Barry Linnert
Institut für Informatik

Freie Universität Berlin

Programmieren

Barry Linnert

Sommersemester 2020

Freie Universität Berlin

Gliederung der heutigen Vorlesung

- Kurze Wiederholung
- Grafische Ein-/Ausgabe mit AWT und Swing
- Threads
- Zusammenfassung

Freie Universität Berlin

Grafische Ein-/Ausgabe

WIEDERHOLUNG

Terminologie

- Ein **abstrakter Datentyp (ADT)** besteht aus einem Wertebereich (d.h. einer Menge von Objekten) und darauf definierten Operationen.

- Die Menge der Operationen bezeichnet man auch als Schnitt-stelle des Datentyps.

- Eine **Datenstruktur** ist eine Realisierung bzw. Implementierung eines ADT.

Verbergen der Implementierung eines abstrakten Datentyps

Meyer, B., "Object-Oriented Software Construction", 2nd ed., Prentice Hall, 1997

Freie Universität Berlin

Grafische Ein-/Ausgabe

JAVA AWT UND SWING

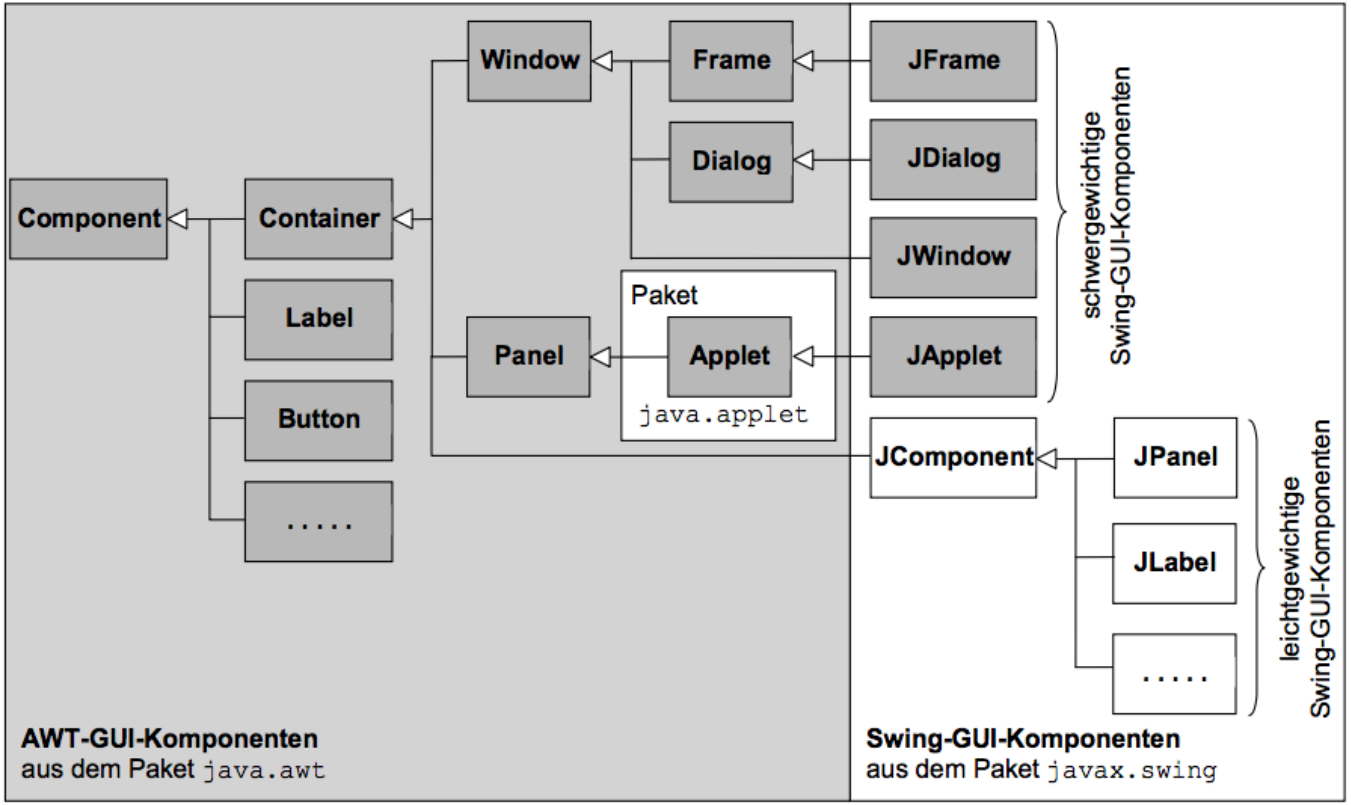
Schwergewichtige vs. Leichtgewichtige Komponenten

- Schwergewichtige Komponenten (Abstract Window Toolkit/AWT):

  Komponente hat plattformspezifische Objekte, die seine Funktionalität erbringen
  - nur kleinster gemeinsamer Nenner umsetzbar
  - Konsistenzprobleme
  - Performanzprobleme

- Leichtgewichtige Komponenten (Swing):

  Vollständige Implementierung der Komponenten in Java (mit konfigurierbarem "Look and Feel”)
  - Plattformunabhängig
  - Größere Funktionalität
  - Verbesserte Leistungsfähigkeit


Component
Container
Label
Button
.....
Window
Frame
Dialog
Panel
Paket
Applet
java.applet
JFrame
JDialog
JWindow
JApplet
schwergewichtige Swing-GUI-Komponenten
JComponent
JPanel
JLabel
.....
leichtgewichtige Swing-GUI-Komponenten
AWT-GUI-Komponenten
aus dem Paket java.awt
Swing-GUI-Komponenten
aus dem Paket javax.swing
Legende:
leichtgewichtige GUI-Komponenten
schwergewichtige GUI-Komponenten

AWT-Ausgabefenster

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

public class TestFrame extends Frame
{
  public TestFrame ()  {
    setTitle(“Mein erstes Fenster");                    // Fenstertitel setzen
    setSize(400,100);                                   // Fenstergröße einstellen
    addWindowListener(new TestWindowListener());        // EventListener für das Fenster hinzufügen
                                                        // (notwendig, damit das Fenster geschlossen
                                                        //  werden kann)
    setVisible(true);                                   // Fenster (inkl. Inhalt) sichtbar machen
  }
```

AWT-Ausgabefenster

```
  class TestWindowListener extends WindowAdapter {
    public void windowClosing(WindowEvent e)  {
      e.getWindow().dispose();                    // Fenster schließen
      System.exit(0);                             // VM beenden
    }
  }

  public static void main (String args[])  {
    new TestFrame ();
  }
}
```

AWT-Button

```
import java.awt.*;
import java.awt.event.*;
public class TestFrame extends Frame {
  Button button = new Button("Schaltfläche");
  public TestFrame ()  {
    setTitle("Schaltflächenbeispiel");
    addWindowListener(new TestWindowListener());
    button.setForeground(Color.RED);                    // Vordergrundfarbe auf "rot" setzen
    button.setBackground(Color.WHITE);                  // Hintergrundfarbe auf "weiß" setzen
    button.addActionListener(new TestActionListener()); // EventListener für Schaltfläche hinzufügen
    add(button);                                        // Schaltfläche zum Fenster hinzufügen
    pack();                                             // Fenstergröße auf die benötigte Größe festlegen
    setVisible(true);
  }
```

AWT-Button

```
class TestWindowListener extends WindowAdapter{
    public void windowClosing(WindowEvent e)  {
      e.getWindow().dispose();
      System.exit(0);
    }
  }

  class TestActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e)  {
      System.out.println("Schaltfläche wurde gedrückt");
    }
  }
```

AWT-Button

```
  public static void main (String args[])  {
    new TestFrame ();
  }
}
```

Schwer- und leichtgewichtige Swing-GUI-Komponenten

- Schwergewichtige AWT-GUI-Komponenten Frame, Dialog, Applet und Window werden durch die „schwergewichtigen“ Swing-GUI-Komponenten JFrame, JDialog, JWindow und JApplet erweitert
  - In Aussehen und Verhalten **abhängig** vom Betriebssystem

- Alle von der Klasse JComponent abgeleiteten Klassen sind sog. „leichtgewichtige“ Swing-GUI-Komponenten
  - In Aussehen und Verhalten **unabhängig** vom Betriebssystem

GUI-Container (1)

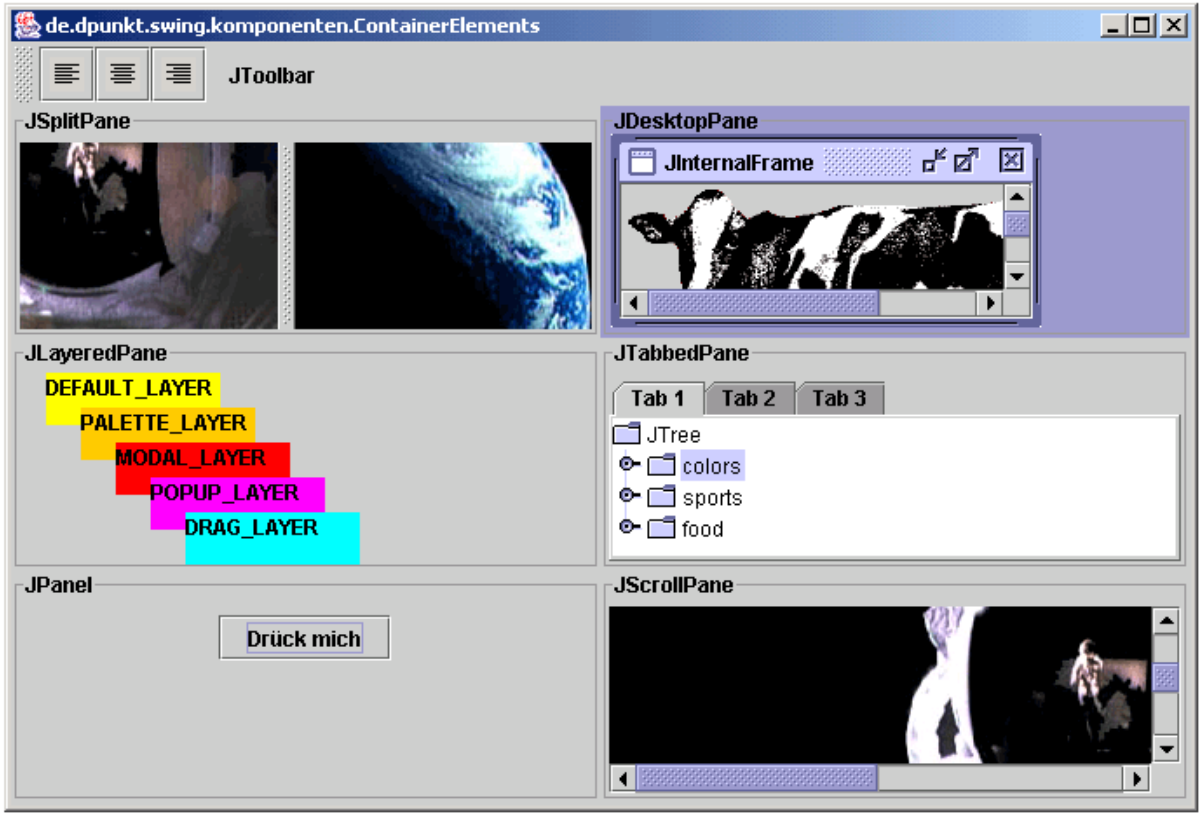
- Ist eine spezielle GUI-Komponente, die andere GUI- Komponenten (entsprechend angeordnet) aufnehmen kann
- Enthält Elemente (analog zu einer *Collection*): Referenzen auf Objekte vom Typ Component
- Nutzt die Eigenschaft der Klasse Container - durch Aufruf der Methode add() können einem GUI-Container beliebig viele GUI-Komponenten hinzugefügt werden

- **Wie?**
  - Durch bekannte Konzepte wie Vererbung und Aggregation
  - Aber auch Unbekanntes wie das Entwurfsmuster Kompositum (siehe Literatur)

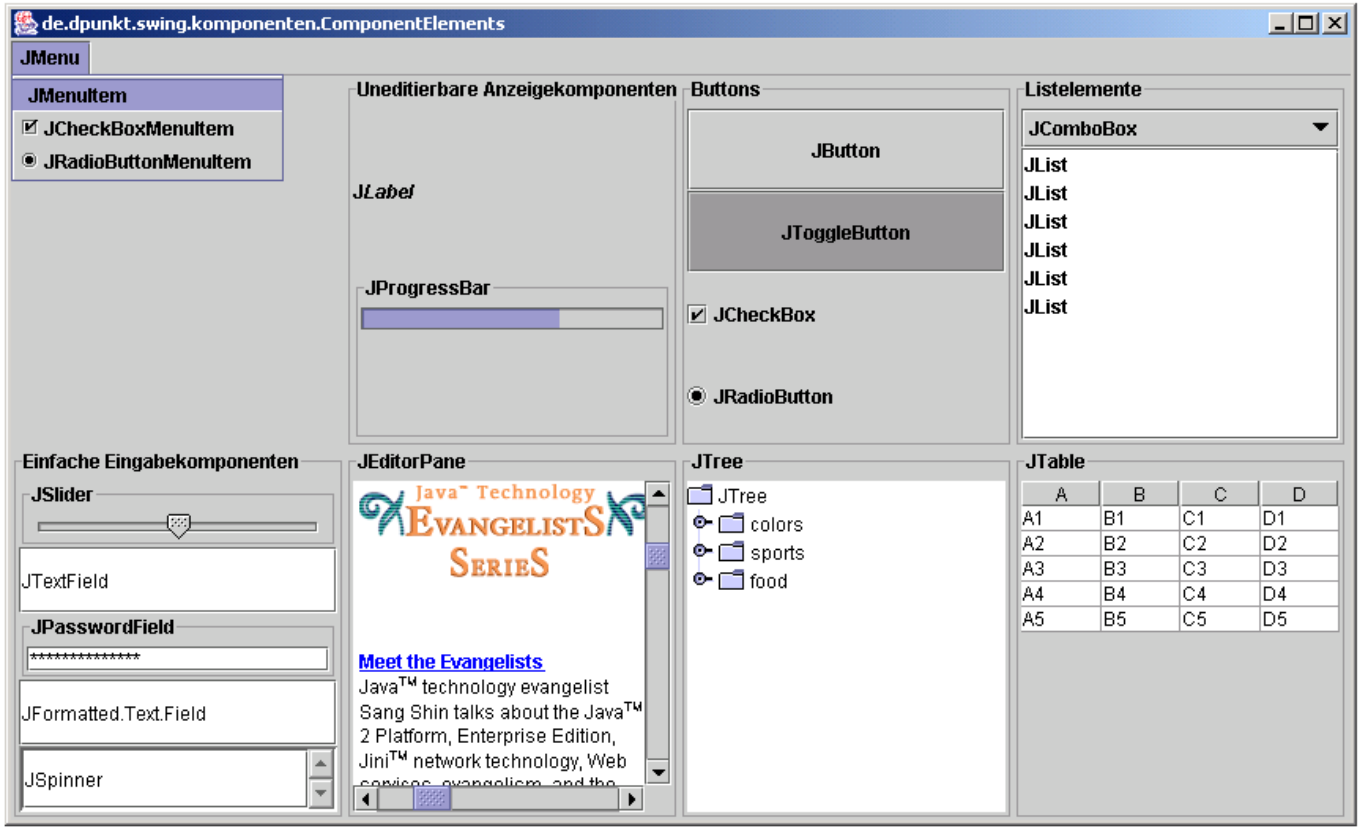
GUI-Container (2)

- Klassen JFrame, JDialog und JWindow können zur Darstellung eines Bildschirmfensters verwendet werden
- Klasse JApplet: für das Erzeugen von Browser-Applets
- Der schwergewichtige GUI-Container wird als Hauptfenster oder Top-Level-Container (= oberster Container einer GUI, Elternfenster) bezeichnet
  - Umhüllt die grafische Bedienoberfläche einer Anwendung
  - Stellt eine visuelle Grenze zu der grafischen Oberfläche des Betriebssystems und den Fenstern anderer Anwendungen dar
  - Interagiert auch mit dem Window-Manager des Betriebssystems (dieser verwaltet alle Fenster einer grafischen Oberfläche des Betriebssystems)

Container-Klassen bei Swing



Komponenten-Klassen bei Swing



Swing-Button

```
import javax.swing.JFrame;        // Swing-GUI-Komponenten befinden sich im Paket javax.swing
import javax.swing.JButton;
import java.awt.FlowLayout;
public class View{
  public static void main (String[] args) {
    JFrame frame = new JFrame ("Hauptfenster");          // Hauptfenster erzeugen
    JButton button1 = new JButton ("Schaltfläche1");     // Schaltflaeche erzeugen
    frame.setLayout (new FlowLayout());
    frame.add (button1);                                 // Schaltflaeche dem Hauptfenster hinzufuegen
    frame.setSize (400, 100);
    frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
    frame.setVisible (true);
  }
}
```

Swing-Button

```
import javax.swing.JFrame; import javax.swing.JButton;
import java.awt.FlowLayout; import java.awt.event.*;
public class View {
  public static void main (String[] args) {
  JFrame frame = new JFrame ("Hauptfenster");              // Hauptfenster erzeugen
  JButton button1 = new JButton ("Gedrückt: 0");
  // Selbst geschriebener Controller für das Ereignis anmelden
  button1.addActionListener (new ButtonController());
  frame.setLayout (new FlowLayout());
  frame.add (button1);                                     // Schaltflaeche dem Hauptfenster hinzufuegen
  frame.setSize (400, 100);
  frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
  frame.setVisible (true);}
}
```

Swing-Button

```
// Selbst geschriebene Controller-Funktionalitaet: Moechte man Ereignisse abfangen, um darauf individuell zu
// reagieren, so muss man die zum Ereignis passende Listener-Schnittstelle implementieren
class ButtonController implements ActionListener {
  private int counter = 0;
  // In der Methode actionPerformed() muss nun die gewuenschte Reaktion auf Ereignis implementiert werden
  public void actionPerformed(ActionEvent action) {
    counter++;                                    // internen Zaehler erhoehen
    // Referenz auf die Schaltflaeche - durch Aufruf der Methode
    // getSource() zum Uebergabeparameter action – besorgen
    JButton refSource = (JButton) action.getSource();
    // neuen Text fuer Schaltflaeche setzen
    refSource.setText("Gedrückt: " + counter);
  }
}
```

Freie Universität Berlin

Nebenläufige Programmierung

THREADS

Threads

```
public class SimpleThread extends Thread {
    private int countDown = 5;
    private int threadNumber;
    private static int threadCount = 0;
    public SimpleThread() {
        threadNumber = ++threadCount;
        System.out.println("Making " + threadNumber);
    }
```

```
    public void run() {
        while(true) {
            System.out.println("Thread " + threadNumber + "(" + countDown + ")");
            if(--countDown == 0) return;
        }
    }
    public static void main(String[] args) {
        for(int i = 0; i < 5; i++)
            new SimpleThread().start();
        System.out.println("All Threads Started");
    }
}
```

Freie Universität Berlin

ENDE

PROJEKTBEGINN!