Auf der Suche nach Werten in der Softwaretechnik: Werte und Objekte in objektorientierten Programmiersprachen

Jörg Rathlev
Beate Ritterbach
Axel Schmolitzky

Arbeitsbereich Softwaretechnik
Universität Hamburg
Vogt-Kölln-Str. 30
22527 Hamburg
{rathlev,schmolitzky}@informatik.uni-hamburg.de
b.ritterbach@logimod.de

Objektorientierte Programmiersprachen (OOPS) sind traditionell stark bei der Definition benutzerdefinierter *Objekttypen*. Für benutzerdefinierte *Werttypen* hingegen bieten sie wenig Unterstützung. Fachlich motivierte, vom Entwickler zu definierende Werttypen (*Fachwerte*) spielen jedoch eine wichtige Rolle in Softwareprojekten [Zül04], siehe auch die explizite Darstellung des Value-Object-Patterns bei Fowler [Fow02].

OOPS stellen zur Entwicklung benutzerdefinierter Typen lediglich Objekttypen zur Verfügung. Deren Eigenschaften entsprechen jedoch nicht den konzeptionellen Eigenschaften von Werttypen [McL82]: *Werte sind unveränderlich und zur Laufzeit nicht erzeugbar. Die Operationen von Werttypen verhalten sich referenziell transparent und sind frei von Seiteneffekten.* Im Gegensatz dazu besitzen Objekte einen potenziell veränderlichen Zustand und werden zur Laufzeit erzeugt und vernichtet.

Die Unterscheidung zwischen Werten und Objekten ist nicht zu verwechseln mit den orthogonalen Begriffspaaren *Wertsemantik/Referenzsemantik* und *Wertparameter/ Referenzparameter*. Ob es sich bei einer Abstraktion um ein Objekt oder um einen Wert handelt, ist unabhängig von der Speicherverwaltung und der Art der Parameterübergabe.

Unerzeugbarkeit ist eine wesentliche Eigenschaft von Werten. Unveränderbarkeit allein ist lediglich notwendig, aber nicht hinreichend. *Unveränderliche Objekte* sind keine Werte, es sind Objekte mit Lebenszyklus, die erzeugt werden und zerstörbar sind.

Aufgrund der Nichterzeugbarkeit von Werten besteht ein asymmetrisches Abhängigkeitsverhältnis zwischen Wert- und Objekttypen: Objekte können auf Werte Bezug nehmen, aber nicht umgekehrt.

Die aus Sicht der Softwaretechnik praxisrelevanten OOPS wie Java, C#, Eiffel, Smalltalk und C++ bieten keine Sprachmittel zur Definition benutzerdefinierter Werttypen im obigen Sinne. In einigen dieser Sprachen kann zwischen Wert- und Referenzsemantik und zwischen Wert- und Referenzparametern gewählt werden, jedoch bieten die Sprachen keine konzeptionelle Trennung von Wert- und Objekttypen.

Dies führt zu Entwürfen für Fachwerttypen, die mit Entwurfsmustern und Rahmenwerken auf Basis von Objekttypen vorgenommen werden müssen, und resultiert meist in aufwändigen und schwer wartbaren Konstruktionen. Softwaretechnisch ist es jedoch nützlich, wenn Entwickler sich auf *durch die Sprache garantierte* Eigenschaften verlassen können, anstatt sie selbst manuell sicherstellen zu müssen (für Wartbarkeit, Klarheit und Typsicherheit). Deshalb betrachten wir es als erforderlich, dass OOPS einen expliziten Mechanismus für benutzerdefinierte Werttypen anbieten.

Die Programmiersprache VJ [Rat06] ist eine prototypische Java-Erweiterung, die *strukturierte Werttypen* unterstützt und deren konzeptionelle Eigenschaften zur Übersetzungszeit sicherstellt:

- In VJ sind die primitiven Typen von Java sowie der Typ `String` vordefinierte Werttypen.
- Neue Werttypen werden mit einem neuen Schlüsselwort deklariert (`valueclass`, statt `class` für Objekttypen).
- Strukturierte Werttypen setzen sich aus anderen Werttypen zusammen, die wir als die *konstituierenden Eigenschaften* des strukturierten Werttyps bezeichnen. Diese Eigenschaften definieren die Gleichheit von Exemplaren dieser Werttypen.
- Werte werden nicht erzeugt, sondern ausgewählt; dazu existieren in VJ *Wertwähler* – Operationen eines Werttyps, die ein konzeptionell bereits existierendes Exemplar dieses Typs liefern. Wertwähler bilden das Gegenstück zu den Literalen primitiver Typen.
- Eine gemeinsame Abstraktion über Werte und Objekte kann in VJ durch Interfaces erreicht werden, für die gewisse Einschränkungen postuliert werden müssen.

Verwandte Ansätze finden sich in *Fortress* [Fortress], einer in der Entwicklung befindlichen Sprache, die sogenannte *Wertobjekte* unterstützt, die den in dieser Arbeit beschriebenen Werten ähneln. *Kava* [Bac03] ist eine javabasierte Programmiersprache, die ohne primitive Typen auskommt und auch einfache Typen wie Ganzzahlen mit den Mitteln der Sprache selbst definiert.

Literatur

[Bac03] Bacon, D. F.: "Kava: A Java Dialect with a Uniform Object Model for Lightweight Classes", *Concurrency - Practice and Experience,* 15:3-5, S. 185-206, 2003.

[Fortress] *Fortress*, http://research.sun.com/projects/plrg/, zuletzt besucht am 08. Januar 2007.

[Fow02] Fowler, M.: *Patterns of Enterprise Application Architecture*, Addison-Wesley, 2002.

[McL82] MacLennan, B. J.: "Values and Objects in Programming Languages", *ACM SIGPLAN Notices,* 17:12, S. 70-79, 1982.

[Rat06] Rathlev, J.: *Ein Werttypkonstruktor für Java - Erweiterung einer objektorientierten Programmiersprache um Werttypen*, Diplomarbeit, Universität Hamburg, 2006.

[Zül04] Züllighoven, H.: *The Object-Oriented Construction Handbook*, Morgan-Kaufmann Publishers, 2004.