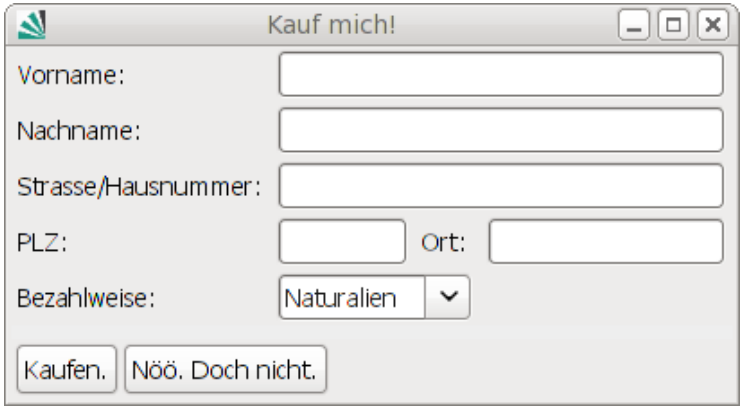
Entwurfsmuster „Kompositum“



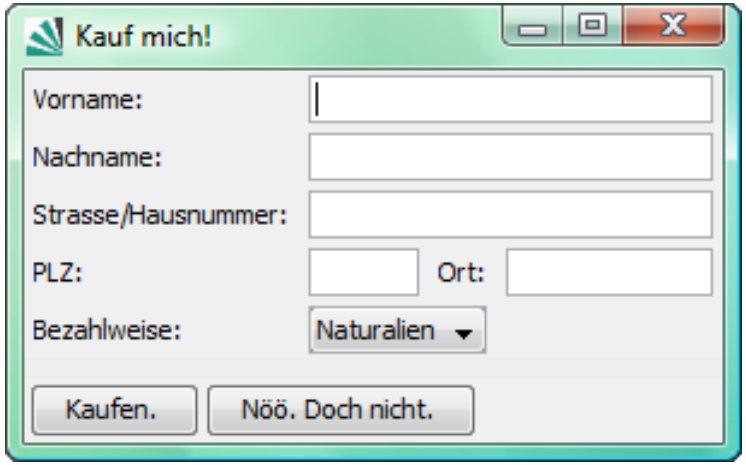
Anmutung: Look and Feel



javax.swing.plaf.metal.MetalLookAndFeel



com.sun.java.swing.plaf.gtk.GTKLookAndFeel



com.sun.java.swing.plaf.windows.WindowsLookAndFeel

Steuerelemente in Swing

- Swing stellte eine Vielzahl an Steuerelementen zur Verfügung.
- Mit wenigen Ausnahmen (z.B. den Fenstern) sind alle Steuerelemente leichtgewichtig.
- Steuerelemente werden mit Hilfe der Methode `add(Component c[, …])` zu einem Container hinzugefügt.
- Für die Anordnung der Steuerelemente in einem Container stellt Swing verschiedene **Anordner** (engl. *Layout Manager*) bereit.

Anordner: Layout Manager

- Swing stellt zur Anordnung der Steuerelemente mehrere Anordner zur Verfügung.
- Dazu gehören einfache Anordner, wie bspw. das `FlowLayout`, bei welchem die Steuerelemente zeilenweise angeordnet werden, bis die gesamte Breite des Fensters ausgenutzt ist und dann erst die nächste Zeile mit Steuerelementen gefüllt wird.
- Durch geschickte Verwendung bzw. Schachtelung mehrerer Anordner sind auch komplexe Anordnungen von Steuerelementen möglich.
- **Hinweis:** Verwende immer den einfachsten Anordner, der den Anforderungen genügt. Wozu sich mit `GridBagLayout` herumärgern, wenn ein `BorderLayout` ausreicht?

Einsatz von Anordnern

- **Wichtig**: Der Entwickler muss den geeigneten Anordner selbst festlegen. Swing gibt je nach Steuerelement unterschiedliche Anordner als Standard vor.
  Beispiel:
  `JPanel` verwendet `FlowLayout`
  Eine `JContentPane` hingegen `BorderLayout`
- Abhängig vom verwendeten Anordner sind beim hinzufügen von Steuerelementen weitere Angaben zu machen.
- Bei Verwendung eines `GridBagLayout` müssen bspw. `GridBagConstraints` angegeben werden, damit der Anordner das Steuerelement korrekt positionieren kann.

Weitere Informationen zum Einsatz von Anordnern

- Mehr Informationen zu den angebotenen Anordnern sowie deren Verwendung finden Sie in der Java Dokumentation im Paket `java.awt`:
  http://java.sun.com/javase/6/docs/api/java/awt/package-summary.html

- Die Anordner finden Sie am einfachsten, wenn Sie sich die Klassen anschauen, welche die Schnittstellen `LayoutManager` oder `LayoutManager2` implemetieren:
  http://java.sun.com/javase/6/docs/api/java/awt/LayoutManager.html
  http://java.sun.com/javase/6/docs/api/java/awt/LayoutManager2.html

Wie geht's weiter?

- Bis jetzt können Sie mit Containern, Anordnern und Steuerelementen eine Oberfläche erstellen.
- Aber wie bekommt man mit, ob der Benutzer auf eine Schaltfläche geklickt hat oder die Maus bewegt?
- Dafür bietet Java Ereignisse und Beobachter an.

Ereignisse in Java/AWT/Swing

- Es gibt zwei Arten von **Ereignissen** (engl. *event*):
- Primitive Ereignisse(engl. *low-level event*):
  - Ereignisse auf Ebene des Betriebssystems
    Beispiele: Mausbewegung, Fokus auf eine Komponente
- Semantische Ereignisse (engl. *high-level event*):
  - In Swing werden diese Ereignisse oft durch primitive Ereignisse ausgelöst, die in semantische Ereignisse überführt werden
    Beispiele: Klick auf eine Schaltfläche, Markieren von Text in einem Textfeld.

Behandlung von Ereignissen in Java

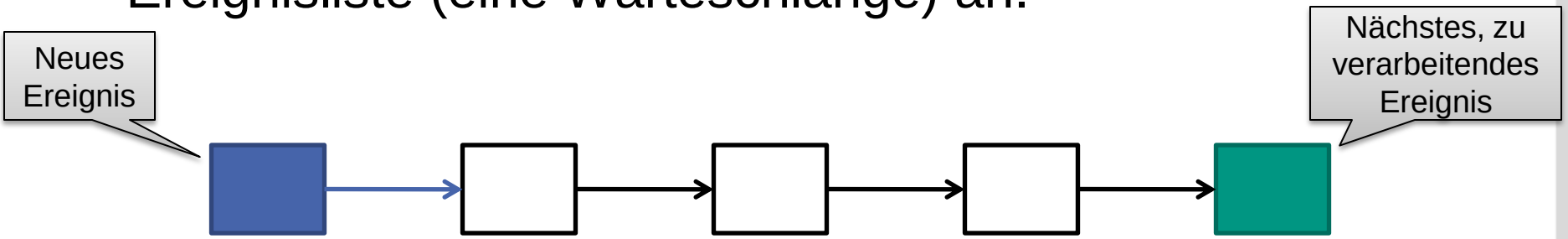
1. Benutzer bewegt die Maus, betätigt die Tastatur

2. Betriebssystem erzeugt daraus ein primitives Ereignis (Δx und Δy der Mausbewegung, welcher Knopf wurde wann gedrückt) und übergibt es an die Ereignissteuerung der Java-Plattform.

3. Die Ereignissteuerung wandelt das primitive Ereignis in ein Ereignisobjekt um (`EventObject` oder Unterklasse davon). Das Ereignisobjekt enthält min. die Ereignisquelle, d.h. das Objekt, dem das Ereignis zugeschrieben wird. Bei grafischen Benutzeroberflächen ist dies meist ein grafisches Objekt.

Behandlung von Ereignissen in Java

4. Die Ereignisbehandlung hängt das Ereignis an die Ereignisliste (eine Warteschlange) an.



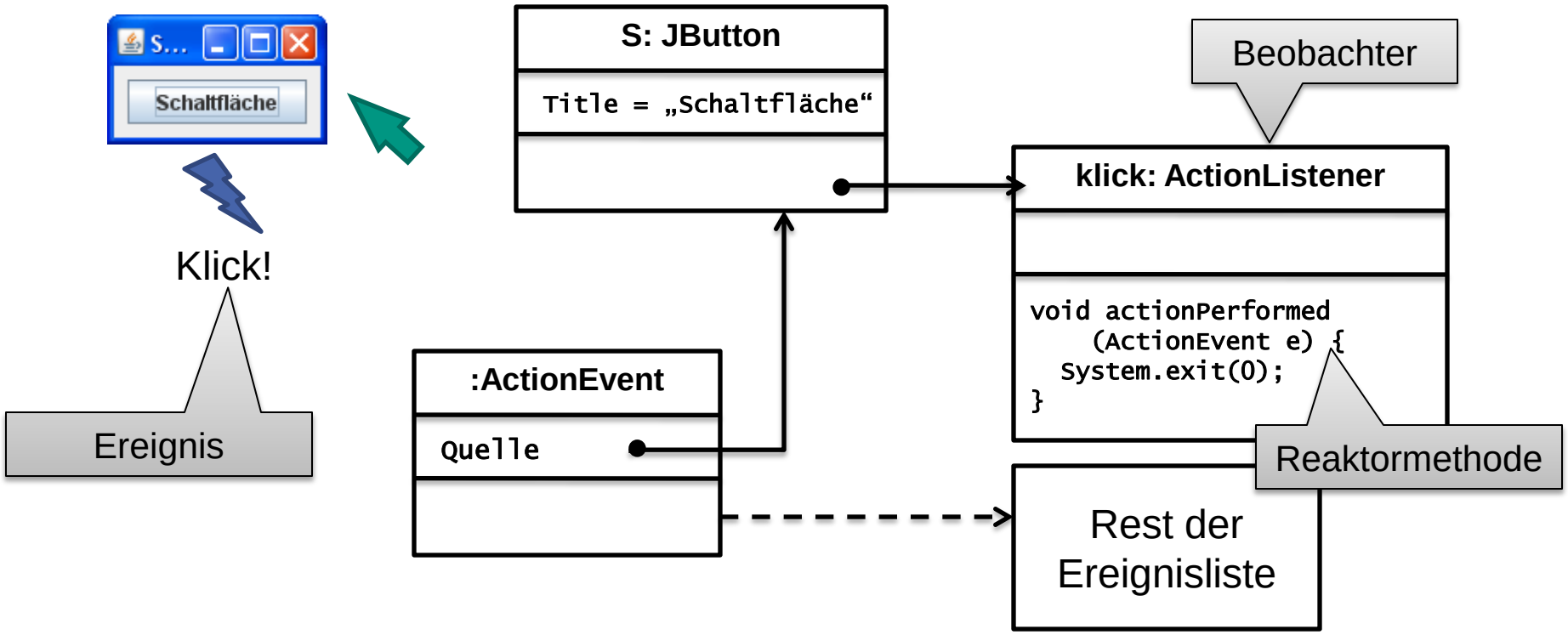
5. Ein separater Prozess (Ereignisschleife, engl. *Eventloop*) behandelt ein Ereignis nach dem anderen
   - Ereignis abholen
   - Ereignisquelle auslesen
   - Beobachter, die an der Ereignisquelle registriert sind und auf das Ereignis warten, bestimmen und die geeignete Reaktormethode mit dem Ereignisobjekt als Parameter, aufrufen.

Behandlung von Ereignissen in Java: Die Ereignisliste

- Der Ereignisbehandler kann mehrere Ereignisse in der Ereignisliste kombinieren.
  Beispiel: Mehrere Mausbewegungen hintereinander können zu einem Ereignis zusammengefasst werden.
- Die Ereignisliste kann auch aus dem eigenen Programm manipuliert werden.
  Beispiel: Der Aufruf der Methode `repaint()` fügt ein Ereignis zum Neuzeichnen der zugeh. Komponente ein.
- Die Ereignisliste puffert Ereignisse, wenn sie nicht schnell genug behandelt werden können.

Behandlung von Ereignissen in Java: Übergabe eines Ereignisobjektes

- Die Behandlung des Ereignisses wird von der Ereignisquelle an die Beobachter delegiert.



Behandlung von Ereignissen in Java: Vorteile der Delegation

- Graphische Darstellung und Anwendungskern können sauber getrennt werden.
- Die Zuordnung der Ereignisse ist sehr einfach
  - Ereignisquelle wird automatisch bestimmt

Ereignisse in Java/AWT/Swing

- Nahezu jede Swing-Komponente liefert Ereignisse.
- Die wichtigsten Ereignisklassen sind:
  - **ComponentEvent**: Verschieben, vergrößern, ... der Komponente
  - **FocusEvent**: Erhalten oder verlieren des Fokus (Mauszeiger)
  - **KeyEvent**: Taste gedrückt, Taste losgelassen, …
  - **MouseEvent**: Maustaste gedrückt, Maus wird bewegt …

Beispiel: MouseEvent geht an MouseListener

- In der Java Dokumentation sehen Sie, welche Informationen Sie von einem MouseEvent bekommen, wenn dieser bspw. einer `MouseListener`-Methode übergeben wird:

Method Summary	
int	**getButton**() Returns which, if any, of the mouse buttons has changed state.
int	**getClickCount**() Returns the number of mouse clicks associated with this event.
Point	**getLocationOnScreen**() Returns the absolute x, y position of the event.
static String	**getMouseModifiersText**(int modifiers) Returns a String describing the modifier keys and mouse buttons that were down during the event, such as "Shift", or "Ctrl+Shift".
Point	**getPoint**() Returns the x,y position of the event relative to the source component.
int	**getX**() Returns the horizontal x position of the event relative to the source component.
int	**getXOnScreen**() Returns the absolute horizontal x position of the event.
int	**getY**() Returns the vertical y position of the event relative to the source component.
int	**getYOnScreen**() Returns the absolute vertical y position of the event.
boolean	**isPopupTrigger**() Returns whether or not this mouse event is the popup menu trigger event for the platform.
String	**paramString**() Returns a parameter string identifying this event.
void	**translatePoint**(int x, int y) Translates the event's coordinates to a new position by adding specified x (horizontal) and y (vertical) offsets.

Oder ein doppelter Mausklick?

Welchen Wert hat die x-Koordinate des Mauszeigers (innerhalb dieses Steuerelements)

Welchen Wert hat die y-Koordinate des Mauszeigers (innerhalb dieses Steuerelements)

War es ein „rechter“ Mausklick?

Ereignisse in Java/AWT/Swing

- Die so ausgelösten Ereignisse können vom Entwickler abgefangen und weiterverwendet werden.
- Je nach Steuerelement werden verschiedene Ereignisse ausgelöst, die von einem **Beobachter** (engl. *listener*) aufgefangen werden können.
- Abhängig von der Art des abzufangenden Ereignisses werden verschiedene Beobachter benötigt.

Ereignisse in Java/AWT/Swing

- Die wichtigsten Beobachter sind:
  - Der `ActionListener` reagiert auf alle ActionEvent, die vom Benutzer ausgelöst werden.
    Beispiel: Der Benutzer klickt auf eine Schaltfläche.
  - Der `ChangeListener` reagiert auf Änderungen am zugehörigen Objekt.
    Beispiel: Der Benutzer ändert den Wert eines Rollbalken.
  - Der `KeyListener` reagiert auf Tastendrücke des Benutzers.
    Beispiel: Der Benutzer drückt die Tastenkombination [Strg] + [Esc]
  - Der `MouseListener` reagiert auf Aktionen, die mit der Maus getätigt werden.
    Beispiel: Der Benutzer bewegt die Maus in ein Fenster.
  - Der `WindowListener` reagiert auf Änderungen am Fensterstatus.
    Beispiel: Der Benutzer minimiert das Fenster.

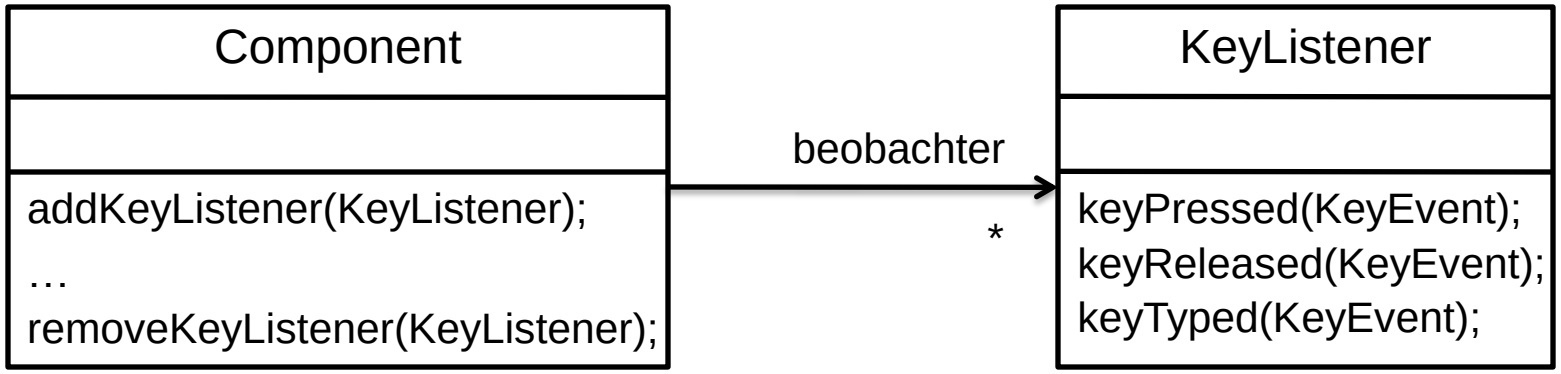
Beispiel: Schaltfläche mit ActionListener

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

public class KlickMich extends JFrame {

	public KlickMich() {
		super(„Swing: Ereignisse");

		JButton schaltflaeche = newJButton(„Klick mich!");

		schaltflaeche.addActionListener(new ActionListener() {
		   public void actionPerformed(ActionEvent e) {
				// Wird ausgelöst, wenn die Schaltfläche betätigt wird
				// z.B. durch einen Linksklick oder einen Druck auf [Return]
				// Hier den Code für die durchzuführende Aktion einfügen!
		   }
		});

		this.add(schaltflaeche);
		this.setDefaultCloseOperation(EXIT_ON_CLOSE);
		this.pack();
	}

	public static void main(String[] args) {
		KlickMich fenster = new KlickMich();
		fenster.setVisible(true);
	}
}
```

Die Schaltfläche soll auf die Aktivierung durch den Benutzer reagieren.

Ereignisse in Java/AWT/Swing: Das Beobachter-Entwurfsmuster

- In Java wird das Entwurfsmuster „Beobachter“ in leicht abgewandelter Form eingesetzt.
- **Beispiel:**
  `java.awt.Component` und `KeyListener`



- Wenn ein Ereignis aus der Warteschlange entnommen wird, werden an der zugehörigen Komponente für **alle** eingetragenen Beobachter die entsprechenden Methoden aufgerufen, wobei das Ereignis übergeben wird.

Implementierung von Beobachtern

- Ein Beobachter kann in Java auf unterschiedliche Weise implementiert werden.
- Eine eigene Beobachter-Klasse (auch als innere Klasse)
  - `class MeinBeobachter implements MouseListener {//...}`
- Verwenden von anonymen Klassen
  - ```
    addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {//...}
    });
    ```
- Selbstbeobachtendes Steuerelement
  - ```
    class MeineSchaltflaeche extends JButton
    implements MouseListener {//...}
    ```
- Verwenden von Adapter-Klassen
  - `class MeinBeobachter extends MouseAdapter {//...}`

Eigene Beobachter-Klasse

```
public class MeinBeobachter implements MouseListener {

    @Override
    public void mouseClicked(MouseEvent arg0) {
        System.exit(0);
    }

    @Override
    public void mouseEntered(MouseEvent e) {  }

    @Override
    public void mouseExited(MouseEvent e) {  }

    @Override
    public void mousePressed(MouseEvent e) {  }

    @Override
    public void mouseReleased(MouseEvent e) {  }
}
```

```
JButton b = new JButton("Beenden");
b.addMouseListener(new MeinBeobachter());
getContentPane().add(b);
```

Selbstbeobachtendes Steuerelement

```
public class BeendenKnopf extends JButton implements MouseListener {

    public BeendenKnopf() {
        super("Beenden");
        addMouseListener(this);
    }

    @Override
    public void mouseClicked(MouseEvent e) {  }

    @Override
    public void mouseEntered(MouseEvent e) {  }

    @Override
    public void mouseExited(MouseEvent e) {  }

    @Override
    public void mousePressed(MouseEvent e) {  }

    @Override
    public void mouseReleased(MouseEvent e) {  }
}
```

Knopf trägt sich bei sich selbst als Beobachter ein.

```
BeendenKnopf b = new BeendenKnopf();

getContentPane().add(b);
```

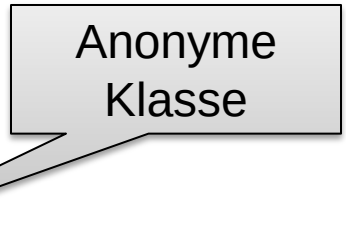
Adapterklasse

```
public class MeinBeobachter extends MouseAdapter {

    @Override
    public void mouseClicked(MouseEvent e) {
        System.exit(0);
    }
}
```

```
JButton b = new JButton("Beenden");
b.addMouseListener(new MeinBeobachter());
getContentPane().add(b);
```

Anonyme Klassen

```
public class MeinFrame extends JFrame {
    public MeinFrame() {
        JButton b = new JButton("Beenden");
        b.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent arg0)
                System.exit(0);
            }
        });
        //...
    }
}
```



Weitere Informationen zu Ereignissen und Listenern

- Weitergehende Informationen zu Ereignissen und deren Behandlung finden Sie in der Java-Dokumentation sowie im Swing-Tutorial:

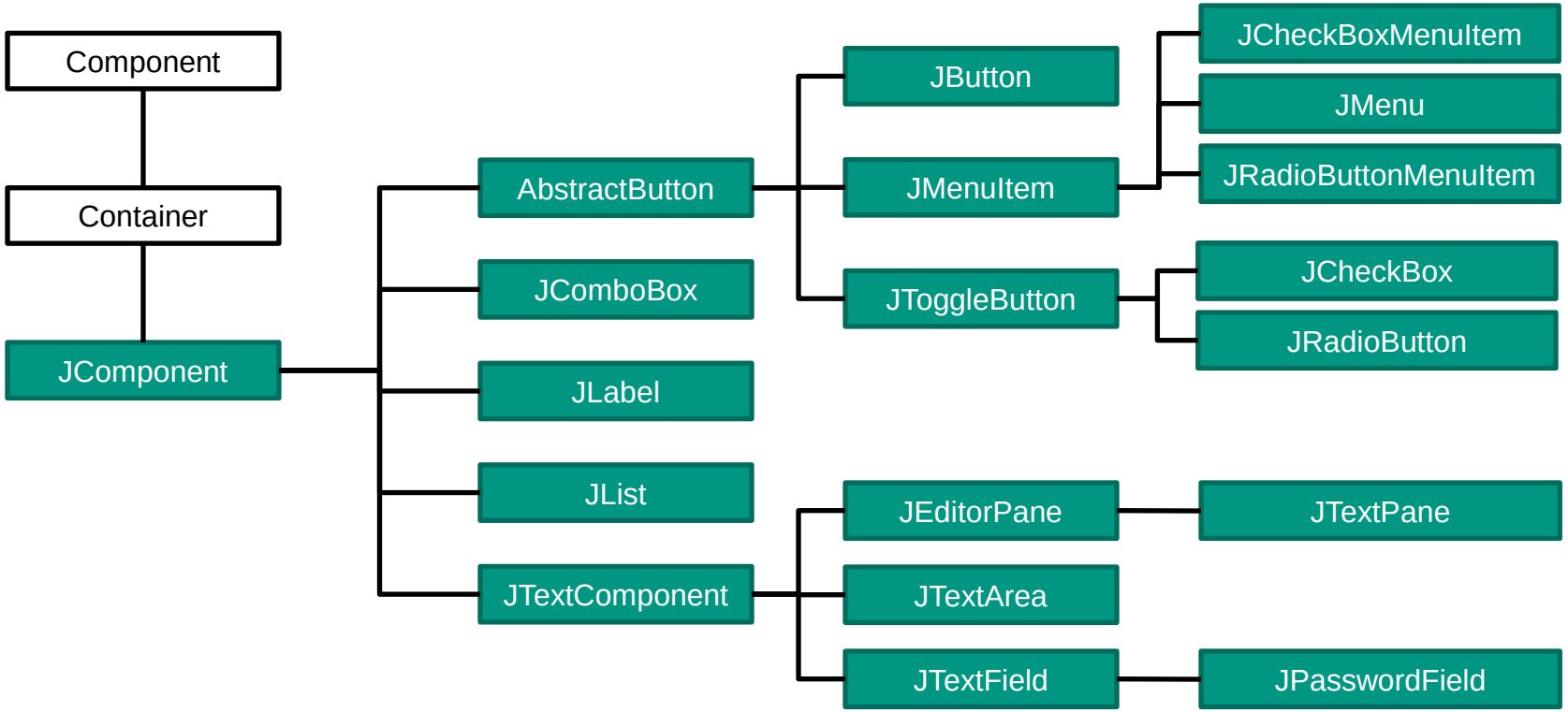
- http://java.sun.com/javase/6/docs/api/javax/swing/event/package-summary.html
- http://java.sun.com/javase/6/docs/api/java/awt/event/package-summary.html
- http://java.sun.com/docs/books/tutorial/uiswing/events/index.html

Komponentenübersicht

- Auf den folgenden Folien werden die wichtigsten Steuerelemente kurz vorgestellt.

- **Anmerkung:**
  Diese Übersicht ist nicht vollständig.

Komponentenübersicht



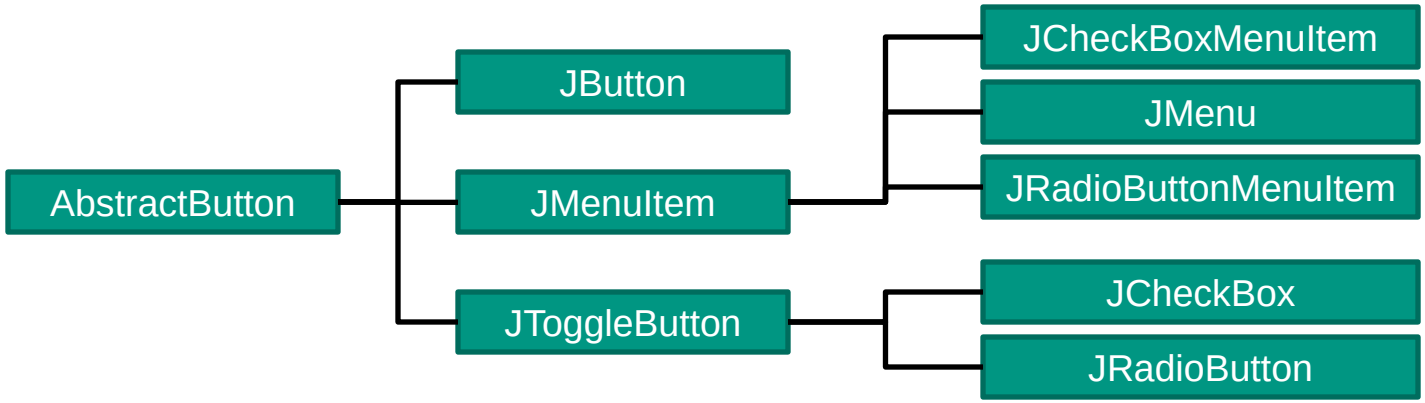
Pakete:

java.awt.*

java.swing.*

Schaltflächen

- Swing bietet verschiedene Arten von Schaltflächen:
  - Einfache Schaltflächen (`JButton`)
  - Schaltflächen für Menüs (`JMenuItem`)
    - Mit und ohne umschaltbarem Status
  - Schaltflächen mit umschaltbarem Status (`JToggleButton`)



Schaltflächen: JButton

- Objekte der Klasse `JButton` stellen einfache Schalflächen dar.
- Wenn ein Steuerelement dieser Klasse aktiviert wird (z.B. durch einen Klick mit der linken Maustaste), löst die Schaltfläche ein Ereignis (in diesem Fall ein `ActionEvent`) aus.

Menueintrag: JMenuItem (1)

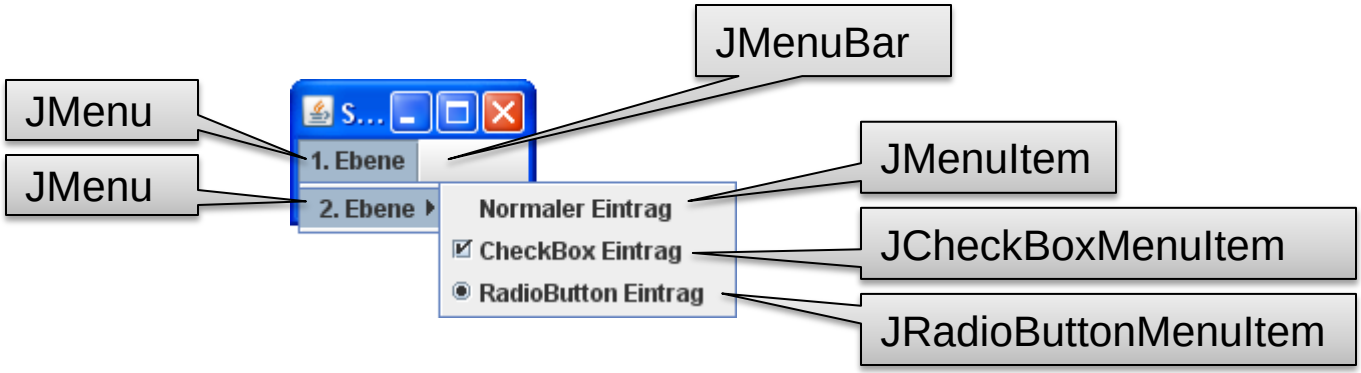
- Steuerelemente dieser Klasse, bzw. davon abgeleiteter Klassen, stellen einen Eintrag in einem Menü (z.B. `JMenuBar`) dar.

- Von dieser Klasse abgeleitet sind
  - `JCheckBoxMenuItem`
  - `JMenu`
    - Ein `JMenu` ist eine Schaltfläche, welche bei einem Klick ein `JPopupMenu` anzeigt
  - `JRadioButtonMenuItem`

- Objekte der Klasse `JCheckBoxMenuItem` und `JRadioButtonMenuItem` können die zwei Zustände „Gewählt“ oder „Nicht gewählt“ annehmen.

- Objekte der Klasse `JRadioButtonMenuItem` werden eingesetzt, wenn sich die gebotenen Auswahlmöglichkeiten gegenseitig ausschließen.

Menüeintrag: JMenuItem (2)



Umschalter: JToggleButton (1)

- Steuerelemente dieses Typs können zwischen mehreren Zuständen umgeschaltet werden.
- Von dieser Klasse erben die Steuerelemente
  - **Auswahlfeld:** `JCheckBox`
  - **Tastknopf:** `JRadioButton`
- Objekte der Klasse `JCheckBox` und `JRadioButton` können die zwei Zustände „Gewählt“ oder „Nicht gewählt“ annehmen.
- Objekte der Klasse `JRadioButton` werden eingesetzt, wenn sich die gebotenen Auswahlmöglichkeiten gegenseitig ausschließen.

Umschalter: JToggleButton (2)

JCheckBox

JRadioButton


```
Container c = getContentPane();
ButtonGroup bg = new ButtonGroup();

JRadioButton rb1 =
new JRadioButton("Ausgewählt", true);
JRadioButton rb2 =
new JRadioButton("Nicht ausgewählt", false);

bg.add(rb1); bg.add(rb2);
c.add(rb1); c.add(rb2);
```

Hinweis:

`JRadioButton`-Steuerelemente müssen nicht nur einem Container (z.B. der `ContentPane`) hinzugefügt werden, sondern auch einer `ButtonGroup`, welche den wechselseitigen Ausschluss gewährleistet.

Beschriftungen: JLabel

- `JLabels` können zur Darstellung eines Bildes, eines kurzen Textes oder zur Darstellung von Text und Bild verwendet werden.
- Es kann angegeben werden, wie Text und Bild im `JLabel` ausgerichtet werden.
- `JLabels` können nicht auf Aktionen reagieren.
- Werden oft dafür eingesetzt um Zellen von Tabellen darzustellen.

Auswahlliste: JComboBox

- Eine `JComboBox` besteht aus einer Schaltfläche oder bearbeitbarem Feld und einer Drop-Down-Liste.
- In einer `JComboBox` kann nur ein Eintrag gleichzeitig ausgewählt sein.
- In `JComboBox` kann ein Eintrag durch Tastendruck ausgewählt werden.
- Der Entwickler kann festlegen, ob der Benutzer neben den Werten in der Auswahlliste auch eigene Werte eingeben darf.

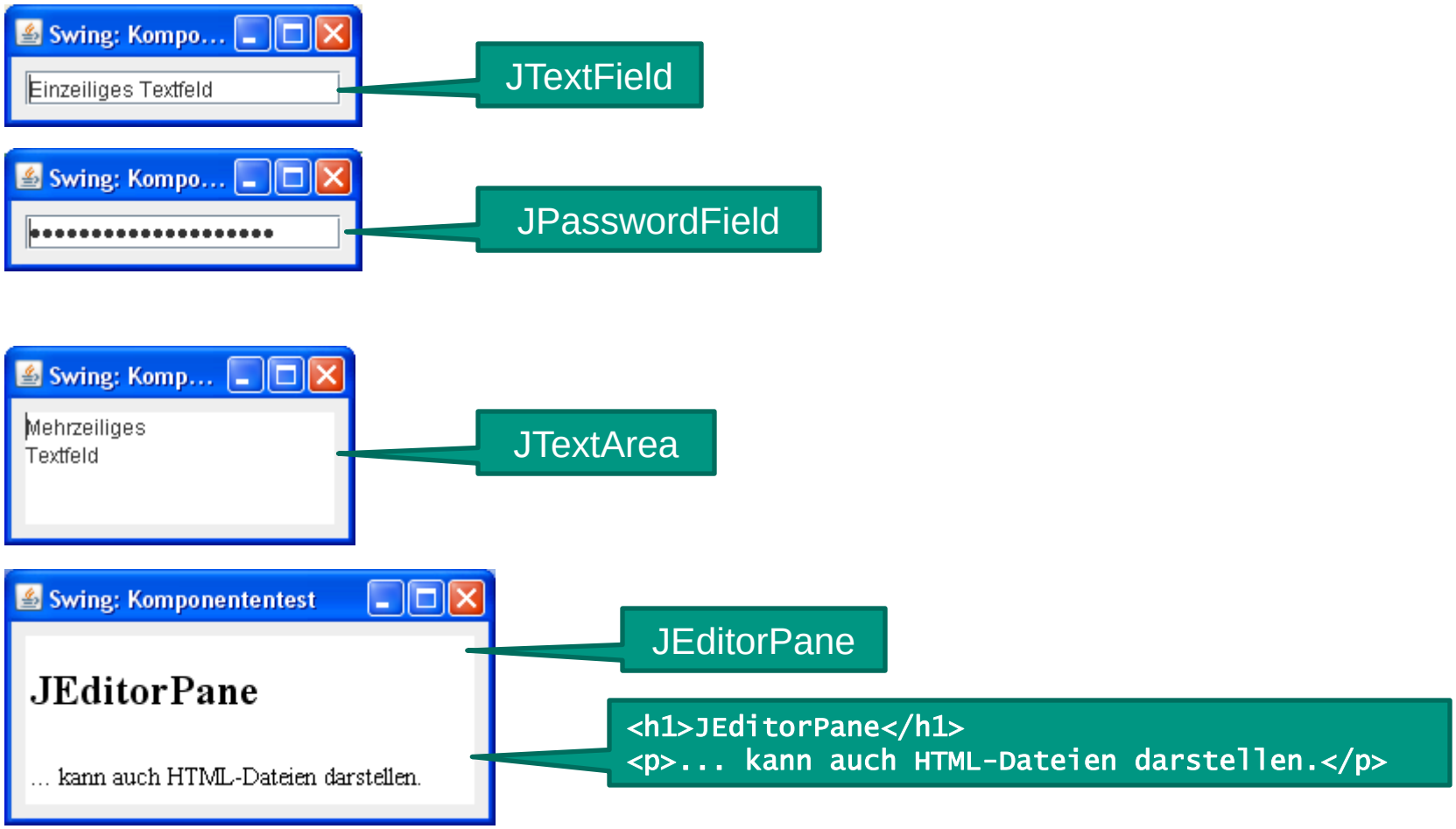
Auswahlliste: JList

- In Steuerelementen der Klasse `JList` kann der Benutzer aus einer gegebenen Menge von Einträgen eine bestimmte Teilmenge von Einträgen auswählen.
- Wie eine Auswahl von Einträge aussehen darf, kann der Entwickler festlegen.
- Einträge in einer `JList` können nicht bearbeitet werden
- Eine `JList` bietet keine Methoden zum Hinzufügen, Einfügen oder Löschen von einzelnen Einträgen an. Für alle auf einmal geht das entweder über die Methode `JList.setListData()` oder über ein eigenes Listenmodell.

Textfelder: JTextComponent (1)

- Steuerelemente des Typs `JTextComponent`, bzw. davon abgeleiteten Steuerelementen, ermöglichen die Eingabe von Texten durch den Benutzer.
- Von dieser Klasse erben die Steuerelemente
  - `JTextArea`
    - Erlaubt mehrzeilige Eingaben mit einer Schriftart
  - `JTextField`
    - Erlaubt eine Textzeile mit einer Schriftart
    - Das Steuerelement `JPasswordField` dient zur Eingabe eines einzeiligen Kennwortes.
  - `JEditorPane`
    - Kann verschiedene Dokumenttypen anzeigen (z.B. RTF)
    - Das davon abgeleitete Steuerelement `JTextPane` kann zusätzlich Bilder anzeigen und weitere Steuerelemente einbinden.

Textfelder: JTextComponent (2)



Fenster, Frames und Dialoge

- Fenster, Frames und Dialoge sind in Swing schwergewichtige Komponenten, welche die AWT-Klassen `Window`, `Frame` und `Dialog` erweitern.

Eigenschaft	Window	Frame	Dialog
Modal*	Nein	Nein	Optional
Größe anpassbar	Nein	Optional	Optional
Titelleiste	Nein	Ja	Ja
Titel	Nein	Ja	Ja
Menüleiste	Nein	Optional	Nein

* Ein modales Fenster erzwingt, dass die Interaktionen nur mit diesem Fenster geschieht, bis es geschlossen wird.

Fenster, Frames und Dialoge

- In Swing können Sie Fenster erstellen, indem Sie von einer der Klassen `JWindow`, `JFrame` oder `JDialog` erben.
- Je nach Einsatzzweck müssen Sie zuvor die geeignete Klasse wählen, da nicht jede der drei Klassen die gleichen Möglichkeiten bietet.
- Benötigen sie bspw. ein modales Fenster, müssen Sie zur Klasse `JDialog` greifen.