Arrays

In Java werden Datenstrukturen, die null oder mehr Elemente enthalten können, als Kollektionen (Collections) bezeichnet. Ein Array ist eine Kollektion (Collection), die eine **feste Größe** hat und deren `Elemente alle vom gleichen Typ` sein müssen. Elemente können einem Array zugewiesen oder aus ihm mithilfe eines **Index** abgerufen werden. Java-Arrays verwenden eine nullbasierte Indizierung: Der Index des ersten Elements ist 0, der des zweiten Elements ist 1, usw.

Hier ist die vollständige Syntax zur Initialisierung eines Arrays mit dem Namen `arrayName`:

```
type[] arrayName = new type[size];
```

`type` legt den Datentyp der Elemente im Array fest, das kann ein primitiver Datentyp (z.B. int) oder eine Klasse (z.B. String) sein – auch Datentypen, die aus eigenen Klassen abgeleitet sind, sind selbstverständlich erlaubt. Die beiden eckigen Klammern nach dem Typ (`[]`) legen fest, dass hier ein Array definiert werden soll.

`size` ist die Größe des Arrays - das ist die Anzahl der Elemente, die dieses Array aufnehmen kann. Die Größe kann später nicht mehr geändert werden. Nach der Erstellung des Arrays werden die Elemente auf ihre Standardwerte initialisiert (typischerweise 0, false oder null).

Beispiele:

- `int[] zahlen = new int[22];` - Ein Array mit dem Variablen-Namen `zahlen`, das 22 Integer Zahlen speichern kann.
- `double[] kommazahlen = new double[8];` - Ein Array mit dem Variablen-Namen `kommazahlen`, das 8 Zahlen vom Typ "double" speichern kann.
- `String[] saetze = new String[5];` - Ein Array mit dem Namen `saetze`, das 5 Zeichenketten speichern kann.
- `Auto[] fuhrpark = new Auto[200];` - Ein Array mit dem Namen `fuhrpark`, das 200 Objekte des Typs `Auto` speichern kann. Die Klasse `Auto` muss man natürlich selbst definieren, damit das funktioniert.

Arrays können auch mithilfe einer **Kurznotation** definiert werden, die es ermöglicht, das Array gleichzeitig zu erstellen und zu initialisieren:

```
// Die beiden folgenden Zeilen machen dasselbe:
// Sie deklarieren ein Array, das ganze Zahlen speichern kann
// und initialisieren die Element direkt als 4, 9 und 7
int[] dreiZahlenV1 = new int[] { 4, 9, 7 };
int[] dreiZahlenV2 = { 4, 9, 7 };
```

Die Länge des Arrays wird durch die Initialisierung mit drei Elementen auf 3 festgelegt und kann, wie immer bei Arrays, anschließend nicht mehr geändert werden.

```
int[] zahlenarray = {11,32,42,2,4};
```

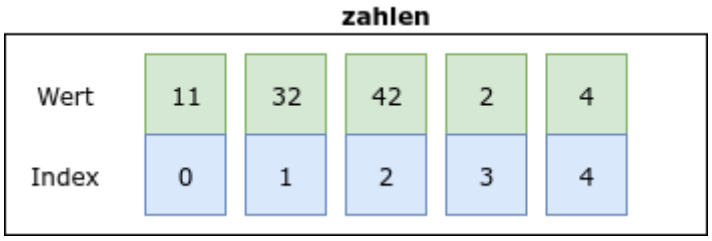
Ein Array kann man als Instanzen einer spezielle Klasse verstehen. Array werden also als Objekte behandelt und müssen durch den Operator new instanziiert werden. Die Array-Klasse bringt spezielle Methoden und Operationen mit, auf die man beim Umgang mit Arrays zurückgreifen kann, z.B liefert die Methode `length`, die Länge des Arrays zurück:

```
zahlen.length // in unserem Fall: 5
```

1)

Zugriff auf Array-Elemente

Jedes Element eines Array hat einen Wert und einen Index. **Die Zählung des Index beginnt immer bei Null**. Für unser Beispiel-Array sieht das also folgendermaßen aus:



Über den Index eines Elements, kann man auf dessen Wert zugreifen:

```
zahlen[1] // hier: 32
zahlen[4] // hier: 4
```

Mit einer entsprechenden Wertzuweisung kann auf diese Art auch der Wert eines Arrayelements gesetzt werden:

```
zahlen[1] = 33;
zahlen[4] = 78;
```

(A1)

Passe das folgende BlueJ-Projekt (arrays01) so an, dass es das oben definierte Array `zahlenarray` als Instanzvariable besitzt. Die Werte sollen dabei auch übernommen werden. Lass dir die Länge des Arrays im Konstruktor bei der Erzeugung eines Array Objekts auf der Konsole ausgeben.

(A2)

Erweitere das Programm aus (A1) so, dass für alle Elemente des Arrays eine Zeile wie die folgende ausgegeben wird:

```
Das Array-Element mit dem Index 0 hat den Wert 11
Das Array-Element mit dem Index 1 ...
Das Array-Element mit dem Index 2 ...
```

Verwende dazu eine Zählschleife (for-Schleife). Welches ist der Startwert, welches der Endwert, den diese Schleife haben muss?

(A3)

Schreibe eine Methode, die den Wert des dritten Elements des Arrays ausgibt.

(A4)

Verändere im Schleifenkopf der for-Schleife aus Aufgabe 2 den Vergleichsoperator, indem du ihn umkehrst (> statt <). Erläutere die Ausgabe.

(A5)

Erzeuge ein Array vom Datentyp String, das die Wochentage enthält. Lasse ebenfalls direkt bei der Instanziierung auf der Konsole die Länge des Arrays ausgeben. Beim Aufrufen von printArray() sollen nun auch die Wochentage ausgegeben werden.

arrays.odp	96.8 KiB	08.11.2022 14:39
arrays.pdf	100.8 KiB	08.11.2022 14:40
robin-6906521_640.jpg	27.9 KiB	09.10.2024 18:36

1)

Java verfügt auch über Klassen, mit denen man Objektlisten und Arrays effizienter verwalten kann, fürs Erste begnügen wir uns mal mit dem Weg "zu Fuß"

From:
https://www.info-bw.de/ -

Permanent link:
https://www.info-bw.de/faecher:informatik:oberstufe:java:algorithmen:arrays:definition:start?rev=1728486026

Last update: **09.10.2024 15:00**