E.Tutorial
www.et.ethz.ch

Programmieren mit Java Modul 3

Arrays

Theorieteil

Inhaltsverzeichnis

Begriffe

Datenstruktur	Array-Element	Array-Durchlauf
Array	Array-Dimension	
Array-Index	Array-Länge	Zweidimensionales Array

Autoren:

Lukas Fässler, Barbara Scheuner, David Sichau

E-Mail:

et@ethz.ch

Datum:

09 May 2023

Version: 1.1

Hash: 5f24db4

Trotz sorgfältiger Arbeit schleichen sich manchmal Fehler ein. Die Autoren sind Ihnen für Anregungen und Hinweise dankbar!

1 Modulübersicht

Mit den Standarddatentypen, die Sie bis hierhin kennen gelernt haben, kann fast jede beliebige Zahl oder jedes beliebige Zeichen dargestellt werden. Oft werden beim Programmieren aber **zusammengehörige Daten** verwendet (z.B. Lottozahlen, Temperaturen, Abfahrtszeiten). Eine Möglichkeit, eine zusammengehörige Gruppe von Elementen des gleichen Typs abzuspeichern, bieten **Arrays** (*Reihe*, *Felder*). Auf diese Weise muss nicht für jedes Element eine eigene Variable deklariert werden, sondern sie können alle unter einem Bezeichner gespeichert werden. Die Datenstruktur Array kommt in fast jeder modernen Programmiersprache vor.

2 Eindimensionale Arrays

Eindimensionale Arrays sind die einfachste Form von Arrays. Sie bestehen aus einer geordneten Menge von n Elementen desselben Datentyps. Die Elemente können über einen sogenannten **Index** angesprochen werden. Dieser gibt die Position eines Elements im Array an. In vielen Programmiersprachen (so auch in Java) hat das **erste Element** den **Index 0**, das zweite den Index 1 und das letzte den Index $n - 1$ (siehe Beispiel in Tabelle 1).

Index	0	1	2	3	4	5
Wert	12	13	15	17	23	32

Tabelle 1: Beispiel für ein eindimensionales Array mit sechs Elementen.

Folgende Operationen werden mit Arrays typischerweise ausgeführt: Array deklarieren, erzeugen, Werte in ein Array ein- und auslesen.

2.1 Arrays deklarieren

Arrays müssen wie Variablen zunächst deklariert werden. Das heisst, es werden Name und Datentyp festgelegt. Um anzuzeigen, dass nicht nur ein Element in der Variablen gespeichert werden kann, werden in Java **eckige Klammern** `[]` verwendet. Zur Zeit der Deklaration ist die Anzahl Elemente noch nicht festgelegt.

Schreibweise:

```
Datentyp[] name;
```

Beispiel: Folgende Anweisung deklariert ein Array mit dem Namen `zahlen` vom Datentyp Integer:

```
int[] zahlen;
```

2.2 Arrays erzeugen

Um ein Array zu erzeugen, wird der Operator `new` verwendet. Die Array-Länge (d.h. die Anzahl Elemente) wird in **eckigen Klammern** `[]` hinter den Datentyp geschrieben. Ist die Länge einmal festgelegt, kann sie nachher nicht mehr geändert werden. Die Länge des Arrays muss vom Datentyp Integer sein. Die Länge kann mit einer Zahl, einer Konstanten oder einem Ausdruck angegeben werden.

Schreibweise:

```
name = new Datentyp[anzahl];
```

Beispiel: Folgende Anweisung erzeugt sechs Speicherplätze im Array mit dem Namen `zahlen`, in welchem sechs Elemente von Typ `int` gespeichert werden können:

```
zahlen = new int[6];
// oder
zahlen = new int[4+2];
```

Bei der Erzeugung des Arrays werden die Elemente mit Default-Werten belegt. Beim Datentyp Integer ist dies der Wert 0.

Deklaration und Erzeugen von Arrays kann alternativ auch in einer einzigen Anweisung durchgeführt werden.

Schreibweise:

```
Datentyp[] name = new Datentyp[anzahl];
```

Beispiel: Folgende Anweisung erzeugt ein Array mit dem Namen `zahlen` vom Datentyp Integer mit sechs Elementen:

```
int[] zahlen = new int[6];
```

2.3 Arrays initialisieren

Einem Array-Element kann unter Angabe des Indexes ein Wert zugewiesen werden. In Java hat das erste Element den Index 0. Ein Array mit sechs Elementen hat somit die Indizes 0, 1, 2, 3, 4 und 5.

Schreibweise:

```
name[Index] = Wert;
```

Beispiel: So weisen wir dem ersten Element des Arrays den Wert 12 und dem zweiten den Wert 13 zu:

```
zahlen[0] = 12;
zahlen[1] = 13;
```

In Java können bei der Erzeugung des Arrays die Elemente auch direkt initialisiert werden, indem **geschweifte Klammern** `{}` gesetzt und die einzelnen Werte getrennt mit **Kommata** (`,`) eingegeben werden. Die Grösse des Arrays wird durch die Anzahl der Werte festgelegt. Der Operator `new` entfällt in diesem Fall.

Schreibweise:

```
Typ[] name = {wert1, wert2,..., wertN};
```

Beispiel: Deklaration, Erzeugung und Initialisierung des Arrays `zahlen` mit den sechs Elementen 12, 13, 15, 17, 22 und 32:

```
int[] zahlen = {12, 13, 15, 17, 22, 32};
```

2.4 Auf Array-Elemente zugreifen

Auf einzelne Elemente eines Arrays wird über einen **Index** (z.B. `i`) zugegriffen. `x[i]` liefert somit das Element aus dem Array `x` an der Position `i`. Es gilt zu beachten, dass die Indizes bei 0 beginnen und bei einem weniger als der Anzahl der Elemente des Arrays enden. Es können einzelne Elemente oder Bereiche von Arrays aufgerufen werden und es kann mit Elementen von Arrays gerechnet werden.

Beispiel:

```
// Array mit 3 Elementen.
int[] c = new int[3];

c[0] = 1;
c[1] = 2;
c[2] = 3;

//Aufruf eines Elements (Resultat: 1).
System.out.println(c[0]);

//Addition zweier Elemente (Resultat: 5).
System.out.println(c[1]+c[2]);
```

2.5 Array-Durchlauf mit Schleifen

Es ist üblich, zur Bearbeitung von Arrays **for-Schleifen** zu verwenden. Der Wert der Laufvariablen entspricht dabei dem Index-Wert des Arrays. Der Aufwand reduziert sich dadurch auf wenige Anweisungen, egal wie viele Elemente ein Array besitzt. Dieser Vorgang wird auch **Array-Durchlauf** genannt.

Beispiel: Mit folgender Anweisung können die sechs Elemente des Arrays `zahlen` am Bildschirm untereinander ausgegeben werden:

```
for (int i=0; i<6; i++) {
    System.out.println(zahlen[i]);
}
```

Die for-Schleife zählt von 0 bis 5. Bei jedem Schleifendurchlauf wird die Variable `i` als Index verwendet, um das Array Element an der entsprechenden Stelle auszugeben.

2.6 Länge eines Arrays bestimmen

Jedes Array hat in Java eine Eigenschaft `length`, mit welcher die Länge des Arrays abgefragt werden kann.

Beispiel:

```
for (int i=0; i<zahlen.length; i++) {
    System.out.println(zahlen[i]);
}
```

Das ist vor allem beim Durchlaufen des Arrays hilfreich, um die obere Grenze der Schleife auszurechnen. Dies hat den Vorteil, dass bei einer Änderung der Array-Länge die Schleife nicht angepasst werden muss.

3 Zwei- und mehrdimensionale Arrays

Besteht ein Element eines Arrays selbst wieder aus einem Array, entsteht ein **zweidimensionales Array**. Man kann es sich als Tabelle mit `m` mal `n` Elementen vorstellen, die jeweils über zwei Indizes angesprochen werden (siehe Beispiel in Tabelle 2).

Index	0	1	2	3	4	5
0	12	13	15	17	23	39
1	14	53	45	87	27	62
2	22	33	17	19	83	32

Tabelle 2: Beispiel für ein zweidimensionales Array mit drei mal sechs Elementen.

3.1 Initialisieren und Erzeugen eines zweidimensionalen Arrays

Schreibweise:

```
Typ[][] name = new Typ[anzahlZeilen][anzahlSpalten];
```

Beispiel: Folgende Anweisung erzeugt ein zweidimensionales Array mit dem Namen `zahlen` vom Datentyp Integer mit 3 mal 5 Elementen:

```
int[][]zahlen = new int[3][5];
```

3.2 Werte ins zweidimensionale Array ein- und auslesen

Um auf ein einzelnes Element eines zweidimensionalen Arrays zuzugreifen, werden die zwei Indizes für die Zeilen- und Spaltennummer angegeben:

```
name[zeilennummer][spaltennummer] = wert;
```

Beispiel:

```
zahlen[0][0] = 22;
```

Um zweidimensionale Arrays iterativ zu bearbeiten, sind **geschachtelte Schleifen** mit zwei Indexvariablen notwendig.

Beispiel: Folgende Zeilen geben alle Elemente des zweidimensionalen Arrays zahlen am Bildschirm aus:

```
for (int i=0; i<3; i++) {
    for (int j=0; j<6; i++) {
        System.out.println(zahlen[i][j]);
    }
}
```

3.3 Mehrdimensionale Arrays

Ein Array kann auch mehr als zwei Dimensionen haben. Für jede weitere Dimension wird ein weiterer Index für den Zugriff auf die Elemente benötigt.

4 Zeichenketten (Strings) als Arrays

Wie bereits in Modul 1 erwähnt, ist eine Variable des Types **String** eine Zusammenfassung mehrerer Variablen des Typs Character (`char`). In Java wird eine Zeichenkette als Array des Datentyps Character angelegt. Ein String kann somit auch aus einem Character-Array erzeugt werden. Auf die einzelnen Buchstaben im String kann somit auch wie beim Array über den Index zugegriffen werden.

Beispiel:

```
char[] meinText = {'d','e','r',' ','T','e','x','t'};
String meinString = new String(meinText);
char erstesZeichen= meinString.charAt(0);
```

Bei diesem Beispiel wird als erstes ein char-Array der Länge 8 erzeugt und direkt mit acht Zeichen initialisiert. Auf Basis dieses Arrays wird dann ein String erzeugt, der den Text „der Text“ enthält. Aus diesem String wird anschliessend das erste Zeichen ausgelesen. Will man alle Zeichen eines Textes auf diese Weise einzeln auslesen, kann man eine Schleife einsetzen:

```
for (int i=0; i<meinString.length(); i++){
    System.out.print(meinString.charAt(i));
}
```