Buch: Softwareentwicklung für die Naturwissenschaften

Dr. habil. Till Biskup

— Glossar zu Kapitel 11: „Programmierparadigmen“ —

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.

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

funktionale Programmierung ein ↑Programmierparadigma, das auf dem ↑Lambda-Kalkül basiert, sich stark an der mathematischen Definition von Funktionen orientiert und im strengen Sinn auf die explizite Zuweisung von Werten zu Variablen verzichtet. Wichtige Elemente sind ↑Listen und die ↑Rekursion. Durch den Verzicht auf explizite Zuweisungen sind funktionale Programme frei von ↑Seiteneffekten und eignen sich damit hervorragend für Parallelisierung. Wichtige Vertreter funktionaler Programmiersprachen sind Lisp, Haskell und Erlang, die Konzepte finden sich mittlerweile aber auch in vielen anderen Programmiersprachen, darunter Python, C++ und Java.

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).

Lambda-Kalkül von Alonzo Church und Stephen Cole Kleene in den 1930er Jahren eingeführte formale Sprache zur Untersuchung von Funktionen. Beschreibt die Definition von Funktionen und gebundenen Parametern und ist heute ein wichtiges Konstrukt für die Theoretische Informatik, Logik höherer Stufe und Linguistik. Die ↑funktionale Programmierung beruht auf dem Lambda-Kalkül.

Liste (*list*, *array*) geordnete Sequenz von Elementen, u.a. wesentliches Element in der ↑funktionalen Programmierung

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.

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

objektorientierte Programmierung ein ↑Programmierparadigma, bei dem Daten (Variablen zugewiesene Werte, als ↑Attribute bezeichnet)

und Funktionen (↑Methoden), die auf diesen Daten (Attributen) operieren, eine Einheit bilden. Die in den ↑Attributen gespeicherten Daten lassen sich nur vermittelt durch (öffentlich zugängliche) ↑Methoden der ↑Klasse bzw. des daraus erzeugten ↑Objektes ansprechen. Es gibt eine klare Trennung zwischen öffentlicher Schnittstelle und internen Verarbeitungsroutinen. Wichtige Vertreter objektorientierter Programmiersprachen sind Smalltalk, C++ und Java, aber auch Python.

Paradigma nach Thomas S. Kuhn ein Satz allgemein anerkannter wissenschaftlicher Leistungen, der für eine gewisse Zeit einer Gemeinschaft von Fachleuten maßgebende Probleme und Lösungen liefert

Programmierparadigma ein ↑Paradigma der Art zu programmieren. Wichtige Beispiele sind ↑strukturierte Programmierung, ↑objektorientierte Programmierung und ↑funktionale Programmierung.

Rekursion Aufruf einer Funktion in ihrer eigenen Definition. Allgemein lässt sich eine Rekursion immer auch als ↑Iteration und umgekehrt ausdrücken. Da die ↑funktionale Programmierung keine ↑Iteration kennt, ist hier nur die Rekursion möglich. Die Definition über eine Rekursion ist meist eleganter, die über eine ↑Iteration oft effizienter, was den Speicherbedarf angeht.

Seiteneffekt *side effect*, Wirkung; in der theoretischen Informatik die Veränderung des Programmzustands einer abstrakten Maschine. In der Praxis der Programmierung meist die Auswirkung einer Zuweisung eines Wertes zu einer Variablen, die außerhalb des konkret betrachteten Kontextes liegt. Da in der ↑funktionalen Programmierung auf Zuweisungen gänzlich verzichtet wird, sind funktionale Programmiersprachen frei von Seiteneffekten, was sie für die parallele Verarbeitung prädestiniert.

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

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.