

# Programmieren I

## Arrays



Heusch 7.2  
Ratz 5.1

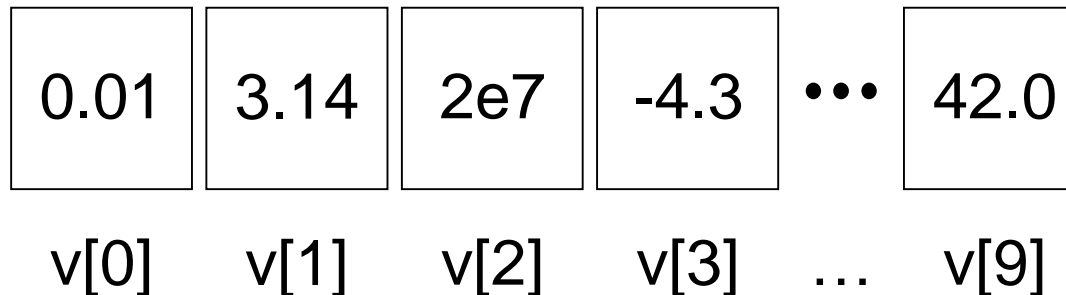
Institut für Automation und angewandte Informatik

```
final List<String> allResults = new ArrayList<String>();  
final Map<String, Integer> typeWordResultCount = new HashMap<String, Integer>();  
final Map<String, Integer> typePoints = new HashMap<String, Integer>();  
evaluation.put(type, typePoints);  
  
for (final Sheet sheet : this.sheets) {  
    final String sheetResult = sheet.getPlayerInput(type);  
    if (sheetResult.startsWith(start) && this.isValidWord(sheetResult, type)) {  
        validWordCountForType++;  
        allResults.add(sheetResult);  
    }  
}
```

# Arrays: Definition

- Arrays (dt. Felder) dienen zum Speichern mehrerer gleichartiger Daten („Vektoren“).
- Beispiel: Array mit Name  $v$  und 10 Gleitpunktzahlen als Komponenten

```
float[] v = new float[10];
```



# Arrays: Eigenschaften (1)

- Arrays sind Objekte, keine primitiven Datentypen.
- Sie werden dynamisch erzeugt.
- Beispiel für Deklaration und Erzeugung:

```
float[] v;           // Deklaration der Feld-Variablen  
v = new float[10];   // Erzeugung des Arrays und  
                     // Zuweisung der Referenz an v
```

- Die Klammern `[]` in der Deklaration der Feld-Variablen können auch hinter dem Variablennamen angegeben werden:

```
float v[];
```

- Die Anzahl der Komponenten des Arrays muss beim Erzeugen angegeben werden.
- Für Arrays gibt es keine Konstruktoren, sondern es wird der `new`-Operator mit den eckigen Klammern `[]` verwendet.

## Arrays: Eigenschaften (2)

- Die Anzahl der Komponenten eines Arrays erhält man durch das `length`-Attribut des Arrays: `v.length` ist in unserem Beispiel gleich 10.
- Die Indizes der Komponenten müssen ganzzahlig sein (`int`-Werte).
- Die Komponenten selbst werden mit 0 beginnend nummeriert.
- Der größte zulässige Index ist immer um 1 kleiner als die Anzahl der Komponenten (`length-1`).
- Negative Indizes und Indizes größer oder gleich der Länge `length` sind nicht zugelassen (z.B. führt `v[v.length]` zum (Laufzeit-)Fehler).

## Arrays: Eigenschaften (3)

- Die Elemente eines auf diese Weise erzeugten Arrays werden mit dem für diesen Typ üblichen Standardwert initialisiert.  
(Bei unserem Array  $v$  beispielsweise mit  $0.0f$ )
- Direkter Zugriff auf jede Komponente (z.B.  $v[1] = 1.2;$ ) wie auf eine einzelne Variable („Skalar“).
- Die Feldlänge kann auch in Form eines *Ausdrucks* angegeben werden;

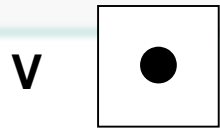
```
int halfLength = 5;  
float[] v = new float[2 * halfLength];
```

- Arrays werden automatisch vom Garbage Collector entfernt.

# Erzeugung mit new-Operator grafisch dargestellt

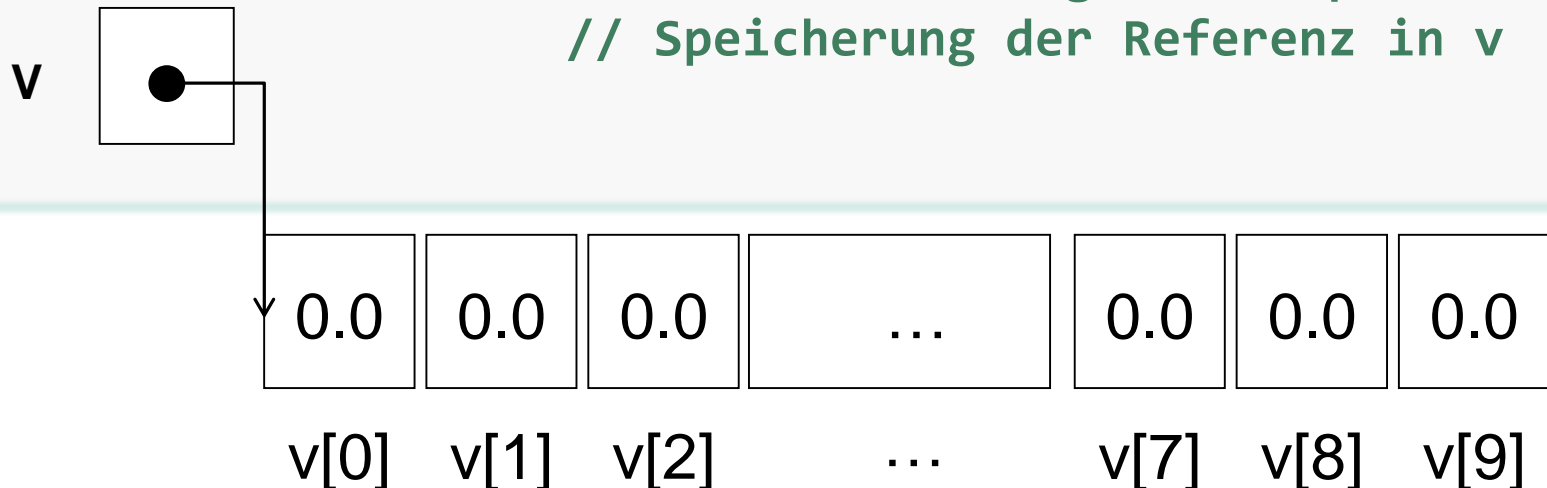
## 1. Anweisung im Beispiel (Deklaration):

```
float[] v;           // Anlegen der Referenzvariable v
                     // Noch kein Array angelegt
```



## 2. Anweisung im Beispiel (Erzeugung mit new-Operator):

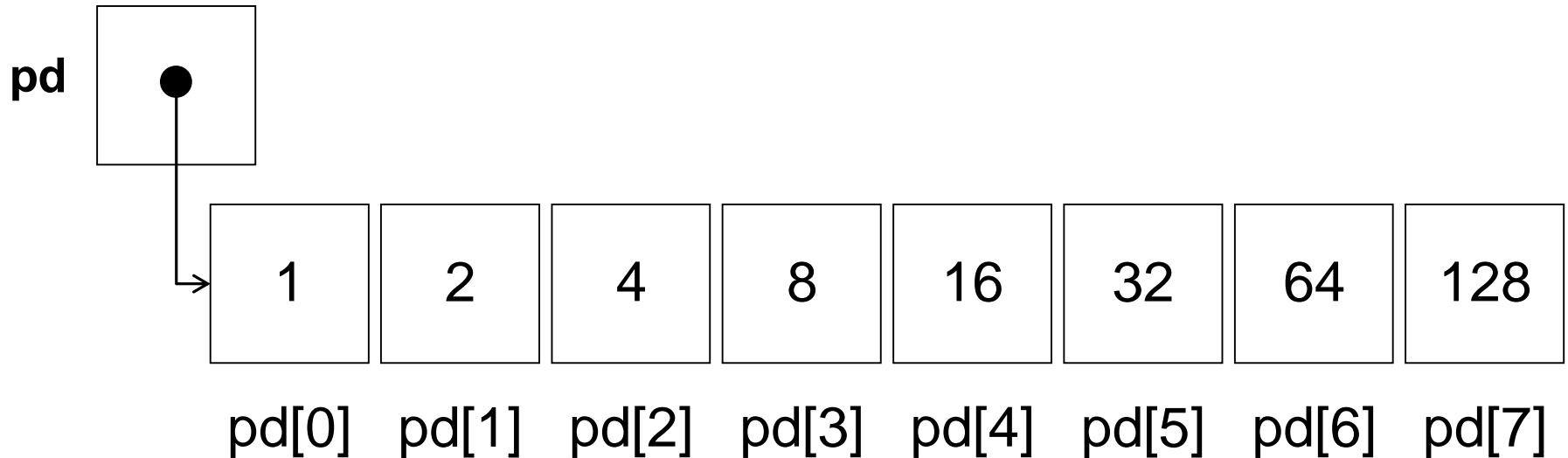
```
v = new float[10];  // Anlegen des Feldes
                     // Initialisierung der Komponenten
                     // Speicherung der Referenz in v
```



# Erzeugung durch Initialisierer

## ■ Beispiel:

```
int[] pd = {1,2,4,8,16,32,64,128}; // oder  
int[] pd = new int[] {1,2,4,8,16,32,64,128};
```



## ■ Implizite Festlegung der Array-Größe (hier: 8)

# Beispiel: Ausgabe des Arrays pd

```
public class PD {  
  
    public static void main(String[] args) {  
        int[] pd = { 1, 2, 4, 8, 16, 32, 64, 128 };  
  
        for (int i = 0; i < pd.length; i++) {  
            System.out.println(pd[i]);  
        }  
    }  
}
```



```
> java PD  
1  
2  
4  
8  
16  
32  
64  
128
```



# Beispiel: Summe des Arrays pd

```
public class PDSum {  
  
    public static void main(String[] args) {  
        int[] pd = { 1, 2, 4, 8, 16, 32, 64, 128 };  
        int sum = 0;  
        for (int i = 0; i < pd.length; i++) {  
            sum = sum + pd[i];  
        }  
        System.out.println("Sum: " + sum);  
    }  
}
```



```
> java PDSum  
Sum: 255
```

# Beispiel: Summe des Arrays pd mit alternativer Schleife

```
public class PDSumAlternate {  
  
    public static void main(String[] args) {  
        int[] pd = { 1, 2, 4, 8, 16, 32, 64, 128 };  
        int sum = 0;  
        for (int val : pd) {  
            sum = sum + val;  
        }  
        System.out.println("Sum: " + sum);  
    }  
}
```

**Alternative for-Schleife ohne Nutzung eines Index.**  
**Der Basisdatentyp des Arrays wird als Schleifenvariable verwendet.**



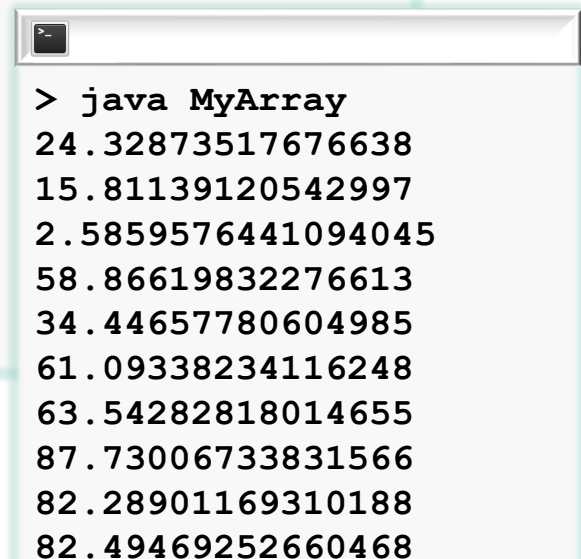
**Heusch 8.2.5**  
**Ratz 4.5.4.2**



```
> java PDSumAlternate  
Sum: 255
```

# Beispiel: Feld mit *reellen* Zufallszahlen

```
public class MyArray {  
  
    public static void main(String[] args) {  
        double[] nums = new double[10];  
        for (int i = 0; i < nums.length; i++) {  
            nums[i] = Math.random() * 100;  
        }  
        for (double d : nums) {  
            System.out.println(d);  
        }  
    }  
}
```



```
> java MyArray  
24.32873517676638  
15.81139120542997  
2.5859576441094045  
58.86619832276613  
34.44657780604985  
61.09338234116248  
63.54282818014655  
87.73006733831566  
82.28901169310188  
82.49469252660468
```

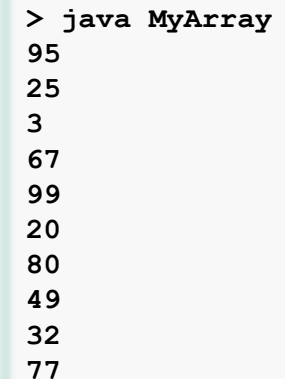
- Werte von `Math.random()`:  
Zufallszahlen aus dem Bereich  $[0,1)$

# Beispiel: Feld mit *ganzen* Zufallszahlen

```
import java.util.Random;

public class MyArray {

    public static void main(String[] args) {
        Random rnd = new Random(); // Random-Klasse
        int[] nums = new int[10];
        for (int i = 0; i < nums.length; i++) {
            nums[i] = rnd.nextInt(100); // aus 0..99
        }
        for (int z : nums) {
            System.out.println(z);
        }
    }
}
```



```
> java MyArray
95
25
3
67
99
20
80
49
32
77
```

# Beispiel: Größter Wert im Array bestimmen

```
public class ArrayGreat {  
  
    public static void main(String[] args) {  
        // Deklaration und Initialisierung des Arrays  
        double[] nums = new double[10];  
        for (int i = 0; i < nums.length; i++) {  
            nums[i] = Math.random() * 100;  
        }  
        // Grössten Wert bestimmen  
        int max = 0; // Index des grössten Elementes  
        for (int i = 1; i < nums.length; i++) {  
            if (nums[max] < nums[i]) {  
                max = i;  
            }  
        }  
        System.out.println("Maximum: " + nums[max]);  
    }  
}
```



```
> java ArrayGreat  
Maximum: 98.13110067829804
```

# Beispiel zur Deklaration und Initialisierung eines Arrays

```
public class MyArray1 {  
  
    public static void main(String[] args) {  
        int[] a1 = { 1, 2, 3, 4, 5 };  
        int[] a2;  
        for (int i = 0; i < a1.length; i++)  
            System.out.println(a1[i]);  
        for (int i = 0; i < a2.length; i++) {  
            System.out.println(a2[i]);  
        }  
    }  
}
```

- Ergibt (Compiler-)Fehler, da `a2` nicht initialisiert wurde

# Beispiel: Zuweisung von Arrays

```
public class MyArray1 {  
  
    public static void main(String[] args) {  
        int[] a1 = { 1, 2, 3, 4, 5 };  
        int[] a2;  
        a2 = a1; // Zusätzlich: Kopieren Inhalt von a1 nach a2  
                 // d.h. Referenzkopie (by reference)  
        for (int i = 0; i < a1.length; i++)  
            System.out.println(a1[i]);  
        for (int i = 0; i < a2.length; i++)  
            System.out.println(a2[i]);  
    }  
}
```

? Was wird ausgegeben?

# Beispiel: Kopieren von Arrays (by reference)

```
int[] a = { 1, 2, 4, 8, 16 };
System.out.print("a: ");
for (int i = 0; i < a.length; i++)
    System.out.print(a[i] + " ");
System.out.println();
int[] b = { 32, 64, 128, 256, 512, 1024 };
System.out.print("b: ");
for (int i = 0; i < b.length; i++)
    System.out.print(b[i] + " ");
System.out.println();
a = b;
System.out.print("a: ");
for (int i = 0; i < a.length; i++)
    System.out.print(a[i] + " ");
System.out.println();
a[0] = 33;
System.out.print("a: ");
for (int i = 0; i < a.length; i++)
    System.out.print(a[i] + " ");
System.out.println();
System.out.print("b: ");
for (int i = 0; i < b.length; i++)
    System.out.print(b[i] + " ");
```

Ausgabe

```
a: 1 2 4 8 16
b: 32 64 128 256 512 1024
a: 32 64 128 256 512 1024
a: 33 64 128 256 512 1024
b: 33 64 128 256 512 1024
```

## Referenzdatentypen


Die komplexeren Datentypen von Java sind Objekte und Arrays. Sie werden als „Referenztypen“ bezeichnet, weil sie „per Referenz“ („by reference“) verarbeitet werden.

Im Gegensatz dazu werden die primitiven Datentypen „by value“, also „per Wert“ verarbeitet.



# Beispiel: Kopieren von Variablen (by value)

```
int a = 1;  
System.out.println("a: " + a);  
  
int b = 2;  
System.out.println("b: " + b);  
  
a = b;  
System.out.println("a: " + a);  
System.out.println("b: " + b);  
  
a = 3;  
System.out.println("a: " + a);  
System.out.println("b: " + b);
```

  
a: 1  
b: 2  
a: 2  
b: 2  
a: 3  
b: 2

# Beispiel: Kopieren von Arrays (by value)

```
int[] a = { 1, 2, 4, 8, 16 };
System.out.print("a: ");
for (int i = 0; i < a.length; i++)
    System.out.print(a[i] + " ");
System.out.println();
int[] b = {32,64,128,256,512,1024};
System.out.print("b: ");
for (int i = 0; i < b.length; i++)
    System.out.print(b[i] + " ");
System.out.println();
System.arraycopy(a,0,b,0,a.length);
System.out.print("a: ");
for (int i = 0; i < a.length; i++)
    System.out.print(a[i] + " ");
System.out.println();
System.out.print("b: ");
for (int i = 0; i < b.length; i++)
    System.out.print(b[i] + " ");
System.out.println();
```

```
a[0] = 33;
System.out.print("a: ");
for (int i = 0; i < a.length; i++)
    System.out.print(a[i] + " ");
System.out.println();
System.out.print("b: ");
for (int i = 0; i < b.length; i++)
    System.out.print(b[i] + " ");
System.out.println();
```

## Ausgabe

```
a: 1 2 4 8 16
b: 32 64 128 256 512 1024
a: 1 2 4 8 16
b: 1 2 4 8 16 1024
a: 33 2 4 8 16
b: 1 2 4 8 16 1024
```

# System.arraycopy (Package `java.lang`)

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

- Kopiert vom spezifizierten Quell-Array `src` in das Ziel-Array `dest`.  
Kopiert ab der angegebenen Position der Quelle (`srcPos`) in Elemente ab der angegebene Position des Ziels (`destPos`).
- `IndexOutOfBoundsException` und unverändertes Ziel bei Überschreitung der Array-Grenzen, d.h.
  - falls `srcPos` negativ
  - falls `destPos` negativ
  - falls `length` negativ
  - falls `srcPos+length` größer als `src.length`
  - falls `destPos+length` größer als `dest.length`

# Beispiel: Überschreitung der Arraygrenzen

```
int[] a = { 1, 2, 4, 8, 16 };
System.out.print("a: ");
for (int i = 0; i < a.length; i++)
    System.out.print(a[i] + " ");
System.out.println();
int[] b = { 32, 64, 128, 256, 512, 1024 };
System.out.print("b: ");
for (int i = 0; i < b.length; i++)
    System.out.print(b[i] + " ");
System.out.println();
System.arraycopy(b, 0, a, a.length, b.length);
System.out.print("a: ");
for (int i = 0; i < a.length; i++)
    System.out.print(a[i] + " ");
System.out.println();
System.out.print("b: ");
for (int i = 0; i < b.length; i++)
    System.out.print(b[i] + " ");
System.out.println();
```

```
a[0] = 33;
System.out.print("a: ");
for (int i = 0; i < a.length; i++)
    System.out.print(a[i] + " ");
System.out.println();
System.out.print("b: ");
for (int i = 0; i < b.length; i++)
    System.out.print(b[i] + " ");
System.out.println();
```



## Fehler zur Laufzeit!

*IndexOutOfBoundsException*

a.length ist hier 5

b.length ist hier 6

a.length+b.length ist 11 und damit größer als a.length (5)

**Keine Erweiterung eines bestehenden Arrays möglich!**

## Beispiel: Sicheres Kopieren (by value)

```
/*  
 * Sicheres Kopieren durch Erzeugen  
 * eines weiteren Arrays  
 */  
int[] a ..., b = ...;  
int[] c = new int[a.length + b.length];  
System.arraycopy(a, 0, c, 0, a.length);  
System.arraycopy(b, 0, c, a.length, b.length);  
// und dann ggf.  
b = c;
```

# Beispiel: Sortieren eines Arrays

```
double[] nums = { 1.2, 4.0, 0.9, 2.7 };

/*
 * Idee: Suche das kleinste Elementes im Feld und
 * setze es an die erste (noch) unsortierte Stelle
 */
for (int i = 0; i < nums.length; i++) {
    int min = i;
    // Suche kleinstes Element zwischen i und Arrayende
    for (int j = i; j < nums.length; j++)
        if (nums[j] < nums[min])
            min = j;
    // Tausche kleinstes Element mit dem an Stelle i
    double tmp = nums[i];
    nums[i] = nums[min];
    nums[min] = tmp;
}
```

# ÜBUNG

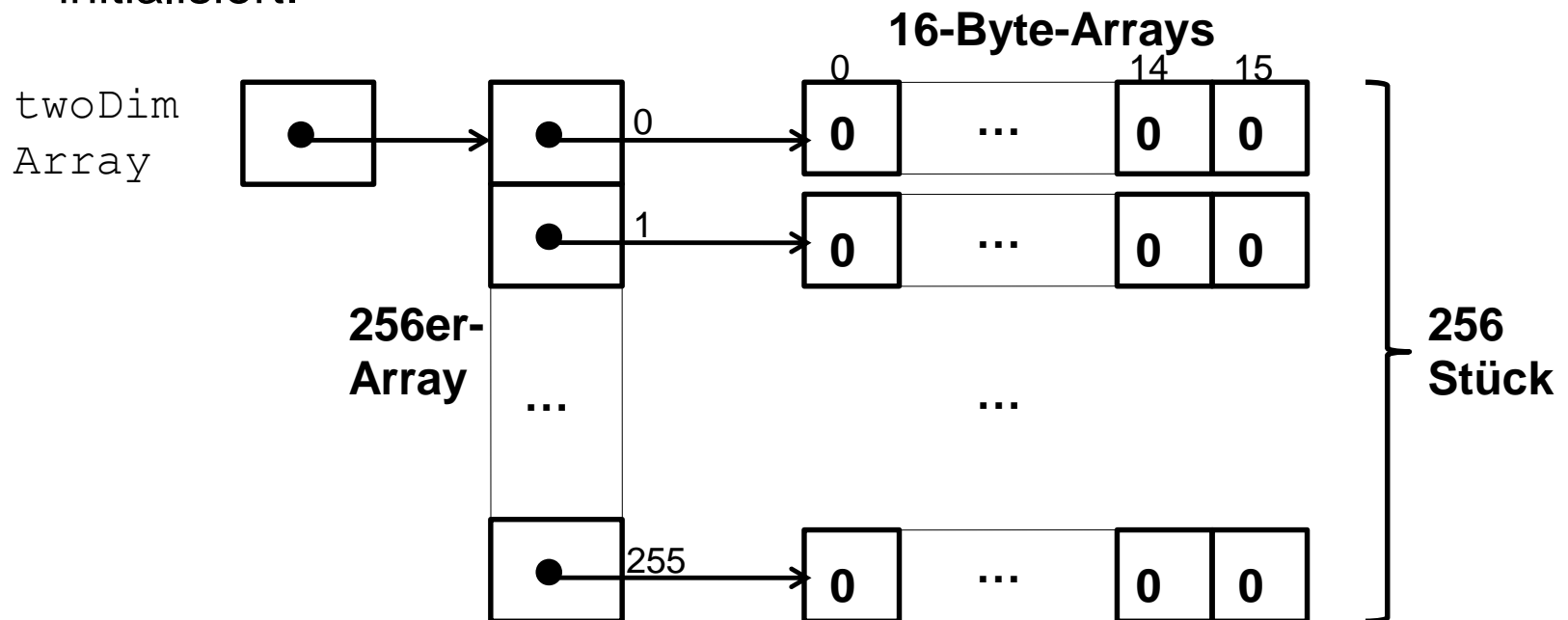
# Arrays: Mehrdimensionale Arrays (1)

- Mehrdimensionale Arrays sind in Java in Form von Arrays von Arrays realisiert.
- Beispiel: Zweidimensionales `byte`-Array
- `byte[][] twoDimArray;`
  - Deklaration der Variablen `twoDimArray` als vom Typ `byte[][]`, d.h. zweidimensionales Array (Array-von-Arrays) mit Komponenten vom Datentyp `byte`.
- `byte[][] twoDimArray = new byte[256][];`
  - Deklaration der Variablen `twoDimArray` wie oben.
  - Dynamische Bereitstellung eines Arrays mit 256 Komponenten.
  - Jedes Element dieses Arrays ist vom Typ `byte[]` - ein eindimensionales Array von `bytes` (das noch nicht angelegt ist!).



## Mehrdimensionale Arrays (2)

- `byte[][] twoDimArray = new byte[256][16];`
  - Deklaration der Variablen `twoDimArray` und dynamische Bereitstellung eines 256er Arrays wie oben.
  - Zusätzlich Bereitstellung von 256 16-Byte-Arrays.
  - Die 16 Bytes dieser 256 Arrays werden mit dem Standardwert 0 initialisiert.



# Arrays: Mehrdimensionale Arrays (3)

- Nicht erlaubt:

- „Höhere“ Dimension nicht definiert.

```
byte[][] twoDimArray = new byte[][16];
```

- Erlaubt:

- Beliebige Anzahl der Dimensionen :

```
int[][][][][] fiveDimArray = new int[3][4][5][6][7];
```

- Nicht „rechteckige“ Form von Arrays („nicht vollständig besetzt“):

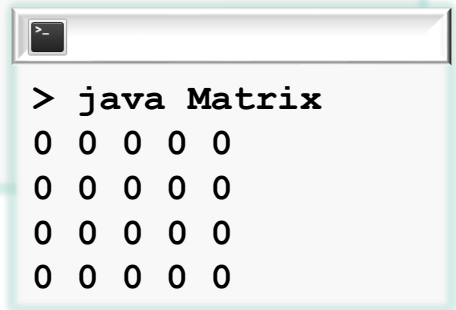
```
int[][] twoDimArray = { {1,2}, {3,4,5}, {5,6,7,8} };
```

- Dimension des Array nach dem Variablennamen angeben:

```
byte twoDimArray[][] = new byte[16][];
```

# Beispiel: Generieren und Ausgabe einer Matrix

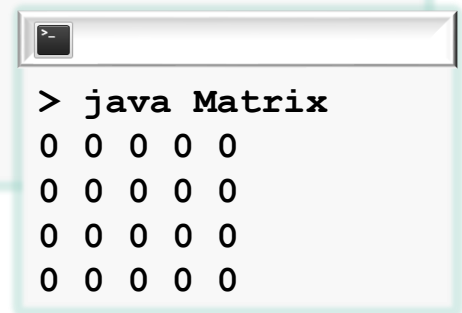
```
public class Matrix {  
  
    public static void main(String[] args) {  
        int[][] matrix = new int[4][5];  
        // Ausgabe der Matrix  
        for (int i = 0; i < matrix.length; i++) {  
            for (int j = 0; j < matrix[i].length; j++) {  
                System.out.print(matrix[i][j] + " ");  
            }  
            System.out.println();  
        }  
    }  
}
```



```
> java Matrix  
0 0 0 0 0  
0 0 0 0 0  
0 0 0 0 0  
0 0 0 0 0
```

# Beispiel: Generieren und Ausgabe einer Matrix mit alternativer for-Schleife

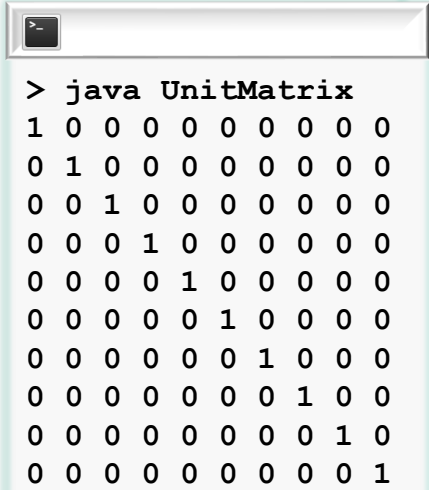
```
public class Matrix {  
  
    public static void main(String[] args) {  
        int[][] matrix = new int[4][5];  
        // Ausgabe der Matrix  
        for (int[] zeile : matrix) {  
            for (int element : zeile) {  
                System.out.print(element + " ");  
            }  
            System.out.println();  
        }  
    }  
}
```



```
> java Matrix  
0 0 0 0 0  
0 0 0 0 0  
0 0 0 0 0  
0 0 0 0 0
```

# Beispiel: Generieren und Ausgabe der Einheitsmatrix

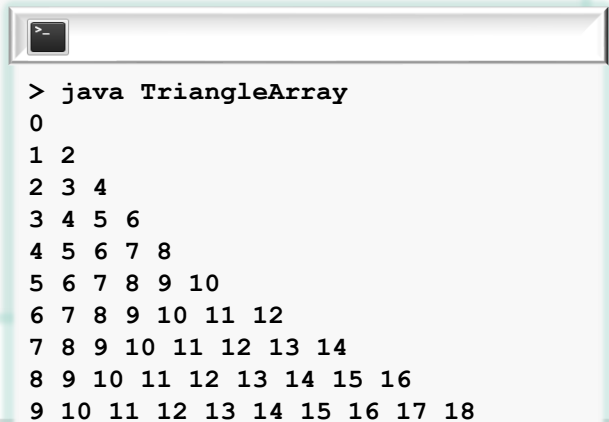
```
public class UnitMatrix {  
  
    public static void main(String[] args) {  
        int[][] matrix = new int[10][10];  
        // Generieren der Einheitsmatrix  
        for (int i = 0; i < matrix.length; i++) {  
            for (int j = 0; j < matrix[i].length; j++) {  
                if (i == j)  
                    matrix[i][j] = 1;  
                // else matrix[i][j] = 0;  
            }  
        }  
        for (int i = 0; i < matrix.length; i++) {  
            for (int j = 0; j < matrix[i].length; j++) {  
                System.out.print(matrix[i][j] + " ");  
            }  
            System.out.println();  
        }  
    }  
}
```



```
> java UnitMatrix  
1 0 0 0 0 0 0 0 0 0  
0 1 0 0 0 0 0 0 0 0  
0 0 1 0 0 0 0 0 0 0  
0 0 0 1 0 0 0 0 0 0  
0 0 0 0 1 0 0 0 0 0  
0 0 0 0 0 1 0 0 0 0  
0 0 0 0 0 0 1 0 0 0  
0 0 0 0 0 0 0 1 0 0  
0 0 0 0 0 0 0 0 1 0  
0 0 0 0 0 0 0 0 0 1
```

# Beispiel: Nicht-rechteckiges zweidimensionales Array

```
public class TriangleArray {  
  
    public static void main(String[] args) {  
        short[][] triangle = new short[10][];  
        for (int i = 0; i < triangle.length; i++) {  
            triangle[i] = new short[i + 1];  
            for (int j = 0; j < triangle[i].length; j++) {  
                triangle[i][j] = (short) (i + j);  
            }  
        }  
        for (int i = 0; i < triangle.length; i++) {  
            for (int j = 0; j < triangle[i].length; j++) {  
                System.out.print(triangle[i][j] + " ");  
            }  
            System.out.println();  
        }  
    }  
}
```



```
> java TriangleArray  
0  
1 2  
2 3 4  
3 4 5 6  
4 5 6 7 8  
5 6 7 8 9 10  
6 7 8 9 10 11 12  
7 8 9 10 11 12 13 14  
8 9 10 11 12 13 14 15 16  
9 10 11 12 13 14 15 16 17 18
```

# Arrays: Zusammenfassung (1)

- Arrays haben feste Anzahl von Komponenten (einmal festgelegt nicht erweiterbar)
- Beliebiger, aber für alle Komponenten gleicher Typ
- Indizes der Komponenten gehen von  $0 \dots \text{length}-1$
- Mehrdimensionale Arrays möglich
- Arrays sind Objekte.  
Dennoch gibt es Unterschiede zu »normalen« Objekten.

# Arrays: Zusammenfassung (2)

- Gemeinsamkeiten von Objekten und Arrays (Ausblick):
  - Arrays werden wie Objekte grundsätzlich dynamisch angelegt und am Ende ihrer Verwendung automatisch vom Garbage Collector entfernt.
  - Arrays sind wie Objekte Referenztypen.
  - Arrays sind wie alle Klassen implizit Unterklassen der Klasse `Object`.
  - Array-Elemente, die primitive Datentypen sind, werden wie die Datenelemente von Objekten auch stets automatisch initialisiert.
- Unterschiede von Objekten und Arrays (Ausblick):
  - Arrays haben keine Konstruktoren. Statt dessen gibt es für Arrays eine spezielle Syntax des `new`-Operators.
  - Keine Unterklassen von Arrays möglich.