II.1.5 Arrays

Dienstag, 24. Oktober 2017 09:00

Arrays speichern viele zusammengehörige Werte des gleichen Typs.
ZB: folge besteht 4 int-Zahlen
Zugriff über Index des Eintrags,
erster Eintrag hat Index 0.
(folge[2] ist der 3. Eintrag,
d.h. 0)

- Mehrdimensionale Arrays möglich
t.B. bestand [i] [j] gibt an, wieviele Exemplare von Artikel i an Ort j vorrätig sind.
- Im Index können bel. Ausdrücke auftreten:

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

berechnet die Summe d. Artikel 0,1,2 an Ort 3 (d.h. 7+0+1 = 8).

- Jede Array-Var. muss vor Verwendung deklariert werden.

Dabei muss man festlegen:

- Typ der Elemente des Arrays
- Dimension d. Arrays (Anzahl der [ ])

z.B. `int [] x;`

`float [][] y;`

Datentyp der 2-dimensionalen Arrays mit float-Elementen.

- Deklaration einer Variable:

Java reserviert Speicherplatz für die Variable. Datentyp bestimmt, wie groß der Speicher-

platz ist.

- Bei primitiven Datentypen:
  Variable - Speicherplatz enthält den entsprechenden Wert.
- Bei nicht-primitiven Datentypen (wie Arrays, Strings etc.):
  Variable enthält die Speicher-adresse, wo der Wert zu finden ist. D.h: Variable enthält eine Referenz / einen Verweis auf den Wert.
- Bei Dekl. `int[] x` wird Speicherplatz für `x` reserviert, in den später die Adresse des Arrays geschrieben werden kann. Z.B. gilt `x = null` ("Zeiger ins Leere")
- Erzeugung des Speicherplatzes

für die Array-Einträge durch "new".

new int [3]

legt Speicherplatz für 3 int-Array Einträge an ← auf dem Haldenspeicher (Heap)

Der Speicherbereich für die Variablen heißt Kellerspeicher (Stack)

- In Java hat man keinen direkten Zugriff auf die Adressen im Speicher
- Wahlfreier Zugriff: Man kann aus der Adresse des ersten Array-Elements, dem Index und dem Datentyp der Array Elemente sofort die Adresse jedes Array-Elements berechnen. Auf jedes Array-Element kann daher gleich

schnell zugegriffen werden.

- Zuweisung $y = x$
bedeutet: Schreibe den Inhalt der Speicherzelle x in die Speicherzelle y.
In Java hängt es vom Datentyp ab, ob in Variable der Wert oder die Adresse des Werts gespeichert ist.
Dh: Bei prim. Datentypen wird der Wert kopiert. Bei anderen Datentypen zeigen x und y anschließend auf das gleiche Objekt.
⇒ Änderung über Var. y bewirkt dann auch Änderung, die v. x aus sichtbar ist.
Seiteneffekt

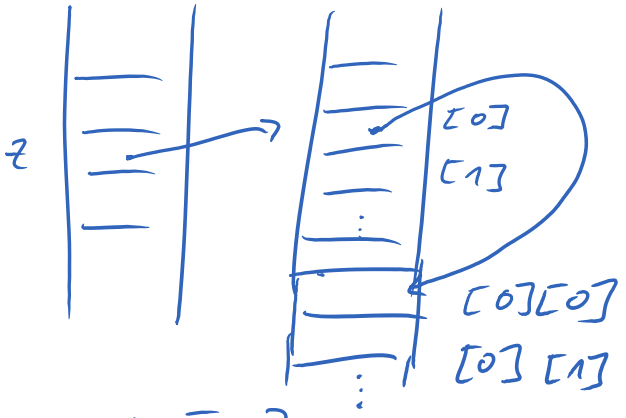
- Es kann Einträge auf dem

Heap gelöscht, die vom Stack aus nicht mehr erreichbar sind.
Garbage Collector sucht immer wieder nach solchen Einträgen u. gibt den entsprechenden Heap-Speicher wieder frei.

- Lässt sich gut veranschaulichen mit BlueJ.
Integrierte Entwicklungsumgebung (Integrated Development Environment, IDE) für Java-Anfänger. Professioneller: Eclipse.
- Initialisierung für mehrdimensionale Arrays:

```
int [][] y, z;
```

```
Y = new int [2] [3];
z = new int [2] [ ];
```




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

Zur Initialisierung von Arrays ex. eine Kurzschreibweise:

```
int [ ] a = { 8, 7, 9, 6 };
```

↑ ist Kurzschreibweise für:

```
int [ ] a = new int [4];
a[0] = 8;
   :
a[3] = 6;
```

Auch für mehrdim. Arrays:

```
int[][] b = { {0,1}, {}, {2} }
```

b[0][1]=1 b[2][0]=2

Jedes Array besitzt eine Länge. Dies ist eine Eigenschaft (Attribut) des Arrays.

a.length ergibt 4

b.length ergibt 3

(Länge in der 1. Dimension)

b[0].length ergibt 2

Bspe für Array-Algorithmen

1. Bsp: palindrom

Palindrom: Wort, das von links und von rechts gleich ist.

z.B: rentner ✓

rentier ✗

Reliefpfeiler ✓

main Methode bekommt ein String-Array als Eingabe.

Aufruf der main-Methode der Klasse Palindrom durch:

java Palindrom string0 ... stringn

Dadurch wird das Eingabe-Array args mit dem Array {string0, ..., stringn} belegt.

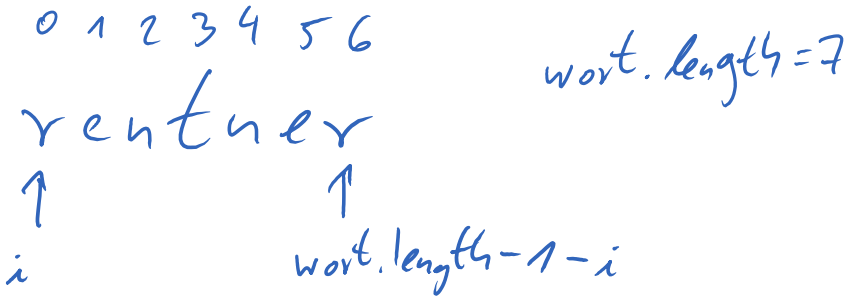
Beim Aufruf

java Palindrom reuther

ist args[0] der String "reuther".

Objekte können Eigenschaften haben (z.B. length ist Attribut-Eigenschaft v. Arrays).

Strings haben z.B. die Eigenschaft toCharArray() (Diese Eigenschaft muss berechnet werden, ist eine Methode von Strings — erkennt man an (..).)



Zeiger laufen bis zur Mitte: $\left\lfloor \frac{wort.length-1}{2} \right\rfloor = \left\lfloor \frac{5}{2} \right\rfloor = 2$

2. Bsp Sortieren v. Arrays

Laufvar. i durchläuft alle Array-Var. bis auf letztes.

Innere Schleife: überprüft, ob a[i] größer ist als einer seiner Nachfolger. Falls ja, dann

werden die Elemente getauscht.

⇒ Zum Schluss der inneren Schleife steht an Pos. i das kleinste der Elemente von i bis zum Schluss des Arrays.

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

Warum nicht:

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

← schlecht, denn so haben hinterher a[i] u. a[j] beide den alten Wert v. a[j].

Bsp: Sortiere {4, 2, 5, 1},

i=0 j=1 : tausche Elemente

{2, 4, 5, 1}

j=3 : tausche Elemente

{1, 4, 5, 2}

i=1 j=3: tausche Elemente

{1, 2, 5, 4}

i=2 j=3: tausche Elemente

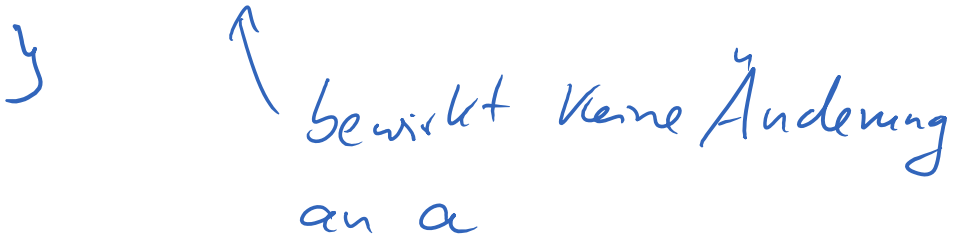
{1, 2, 4, 5}

Man hat oft Schleifen, in denen alle Array-Elemente durchlaufen werden. Dies lässt sich kürzer als foreach-Schleife schreiben.

Bsp: Vereinfache Sort-Prog. durch foreach-Schleifen.

Aber: eignet sich nur für Lese-Operationen, nicht für Schreib-operationen.

```
for (int x : a) {
    x = t;
```


bewirkt keine Änderung
an a