II.1.5 Arrays

Mittwoch, 24. Oktober 2018 09:00

Ziel: Behandlung einer Sammlung von Werten (z.B. Listen, Tabellen, ...)
Zugriff auf Werte in der Sammlung über Index (beginnt mit 0).
Arrays können mehrdimensional sein.

- Array-Variable fasst mehrere Werte des gleichen Datentyps zusammen.
- Zugriff über Index:

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

berechnet Bestand der Artikel 0,1,2 an Ort 3, d.h. summe = 7+0+1 = 8

- Deklaration von Array-Variablen:

```
int[] folge;
```

Datentyp der 1-dimensionalen

Arrays mit int-Werten

```
int[][] bestand;
```

Datentyp der 2-dimensionalen int-Arrays

```
double[] xs;
```

⋮

Variablendeklaration reserviert benötigten Platz im Speicher.

Java behandelt primitive Datentypen (int, double, boolean, char...) anders als alle anderen Datentypen (String, Arrays, etc.)

Wertvariablen:

`int x;` Speicherplatz x der benötigten Größe (4 Byte) wird reserviert. (auf Stack)

`x=10;` Wert 10 wird an Speicherplatz x geschrieben.

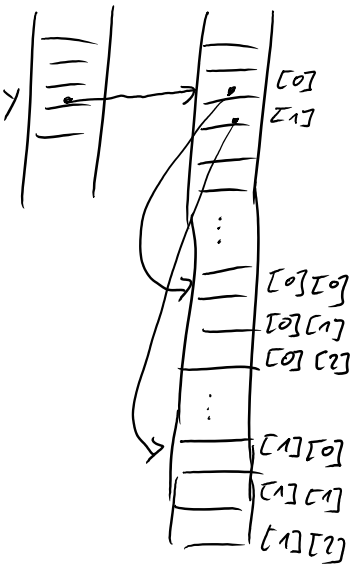
Referenzvariablen

`int[] x;` Speicherplatz x

wird bereitgestellt (auf dem Stack)

`x = new int[3];` Speicherplatz auf dem Heap wird für 3 int-Werte bereitgestellt. x enthält die Adresse dieses Heap-Speicherplatzes.

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



`int[][] z;`
`z = new int[2][];`
`z[0] = new int[5];`
`z[1] = new int[7];`

Stack (Kellerspeicher):
Wie in einem Keller werden (beim Betreten von Blöcken oder beim Methodenaufruf) immer neue Dinge hineingetan.

Man kommt nur an die Dinge, die zuletzt hinzugefügt wurden ("Last in-first out").

Heap (Haldenspeicher): "ungeordneter Haufen"

Vorteil von Arrays:
Wahlfreier Zugriff, d.h. man kann auf jedes Array-Element gleich schnell zugreifen, weil man direkt die Adresse ausrechnen kann, an der es gespeichert ist.

Nachteil: Größe des Arrays liegt fest, sobald es mit new erzeugt wurde.

In Java:
- Kein direkter Zugriff auf Adressen
- Kein Reservieren + Freigeben von Speicherbereichen durch Programmierer.
- Keine expliziten Pointer: OS Variable weist oder

Referenz/Adresse des Werts
speichert, hängt am Datentyp.

Zuweisung $y = x$

bedeutet: Kopiere den Inhalt
der Speicherzelle x (auf Stack)
in die Speicherzelle y (auf Stack).

Wenn x und y Array-Variablen
sind, zeigen sie hinterher
auf das gleiche Array.

Seiteneffekt: Änderung
des Arrays über y be-
wirkt Änderung des Arrays x.

Garbage Collector:

löscht nicht mehr benötigte
Werte auf dem Heap
(wird immer wieder auto-
matisch ausgeführt!)

Nach der Deklaration einer
Referenzvariable hat diese
zunächst den Wert null.

→ "Zeiger ins Leere"

BlueJ-Entwicklungs-

BlueJ - Entwicklungsumgebung

- geeignet für Anfänger (später Eclipse)
- IDE (integrated development environment)

Gleichheit von Arrays:

```
int [] x, y;
x = new int [2];
y = new int [2];

x[0] = 5;
y[0] = 5;
x[1] = 7;
y[1] = 7;
```

Gilt dann x == y ? Nein.

```
x = y;
```

Danach gilt tatsächlich x == y.

Zum inhaltlichen Vergleich von Objekten sollte man nicht == benutzen.

```
int[][] a = new int[3][2];
int[][] b = new int[3][];
```

Kurzschreibweise: (nur zulässig bei Initialisierung eines neuen Arrays)

```
int[] x = {2,3,4};
```

statt

```
int[] x = new int[3];
x[0] = 2;
x[1] = 3;
x[2] = 4;
```

← x.length ist 3

↑ Eigenschaft des Array-Objekts x

```
int[][] a;
:
a = new int[][] { {2,3}, {}, {4,5,6} };
```

a.length ist 3

a[0].length ist 2

a[1].length ist 0

Bsp: Palindrom

soll herausfinden, ob ein Wort ein Palindrom ist (d.h. von links und rechts gelesen gleich).

LEGOVOGEL ✓

RENTNER ✓

RELIEFPFEILER ✓

RELIEFPFEILER ✓

RENTIER X

- main bekommt String[] args als Eingabe

Aufruf:

java Palindrom string0 string1 ...

⇒ args wird ein Array zugewiesen mit

args[0] = string0

args[1] = string1

⋮

z.B.: java Palindrom legovogel

- toCharArray ist eine Methode, um aus einem String das entsprechende char-Array zu bekommen.
- Eigenschaften:
  a.length ← Attribut des Array-Objekts a
  s.toCharArray() ← Methode des String-Objekts s (Sieht man an (...))
- for-Schleife: Durchlaufe Wort von links und rechts bis zur Mitte und überprüfe, ob die

jeweiligen Zeiden gleich sind:

0 1 2 3 4 5 6 7 8 wort.length=9

l e g o v o g e l

↑ ↑ $\left\lfloor \frac{w.length-2}{2} \right\rfloor = 3$

Bsp: Sortieren

- Äußere Schleife: Schreibe das kleinste verbliebene Element an die Stelle i.
- Innere Schleife: Vergleiche a[i] mit allen Nachfolgern. Wenn Nachfolger kleiner als a[i], dann vertausche die Elemente.

Bsp: a = {4, 2, 5, 1}

i=0 ↑ ↑

j=1 i j

Vertausche 4 und 2:

{2, 4, 5, 1}

i=0 ↑ ↑

j=3 i j

Vertausche 2 und 1

{1, 4, 5, 2}

i=1 ↑ ↑

j=3 i j

Vertausche 4 und 2

{1, 2, 5, 4}

i=2 ↑

{1, 2, 5, 4}

i=2 ↑ ↑

j=3 i j

Vertausche 5 und 4

{1, 2, 4, 5}

Oft hat man Schleifen, die alle Elemente eines Arrays durchlaufen.

⇒ foreach-Schleife

Eignet sich nur zum lesen, nicht zum Schreiben des Arrays.

bedeutet:

```
X = a[i];
X = SimpleIO.getInt(...);
```

⇒ ändert nichts an a[i]

Vertauschung von a[i] und a[j]:

Warum nicht

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

Weil man dann den ursprünglichen Eintrag von a[i] im 1. Schritt überschreibt.