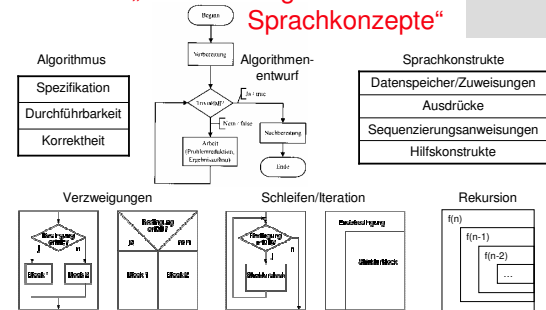


Konzepte benutzerdefinierter Datenstrukturen

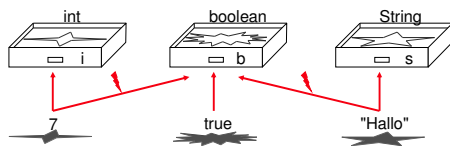
Thomas Röfer

Variablen
Reihungen
Verbunde (Klassen)
Enthaltensein

Rückblick „Abstrakte Algorithmen und Sprachkonzepte“



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 Speicherverbrauch 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**

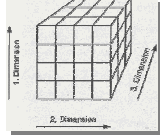
```
String[][] a = {
    {"Herbert", "Pappelbusch", "Herr"},
    {"Irene", "Schmidt", "Frau"},
    {"Claudia", "Miesmuffel", "Frau"},
    ...
    {"Tobias", "Kleinemann", "Herr"},
    {"Mareike", "Bumtrupp", "Frau"}
};
```
- Variante 2**

```
String[][] a = new String[50][3];
a[0][0] = "Herbert";
a[0][1] = "Pappelbusch";
...;
```

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

Name	"Mustermann"
Vorname	"Martin"
GebTag	10
GebMonat	05
GebJahr	1930
Familienstand	"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;
}
Stammdaten s = new Stammdaten();
s.name = new Name();
s.gebDatum = new Datum();
s.name.nachname = "Mustermann";
s.name.vorname = "Martin";
s.gebDatum.tag = 10;
```

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

Nochmal mit Konstruktoren

```

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

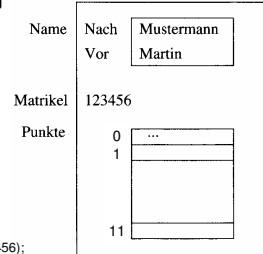
class Datum
{
    int tag;
    int monat;
    int jahr;
    Datum(int t, int m, int j)
    {
        tag = t;
        monat = m;
        jahr = j;
    }
}

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

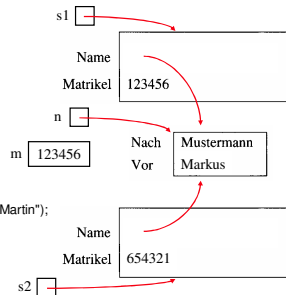
```

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[12];
    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;
}
    
```



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
 - entweder auf ein Objekt eines kompatiblen Typs
 - oder auf null
- Beispiel


```

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

 - 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)));

