

Bearbeiten von Arrays mit Schleifen

Arrays sind nützlich, weil man

- viele Werte bzw. Objekte speichern, und
- diese Objekte schnell und einfach bearbeiten kann, z.B. verändern, analysieren, ausgeben usw.



Beispiel: Es sollen alle Werte eines Arrays der Länge 10 auf der Konsole ausgegeben werden.

Möglichkeit 1: Man kann jedes Element **einzeln** über die Indizes von 0 bis 9 ausgeben.

```
public void alleAusgeben()
{
    Console.println(liste[0]);
    Console.println(liste[1]);
    Console.println(liste[2]);
    ...
    Console.println(liste[8]);
    Console.println(liste[9]);
}
```

Dabei geht jedoch der Vorteil von Arrays verloren – da könnte man statt des Arrays auch zehn Variablen verwenden. Und hätte das Array 1000 Elemente, wäre diese Methode sehr, sehr lang ...

Möglichkeit 2: Man verwendet eine **Schleife**, um das **ganze** Array zu bearbeiten.

Dabei verwendet man eine **Variable für die Indizes** des Array: Wenn die Variable bei 0 anfängt und beim letzten Index (hier: 9) aufhört, kann man jedes Element des Arrays ansprechen.

```
public void ausgebenSchleife()
{
    int i;
    i = 0;
    while (i < 10)
    {
        Console.println(liste[i]);
        i++;
    }
}
```

Die zweite Methode sieht wesentlich eleganter aus. Sie tut das genau gleiche wie die erste, und es wäre für den Programmierer kein Aufwand, auf die gleiche Art ein Array mit 1000 Elementen zu bearbeiten – indem einfach die Bedingung zu „ $i < 1000$ “ geändert wird.

Noch besser, als die Bedingung `while (i < 10)` zu verwenden, ist die Verwendung des Attributs „length“ – dieses gibt die Anzahl der Elemente eines Arrays an:

```
while (i < liste.length) ...
```

Aufgabe

Verwende die ausgeteilte BlueJ-Vorlage und bearbeite die Klasse ArrayFüllen.

- a) Die Klasse soll ein Array für 10 Integer-Werte deklarieren und im Konstruktor erzeugen.

Zu b) bis f) schreibe je eine Methode. Jede Methode soll eine while-Schleife verwenden. Benenne die Methoden sinnvoll, z.B. „ausgeben“, „einlesen“ usw.

- b) Das gesamte Array wird auf der Konsole ausgegeben.
- c) Die Methode bittet den Benutzer um die Eingabe von 10 Zahlen und speichert diese Zahlen in den 10 Elementen des Arrays.
- d) Jedem Element des Arrays wird ein zufälliger Wert zwischen -100 und +100 zugewiesen. Die Zufallszahlen erzeugst du so: `z = (int) (Math.random() * 201 - 100);`
- e) Den Elementen des Arrays werden die Werte von 0 bis 90 zugewiesen (in 10er-Schritten).
- f) Den Elementen des Arrays werden die ersten 10 Quadratzahlen zugewiesen (1, 4, 9, 16 usw.)
- g) Die **main()-Methode** ruft jede Methode von c) bis f) einmal auf. Vor jedem Aufruf gibt sie einen Satz auf der Konsole aus, die beschreibt, was der Aufruf tut. Nach jedem Aufruf wird das Array mit der Methode aus b) ausgegeben. So kannst du prüfen, ob die Methoden tun, was sie sollen.