Universität Bremen

Praktische Informatik 1 Graphische Benutzungsoberflächen 2

Thomas Röfer

Cyber-Physical Systems
Deutsches Forschungszentrum für Künstliche Intelligenz

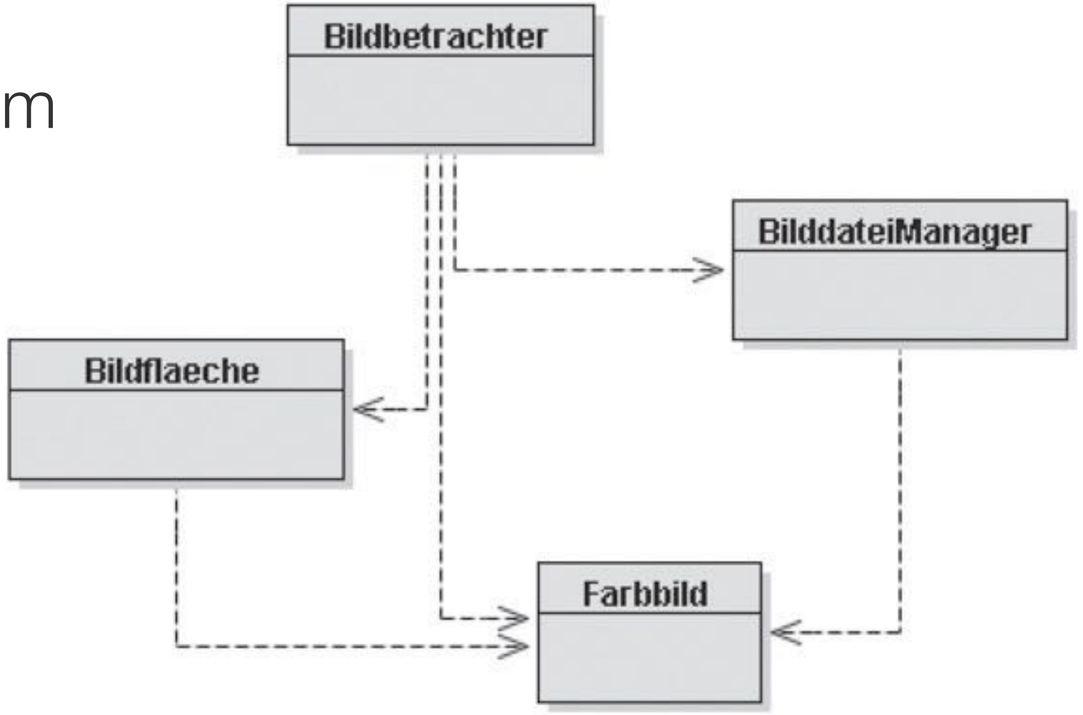
Multisensorische Interaktive Systeme
Fachbereich 3, Universität Bremen

Bild: ©Michael Ihle

Universität Bremen

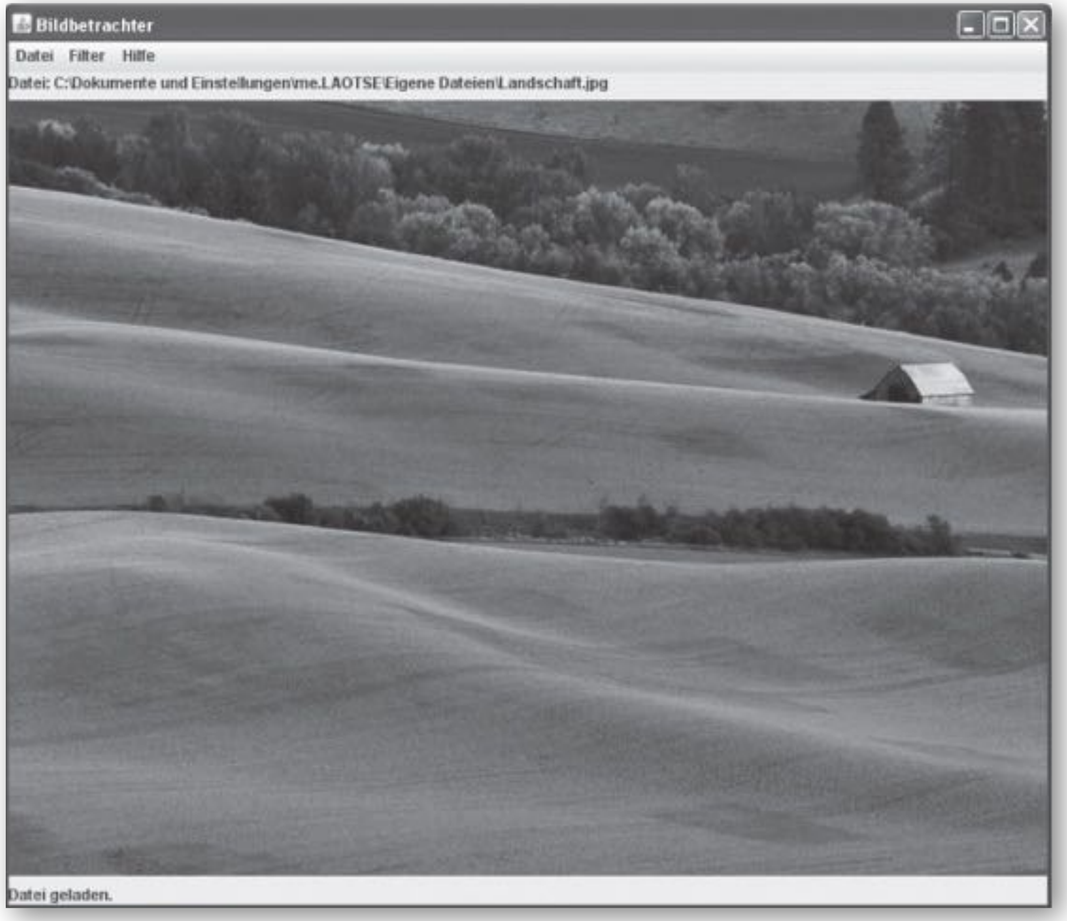
Struktur des Bildbetrachters

- **Bildbetrachter**: Initialisiert GUI-Struktur
- **BilddateiManager**: Klassenmethoden zum Laden und Speichern von Bilddateien
- **Bildflaeche**: Anzeige des Bildes in GUI
- **Farbbild**: Modelliert zweidimensionales Bild



Universität Bremen

Einfügen eines Bildes: Demo



Universität Bremen

JComponent

- Superklasse aller Swing-Komponenten
- Definiert viel Funktionalität
  - „Look & Feel“, Tastaturbedienung, Tooltips, Rahmen, gewünschte Ausrichtung ...
- Subklasse von (AWT-)**Container**, kann also andere Komponenten enthalten
  - **Layout-Manager** ordnet die Subkomponenten an

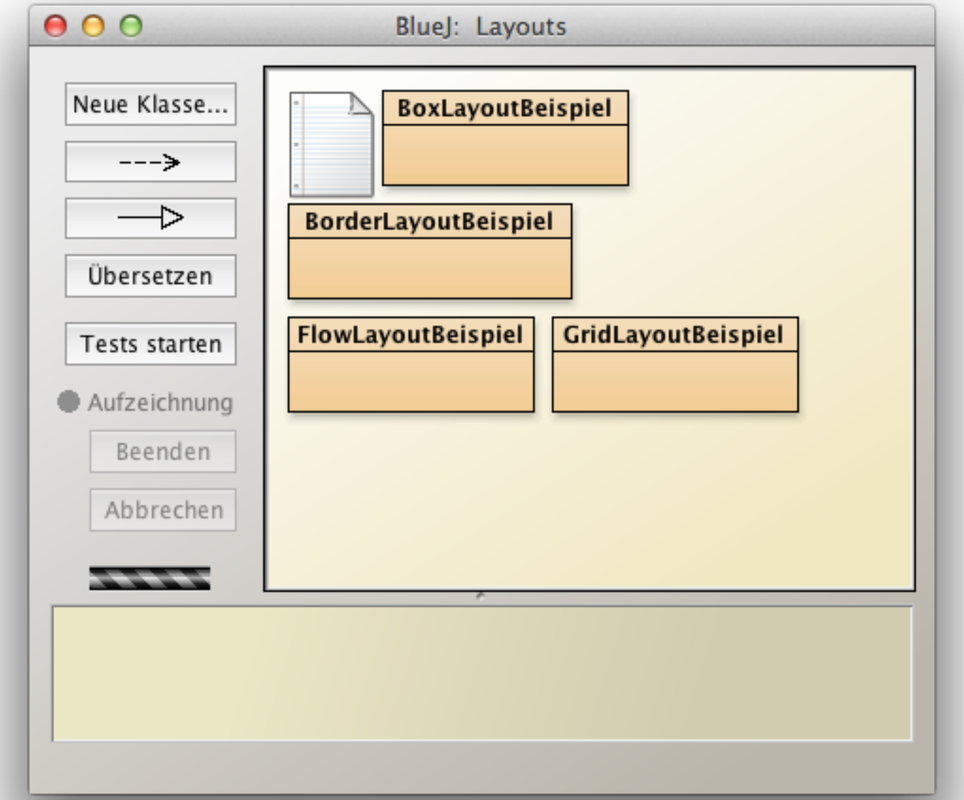
Universität Bremen

Grafik

- In Komponenten kann durch Überschreiben von Methoden gezeichnet werden
  - **paintComponent(Graphics)**: Inhalt zeichnen (Rand/Kinder bleiben)
  - **paint(Graphics)**: Alles zeichnen
  - Hintergrund sichtbar → Methode der Superklasse am Anfang aufrufen
- **Graphics**: Zeichenschnittstelle
  - Bietet Methoden zum Zeichnen
  - Speichert den aktuellen Zustand des Zeichenwerkzeugs (Farbe, Schriftart usw.)
- Gezeichnet wird, wenn Oberfläche dies fordert oder **repaint()** aufgerufen wurde

Universität Bremen

Layout-Manager: Demo



Layout-Manager: FlowLayout

- Kann beliebig viele Komponenten enthalten, die an Fenstergrenzen umgebrochen werden
- Unterschiedliche Ausrichtungen: **FlowLayout.LEFT**, **FlowLayout.CENTER**, **FlowLayout.RIGHT**
- Standard-LayoutManager für **JPanel**


```
contentPane.setLayout(new FlowLayout());
contentPane.add(new JButton("Erster"));
contentPane.add(new JButton("Zweiter"));
contentPane.add(new JButton(
        "Der dritte String ist lang"));
contentPane.add(new JButton("Vierter"));
contentPane.add(new JButton("Fünfter"));
```

Layout-Manager: BorderLayout

- Speichert Komponenten in fünf Bereichen: **NORTH**, **SOUTH**, **WEST**, **EAST**, **CENTER**
- Überschüssiger Raum wird mittlerem Bereich zugeordnet
- Standard-LayoutManager für alle Fenster, z.B. Rahmen und Dialoge

```
contentPane.setLayout(new BorderLayout());
contentPane.add(new JButton("Norden"), BorderLayout.NORTH);
contentPane.add(new JButton("Süden"), BorderLayout.SOUTH);
contentPane.add(new JButton("Mitte"), BorderLayout.CENTER);
contentPane.add(new JButton("Westen"), BorderLayout.WEST);
contentPane.add(new JButton("Osten"), BorderLayout.EAST);
```

Layout-Manager: GridLayout

- Ordnet Komponenten in einem Gitter an
- Alle Komponenten haben die gleiche Größe
- Breite/Höhe **0** heißt „beliebig viele“


```
contentPane.setLayout(new GridLayout(3, 2));
contentPane.add(new JButton("Erster"));
contentPane.add(new JButton("Zweiter"));
contentPane.add(new JButton("Der dritte String ist lang"));
contentPane.add(new JButton("Vierter"));
contentPane.add(new JButton("Fünfter"));
```

Universität Bremen

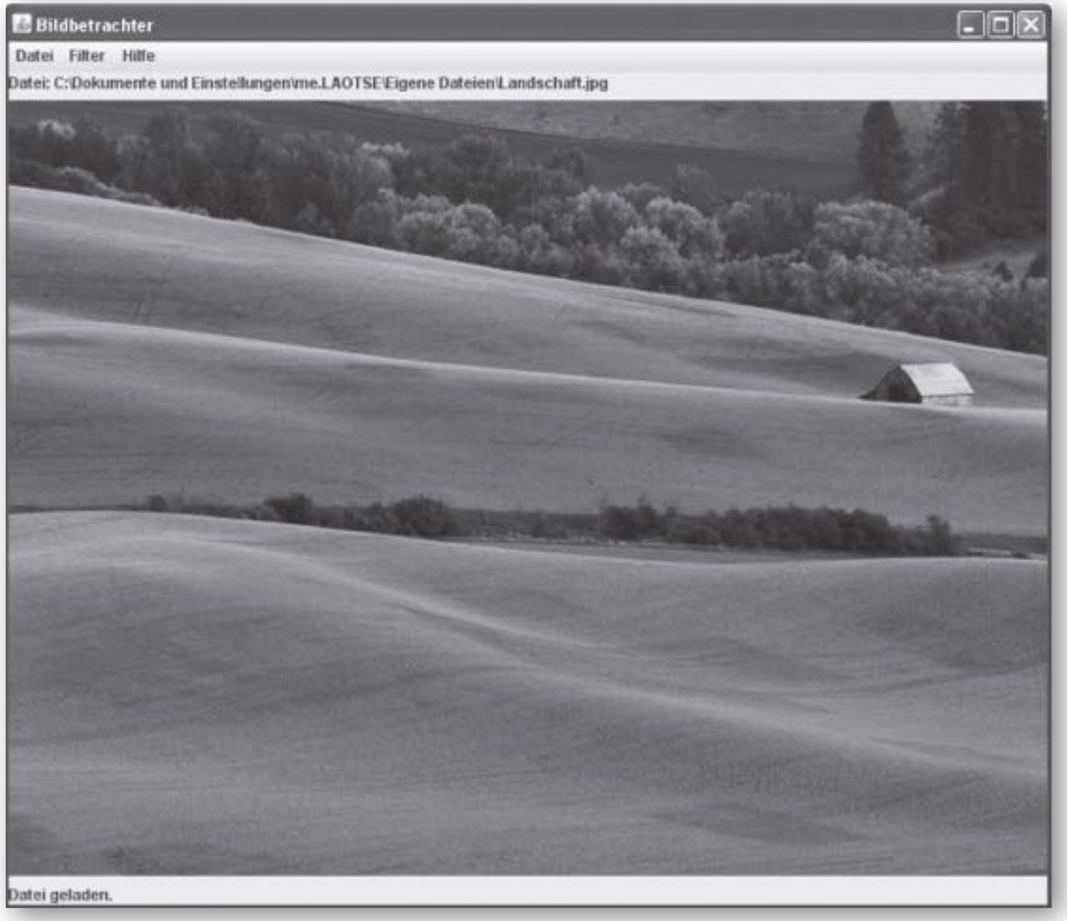
Layout-Manager: BoxLayout

- Ordnet Komponenten entlang einer Achse an (**X_AXIS**, **Y_AXIS**)
- Beachtet die gewünschte Ausrichtung von Komponenten
  - **setAlignmentX**/**setAlignmentY**: Wo soll gemeinsamer Ursprung relativ zur Komponente liegen **[0.f ... 1.f]**?
  - Standard ist links (**Y_AXIS**) bzw. mittig (**X_AXIS**)


```
contentPane.setLayout(new BoxLayout(contentPane, BoxLayout.Y_AXIS));
```

Universität Bremen

Dialoge: Demo



Universität Bremen

Dialoge: Standard

```
private void zeigeInfo()
{
    JOptionPane.showMessageDialog(fenster,
            "Bildbetrachter\n" + VERSION,
            "Info zu Bildbetrachter",
            JOptionPane.INFORMATION_MESSAGE);
}
```

- Werden von **JOptionPane** bereitgestellt
- **showMessageDialog**: Meldungstext plus OK-Schaltfläche
- **showConfirmDialog**: Optionen: Ja, Nein, Abbrechen
- **showInputDialog**: Meldungstext und Eingabefeld
- **showOptionDialog**: Alles zusammen

Universität Bremen

Dialoge: Dateiauswahl

```
JFileChooser dialog = new JFileChooser(".");
if (dialog.showOpenDialog(null) ==
            FileChooser.APPROVE_OPTION) {
   File file = dialog.getSelectedFile(); ...
```

- Werden von **JFileChooser** bereitgestellt
- **showOpenDialog** zeigt „Öffnen“-Dialog, **showSaveDialog** zeigt „Speichern unter“-Dialog
- Erwarten Elternkomponente (oder **null**) als Parameter und liefern zurück, ob Datei gewählt wurde
- **getSelectedFile** liefert die gewählte Datei zurück
- **getSelectedFiles** liefert mehrere gewählte Dateien zurück (wenn **setMultiSelectionEnabled(true)**)

Universität Bremen

Bildbetrachter mit Knöpfen: Demo

Geschachtelte Layouts

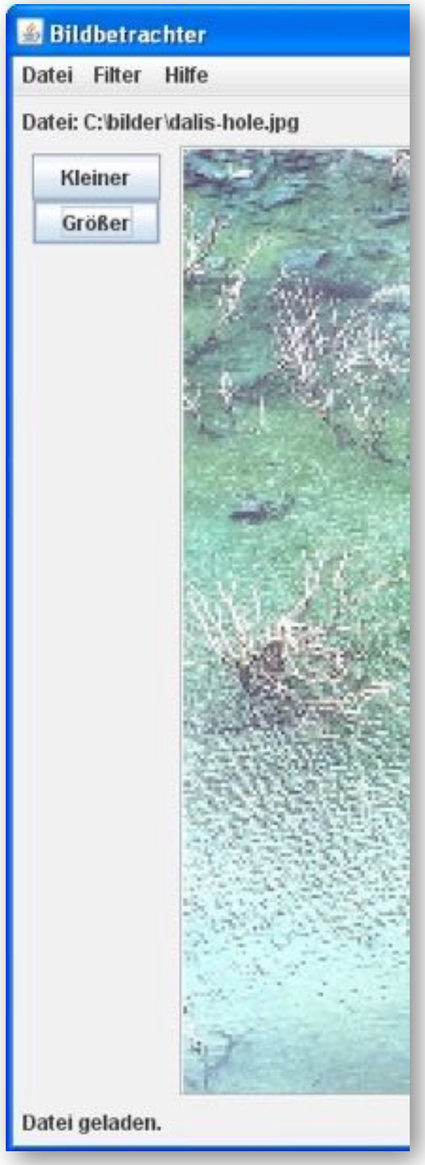
- **GridLayout** ordnet Buttons vertikal

  innerhalb von

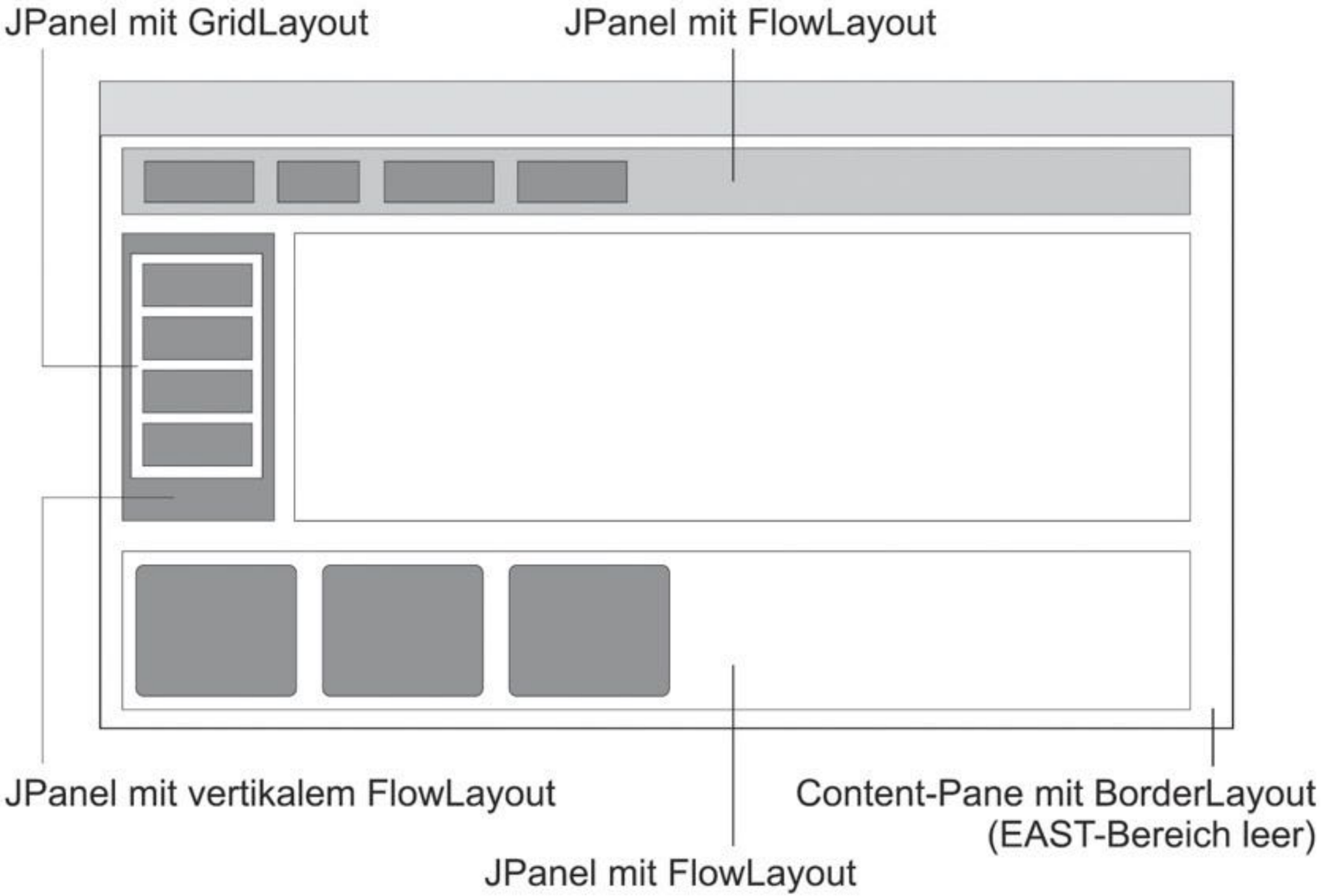
- **FlowLayout** beschränkt ihre Größe

  innerhalb von

- **BorderLayout** platziert Bereich an linkem Fensterrand



Geschachtelte Container



Universität Bremen

Schriften: Begriffe

- Grundlinie, Oberlänge, Unterlänge, Höhe
- Proportionalschrift, (`Feste`) Schrittweite (monospaced)
- Unterschneidung (Kerning) (AV → AV)
- Ligaturen (ffi → ffi)
- Serifen vs. serifenlos

Universität Bremen

Grafik und Schriften: Demo

Universität Bremen

Schriften in Java

- **Font**: Schriftauswahl
  - Name, z.B. **"SansSerif"** oder **Font.SERIF**
  - Stil, (ver-odert) z.B. **Font.BOLD | Font.ITALIC**
  - Größe in Punkten (1pt = 1/72 Zoll)
- **FontMetrics**: Informationen über eine Schrift
  - **getAscent()**, **getDescent()**, **getHeight()**
  - **stringWidth(String)**

```
void topMid(final Graphics g, final String text)
{
    g.setFont(new Font(Font.SERIF, Font.PLAIN, 16));
    final FontMetrics f = g.getFontMetrics();
    final int w = f.stringWidth(text);
    g.drawString(text, (getSize().width - w) / 2,
            f.getAscent());
}
```

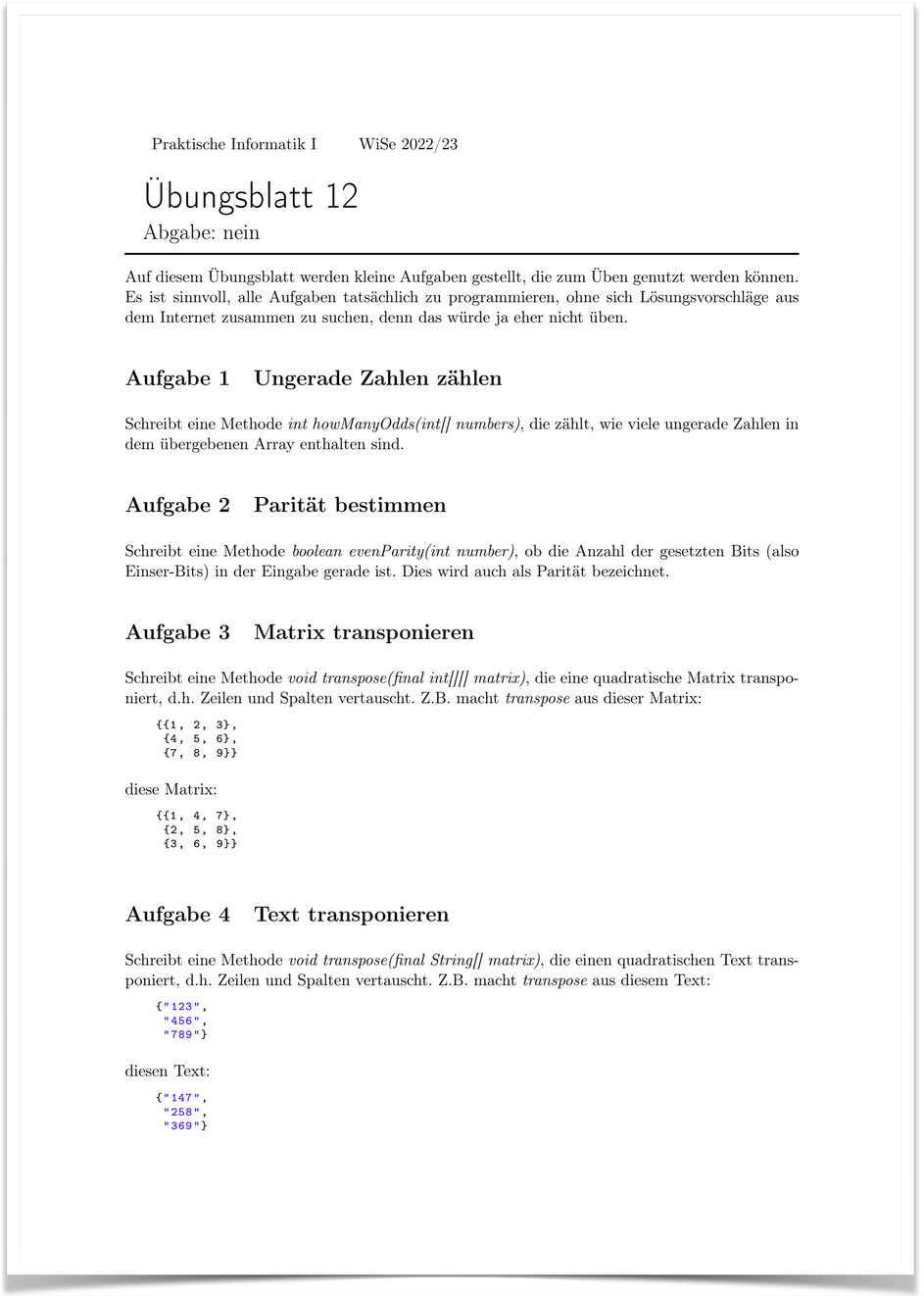
Universität Bremen

Zusammenfassung der Konzepte

- **Layout** und **Container**
- **Dialog**
- **Grafik** und **Schrift** (mit und ohne **Serifen**)
  - **Grundlinie**, **Unter-/Oberlänge**, **Höhe**
  - **Proportionalschrift**
  - **Unterschneidung** und **Ligaturen**

Übungsblatt 12

- 8 Aufgaben zum Üben
- Mit steigendem Schwierigkeitsgrad
- Umgang mit Arrays, Strings und Listen
- Typische Klausuraufgabe
- Vertiefung zu weiteren Themen



Praktische Informatik I WiSe 2022/23

Übungsblatt 12

Abgabe: nein

Auf diesem Übungsblatt werden kleine Aufgaben gestellt, die zum Üben genutzt werden können. Es ist sinnvoll, alle Aufgaben tatsächlich zu programmieren, ohne sich Lösungsvorschläge aus dem Internet zusammen zu suchen, denn das würde ja eher nicht üben.

Aufgabe 1 Ungerade Zahlen zählen

Schreibt eine Methode *int howManyOdds(int[] numbers)*, die zählt, wie viele ungerade Zahlen in dem übergebenen Array enthalten sind.

Aufgabe 2 Parität bestimmen

Schreibt eine Methode *boolean evenParity(int number)*, ob die Anzahl der gesetzten Bits (also Einser-Bits) in der Eingabe gerade ist. Dies wird auch als Parität bezeichnet.

Aufgabe 3 Matrix transponieren

Schreibt eine Methode *void transpose(final int[][] matrix)*, die eine quadratische Matrix transponiert, d.h. Zeilen und Spalten vertauscht. Z.B. macht *transpose* aus dieser Matrix:

```
{{1, 2, 3},
 {4, 5, 6},
 {7, 8, 9}}
```

diese Matrix:

```
{{1, 4, 7},
 {2, 5, 8},
 {3, 6, 9}}
```

Aufgabe 4 Text transponieren

Schreibt eine Methode *void transpose(final String[] matrix)*, die einen quadratischen Text transponiert, d.h. Zeilen und Spalten vertauscht. Z.B. macht *transpose* aus diesem Text:

```
{"123",
 "456",
 "789"}
```

diesen Text:

```
{"147",
 "258",
 "369"}
```