3.3.7 Objekt: Handlungsträger kontra Variablentypus

Und noch eine kleine Bemerkung – Objekte in unserem Sprachgebrauch sind Handlungsträger unseres Problems, einzelne Funktionalitäten und Berechnungen bleiben bei dieser Betrachtung noch vollkommen außen vor.

Dies deckt sich zwar mit verbreiteten Beschreibungen in den Lehrbüchern, widerspricht jedoch ein wenig der gängigen Programmier-Praxis, jede öffentliche Sammelstelle von Funktionen als Objekt zu bezeichnen. Hier überschneiden sich denn auch Programmier-Logik und Programmiersprache, denn ein solches Objekt, das mehr oder minder einfache Funktionen zur Verfügung stellt, ist eher dem zuzurechnen, was die Sprachen unter Variablen-Typus bezeichnen. Ein Programmiersprachen-Objekt des Typus „Liste“ kann eben Listenspalten sortieren und vergleichen, ein Programmiersprachen-Objekt des Typus „Zahl“ kann zwischen den diversen Zahlentypen konvertieren, ein Programmiersprachen-Objekt des Typus „String“ kann diverse Verarbeitungen eingegebener Texte durchführen.

Von solchen Objekten reden wir bei der Programm-Logik nur dann, wenn unser Problem heißt: „bereite Text auf“ oder „konvertiere Zahlen“ – ansonsten interessieren uns solche „Objekte“ nicht, da sie als Variablentypisierung unter die Fähigkeiten und Möglichkeiten unserer Programmiersprache fallen und deshalb erst bei der Detail-Ausarbeitung Bedeutung erhalten, genau wie die sonstigen internen Variablen und Methoden.

Sicher sind auch sie Handlungsträger unseres Systems, genauso wie Funktionen auch Bestandteile von Impulsen sind – aber beide sind austauschbar, wie die ganze Programmiersprache. Lösungen für die Bearbeitungen, die unser Problem erfordert, sind häufig auf verschiedene Weise realisierbar. Austauschbarkeit bedeutet aber Ununterscheidbarkeit und deshalb keine Information für uns, für unser spezielles Problem, das wir gerade zu beschreiben versuchen. Das ist der Grund, warum wir sie in diesem Stadium der Analyse, wenn es um das ganze Problem geht, völlig übergehen. Erst dann, wenn das Problem selbst geklärt ist, werden wir daran gehen, uns die bestmögliche Programmiersprache auszusuchen und sogar erst anschließend daran die einzelnen Detail-Lösungen aufstellen.

Erst:
- Problem lösen
dann:
- Programmiersprache
danach:
- Detail-Lösungen

Der zentrale Unterschied zwischen „unserem“ Objekt und dem Variablentypus hat damit Ähnlichkeit mit dem Unterschied zwischen Prozess und Funktion: es ist die Unterscheidbarkeit hinsichtlich des Problems, das im Fokus der Aufmerksamkeit steht. Ein Prozess ist das, was wir Thread nannten, eine Wirkungskette des betrachteten Problems, ganz spezifisch von diesem und seinem aktuellen Zustand geprägt. Eine Funktion dagegen ist anonym, vielseitig verwendbar, austauschbar – das Problem ist völlig uninteressant, selbst wenn seine Wirkungsketten schlussendlich nur aus Funktionen aufgebaut sind.

Vielleicht gibt dies auch den Hinweis darauf, warum die Grenze beim Objekt zwischen unserem Handlungsträger und dem üblichen flexiblen Variablentypus oder auch die zwischen Prozess und Funktion gelegentlich schwierig zu bestimmen ist: weil es keine allgemeingültige Größe ist, die sich nicht überwinden lässt, sondern als Programmierbestandteil Menschen überlassen ist. Und ein einziges Statement kann dies bereits ändern und damit die Gesamtkonstruktion angreifen. Warum?

Weil unser Rat zu Unterscheidung von frei verwendbarem, flexiblem Variablentypus und sorgfältig zu analysierenden Handlungsträger folgender ist:

Variable:

> ist eine Sammlung von Methoden mit ihren Variablen nur vom eigenen Input abhängig, so ist sie prinzipiell unabhängig von jedem Problem und vor allem, von jedem möglichen Zustand des Problems, und kann unbekümmert angewendet werden: „variabel hinsichtlich des Problems“.

Handlungsträger:

> ist eine solche Kollektion von Methoden und Variablen jedoch in der Lage, bei Bedarf selbst auf andere „Sammlungen“ zurückzugreifen oder sich selbst aus Datenbanken oder sonstigen Stellen aktuelle Informationen zu besorgen, so ist sie keinesfalls unabhängig vom Problem und seinem akuten Zustand. Dann sollte sie auf keinen Fall mehr als an einer einzigen Stelle vorzufinden sein.

Dabei reden wir wieder von Objektklassen, nicht von Instanzen.

Informationsverarbeitung braucht Kontrolle

Woher unser Ratschlag rührt? Aus dem Wunsch, „Information“ zu verarbeiten und diese benötigt Wiederholbarkeit. Wiederholbarkeit ist aber zentral vom Ausgangszustand abhängig. Soll etwas unkontrolliert wie eine Variable eingesetzt werden, muss dieser Anfangszustand demnach kontrollierbar sein - dies ist am Einfachsten und Sichersten erreichbar dadurch, dass alle verwendeten Werte im eigenen Machtbereich liegen, wie beispielsweise Parameter. Ist dies nicht gewährleistet, muss zur Kontrollierbarkeit des Anfangszustandes dann die Unkontrollierbarkeit der Verwendung geopfert werden - am Zuverlässigsten durch die Einfachverwendung.