Ein Text zum Entwurfsmuster Iterator

Der folgende Text führt in einem Buch[1] den Abschnitt zum Entwurfsmuster Iterator ein.

"Es gibt viele Möglichkeiten, Objekte in eine Sammlung zu packen.

Stecken Sie sie in ein Array-, Stack-, -List- oder Hashtable-Objekt. Sie haben die freie Auswahl. Und jede hat ihre Vor- und Nachteile.
Aber irgendwann wird Ihr Client über diese Objekte iterieren wollen. Werden Sie ihm Ihre Implementierung zeigen, wenn er das tut? Wir hoffen ganz entschieden, dass Sie das nicht tun werden! Es wäre einfach nicht professionell. Sie müssen Ihre Karriere nicht riskieren.
Sie werden sehen, wie Sie es Clients ermöglichen, über Ihre Objekt zu iterieren, ohne dass er je sieht, wie Sie Ihre Objekte speichern. Sie werden auch lernen, wie Sie Super Collections von Objekten pflegen, die mit einem einzigen Satz einige beeindruckende Datenstrukturen überspringen können.
Und wenn Ihnen das immer noch nicht ausreicht, werden Sie außerdem ein oder zwei Dinge über Objektverantwortlichkeit lernen."

Sammlungsklassen

Es geht in der Datenstruktur um Sammlungsklassen, also die Gruppe von Klassen, die das interface Collection implementieren, das *"collection framework"* von Java. Sie alle müssen einen Iterator bereitstellen, der dafür sorgt, dass die Aufgabe "mach das einmal nacheinander für alle Elemente aus …" unabhängig von der konkreten Realisierung in der Sammlungsklasse ausgeführt werden kann.
Die vollständige Definition des interfaces Collection lese man in der Javadoc nach, hier sind nur die wichtigsten Methoden kurz zusammengestellt:

Name	*Zweck*	*Wert*
add(o)	fügt das Objekt hinzu	
addAll(coll)	fügt alle Objekte einer Sammlung hinzu	
contains(o)	prüft, ob das Objekt enthalten ist	boolean
equals()	vergleicht zwei Sammlungen (spannend: was heißt gleich?)	boolean
isEmpty()	prüft, ob die Sammlung leer ist	boolean
iterator()	gibt einen Iterator zurück	Iterator
remove(o)	entfernt ein Objekt	
size()	gibt die Anzahl der Elemente in der Sammlung zurück	int

Und schon wieder alles anders

Seit Java 5 gibt es die Möglichkeit über alle Objekte in einer Collection zu iterieren, ohne sich explizit einen Iterator zu beschaffen. Dazu benutzt man die dafür erweiterte for-Anweisung, die man mit for/in beschreiben kann. Eine Anweisung wie

```
for (Object o: warteschlange) System.out.println(o.toString());
```

ist zulässig, holt sich von **`warteschlange`** das Iteratorobjekt und wird sinnvoll bearbeitet.

1 Quelle: Freeman; Entwurfsmuster von Kopf bis Fuß