1 Einführung

Wer die Zukunft erforschen will, muss die Vergangenheit kennen.
– Chinesische Weisheit

Die Programmiersprache Java wird an vielen Schulen und Hochschulen als Basis für den Einstieg in die Programmierung verwendet. Beim Erlernen dieser erfolgreichen Programmiersprache möchte dieses Buch Sie unterstützen. Besonderer Wert wurde bei der Erstellung darauf gelegt, dass keine Programmierkenntnisse vorausgesetzt werden, sodass Sie das Programmieren von Grund auf lernen können. Bei der Auswahl der Entwicklungsumgebung wurde darauf geachtet, dass diese eine komfortable Arbeit ermöglicht und dass sie frei und für alle gängigen Betriebssysteme (also Windows, Linux und macOS) gleichermaßen verfügbar ist. *Eclipse* ist für den Einstieg nicht ganz einfach; Sie werden mit diesem Buch aber den Umgang damit lernen und leicht damit arbeiten können.

Dieses Buch führt Sie schrittweise in die Programmierung mit *Java* ein. Sie lernen alle wichtigen Sprachstrukturen anhand von Anwendungsbeispielen kennen, damit Sie schon bald Ihre eigenen Programme entwickeln können. An die meisten Kapitel schließen sich Übungsaufgaben an. Durch deren Bearbeitung wird der Umgang mit den neuen Sprachelementen und Strukturen eingeübt und gefestigt. Musterlösungen zu den Übungsaufgaben finden Sie als Quellcodedateien im Downloadmaterial oder im Anhang. Verweise auf die betreffenden Fundstellen sind im Text angegeben. Neben den Musterlösungen erhalten Sie als Downloadmaterial auch die Versionen des *JDK* (*Java Development Kit*) und der Entwicklungsumgebung Eclipse für die Betriebssysteme Windows, Linux und macOS. Da in einem einführenden Buch nicht alle Sprachdetails bis in die letzten Einzelheiten behandelt werden können, stellt das Buch »Java ist auch eine Insel« vom Rheinwerk Verlag eine sehr gute Ergänzung dar. Weitere Hinweise zu ergänzender Literatur, die zum Teil kostenlos im Internet zur Verfügung steht, sind im Literaturverzeichnis am Ende des Buches zusammengestellt.

1.1 Was bedeutet Programmierung?

Bevor Sie mit dem Programmieren loslegen können, brauchen Sie ein Grundverständnis dafür, was Programmierung überhaupt ist und wie sie »funktioniert«. Das sollen Ihnen die nächsten Abschnitte näherbringen.

1.1.1 Von den Anfängen bis heute

Das erste Computerprogramm, das jemals erstellt wurde, wird einer Frau zugeschrieben. Die britische Mathematikerin *Ada Lovelace* (1815–1852) entwickelte einen schriftlichen Plan, wie man mithilfe der mechanischen Rechenmaschine von *Charles Babbage* (1791–1871) Bernoulli-Zahlen

berechnen kann. Das ist umso erstaunlicher, weil zu diesem Zeitpunkt lediglich Pläne für diese Rechenmaschine vorlagen. Diese mechanische Rechenmaschine (*Analytical Engine*), die zu Lebzeiten ihres Erfinders nie gebaut wurde, gilt als Vorläufer der heutigen Computer, und der Plan von Ada Lovelace wird als das erste Computerprogramm angesehen.

Seit der Erfindung der ersten mechanischen Rechenmaschinen bis zu den heutigen elektronischen Computersystemen haben sich viele weitreichende Veränderungen eingestellt. Das gilt sowohl für die Hardware als auch für die Arbeitsmittel, die zur Programmierung verwendet werden. Allerdings haben die grundlegenden Zusammenhänge bis heute ihre Gültigkeit bewahrt. Die frühen Programmierwerkzeuge orientierten sich noch sehr an der Hardwarestruktur und machten es notwendig, dass der Programmierer die Bausteine des Prozessors explizit kannte und ansprechen konnte. Zum Addieren der beiden Zahlen 12 und 38 können Sie in Java einfach die Anweisung

```
x = 12 + 38;
```

verwenden. Diese Schreibweise ist nicht neu, denn wir verwenden sie auch in der Mathematik, wenn die Variable `x` die Summe der beiden Zahlen 12 und 38 annehmen soll. In anderen aktuellen Programmiersprachen wie C/C++ sieht eine Addition genauso aus, und sie gilt unabhängig vom verwendeten Rechner bzw. dem darin verbauten Prozessor. In einer älteren hardwarenahen Sprache wie Assembler mussten Sie dafür etwa folgende Anweisungsfolge verwenden:

```
mov eax, 12
add eax, 38
```

Zuerst wurde die Zahl 12 in ein Prozessorregister mit dem Namen `eax` geschrieben (`mov` steht für das englische *move*), um in einem zweiten Schritt den Registerinhalt um den Wert 38 zu erhöhen. Der Programmierer musste z. B. wissen, wie die Register des Prozessors heißen und mit welchen Registern eine Addition ausgeführt werden kann. Dass unterschiedliche Prozessoren auch unterschiedliche Registerbezeichnungen verwenden können, hat das Programmieren zusätzlich erschwert.

Die frühen Computersysteme waren so einfach aufgebaut, dass dies auch noch zu leisten war. Moderne Computersysteme sind heute so komplex und entwickeln sich so schnell weiter, dass es kaum noch möglich ist, die Prozessordetails zu kennen. Glücklicherweise haben sich in dem gleichen Maße, in dem die Komplexität der Systeme zugenommen hat, auch die Programmierwerkzeuge weiterentwickelt. Zu diesen gehören Editoren, die schon beim Schreiben von Programmanweisungen auf mögliche Fehler aufmerksam machen und dabei helfen, den Programmtext übersichtlich zu formatieren. Auch Übersetzungsprogramme, die die Programmdateien so aufbereiten, dass sie auf verschiedenen Rechnern mit unterschiedlichen Prozessoren ausgeführt werden können, gehören dazu. Die Programmiersprachen haben sich der menschlichen Sprache angenähert und können wesentlich leichter als die frühen, sehr hardwarenahen Sprachen erlernt werden.

1.1.2 Wozu überhaupt programmieren?

Die Programmierung, d. h. die Erstellung eines Computerprogramms, besteht darin, die Lösungsschritte für eine Problemstellung so zu formulieren, dass sie von einem Computersystem ausgeführt werden können. Das bedeutet, dass dem Programmierer die

notwendigen Lösungsschritte bekannt sein müssen. Entweder muss er sich den Lösungsweg selbst erarbeiten oder dieser wird ihm zur Verfügung gestellt. Beim Programmieren wird dieser allgemein formulierte Lösungsweg in eine Programmiersprache übertragen, die vom Computersystem weiterverarbeitet werden kann.

Da die Programmierung einen zeitaufwendigen Prozess darstellt, muss die Frage beantwortet werden, wann es sich lohnt, diese Zeit zu investieren. Die Übertragung einer Aufgabenstellung auf ein Computersystem ist dann sinnvoll, wenn dieses System seine speziellen Fähigkeiten auch ausspielen kann. Diese Fähigkeiten sind vor allem:

- die hohe Verarbeitungsgeschwindigkeit
- die zuverlässige Wiederholbarkeit

Die hohe Verarbeitungsgeschwindigkeit kann nur genutzt werden, wenn die zeitaufwendige Programmerstellung nicht ins Gewicht fällt. Das ist immer dann der Fall, wenn das Programm häufig verwendet wird und oft seinen Geschwindigkeitsvorteil ausspielen kann. Gleiches gilt für die hohe Zuverlässigkeit. Im Gegensatz zum Menschen zeigt ein Computersystem bei der Ausführung sich ständig wiederholender Anweisungen keinerlei Ermüdungserscheinungen. Konzentrationsfehler wegen Übermüdung sind ihm vollkommen fremd.

Die Arbeitsschritte zur Lösung einer Problemstellung werden allgemein auch als *Algorithmus* bezeichnet. Dieser Begriff wurde ursprünglich für die Beschreibung von Lösungswegen in der Mathematik verwendet und später auf die Informatik, die Wissenschaft, der die Programmierung zuzuordnen ist, übertragen. Jedem Computerprogramm liegt ein Algorithmus zugrunde. Deshalb liefert die Definition des Begriffs entscheidende Hinweise für die Beantwortung der Frage, ob eine Problemstellung mit einem Computerprogramm gelöst werden kann.

Ein Algorithmus muss die folgenden Anforderungen erfüllen:

- Er muss aus Arbeitsschritten bestehen, die zur Lösung einer Problemstellung führen.
- Er muss in einem endlichen Text vollständig beschreibbar sein.
- Jeder Schritt muss zu einem eindeutigen Zwischenergebnis führen.
- Er muss für gleiche Eingabewerte immer zum gleichen Ergebnis führen.
- Das Verfahren muss zum richtigen Ergebnis führen.
- Das Verfahren muss allgemeingültig sein, d. h., es muss auf alle zulässigen Daten anwendbar sein.

Die letzte Eigenschaft macht deutlich, dass ein Algorithmus der Lösung eines allgemeinen Problems dienen muss. Ein Programm, das nur die Zahlen 3 und 5 addieren kann, ergibt keinen Sinn – es muss in der Lage sein, zwei beliebige Zahlen zu addieren. Das bedingt aber, dass dem Programm mitgeteilt werden muss, welche beiden Zahlen addiert werden sollen. Dieses einfache Beispiel zeigt, dass die Lösung allgemeiner Probleme den Dialog zwischen Anwender und Programm notwendig macht.

Häufig erfordert der Entwurf eines Algorithmus als Vorstufe zur Programmierung Kreativität und Einfallsreichtum. Er wird oft als der schwierigste Teil im Prozess der Programmentwicklung bezeichnet.

1.1.3 Hilfsmittel für den Programmentwurf

Der Entwurf eines Algorithmus kann unabhängig von der zu verwendenden Programmiersprache erfolgen. Alle prozeduralen Programmiersprachen stellen die gleichen Sprachstrukturen zur Verfügung. Deshalb liegt es nahe, allgemeingültige Hilfsmittel zur Entwicklung von Algorithmen zu entwickeln und einzusetzen.

Computerprogramme bestehen sehr schnell aus umfangreichen Textdateien, die dann entsprechend unübersichtlich werden. Gerade in der Planungsphase ist es wichtig, den Überblick zu behalten und eine Grobstruktur des fertigen Programms herauszuarbeiten, die in weiteren Phasen der Entwicklung verfeinert wird. Zur übersichtlichen Darstellung von Programmstrukturen eignen sich grafische Symbole. Für die Programmierung werden hierfür *Programmablaufpläne* (DIN 66001) oder *Nassi-Shneiderman-Struktogramme* (DIN 66261) verwendet. Die Gegenüberstellung in Abbildung 1.1 zeigt die in beiden Darstellungsformen verwendeten Symbole.

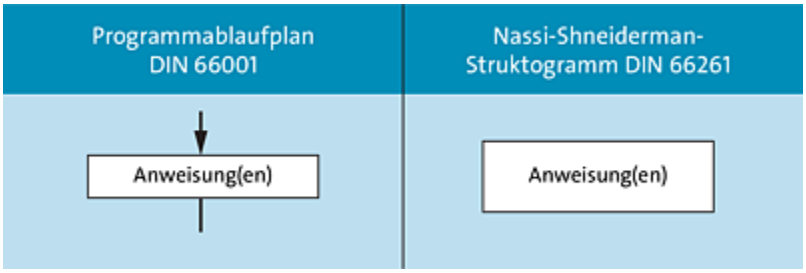

Abbildung 1.1 Einzelanweisung bzw. Anweisungsblock

In beiden Formen wird das Rechtecksymbol zur Darstellung einer einzelnen Anweisung oder eines zusammengehörenden Anweisungsblocks verwendet. Damit kann die Darstellung einer Programmlogik sehr detailliert und nahe am späteren Programmtext, aber auch sehr komprimiert dargestellt werden. In den Programmsymbolen können frei formulierte Aufgabenbeschreibungen stehen.

Mit einer *Anweisungsfolge* kann die Reihenfolge der Abarbeitung von Anweisungen verdeutlicht werden (siehe Abbildung 1.2). Im Programmablaufplan (PAP) wird die Reihenfolge der Abarbeitung zusätzlich durch Pfeile verdeutlicht. Die Abarbeitung in der Struktogrammdarstellung erfolgt grundsätzlich von oben nach unten.

Lediglich drei Grundstrukturen werden benötigt, um einen Algorithmus zu beschreiben:

- Anweisungsfolge (Sequenz)
- Auswahlstruktur (Selektion)
- Wiederholungsstruktur (Iteration)

Zur Vereinfachung stellen Programmiersprachen unterschiedliche Varianten von Auswahl- und Wiederholungsstrukturen zur Verfügung.

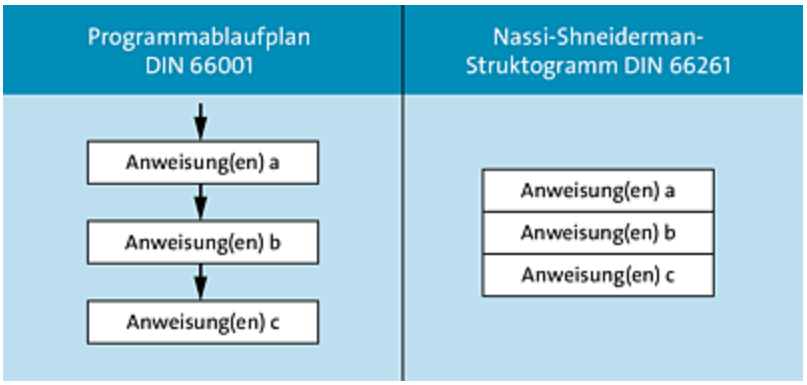

Abbildung 1.2 Anweisungsfolge

Die einfachste Auswahlstruktur ist die *bedingte Verzweigung* (siehe Abbildung 1.3). Hierbei stehen zwei Alternativen zur Auswahl, die an eine Bedingung geknüpft sind. Ist die Bedingung erfüllt, wird die Anweisung bzw. werden die Anweisungen *a* ausgeführt, ansonsten die Anweisung bzw. die Anweisungen *b*.

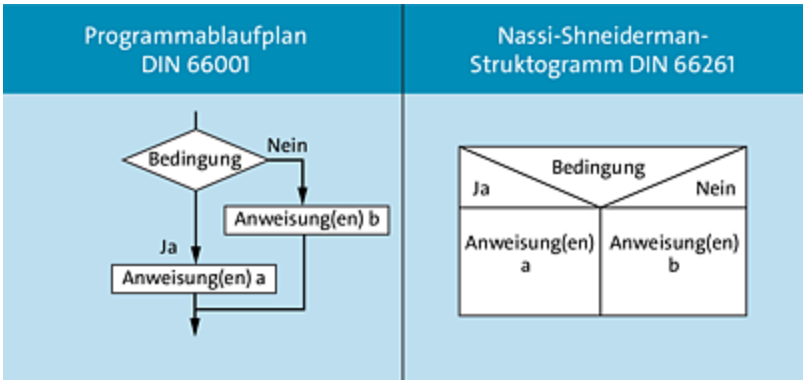

Abbildung 1.3 Bedingte Verzweigung

Wie diese Grundstrukturen in der Programmiersprache Java realisiert werden, erfahren Sie in Kapitel 3, »Kontrollstrukturen«. Im folgenden Abschnitt wird der gesamte Prozess der Anwendungsentwicklung anhand eines einfachen Beispiels erläutert.

1.1.4 Von der Idee zum Programm

Am Anfang des Entwicklungsprozesses steht immer eine Idee oder eine Vorstellung davon, was ein Programm leisten soll. Stellen Sie sich vor, Sie möchten ein kleines Computerspiel programmieren. Das Computerprogramm soll eine zufällig bestimmte ganze Zahl im Bereich von 1 bis 100 vorgeben, die vom Anwender erraten werden soll. Der Anwender soll beliebig viele Versuche haben, um die gesuchte Zahl zu erraten. Wurde die gesuchte Zahl erraten, soll das Programm dem Anwender die Anzahl der benötigten Versuche mitteilen.

Diese kurze Beschreibung steht für die *Produktdefinition*. Im professionellen Bereich spricht man vom *Lastenheft*, das möglichst präzise beschreibt, was das fertige Programm leisten soll. Für uns interessanter ist aber vielleicht eine Abwandlung des Lastenhefts, die als *User-Story* bezeichnet wird. Sie hat ihren Ursprung in der agilen Softwareentwicklung, beschränkt sich bei