Arrays / Listen

Werte vom *gleichen Typ* als Liste speichern

Notenliste konventionell

```
double note1 = 6.0d;
double note2 = 5.5d;
double note3 = 4.0d;
double note4 = 5.0d;
double note5 = 4.5d;
```

Viele Variablen

❌ Unflexibel für Veränderungen bei mehr oder weniger Daten

❌ Jede Variable muss speziell behandelt werden

❗ in jeder Variable kann nur einen Wert gespeichert werden

Für jede Variable eine Kommode

Notenliste als Array

```
double[] noten = new double[5];
```

✅ nur eine Variable als Array nötig

💡 **Eckige Klammern** `[]` definieren einen Array

Zahl `[5]` in Eckiger Klammer definiert die **Länge**

Es ist also ein Array vom Datentyp **double** mit der **Länge 5**

Array - Länge durch Zahl in eckiger Klammer

```
double[] noten = new double[3];
```


```
double[] noten = new double[7];
```

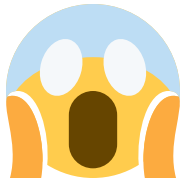
Array - Werte zuweisen

Direkt (bestimmt auch die Länge!)

```
double[] noten = { 6.0d, 5.5d, 4.0d, 5.0d, 4.5d }
```

Via Index

```
double[] noten = new double[5];
noten[0] = 6.0d;
noten[1] = 5.5d;
noten[2] = 4.0d;
noten[3] = 5.0d;
noten[4] = 4.5d;
```



OMG

Länge *startet bei* **1**

```
int size = 100;
int[] values = new int[size];
```

Index *startet bei* **0**

```
int firstValue = values[0]
// index: 100 - 1 = 99
int lastValue = values[size - 1];
```


If you love me, meet me at 1st table
TABLE 00
TABLE 01

🚶 Durch Array iterieren *(schrittweise)*

Mit `for`-Schleife

```
int values = new values[5];

for (int i = 0; i < values.length; i++) {
    // Zuweisung
    values[i] = Math.rand();

    // Zugriff
    System.out.println(values[i]);
}
```

💡 Zugriff und Zuweisung via Index `i`

Mit `foreach`-Schleife

```
int values = new values[5];

for (int value : values) {
    // nur Zugriff
    System.out.println(value);
}
```

💡 Nur Zugriff dafür übersichtlicher

Arrays sind Magier!

Wieso denkt Ihr?

🙇 Deklarationszauber

Konventionell

```
int value1;
int value2;
int value3;
// immer weiter so
int value100;
```

😓 Es müssen 100 Zeilen geschrieben werden für 100 Variablen vom gleichen Typ

⭐ Mit Array

```
int[] values = new int[100];
```

🪄 Eine Zeile reicht aus!

🧙 Zuweisungszauber

Konventionell

```
value1 = Math.rand();
value2 = Math.rand();
value3 = Math.rand();
// immer weiter so
value100 = Math.rand();
```

😓 Es müssen 100 Zeilen geschrieben werden um 100 Variablen einen neuen Wert zuzuweisen

⭐ Mit Array und `for`

```
for (int i = 0; i < values.length; i++) {
    values[i] = Math.rand();
}
```

🪄 Drei Zeilen reichen aus!

Und zwar **auch für 1 Mio Werte**

🧙 Zugriffszauber

Konventionell

```
System.out.println(value1);
System.out.println(value2);
System.out.println(value3);
// immer weiter so
System.out.println(value100);
```

😓 Es müssen 100 Zeilen geschrieben werden um 100 Variablen auszugeben

⭐ Mit Array und `foreach`

```
for (int value : values) {
    System.out.println(value);
}
```

🪄 Drei Zeilen reichen aus!

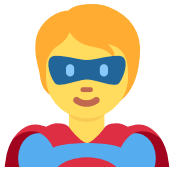
Und zwar **auch für 1 Mio Werte**

💡 Da wir nur auf Daten zugreifen können wir mit `foreach` uns den index

sparen

Merken

Wenn eine **manuelle Nummerierung** in Variablennamen oder Methodennamen vorkommt, sollte man an **Arrays** denken.



Ab hier nur für Interessierte

🦸 Feld / *zwei Dimensionen* - 🚨 Nicht Pflicht!

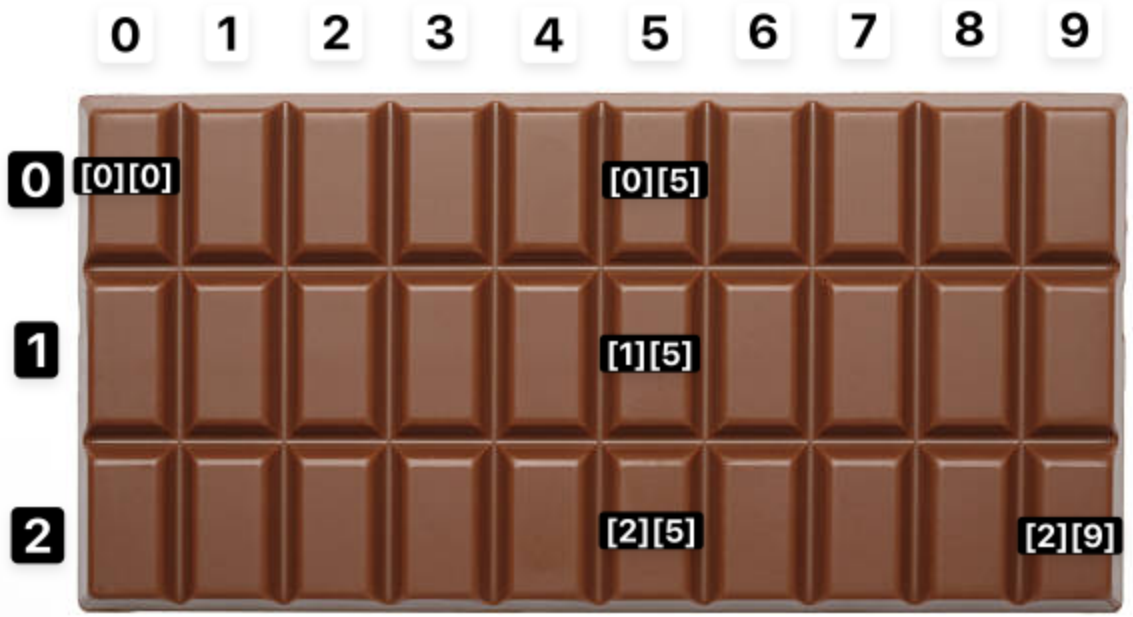
`int` Beispiel

```java
// Deklaration
int[][] numbers = new int[3][10];

// Zuweisung (expliziter Index)
numbers[0][0] = 1000;
numbers[2][9] = 30000;
// oder (impliziter Index)
int index1 = numbers.length - 1;
int index2 = numbers[0].length - 1;
numbers[index1][index2] = 30000;

// Zugriff
int firstValue = numbers[0][0];
int lastValue = numbers[index1][index2]
```

Ein Feld mit mehreren Reihen



💡 Schiffchen versenken, Schachbrett, Koordinatensystem