Jens-Peter Misch, Ingo Patett

Java 4 U

Programmentwicklung mit Java

5. Auflage

Bestellnummer 01170

Bildungsverlag EINS
westermann

Die in diesem Werk aufgeführten Internetadressen sind auf dem Stand zum Zeitpunkt der Drucklegung. Die ständige Aktualität der Adressen kann vonseiten des Verlages nicht gewährleistet werden. Darüber hinaus übernimmt der Verlag keine Verantwortung für die Inhalte dieser Seiten.

service@bv-1.de
www.bildungsverlag1.de

Bildungsverlag EINS GmbH
Ettore-Bugatti-Straße 6-14, 51149 Köln

ISBN 978-3-427-**01170**-5

westermann GRUPPE

Vorwort

Dieses Buch setzt ein Mindestmaß an Grundkenntnissen im Umgang mit dem PC voraus. Es ermöglicht dem Leser im Selbststudium das Erlernen der Programmiersprache Java. Java ist eine von der Firma SUN entwickelte Sprache und seit 1995 auf dem Markt. Sie ist immer noch eine relativ junge Programmiersprache und wird dadurch noch vielen Wandlungen unterworfen sein.

Im Jahre 2010 wurde SUN durch die Firma Oracle übernommen und Oracle veröffentlichte – nach längerer Entwicklungszeit – im Herbst 2017 die Version Java 9.
Änderungen, die die Grundlagen der Sprache betreffen, sind in der 5. Auflage mit in das Schulbuch eingeflossen. Ferner wurde der Inhalt an die aktuellen Versionen von Tomcat, MySQL und der Servlet-API von Oracle angepasst. Als Betriebssystem kommt nun Windows 10 zum Einsatz. Inhalte und Aufgaben können aber auch unter Linux oder einem anderen Betriebssystem umgesetzt werden. Die Installation von Tomcat, MySQL oder der Servlet API ist dann entsprechend anzupassen.

Die einzelnen Kapitel des Werkes bauen aufeinander auf. Übungen bzw. Aufgabenstellungen führen oft über mehrere Kapitel des Werkes hinweg und betrachten Sachverhalte aus verschiedenen Blickwinkeln.

In dem Kapitel Datenbank werden minimale Kenntnisse der grundlegenden SQL-Befehle vorausgesetzt. Auch zum Kapitel Servlet sollten zum besseren Verständnis Basiskenntnisse zum Aufbau eines HTML-Dokumentes vorhanden sein.

Inhaltsverzeichnis

Java ist eine sehr sichere Sprache und verlangt vom Programmierer bei Eingaben über die Tastatur, dass er sich um auftretende Fehler kümmert. Der Programmierer kennzeichnet die Methode, in der ein Fehler auftreten kann, durch das Schlüsselwort **`throws`** `<Fehlergruppe>` nach dem Methodenkopf (Zeile 4). Ein Fehler wird jetzt an die virtuelle Maschine zurückgegeben, später werden Mechanismen gezeigt, wie im Programm Fehler abgefangen werden können.

Grundsätzlich werden Fehler zu Gruppen zusammengefasst. Die bei der Eingabe möglichen Fehler sind in der Gruppe **`IOException`** (IO = input/output; exception = Ausnahme/Fehler) enthalten. Da diese Gruppe nicht mehr zum allgemeinen Sprachumfang gehört, muss sie zusätzlich über die Anweisung (Zeile 1): **`import`** `java.io.*;` eingebunden werden.

```
 [1]       import java.io.*;
 [2]       public class EinfachsteEingabe {
 [3]
 [4]        public static void main(String argv[])throws IOException {
 [5]            int zeichen;
 [6]            zeichen=System.in.read();
 [7]            System.out.print("ASCII-Wert: ");
 [8]            System.out.println(zeichen);
 [9]            System.out.print("ASCII-Zeichen");
[10]            System.out.print((char)zeichen);
[11]            System.out.println("Programmende EinfachsteEingabe.");
[12]          }
[13]       }
```

Die Methode `read ()` liest genau ein Zeichen über Tastatur ein und liefert als Ergebnis den ASCII-Wert (Zeile 7). Will man das ASCII-Zeichen auf dem Bildschirm ausgeben, so muss man den ASCII-Wert in ein ASCII-Zeichen umwandeln. Dieser Vorgang wird `typecast` genannt.

- Allgemein: `(Typ)Bezeichner`
- Hier: `(char)zeichen`

Will man komfortable Eingaben über die Tastatur erledigen, so gibt es in Java die Klasse **`BufferedReader`**. In dieser Klasse ist die Methode `readLine ()` definiert, die zeilenweise Zeichenketten einlesen kann. Damit entfällt auch das umständliche Umwandeln von Uni-Code-Wert/Zeichen.

Zeilenweises Einlesen mit der Tastatur erfolgt in Java über einen Puffer, der uns nach Eingabe des Returns (bzw. Enter) die eingegebenen Zeichen als String (Zeichenkette) zur Verfügung stellt. Diese gepufferte Eingabe wird auch beim Einlesen aus Textdateien oder beim Lesen von Textdateien, die auf einem anderen Rechner (z. B. HTML-Dokumente) liegen, benutzt.

Der Aufbau der Anweisung ist sehr komplex und wird in den kommenden Kapiteln nach und nach erklärt. Sie müssen die Anweisung zu diesem Zeitpunkt noch nicht bis ins Letzte verstanden haben.

Quellcode Eingabe.java:

```
 [1]       import java.io.*;
 [2]       public class Eingabe {
 [3]
 [4]        public static void main(String argv[])throws IOException {
 [5]           String str;
 [6]           BufferedReader input = new BufferedReader(
 [7]                                  new InputStreamReader(
 [8]                                      System.in));
 [9]           System.out.print("Name eingeben: ");
[10]           str = input.readLine();
[11]           System.out.print("Guten Tag Benutzer ");
[12]           System.out.println(str);
[13]           System.out.println("Programmende Eingabe.");
[14]         }
[15]       }
```

Wie im Quellcode zu sehen, ist eine Eingabe über Tastatur in einer Anwendung nicht ganz so problemlos wie die Vereinbarung einer einfachen Variablen. Die Zeilen 6 bis 8 wurden aus Platzgründen auf 3 Zeilen verteilt, was aber nicht zwingend notwendig ist.

Um die Eingabe über Tastatur zu ermöglichen, müssen immer drei Bestandteile im Quellcode ergänzt werden:

1. Importieren aller Klassen aus dem Paket `java.io` durch `import java.io.*;` (Zeile 1)
2. Ergänzen des Methodenkopfes, in der die Eingabe über Tastatur gemacht werden soll, um die Schlüsselwörter `throws IOException` (Zeile 4)
3. Erzeugen eines Objekts (Objekt ist eine „spezielle Variable“), das zur Eingabe über Tastatur genutzt werden kann (Zeile 6–8)

Diese drei Schritte sind immer notwendig, um Eingaben über die Tastatur zu tätigen. Jetzt ist es möglich, mit der Anweisung `input.readLine()` die gesamte eingegebene Zeile einzulesen (Zeile 10). Damit die Information nicht verloren geht, wird sie einer Variablen vom Typ `String` zugewiesen (hier str, Zeile 10).

1.8.2 Einlesen über Tastatur mit der Scanner-Klasse

Ab der Java-Version 1.6 steht in dem Paktet java.util die Klasse Scanner zur Verfügung. Mithilfe der Klasse Scanner ist das Einlesen über Tastatur (inkl. Umwandlung in den gewünschten Datentyp – int usw.) erleichtert worden.

Benutzen wir die Aufgabenstellung des vorangegangenen Kapitels, bei der es um das Einlesen eines Benutzernamens geht. Schauen Sie sich folgenden Quellcode an:

Quellcode EingabeNEU.java:

```
 [1]       import java.util.*;
 [2]       public class EingabeNEU {
 [3]
 [4]         public static void main(String argv[]) {
 [5]           String str;
 [6]           Scanner input = new Scanner(System.in);
 [7]           System.out.print("Name eingeben: ");
 [8]           str = input.nextLine();
 [9]           System.out.print("Guten Tag Benutzer ");
[10]           System.out.println(str);
[11]           System.out.println("Programmende Eingabe.");
[12]         }
[13]       }
```

In Zeile 1 steht eine Import-Anweisung für das Paket java.util, da dort die Klasse Scanner definiert ist. In Zeile 6 wird ein Objekt der Klasse Scanner angelegt, das als Eingabestrom die Standardeingabe (System.in) nutzt. Ab dieser Zeile sind nun einfachste Leseoperationen möglich. In Zeile 8 wird die ganze Zeile (bis zum „ENTER“) von der Konsole gelesen und als Zeichenkette (String) der Variablen str übergeben.

In den nun folgenden Seiten wird diese Variante zum Lesen von der Konsole verwendet.

1.9 Lineare Programme

Sollen für die Lösung einer Aufgabe mehrere Anweisungen hintereinander ausgeführt werden, können sie im Algorithmus in der gleichen Reihenfolge beschrieben werden.

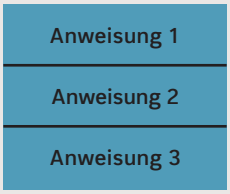

Abb. 1.2: Struktogramm lineare Programme

Auftrag

`(Summe.java)` Es soll eine Applikation erstellt werden, die die Summe zweier einzugebender ganzer Zahlen bildet und das Ergebnis auf dem Bildschirm ausweist.

Auf das Wesentliche verkürzt, hat das Struktogramm folgendes Aussehen:

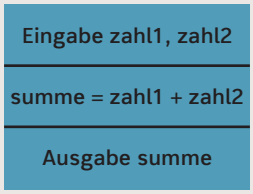

Abb. 1.3: Struktogramm Summe.java

Die Verkürzungen, die hier möglich waren, beziehen sich auf das Thema „benutzerfreundliche Programmierung“. Wenn im Struktogramm steht „Eingabe zahl1, zahl2“, ist damit die benutzerfreundliche Eingabe von beiden Werten gemeint, die die entsprechenden Ausgaben von Eingabeaufforderungen beinhalten. Durch diese Verkürzungen lenken im Struktogramm die benutzerfreundlichen Teile nicht von den wesentlichen Teilen der Programmierung ab.

Quellcode Summe.java (Variante mit BufferedReader):

```
 [1]       import java.io.*;
 [2]       public class Summe {
 [3]
 [4]         public static void main(String argv[])throws IOException {
 [5]           String str;
 [6]           int zahl1,zahl2,ergebnis;
 [7]           BufferedReader input = new BufferedReader(
 [8]                                  new InputStreamReader(
 [9]                                      System.in));
[10]           System.out.print("Geben Sie die erste Zahl ein: ");
[11]           str = input.readLine();
[12]           zahl1 = Integer.parseInt(str);
[13]           System.out.print("Geben Sie die zweite Zahl ein: ");
[14]           str = input.readLine();
[15]           zahl2 = Integer.parseInt(str);
[16]           ergebnis = zahl1 + zahl2;
[17]           System.out.println("Die Summe von "+zahl1+" und "+
[18]                                 zahl2 + "ist: "+ergebnis);
[19]           System.out.println("Programmende Summe.");
[20]         }
[21]       }
```

`Integer.parseInt(str) (Zeile 12,15):`

Alle über `input.readLine()` eingelesenen Informationen werden als String (Zeichenkette) abgespeichert. Da man aber die Informationen nicht als Zeichenkette benötigt, sondern damit rechnen will, muss die Zeichenkette in eine Zahl umgewandelt werden. Diese Umwandlung von `String` nach `int` geht über die Anweisung `Integer.parseInt(str)`. Es ist aber auch möglich, diese Umwandlung direkt über die Scanner-Klasse durchzuführen (siehe folgenden Quellcode, der beide Varianten enthält). Es gibt für alle einfachen Datentypen eine entsprechende Methode (nextFloat, nextDouble, nextByte ...).

Die Möglichkeit der Umwandlung von `String` in einen einfachen Datentyp gibt es für: `byte`, `short`, `int`, `long`, `float` und `double`. Beachten Sie die Groß- und Kleinschreibung.

Weitere Informationen finden Sie in der API-Dokumentation im Paket `java.lang`.

Quellcode Summe.java (Variante mit Scanner):

```
[1]       import java.util.*;
[2]       public class Summe {
[3]
[4]         public static void main(String argv(]) {
[5]           int zahl1,zahl2,ergebnis;
[6]           Scanner input = new Scanner(System.in);
[7]           System.out.print("Geben Sie die erste Zahl ein: ");
[8]           zahl1 = Integer.parseInt(input.nextLine());
[9]           System.out.print("Geben Sie die zweite Zahl ein: ");
[10]          zahl2 = input.nextInt();
[11]          ergebnis = zahl1 + zahl2;
[12]          System.out.println("Die Summe von "+zahl1+" und "+
[13]                                zahl2 + " ist: "+ergebnis);
[14]          System.out.println("Programmende Summe.");
[15]        }
[16]      }
```

`System.out.println(„Die Summe von "+zahl1+" und "+ zahl2 + "ist: "+ergebnis);`

Will man Text und Inhalte von Variablen ausgeben, so kann man dies entweder in getrennten `System.out.print(..)`-Anweisungen angeben oder durch das Verkettungszeichen „+“ in einer Anweisung zusammenfassen.

Beispiel

```
System.out.print(4);
System.out.print("+");
System.out.print(3);
System.out.print("=");
System.out.println(7);
```

Oder in einer Anweisung:

```
System.out.println(4+"+"+3+"="+7);
```

Aufgaben

1. `(Subtraktion.java)` *Verändern Sie das Programm Summe, sodass es die Differenz der beiden eingegebenen ganzen Zahlen ausgibt.*

```
Subtraktion zweier ganzer Zahlen
Zahl 1: 23
Zahl 2: 4
Das Ergebnis der Subtraktion 23-4 ist 19
Programmende Subtraktion.
```

2 Kontrollstrukturen

Im zweiten Kapitel werden die grundlegenden **Kontrollstrukturen**, wie Auswahl, Schleifen und Felder, ausführlichst behandelt. Zusätzlich finden Sie nach jedem Unterkapitel Übungsaufgaben.

2.1 Auswahl (Selektion)

In Abhängigkeit von einer Bedingung können Anweisungen ausgeführt oder weggelassen werden. **Bedingungen** sind in der Regel an Operatoren wie `>` (größer als), `>=` (größer gleich), `<` (kleiner als), `<=` (kleiner gleich), `==` (gleich) und `!=` (ungleich) geknüpft.

2.1.1 Einseitige Auswahl

Bei der **einseitigen Auswahl** wird eine Anweisung oder eine Gruppe von Anweisungen nur dann ausgeführt, wenn eine bestimmte Bedingung erfüllt ist.

Auftrag

`(Warnung1.java)` Es soll eine Applikation erstellt werden, die einen Warnhinweis ausgibt, wenn ein Grenzwert überschritten wird.

Im Struktogramm wird die einseitige Auswahl wie folgt dargestellt:

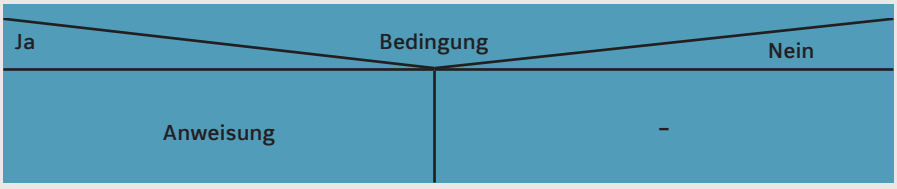

Abb. 2.1: Struktogramm einseitige Auswahl (allgemeine Darstellung)

Die Syntax lautet in Java:

```
if (Bedingung) Anweisung;
```

Die Bedingung muss immer in runden Klammen eingeschlossen sein.

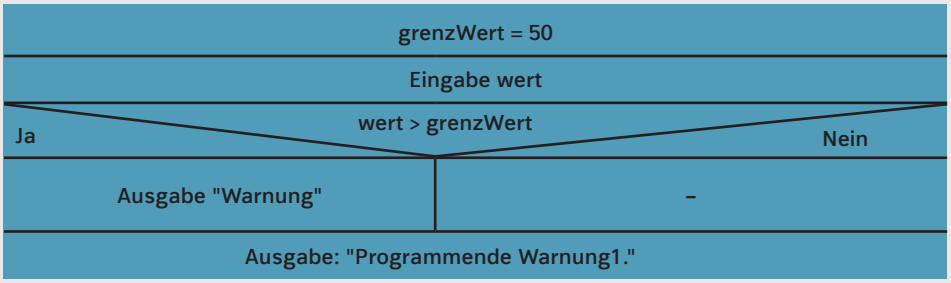

Abb. 2.2: Struktogramm Warnung1.java

Quellcode Warnung1.java:

```
[1]       public class Warnung1 {
[2]
[3]         public static void main(String argv[]) {
[4]           int grenzWert = 50;
[5]           int wert = 320;
[6]           if (wert > grenzWert) {
[7]
[8]               System.out.println("Warnung");
[9]           }
[10]          System.out.println("Programmende Warnung1.");
[11]        }
[12]      }
```

Im Beispiel Warnung1 wurde aus Platzgründen auf die Eingabe über Tastatur verzichtet. Der Variablen wurde direkt ein Wert zugewiesen.

Sollen mehrere Anweisungen in Abhängigkeit der Bedingung ausgeführt werden, muss die Anweisungsgruppe mit `{…}` zu einem Block zusammengefasst werden (Warnung2 – Zeile 7 … 10).

Auftrag

Es soll eine Applikation mit der Programmiersprache Java erstellt werden, die einen Warnhinweis ausgibt, wenn ein Grenzwert überschritten wird (`Warnung2.java`). Außerdem soll eine Variable (anzahlFehler) um eins erhöht werden.

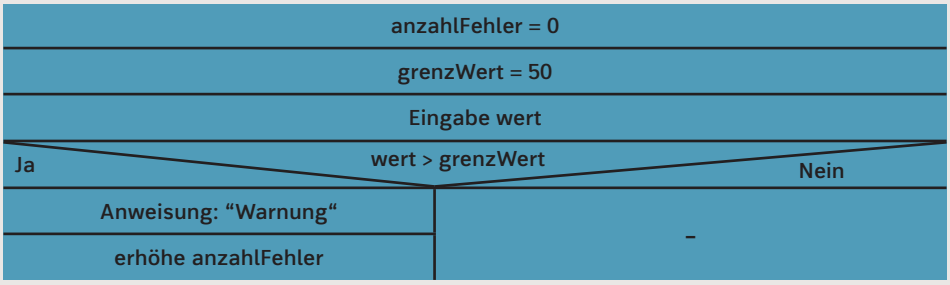

Abb. 2.3: Struktogramm Warnung2.java

Quellcode Warnung2.java:

```
[1]       public class Warnung2 {
[2]
[3]         public static void main(String argv[]) {
[4]           int anzahlFehler = 0;
[5]           int grenzWert = 50;
[6]           int wert = 320;
[7]           if (wert > grenzWert) {
[8]            System.out.println("Warnung");
[9]            anzahlFehler++;
[10]          }
[11]          System.out.println("Anzahl Fehler: "+anzahlFehler);
[12]          System.out.println("Programmende Warnung2.");
[13]        }
[14]      }
```

Die angegebene Bedingung ist entweder wahr oder falsch, sie nimmt also die booleschen Werte `true` oder `false` an. Boolesche Werte können mit den Operatoren „**&&**“ für **UND**, „**||**“ für **ODER** und „**!**“ für **NICHT** verknüpft werden.

a	b	(a && b)	!(a && b)	(a || b)	!(a || b)
false	false	false	true	false	true
false	true	false	true	true	false
true	false	false	true	true	false
true	true	true	false	true	false

Vergleichsoperationen werden vorrangig vor **booleschen Operationen** behandelt. `a && b` ist also nur dann wahr, wenn sowohl a als auch b wahr ist. Bei `a || b` genügt es, wenn mindestens a oder b wahr ist.

Auftrag

`(Warnung3.java)` Es soll eine Applikation erstellt werden, die einen Warnhinweis ausgibt, wenn ein Wert in einem Grenzbereich liegt.

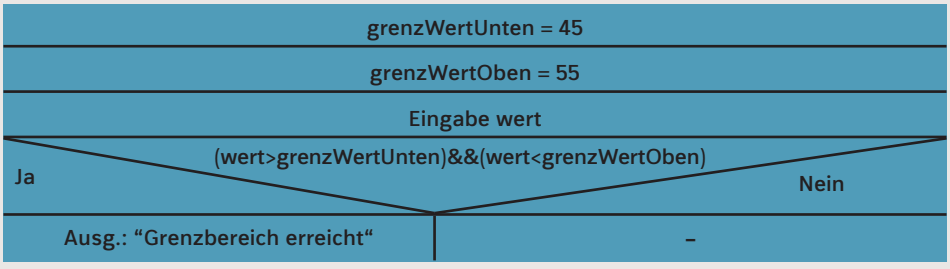

Abb. 2.4: Struktogramm Warnung3.java

Bildquellenverzeichnis

Umschlag:
fotolia.com, New York: links (adempercem), rechts (Lucky Dragon)
stock.adobe.com, Dublin: mitte (Cake78)

Inhalt:
Alle Screenshots stammen vom Autor.