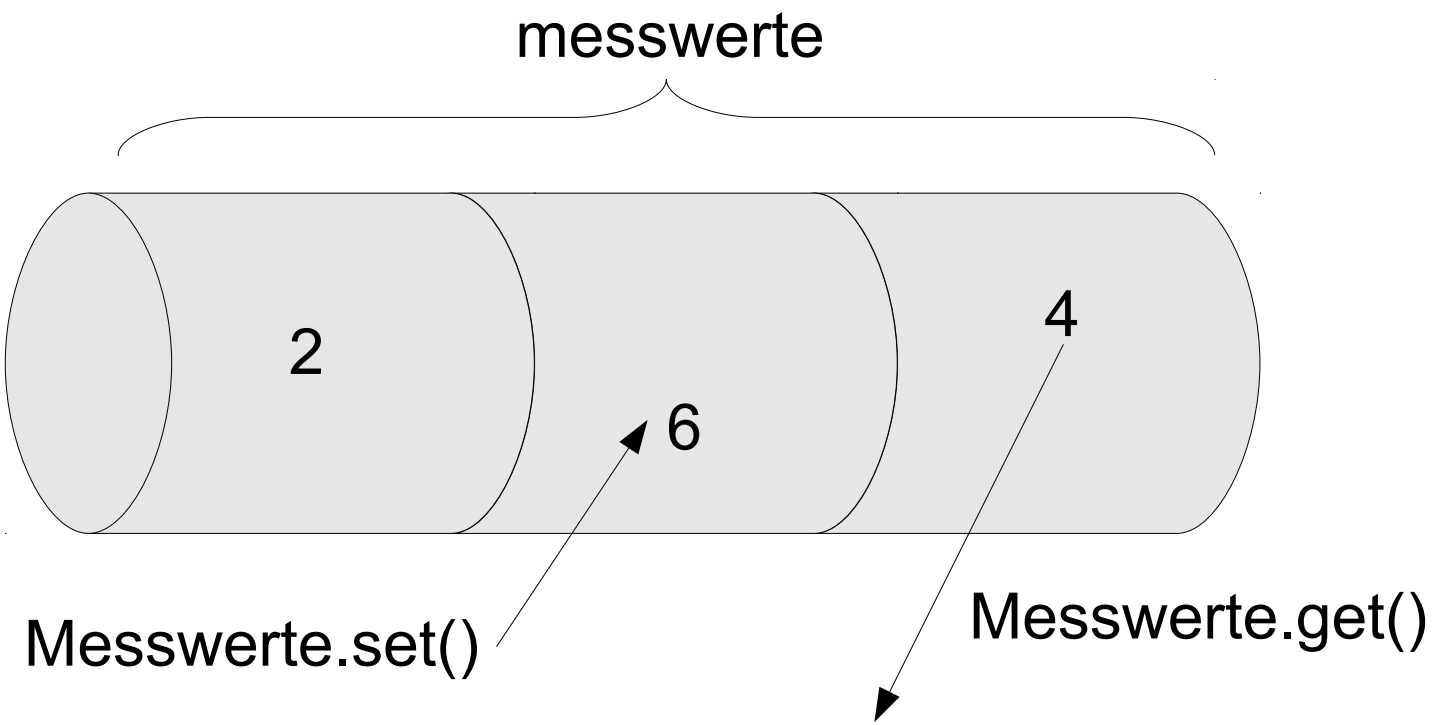
Java - Arrays

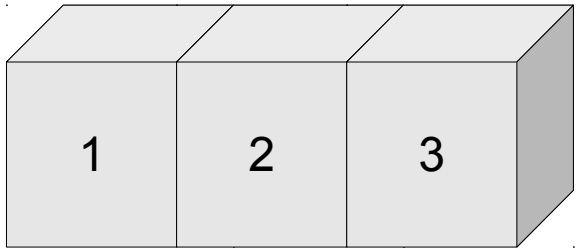


Array (Felder, Vektoren)

- Zusammenfassung von vielen Werten gleichen Datentyps.
- Gruppieren von Variablen zu einem Thema (zum Beispiel Temperaturwerte eines Monats).
- Speicherung einer festen Anzahl von Werten gleichen Datentyps.
- Ein- oder mehrdimensional.

Eindimensionale Arrays

- Folge von Werten gleichen Datentyps.
- In einem Container befinden sich mehrere kleinere Boxen gleicher Größe, aber unterschiedlichen Inhalts. Die Größe und die Art des Inhalts ist abhängig vom Datentyp.
- Sammlung von Werten zu dem gleichen Thema, wie zum Beispiel durchschnittliche Temperaturen pro Monat. Der Monat wird durch die Position im Array symbolisiert. Der Wert an der Position bildet die Temperatur an.



… deklarieren

```
int[] monat;
double[] messwerte;
char[] alphabet;
```

- Die eckigen Klammern kennzeichnen einen Container, der viele kleine Boxen enthält.
- Durch die Angabe des Datentyps wird die Größe der einzelnen Boxen im Container festgelegt.
- Der Speicherplatz für den Container ist noch nicht reserviert. Die Gesamtzahl der Boxen im Container ist noch nicht bekannt.
- Der Container hat einen eindeutigen Namen.

Deklaration von eindimensionalen Arrays

`int`	`[`	`]`		`monat`	`;`
`Datentyp`	`[`	`]`		`feldname`	`;`

- Die Variable verweist auf ein Feld von x Werten vom Datentyp ...
- Jeder primitiver Datentypen kann für ein Feld genutzt werden.
- Der Feldname ist ein Platzhalter für den Speicherort des ersten Elementes im Feld.
- Der Feldname ist frei wählbar.

Weitere Möglichkeit

```
int monat[];
double messwerte[];
char alphabet[];
```

- Die Variable ist vom Typ „Feld“ und speichert Werte vom Datentyp ...
- Diese Möglichkeit wird nicht empfohlen.

Feldnamen

- Label für den Speicherort der ersten Box in einem Container.
- Jeder Name kommt nur einmal in einem Block vor. Der Block von Anweisungen beginnt und endet mit den geschweiften Klammern.
- Häufig wird die Pluralform von Dingen für benutzerdefinierte Namen genutzt. Zum Beispiel `temperatur` speichert einen Temperaturwert in eine Variable. Der Name `temperaturen` speichert x Temperaturwerte.

… initialisieren

```
monat = new int[11];
messwerte = new double[12];
alphabet = new char[26];
```

- Die Anweisung new erzeugt ein Feld von einer bestimmten Größe von einem Datentyp.
- Die Gesamtgröße des Containers und die Anzahl der Boxen im Container werden festgelegt.

Initialisierung von eindimensionalen Arrays

`monat`	`=`	`new`	`int`	`[`	`11`	`]`	`;`
`feldname`	`=`	`new`	`Datentyp`	`[`	`anzahl`	`]`	`;`

- Mit Hilfe von `new` wird Speicherplatz für x Felder reserviert.
- Die Anzahl der Felder wird in den eckigen Klammern angegeben. Die Anzahl kann nachträglich nicht verändert werden.
- Jedes Feld ist so groß, dass es einen Wert entsprechend des Datentyps speichern kann.

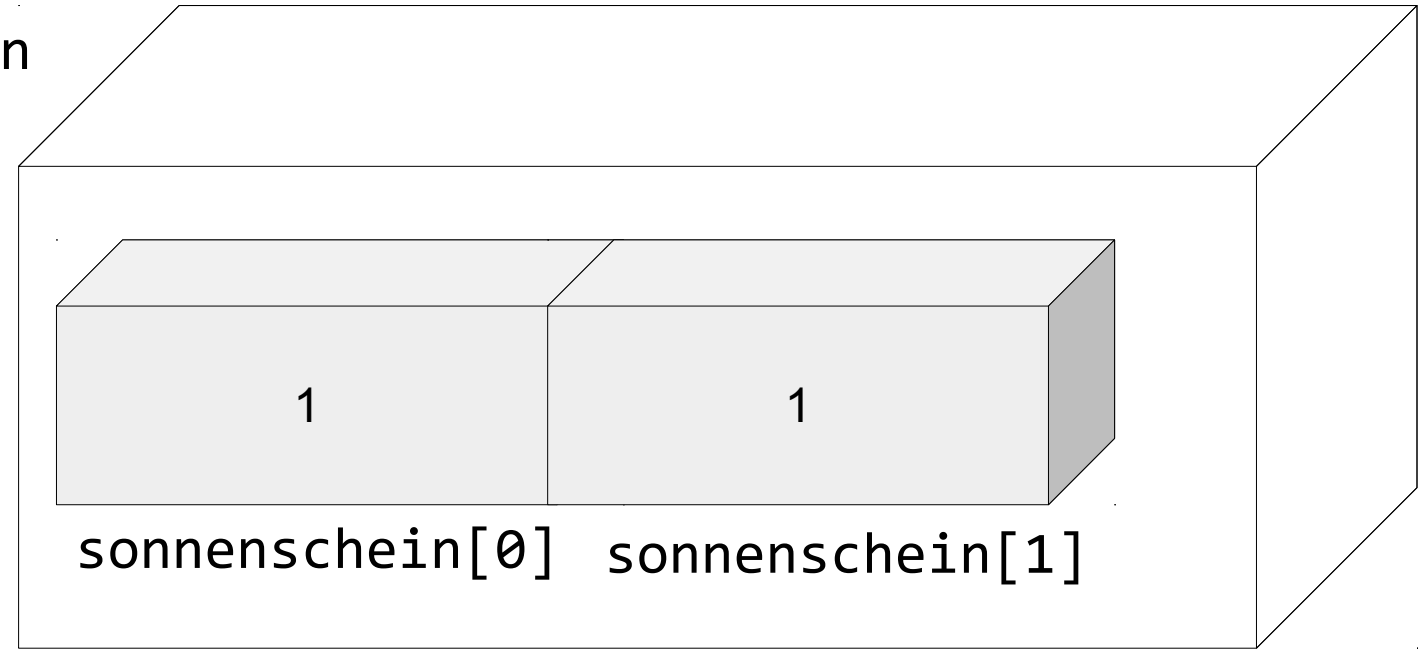
… deklarieren und initialisieren

```
double[] sonnenScheinDauer = new double[11];
```

- Die Variable `sonnenScheinDauer` wird als Feld vom Typ `double` deklariert.
- Die Anweisung `new` erzeugt ein Feld von einer bestimmten Größe von dem Datentyp double.
- Die Adresse des ersten Feldes im Speicher wird der Variablen `sonnenScheinDauer` übergeben

Beispiel

```
sonnenschein
[I@520ed128
```



Zuweisung von Werten

```
monat[0] = 1;
messwerte[1] = 0.3;
alphabet[pos] = 'a';
```

- Die erste Box im Container hat die Position 0. Die zweite Box hat die Position 1 und so weiter.
- Mit Hilfe des Zuweisungsoperators wird der Box ein Wert zugewiesen. Der Wert kann entsprechend des Datentyps interpretiert werden.

„Initialisierung“ der einzelnen Felder

`monat`	`[`	`0`	`]`	`=`	`1`	`;`
`feldname`	`[`	`pos`	`]`	`=`	`wert`	`;`

- Jede Box im Container wird durch einen Index bestimmt. Der Index legt die Position des Elements fest.
- Der Index folgt durch eckige Klammern begrenzt direkt dem Feldnamen.
- Die Index ist immer eine Ganzzahl.
- Der Index kann mit Hilfe einer Variablen oder eines Literals bestimmt werden.

Standardwerte

- Jedes Element in einem Array wird entsprechend des Datentyps mit einem Standardwert initialisiert.
- Felder vom Datentyp „Ganzzahl“ oder „Gleitkommazahl“ haben den Standardwert 0.
- Felder vom Datentyp `char` enthalten `' '`.

„Leeres“ char-Element

```
char[] alphabet;
int pos = 1;

alphabet = new char[26];
alphabet[pos] = 'a';

if((alphabet[2] == ' ') || (alphabet[2] == 0))
{
  alphabet[2] = 'b';
}
```

Erläuterung

- Jedes „char“ hat eine Ganzzahl als ASCII-Kodierung. Null kennzeichnet ein leeres Zeichen. Die Variable hat keinen definierten Inhalt.

```
(alphabet[2] == 0)
```

- Enthält das Element ein Leerzeichen?

```
(alphabet[2] == ' ')
```

Angabe der Werte bei der Deklaration

```
char[] wort = {'H', 'e', 'l', 'l', 'o'};
```

- Mit Hilfe des Zuweisungsoperators wird dem Array eine Liste von Werte übergeben.
- Die Liste von Werten ist durch die geschweiften Klammern begrenzt.
- Die einzelnen Werte der Liste werden durch Kommata getrennt.
- Die Größe des Arrays wird automatisch aus der Anzahl der Werte berechnet.

Angabe der Werte bei der Initialisierung

```
double[] temperaturen;
temperaturen = new double[]{0.4, 1.2, 4.5};
```

- Mit Hilfe von `new` wird Platz für Werte vom Datentyp `double` reserviert.
- Die Angabe der Elemente in den eckigen Klammern ist leer. Die Anzahl der Elemente wird automatisch durch die Elemente in der Liste aller Werte ermittelt.
- Die Liste der Werte wird durch geschweifte Klammern begrenzt und folgt direkt den eckigen Klammern.

Anzahl der Elemente

```
double[] temperaturen;
int anzahl = 0;

temperaturen = new double[]{0.4, 1.2, 4.5};
anzahl = temperaturen.length;
```

- Jedes Array in Java hat die Methode `.length`.
- Der Name des Arrays und die Methode werden durch den Punktoperator verbunden.
- Die Methode `.length` gibt die Anzahl der Elemente in einem Array zurück.

Ausgabe eines Feldes vom Typ ...

```
double[] temperaturen = new double[12];
temperaturen[0] = 0.6;
temperaturen[1] = 1.6;

System.out.println(temperaturen);
```

- Der Name eines Feldes ist ein Synonym für die Speicheradresse des ersten Feldes.
- Der Name des Feldes verweist auf das erste Element in einem Array.
- In diesem Beispiel wird die Speicheradresse des ersten Elements des Arrays `temperaturen` ausgegeben.

Ausnahme: Felder vom Typ char

```
char[] wort = {'H', 'e', 'l', 'l', 'o'};
System.out.println(wort);
```

- Feldes vom Datentyp `char` werden bei der Ausgabe als Objekt vom Typ `String` definiert.
- Das Feld wird vollständig als Satz ausgegeben.

Eckige Klammern in Java

```
datentyp[] array;

array[index] = value
```

- Kennzeichnen eine Variable als Array (Feld).
- Identifizierung eines Elements in einem Array mit Hilfe eines ganzzahligen Index.

Runde Klammern in Java

```
boolean = (Bedingung)

Variable = (ausdruck) operator (ausdruck)

System.out.println(wert)
```

- Runde Klammern fassen Ausdrücke zusammen.
- Runde Klammern erhöhen die Lesbarkeit von komplexen Ausdrücken.
- Begrenzung der Argumentliste einer Methode wie zum Beispiel `println()`.

Geschweifte Klammern in Java

- Geschweifte Klammern fassen Blöcke von Anweisungen zusammen.

```
{
    Anweisung 1;
    Anweisung 2;
}
```

- Zusammenfassung von Elementen eines Arrays.

```
char[] var = {'a', 'e'}
```

Mehrdimensionale Arrays

- Darstellung von mehreren Dimensionen.
- Zwei Dimensionen: x-, y-Koordinatensystem; Schachbrett; Matrizen.
- Drei Dimensionen: x-, y-, z-Koordinatensystem.

Ein- und zweidimensionale Arrays

Tage

(0)	(1)	(2)	...	(30)

Umsätze

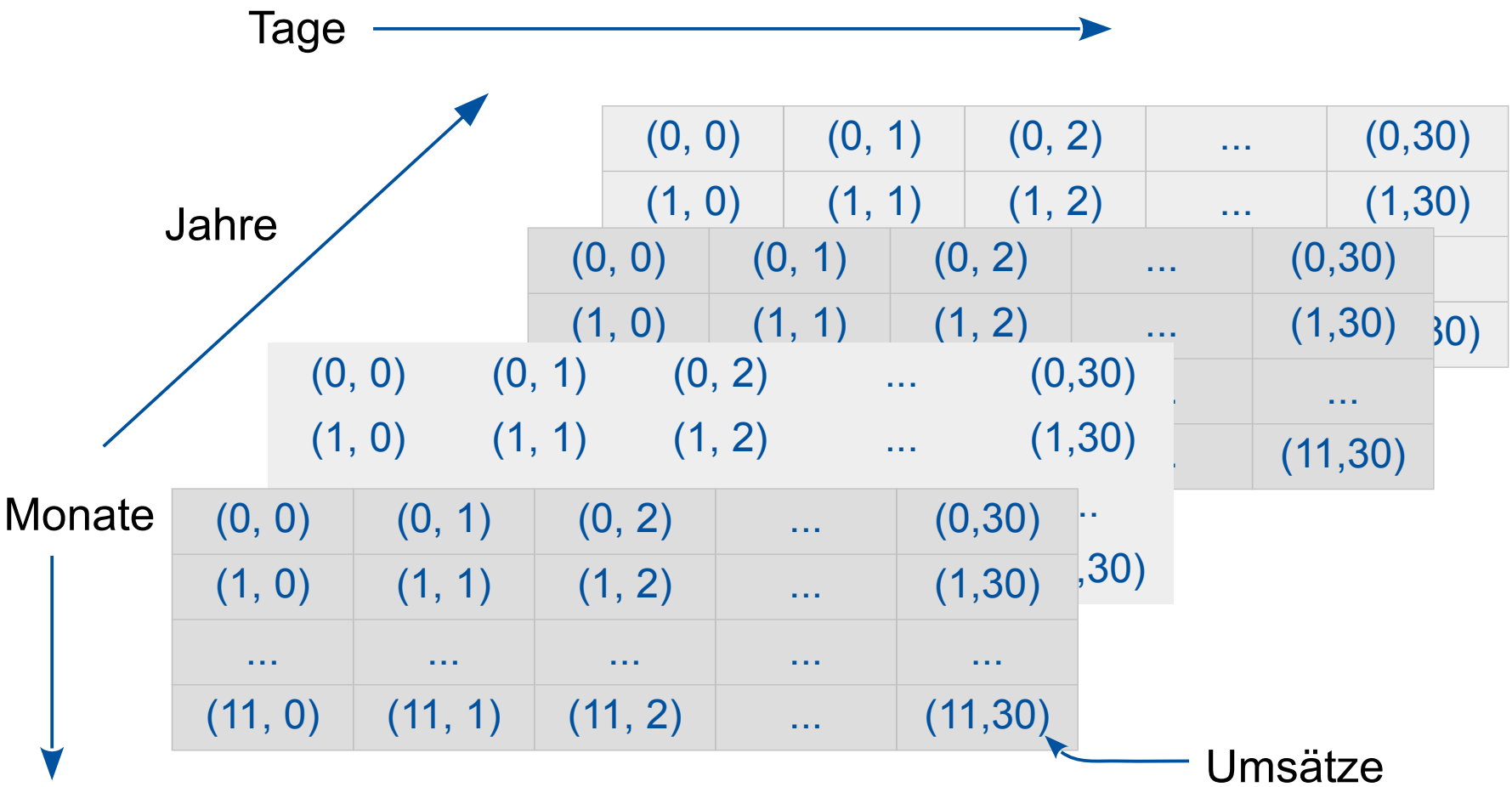
Tage

Monate

(0, 0)	(0, 1)	(0, 2)	...	(0,30)
(1, 0)	(1, 1)	(1, 2)	...	(1,30)
...	...	...	...	...
(11, 0)	(11, 1)	(11, 2)	...	(11,30)

Umsätze

Dreidimensionale Arrays



… deklarieren

```
double[][] temperaturen;
```

- Dem Datentypen folgen die eckigen Klammern.
- Jedes Paar von eckigen Klammern [] steht für eine Dimension.
- In diesem Beispiel ist das Array zweidimensional.

… initialisieren

```
temperaturen = new double[11][30];
```

- Die Anweisung `new` erzeugt ein Feld von einer bestimmten Größe von einem Datentyp.
- Für jede Dimension wird die Anzahl der Elemente angegeben.
- In diesem Beispiel können in der ersten Dimension 12 Elemente gespeichert werden. Die Anzahl der Zeilen ist 12. In der zweiten Dimension können 31 Elemente gespeichert werden. Die Anzahl der Spalten beträgt 31.

… deklarieren und initialisieren

```
int[][][] koordinaten = new int [10][10][20];
```

- Die Variable `koordinaten` wird als Feld vom Typ `double` deklariert.
- Durch die eckigen Klammern wird das Feld als dreidimensional gekennzeichnet.
- Die Anweisung `new` erzeugt ein Feld von einer bestimmten Größe von dem Datentyp double.
- Die Adresse des ersten Feldes im Speicher wird der Variablen `koordinaten` übergeben.

Zuweisung von Werten

```
temperaturen[2][0] = 4.5;

koordinaten[0][1][3] = 3;
```

- Das erste Element in einem zweidimensionalen Array hat den Index [0][0]. Das erste Element in einem dreidimensionalen Array hat den Index [0][0][0].
- Mit Hilfe des Zuweisungsoperators wird einem Element ein Wert entsprechend des Datentyps zugewiesen.

Index eines Array-Elements

```
temperaturen[2][0] = 4.5;

koordinaten[0][1][3] = 3;
```

- Der Index ist immer eine Ganzzahl. Der Index kennzeichnet eindeutig eine Box in einem Container.
- Für jede Dimension des Arrays muss ein Index angegeben.
- Der Index wird durch die eckigen Klammern begrenzt.

Angabe der Werte bei der Deklaration

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

int [][][] achsen = {{{111, 112}, {121, 122},
   {131, 132}},{{211, 212}, {221, 222}, {231, 232}}};
```

- Dem Datentypen folgen die eckigen Klammern. Jedes Paar von eckigen Klammern [] steht für eine Dimension.
- Mit Hilfe des Zuweisungsoperators wird dem Array eine Liste von Werten zu gewiesen.

Schreiben von „Listen“

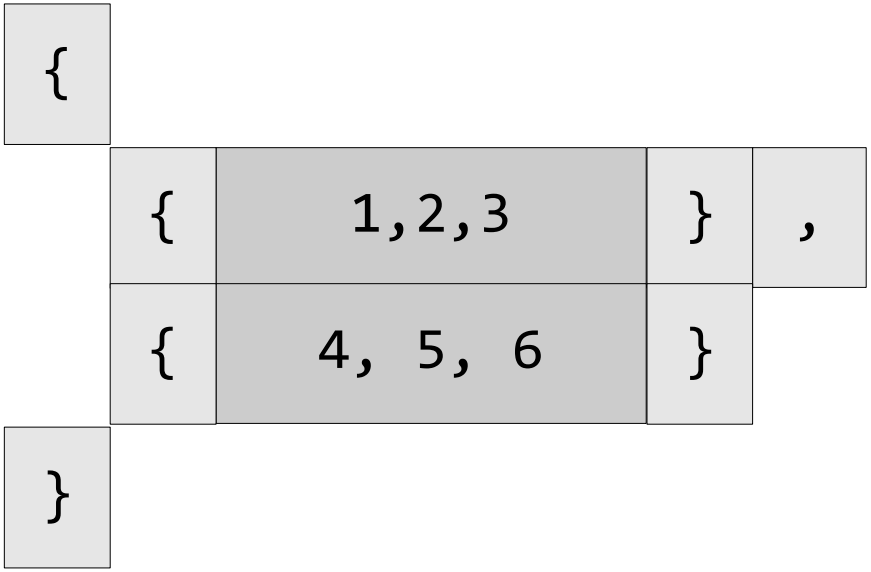
```
int[][] matrix = {{1,2,3},{4,5,6}};

int [][][] achsen = {{{111, 112}, {121, 122},
   {131, 132}},{{211, 212}, {221, 222}, {231, 232}}};
```

- Jede Liste fasst Werte eines bestimmten Datentyps zusammen.
- Die Liste beginnt und endet mit den geschweiften Klammern.
- Die Elemente in einer Liste werden durch Kommata getrennt.
- Für jede Dimension eines Arrays wird eine Liste erstellt.

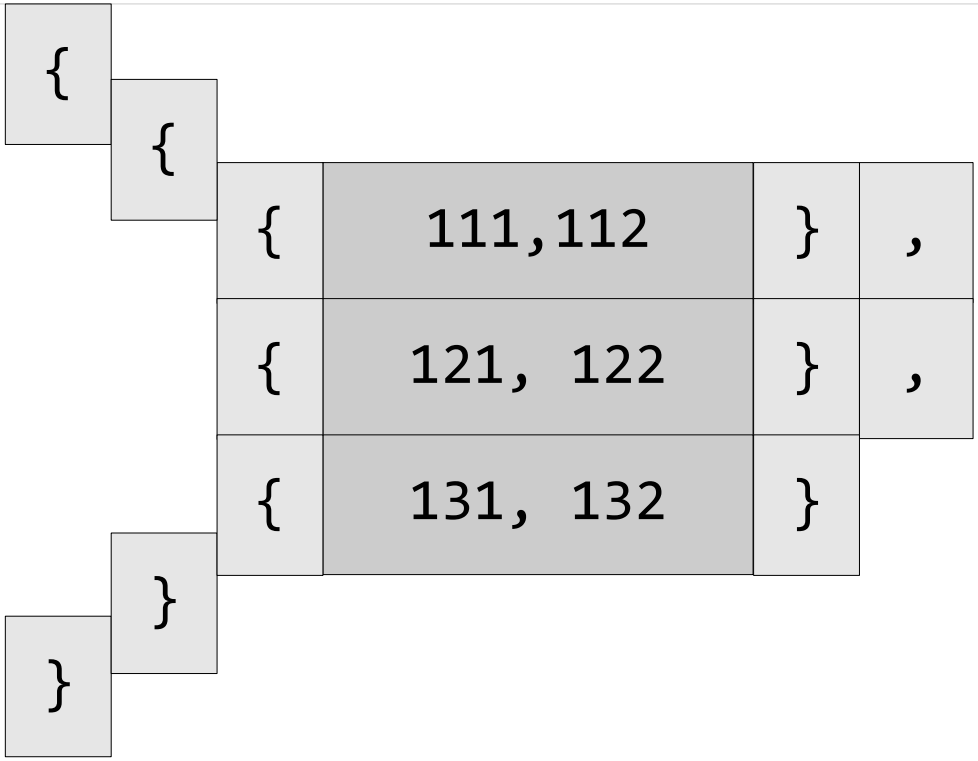
Beispiel: Zweidimensionales Array

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



Beispiel: Dreidimensionales Array

```
int [][][] achsen = {{{111, 112}, {121, 122},
   {131, 132}},{{211, 212}, {221, 222}, {231, 232}}};
```



Angabe der Werte bei der Initialisierung

```
double[] messwerte;
messwerte = new double[][]{{0.3, 2.1, 3.4},
                           {1.2, 3.4, 2.6}};
```

- Mit Hilfe von `new` wird Platz für Werte vom Datentyp `double` reserviert.
- Die Angabe der Elemente in den eckigen Klammern ist leer. Die Anzahl der Elemente wird automatisch durch die Elemente in der Liste aller Werte ermittelt.
- Für jede Dimension wird eine Liste von Werten erstellt und die Speicheradresse der Variablen zugewiesen.

Irreguläre Arrays

- Irreguläre Array müssen gleichzeitig deklariert und initialisiert werden.
- Für die letzte Dimension des Arrays wird keine Angabe gemacht.

… deklarieren und initialisieren

```
double temperaturen[][] = new double[11][];
```

- Mit Hilfe von **new** wird ein irreguläres Array erzeugt.
- In den eckigen Klammern wird die Anzahl der Elemente angegeben.
- Die Angabe für die letzte (hier: die zweite) Dimension kann weggelassen werden. Die eckigen Klammern sind leer.

Initialisierung der letzten Dimension

```
double temperaturen[][] = new double[11][];

temperaturen[0] = new double[30];
temperaturen[1] = new double[27];
temperaturen[2] = new double[30];
temperaturen[3] = new double[29];
```

- Beispiel: Für jede Zeile wird die Anzahl der benötigten Spalten festgelegt.
- Mit Hilfe von `new` wird ein Array mit den gewünschter Anzahl von Elementen für die aktuelle Dimension erzeugt.

Anzahl der Elemente

```
double temperaturen[][] = new double[11][];
int anzahl = 0;

anzahl = temperaturen.length;
temperaturen[0] = new double[30];

anzahl = temperaturen[0].length;
```

- `feld.length` gibt die Anzahl der Elemente in der ersten Dimension zurück.
- `feld[index].length` gibt die Anzahl der Elemente in der aktuellen Dimension zurück.