Javakurs für Anfänger

Einheit 07: Arrays

Kyrill Schmid
Lehrstuhl für Mobile und Verteilte Systeme

Heutige Agenda

1. Teil: Arrays

- Motivation und Eigenschaften
- Deklaration, Erzeugung und Initialisierung
- Array-Elemente durchlaufen
- Arrays anwenden
- Mehrdimensionale Arrays

2. Teil: Übungen

- Praktische Übungen zum Umgang mit Arrays

Lernziele

- Die Struktur von Arrays kennenlernen und verstehen
- Arrays anwenden können
- Den Umgang mit Arrays programmatisch einüben

Arrays: Motivation

Separates Speichern mehrerer Variablen des **gleichen Typs** wäre äußerst umständlich

- Beispiel:

```
// Es sollen gemessene Temperaturen der letzten 10 Tage gespeichert werden

double temperaturTag0 = 23.3;
double temperaturTag1 = 22.5;
double temperaturTag2 = 20.9;
double temperaturTag3 = 17.7;
double temperaturTag4 = 16.6;
double temperaturTag5 = 16.3;
double temperaturTag6 = 15.2;
double temperaturTag7 = 15.7;
double temperaturTag8 = 10.1;
double temperaturTag9 =  9.5;
```

Dafür bräuchten wir 10 Variablen vom Typ double => sehr aufwändig!

- Diesen Umstand wollen wir mit einer anderen Speicherstruktur lösen….

Arrays: Einführung

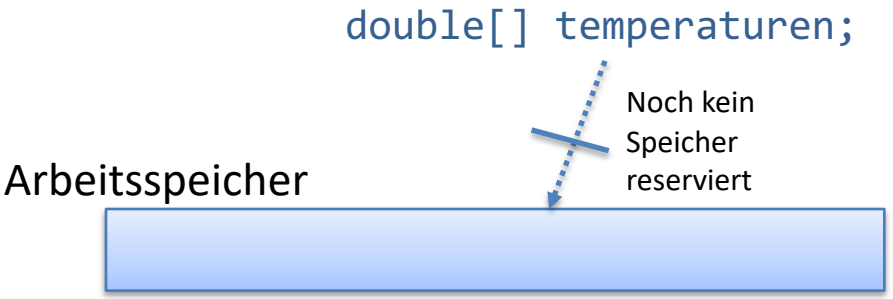
Arrays

- Zählen zu den sog. Containerklassen
  - Definition von Arrays folgt der üblichen Definition von Objekten
- Gehören zur Java-Standardbibliothek
  - Stehen somit ohne Einbindung weiterer Bibliotheken zur Verfügung
- Sind ein Verweis auf eine Folge von Variablen, der sog. Fächer.
- In einer als Array definierten Variablen können **mehrere Werte des gleichen Typs** gespeichert werden
  - Primitive Datentypen oder Referenztypen (Objekte) möglich
  - Die gespeicherten Werte können jederzeit verändert werden
- Die Anzahl der Elemente eines Arrays sind nach dessen Definition nicht mehr veränderbar!
  - Statische Datenstruktur mit fester Länge!

Array Definition

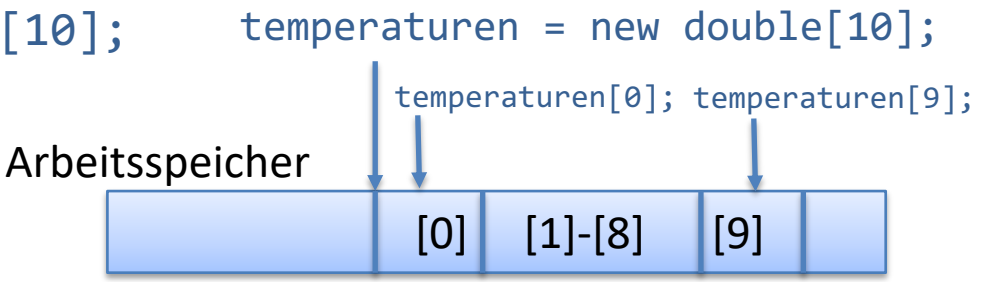
Deklaration: analog zu Variablen nur mit []

- `datentyp[] name;`
- Bsp.: `double[] temperaturen;`



Erzeugung: analog zu Objekten, da Containerklasse!

- `name = new datentyp[int Array_länge];`
- Bsp.: `temperaturen = new double[10];`



Auch in einem Schritt möglich:

- `datentyp[] name = new datentyp[int Array_länge];`
- Bsp.: `double[] temperaturen = new double[10];`

Initialisierung

Belegung der Werte:

- `name[int index] = wert;`
- Bsp.:
  - `temperaturen[0] = 23.3;`
  - `temperaturen[9] = 9.5;`
- `int index` liegt zwischen `0` und `Array_länge -1`
- Falls auf ein Element außerhalb des Arrays zugegriffen wird, erscheint Fehlermeldung: `java.lang.ArrayIndexOutOfBoundsException: index`
- Bsp.:

```
double[] temp = new double[5];

temp[0] = 3.4;
temp[5] = 1.0; //Fehler! Unerlaubter Index also:

→ java.lang.ArrayIndexOutOfBounds: 5
```

Deklaration und Initialisierung

Bei der Deklaration gleich die Werte belegen:

- `datentyp[] name = {wert1, wert2, …, wertN};`
- Bsp.:
  - `int[] tmp = {0,1,2,3,4,5,6,7,8,9};`
  - `Hund[] meine_Hunde = {bello,leo,justin};`

```
// Die Hund-Objekte müssen natürlich zuvor auch erzeugt worden sein!
```

Auf Werte eines Arrays zugreifen:

- `name[int index];`
- Bsp.:
  - `int index0 = tmp[0];`
  - `Hund bello = meine_Hunde[0];`

Array-Elemente durchlaufen

Wie können alle Elemente eines Arrays ausgegeben werden, ohne den erlaubten Bereich zu verlassen?

- Jedes Array besitzt festes Attribut `.length`
  - Beinhaltet die Länge des Arrays
  - Damit können Schleifen programmiert werden, die den Bereich nicht verlassen

- Beispiele zum Durchlaufen von Arrays mit verschiedenen Schleifen:

```
double[] temperaturen = {23.5, 22.5, 20.9, 17.7, 16.6, 16.3, 15.2, 15.7,
10.1, 9.5};

// mit for-Schleife:

for(int i=0; i<temperaturen.length; i++){

  System.out.println(„Temperatur am Tag “+i+“ liegt bei “+temperaturen[i]+“
  Grad Celsius.“);
}
```

Array-Elemente durchlaufen


```
// Mit while-Schleife:

int i = 0;
while (i<temperaturen.length){

   System.out.println(„Temperatur am Tag “+i+“ liegt bei “+temperaturen[i]+“
   Grad Celsius.“);

   i++;
}

// Oder mit foreach-Schleife:

int index = 0;
for(double temp : temperaturen){

   System.out.println(„Temperatur am Tag “+index+“ liegt bei “+temp+“  Grad
   Celsius.“);

   index++;
}
```

Anmerkungen zur `foreach`-Schleife

`foreach`-Schleife ist eine Kurzform der `for`-Schleife zur Abarbeitung aller Elemente eines Arrays

- Es gelten aber Einschränkungen zur `for`-Schleife:
  - Array wird immer vom ersten bis zum letzten Element durchlaufen
    - Reihenfolge ist unveränderbar
    - Elemente können nicht übersprungen werden
  - Elemente werden nur gelesen!
    - Kein schreibender Zugriff auf die Array-Elemente möglich, da auf einer Kopie gearbeitet wird
    - Aber: Verändern von referenzierten Werten durchaus möglich

```
// for Schleife:
for (int i =0; i<mein_array.lenght; i++){
   System.out.println(„Mein Array-Element: “+mein_array[i]);
}

// analog: foreach-Schleife
for (datentyp iter : mein_array){
   System.out.println(„Mein Array-Element: “+iter);
}
```

Arrays als Parameter

Eine als Array deklarierte Variable kann auch als Methoden-Parameter übergeben werden

```
// Beispiel: Methode mit Array als Parameter
// Gibt den Mittelwert der Temperaturen zurück

public double getMittelwert(double[] temperaturen){

   if (temperaturen.length == 0)
      return 0.0;
   double summe = 0;
   for (int i = 0; i < temperaturen.length; i++)
      summe = summe + temperaturen[i];

   return summe / temperaturen.length;
}
```

- Aufruf über `getMittelwert(temperaturen);`
- Liefert den Mittelwert als `double` zurück

Arrays als Rückgabewerte

Genauso kann ein Array-Typ durch eine Methode zurückgegeben werden

Und Arrays können auch Instanzvariablen (Attribute) sein

```
// Beispiel: Getter-Methode liefert Array vom Typ double zurück

private double[] temperaturen;

//…

public double[] getTemperaturen(){

   return this.temperaturen;
}
```

Arrays von Objekten

Ein Array kann auch Referenzdatentypen enthalten.

- Bsp.: `Auto[] meine_Autos = {fiat,mercedes,golf};`
- Nun kann über alle Autos iteriert werden, um so ggf. Manipulationen vorzunehmen:

```
// Beispiel: Setzt den Preis eines jeden Autos im Array auf 1000 Euro:

for(Auto auto : meine_Autos){
   auto.setPreis(1000);
}
```

Mehrdimensionale Arrays

Die Elemente von Arrays können wiederum Arrays sein!

- Verschachtelte Arrays
- Können in beliebigen Dimensionen vorliegen
  - Bsp.: 2-Dimensionales Array:
    - `double [][] matrix = new double[10][20]; // Beispiel für eine 10x20 Matrix`
    - `double [][] meine_matrix = {{1,2},{3,4},{5,6}};`
  - Bsp.: 3-Dimensionales Array:
    - `int[][][] mein_array = new int[25][10][4];`

- Die Array-Länge der inneren Arrays kann dabei variieren
  - Bsp.:

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

So entsteht ein dreieckiges Array mit 3 unterschiedlich langen Array-Elementen

Beispiel 2-Dimensionales Array

Deklaration, Erzeugung und Initialisierung ähnlich zu ein-dimensionalen Arrays!

- Aber:
  - Man muss nur die Länge des äußeren Arrays zwingend angeben!
    - `int[][] a = new int[10][]; // Damit ist die Länge des inneren Arrays flexibel`
  - Man kann auch beide Längen angeben
    - `int[][] b = new int[10][12]; // Dann ist die Länge des inneren Arrays fest!`

Das Durchlaufen geschachtelter Arrays erfolgt auch analog mittels geschachtelter Schleifen:

```
double[][] d_array = new double[3][4];

// for-Schleife:
for (int x=0; x<d_array.length;x++){
   for(int y=0; y < d_array[x].length;y++){
      System.out.println(„Array-Element: “+d_array[x][y]);
   }
}
```

Beispiel 2-Dimensionales Array


```
// Oder mittels foreach-Schleife

for(double[] a: d_array){
   for(double b: a){
      System.out.println(„Array-Element: “+b);
   }
}

// while wäre auch möglich, aber unüblich
```

Zugriff auf Elemente ebenfalls analog:

- Bsp.:

```
int[][] mein_array = {{2,3,6},{1,2,2},{7,1,1,5,8}};
int value = mein_array[2][3]; // value?
// Geht auf das 3. Element und dort auf den 4. Eintrag, also value=5
```

Aufgabe 1

Sie wollen für Ihre erstellten Autos (Auto-Klasse aus der 3.Kursstunde) die Preise anheben.

- Speichern Sie daher innerhalb der Main-Methode alle Ihre erzeugten Autoobjekte in ein Array
- Lassen Sie sich nun die Preise all Ihrer erzeugten Autos über das Array anzeigen!
- Führen Sie anschließend eine Preiserhöhung von 1000 Euro für alle Ihre Autos durch.
- Überprüfen Sie, ob die Preiserhöhung funktioniert hat, indem Sie sich wieder die aktuellen Preise all Ihrer Autos anzeigen lassen.

Aufgabe 2

Erstellen Sie ein neues Eclipse-Projekt *Uebung08* und schreiben Sie ein Programm, das folgende Aufgabe erledigt:

- Das Programm soll sich zunächst eine bestimmte Anzahl an Zufallszahlen (`int`) ausdenken und diese in einer geeigneten Datenstruktur abspeichern.
- Sie sollen dabei die Anzahl der zu erzeugenden Zufallszahlen über eine lokale Variable festlegen können
  - Hinweis: Zufallszahlen können mit der Klasse Random aus dem Paket `java.util` erzeugt werden (wie letztes mal: mit `obj.nextInt(int)`)
- Wurden die Zufallszahlen erzeugt, sollen diese auf der Konsole ausgegeben werden.
- Nun soll das Programm noch die kleinste erzeugte Zufallszahl suchen und ausgeben.
  - Hinweis: Gehen Sie dazu alle erzeugten Zufallszahlen durch und merken Sie sich immer die kleinste aktuelle Zahl

Aufgabe 3

Eine Wetterstation hat für 7 Tage folgende Temperaturwerte aufgenommen:

Tag	0	1	2	3	4	5	6
Temperatur	23.2	17.8	20.0	14.1	15.4	19.9	18.0

- Erstellen Sie eine Klasse Wetterstation, welche Temperaturdaten in einer geeigneten Datenstruktur als Attribut hält.
- Die Klasse soll folgende Methoden anbieten:
  - Bestimmung und Rückgabe der Durchschnittstemperatur
  - Bestimmung und Rückgabe der maximalen Temperatur
  - Bestimmung und Rückgabe des Tages, an dem die niedrigste Temperatur gemessen wurde
- Erstellen Sie nun eine Anwendung, welche ein Objekt der eben implementierten Wetterstation mit den Daten aus der obigen Tabelle erzeugt.
- Dann soll die Anwendung der Reihe nach die Methoden aufrufen und die Ergebnisse auf der Konsole ausgeben.

Übungsaufgaben zu 2-dimensionalen Arrays

Aufgabe 4:

Schreiben Sie ein Programm „Schachbrett“ das folgende Aufgaben erledigt:

- Das Programm soll ein Schachbrett mit Grundaufstellung (also vor dem ersten Spielzug) simulieren und auf der Konsole ausgeben, siehe Abbildung rechts →

```
Console
<terminated> Schachbrett [Java Application]
T S L D K L S T
B B B B B B B B
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
B B B B B B B B
T S L D K L S T
```

- Das Schachbrett soll als 8x8 `char`-Array gespeichert sein.
- Jedes Feld des Schachbretts, auf dem **keine** Spielfigur steht, zeigt eine **0** an
- Jedes Feld des Schachbretts, auf dem **eine** Spielfigur steht, zeigt den **Anfangsbuchstaben** der jeweiligen Spielfigur (ohne Farbe) an:
  - K=König, D=Damen, L=Läufer,
  - S=Springer, T=Turm und B=Bauer
- Verwenden Sie passende Kontrollstrukturen und achten Sie auf eine effiziente Programmierung!