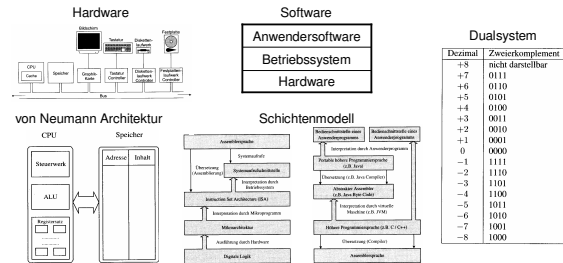


Datenorganisation und Datenstrukturen

Thomas Röfer

Zeichensätze/Gleitkommazahlen
Variablen
Reihungen
Verbunde (Klassen)
Enthaltensein

Rückblick „Aufbau und Funktionsweise eines Computers“



Zeichensätze

- › EBCDIC (IBM)
- › ASCII (ISO 7 Bit)
- › ISO Latin 1 (ISO 8859-1)
- › Unicode (16 Bit)
- › UTF-8

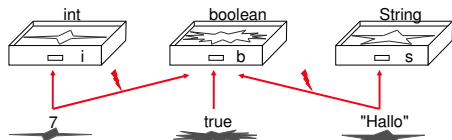
oct	hex	0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160
0	0	nul	dle	!	1	A	Q	a	q									
1	1	soh	dc1	"	2	B	R	b	r									
2	2	stx	dc2	#	3	C	S	c	s									
3	3	etx	dc3	\$	4	D	T	d	t									
4	4	eot	dc4	%	5	E	U	e	u									
5	5	enq	nak	&	6	F	V	f	v									
6	6	ack	syn	'	7	G	W	g	w									
7	7	bel	etb	(8	H	X	h	x									
10	8	bs	can)	9	I	Y	i	y									
11	9	ht	em	*	:	J	Z	j	z									
12	A	lf	sub	+	;	K	[k	{									
13	B	vt	esc	,	<	L	\	l										
14	C	ff	fs	.	=	M]	m	}									
15	D	cr	gs	-	>	N	^	n	~									
16	E	so	rs	/	?	O	_	o	del									
17	F	si	us															

Gleitkommazahlen (IEEE 754)

- › **Darstellung**
 - › Mantisse und Exponent
 - $z = (-1)^s \cdot \text{Mantisse} \cdot 2^{\text{Exponent}}$
- › **Darstellungsbereiche**
 - › float: $\pm 1.40239846E-45$ bis $\pm 3.40282347E+38$
 - › double: $\pm 4.94065645841246544E-324$ bis $\pm 1.79769313486231570E+308$
- › **Vorteile**
 - › Erlauben Nachkommastellen
 - › Decken sehr großen Zahlenbereich ab
- › **Nachteile**
 - › Abdeckung ist lückenhaft
 - › Rundungsfehler
- › **Beispiel**
 - › $1 + 10^{20} - 1 = 0$

	v	30 - Exponent - 23	22 - Mantisse - 0
float	1 Bit	8 Bit	23 Bit
	v	62 - Exponent - 52	51 - Mantisse - 0
double	1 Bit	11 Bit	52 Bit

Variablen



- › **Variablen haben**
 - › einen Namen (*i, b, s, ...*)
 - › einen Typ (*int, boolean, String, ...*)
 - › einen Wert (*7, true, "Hallo", ...*)
- › **Richtig**
 - › `int i = 7;`
 - › `boolean b = true;`
 - › `String s = "Hallo";`
- › **Falsch**
 - › `boolean b = 7;`
 - › `boolean b = "Hallo";`

Reihungen (arrays)

- › **Definition**
 - › Eine Reihung besteht aus einer festen Anzahl von Elementen gleichen Typs, auf die mit einem Index zugegriffen werden kann.
- › **Beispiel**

Zeitpunkt	1	2	3	4	...	30	31
Signalstärke	10.5	10.5	12.2	9.8	...	13.1	13.3
- › **Mathematische Interpretation**
 - › Eine Reihung repräsentiert eine Funktion vom Indexbereich in den Wertebereich
 - › Obiges Beispiel kann also als eine Funktion $f: \mathbb{N} \rightarrow \mathbb{R}$ angesehen werden
- › **Anwendung**
 - › Nur Funktionen durch Reihungen abbilden, die für die meisten Werte des Indexbereiches definiert sind, sonst ist der Speicherbedarf zu hoch
 - › Der Zugriff auf Elemente einer Reihung erfolgt in konstanter Zeit, ist also unabhängig von der Anzahl der Elemente in der Reihung

Reihungen in Java

Variante 1

```
String[] a = new String[50];
a[0] = "Herbert Pappelbusch";
a[1] = "Irene Schmidt";
a[2] = "Claudia Miesmuffel";
.
.
.
a[48] = "Tobias Kleinemann";
a[49] = "Mareike Bumtrupp";
```

Variante 2

```
String[] a = {
    "Herbert Pappelbusch",
    "Irene Schmidt",
    "Claudia Miesmuffel",
    .
    .
    .
    "Tobias Kleinemann",
    "Mareike Bumtrupp"
};
```

0	Herbert Pappelbusch
1	Irene Schmidt
2	Claudia Miesmuffel
.	.
.	.
.	.
48	Tobias Kleinemann
49	Mareike Bumtrupp

Zweidimensionale Reihungen

Variante 1

```
String[][] a = {
    {"Herbert", "Pappelbusch", "Herr"},
    {"Irene", "Schmidt", "Frau"},
    {"Claudia", "Miesmuffel", "Frau"},
    .
    .
    .
    {"Tobias", "Kleinemann", "Herr"},
    {"Mareike", "Bumtrupp", "Frau"}
};
```

Variante 2

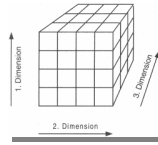
```
String[][] a = new String[50][3];
a[0][0] = "Herbert";
a[0][1] = "Pappelbusch";
.
.
.
a[48][0] = "Tobias";
a[48][1] = "Kleinemann";
a[48][2] = "Herr";
a[49][0] = "Mareike";
a[49][1] = "Bumtrupp";
a[49][2] = "Frau";
```

		2. Dimension		
		0	1	2
0	Herbert	Pappelbusch	Herr	
1	Irene	Schmidt	Frau	
2	Claudia	Miesmuffel	Frau	
.	.	.	.	
.	.	.	.	
.	.	.	.	
48	Tobias	Kleinemann	Herr	
49	Mareike	Bumtrupp	Frau	

Mehrdimensionale Reihungen

Dimensionen

- Eine beliebige Anzahl von Dimensionen ist möglich
 - Speicherplatzverbrauch bedenken!
- Eine zweidimensionale Reihung ist eine Reihung von Reihungen
- Daher können Reihungen in weiteren Dimensionen selbst auch wieder unterschiedliche Größen haben



Beispiel

```
char[][] a = new char[5][];
a[0] = new char[3];
a[1] = new char[2];
a[2] = new char[4];
.
.
.
a[0][0] = 'A';
```

Index	0	1	2	3
0	A	B	C	
1	D	E		
2	F	G	H	I
.

Zeichenketten (strings)

Spezielle Form der Reihung

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Zeichen	'H'	'a'	'l'	'l'	'o'	' '	'E'	'r'	's'	't'	's'	'e'	'm'	'e'	's'	't'	'e'	'r'

Gebräuchliche Schreibweise: "Hallo Erstsemester"

In Java

- Möglich, aber umständlich und nicht weiter unterstützt
 - `char[] c = {'H','a','l','l','o',' ','E','r','s','t','s','e','m','e','s','t','e','r'};`
- Gebräuchlich:
 - `String s = "Hallo Erstsemester";`
- String ist ein Verbund (eine Klasse), die eine Zeichenreihung kapselt:
 - `char[] c = {'H','a','l','l','o',' ','E','r','s','t','s','e','m','e','s','t','e','r'};`
 - `String s = new String(c);`

Verbunde (records, structs)

Definition

- Ein Verbund besteht aus Elementen möglicherweise unterschiedlichen Typs, auf die über ihren Namen zugegriffen werden kann.

Beispiel

- Stammdaten eines **Beschäftigten** (lt. Buch)
- Felder *Name*, *Vorname*, ...

Anwendung

- Für Daten unterschiedlichen Typs oder unterschiedlicher Bedeutung, die inhaltlich zusammen gehören
- Der Zugriff über einen (berechneten) Index wird nicht benötigt oder ist nicht sinnvoll

Name	"Mustermann"
Vorname	"Martin"
GebTag	10
GebMonat	05
GebJahr	1930
Familienstand	"verheiratet"
...	...

Verbunde (Klassen) in Java

Beispiel

```
class Stammdaten
{
    String name;
    String vorname;
    int gebTag;
    int gebMonat;
    int gebJahr;
    String familienstand;
}
.
.
.
Stammdaten s = new Stammdaten();
s.name = "Mustermann";
s.vorname = "Martin";
s.gebTag = 10;
```

Name	"Mustermann"
Vorname	"Martin"
GebTag	10
GebMonat	05
GebJahr	1930
Familienstand	"verheiratet"
...	...

Konstruktoren für Klassen

- Definition
 - Ein *Konstruktor* initialisiert eine *Instanz einer Klasse*
- Beispiel


```
class Stammdaten
{
    String name;
    :
    Stammdaten(String n, String v, int t, int m,
                int j, String f)
    {
        name = n;
        vorname = v;
        gebTag = t;
        gebMonat = m;
        gebJahr = j;
        familienstand = f;
    }
}
```

Name	"Mustermann"
Vorname	"Martin"
GebTag	10
GebMonat	05
GebJahr	1930
Familienstand	"verheiratet"
...	...

Stammdaten s = new Stammdaten("Mustermann", "Martin", 10, 5, 1930, "verheiratet");

Verbund in Verbund

- class Name


```
{
    String nachname;
    String vorname;
}
```
- class Datum


```
{
    int tag;
    int monat;
    int jahr;
}
```
- class Stammdaten


```
{
    Name name;
    Datum gebDatum;
    String familienstand;
}
```

Name	Nachname	"Mustermann"
	Vorname	"Martin"
GebDatum	Tag	10
	Monat	05
	Jahr	1930
Familienstand	"verheiratet"	

Stammdaten s = new Stammdaten();
s.name = new Name();
s.gebDatum = new Datum();
s.name.nachname = "Mustermann";
s.name.vorname = "Martin";
s.gebDatum.tag = 10;
:

Nochmal mit Konstruktoren

- class Name


```
{
    String nachname;
    String vorname;
    Name(String n, String v)
    {
        nachname = n;
        vorname = v;
    }
}
```
- class Datum


```
{
    int tag;
    int monat;
    int jahr;
    Datum(int t, int m, int j)
    {
        tag = t;
        monat = m;
        jahr = j;
    }
}
```
- class Stammdaten


```
{
    Name name;
    Datum gebDatum;
    String familienstand;
    Stammdaten(Name n, Datum g, String f)
    {
        name = n;
        gebDatum = g;
        familienstand = f;
    }
    Stammdaten(String n, String v, int t, int m, int j, String f)
    {
        name = new Name(n, v);
        gebDatum = new Datum(t, m, j);
        familienstand = f;
    }
    Stammdaten s = new Stammdaten(
        new Name("Mustermann", "Martin"),
        new Datum(10, 5, 1930),
        "verheiratet");
}
```

Kombinationen aus Reihung und Verbund

- Verbund und Reihung können beliebig kombiniert werden
 - Reihungen von Verbunden
 - Reihungen in Verbunden
- In Java


```
class Student
{
    Name name;
    int matrikel;
    int[] punkte = new int[13];
    Student(Name n, int m)
    {
        name = n; matrikel = m;
    }
    Student[] semester = new Student[300];
    semester[0] = new Student(
        new Name("Mustermann", "Martin"), 123456);
    semester[0].punkte[0] = 95;
}
```

Name	Nach	Mustermann
	Vor	Martin
Matrikel	123456	
Punkte	0	...
	1	
	12	

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";
```