Felder (arrays) Datum:

Es kommt häufig vor, dass in einem bestimmten Kontext nicht nur eine Variable, sondern dutzende oder hunderte gebraucht werden, um manche Dinge zu modellieren, Beispiele:

- Tic-Tac-Toe-Feld, Schachbrett, allgemein: Spielfeld.
- Liste von Zufallszahlen.
- Noten einer Klasse bei einer Arbeit, etc.

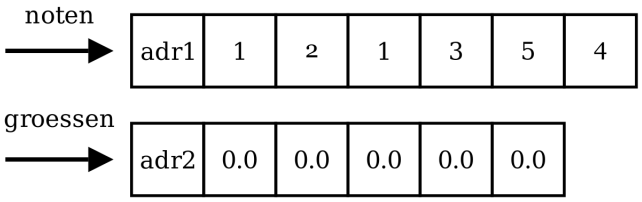
Definition Feld
Ein Feld (array) ist eine Datenstruktur, in der mehrere Werte (gleichen Typs) gespeichert werden können.

Hinweise:

- Der englische Begriff „array“ hat sich im Deutschen bereits so eingebürgert, dass außerhalb der Schule so gut wie niemand von „Feldern“ spricht.
- Streng genommen ist die Voraussetzung „gleicher Typ“ in Java nicht immer für die Speicherung notwendig, wir beschränken uns aber zunächst darauf.

Ein array wird wie üblich mit einer Variablen referenziert. Mit Hilfe dieser Variablen kann dann auf jeden der Einträge des Arrays zugegriffen werden, d.h. es muss nicht für jeden einzelnen Eintrag eine eigene Referenz definiert werden, dies geschieht „automatisch“. Zuerst muss ein array aber **initialisiert** werden. Dazu gibt es in Java zwei Möglichkeiten:

```
// Ein array kann direkt durch Angabe
// seiner Werte in geschweiften Klammern erzeugt werden;
int[] noten = {1,2,1,3,5,4};
// Alternativ kann das array auch erst "leer" erzeugt werden:
double[] groessen = new double[5];
// So wird ein array mit 5 Einträgen erzeugt, die standardmäßig auf 0.0 gesetzt werden
```



Die obige Darstellung ist noch nicht realitätsnah, aber dazu später mehr Details!

Beide Methoden können nützlich sein, je nachdem wie groß das array sein soll, bzw. ob die einzufügenden Daten bereits von Anfang an bekannt sind.
Will man auf einen Eintrag im array zugreifen, kann die **Klammerschreibweise** verwendet werden, z.B.:

```
System.out.println(noten[1]);
// Ausgabe: 2 - das liegt daran, dass in der Informatik der erste
// Eintrag in der Regel den Index 0 hat.
System.out.println(groessen[3]);
// Ausgabe: 0.0 - möchte man dem array einen neuen Wert zuweisen:
groessen[3] = 1.71;
System.out.println(groessen[3]);
// Ausgabe: 1.71
```

Versucht man das array insgesamt auszugeben, so stößt man zunächst auf eine rätselhafte Ausgabe:

```
System.out.println(groessen);
// Ausgabe (z.B.): [D@2f92e0f4
```

Das liegt daran, dass die Variable „groessen“ nur ein Zeiger ist, der zum Speicherort des arrays weist, dort werden weitere Referenzen auf die einzelnen Elemente des arrays gespeichert (dazu später mehr). Will man also ein array komplett ausgeben kann dies z.B. mit einer Wiederholung realisiert werden:

```
for(int i = 0; i < groessen.length; i = i+ 1) {
    System.out.println(groessen[i]);
}
```

In jedem Schritt verändert sich i und damit auch der Eintrag auf den zugegriffen wird. Damit man nicht über das Ende „hinausläuft“ (das würde eine Fehlermeldung produzieren) darf i nur bis zum Index „Länge minus 1“ laufen (da wir ja bei 0 anfangen!). Java bietet hier praktischerweise ein Attribut *length* an, dass jedes array automatisch besitzt und z.B. für diesen Zweck wie oben verwendet werden kann.
Zusammengefasst ist die obige Wiederholung also nur eine Kurzschreibweise für folgenden Code:

```
System.out.println(groessen[0]);
System.out.println(groessen[1]);
System.out.println(groessen[2]);
System.out.println(groessen[3]);
System.out.println(groessen[4]);
```