Ludwig
Maximilians
Universität
München

Java-AWT

Die

Abstract Window Toolkit

Klassenbibliothek

Ludwig Maximilians Universität München

Inhalt

- Struktur und
- wichtige Elemente
  - Events
  - Listener
  - Adapter
  - Layout-Manager

Ludwig Maximilians Universität München

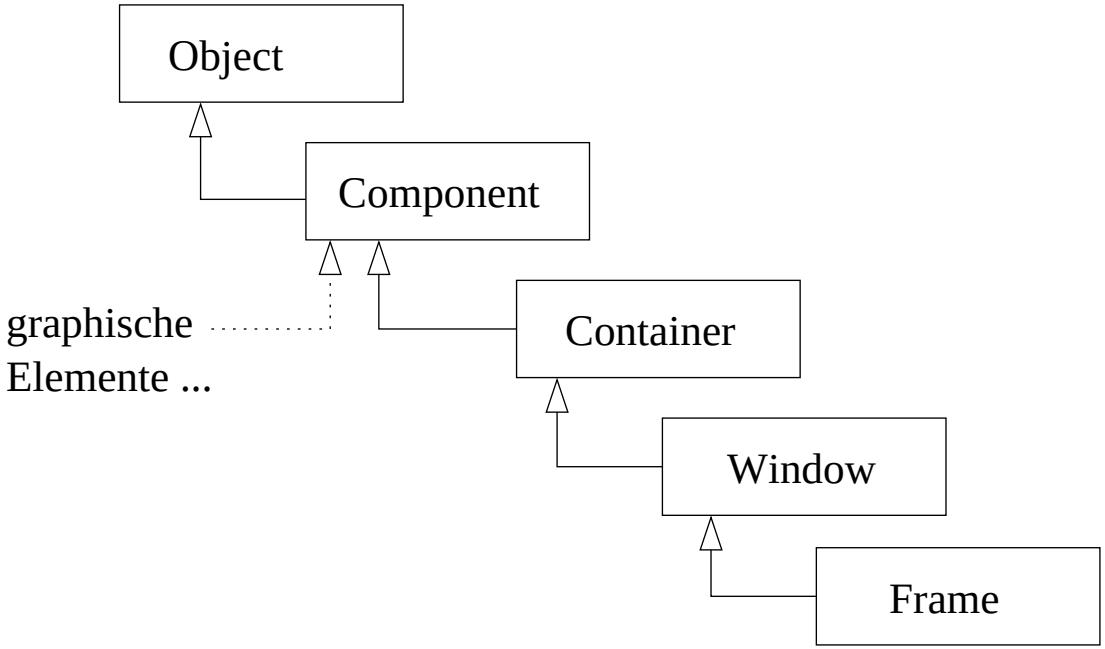
Allgemeines

- Delegation der Aufgaben an das OS (AWT)
- Elemente sind der "kleinste gemeinsamer Nenner"
- Oberfläche nur bedingt portierbar

Ludwig Maximilians Universität München

Klassenhierarchie

- Oberklasse `Component`
- `Container` enthalten graphische Elemente
- Graphische Elemente (Buttons, Labels, Textfelder etc.)


Object
Component
graphische
Elemente ...
Container
Window
Frame

Ludwig Maximilians Universität München

LMU

Aufgaben graphischer Elemente

- Eingabe von Daten
- Steuerung durch den Benutzer
- Darstellung von Daten

Ludwig Maximilians Universität München

Events

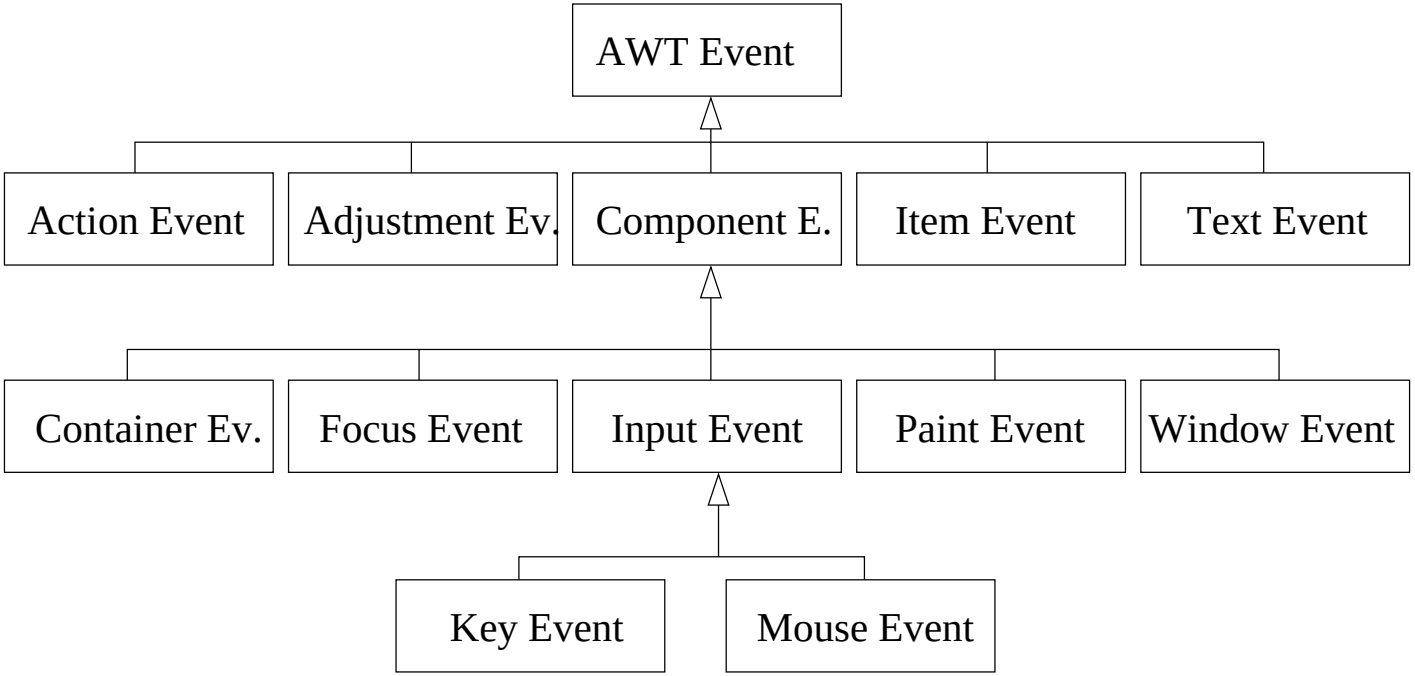
Ludwig Maximilians Universität München

Benutzer-Interaktion

- Ursache: Tastatur und Maus
- Ziel: verschiedene graphische Elemente
- Wirkung: AWT-Ereignis (`AWTEvent`)
  dh. eine AWT-Komponente ist eine "Event"-Quelle
- Jede Komponente hat spezifische Events

Ludwig Maximilians Universität München

Eventhierarchie



Ludwig Maximilians Universität München

EventListener

- Implementierung eines `EventListener`
  - Bsp: `ActionListener`
  - Bsp: `MouseListener`
  - Bsp: `MouseMotionListener`
- `EventListener` an die Eventquelle anschliessen mit `addListener ()`
- ggf. Nutzung von Adaptern

Ludwig Maximilians Universität München

Adapter

- Adapter sind abstrakte Klassen mit leeren Methoden-Rümpfen
  - Bsp: `MouseAdapter`
  - Bsp: `WindowAdapter`
- Adapter sparen Tipp-Arbeit
- mindestens eine Methode muss überschrieben werden
- dann eigenen Adapter instantiieren und anschliessen

Ludwig Maximilians Universität München

LMU

Event-Queue

- Behandlung der AWT-Events in einer Queue
- Nebenläufigkeit/Synchronisationsprobleme beachten
- Event-Queue evtl. in eigenem Thread implementieren

Ludwig
Maximilians
Universität
München

Layoutmanager

Darstellung mehrerer Komponenten

in einem Container

Ludwig Maximilians Universität München

Problemstellung

- Container beinhalten mehrere Komponenten
- Komponenten können meist keine feste Anordnung haben (Grund: `resize`)
- Lösung des Layoutproblems in Java:
  - Layoutmanager
  - Angaben über den Aufbau abstrakt
  - Verschiedene Layoutmanager mit verschiedenen Schemen/Policies

Ludwig Maximilians Universität München

Beispiele

- `FlowLayout`
- `BorderLayout`
- `GridBagLayout`

Ludwig Maximilians Universität München

FlowLayout

- Einfachstes Layout (meist voreingestellt)
- Anordnung der Komponenten waagrecht von links nach rechts

  `setLayout(new FlowLayout(FlowLayout.LEFT))`
- Änderungsmöglichkeiten:
  - Links-/Rechtsbündig (`LEFT/RIGHT`)
  - Zentriert (`CENTER`)
  - Abstand (`hgap/vgap`)

Ludwig Maximilians Universität München

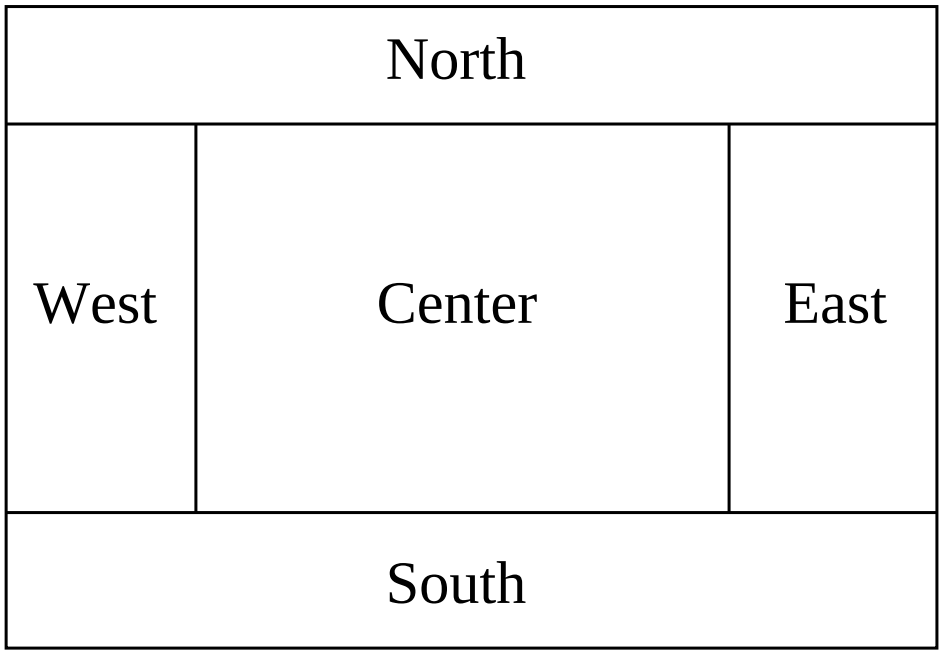
BorderLayout

- Abstraktes "Bereichsmodell"
- Zuordnung von Komponenten zu ausgewählten Bereichen

  `add(myButton,"South")`
- Parameter:
  - "North", "South", "East", "West" und "Center" (beim Hinzufügen einer Komponente)
  - hgap (im Konstruktor des Managers)
  - vgap

Ludwig Maximilians Universität München

BorderLayout - Modell



Ludwig Maximilians Universität München

LMU

Verschachtelte Layouts

- Container können auch Container beinhalten
- Jeder Container hat ein eigenes Layout
- Container werden durch den LayoutManager des Containers angeordnet, in dem sie enthalten sind.

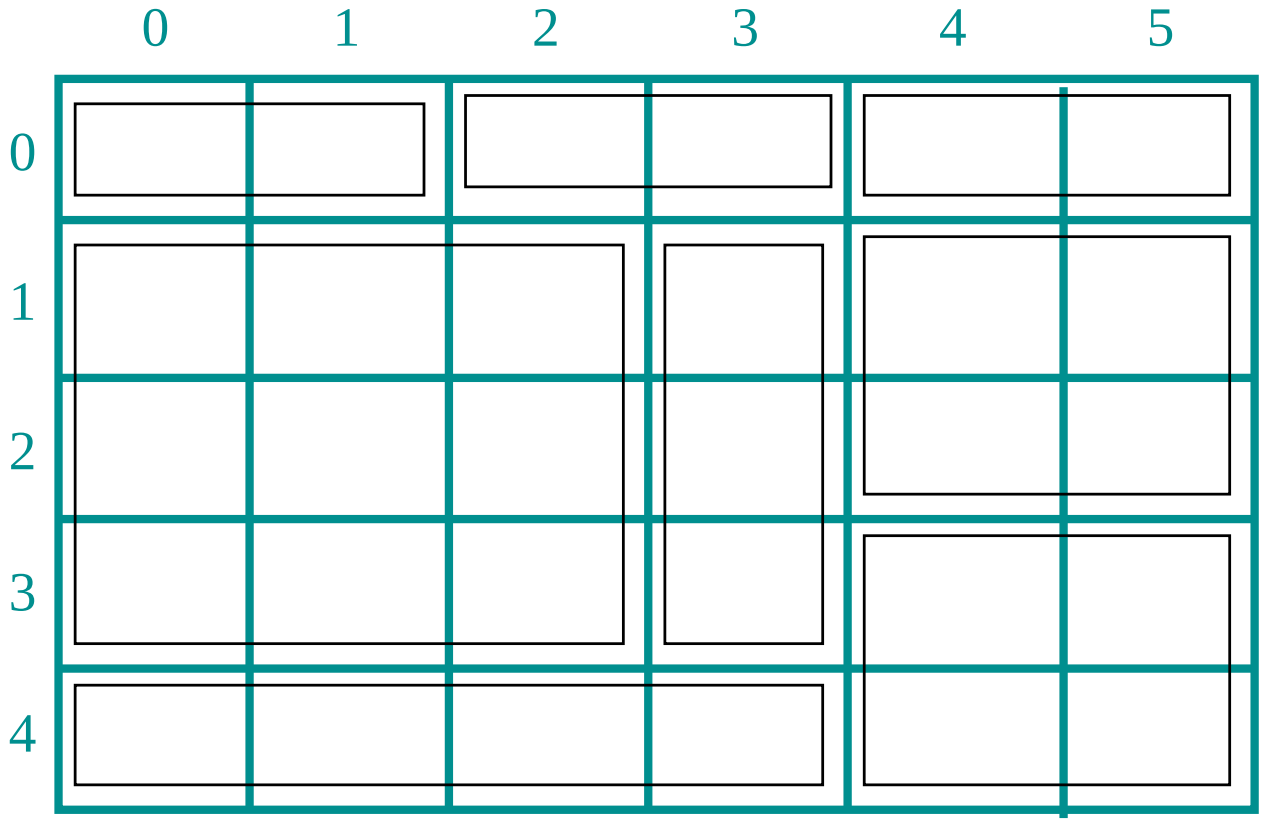
Ludwig Maximilians Universität München

GridBagLayout

- "Befestigung" von Komponenten an einem Raster
- Mehrere Felder eines Rasters zu einer Zelle verschmelzbar
- Jeder Zelle kann man eigene `GridBagConstraints` zuordnen

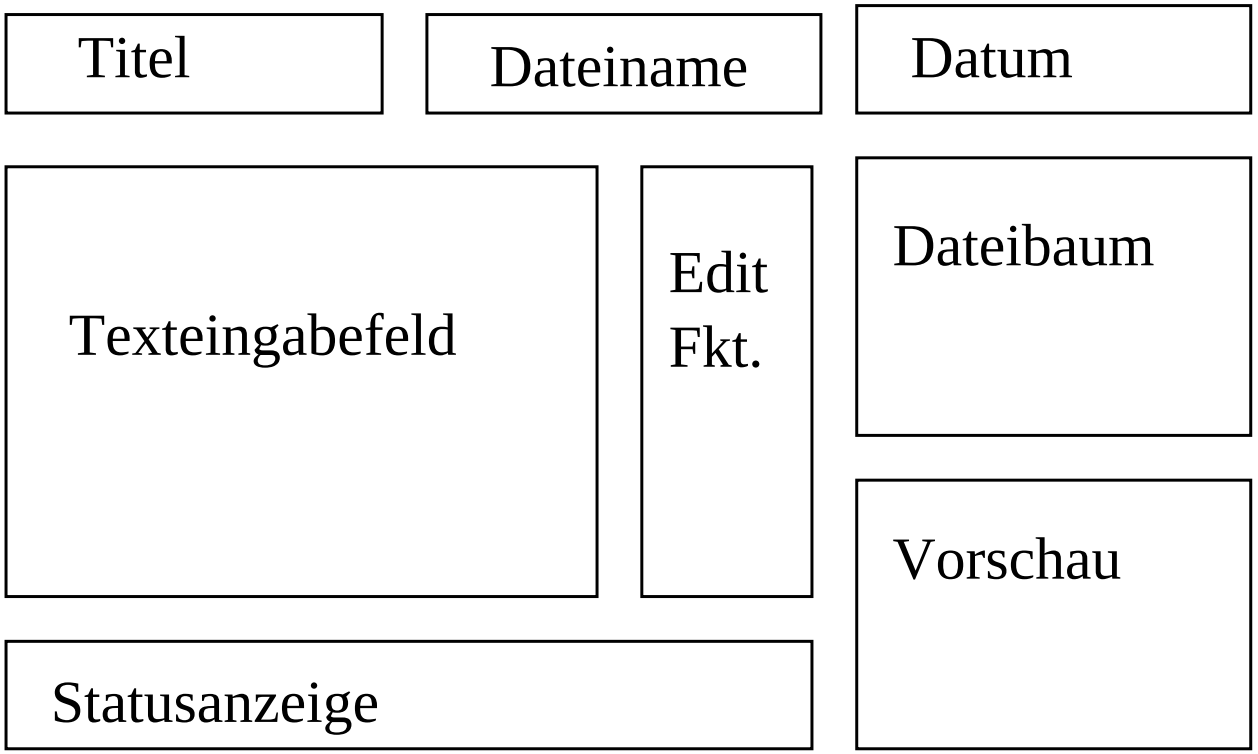
Ludwig Maximilians Universität München

GridBagLayout - Modell



Ludwig Maximilians Universität München

GridBagLayout - Anwendung



Constraints

Ludwig Maximilians Universität München

- Position durch `gridx/gridy`
- Größe durch `gridwidth/gridheight`
- Verhalten bei Änderung der Fenstergröße `gridweightx/gridweighty`
- Füllverhalten durch `fill`
  - NONE, HORIZONTAL, VERTICAL oder BOTH
- Ausrichtung durch `anchor`
  - CENTER, NORTH, EAST etc.
- Verhalten bei Größenänderung durch `weightx,weighty` (0 = unveränderliche Größe)

Ludwig Maximilians Universität München

Layout Entwicklung

1. Skizze des Komponentenlayouts
2. Festlegung der Rastergröße (kleinste Komponente = 1 Feld)
3. Spaltennummerierung
4. Festlegung der `x,y,width` und `height` Werte jeder Komponente
5. Festlegung des Füllungsverhaltens `fill/anchor`
6. Festlegung des Verhaltens bei Größenänderung
7. Erzeugung der `GridBagContraint`-Objekte für jede Komponente
8. Aufbau des Containers nach sorgfältiger Prüfung

Ludwig
Maximilians
Universität
München

SWING

Lightweight-Components,

bessere graphische Darstellung und

beliebige neue Komponenten...