10. Felder (Arrays)

Teil 1

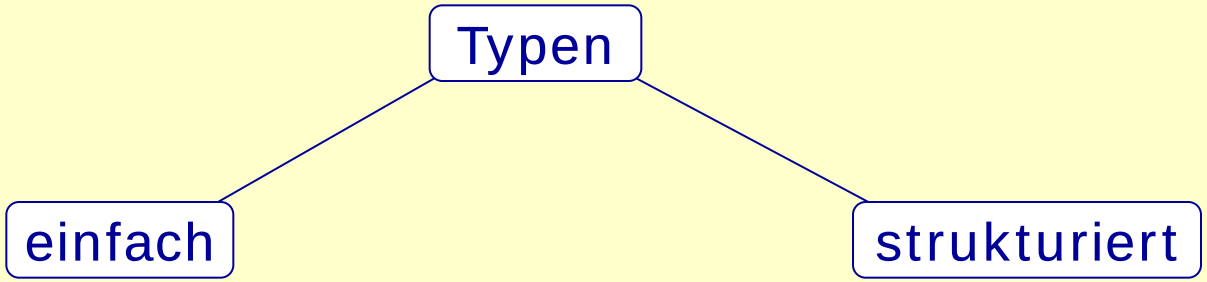
Java-Beispiele:

Echo.java
Primzahlen.java
Monate.java

Schwerpunkte

- Klassifikation von Typen in Programmiersprachen
- Array: einziger strukturierter Typ in Java
- Deklaration, Erzeugung
- Initialisierung, Zugriff
- Typische Verarbeitung von Arrays: Iteration
- Methoden (Prozeduren) für Arrays variabler Länge
- Programmparameter
- Techniken mit Arrays

Klassifikation von Typen in Programmiersprachen



einfach - elementar	zusammengesetzt aus anderen Typen
• Ganze Zahlen (byte, short, int, long) • Reelle Zahlen (float, double) • Boolean • Zeichen	• Arrays (Felder) 1) • Records (Strukturen) 2) • Mengen 3) • Zeiger (Pointer) 2)

1) in Java direkt als Typ
2) in Java: über Klassen
3) in Java:
- selbst implementieren
- oder: API-Klasse Set

Grundprinzip von Arrays

Arrays (Felder)

Folge von Werten
- desselben Typs
- fester Länge
der Werte

Warum ?

Nr. des Elements
→ Adresse steht fest
(Direktzugriff)

- Array von 'int'-Werten:

2	1000	-2	0	-1	1000

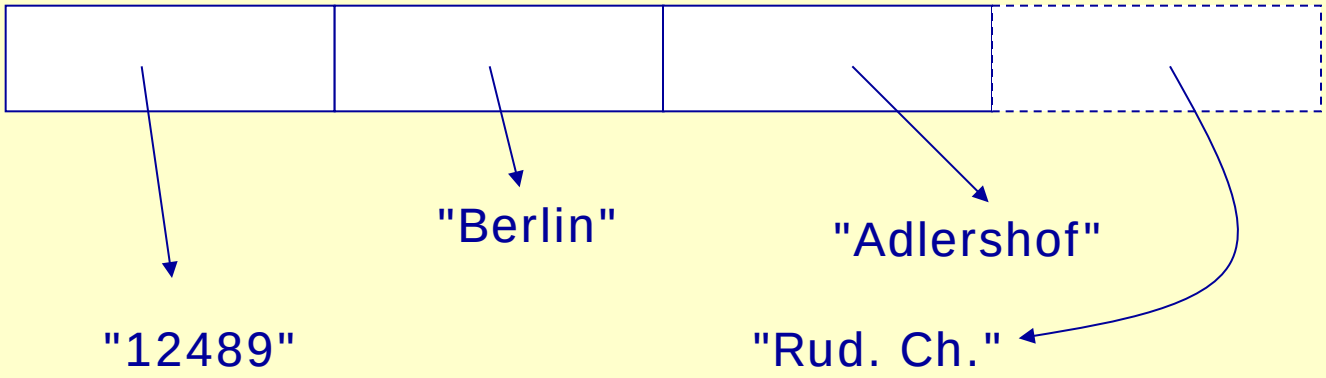
- Array von 'char'-Werten:

'a'	'b'	'e'	'r'

Array von 'String'–Werten: Strings: feste Länge?

Strings: Objekte einer Klasse
→ über Adressen repräsentiert
(Vorgriff auf Teil III)

also:



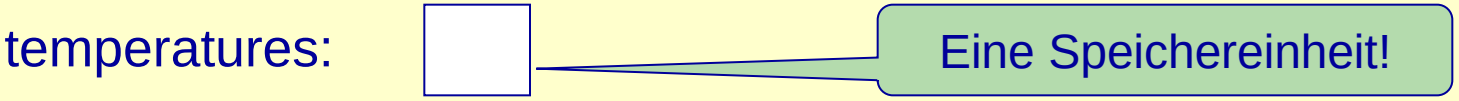
Arbeit mit Arrays in Java:

- Deklaration
- Erzeugung
- Zugriff

Erstellen von Arrays (1)

1. Schritt: Deklaration

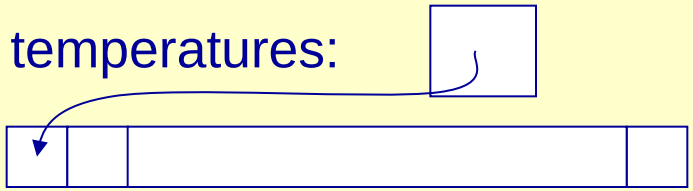
```
double [] temperatures;
String [] wochenTage;
```



2. Schritt: Erzeugung

```
temperatures = new double [20];
wochenTage = new String [7];
```

Von nun an steht die Größe fest



Array-Variablen sind Referenzen, d.h. Adressen → beliebig große Arrays

Erstellen von Arrays (2)

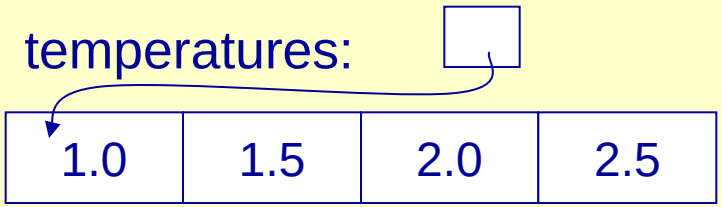
1. und 2. Schritt: zusammen

(bei Deklaration: Anzahl der Elemente steht fest)

```
double [] temperatures
   = new double [20];
```

Erzeugung durch Anfangswert

```
double [] temperatures
     = {1.0, 1.5, 2.0, 2.5};
```



Arrays in Pascal

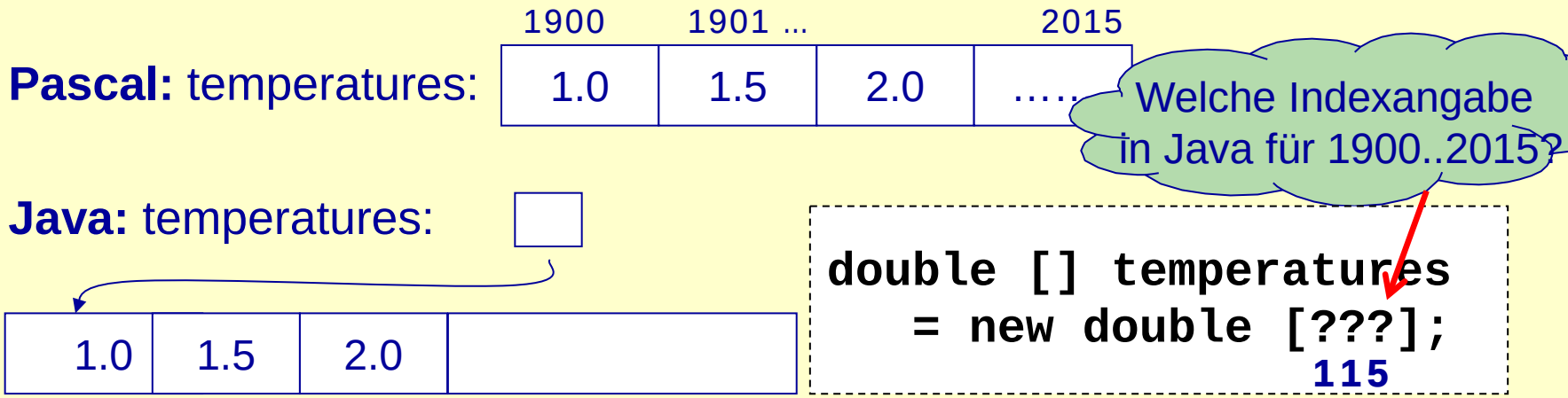
Deklaration und Erzeugung nicht getrennt

```
VAR temperatures:
  ARRAY [1900..2015] OF real;
```

Semantische Indexbereiche

Arrays keine Referenzen

Syntax lesbarer



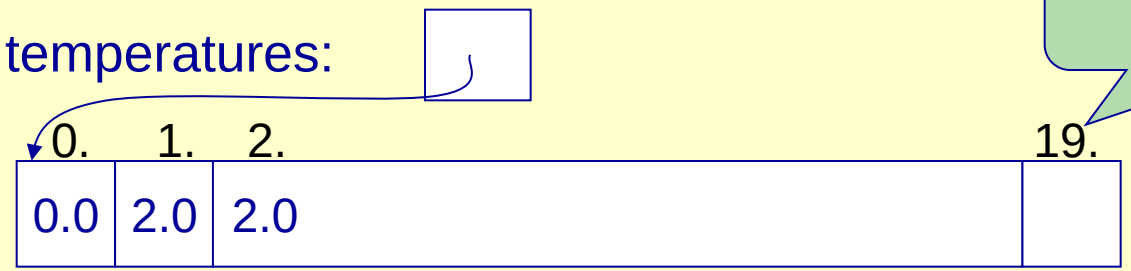
Zugriff: einzelne Komponenten

```
double [] temperatures
   = new double [20];

temperatures[0] = 0.0;
temperatures[1] = 2.0;
temperatures[2] =
   temperatures[1];
```

Anzahl der Elemente: 20
Index von 0 .. 19

Indexbereich:
0 .. 19



Array: sequentielle Struktur fester Länge

Typische Verarbeitung: for-Anweisung

```
final int N = 100;
int [] arr = new int[N];

for (int i = 0; i < N; i++)
     // i: 0 .. 99
   arr[i] = -i;
```

nicht: i <= N !
→ Laufzeitfehler

Methoden: für Arrays beliebiger Länge

Methode: Summe der Array-Elemente für beliebig lange Arrays

```
int summe (int[] vektor) {
   int gesamt = 0;

   for (int i = 0; i < vektor.length; i++)
      gesamt += vektor[i];
   return gesamt;
}
```

Anwendung:

```
int [] v1 = new int[8];
int [] v2 = new int[100];
   . . . // v1, v2: fuellen
s1 = summe(v1);
s2 = summe(v2);
```

Technik möglich, da Arrays in Java Referenzen

Programmparameter

(Konsolenargumente)

```
% java Square 7
% java Square 100
```

Programmparameter

```
class Temperature {
   // Convert temperature
   // from Fahrenheit to Centigrade

   public static void main (String[] args) {
        double tempFahr; // Fahrenheit
        double tempCels; // Celsius
        System.out.print("Temperature (deg F): ");
        tempFahr = Keyboard.readDouble();
        tempCels = (5.0 * (tempFahr - 32.0)) / 9.0;
        System.out.print(tempFahr);
        System.out.print(" deg F is ");
        System.out.print(tempCels);
        System.out.println(" deg C");
   }
}
```

Formaler Parameter: Verwendung im Methodenkörper?

Typ des Parameters von main?

Aktueller Parameter ?

Aufruf von main ?

```
% javac Temperature.java
% java Temperature

Temperature (deg F): 10
10 deg F is -12.222222222222221 deg C
```

Interpreter ruft main() auf …

… und übergibt Programmparameter

Programmparameter

```
class select {
   public static void main (String[] args) {
      ...
   }
}
```

formaler Parameter vom Typ 'Array' von 'String'

aktueller Parameter ?

```
% java select -p1-5 file.ps
```

Parameter der Kommandozeile

Programmparameter

args:

-p1-5 | file.ps

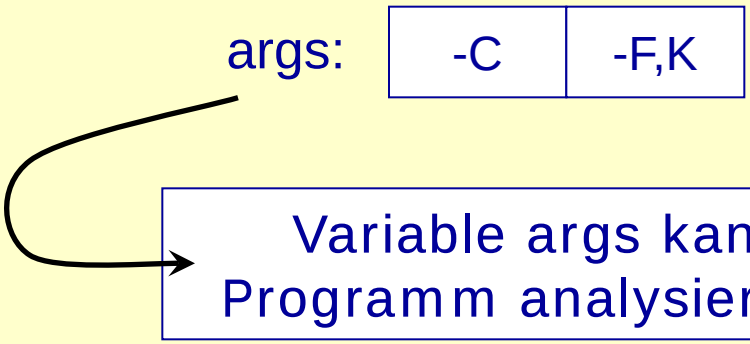
= aktueller Parameter

Programm: gesteuert von Parametern beim Aufruf des Programms

```
% psnup -4 folie.ps f4.ps
% psselect -p1-2,6 f1.ps f2.ps
% java temp -C -F,K
% java temp -K -C,F
```

UNIX-Dienstprogramme

Java-Programme



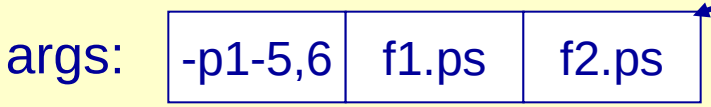
Beispiel: Programmparameter

Echo.java

```
public class Echo {
   public static void main(String[] args) {
      for (int i = 0; i < args.length; i++)
         System.out.print(args[i] + "\n");
   }
}
```

Wirkung?

```
% java Echo -p1-5,6 f1.ps f2.ps
-p1-5,6
f1.ps
f2.ps
```



3 Programmparameter
→ 1 aktueller Parameter vom Typ Array der Länge 3

Echo: Variante

bisher:

```
% java Echo -p1-5,6 f1.ps f2.ps
-p1-5,6
f1.ps
f2.ps
```

neu:

```
% java Rev -p1-5,6 f1.ps f2.ps
f2.ps f1.ps -p1-5,6
```

Änderungen?

bisher:

```
public static void main(String[] args) {
  for (int i = 0; i < args.length; i++)
    System.out.print(args[i] + "\n");
}
```

Echo: Variante

bisher:

```
% java Echo -p1-5,6 f1.ps f2.ps
-p1-5,6
f1.ps
f2.ps
```

neu:

```
% java Rev -p1-5,6 f1.ps f2.ps
f2.ps f1.ps -p1-5,6
```

Änderungen?

```
public static void main(String[] args) {
   for (int i = args.length - 1; i >= 0; i--)
     System.out.print(args[i] + " ");
}
```