

# Grundlagen der Programmierung



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Vorlesungsskript zum Sommersemester 2020

4. Vorlesung (11. Mai 2020)



**Dr. Jin Gerlach**  
**Christian Olt**

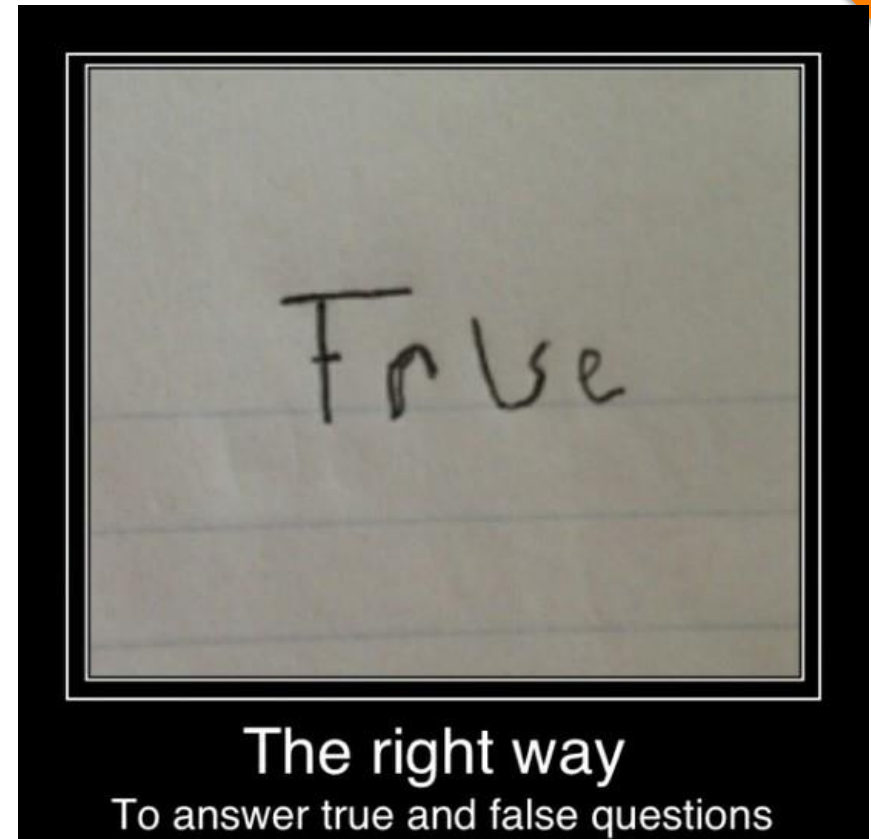
Fachgebiet Wirtschaftsinformatik | Software & Digital Business  
Fachbereich Rechts- und Wirtschaftswissenschaften  
Technische Universität Darmstadt

# True oder false?



ISCHE  
TAT  
T  
D  
<http://pingo.upb.de>  
#9456

1. Die Bedingung einer if-Anweisung muss einem Boolean-Ausdruck entsprechen.
2. Das Ergebnis der Anwendung eines Vergleichsoperators ist ein Wahrheitswert (true bzw. false).
3. Das Ergebnis der Anwendung eines boolschen Operators ist ein Wahrheitswert (true bzw. false).
4. Die Default-Marke eines Switch-Statements bestimmt, welche Befehle in jedem Fall (für jeden Case) ausgeführt werden sollen.





# if-Anweisung: Bedingungen

## ■ Syntax:

- `if ( Bedingung ) {Anweisung}`
- `if ( Bedingung ) {Anweisung1} else {Anweisung2}`

- Die Bedingung einer if-Anweisung muss einem Boolean-Ausdruck entsprechen und einen Wahrheitswert (true oder false) als Ergebnis haben.
- Dabei kann der Boolean-Ausdruck auch aus mehreren verknüpften Ausdrücken und Befehlen zusammengesetzt sein.

## ■ Beispiele:

- `if (true) {Anweisung}`
- `if (x < 4) {Anweisung}`
- `if (false || ! false && true) {Anweisung}`
- `if (x < 4 || ! false && true) {Anweisung}`



# Beispiele: Boolesche Ausdrücke

	<b>( x &gt; 3 )</b>	<b>&amp;&amp;</b>	<b>(x &lt;= 5)</b>	
<b>für x = 1</b>	false	&&	true	= false
<b>für x = 4</b>	true	&&	true	= true

<b>x = 5, b = true</b>					
<b>(x &gt;= 3)</b>	<b>  </b>	<b>(x &lt; 7)</b>	<b>&amp;&amp;</b>	<b>!b</b>	
true		true	&&	false	
true		false			= true

<b>x = 5, b = true</b>							
<b>(</b>	<b>(x &gt;= 3)</b>	<b>  </b>	<b>(x &lt; 7)</b>	<b>)</b>	<b>&amp;&amp;</b>	<b>!b</b>	
	true		true			false	
	true				&&	false	= false



# Alternativen: switch-Anweisung

- **Switch-Anweisungen** ermöglichen eine kompakte Schreibweise

- **Syntax:**

```
switch (Ausdruck) {  
    case Wert : Anweisung;  
                Anweisung;  
                break;  
  
    case Wert 2: Anweisung;  
    default : Anweisung;  
}
```

- Ausdruck ist vom Typ `int` (oder `char`)
- Das Programm springt zu der **Marke**, die mit dem berechneten Wert übereinstimmt
- Stimmt keine Marke mit dem Wert überein, springt das Programm zur **default-Marke**



# Beispiel: switch-Anweisung (Code)

```
public class Sample2 {  
    public static void main(String[] args) {  
        int note = 1;  
        String praed = "";  
        switch (note) {  
            case 1:  
                praed = "sehr gut";  
                break;  
            case 2:  
                ...  
            case 6:  
                praed = "ungenügend";  
                break;  
            default:  
                praed = "Die Eingabe ist keine Note!";  
                break;  
        }  
        System.out.println("Das Prädikat für die Note " +  
            note + " lautet " + praed + ".");  
    }  
}
```

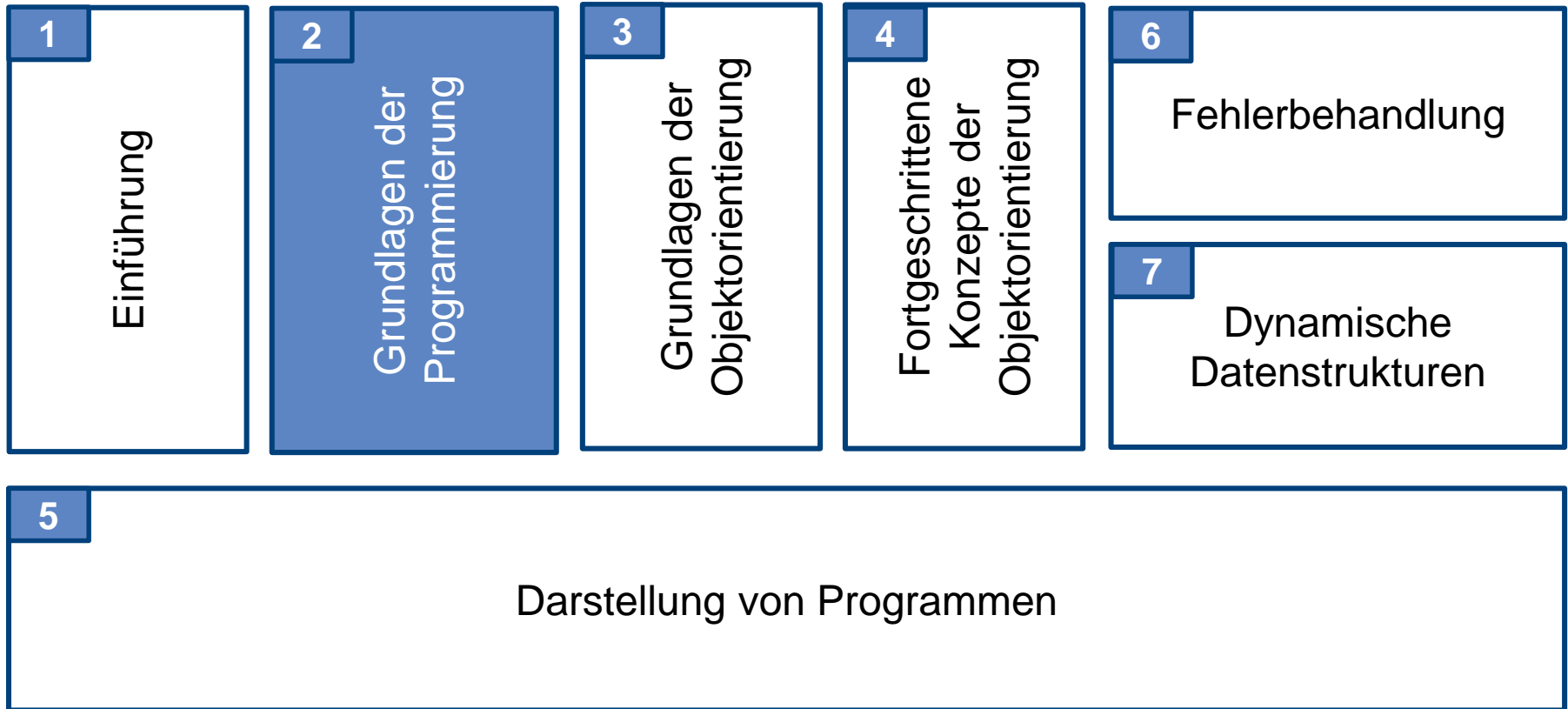


# Kontrollstrukturen

- Kontrollstrukturen dienen dazu, Programmteile unter bestimmten Bedingungen auszuführen.
- Zum Ausführen alternativer Programmteile (sog. **Verzweigungen**):
  - **if**-Anweisung
  - **if-else**-Anweisung
  - **switch**-Anweisung
- Neben der Verzweigung dienen **Schleifen** dazu, Programmteile mehrmals auszuführen:
  - **while**-Schleife
  - **do-while**-Schleife
  - **for**-Schleife



# Thematische Übersicht





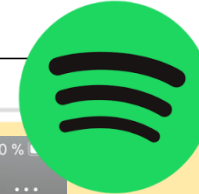
# Kapitel 2: Grundlagen der Programmierung

- Kontrollstrukturen für Wiederholungen: „while“-Schleife, „do-while“-Schleife und „for“-Schleife
- Arrays (ein- und mehrdimensional) als strukturierte Datentypen


## Lernziele:

- Die Kontrollstrukturen „while“-Schleife, „do-while“-Schleife und „for“-Schleife kennen und anwenden können
- Den strukturierten Datentyp „Array“ kennen sowie deklarieren, initialisieren und verwenden können

# Schleifen (Beispiel)



2. Mai - 8. Mai



Lösungsvorschlag 2. Üb

Telekom.de 09:37 100 %

Get Psyched Mix

FOLGEN

4.011 FOLLOWER • VON 12154539609

SHUFFLE

Herunterladen

You Give Love A Bad Name  
Bon Jovi • Bon Jovi Greatest Hits

Livin' On A Prayer  
Bon Jovi • Bon Jovi Greatest Hits

I'm Gonna Be (500 Miles)  
The Proclaimers • Finest

You Give Love A Bad Name • Bon Jovi  
Verfügbare Geräte

Start Browse Suche Radio Bibliothek

alles nichts.  
us Mozart)

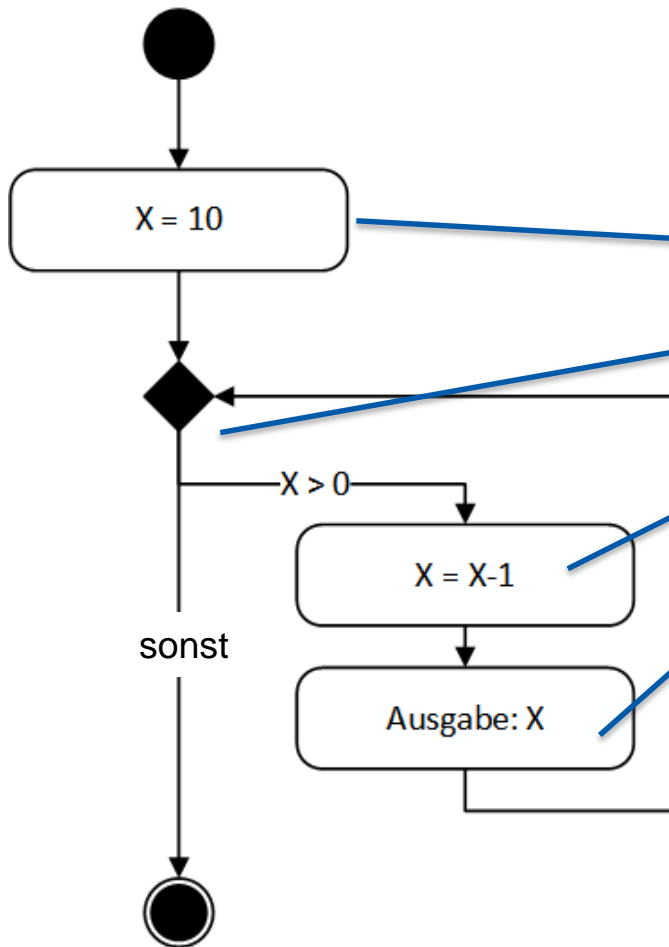
gutezitate.com

- **Syntax:** `while ( Bedingung ) {Anweisung} ;`
- Solange `Bedingung` (d.h. ein Boolean-Ausdruck) gleich `true` ist, wird die `Anweisung` ausgeführt.
- Beispiel:

```
int n = 1;
while (n <= 3) {
    n = n + 1;
}
System.out.println(n);
// nach Durchlauf der Schleife ist n = 4
```

- Hinweis: Auch Schleifen können zusammengesetzte Ausdrücke als `Bedingung` sowie Blockanweisungen beinhalten.

# Beispiel: while-Schleife



```
public class Sample3 {  
    public static void main(String[]  
        args) {  
        int X = 10;  
        while (X > 0) {  
            X = X-1;  
            System.out.println(X);  
        }  
    }  
}
```

# Aufgabe: Was ist hier anders?



## Programm 1: Was ist die Ausgabe des Programms?

```
public static void main(String[] args) {  
    int n = 1;  
    while (n <= 3)  
        n++; // n = n + 1;  
    System.out.println (n);  
}
```

## Programm 2: Was ist die Ausgabe des Programms?

```
public static void main(String[] args) {  
    int n = 1;  
    while (n <= 3) {  
        n++; // n = n + 1;  
        System.out.println (n);  
    }  
}
```

# Aufgabe: verschachtelte Schleifen



<http://pingo.upb.de>  
#9456

Wie lautet die Bildschirmausgabe des Programms?

```
public class Schleifen {  
  
    public static void main(String[] args) {  
        int i = 2;  
        while (i > 0) {  
            int j = 1;  
            System.out.println("a");  
            while (j <= 2) {  
                System.out.println("b");  
                j = j + 1;  
            }  
            i = i - 1;  
        }  
    }  
}
```

- Eine „do-while“-Schleife ist eine sogenannte „fußgesteuerte“ Schleife (Versus der kopfgesteuerten „while“-Schleife)
- **Syntax:** `do {Anweisung} while ( Bedingung ) ;`
- Wiederhole Anweisung, solange Bedingung (d.h. ein Boolean-Ausdruck) gleich `true` ist

- Beispiel:

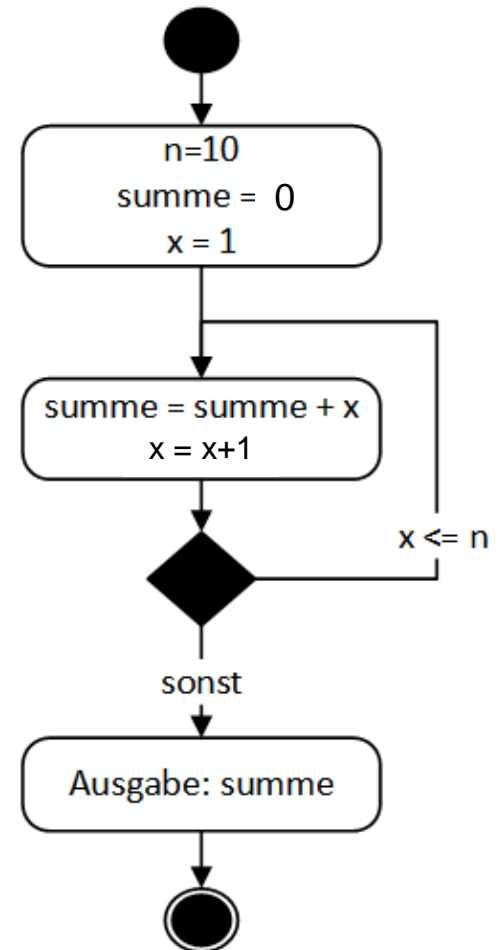
```
int n = 1;
do {
    n = n + 1;
} while (n <= 3);
// nach Durchlauf der Schleife ist n = 4
```

# Beispiel: do-while-Schleife



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
public class Sample4 {  
    public static void main(  
        String[] args) {  
        int n = 10;  
        int summe = 0;  
        int x = 1;  
        do {  
            summe = summe + x;  
            x = x + 1;  
        } while (x <= n);  
        System.out.println(summe);  
    }  
}
```





## Aufgabe:



vs.



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Was ist die Ausgabe des folgenden Programms?

```
public static void main(String[] args) {  
    boolean Bedingung = false;  
    while (Bedingung) {  
        System.out.println("Kopfgesteuert");  
    }  
    do {  
        System.out.println("Fußgesteuert");  
    } while (Bedingung);  
}
```

# „for“-Schleife

- Eine „for“-Schleife ist eine „kopfgesteuerte“ Schleife.
- **Syntax:** `for (Initialisierung; Bedingung; Aktualisierung)  
{Anweisung}`
- **Vorgehensweise:**
  1. Führe Initialisierung aus
  2. Solange Bedingung wahr ist, führe Anweisung aus
  3. Nach jeder Ausführung von Anweisung führe Aktualisierung aus
- **Beispiel:**

```
for (int i=10; i > 0; i--) {  
    System.out.println(i);  
}
```
- **Hinweis (Wiederholung):**

```
i++; // Abkürzung für i = i + 1  
j--; // Abkürzung für j = j - 1
```



# „for“-Schleifen vs. „while“-Schleifen

- Jede for-Schleife kann in eine while-Schleife transformiert werden!

- Die **for-Schleife**

```
for (Initialisierung; Bedingung; Aktualisierung)  
    {Anweisung}
```


ist **äquivalent** zu der **while-Schleife**

```
Initialisierung ;  
while (Bedingung) {  
    Anweisung ;  
    Aktualisierung ;  
}
```

# Schleifen (Beispiel)



2. Mai - 8. Mai



Lösungsvorschlag 2. Üb.

Get Psyched Mix

FOLGEN

4.011 FOLLOWER • VON 12154539609

SHUFFLE

Herunterladen

You Give Love A Bad Name  
Bon Jovi • Bon Jovi Greatest Hits

Livin' On A Prayer  
Bon Jovi • Bon Jovi Greatest Hits

I'm Gonna Be (500 Miles)  
The Proclaimers • Finest

You Give Love A Bad Name • Bon Jovi  
Verfügbare Geräte

Start Browse Suche Radio Bibliothek

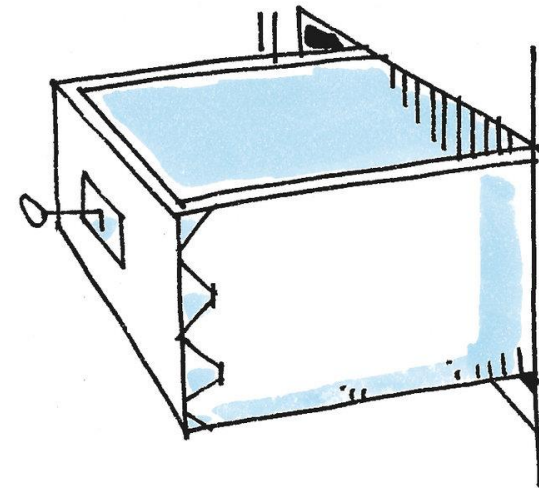
alles nichts.  
us Mozart)

gutezitate.com



# Variablen und Arten von Datentypen

- **Variablen** sind Behälter für genau einen Wert und es gilt:
  - Variablen haben einen **Datentyp** (z. B. `int`)
  - Variablen haben einen **Bezeichner** (z. B. `x` oder `besteVariableDerWelt`)
  - Variablen haben einen **Wert** (z. B. `42`)
- Wir unterscheiden **zwei Arten von Datentypen**:
  - **Einfache** Datentypen:
    - Wahrheitswert (`boolean`) → `true`, `false`
    - Einzelzeichen (`char`) → z. B.: `'a'`, `'b'`, `'1'`, `'2'`, `'%'`, `'_'`
    - Numerische Datentypen
      - Ganzzahlige Datentypen (`byte`, `short`, `int`, `long`) → z. B. `123`
      - Gleitkommatypen (`float`, `double`) → z. B. `1.25`
  - **Strukturierte** Datentypen (auch: Referenz-Datentypen)



# Strukturierte Datentypen: Arrays

- Ein Array (Feld) ist ein strukturierter Datentyp
- Ein Array fasst mehrere Werte (gleichen Datentyps) zu einer Einheit zusammen. Er ist mit einem „Regal“ vergleichbar, in dem die Plätze durchnummeriert (d.h. mit einem Index versehen) sind.
- Zu jedem beliebigen Datentyp  $T$  existiert ein zugehöriger Array-Datentyp  $T[]$ , z. B. `int[]`, `char[]`, ...

Index	0	1	2	3
Datentyp	<code>int</code>	<code>int</code>	<code>int</code>	<code>int</code>
Inhalt	40	50	20	5



# Arrays erzeugen

- Ein Array muss erst erzeugt werden, bevor er verwendet werden kann

- **Statische Erzeugung:**

```
int[] zahlen =           // Deklaration
    {40, 50, 20, 5};      // Statische Erzeugung
```

- **Dynamische Erzeugung:**

```
int[] zahlen =           // Deklaration
    new int[4];           // Dynamische Erzeugung mit Defaultwerten
zahlen[0] = 40;
zahlen[1] = 50;
...
```

- Das erzeugte Array wird zur Initialisierung der Array-Variablen verwendet

# Zugriff auf Array-Elemente

- Der Zugriff auf ein Element erfolgt über einen ganzzahligen Index
- In einem Array der Länge  $n$  hat das erste Element den Index 0 und das letzte Element den Index  $n-1$
- Beispiel:

```
int[] zahlen = new int[4];  
zahlen[0] = 40;  
zahlen[1] = 50;  
zahlen[2] = 20;  
zahlen[3] = zahlen[2] - zahlen[0];
```

Index	0	1	2	3
Datentyp	int	int	int	int
Inhalt	40	50	20	-20



# Beispiel: Arrays als Fehlerquelle

Die folgenden Array-Beispiele enthalten jeweils einen Fehler!

1

```
int[] zahlen;  
zahlen[0] = 1;  
zahlen[1] = 42;
```

2

```
int[] zahlen =  
    new int[1, 2, 3];
```

3

```
int[] zahlen = 42;
```

4

```
int[] zahlen =  
    new int[3];  
zahlen[999] = 42;
```

5

```
int[] zahlen =  
    {true, false}
```

# Länge eines Arrays

- Über `array.length` kann die Länge eines Arrays ermittelt werden
- Beispiel:

Index	0	1	2	3
Datentyp	int	int	int	int
Inhalt	40	50	20	-20

```
int[] zahlenNeu = {40, 50, 20, -20};  
System.out.println(zahlenNeu.length);  
// Ausgabe: 4
```

- Eindimensionale Arrays werden typischerweise in einer for-Schleife durchlaufen
- Bildschirmausgabe der Elemente eines eindimensionalen Arrays:

```
int[] primes = {2,3,5,7,11,13,17,19,23,29};  
for (int i=0; i < primes.length; i++) {  
    System.out.println(primes[i]);  
}
```

# Aufgabe: Arrays und Schleifen



<http://pingo.upb.de>  
#9456

Was ist die Ausgabe des Programms?

```
public static void main(String[] args){  
    int[] numbers = {10,2,40,90,100,20};  
    int value = numbers[0];  
    for(int i=0; i < numbers.length; i++){  
        if(numbers[i] > value){  
            value = numbers[i];  
        }  
    }  
    System.out.println(value);  
}
```



# Mehrdimensionale Felder

- Mehrdimensionale Felder sind Felder von Feldern
- Felder können beliebig geschachtelt werden
- Bei Deklaration, Erzeugung und Benutzung müssen entsprechend viele Paare von eckigen Klammern angegeben werden
- Beispiel:

```
int[][] nDim = { {10, 20, 5}, {1, 2} };
```



# Beispiel: mehrdimensionales Array

```
int[][] nDim = { {10, 20, 5}, {1, 2} };  
System.out.println(nDim[0][0]); // 10  
System.out.println(nDim[0][1]); // 20  
System.out.println(nDim[0][2]); // 5  
System.out.println(nDim[1][0]); // 1  
System.out.println(nDim[1][1]); // 2
```

	Spalte 0	Spalte 1	Spalte 2
Zeile 0	10	20	5
Zeile 1	1	2	



# Mehrdimensionale Arrays und Schleifen

- Mehrdimensionale Arrays werden typischerweise in mehreren verschachtelten for-Schleifen durchlaufen
- Bildschirmausgabe eines zweidimensionalen Arrays (einer Matrix):

```
int [][] matrix = { {1,0,0}, {0,1,0}, {0,0,1} };  
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();  
}
```