Institut für Physikalische Chemie

Vorlesung Physikalische Chemie V: Programmierkonzepte in der Physikalischen Chemie im Wintersemester 2018/19

Dr. Till Biskup

— Glossar zu Lektion 13: „Namen“ —

Hinweis: Die nachfolgend genannten Begriffe und Definitionen erheben keinen Anspruch auf formale Korrektheit, sondern dienen lediglich dem besseren Verständnis der in der Vorlesung behandelten Themen und sind im jeweiligen Kontext zu sehen. Mehrfache, voneinander abweichende Definitionen in unterschiedlichen Kontexten sind daher möglich. Englische Begriffe werden zwar nach Möglichkeit übersetzt, erscheinen aber ggf. unter ihrem englischen Namen in der Liste. Verweise untereinander sind durch ↑ gekennzeichnet.

Attribut im Kontext der ↑objektorientierten Programmierung eine Variable, die innerhalb einer ↑Klasse definiert wird. ↑Methoden operieren auf den Attributen einer ↑Klasse bzw. dem daraus erzeugten ↑Objekt.

CamelCase Schreibweise, bei der zusammengesetzte Worte direkt miteinander verbunden werden und jeder Wortteil mit einem Großbuchstaben beginnt. I.d.R. werden Abkürzungen, die in Großbuchstaben erscheinen, komplett groß geschrieben und das nächste Wort mit einem Großbuchstaben begonnen, z.B. `HTTPResponse`.

Eineindeutigkeit Bijektivität, Eindeutigkeit in beiden Richtungen. Mathematisch die Abbildung eines Elements einer Menge auf genau ein Element einer zweiten Menge und umgekehrt, weshalb Definitions- und Zielmenge die gleiche Mächtigkeit aufweisen. Im Kontext von Namen bedeutet das: Jeder Name bildet auf *genau ein* Konzept ab, so dass man eindeutig vom Namen auf das dahinterstehende Konzept und vom Konzept eindeutig auf den Namen schließen kann.

Funktion im Kontext der ↑strukturierten Programmierung eine Liste von Anweisungen, die eine bestimmte Aufgabe erfüllt und der Programmiersprache unter einem festen Namen bekannt ist.

Iteration eine von zwei Kontrollstrukturen der ↑strukturierten Programmierung, neben der ↑Selektion. Meist als Schleife implementiert, die über alle Elemente einer Liste läuft und für jedes Element bestimmte Anweisungen ausführt.

Klasse (*class*) im Kontext der ↑objektorientierten Programmierung die Blaupause für die Erzeugung eines ↑Objektes; Definition der Daten (↑Attribute) und des zugehörigen Verhaltens (↑Methoden).

Lösungsraum (*solution domain*) Kontext der Programmierer eines Programms, Gegensatz zum ↑Problemraum. Namen aus dem Lösungsraum bestehen i.d.R. aus Begriffen aus der Welt der Programmierung.

magische Zahl i.d.R. jede Zahl außer „0“ und „1“, die im Quellcode vorkommt. Sollte *immer* einer Variablen bzw. Konstanten mit sprechendem Namen zugewiesen werden und dann nur noch indirekt über diese Variable/Konstante verwendet werden.

Methode im Kontext der ↑objektorientierten Programmierung eine ↑Funktion, die innerhalb

einer ↑Klasse definiert wird und auf den ↑Attributen einer ↑Klasse bzw. dem daraus erzeugten ↑Objekt operiert.

Namensraum (*namespace*) Kontext, innerhalb dessen der Name eines Objektes (im allgemeinen Sinn, also Variable, Funktion, Methode, Objekt, Klasse, ...) eindeutig ist. Wird häufig durch Funktionen/Methoden oder in der objektorientierten Programmierung durch Objekte/Klassen hergestellt. Der gleiche Name kann in unterschiedlichen Namensräumen zur Bezeichnung unterschiedlicher Objekte (im allgemeinen Sinn) verwendet werden. Namensräume dienen damit der Organisation und Strukturierung.

Objekt (*object*) im Kontext der ↑objektorientierten Programmierung der grundlegende Baustein eines Programms, bestehend aus den Daten (↑Attribute) und dem zugehörigen Verhalten (↑Methoden).

Problemraum (*problem domain*) Kontext der Fragestellung, die mit einem Programm (d.h. Software) angegangen werden soll, Gegensatz zum ↑Lösungsraum. Namen aus dem Problemraum verweisen i.d.R. auf Konzepte, mit denen die Anwender eines Programms vertraut sind (aber nicht notwendigerweise die Programmierer/Entwickler).

strukturierte Programmierung ein ↑Programmierparadigma, das die Zahl möglicher Kontrollstrukturen auf nur zwei (↑Iteration, ↑Selektion) beschränkt, insbesondere den `goto`-Befehl eliminiert (E. Dijkstra). Idealerweise hat ein Codeblock nur jeweils genau einen Ein- und Ausgang. Nach D. Knuth der systematische Einsatz von Abstraktion, der es ermöglicht, große Programme aus kleine(re)n Komponenten zusammenzusetzen. Wichtige frühe Vertreter strukturierter Programmiersprachen sind C und Pascal. Die meisten heutigen Programmiersprachen (mit Ausnahme der funktionalen Programmiersprachen) unterstützen die strukturierte Programmierung.

Selektion eine von zwei Kontrollstrukturen der ↑strukturierten Programmierung, neben der ↑Iteration. In den meisten Sprachen über Bedingungen (`if...else...end`) realisiert, die sich ggf. beliebig verschachteln lassen.