II.1.5 Arrays

Freitag, 4. November 2016 16:00

Ziel: Repräsentiere eine Folge von Werten mit nur einem Objekt.

Array hat Index, beginnt mit 0.

Array kann mehrdimensional sein.

```
int summe = 0;
for (int i=0; i<=2; i++) {
    summe += bestand[i][3];
}
```

Berechne die Anzahlen der Artikel 0-2 an Ort 3

Ergibt 7+0+1=8

- Array fasst Vielzahl von Variablen des gleichen Typs zusammen.
- Einzelne Elemente werden durch ganzzahligen Index unterschieden.
- Auch Array-Variablen müssen vor Verwendung deklariert werden. Dabei müssen 2 Dinge festgelegt werden:
  - Datentyp der Elemente im Array
  - Dimension des Arrays

z.B.

```
int[] folge;
int[][] bestand;
double[] x;
```

bedeutet: Dimension 2

- Bei primitiven Datentypen (wie int) wird der Wert

der Variablen x an den "Speicherplatz x" geschrieben.

- Bei Arrays:
  Bei Variablendeklaration (int[] x;) wird der Array-Variablen noch kein Speicherplatz für die Array-Elemente zugewiesen.
  Bei der Erzeugung eines neuen Arrays
  (x = new int[3];)
  wird dieser Speicherplatz für die Array-Elemente erzeugt und an die Speicherstelle x wird ein Verweis / eine Referenz auf diesen Speicherplatz geschrieben.
- In Java: OS Variablen Werte oder Referenzen speichern, liegt am Datentyp: Primitive Datentypen werden als Werte gespeichert, alle anderen Datentypen als Referenzen.
- Vorteil von Arrays: Wahlfreier Zugriff, d.h. man kann auf jedes Array-Element gleich schnell zugreifen, weil man die Adresse jedes Array-Elements sofort aus der Startadresse u. der Größe (des Datentyps) der Array-Elemente berechnen kann.
- Durch Referenzvariablen hat man Seiteneffekte:
  Auswirkung des Schreibzugriffs über eine Referenzvariable y auf ein Objekt, das auch über eine andere Referenzvar. x erreichbar ist.

- Um eine wirkliche Kopie zu erzeugen, muss man elementweise kopieren.
- Zwei Arrays sind nur gleich (bzgl. ==), falls ihre Variablen auf das identische Array zeigen.

```
y == x        true
```

```
int [] z = new int[3];
z[0] = 14;
z[1] = 8;
z[2] = 5;
```

```
z == x        false
```

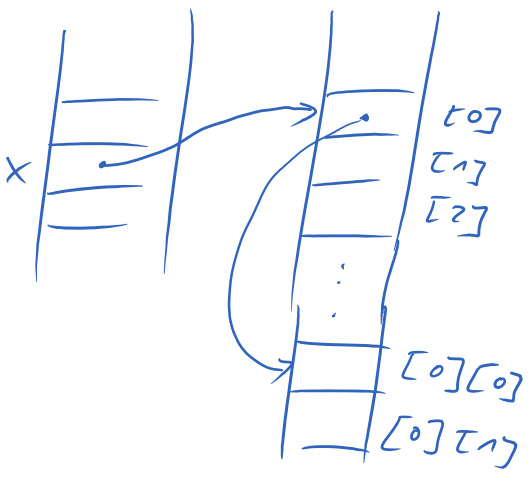
- Einträge im Heap, die nicht mehr erreichbar sind, werden automatisch vom sogenannten Garbage Collector gelöscht.
- `int []` ist ein Typ.

```
int [] y;
```
← Man kann y bel. 1-dimensionale int-Arrays (beliebiger Länge) zuweisen.

```
y = x;
```
← Möglich, falls es schon int-Array x gibt.

- Folgendes ist möglich:

```
int [] [] x;
x = new int [3] [];
x[0] = new int [2];
x[1] = new int [5];
x[2] = new int [3];
```



```
x[2] = new int[3];
```

oder:

```
int[][] x;
x = new int[3][2];
```

- Als Kurzschreibweise darf man direkt bei der Variablen-deklaration schon die Array-Elemente mit angeben:

```
int[] x = {14, 2, 7};
int[][] y = { {14,2}, {7,5,3}, {0} };
```

- Für jedes Array x liefert x.length seine Länge (Attribut des Array-Objekts x).

oben: x.length == 7

y.length == 3

y[0].length == 2

- Palindrom: Wort von links gelesen = Wort von rechts gelesen, z.B. "reutner"
Aber nicht "rentier"
- Eingabe von "main"-Methode ist String-Array,
d.h. Aufruf

```
java Palindrom reutner hallo ....
```

args[0] args[1]

- Jeder String kann mit der Methode toCharArray in ein char-Array umgewandelt werden.

- 0 1 2 3 4 5 6
  r e n t n e r length = 7 i läuft von 0 bis (7-2)/2 = 2
  0 1 2 3 4 5
  n e f f e n length = 6 i läuft von 0 bis (6-2)/2 = 2

Weiteres Bsp: Sort

- Laufvar. i durchläuft alle Array-Elemente bis auf letztes. Für jedes Element a[i] wird geprüft, ob einer seiner Nachfolger im Array kleiner. Falls ja, wird a[i] mit diesem Nachfolger getauscht. Nach der inneren Schleife steht an Pos. i das kleinste Element von Stelle i bis zum Ende d. Arrays.
- Vertausche a[i] und a[j]: Wozu braucht man?

```
a[i] = a[j];
a[j] = a[i];
```

funktioniert nicht, danach hätte man den gleichen Wert an a[i] und a[j].

- Bsp: { 4, 2, 5, 1 } ⇒ { 2, 4, 5, 1 } ⇒ { 1, 4, 5, 2 }
  (i, j) (i, j) (i, j)

  ⇒ { 1, 2, 5, 4 } ⇒ { 1, 2, 4, 5 }
  (i, j)

Java hat eine eigene Variante der for-Schleife,

um Sammlungen von Werten (wie Arrays) zu durchlaufen:

foreach-Schleife

foreach-Schleife eignet sich gut, falls alle Elemente des Arrays gelesen werden, aber nicht, wenn sie geschrieben werden.
Die Änderung der Eingabe-Schleife von "Satz" in eine foreach-Schleife hätte zur Folge, dass die vom Benutzer eingegebenen Werte nicht ins Array geschr. werden.