

Praktische Informatik 1

Konzepte benutzerdefinierter Datenstrukturen

Thomas Röfer

- Variablen
- Reihungen (Arrays)
- Zeichenketten (Strings)
- Verbunde (Klassen)
- Enthaltensein
- Entwicklungszyklus
- Musterlösung, Übungsblatt

Modulanmeldung

- Bis **30.11.2007**
- Ausgefüllte Formulare **unterschieden** im Postfach des Prüfungsamtes (MZH, 6. Ebene, Nr. 153) abgeben
 - <http://www.informatik.uni-bremen.de/~res/temp/modul/modulanmeldung07.pdf>
- Studiengang ankreuzen!
- PI-1
 - 03-05-G-700.01
 - Praktische Informatik 1

An das Prüfungsamt Informatik/Medieninformatik
z. Hd. Frau G. Erradi, I. Schabbehard (Informatik)
z. Hd. Frau I. Wolf, M. Schön (Digitale Medien)
Postfach Nr. 153, MZH 6. Ebene

>> Dieses Formular muss fristgerecht im
Prüfungsamt abgegeben werden:
Wintersemester: bis 15. Nov. (WS07/08: 30.11.)
Sommersemester: bis 15. Mai
Wird das Formular nicht fristgerecht abgegeben,
können in diesem Semester keine Prüfungsleistungen
im jeweiligen Modul erbracht werden!

Anmeldung zu Modulen		<input type="checkbox"/> im Wintersemester 20__ / ____ <small>(Abgabe bis 15. November)</small> <input type="checkbox"/> im Sommersemester 20__ <small>(Abgabe bis 15. Mai)</small>
Name: Vorname: Matrikel-Nr.:		Studiengang (bitte ankreuzen): > Informatik <input type="checkbox"/> Diplom <input type="checkbox"/> B.Sc. <input type="checkbox"/> M.Sc. > Digitale Medien <input type="checkbox"/> B.Sc. <input type="checkbox"/> M.Sc. Ich bin im ____ . Fachsemester

Hiermit melde ich mich verbindlich in folgenden Modulen zu Prüfungsleistungen jeglicher Art (gemäß der Scheinverhandlungen) an. Ich versichere, dass ich keine der Prüfungen endgültig nicht bestanden (im Sinne der Prüfungsordnung) habe.

Modul-Nr.	Kurztitel	
		<input type="checkbox"/> erstmalige Anmeldung <input type="checkbox"/> Wiederholung
		<input type="checkbox"/> erstmalige Anmeldung <input type="checkbox"/> Wiederholung
		<input type="checkbox"/> erstmalige Anmeldung <input type="checkbox"/> Wiederholung
		<input type="checkbox"/> erstmalige Anmeldung <input type="checkbox"/> Wiederholung
		<input type="checkbox"/> erstmalige Anmeldung <input type="checkbox"/> Wiederholung
		<input type="checkbox"/> erstmalige Anmeldung <input type="checkbox"/> Wiederholung
		<input type="checkbox"/> erstmalige Anmeldung <input type="checkbox"/> Wiederholung
		<input type="checkbox"/> erstmalige Anmeldung <input type="checkbox"/> Wiederholung
		<input type="checkbox"/> erstmalige Anmeldung <input type="checkbox"/> Wiederholung

Die Modul-Nummer ist im Studiengang „Informatik“ in der Regel identisch mit der VAK.
Für Projekte im Diplom-Hauptstudium bzw. für Bachelorprojekte muss die Anmeldung im jeweiligen ersten Projektsemester erfolgen.

.....

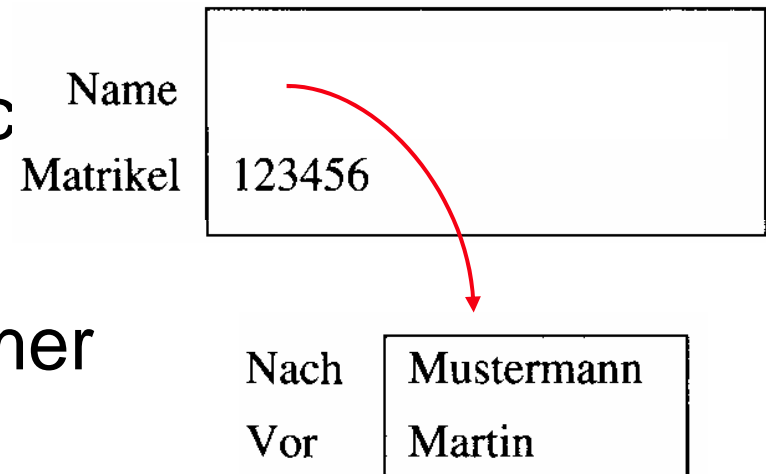
Modellierung des Enthaltenseins

- Möglichkeiten des Enthaltenseins

- Per Wert (by value)
- Per Referenz (by reference)

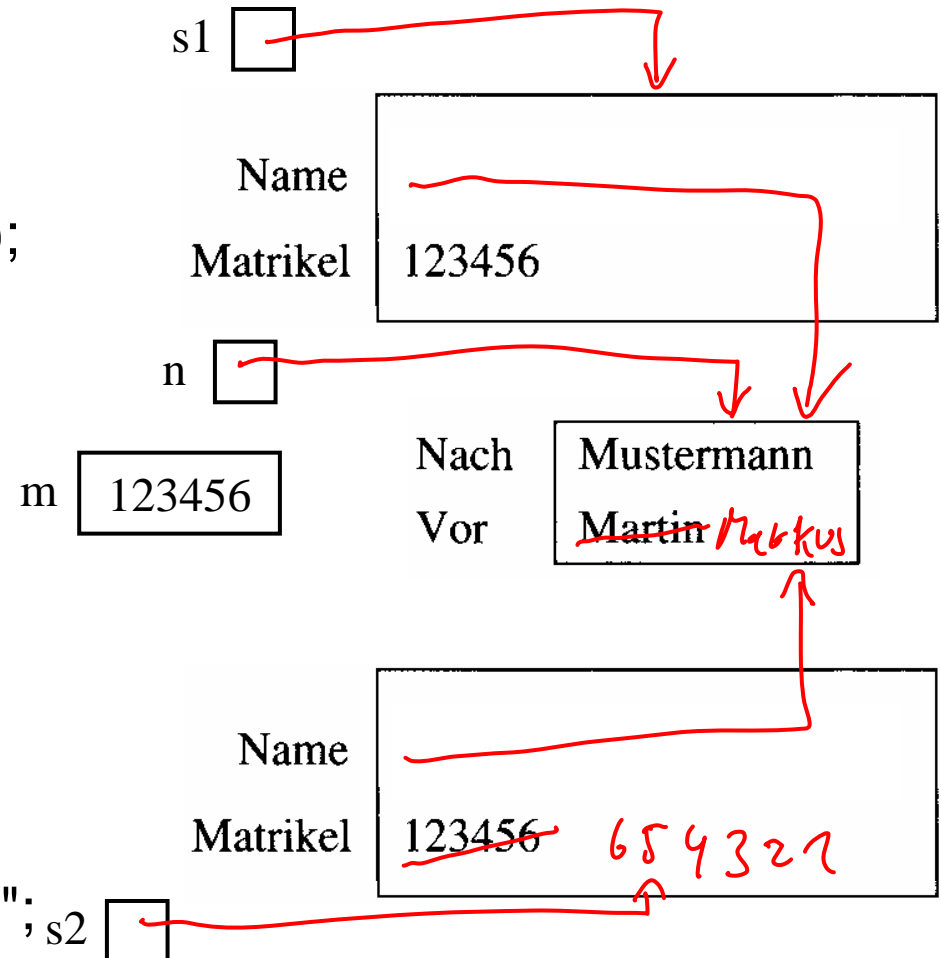
- In Java

- Basisdatentypen sind immer per Wert enthalten
- Klassen und Reihungen sind immer per Referenz enthalten



Beispiel

- `int m = 123456;`
- `Name n = new Name("Mustermann", "Martin");`
- `Student s1 = new Student(n, m);`
- `Student s2 = new Student(n, m);`
- `s2.matrikel = 654321;`
- `s2.name.vorname = "Markus";`



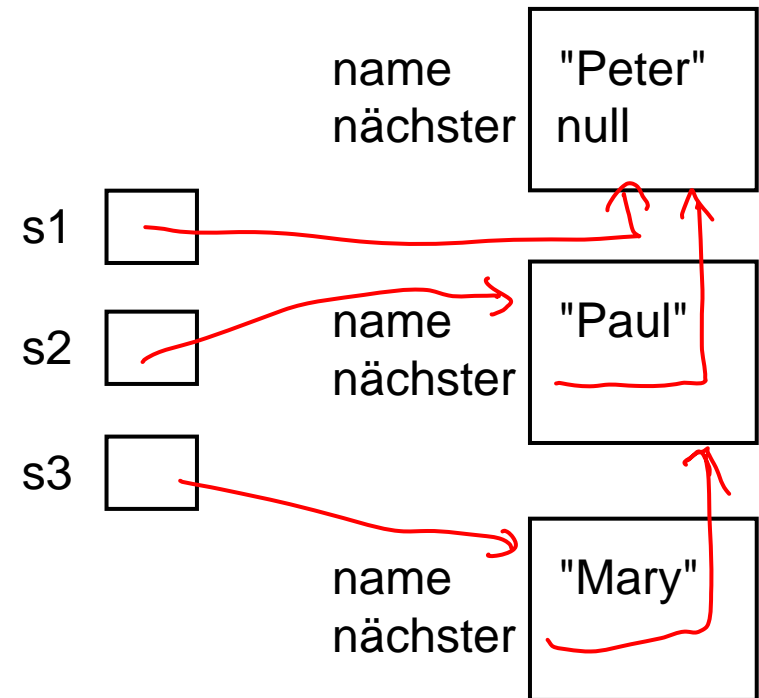
Referenzen in Java

- Eine Referenz zeigt immer
 - auf ein Objekt eines kompatiblen Typs
 - oder auf *null*
- *null* steht für „nichts“

```
class SkatSpieler {  
    String name;  
    SkatSpieler nächster;  
  
    SkatSpieler(String n, SkatSpieler s) {  
        name = n;  
        nächster = s;  
    }  
}
```

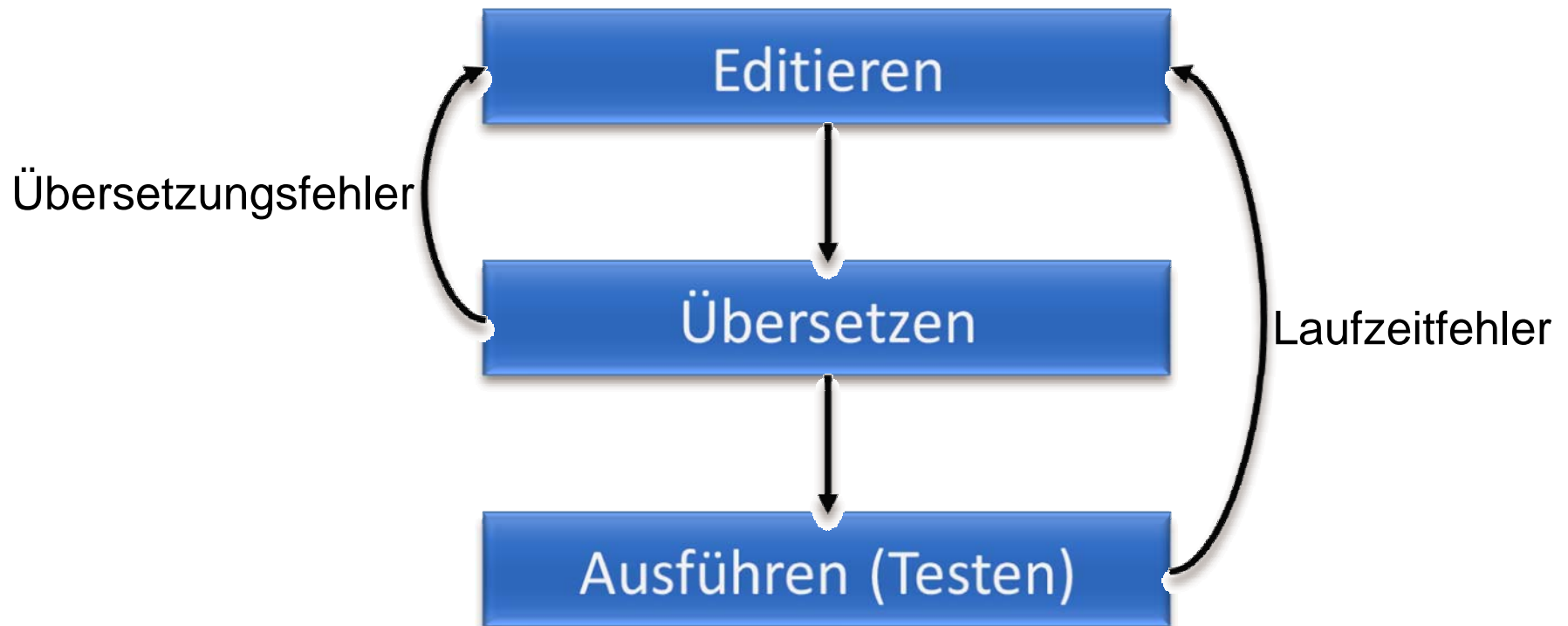
Referenzen in Java

- SkatSpieler s1 = new SkatSpieler("Peter", null);
- SkatSpieler s2 = new SkatSpieler("Paul", s1);
- SkatSpieler s3 = new SkatSpieler("Mary", s2);



```
SkatSpieler s = new SkatSpieler("Mary",  
    new SkatSpieler("Paul", new SkatSpieler("Peter", null)));
```

Entwicklungszyklus



Fehler beim Programmieren

- Tippfehler (auch Copy and Paste)
 - Semikolon vergessen, Klammern passen nicht...
 - Können vom Compiler entdeckt werden, müssen aber nicht
- Strukturfehler
 - Compiler verwendet andere Zuordnung als man denkt
- Typfehler
 - Werden bei Zuweisungen erkannt
 - Bleiben in Ausdrücken teilweise unerkannt
- Fehler im Algorithmus
 - Kommt bei Informatikern nicht vor :-)

```
for (int i = 0; i < 10; ++i) {  
    for (int j = 0; j < 10; ++i) {  
        System.out.println(i + j);  
    }  
}
```

```
if (a == 1)  
    if (b == 1)  
        System.out.println("ok");  
else  
    System.out.println("a != 1");
```

```
int i = "1";  
float f = 3 / 2;
```


Fehlerbehebung: Übersetzungsfehler

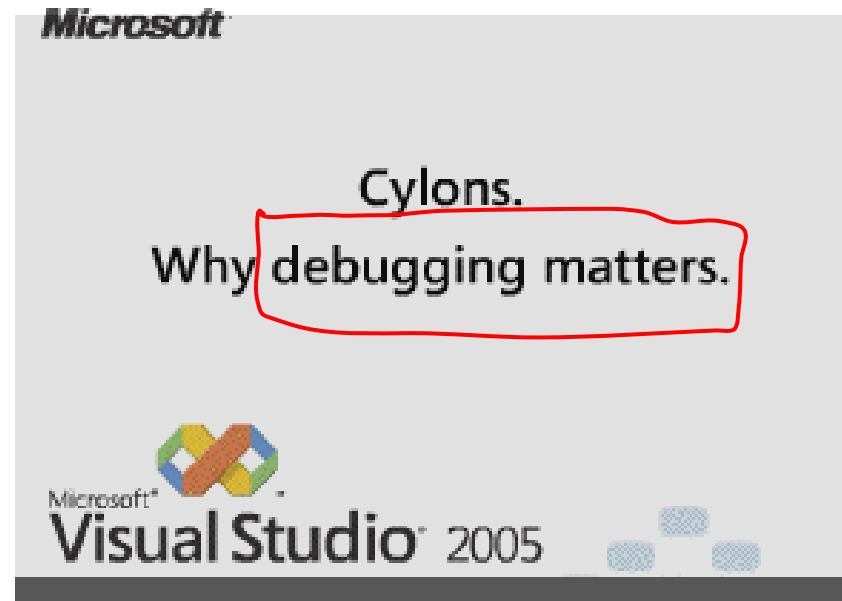
- Nur den ersten Fehler beachten, die anderen könnten Folgefehler sein
 - BlueJ zeigt nur den ersten Fehler an
- Fehlermeldung lesen
 - Dokumentation zu Fehlermeldung lesen
- Quelltext bei der angegebenen Zeilennummer ansehen
 - Dokumentation zu falschem Konstrukt lesen



Fehlerbehebung: Fehler während der Programmausführung

- Falls eine Fehlermeldung angezeigt wird, diese lesen und den Quelltext bei der angegebenen Zeilennummer ansehen
- Testausgaben einbauen oder Debugger verwenden
 - Aber keine Fehler durch solche Ausgaben einbauen!
- Verschiedene Testfälle durchspielen
 - Wann tritt der Fehler auf, wann nicht?

Hinweise



**Nur ein verstandener Fehler
ist ein beseitigter Fehler!**

Debugger benutzen!

**Nach Änderungen
Übersetzen nicht vergessen!**

Musterlösung zu Übungsblatt 1

• Gleichheit

- $-3_{10} \neq 1011_2$
- $(16_{10} - 3_{10}) \bmod 16_{10} = 1011_2 \bmod 10000_2$

• L^AT_EX

- $2_{10} \rightarrow 2_10$
- $2_{\{10\}} \rightarrow 2_{10}$
- $\sum_{j=0}^{n-1} a_j \rightarrow \sum_{j=0}^{n-1} a_j$

Praktische Informatik I WS 2007/08

Übungsblatt 1 Musterlösung

Aufgabe 1 von Neumann Architektur (50%)

a) Erläutert die von Neumann Architektur, insbesondere den Aufbau des Prozessors.

Prinzipiell besteht ein Rechner aus der **Prozessoreinheit** und der **Speichereinheit**. Heutige Rechner haben zusätzlich noch eine Reihe von **Input-/Output-Geräten**, z.B. Grafikkarte, Modem, Maus etc., die durch einen (oder mehrere) **Bus** zwecks Kommunikation miteinander und dem Speicher verbunden sind.

Ein Prozessor besteht aus dem **Steuerwerk**, der **arithmetisch-logischen Einheit (ALU)**, die mathematische Operationen ausführt, und einigen **Registern**. Bei der von Neumann Architektur werden Befehle und Daten gemeinsam in einem Speicher gehalten. Das Steuerwerk ist für den Transfer dieser Daten vom Speicher zum Prozessor zuständig. Die Programme werden in einem **fundamentalen Instruktionszyklus** (basic instruction cycle) ausgeführt, ein spezielles Register, der **Befehlszähler** (instruction counter) speichert die Adresse des jeweils nächsten Befehls, das **Instruktionsregister** speichert den gerade auszuführenden Befehl selbst. Das Buch zur Vorlesung beschreibt den Ablauf folgendermaßen:

Fetch. Hole den Befehl, dessen Adresse im Befehlszähler steht, aus dem Speicher in das Instruktionsregister.

Increment. Inkrementiere den Befehlszähler, damit er auf die nächste auszuführende Instruktion verweist.

Decode. Dekodiere die Instruktion, damit klar wird, was zu tun ist.

Fetch Operands. Falls nötig, hole die Operanden aus den im Befehl bezeichneten Stellen im Speicher.

Execute. Führe die Instruktion aus, ggf. durch die ALU. Bei einem Sprung wird hier ein neuer Wert in den Befehlszähler geschrieben.

Loop. Beginne wieder von vorne.

b) Erläutert den von Neumann'schen Flaschenhals (bottleneck). Was wird verwendet, um diesen Effekt abzumildern?

Der Prozessor hat einen bestimmten **Taktyklus**, dieser ist üblicherweise schneller als die **Datenübertragungsrate** vom Speicher zum Prozessor. Daher muss der Prozessor eventuell mit der Weiterverarbeitung warten, bis der Datentransfer neuer Daten/Instruktionen vom Speicher abgeschlossen ist.

Um dieses Problem zu verringern, verwendet man auf der CPU viele Register und einen schnellen Zwischenspeicher, den **Cache**, in dem man Instruktionen und Daten, die (voraussichtlich) in nächster Zeit benötigt werden, ablegen kann.

c) Erläutert das Schichten-Modell der Software und skizziert kurz seine Komponenten.

Übungsblatt 3 – Aufgabe 1

Korrektheitsbeweis nach Floyd

- Schleifeninvariante
- Vor der Schleife
- Invarianz in der Schleife
 - Zu Beginn der Schleife
 - Nach einem Durchlauf
- Nach der Schleife

```
int calcSum(int[] a) {  
    int n = a.length;  
    int sum = 0;  
    int i = 0;  
    while (i < n) {  
        sum = sum + a[i];  
        i = i + 1;  
    }  
    return sum;  
}
```

Übungsblatt 3 – Aufgabe 1 – Beispiel

- Schleifeninvariante

$$\min_{j=0}^{n-1} a_j = \min(m, \min_{j=1}^{n-1} a_j)$$

- Vor der Schleife

$$\begin{aligned} \min_{j=0}^{n-1} a_j &= \min(a_0, \min_{j=1}^{n-1} a_j) \\ &= \min_{j=0}^{n-1} a_j \end{aligned}$$

```
static int minimum(int[] a) {
    int n = a.length;
    int m = a[0];
    int i = 1;
    while (i < n) {
        m = Math.min(m, a[i]);
        i = i + 1;
    }
    return m;
}
```

Übungsblatt 3 – Aufgabe 1 – Beispiel

• Invarianz in der Schleife

• Zu Beginn

$$\min_{j=0}^{n-1} a_j = \min(m, \min_{j=i}^{n-1} a_j)$$

$$m' = \min(m, a_i)$$

$$i' = i + 1$$

• Nach einem Durchlauf

$$\begin{aligned} \min_{j=0}^{n-1} a_j &= \min(m', \min_{j=i'}^{n-1} a_j) \\ &= \min(\min(m, a_i), \min_{j=i+1}^{n-1} a_j) \\ &= \min(m, \min_{j=i}^{n-1} a_j) \end{aligned}$$

```
static int minimum(int[] a) {
    int n = a.length;
    int m = a[0];
    int i = 1;
    while (i < n) {
        m = Math.min(m, a[i]);
        i = i + 1;
    }
    return m;
}
```


Übungsblatt 3 – Aufgabe 1 – Beispiel

- Nach der Schleife

$$\min_{j=0}^{n-1} a_j = \min(m, \min_{j=n}^{n-1} a_j)$$

$$\underline{\hspace{1cm}} = \min(m)$$


$$\underline{\hspace{1cm}} = m$$

qed

```
static int minimum(int[] a) {
    int n = a.length;
    int m = a[0];
    int i = 1;
    while (i < n) {
        m = Math.min(m, a[i]);
        i = i + 1;
    }
    return m;
}
```


Übungsblatt 3 – Aufgabe 2

- Algorithmus
 - Wegstreichen der Vielfachen von Primzahlen
 - Primzahlen bleiben übrig
- Testen
 - Trivialfälle, Normalfälle, Grenzfälle
 - Nur Anzahl der gefundenen Primzahlen abdrucken



0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19