Vom traditionellen zum modernen Java

Matthias Eurich – 2017-02-07

Table of Contents

Vom Schleifenkonstrukt zum Datenfluss

for-Schleife

Die klassische `for`-Schleife gab es bereits vor Java, z.B. in der Programmiersprache C. Im folgenden Beispiel werden die Elemente einer Liste nacheinander ausgegeben.

```
List<String> friends = Arrays.asList("Liam", "Kim", "Laura");

for (int i = 0; i < friends.size(); i++) { System.out.println(friends.get(i)); }
```

foreach-Schleife

Mit Java 1.5 wurde die *foreach*-Schleife eingeführt. Durch Wegfall der Hilfsvariable `i`, die zur Iteration in einer klassischen `for`-Schleife verwendet wird, wird der Code lesbarer. Typische Fehler, die bei der Verwendung der `for`-Schleife, z.B. *Off-By-One-Error*, auftreten können, sind bei einer *foreach*-Schleife so nicht möglich. Ersetzen wir die for-Schleife durch eine *foreach*-Schleife, erhalten wir etwas zivilisierteren Code.

```
for (String name : friends) { System.out.println(name); }
```

Stream.forEach und Iterable.forEach

Mit Java 1.8 wurde das Interface `Iterable` um die default-Methode `forEach()` erweitert. Die neu hinzugekommen Interfaces `Stream`, `IntStream`, `LongStream` und `DoubleStream` verfügen ebenfalls über die Methode `forEach()`. Durch die Methode

`forEach()` lässt sich der Code aus dem obrigen Beispiel mit Hilfe eines Lambda-Ausdrucks weiter vereinfachen.

```
friends.forEach(name -> System.out.println(name));
```

Noch kürzer (und lesbarer) geht es mit einer Methoden-Referenz.

```
friends.forEach(System.out::println);
```

Von anweisungsorientierter zu ausdrucksorientierter Programmierung

Die imperative Programmierung ist ein anweisungsorientiertes Programmierparadigma. Eine `set()`-Methode hat keinen Rückgabewert bzw. den Rückgabetyp `void`. Will man nun mit dem gleichen Objekt weiterarbeiten, müssen weitere Aufrufe von Methoden folgen. Deutlich wird das an folgendem Beispiel:

```
Beverage cola = new Beverage();
cola.setName("Cola");
cola.setPrice(0.5);
cola.setSize(0.3);
```

Jeder Methodenaufruf ist eine Anweisung, also ein Programmkonstrukt, daseine Aktion beschreibt.

Ein anderes Programmierparadigma, das vor allem in funktionalen oder funktional-angehauchten Programmiersprachen zu finden ist, ist die **ausdrucksorientierte Programmierung**. Ein Ausdruck ist ein Programmkonstrukt in Java, das eine Rechnung beschreibt. Bei der ausdrucksorientierten Programmierung soll ein Ausdruck am Ende seiner Auswertung einen Wert ergeben. Im nachfolgenden Beispiel ist zu sehen, dass die `set()`-Methoden einen Rückgabewert vom Typ `Beverage` haben. Das erlaubt die Verkettung der Methodenaufrufe.

```
Beverage cola = new Beverage()
                  .setName("Cola")
                  .setPrice(0.5)
                  .setSize(0.3);
```

Je nach Programmiersprache und Implementierung kann der Rückgabewert das veränderte Ausgangs-Objekt oder eine Kopie sein.

Last updated 2017-04-12 21:46:22 CEST