Teil III

Die Idee der Typisierung

In Teil I haben wir kurz die wesentliche Bedeutung von Typen rekapituliert. Wir benutzen sie vor allem dazu, die Definitions- und Wertebereiche von Funktionen möglichst genau einzugrenzen. Das hat sich in der Praxis bewährt, um die Fehlerquote von Programmen zu reduzieren, genauer: um das frühzeitige Erkennen von Fehlern zu erleichtern. Jetzt wollen wir die Rolle der Typisierung in der Programmierung genauer untersuchen.

Es gibt ein sehr ausgereiftes Gebiet „Theorie der Typsysteme“ (im Folgenden kurz: „Typtheorie“) im Bereich der Theoretischen Informatik. Dort werden grundlegende Fragen der Entscheidbarkeit, Ausdrucksmächtigkeit, Effizienz etc. von unterschiedlichen Arten von Typsystemen untersucht. Diese Aspekte gehen aber über den Rahmen unseres Buches hinaus und werden hier nicht weiter betrachtet. (Die Bücher [117, 118] und der Artikel [30] bieten dem interessierten Leser eine gute Übersicht.)

Unser Ziel ist es, Typisierung aus dem Blickwinkel des Sprach*designs* zu betrachten. Deshalb werden wir Fragen der theoretischen Fundierung höchstens kurz streifen. Vor allem aber bedeutet es, *Entscheidungen* zu treffen. Denn für viele Aspekte der Idee „Typisierung“ gibt es mehr als eine Form der Realisierung, und „Design“ heißt, aus einer Fülle von Möglichkeiten eine Auswahl zu treffen – und zwar so, dass ein homogenes und in sich schlüssiges Gesamtbild entsteht. Ein derart durchgängiges, nach einheitlichen Kriterien gestaltetes System ist die Leitlinie der folgenden Kapitel.