# Felder bzw. Arrays

Das Feld oder Array ist eine Datenstruktur,

die die in der Datei "Motivation\_Felder" aufgestellten Wünsche erfüllt!

Definition Felder bzw. Arrays:

Ein Feld ist eine Zusammenfassung einer beliebig großen Anzahl   
von Elementen gleichen Typs.

Die Elemente werden dabei automatisch benannt,

mit "Indexnummern" versehen.

Alle Elemente können dabei automatisiert durchlaufen werden.

Somit sind alle drei Forderungen erfüllt!

1. Zusammenfassung von Elementen, z.B. von 5 ints:

int

mein\_Feld:

1. Nummerierung:

Indexnummer: 0 1 2 3 4

Ein Feld bekommt einen Namen wie auch eine Variable,

die den Regeln für Bezeichner gehorscht.

Der Name spricht das Feld als Ganzes an,

vergleichbar mit einem Straßennamen.

Wie werden bei einer Straße die einzelnen Häuser angesprochen?

Durch Nachstellen der Nummer: Mozartstraße 17

Wie geschieht dies bei einem Feld?

Durch Angabe der Indexnummer in eckigen Klammern.

Eckige Klammern sind für Felder typisch.

Bei der Nummerierung ist darauf zu achten,

dass sie bei 0 u. nicht bei 1 beginnt.

Ein Feld der Größe 5 wie oben hat also Indizes: 0,1,2,3,4, keine "5"!

Wenn wir eine Element ansprechen,

können wir darauf lesend wie schreibend zugreifen:

mein\_Feld[3] = **17**; // schreibender Zugriff

int

mein\_Feld:

**17**

Indexnummer: 0 1 2 3 4

System.out.println (mein\_Feld[3]); // lesender Zugriff,   
// Ausgabe: 17

Weil es verwirrend ist, dass die Nummerierung mit 0 u. nicht mit 1 beginnt,

führe ich neben dem Index gerne die "Elementnummer" ein,

die von 1..Feldgröße läuft.

Dabei haben wir einen sog. "Offset":

Die Ele-Nr. ist jeweils um 1 größer als die Indexnummer:

int

mein\_Feld:

Indexnummer: 0 1 2 3 4

Elementnummer: 1 2 3 4 5

Wie können wir automatisiert lesen u. schreiben?

Sagen wir, wir wollen unser Feld füllen mit den Elementnummern.

Die Ausgabe soll sein:

An Indexposition 0 steht das Element Nr. 1

An Indexposition 1 steht das Element Nr. 2

An Indexposition 2 steht das Element Nr. 3

An Indexposition 3 steht das Element Nr. 4

An Indexposition 4 steht das Element Nr. 5

Wir durchlaufen ein Feld mit einer for-Schleife,

weil wir die Indexpositionen kennen: Hier z.B. 0..4

Den Index sprechen wir mit der Laufvariablen an:

for (int i=0; i<5; i++) // i:0..4

mein\_Feld[i] = i+1;

int

mein\_Feld:

57

47

31

21

11

Indexnummer: 0 1 2 3 4

Elementnummer: 1 2 3 4 5

Wie durchlaufe ich das Feld lesend,

um den oben aufgefürten gewünschten Output zu erhalten?

for (int i=0; i<5; i++) // i:0..4

System.out.println

("An Indexposition "+i+"steht das Element Nr. "+mein\_Feld[i]);

Wie deklariere ich ein Feld?

Welche Infos müssen in die Deklaration?

1. Zugrundeliegende Typ
2. Unterscheidung: Einzelnes Element des Typs oder ein Feld des Typs
3. Namen für das Feld
4. Größe des Feldes

Die Deklaration scheint auf den ersten Blick "etwas kryptisch" u. enthält Wiederholungen.

Wenn wir Objekte deklarieren, werden wir sehen, dass diese ähnlich deklariert werden:

Als erstes nenne ich,

wie auch bei der Deklaration eines einzelnen Elementes,

den zugrunde liegenden Typ:

int

Dann als nächstes unterscheide ich,

ob ich eine einzelne Variable deklarieren möchte oder ein ganzes Feld.

Würde ich als nächstes nur den Namen nennen,

würde ein einzelnes Element deklariert werden wie bei "int a;".

Wie kennzeichne ich,

dass ein Feld deklariert werden soll?

Oben haben wir gesagt,

dass eckige Klammern typisch sind für Felder.

Deshalb schreiben wir hier als Kennzeichnung dafür,

dass ein Feld u. nicht nur ein einzelnes Elemente deklariert werden soll,

hinter den Typ u. vor den Namen ein leeres eckiges Klammerpaar:

int[]

Danach geben wir den Namen an:

int[] mein\_Feld

Damit haben wir alle Infos untergebracht bis auf die Feldgröße.

Jetzt geht es "etwas kryptisch" weiter:

Wir schreiben: " = new", wiederholen den zu Grunde liegenden Typ

u. schreiben danach in eckige Klammern die Feldgröße.

Danach -wie immer nach einem Befehl bzw. einer Deklaration- ein Semikolon:

int[] mein\_Feld = new int[5];

Wir wollen das int-Feld von oben mit 5 Elementen deklarieren,   
das den Namen "mein\_Feld" tragen soll:

int[] mein\_Feld = new int[5];

Übergabe von Feldern an Methoden:

Felder können an Methoden übergeben werden wie einfache Variablen.

In der Parameterliste wird dann z.B. nicht wie für einen einfachen int geschrieben:

… m (int a) { … }

sondern für ein Feld aus ints:

… m (int**[]** feld) { … }

Die eckigen Klammern dienen hier,

wie schon bei der Deklaration, als Unterscheidung,

ob ein einzelner int oder ein int Feld gemeint ist.

Die Übergabe eines Feldes in einer Methode

läuft intern anders ab als bei einfachen Datentypen:

Bei einfachen Datentypen handelt es sich um einen "call by value",

d.h. es wird in der Methode eine Kopie der ursprünglichen Variablen erzeugt.

Die ursprüngliche Variable bleibt unverändert.

Bei Übergabe eines Feldes erfolgt diese als "call by reference",

d.h. es wird ein Verweis auf das Feld selbst übergeben

u. keine Kopie erzeugt.

Eine Kopie zu erzeugen wäre zu aufwändig.

<Feldname>.length:

Wenn ich mehrere Felder verschiedener Größe habe

u. diese auf gleiche weise belegen u. ausgeben möchte,

ist dies nicht problemlös möglich,

da in den Methoden die Feldgröße nicht bekannt ist.

Eine Möglichkeit wäre,

für verschiedene Feldgrößen verschiedene Methoden zu deklarieren.

Dies wäre aber wieder Codeduplizierung:

Die Methoden würden sich (neben dem Namen) nur an der Stelle unterscheiden,

wo die Obergrenze für den Laufindex steht.

Eine zweite Möglichkeit wäre,

die Feldgröße als Parameter zu übergeben.

Besser ist Folgendes:

Wird ein Feld deklariert,

wird dabei automatisch eine Konstante <Feldname>.lengt ins Leben gerufen,

die die Größe des Feldes enthält.

Sie ist überall vorhanden, wo auch auf das Feld selbst zugegriffen werden kann.

Man kann so eine Methode allgemein für beliebige Feldgrößen definieren,

wenn man sich nicht auf eine Konstante oder einen Parameter bezieht,

sondern prinzipiell auf <Feldname>.length.