Java-Tutorial

– Programmieren lernen –

Karsten Brodmann

www.punkt-akademie.de

(Folge 2 - 21.12.2022)

Java-Programme

Der Begriff des „Programms“ wird in der deutschen Sprache vielfältig verwendet. So beschließt unsere allseits geliebte Bundesregierung ständig Programme, um Geflüchtete zu unterstützen, den Energieverbrauch zu senken, die Steuern (angeblich) zu vereinfachen und so weiter. Auch das Fernsehen bietet uns täglich verschiedene Programme. Wenn wir also von Programmen sprechen, sollten wir den Begriff für unser betrachtetes Anwendungsgebiet, die Java-Programmierung, spezifizieren.

Algorithmus, Programm, Prozess

Algorithmus Eine endlich lange Vorschrift, bestehend aus Einzelanweisungen, heißt *Algorithmus*.

Programm Ein für einen Compiler oder Interpreter formulierter Algorithnus heißt *Programm*.

Prozess Ein Programm in Ausführung heißt Prozess.

Beachten Sie: Die Anzahl der Anweisungen eines Algorithmus ist endlich. Das sagt nicht aus, dass die Laufzeit eines Programmes in Ausführung endlich sein muss!

Ein Backrezept ist ein Algorithmus. Dieser ist für Menschen geschrieben. Üblicherweise, wenn der betreffende Mensch des Lesens mächtig ist, versteht er die darin enthaltenen Anweisungen und kann sie umsetzen.

Programmiersprachen wie Java sind ebenso für Menschen geschrieben. Sie verwenden Anweisungen, die vornehmlich der englischen Sprache entlehnt sind. Das soll es dem menschlichen Programmierer leichter machen, Programm in der Sprache zu schreiben

und/oder entsprechende Quellcodes zu lesen und zu verstehen. Um aus einem Programmcode ein Programm zu machen, also eine dem Computer verständliche Anweisungsfolge, verwendet man einen Compiler oder Interpreter.

Ein *Compiler* übersetzt einen textuellen Programmcode in einen binären Code. Das kann ein Maschinencode sein, der direkt ausführbar ist, oder im Falle von Java ein maschinenunabhängiger Bytecode.

Ein *Interpreter* unterscheidet sich ganz wesentlich von einem Compiler. Er liest den Quellcode bei jeder anstehenden Programmausführung neu ein, analysiert die Programmanweisungen, übersetzt sie zur Laufzeit und bringt sie so zur Ausführung. Der Übersetzungsprozess wird also im Laufe des Lebens eines Programmes vielfach wiederholt ausgeführt, während bei einem Compiler nur ein Übersetzungslauf notwendig ist.

Anweisungen, Ablaufprotokoll (Trace)

Elementare Anweisungen sind Einzelanweisungen. Beispiele:

```
teile x durch 2
erhöhe i um 1
```

Strukturierte Anweisungen beinhalten eine Kontrollstruktur. In Abhängigkeit einer Bedingung werden eine oder mehrere Teilanweisungen formuliert. Beispiele:

```
SOLANGE x < 0 WIEDERHOLE:
  gib x aus
  erhöhe x um 1

WENN alter >= 18:
  DANN  gib "Du bist volljährig." aus
  SONST gib "Du bist minderjährig." aus
```

Hier ein Beispiel für einen Algorithmus in umgangssprachlicher Form:

```
1 lies x ein
2 setze z = 0
3 SOLANGE x != 1 WIEDERHOLE
4   WENN x gerade:
5     DANN halbiere x
6     SONST verdreifache x und erhöhe um 1
7   erhöhe z um 1
8 gib z aus
```

Der dargestellte Algorithmus implementiert eine Collatz-Folge. Beginnend mit einer beliebigen positiven Ganzzahl zählt er die Anzahl der Iterationen, die erforderlich ist, die Startzahl auf den Wert 1 zu transformieren. Anhand der Einrückungen werden die Abhängigkeiten der Anweisungen von Bedingungen deutlich. So werden die Anweisungen

der Zeilen 4 bis inklusive 7 solange wiederholt, wie der Wert der Variablen x ungleich 1 ist.

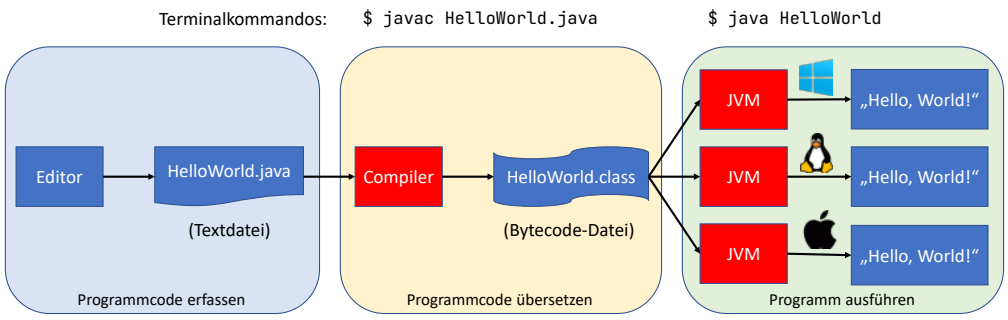
Weil in einer für Menschen verständlichen Form notiert, können wir den Algorithmus mithilfe von Papier und Bleistift ausführen. Notieren wir die dabei durchlaufenen Status, nennen wir das ein *Ablaufprotokoll*. Wird ein solches Ablaufprotokoll von einem Computerprogramm erstellt, wird oft der englische Begriff *Trace* verwendet.

Zeile	x	z	Zeile	x	z
	–	–	5	8	3
1	3	–	7	8	4
2	3	0	5	4	4
6	10	0	7	4	5
7	10	1	5	2	5
5	5	1	7	2	6
7	5	2	5	1	6
6	16	2	7	1	7
7	16	3	8	1	7

Um einen solchen Algorithmus einem Computer verständlich zu machen, bedienen wir uns einer Programmiersprache. In diesem Tutorial ist das Java.

Schema der Java-Programmierung

An dieser Stelle möchte ich Sie mit dem schematischen Ablauf der Programmerstellung und -ausführung von Java-Programmen vertraut machen.



Der Programmcode wird mithilfe eines Texteditors oder auch einer speziell für Java entwickelten Programmierumgebung (IDE) erfasst. Bekannte Programme dieser Kategorie sind IntelliJ, Eclipse oder NetBeans. Der Quellcode, ob mit einem einfachen Texteditor oder einer komplexen IDE erstellt, ist als Textdatei mit der Dateiendung `.java` abzuspeichern.

Das JDK (Java Development Kit) beinhaltet, neben weiteren Werkzeugen, vor allem den Java-Compiler sowie eine JVM (Java Virtual Machine). Letztere ist auch in einem

sogenannten JRE (Java Runtime Environment) enthalten, welches Anwender Ihrer Programme benötigen, die Java-Programme lediglich ausführen wollen.

JDK und JRE sind plattformabhängige Implementierungen. Sie müssen sie, wenn Sie sie nutzen wollen, in der für Ihren Computertypus und das verwendete Betriebssystem passenden Variante installieren.

Der Java-Compiler übersetzt den Quellcode in einen plattformunabhängien *Bytecode*. Dieser ist einem plattformabhängigen Maschinencode nicht unähnlich. Tatsächlich ist er aber für eine virtuelle Maschine formuliert, nämlich eine JVM (Java Virtual Machine). Das ist eine Software die, wie bereits gesagt, plattformabhängig ist. Für ein Java-Programm, welches im Bytecode vorliegt, präsentiert sie aber einen virtuellen Computer, der immer gleich ist, egal auf welcher Plattform die JVM ausgeführt wird. Daher ist es auch egal, auf welcher Art von Computer Sie den Bytecode erstellen, der in einer Datei mit der Dateiendung `.class` gespeichert wird.

Plattformunabhängigkeit

Wenn Sie ein Notebook mit beispielsweise einem Intel-Prozessor besitzen, wissen Sie, keine Programme installieren und ausführen zu können, die für ARM-Prozessoren übersetzt sind. Analog verhält es sich, wenn Sie Microsoft Windows als Betriebssystem nutzen. Dann können Sie keine Programme ausführen, die für einen Mac übersetzt sind, auch wenn es sich dabei um Programme handelt, die für Macs mit einem Intel-Prozessor gedacht sind. Die Kombination aus Hardware und Betriebssystem definiert eine Plattform.

Plattformunabhängigkeit ist übrigens keine Erfindung der Neuzeit. Bereits die Programmiersprache C wurde Ende der 60er Jahre unter diesem Aspekt entwickelt. Die damalige Idee sah vor, den Programmcode für ein Programm einmalig zu schreiben und ihn dann auf verschiedenen Plattformen übersetzen und ausführen zu können. In der Tat funktioniert das auch für ganz primitve Programme. Weil C aber relativ nah an der Hardware implementiert ist, auch die Spezifikationen der Sprache Spielraum für zum Beispiel Datentypgrößen lassen, funktioniert das in der Praxis selten. Mittels `define`-Anweisungen (Makros) kann man versuchen, den Code für verschiedene Plattformen in einer Quellcodedatei zu schreiben. Der wahre Jakob ist das aber nicht. Pflege und Wartung sind schwierig und zeitaufwendig.

Zeit ist jedoch auch der ausschlaggebende Aspekt, weshalb über Plattformunabhängigkeit nachgedacht wird. Die menschliche Programmierleistung ist am Ende der Kosten treibende Faktor in der Softwareentwicklung.

Anmerkung: Java ist bezüglich der Plattformunabhängigkeit die bei weitem einfachste und zuverlässigste Programmiersprache. Wenn Sie einmal versucht haben, ein Python-Programm, welches Zusatzbibiotheken verwendet, die nicht zum Standard von Python gehören, auf verschiendenen Betriebssystemen auszuführen, dann wissen Sie, dass es mit

der Plattformunabhängikeit dann doch nicht so toll ist. Bei Java ist das dagegen wirklich überhaupt keine Herausforderung.