

# Einfache Arrays

---

Philipp Wendler

Zentralübung zur Vorlesung

„Einführung in die Informatik: Programmierung und Softwareentwicklung“

<https://www.sosy-lab.org/Teaching/2017-WS-InfoEinf/>



## Arrays: Elemente gleichen Typs

Ein Array ist ein Tupel von Elementen gleichen Typs

$$\mathbf{a} = [\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \dots, \mathbf{w}_n]$$

Welche Anweisung wird vom Compiler **nicht** akzeptiert?

- a) `int[] a = {1, 2};`
- b) `int[] a = {1, 2.0};`
- c) `double[] a = {1, 2};`
- d) `double[] a = {1, 2.0};`

Wiederholung von ZÜ3: **automatische Typkonversion** zum größeren Typ

`byte < short < int < long < float < double`

## Aufgabe 1: Arithmetisches Mittel

Schreibe eine Methode, die das arithmetische Mittel aller Zahlen in einem `int`-Array berechnet.

Algorithmusidee:

- Summiere alle Elemente des Arrays auf
- Teile die Summe durch die Anzahl der Elemente

## Aufgabe 1: Arithmetisches Mittel (Lösungsidee 1)

```
public static double durchschnitt(int[] daten) {  
    int sum = 0;  
    for (int i = 0; i < daten.length; i++)  
        sum = sum + daten[i];  
    return sum / daten.length;  
}
```

Welches Problem tritt  
in dieser Methode auf?

- a) Die Methode darf nicht **static** sein.
- b) Es werden nicht alle Elemente aufsummiert.
- c) Die Division liefert nicht das gewünschte Ergebnis.
- d) Es kann eine Division durch 0 auftreten.
- e) Es tritt kein Problem auf.

## Aufgabe 1: Arithmetisches Mittel (Lösungsidee 2)

```
public static double durchschnitt(int[] daten) {  
    int sum = 0;  
    for (int i = 0; i < daten.length; i++)  
        sum = sum + daten[i];  
    return (double)sum / daten.length;  
}
```

Array daten muss mind.  
ein Element enthalten,  
sonst Division durch 0!

## Aufgabe 2: Sortierung (I)

Schreibe eine Methode, die überprüft, ob ein `double`-Array aufsteigend sortiert ist.

Algorithmusidee: Durchlaufe das Eingabe-Array elementweise

- Falls **ein** Element größer als sein rechter Nachbar ist, gib `false` zurück
- Falls **alle** Elemente kleiner als ihr rechter Nachbar sind, gib `true` zurück

```
public static boolean istSortiert(double[] arr) {  
    for (int i = 0; i < arr.length-1; i++) {  
        if (arr[i] > arr[i+1])  
            return false;  
    }  
    return true;  
}
```

sofortige Beendigung der Methode

## Aufgabe 2: Sortierung (II)

Schreibe eine Methode, die überprüft, ob ein `double`-Array aufsteigend sortiert ist.

Algorithmusidee: Durchlaufe das Eingabe-Array elementweise

- Falls **ein** Element größer als sein rechter Nachbar ist, gib `false` zurück
- Falls **alle** Elemente kleiner als ihr rechter Nachbar sind, gib `true` zurück

```
public static boolean istSortiert(double[] arr) {  
    for (int i = 1; i < arr.length; i++) {  
        if (arr[i - 1] > arr[i])  
            return false;  
    }  
    return true;  
}
```

Achtung: Beginne mit dem Index 1, sonst ist `arr[i-1]` nicht definiert.



## Aufgabe 2: Sortierung (III)

```
public static boolean istSortiert(double[] arr) {  
    for (int i = 1; i < arr.length; i++) {  
        if (arr[i - 1] > arr[i])  
            return false;  
    }  
    return true;  
}
```

Wann tritt in dieser  
Methode ein Problem auf?

- a) ... falls `arr` leer ist.
- b) ... falls `arr` nur ein Element hat.
- c) ... falls `arr` zwei Elemente hat.
- d) ... nie.

## Aufgabe 3a: Kleines Einmaleins mit Arrays

Gib das kleine Einmaleins auf der Konsole mit Hilfe von Arrays aus.

Algorithmusidee: (siehe ZÜ5)

$$x * y = \text{produkt}$$

- Erzeuge ein leeres doppeltes Array (=> Matrix)
- Für  $x$ : gehe die Zahlen von 1 bis 10 durch
- Für  $y$ : gehe nochmal die Zahlen 1 bis 10 durch  
für jede Möglichkeit der Zahl  $x$
- Berechne das Produkt aus den Zahlen  $x$  und  $y$  und speichere den Wert in der entsprechenden Zelle des doppelten Arrays.

## Aufgabe 3a: Kleines Einmaleins mit Arrays

```
public class EinmaleinsArrays {  
    public static void main(String[] args) {  
  
        int[][] elements = new int[10][10];  
  
        for (int i = 1; i <= 10; i++) {  
  
            for (int j = 1; j <= 10; j++) {  
  
                elements[i-1][j-1] = i*j;  
  
            }  
  
        }  
  
    }  
}
```

Erzeuge ein doppeltes  
leeres Array

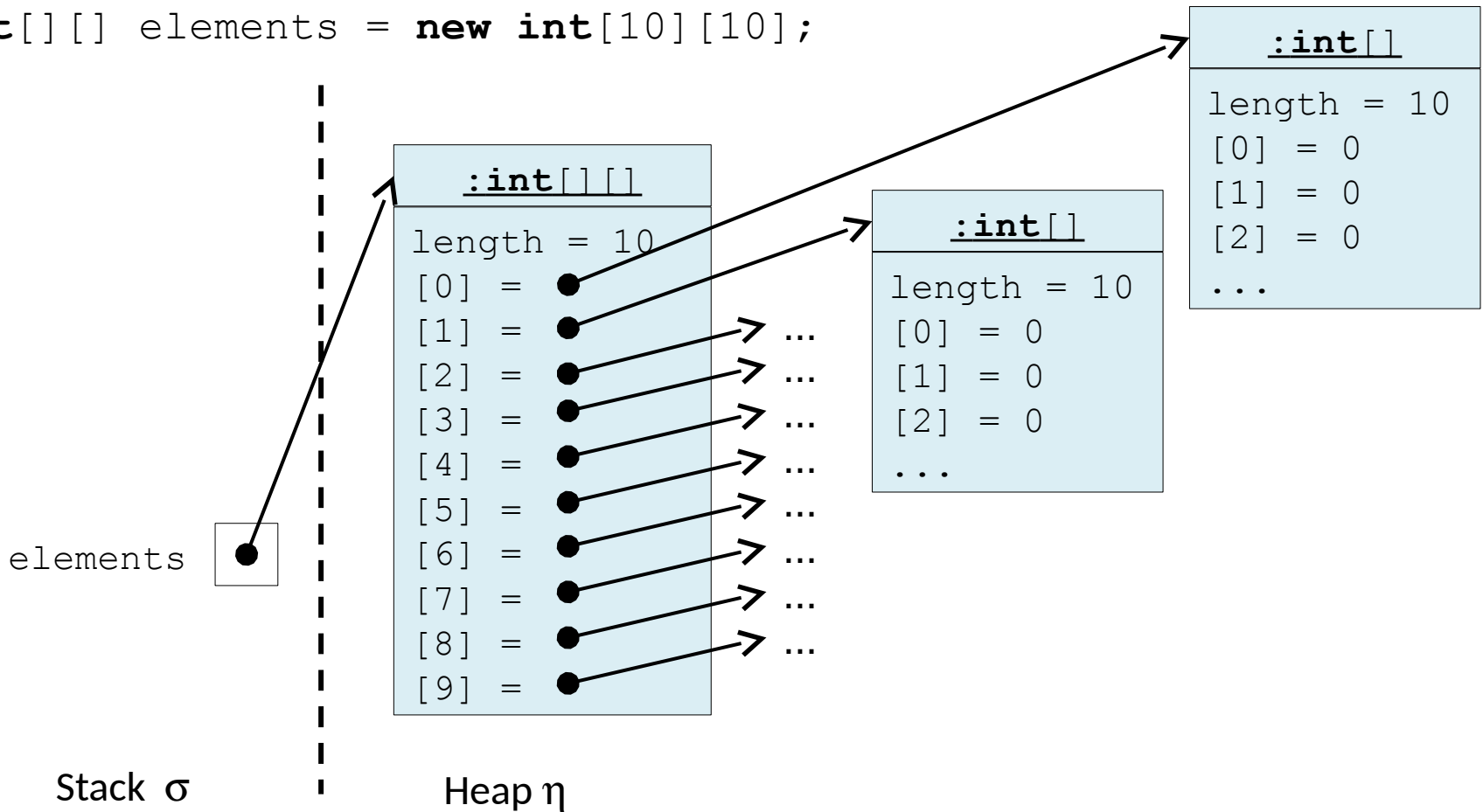
Für  $x$ : gehe die Zahlen von  
1 bis 10 durch

Für  $y$ : gehe nochmal die  
Zahlen 1 bis 10 durch für  
jede Möglichkeit der Zahl  $x$

Berechne das Produkt aus  
den Zahlen  $x$  und  $y$  und  
speichere den Wert.

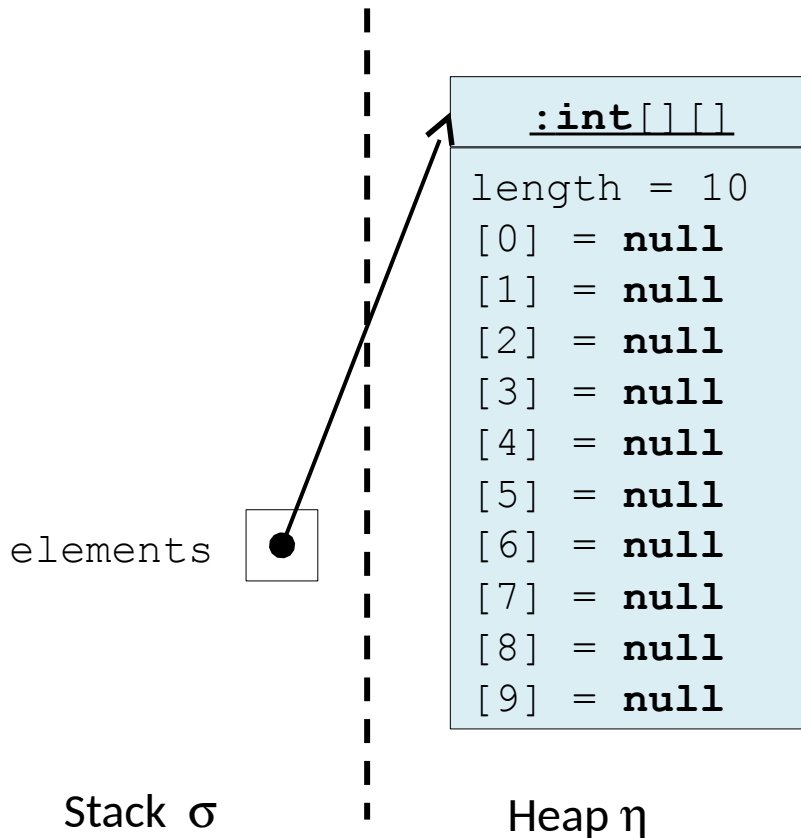
## Aufgabe 3b: Speicherdarstellung

```
int[][] elements = new int[10][10];
```



## Aufgabe 3b: Speicherdarstellung (partiell initialisiert)

```
int[][] elements = new int[10][];
```



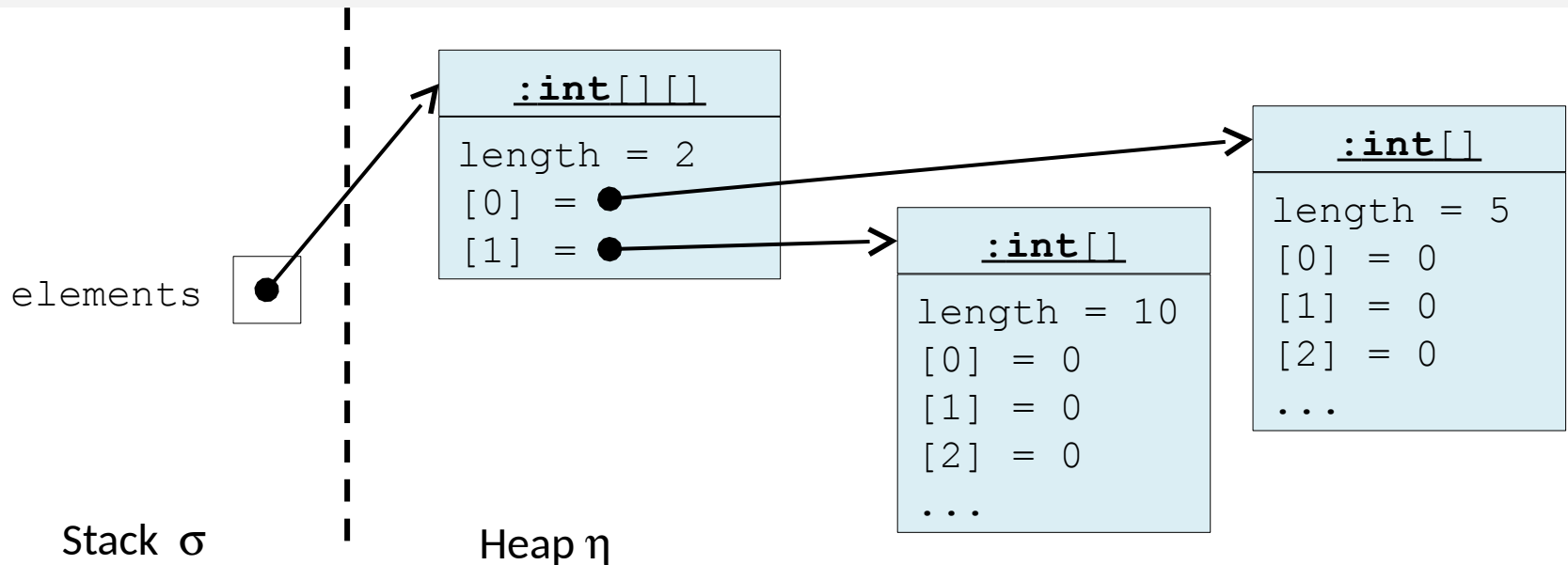
# Einschub:

Kann ein doppeltes Array unterschiedlich lange "Zeilen" haben?

**Codeausschnitt:** `int[][] elements = new int[2][];`

`elements[0] = new int[5];`

`elements[1] = new int[10];`

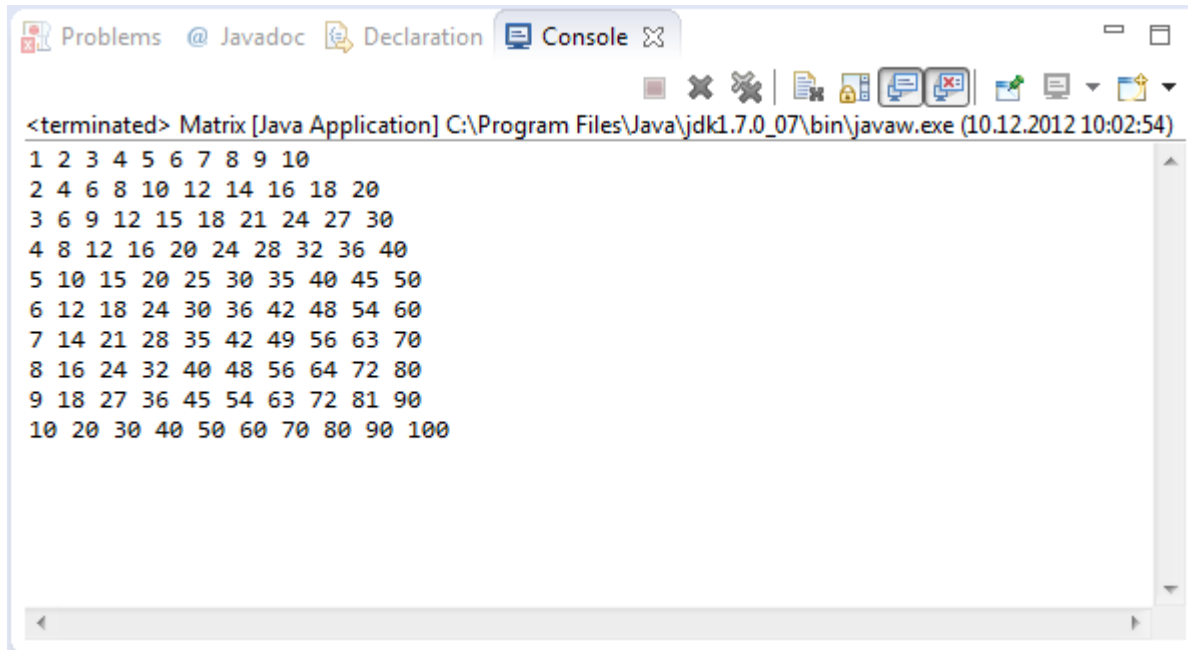


## Aufgabe 3c: Ausgeben einer Matrix

```
public class EinmaleinsArrays {
    public static void main(String[] args) {
        int[][] elements = new int[10][10];
        for (int i = 1; i <= 10; i++) {
            for (int j = 1; j <= 10; j++) {
                elements[i-1][j-1] = i*j;
            }
        }

        for (int i = 0; i < elements.length; i++) {
            for (int j = 0; j < elements[i].length; j++) {
                System.out.print(elements[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

## Aufgabe 3c: Ausgeben einer Matrix



The screenshot shows a Java IDE console window with the following output:

```
<terminated> Matrix [Java Application] C:\Program Files\Java\jdk1.7.0_07\bin\javaw.exe (10.12.2012 10:02:54)
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
```