Nr. 03 | Herbst 2012 | www.ijug.eu

Java aktuell

Das Magazin der Java-Community

Java aktuell

Java in Höchstform

Java EE 6:
GlassFish, JBoss und Geronimo, Seite 11

Android:
Wissenschaftliche Applikationen der nächsten Generation entwickeln, Seite 38

Jnect:
Bewegungen in Java erkennen, Seite 59

D: 4,90 EUR A: 5,60 EUR CH: 9,80 CHF Benelux: 5,80 EUR ISSN 2191-6977

iJUG Verbund

Sonderdruck

4 191978 304903 03

Java Forum Stuttgart

JUGS
java user group stuttgart

Die Java User Group Stuttgart e.V. veranstaltet am 5. Juli 2012 im Kultur- & Kongresszentrum Liederhalle (KKL) in Stuttgart wieder das Java Forum Stuttgart. Wie im Vorjahr werden rund 1.200 Teilnehmer erwartet. Geplant sind 42 Vorträge in sechs parallelen Tracks. Zudem werden bis zu 35 Aussteller vor Ort sein, darunter auch der Interessenverbund der Java User Groups e.V. (iJUG). An der Community-Wand stehen sowohl offene White-Boards als auch BoF-Boards (Bird-of-a-Feather). Abends gibt es die Gelegenheit, sich bei verschiedenen BoF-Sessions mit Gleichgesinnten zu treffen, um über ein bestimmtes Thema zu diskutieren und sich auszutauschen. Darüber hinaus wird es wieder eine Jobbörse/Karriereecke für die Besucher geben.

Workshop „Java für Entscheider"
Die eintägige Überblicksveranstaltung am Vortag (4. Juli 2012) zeigt Begrifflichkeiten und wichtige Technologien aus der seit Jahren in der Industrie etablierten Plattform Java. Ausgehend von strategischen Gesichtspunkten wie Bedeutung und Verbreitung reicht der Blick über das Client-seitige Java (Java SE) und die wesentlichen Entwicklungswerkzeuge bis zum Server-seitigen Java (Java EE). Dort stehen dann die Bedeutung von Java als Integrationsplattform und die verschiedenen Technologien im Mittelpunkt. Weiterhin wird noch der Einsatz von Java in den immer wichtiger werdenden mobilen Lösungen (Android, iOS) gestreift. Abschließend kommen noch das Ausrollen von Java-Lösungen und das sehr interessante Eclipse als Rich-Client zur Sprache, um dann den Bogen von der Software-Entwicklung hin zum Betrieb zu schlagen.

Experten-Forum Stuttgart
Am 6. Juli 2012 findet wieder im Anschluss an das Java Forum Stuttgart ein Experten-Forum Stuttgart statt. Auf dem Programm stehen zwölf halbtägige Workshops in sechs parallelen Tracks. Die Workshops in kleinen Gruppen mit maximal 25 Teilnehmern ermöglichen einen intensiven Austausch zwischen Trainer und Zuhörern.
Anmeldung und weitere Informationen zum Java Forum Stuttgart unter **www.java-forum-stuttgart.de**

Unbekannte Kostbarkeiten des SDK

Heute: Double Brace Initialization und Instance Initializer

Bernd Müller, Ostfalia

Das Java SDK enthält eine Reihe von Features, die wenig bekannt sind. Wären sie bekannt und würden sie verwendet, könnten Entwickler viel Arbeit und manchmal sogar zusätzliche Frameworks einsparen. Wir wollen in dieser Reihe derartige Features des SDK vorstellen: die unbekannten Kostbarkeiten.

```
String[] ziffern =
    new String[] { „eins“, „zwei“, „drei“, „vier“, ...};
```

Listing 1

```
List<String> ziffern = new ArrayList<>();
ziffern.add(“eins”);
ziffern.add(“zwei”);
ziffern.add(“drei”);
ziffern.add(“vier”);
...
```

Listing 2

```
List<String> ziffern =
    Arrays.asList(new String[] { „eins“, „zwei“,
    „drei“, „vier“, ... });
```

Listing 3

```
List<String> ziffern = new LinkedList<String>() {{
    add(„eins“);
    add(„zwei“);
    add(„drei“);
    add(„vier“);
    ...
}};
```

Listing 4

```
Map<String, String> ziffern = new HashMap<String,
String>() {{
    put(“1”,“eins”);
    put(“2”,“zwei”);
    put(“3”,“drei”);
    put(“4”,“vier”);
    ...
}};
```

Listing 5

Wir haben in dieser Reihe bereits den Service-Loader, das Dynamic Proxy und die VisualVM vorgestellt. Während die beiden erstgenannten Kostbarkeiten über Klassen beziehungsweise Interfaces des SDK realisiert werden, ist die VisualVM eine separate und eigenständige Java-Anwendung des SDK. Es gibt jedoch auch unbekannte Kostbarkeiten, die eine Ebene tiefer angesiedelt sind: in der Sprache Java selbst. Die Double Brace Initialization erlaubt die elegante und einfache Initialisierung von Variablen komplexerer Datenstrukturen, etwa Collections, und basiert auf dem Instance Initializer Block. Wir motivieren zunächst die Verwendung der Double Brace Initialization und gehen dann, wenn wir deren innere Funktionsweise analysieren, auf Instance Initializer ein.

Double Brace Initialization

Die Initialisierung von Arrays kann mit dem sogenannten „Array Initializer“ relativ einfach realisiert werden (siehe Listing 1). Anders sieht es etwa mit Klassen des Collection-Frameworks aus. Bei Listen, Mengen oder Maps müssen die einzelnen Listenelemente durch explizite Methodenaufrufe hinzugefügt werden (siehe Listing 2).

Ein wenig Erleichterung für Listen bringt die „asList()“-Methode der Klasse „Arrays“ (siehe Listing 3). Mit der „Double Brace Initialization“ kann dies wesentlich eleganter realisiert werden (siehe Listing 4). Dies funktioniert auch mit „Maps“ (siehe Listing 5).

Die Hintergründe, Teil 1: Anonyme innere Klassen

Java 1.1 führte innere Klassen ein. Eine lokale innere Klasse kann innerhalb einer Methode definiert werden. Benötigt man nur eine einzige Instanz dieser Klasse, kann auf eine Namensvergabe verzichtet werden, und Definition und Instanziierung können zu einem einzigen Konstrukt, einer sogenannten „anonymen inneren Klasse“, zusammengefasst werden. Anonyme innere Klassen können ein Interface implementieren oder von einer Oberklasse erben. Listing 6 zeigt ein Beispiel, das häufig verwendet wird, um einem „JButton“ einen „ActionListener“ hinzuzufügen.

Hier ist anzumerken, dass die anonyme innere Klasse das Interface „ActionListener“ implementiert und damit die einzige Methode dieses Interface definieren muss. Bei der zweiten Verwendungsart wird von einer Oberklasse abgeleitet, wie im Listing 7 gezeigt.

Hier erbt die anonyme innere Klasse von der Klasse „Thread“ und überschreibt die geerbte Methode „run()“. Wir wiederholen nun den Code unseres einführenden Beispiels mit dem doppelten Klammerpaar, löschen jedoch das innere Klammerpaar samt Inhalt (siehe Listing 8).

Was wir hier sehen, ist eine anonyme innere Klasse, die von „LinkedList“ erbt, aber keine der geerbten Methoden überschreibt und auch keine Methode zusätzlich definiert. Wir sind der Syntax der Double Brace Initialization also auf der Spur.

Die Hintergründe, Teil 2: Instance Initializer

Ebenfalls mit Java 1.1 wurde der Instance Initializer eingeführt, dessen Definition in der Java Language Specification im Abschnitt 8.6 nachgelesen werden kann. Ein Instance Initializer ist ein einfacher Block, der beliebige Anweisungen enthalten kann und bei der Erzeugung eines Objekts nach den Initialisierungen der Variablendefinition und vor dem Konstruktor ausgeführt wird (siehe Listing 9).

Beim Erzeugen einer Instanz über den Konstruktor wird zunächst die Variable „a" mit 5 initialisiert. Die beiden Instance Initializer werden in der Reihenfolge ihrer Definition ausgeführt. Die Variable „a" bekommt also zunächst den Wert 23, dann den Wert 77 zugewiesen. Zuletzt werden die Anweisungen des Konstruktor-Rumpfs ausgeführt, hier also der Variablen „a" der Wert 117 zugewiesen. Da Instance Initializer Blöcke sind, können verschiedene Anweisungen, nicht nur Werte-Zuweisungen verwendet werden. Insbesondere sind auch mehrere Anweisungen erlaubt. Man kann sich mit „System.out.println()"-Aufrufen davon überzeugen, dass die Variable „a" tatsächlich nacheinander die entsprechenden Werte zugewiesen bekommt. Wenn wir nun unser einführendes Beispiel nochmals betrachten, wird die Semantik schnell klar (siehe Listing 10).

Hier wird eine anonyme innere Klasse als Unterklasse von LinkedList erzeugt und ein Instance Initializer definiert, der aus den „add()"-Methodenaufrufen besteht. Da „LinkedList" serialisierbar ist, warnt der Compiler bei unserer anonymen inneren Klasse vor einer fehlenden Versionsnummer der Serialisierung. Wir definieren diese im folgenden Beispiel, um den Instance Initializer noch einmal hervorzuheben (siehe Listing 11). Zusätzlich demonstrieren wir eine Variablendefinition im Instance Initializer, die im Beispiel zwar relativ sinnlos ist, aber die allgemeine Verwendbarkeit des Instance Initializer beispielhaft darstellt.

```
Map<String, String> ziffern =
    new HashMap<String, String>() {
        private static final long serialVersionU-
ID = 1L;
        {
            String fuenf = „5";
            put(„1", „eins");
            put(„2", „zwei");
            put("3","drei");
            put("4","vier");
            put(fuenf,"fünf");
        }};
```

Listing 11

Die Definition neuer Methoden im äußeren Block ist ebenfalls möglich, aber wenig sinnvoll: Da die Klasse anonym ist, kann die Variable „ziffern" nicht auf die Klasse gecastet werden, was die Voraussetzung für die Verwendung der Methode ist, wenn man einmal von Reflection absieht.

Fazit

Wir haben hier gezeigt, wie mit der Double Brace Initialization komplexe Variablen einfach initialisiert werden können. Während die Double Brace Initialization ein Java-Idiom darstellt, das bei einer Google-Suche zu 178.000 Treffern (April 2012) führt, ist die Grundlage des Idioms, der Instance Initializer, ein Bestandteil der Sprache Java und wird in der Sprachbeschreibung explizit im Abschnitt 8.6 definiert. Sowohl die Double Brace Initialization als auch der Instance Initializer gehören zu den unbekannten Kostbarkeiten des SDK, hier sogar zu den unbekannten Kostbarkeiten der Sprache Java.

Bernd Müller
bernd.mueller@ostfalia.de

Bernd Müller ist Professor für Software-Technik an der Ostfalia. Er ist Autor des Buches „Java-Server Faces 2.0" und Mitglied in der Expertengruppe des JSR 344 (JSF 2.2).

```
JButton button = new JButton(...);
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        ... // some code
    }
});
```

Listing 6

```
Runtime.getRuntime().addShutdownHook(new Thread() {
    public void run() {
        ... // some code
    }
});
```

Listing 7

```
List<String> ziffern = new LinkedList<String>() {
};
```

Listing 8

```
public class InstanceInitializerTest {
    private int a = 5;
    {
        a = 23;
    }
    {
        a = 77;
    }
    public InstanceInitializerTest() {
        a = 117;
    }
    ...
}
```

Listing 9

```
List<String> ziffern = new LinkedList<String>() {{
    add(„eins");
    add(„zwei");
    add(„drei");
    add(„vier");
    ...
}};
```

Listing 10

Oracle gegen Google: Interne E-Mails

Beim Prozessauftakt am 16. April 2012 zwischen Oracle und Google in San Francisco hat der Datenbankriese seine Position bekräftigt und in seiner Eröffnungserklärung Google vorgeworfen, bei der Entwicklung von dem Betriebssystem Android sich bewusst fremder Technologie bedient zu haben, um im Smartphone-Markt nicht hinter der Konkurrenz zurückzufallen. Der Anwalt von Oracle, Michael Jacobs, belegte die Aussage mit Zitaten aus internen E-Mails von Google. Im Mittelpunkt stand eine E-Mail des Entwicklers Tim Lindholm, in der der ehemalige Sun-Angestellte sich dafür ausgesprochen hatte, mit Sun eine Java-Lizenz auszuhandeln. Oracle beteuerte, es sei dem Suchmaschinenbetreiber durchaus bekannt gewesen, dass er eine Lizenz hätte erwerben müssen.

www.ijug.eu

JETZT ABO BESTELLEN

Sichern Sie sich 4 Ausgaben für 18 EUR

Für Oracle-Anwender und Interessierte gibt es das Java aktuell Abonnement auch mit zusätzlich sechs Ausgaben im Jahr der Fachzeitschrift DOAG *News* und vier Ausgaben im Jahr Business News zusammen für 70 EUR. Weitere Informationen unter www.doag.org/shop/

FAXEN SIE DAS AUSGEFÜLLTE FORMULAR AN

0700 11 36 24 39

ODER BESTELLEN SIE ONLINE

go.ijug.eu/go/abo

Interessenverbund der Java User Groups e.V.
Tempelhofer Weg 64
12347 Berlin

Java aktuell

+++ AUSFÜLLEN +++ AUSSCHNEIDEN +++ ABSCHICKEN +++ AUSFÜLLEN +++ AUSSCHNEIDEN +++ ABSCHICKEN +++ AUSFÜLLEN

☐ **Ja**, ich bestelle das Abo Java aktuell – das IJUG-Magazin: 4 Ausgaben zu 18 EUR/Jahr

☐ **Ja**, ich bestelle den kostenfreien Newsletter: Java aktuell – der iJUG-Newsletter

ANSCHRIFT

Name, Vorname

Firma

Abteilung

Straße, Hausnummer

PLZ, Ort

GGF. RECHNUNGSANSCHRIFT

Straße, Hausnummer

PLZ, Ort

E-Mail

Telefonnummer

X

Die allgemeinen Geschäftsbedingungen* erkenne ich an, Datum, Unterschrift

*Allgemeine Geschäftsbedingungen:

Zum Preis von 18 Euro (inkl. MwSt.) pro Kalenderjahr erhalten Sie vier Ausgaben der Zeitschrift "Java aktuell - das iJUG-Magazin" direkt nach Erscheinen per Post zugeschickt. Die Abonnementgebühr wird jeweils im Januar für ein Jahr fällig. Sie erhalten eine entsprechende Rechung. Abonnementverträge, die während eines Jahres beginnen, werden mit 4,90 Euro (inkl. MwSt.) je volles Quartal berechnet. Das Abonnement verlängert sich automatisch um ein weiteres Jahr, wenn es nicht bis zum 31. Oktober eines Jahres schriftlich gekündigt wird. Die Wiederrufsfrist beträgt 14 Tage ab Vertragserklärung in Textform ohne Angabe von Gründen.