



# Java Einführung

## **Arrays**

Kapitel 7

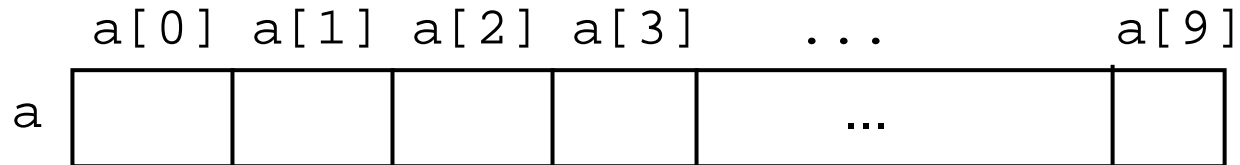
# Inhalt

- Eindimensionale Arrays
  - Erzeugung
  - Benutzung
  - Zuweisung
  - Beispiel
- Mehrdimensionale Arrays

# Arrays

- Oft benötigt man nicht nur Einzelwerte (eine Variable) sondern eine Wertemenge, die gemeinsam verarbeitet werden
- Wertmengen werden oft in Tabellen oder Listen angeordnet
- Ein Array ist eine Tabelle von gleichartigen Elementen (Zahlen, Zeichen,...)

# Eindimensionales Array



- Ein Array hat einen **Namen** (a)
- Die einzelnen Elemente können durch den **Index** ([0], [1], ...) angesprochen werden
- Die einzelnen Elemente verhalten sich wie namenlose Variablen
- Die Größe wird einmal festgelegt und ist dann fix.

# Erzeugung eines Arrays

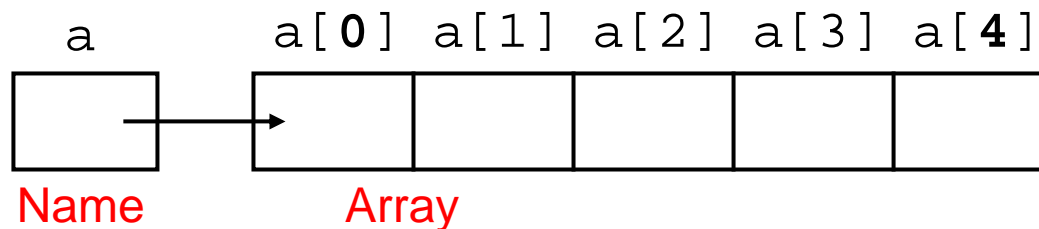
- Deklaration des **Namens** (Arrayvariable, Zeiger)

```
int[] a;
```

```
float[] vector;
```

- Erzeugung des **Arrays**

```
a = new int[5]; // Array mit Index 0...4
```



# Benutzung eines Arrays

- Arrayelemente werden durch den Index angesprochen und können wie normale Variablen verwendet werden

```
int i=1, j=3;
```

```
a[3]=0;
```

```
a[2*i+1]=a[j]; // i und j sind int
```

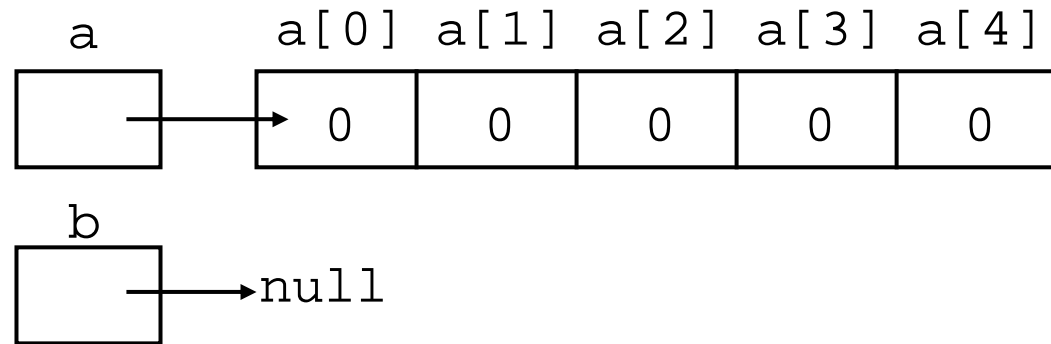
```
a[max(i,j)]=100; // max ist eine Fkt.
```

- Zur Laufzeit wird geprüft ob der verwendete Index gültig ist (ob es das Element gibt)
- `a.length` enthält die Länge des Arrays

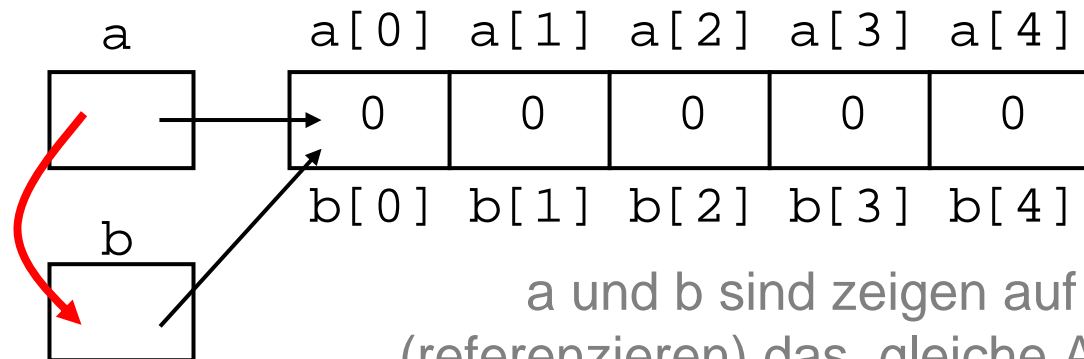
# Zuweisung eines Arrays

- Einer Arrayvariable (Name des Arrays) dürfen alle Arrays des passenden Typs zugewiesen werden

```
int[] a;  
a=new int[5];  
int[] b;
```



```
b=a;
```



a und b sind zeigen auf  
(referenzieren) das gleiche Array!

# Freigabe von Arrays

- Java verwendet zur Freigabe von Speicher eine **automatische Speicherbereinigung** (Garbage Collection)
- Wenn kein Zeiger (Arrayvariable) mehr auf das Array zeigt, wird das Array gelöscht
  - die Arrayvariable wird verändert `a=null;` oder `a=b;`
  - die Arrayvariable verlässt den Bereich ihrer Lebensdauer

# Initialisierung von Arrays

- Arrays können schon bei der Deklaration initialisiert werden

```
int[] primes = {2, 3, 5, 7, 11};
```

- Später können die Elemente nur noch einzeln (über den Index) zugewiesen werden

# Typische Schleife für Arrays

```
class Primes {  
    public static void main (String[] args) {  
        int[] primes = {2, 3, 5, 7, 11};  
  
        for (int i = 0; i < primes.length; i++) {  
            System.out.print(primes[i] + " ");  
        }  
    }  
}
```

# Bsp: Sequentielle Suche

- Bei der sequentiellen Suche werden alle Elemente durchgegangen, bis das richtige Element gefunden wird

```
static int search(int[] a, int x) {  
    int pos = a.length-1;  
    while (pos>=0 && a[pos]!=x) { pos--;}  
    return pos;  
}
```

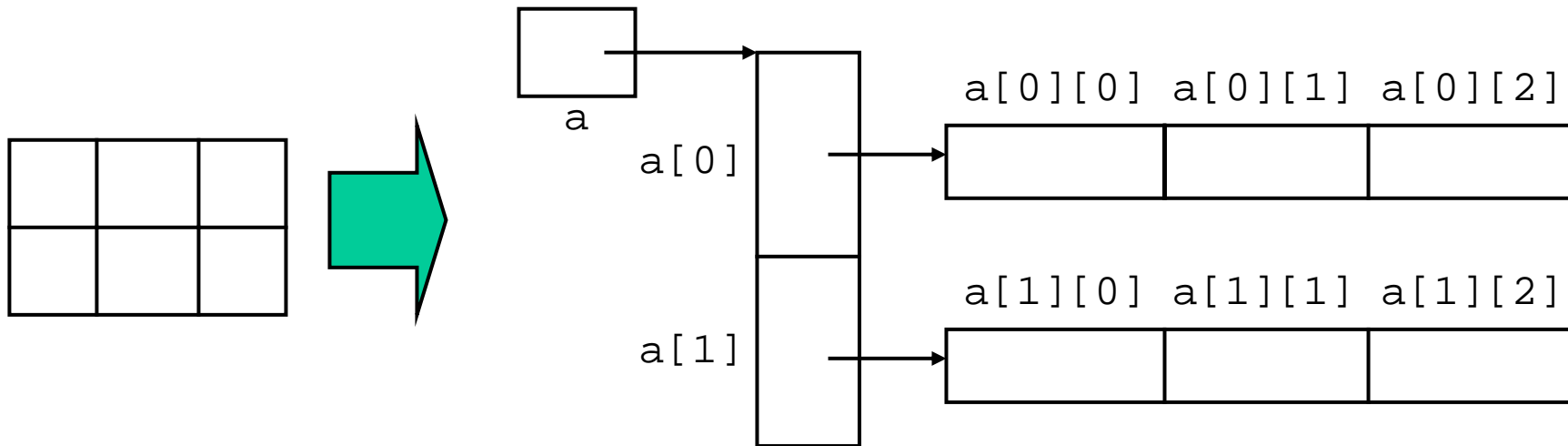
```
static void main (String[] args) {  
    int[] b= { 27, 11, 19};  
    int c= search(b,11);  
}
```

Achtung: das Array wird nicht kopiert, es wird nur eine "Referenz" auf das Array übergeben!

# Mehrdimensionales Array

- Mehrdimensionale Arrays sind **Arrays von Arrays**
- Bsp: Zweidimensionales Array (Matrix)

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



# Benutzung Mehrdimensionaler Array

- Initialisierung

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

- Länge

```
a.length // Zeilen  
a[0].length // Spalten
```

# Nach dieser Einheit sollten Sie ...

- Arrays erzeugen und über den Index benutzen können
- Wissen was der Arrayname ist und was bei der Zuweisung von Arraynamen passiert