Programmiermethodik 1
Programmiertechnik

Arrays

Änderungshistorie

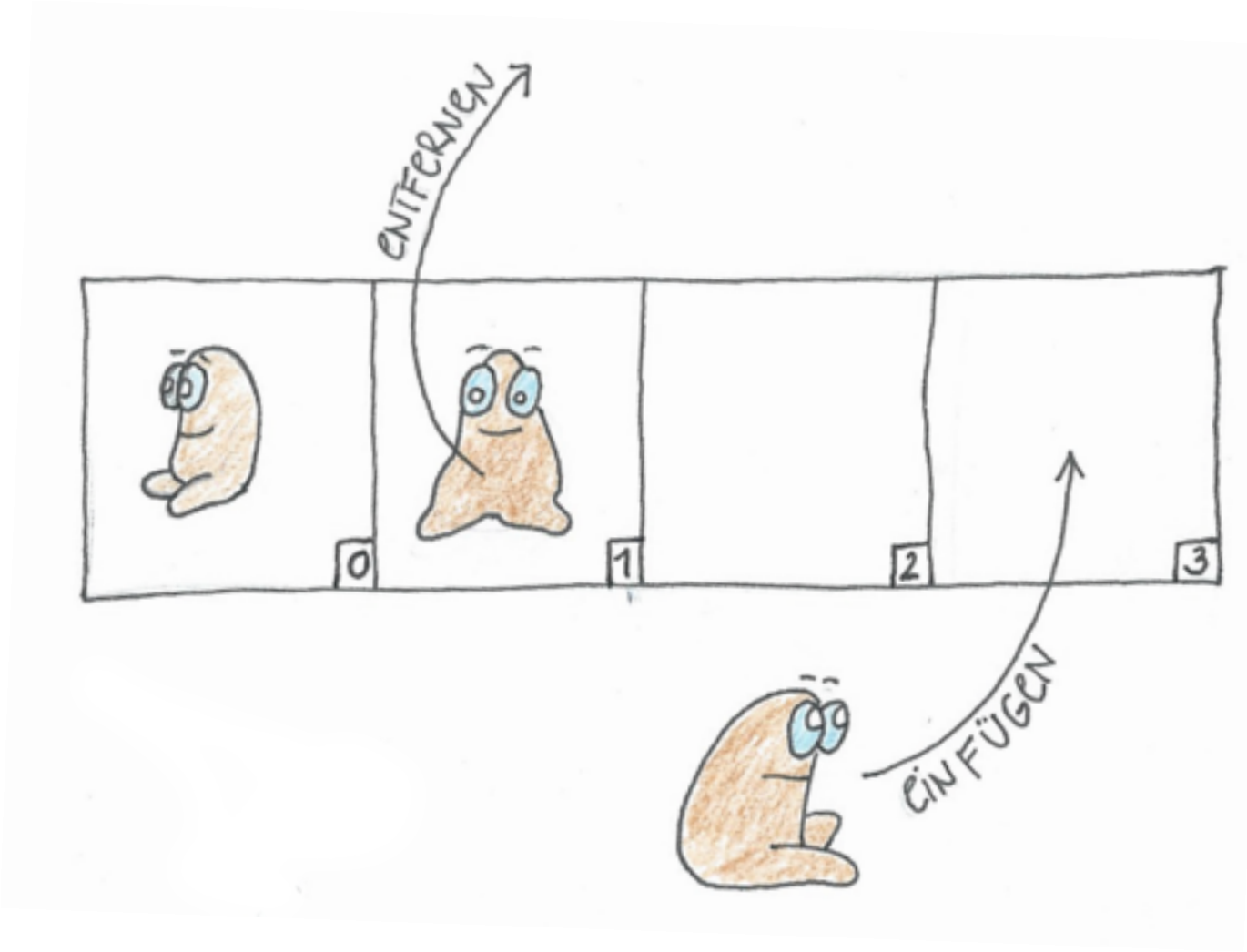
- 05.04.2016
    - Übungsaufgaben konkretisiert

Wiederholung

- Schleifen
  - while
  - do-while
  - for
  - break & continue

Ausblick

Worum gehts?

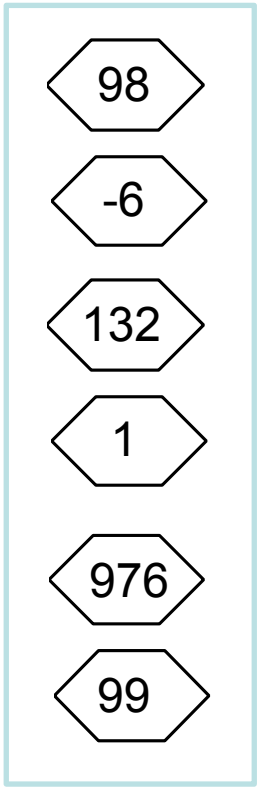


Agenda

- Erzeugung und Elementzugriff
- Traversierung von Arrays
- Mehrdimensionale Arrays
- Kopieren von Arrays

Erzeugung und Elementzugriff

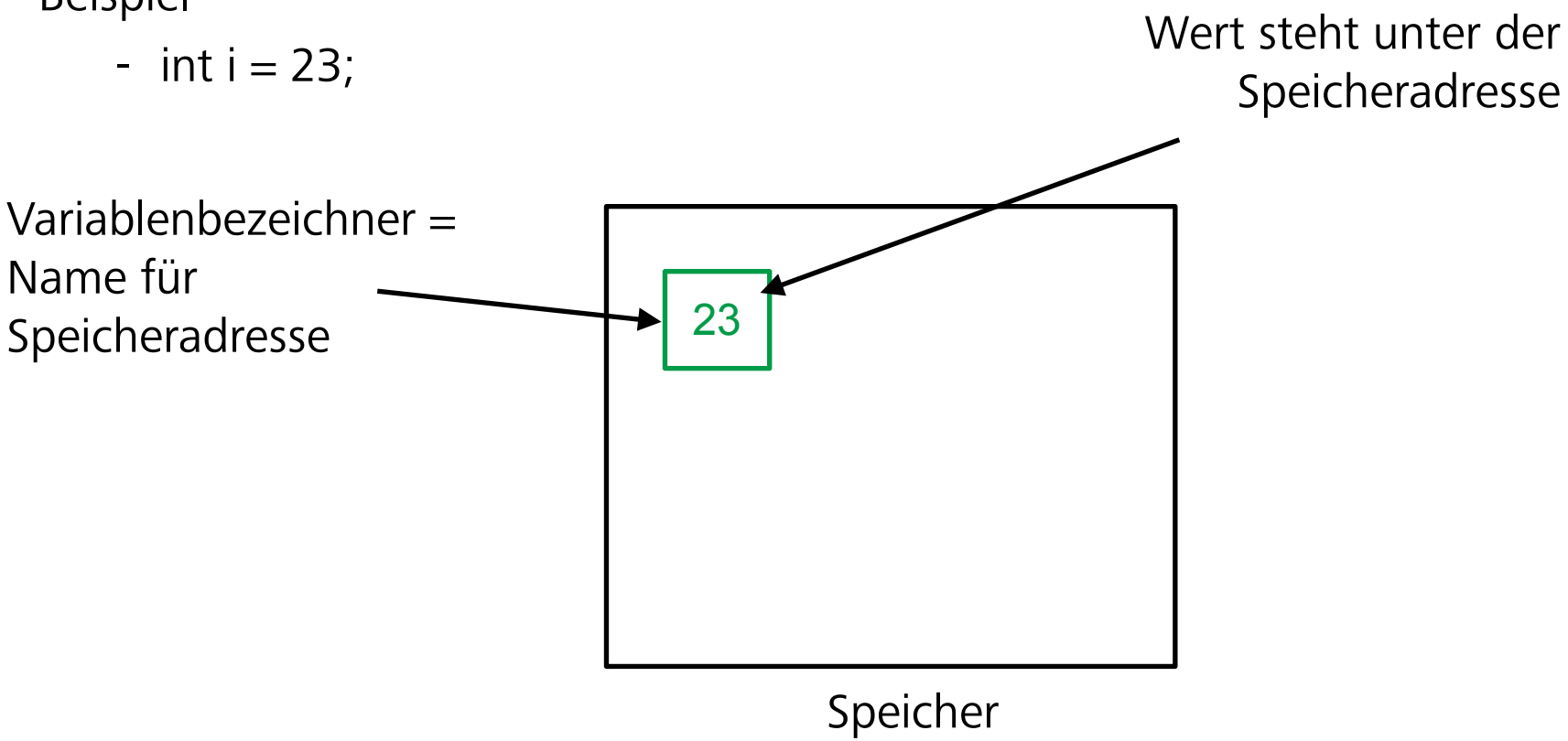
Was ist ein Array?



Beispiel: Array mit
sechs int-Elementen

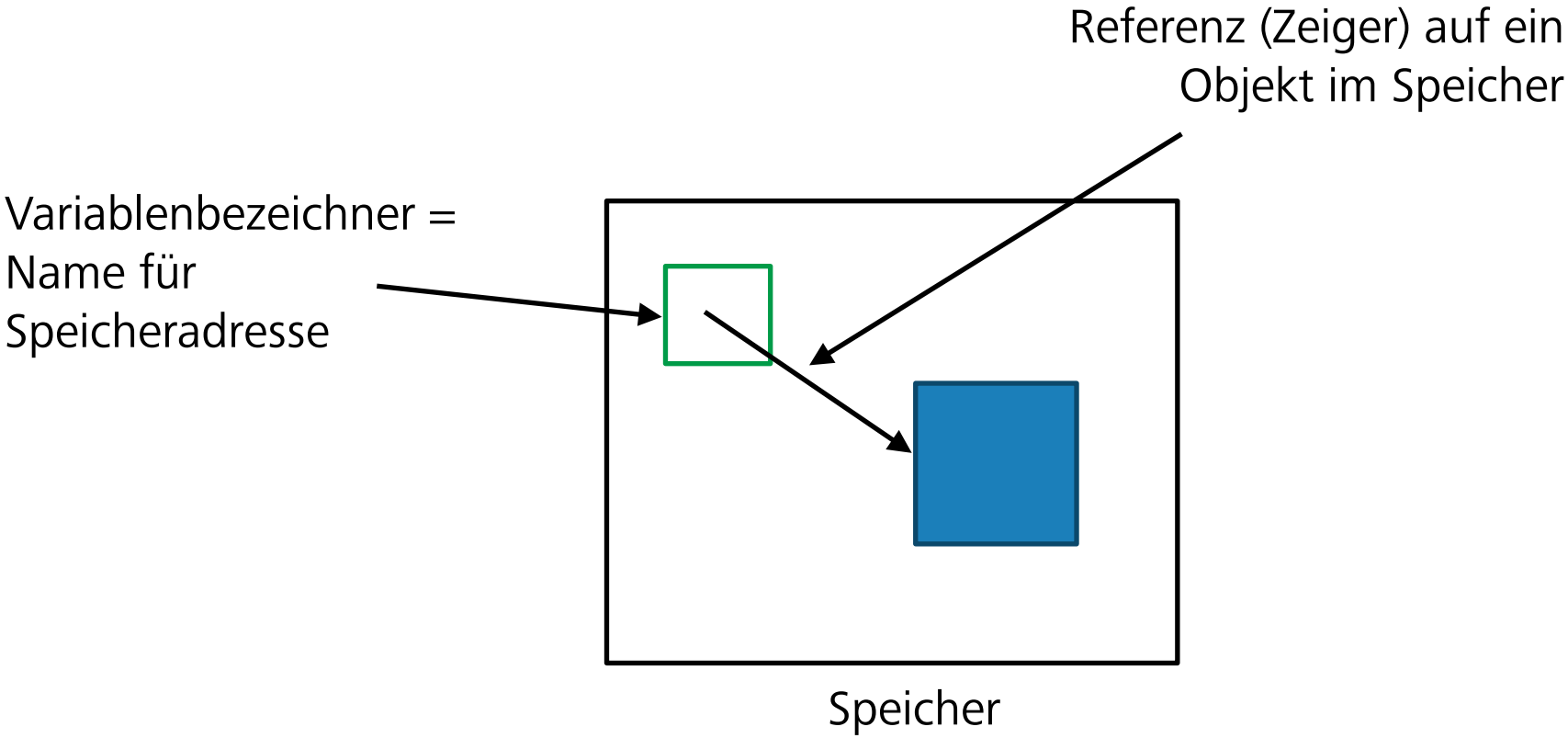
Primitive Datentypen

- bisher: primitive Datentypen
- Variablen reservieren Bereich im Speicher
- Variablenbelegungen (Werte) werden in diesen Bereich geschrieben
- Beispiel
    - int i = 23;



Referenztypen

- alle nicht primitiven Datentypen sind Referenztypen
- alle Klassen (später)



Einführung

- ein Array (auch „Feld“)
  - ist ein Referenztyp (Variablen sind Zeiger)
  - ist ein Container-Objekt: speichert Elemente anderer Typen
- Unterschiede zu Strings
  - Der Elementtyp ist beliebig, aber gleich für alle Elemente
  - Die Werte einzelner Elemente sind austauschbar (Referenzsemantik!)
- die Anzahl der Elemente eines Arrays („Arraylänge“) ist aber nach der Erzeugung unveränderlich!

Einführung

- Elementtyp bestimmt den Typ des Arrays
- Syntax: Deklaration eines Arraytyps:
  - Elementtyp + leere eckige Klammern (ohne Leerzeichen!)
  - <Elementtyp>[]
- zu jedem Elementtyp existiert ein korrespondierender Arraytyp
  - automatische Klassendefinition durch Compiler
- Beispiele
  - int → int[]
  - char → char[] (Zeichen, später)
  - String → String[] (Zeichenketten, später)
  - Bruch → Bruch[] (eigene Klasse, später)
- String ist eine spezielle Luxusversion von char[]
  - mit Sondereigenschaften, also mehr als char[]

Erzeugen

- Erzeugen eines neuen Arrays
    - mit Operator „new“
    - notwendig: Anzahl Elemente
- new <Elementtyp>[<Elementanzahl>]
    - <Elementanzahl>: int-Ausdruck, ggf. zur Laufzeit berechnet
- Beispiele:
    - new int[4]
    - new double[1 + 17*4]
    - new String[1]
    - new Bruch['a']

Erzeugen

- Elementanzahl wird zur Laufzeit beim new-Aufruf festgelegt
    - kann nachher nicht mehr verändert werden
- Vorstellung
    - Array = Liste namenloser Variablen
    - werden gemeinsam definiert
    - bleiben für die Lebensdauer des Arrays beisammen

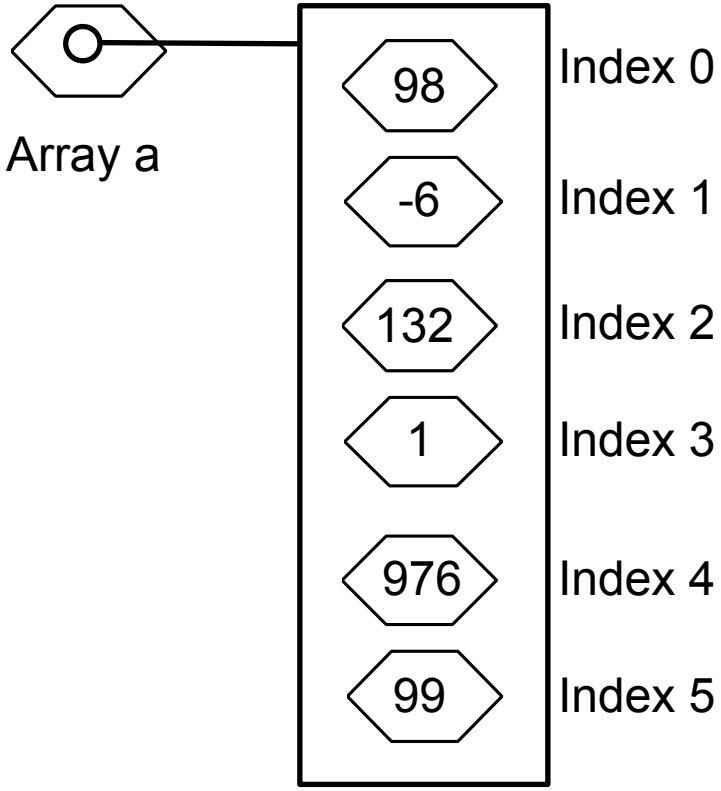
Eigenschaften

- Variablen von Arraytypen („Array-Variablen“) sind Referenzvariablen
    - Variable: Zeiger, Wert: Array-Objekt
- Beispiele:
- int[] a;
- Zuweisung eines Arrays an eine Arrayvariable:
- a = new int[4];
- verschiedene Arrays mit unterschiedlichen Längen als mögliche Werte einer Arrayvariablen:
    - int[] a;
    - a = new int[10];
    - a = new int[1];
    - a = new int[10000];

Zugriff auf Elemente

- alle Elemente eines Arrays folgen linear aufeinander
- jedes Element hat ganzzahligen Index
- Index des ersten Elementes = 0
    - dann fortlaufend weiter
- Index des letzten Elementes
    - Arraylänge – 1
- Zugriff auf alle Element ungefähr gleich schnell
    - random access

Zugriff auf Elemente



Beispiel: sechs Elemente mit Index 0 bis 5

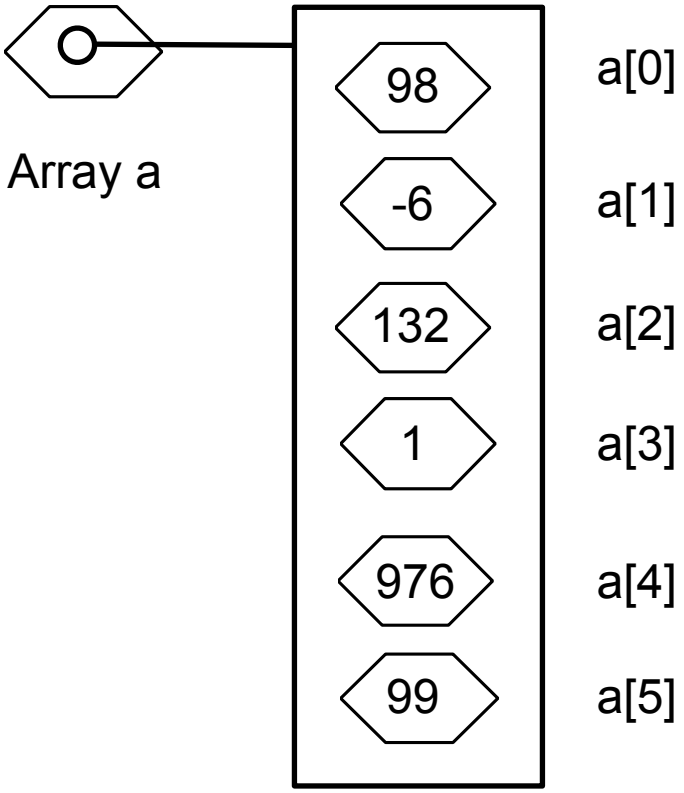
Zugriff auf Elemente

- Zugriff auf ein einzelnes Arrayelements über seinen Index
    - Syntax für Array-Elementzugriff:
- <Array-Variable>[<Index>]
    - <Index> = int-Ausdruck (zur Laufzeit berechnet)
- Zugriff auf ein Element berührt die anderen Elemente des Arrays nicht
- Arrayelemente sind benutzbar wie "normale" Variablen des Elementtyps

Zugriff auf Elemente

- Zugriff auf das zweite Element von

- weitere Beispiele:
  - int[] a = new int[6];
  - a[1] = -6;
  - a[3] = 1;
  - a[1]--;
  - a[152%3] = -a[1]*1805;
  - a[a[3]] = 71;



Vereinfachte Initialisierung

- durch Angabe einer Liste vorgegebener Elemente
- Syntax:
- <Array-Variable> = {<Element>, <Element>, ...};
    - <Element> = beliebiger Ausdruck, kompatibel zum Elementtyp des Arrays
    - Länge des Arrays = Anzahl angegebener Elemente
- Beispiel

  int[] a = {71, -4, 7220};
- ist Kurzfassung für

  int[] a = new int[3];

  a[0] = 71;

  a[1] = -4;

  a[2] = 7220;

Übung: Arrays

- Erzeugen Sie ein Array mit den Elementen "Affe", "Elefant" und "Katze".
- Greifen Sie auf das zweite Element zu und geben es auf der Konsole aus.

Traversierung von Arrays

Länge eines Arrays

- wird bei Erzeugung in der öffentlich lesbaren final-Objektvariablen int length abgelegt
- Zugriff:
  - <Array-Variable>.length
- Beispiel:

```
int[] a = {71, -4, 7220, 0, 238 };
System.out.println( a.length ); // gibt „5“ aus
```

Traversierung

- Iterieren über ein Array
  - alle Elemente durchlaufen
  - Elemente von vorne nach hinten der Reihe nach verarbeiten
- Beispiel mit for-Schleife:

```
double[] array = …;
for(int i = 0;  i < array.length;  i++){
    System.out.println(array[i]);
}
```

Traversierung

- besser: for-each Schleife
    - neuer Typ neben, for, while und do-while
    - nur für Arrays (uns später: Collections)
    - einfachere Form
- Syntax (ebenfalls mit Schlüsselwort for):

```
for( <Elementtyp> <Variable>: <Array> ) <Anweisung>
```

Traversierung

- Beispiel:

```
double[] array = …;
for(double element: array){
      System.out.println(element);
}
```

- for-each-Schleife
  - Kurzform einer for-Schleife
  - lesbar als "für jedes Element element in Array array"

Traversierung

- Äquivalenz for-Schleife und for-each-Schleife
- for-each-Schleife ist ersetzbar durch eine for-Schleife :

```
for(<Elementtyp> element: array) {
    …
}
```

- ist äquivalent zu

```
for(int i = 0;  i < array.length;  i++){
    <Elementtyp> element = array[i];
    …
}
```

- Umkehrung gilt nicht!
    - eine for-Schleife ist nicht immer durch for-each ersetzbar!

Traversierung

- in jedem Schleifendurchgang wird eine neue Schleifenvariable erzeugt
    - enthält Wert des entsprechenden Array-Elements
- Beispiel (Variable element):

```
for(<Elementtyp> element: array) {
    …
}
```

Einschränkungen

- nur Lesen, kein Schreiben der Array-Elemente
    - kein Zugriff auf den Index, nur auf lokale Kopie des Elements
- Start immer mit erstem Element
- sequentieller Durchlauf, keine Sprünge
- nur ein Array, nicht mehrere parallel
- Durchlauf kann nur mit break abgebrochen werden

Anwendung

- for-each geeignet für beispielsweise …
    - Ausgabe von Elementen
    - Suche nach Element
- for-each nicht brauchbar für
    - Initialisierung von Arrays / Änderung von Elementen
    - Kopieren von Arrays / Vergleich zweier Arrays

Übung: Traversierung

- Durchlaufen Sie nun das Array tiere aus der letzten Aufgabe mit einer for-each-Schleife und geben jeder Tier auf der Konsole aus.
- Was müssen Sie ändern, damit Sie alle "Affe"-Elemente in "Menschenaffe"-Elemente verändern können?

Mehrdimensionale Arrays

Mehrdimensionale Arrays

- zu jedem Typ wird ein korrespondierender Arraytyp erzeugt
- auch Arrays selbst können Elemente eines anderen Arrays sein
    - mehrdimensionale Arrays oder geschachtelte Arrays

Erzeugen

- Deklaration eines zweidimensionalen Arraytyps:
  - Elementtyp + zwei leere eckige Klammern

```
<Elementtyp>[][]
```

- Beispiel: Deklaration einer Matrix m

```
int[][] m;
```

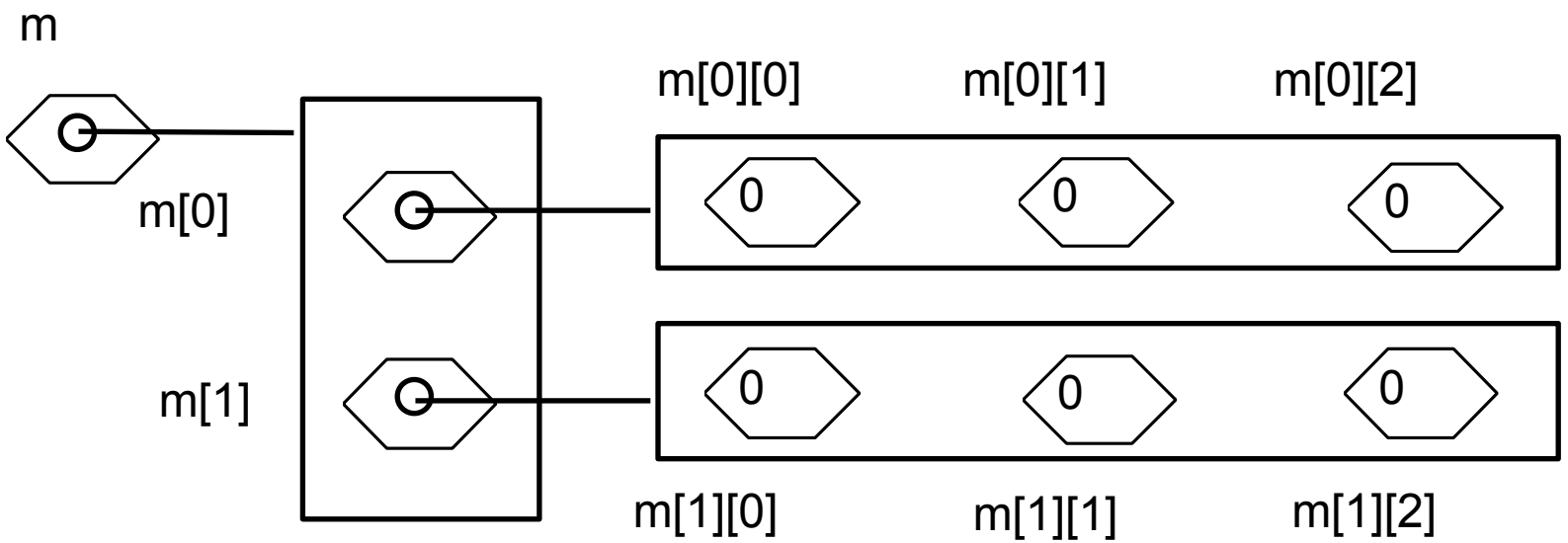
- Erzeugen mit new + Anzahl Elemente in jeder Dimension

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

- erzeugt ein Array mit 2 Elementen, von denen jedes ein Array mit 3 int-Elementen ist

Erzeugen

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



Elementzugriff

- Elementzugriff: ein Index für jede Dimension
- Beispiel:

```
m[1][0] = 10;
```

  - erster Index für das Array der ersten Ebene
  - zweiter Index für das Array der zweiten Ebene

- Beispiel: Array initialisieren

```
int[][] m = new int[2][3];
m[0][0] = 0;
m[0][1] = 34;
m[0][2] = 234;
m[1][0] = -10;
m[1][1] = 1;
m[1][2] = 15452;
```

Traversierung

- Beispiel: Erzeugung und Ausgabe einer 1x1-Tabelle

```
int[][] einmaleins = new int[10][10];
for (int i = 0; i < einmaleins.length; i++) {
   for (int j = 0; j < einmaleins[i].length; j++) {
      einmaleins[i][j] = (i + 1) * (j + 1);
   }
}
```

- Ausgabe mit for-each-Schleife:

```
for (int[] zeile : einmaleins) {
   for (int wert : zeile) {
      System.out.format("%4d", wert);
   }
   System.out.println(); // Zeilenvorschub ausgeben
}
```

Initialisierung

- vereinfachte Initialisierung zweidimensionaler Arrays
- Syntax für eine m x n-Matrix:

```
<Array-Variable> = {
   {<Element00>, <Element01>, ...., <Element0n>},
   {<Element10>, <Element11>, ...., <Element1n>},
   ....
   {<Elementm0>, <Elementm1>, ...., <Elementmn>}
}
```

- Beispiel:

```
int[][] m = { { 0, 1, 2}, {10, 11, 12} };
```

Übung: Mehrdimensionale Arrays

- Gegeben ist das zweidimensionale Array `array` bestehend aus 2 Zeilen und 3 Spalten:

```
int [][] array = {{1,2,3},{4,5,6}};
```

- Schreiben Sie Code zur Ausgabe des Arrays (mit beliebigen Länge) auf der Konsole.

```
1  2  3
4  5  6
```

Kopieren von Arrays

Wertzuweisung

- Erinnerung: Arrays haben Referenzsemantik!
- Wertzuweisung eines Arrays kopiert die Referenz
- nicht das Array-Objekt, nicht die Elemente

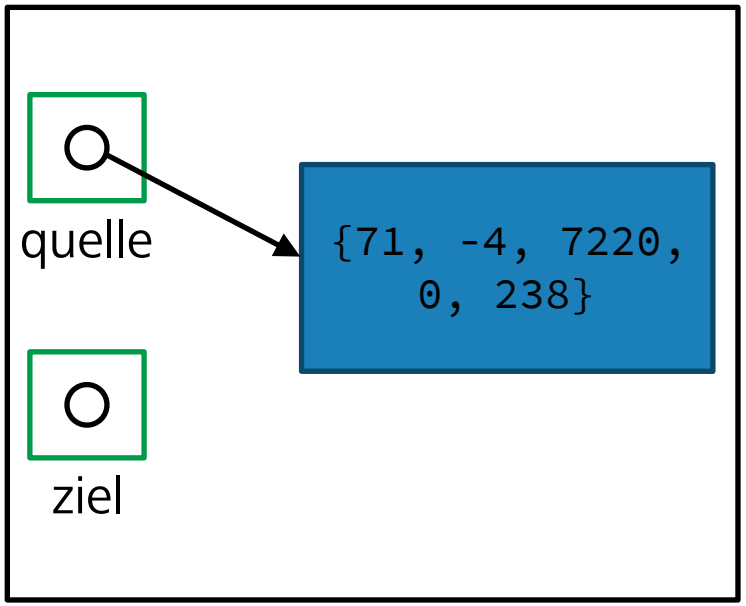
```
int[] quelle = {71, -4, 7220, 0, 238};
int[] ziel = quelle;
```

- Änderungen über eine Variable sind in beiden sichtbar: Aliasing

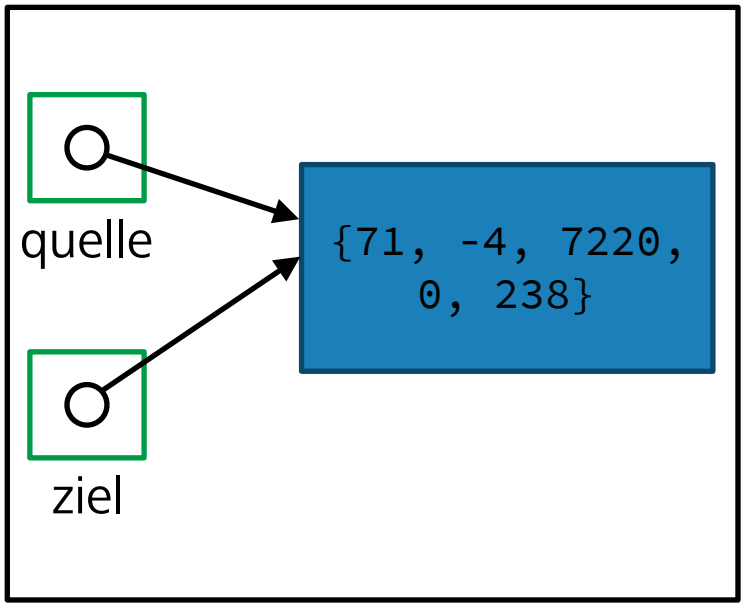
```
quelle[0] = 23;
System.out.println(ziel[0]); // gibt 23 aus
```

Kopieren von Arrays

```
int[] quelle = {71, -4, 7220, 0, 238};
int[] ziel = quelle;
```



Zeile 1



Zeile 2

Kopieren

- Erzeugen einer echten Kopie
  - neues Array erzeugen (new)
  - Originalwerte elementweise der Kopie zuweisen
- Beispiel für Elemente eines primitiven Typs (int):

```
int[] quelle = {71, -4, 7220, 0, 238};
int[] ziel = new int[quelle.length];
for(int i = 0;  i < quelle.length;  i++){
   ziel[i] = quelle[i];
}
```

- Ergebnis:
  - zwei unabhängige Array-Objekte, deren Elemente aber jeweils identische Werte besitzen

Kopieren

- Methode System.arraycopy
- Vordefiniert zum Kopieren von Arrays:
  - statische Methode arraycopy in Klasse System

```
static void arraycopy(Object src, int srcPos, Object dest,
   int destPos, int length)
```

- Argumente:
  - src Original-Array, wird gelesen ("source" = "Quelle")
  - srcPos Index des ersten Elementes in src, das kopiert werden soll
  - dest Ziel-Array, wird geschrieben ("destination" = „Ziel“), dest muss vom gleichen Elementtyp wie src sein
  - destPos Index in dest, ab dem geschrieben wird length Anzahl der Elemente

Kopieren

- Beispiel:

```
int[] quelle = {71, -4, 7220, 0, 238};
int[] ziel = new int[quelle.length];
System.arraycopy(quelle, 0, ziel, 0, quelle.length);
```

Vergleichen

- Vergleich mit == liefert Aussage über Identität, nicht inhaltliche Gleichheit

```
int[] array1 = { 1, 2, 3, 4 };
int[] array2 = { 1, 2, 3, 4 };
System.out.println("Gleich? " + (array1 == array2));
```

- inhaltlicher Vergleich von Arrays:
    - zunächst Längen vergleichen
    - dann paarweise die Elemente

```
boolean istGleich = array1.length == array2.length;
// Abbruch falls Längen ungleich
for (int i = 0; i < array1.length; i++) {
   istGleich = istGleich && array1[i] == array2[i];
}
System.out.println("Gleich? " + istGleich);
```

Übung: Kopieren

- Wir betrachten noch einmal das mehrdimensionale Array aus der letzten Übung:

```
int [][] array = {{1,2,3},{4,5,6}};
```

- Schreiben Sie Code, der dem Array eine weitere Zeile hinzufügt (mit 0'en)
- also:

1 2 3
4 5 6
7 8 9

Zusammenfassung

- Erzeugung und Elementzugriff
- Traversierung von Arrays
- Mehrdimensionale Arrays
- Kopieren von Arrays