Vorkurs Informatik
Der Einstieg ins Informatikstudium
5. Auflage

Musterlösungen

Heinrich Müller, Frank Weichert

Version vom

18. November 2018

Vorwort

Liebe Leserinnen und Leser des Buches Vorkurs Informatik,

die vorliegende Musterlösung stellt für einen Großteil der Aufgaben des Buches exemplarische Lösungen zur Verfügung zu stellen. Bei den vorliegenden Musterlösungen ist zu beachten, dass diese, speziell bei den Programmieraufgaben, immer nur einen möglichen Lösungsweg aufzeigen. Falls Sie abweichende Lösungen erstellt haben, bedeutet das daher nicht, dass diese zwangsläufig verkehrt sind.

Sollten Sie in dieser Musterlösung Fehler finden, die leider auch nicht immer auszuschließen sind, zögern Sie bitte nicht, uns per eMail an die Adresse `buch@vorkurs-informatik.de` zu kontaktieren. Dieses gilt selbstverständlich auch weiterhin bei allen weiteren Fragen zum Buch. Für die vielen interessanten Diskussionen, die auf diesem Wege bisher entstanden sind, möchten wir uns hiermit auch recht herzlich bedanken.

Bei der Erstellung der Musterlösungen zum Buch waren zahlreiche Personen beteiligt. Zu nennen sind hier Knut Krause, Michael Gajda, Martina Vaupel, Azadeh Alebrahim, Franka Bause und Martin Rentz die maßgeblich an dem Zustandekommen der Musterlösungen beteiligt waren, und Heike Rapp, für das Korrekturlesen und Formatieren des Textes. Ihnen allen möchten wir für ihren Beitrag herzlich danken.

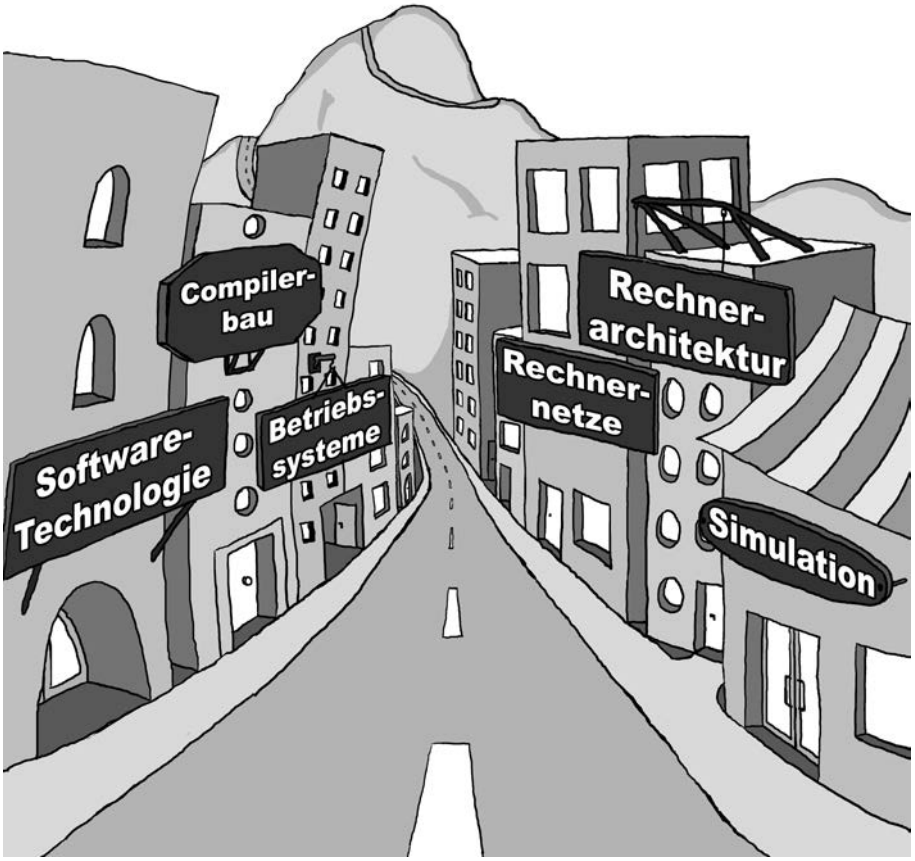
Dortmund, im August 2017

Heinrich Müller
Frank Weichert

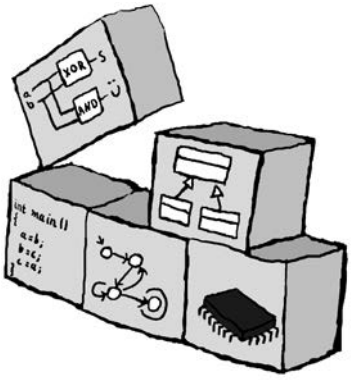
Inhaltsverzeichnis

Einleitung

Zu diesem Kapitel liegen keine Übungsaufgaben vor.



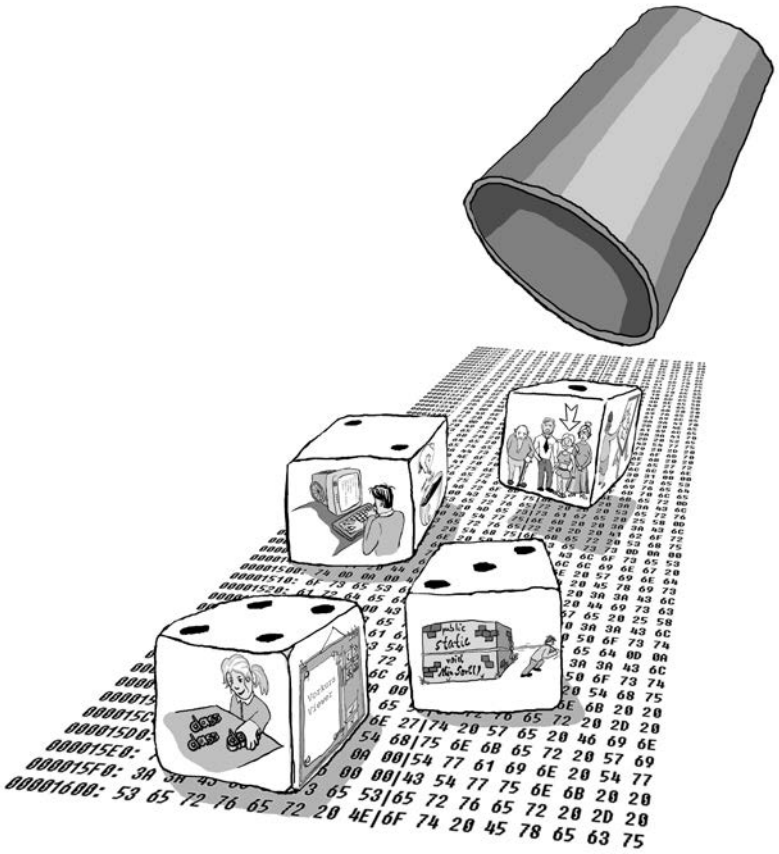
Was ist Informatik?



Kapitel **1**

Informatik

Zu diesem Kapitel liegen keine Übungsaufgaben vor.



Programmierung

Kapitel 2

Vom Problem über den Algorithmus zum Programm

Aufgabe 2.1

- Ein Notenblatt.
- Ein Bauplan.
- ...

Aufgabe 2.2

Problem formulieren In dieser Aufgabe soll festgestellt werden, ob sich in einer Liste von Studierenden eine Person mit dem Alter „19“ befindet.

Problemanalyse, Problemabstraktion, Problemspezifikation Es gibt eine Lösung für dieses Problem. Die Antwort lautet entweder *wahr* oder *falsch*. Die Lösung ist somit auch eindeutig.

Als Alter wird hier die ganze positive Jahreszahl betrachtet. Für dieses Problem ist es notwendig, die bekannte „=“-Relation zu verwenden.

Abschließend ist festzustellen, dass die Personenangaben auch in einer Liste zur Verfügung gestellt werden können. Die Ausgabe des Algorithmus soll lediglich *wahr* oder *falsch* lauten.

Problem Vorhandensein einer Zahl in einer Menge von Zahlen.

Gegeben Eine Folge $a_0, \ldots, a_n$ von positiven Zahlen mit $n > 0$.

Gesucht Der Wahrheitswert der Frage, ob die Schnittmenge zwischen Menge und gesuchter Zahl nicht leer ist.

```java
package task0202;

public class SucheAlter {

	public static void main(String[] args) {
		int[] ages = {20, 18, 19, 21, 20};
		int search = 19;
		boolean found = false;

		for (int i = 0; i < ages.length; i++) {
			if (ages[i] == search) {
				found = true;
				break;
			}
		}
		if (found) {
			System.out.println("Ja.");
		} else {
			System.out.println("Nein.");
		}
	}
}
```

Quellcode 2.1: *SucheAlter.java*

Algorithmenentwurf Die Menge an Zahlen wird Schritt für Schritt durchlaufen. Wenn das gesuchte Alter gefunden wird, erfolgt die Ausgabe `wahr`, sonst `falsch`.

Nachweis der Korrektheit (Semantik, Verifikation) Der Algorithmus endet, wenn das gesuchte Alter gefunden *oder* die Liste von Studenten zu Ende ist.

Aufwandsanalyse Genau wie bei der Suche des Minimums im Buch, wächst bei der Suche nach einem bestimmten Alter die Laufzeit linear mit der Größe der Eingabe.

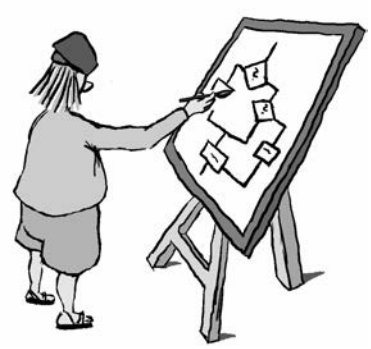
Programmierung Programmiert werden könnte die Suche nach einem Alter wie Quellcode 2.1 angeführt.

Aufgabe 2.3

(a) In dieser Aufgabe wird die Menge der Teilnehmer der Vorlesung verglichen und geprüft, ob jemand den Namen „Müller“ besitzt. Hierbei kann der Name eine beliebige Kombination aus Schriftzeichen sein.

(b) Diese Aufgabe erweitert die vorherige, indem die Menge der Studenten auf Gleichheit zweier Nachnamen geprüft werden sollen. Hierfür muss die Gleichheitsrelation definiert werden.

In diesem Falle bedeutet dieses, dass zwei Nachnamen *exakt* aus denselben Zeichen in derselben Reihenfolge bestehen.



Kapitel 3

Algorithmenentwurf

Aufgabe 3.1

i	a_i	*merker*
	8	8
1	5	5
2	3	3
3	6	3
4	4	3

Tabelle 3.1: *Ablauf des Minimum-Algorithmus*

Aufgabe 3.2

(a) In Zeile 4 von Pseudocode 3.1 müsste das $<$-Zeichen durch ein $>$-Zeichen ersetzt werden.

(b) Die Kurzschreibweise für die Maximumsuche, s. Pseudocode 3.1.

```
1 merker := a[0];
2 i := 1;
3
4 Solange i < n ist, fuehre aus: {
5         Wenn a[i] > merker, dann {
6                 merker := a[i];
7         }
8         i := i + 1;
```

```
 9 }
10 Gib merker zurueck;
```

Pseudocode 3.1: *Die Kurzschreibweise für die Maximumsuche.*

Aufgabe 3.3

Der Pseudocode 3.2 gibt „Ja“ zurück, falls sich eine Zahl *a* in einer Zahlenfolge befindet, sonst „Nein“.

```
1 i := 0;
2
3 Solange i < n ist, fuehre aus: {
4         Wenn folge[i] = a, dann {
5                 Gib "ja" zurueck;
6         }
7         i := i + 1;
8 }
9 Gib "nein" zurueck;
```

Pseudocode 3.2: *Ein Suchalgorithmus*

Aufgabe 3.4

(a) Der Pseudocode 3.3, der den Namen „Müller“ sucht.

```
1  name := "Mueller";
2  i := 0;
3
4  Solange i < n ist, fuehre aus: {
5          Wenn folge[i] = name, dann {
6                  Gib "ja" zurueck;
7          }
8          i := i + 1;
9  }
10 Gib "nein" zurueck;
```

Pseudocode 3.3: *Ein Suchalgorithmus*

(b) Der Pseudocode 3.4, der zwei gleiche Nachnamen sucht.

```
1  i := 0;
2
3  Solange i < n ist, fuehre aus: {
4          j := i + 1;
5          Solange j < n, fuehre aus: {
6                  Wenn folge[i] = folge[j], dann {
7                          Gib "ja" zurueck;
8                  }
9                  j := j + 1;
10         }
11         i := i + 1;
12 }
```

```
13 Gib "nein" zurueck;
```

Pseudocode 3.4: *Ein Suchalgorithmus*

Aufgabe 3.5

(a) Die Wertzuweisung von a und b nach jedem Zuweisungsschritt:

$a := 3;$	$a = 3$
$b := 1;$	$a = 3$
	$b = 1$
$a := a + 2;$	$a = 5$
	$b = 1$
$b := a + 2;$	$a = 5$
	$b = 7$
$a := a + b;$	$a = 12$
	$b = 7$

(b) Grafische Darstellung der Wertzuweisung gemäß Abbildung 3.3 im Buch:

$a := 3;$	$a\boxed{3}$
$b := 1;$	$a\boxed{3}$
	$b\boxed{1}$
$a := a + 2;$	$a\boxed{5}$
	$b\boxed{1}$
$b := a + 2;$	$a\boxed{5}$
	$b\boxed{7}$
$a := a + b;$	$a\boxed{12}$
	$b\boxed{7}$

Aufgabe 3.6

Die Anweisung $a < b$ ist falsch, da $a = 3$ und $b = a - 3 = 0$ ist – somit also $a \not< b$.

Aufgabe 3.7

s. Struktogramm innerhalb der Abbildung 3.1

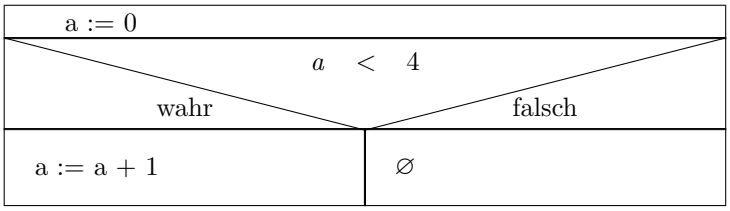


Abbildung 3.1: *Struktogramm zur Aufgabe 3.7*

Aufgabe 3.8

(a) Am Ende sind $a = 2$ und $b = 3$.

(b) s. Struktogramm innerhalb der Abbildung 3.2

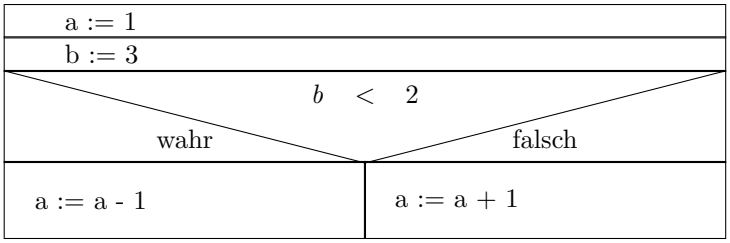


Abbildung 3.2: *Struktogramm zur Aufgabe 3.8b*

Aufgabe 3.9

(a) $a = 1$

(b) s. Struktogramm innerhalb der Abbildung 3.3

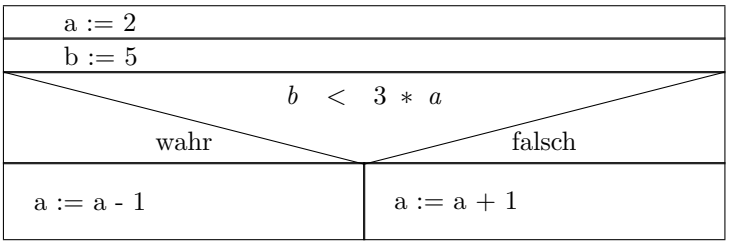


Abbildung 3.3: *Struktogramm zur Aufgabe 3.9b*

Aufgabe 3.10

Nach dem Durchlauf ist $i = 6$.

Aufgabe 3.11

a	b
1	0
2	0
3	1
4	1
5	2

Tabelle 3.2: *Variablenwerte von a und b.*

Aufgabe 3.12

s. Struktogramme innerhalb der Abbildungen 3.4 und 3.5

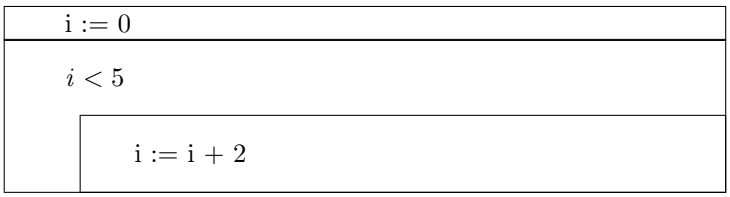


Abbildung 3.4: *Struktogramm zur Aufgabe 3.9b - (zur Aufgabe 3.10)*

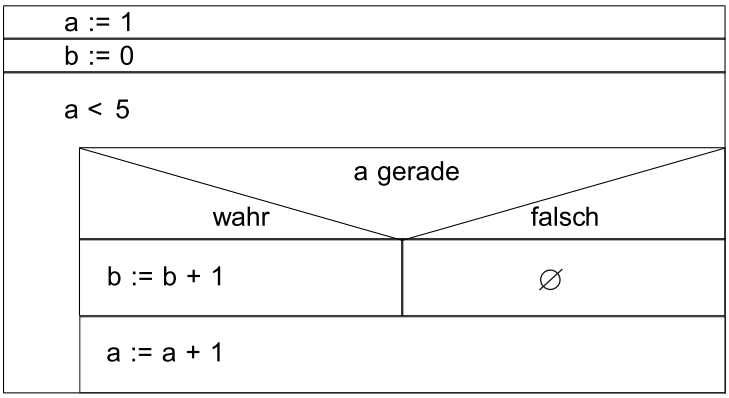


Abbildung 3.5: *Struktogramm zur Aufgabe 3.12 - (zur Aufgabe 3.11)*

Aufgabe 3.13

Pseudocode 3.5, der zwei minimale Eingaben überprüft.

```
1  i := 1;
2  merker = folge[0];
3  counter = 1;
4
5  Solange i < n ist, fuehre aus: {
6          Wenn folge[i] < merker, dann {
7                  merker = folge[i];
8                  counter = 1;
9          } sonst: Wenn folge[i] = merker, dann {
10                 counter = counter + 1;
11         }
12         i := i + 1;
13 }
14
15 Wenn counter >= 2, dann {
16         Gib "ja" zurueck;
17 } sonst {
```

```
18          Gib "nein" zurueck;
19  }
```

Pseudocode 3.5: *Ein Algorithmus, der zwei minimale Eingaben überprüft.*

Aufgabe 3.14

(a) Pseudocode 3.6, der in einer Folge nach einer doppelten Eingabe sucht.

```
1  a := 0;
2  doppelt := falsch;
3
4  Solange a < laengeDerFolge, fuehre aus {
5          b := a + 1;
6
7          Solange b < laengeDerFolge, fuehre aus {
8                  wenn Folge[a] = Folge[b], dann doppelt := wahr;
9          b := b + 1;
10         }
11     a := a + 1;
12 }
13 wenn doppelt = wahr,
14     dann Gib "ja" zurueck;
15     sonst Gib  "nein" zurueck;
```

Pseudocode 3.6: *Ein Algorithmus, der zwei gleiche Eingaben sucht.*

(b) s. Struktogramm innerhalb der Abbildung 3.6

Aufgabe 3.15

s. Pseudocode 3.7

```
1 a := n;
2 b := 0;
3
4 Solange a > 0, fuehre aus {
5         b := b + a;
6         a := a - 1;
7 }
8 Gib b zurueck;
```

Pseudocode 3.7: *Ein Algorithmus, der die Summe von 1–n berechnet.*

Zu beachten ist, dass selbst eine mathematische Gleichung ein Algorithmus ist. Der Algorithmus könnte demzufolge auch ganz einfach folgende Formel sein:

$$\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2} \tag{3.1}$$

Hinweis: Dies ist die *Gaußsche Summenformel*, die auch später noch Anwendung findet und daher gemerkt werden sollte.

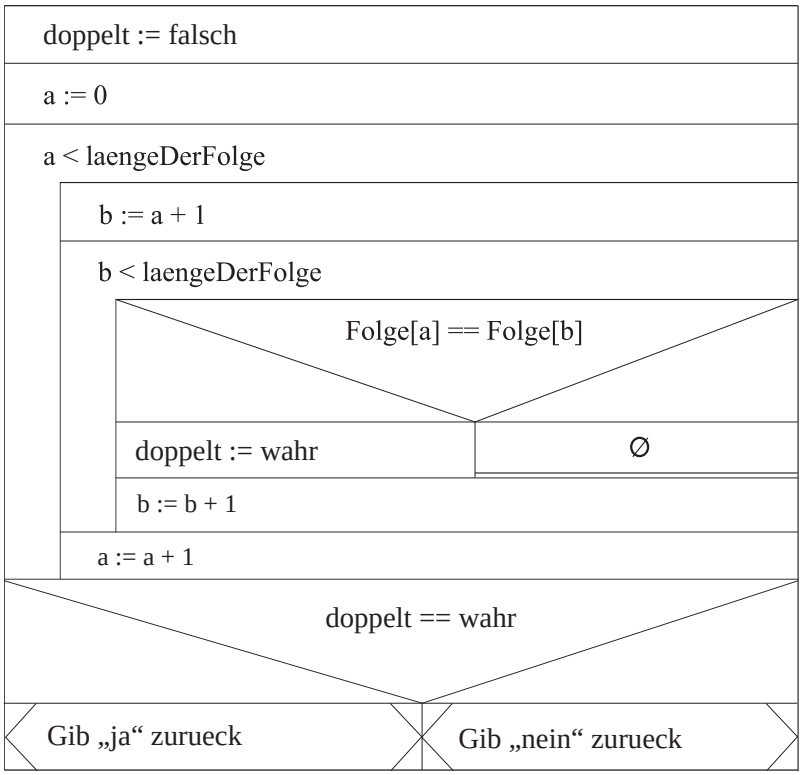


Abbildung 3.6: *Struktogramm zur Aufgabe 3.14 b)*

Kapitel **4**

Grundkonzepte der Programmierung

Aufgabe 4.1

Als Musterlösungsvorschlag wurde das Ganze mit aptitude unter Linux installiert. Auf nicht Debian-basierten Systemen könnte natürlich auch ein entsprechendes Werkzeug wie z. B. `emerge` oder `YAST` verwendet werden.

Aufgabe 4.2

(a)
- Mein Name ist Hans.
- Mein Vorname ist 42.
- Mein Name ist Kuchen.

(b) Das Syntaxdiagramm mit dem man Vor- und Nachname ausgeben kann, ergibt sich wie folgt:

$$\boxed{\text{Mein}} \rightarrow \boxed{\text{Name}} \rightarrow \boxed{\text{ist}} \rightarrow \boxed{\textit{Wort}} \rightarrow \boxed{\textit{Wort}} \rightarrow \boxed{.} \rightarrow$$

Aufgabe 4.3

Mit einer Unix-Shell geht es vergleichbar.

(a) Keine Lösung erforderlich.

(b) Keine Lösung erforderlich.

```java
package task0405;

public class ProgrammMaxSuche {

	public static void main(String[] args) {
		int[] a = {11, 7, 8, 3, 15, 13, 9, 19, 18, 10, 4};
		int merker = a[0];
		int i = 1;
		int n = a.length;

		while (i < n) {
			if (a[i] > merker) {
				merker = a[i];
			}
			i++;
		}
		System.out.println(merker);
	}
}
```

Quellcode 4.1: *ProgrammMaxSuche.java*

Aufgabe 4.4

Keine Lösung erforderlich.

Aufgabe 4.5

s. Quellcode 4.1 zum ProgrammMaxSuche

Aufgabe 4.6

- `int i=5;` Datentyp → Name →=→ Ausdruck → ;
- `int a,i=7;` Datentyp → Name → , → Name →=→ Ausdruck → ;

`int a,;` ist nicht zulässig, da nach einem Komma immer mindestens ein weiterer Variablenname folgen muss.

Aufgabe 4.7

- 6.5 kann Double oder Float sein.
- −5 ist ein Integer.

- $8L$ ist vom Typ Long.
- L
- l
- `true` ist vom Typ Boolean.
- 'L' ist ein Char.

Aufgabe 4.8

(1) zulässig

(2) nicht zulässig

(3) nicht zulässig

(4) zulässig

(5) zulässig

(6) nicht zulässig

Aufgabe 4.9

(a) (1) zulässig
(2) nicht zulässig
(3) nicht zulässig
(4) zulässig
(5) nicht zulässig

(b) (1) -2.1
(2) 4.5
(3) 5
(4) -2.1
(5) 0.7

Aufgabe 4.10

(1) 59

(2) 16

(3) 0

(4) 32

(5) 4

Aufgabe 4.11

(1) `x * x * x`

(2) `2 * a + 7 * b`

(3) `(2 * a + 3) * (2 * a + 3)`

(4) `(x * x - 4) / (-3)`

Aufgabe 4.12

(1) zulässig, `true`

(2) zulässig, `true`

(3) nicht zulässig

(4) zulässig, `true`

Aufgabe 4.13

(1) `false`

(2) `true`

(3) `false`

(4) `true`

Aufgabe 4.14

(1) Verknüpfungstabelle für `x && !y`:

x	y	!y	x && !y
false	false	true	false
false	true	false	false
true	false	true	true
true	true	false	false

(2) Verknüpfungstabelle für `x || (x && y)`:

x	y	x && y	x ‖ (x && y)
false	false	false	false
false	true	false	false
true	false	false	true
true	true	true	true

(3) Verknüpfungstabelle für `!(!x || y) && (!x && !y)`:

x	y	!(!x ‖ y)	(!x && !y)	!(!x ‖ y) && (!x && !y)
false	false	false	true	false
false	true	false	false	false
true	false	true	false	false
true	true	false	false	false

(4) Verknüpfungstabelle für `x && !(y || z)`:

x	y	z	!(y ‖ z)	x && !(y ‖ z)
false	true	true	false	false
false	true	false	false	false
false	false	true	false	false
false	false	false	true	false
true	true	true	false	false
true	true	false	false	false
true	false	true	false	false
true	false	false	true	true

Aufgabe 4.15

(a) s. Struktogramm innerhalb der Abbildung 4.1

(b) Folgende Werte von *wert* ergeben sich, sofern für *a* die Zahlen 1 bis 5 eingesetzt werden:

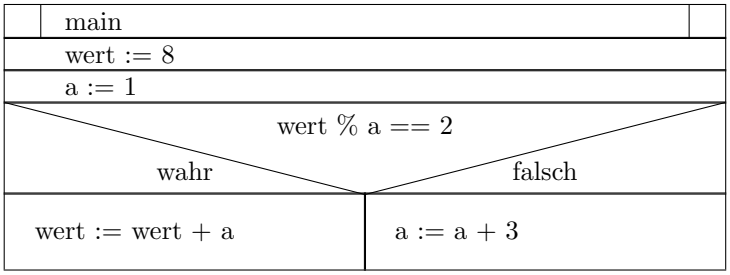


Abbildung 4.1: *Struktogramm zur Aufgabe 4.15a*

- $a = 1 \Rightarrow wert = 8$
- $a = 2 \Rightarrow wert = 8$
- $a = 3 \Rightarrow wert = 11$
- $a = 4 \Rightarrow wert = 8$
- $a = 5 \Rightarrow wert = 8$

Aufgabe 4.16

Quellcode 4.2 mit der `while`-Schleife, welche die ersten 10 Zahlen aufaddiert, könnte wie nachfolgend angeführt aufgebaut sein. In der Musterlösung werden die gebräuchlichen Kurzschreibweisen für bestimmte Befehle verwendet. `sum+= i;` z. B. bedeutet `sum = sum + i;`. Andererseits bedeutet `i++;` lediglich `i = i + 1;`. Ergänzend existiert die Notation `++i;`, auf die an dieser Stelle nicht weiter eingegangen werden soll.

Aufgabe 4.17

Lösung siehe Quellcode 4.3.

Aufgabe 4.18

(a) s. Struktogramm innerhalb der Abbildung 4.2

(b) Die Werte von `zaehler` und `summe` innerhalb der Tabelle 4.1.

(c) Ersetzen Sie die Anweisung `summe = summe - zaehler;` durch `summe = summe + zaehler;` und ergänzen Sie die Anweisung `double mittelwert = summe / (zaehler-);` hinter der `while`-Schleife.

```java
package task0416;

public class SumUp {

	public static void main(String[] args) {
		int sum = 0;
		int i = 1;

		while (i <= 10) {
			sum += i;
			i++;
		}
		// Zusatz fuer Aufgabe 4.19 a)
		System.out.println(sum);
	}
}
```

Quellcode 4.2: *Lösung zur Aufgabe 4.16: SumUp.java*

```java
class JavaProgramm{
	public static void main(String[] argv){
		int wert = 8;
		int counter = 1;
		while(counter < 6) {
		int a = counter;
		if (wert % a == 2){
			wert = wert + a;
			} else {
				a = a + 3;
				}
		counter = counter + 1;
		}
	}
}
```

Quellcode 4.3: *Lösung zur Aufgabe 4.17*

zaehler	summe
2	-1
3	-3
4	-6

Tabelle 4.1: *Wertetabelle für Aufgabe 4.17 b)*

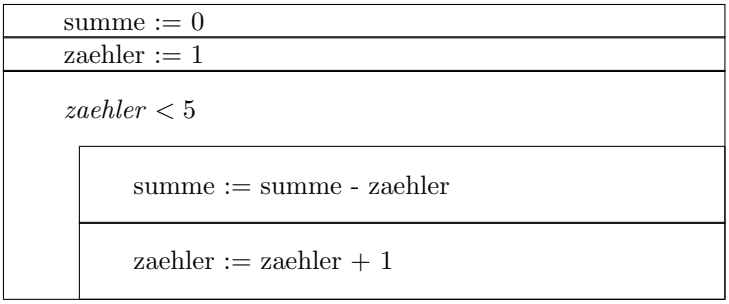


Abbildung 4.2: *Struktogramm zur Aufgabe 4.17a*

Aufgabe 4.19

Lösung siehe Quellcode 4.4.

Aufgabe 4.20

(a) Siehe Quellcode 4.2, Quellcode 4.5 und Quellcode 4.4 für die erweiterten Sourcecodes. Es ist jeweils ein Vermerk als Kommentar im Code angebracht.

(b) Keine Lösung notwendig.

Aufgabe 4.21

Quellcode 4.6 vermittelt die erste und intuitive Lösung des Programms. Das Problem ist, dass Hallo du da ausgeben werden soll, das Programm aber nur Hallo du annimmt. Weiterhin produziert das Programm Fehler für weniger als zwei Parameter.
Da für den nachfolgenden Code Kenntnisse nötig sind, die nicht durch das Buch vermittelt werden, wurden an dieser Stelle zwei Lösungen aufgezeigt, die einen Anhaltspunkt für spätere Lösungswege geben sollen.
Eine recht gute Lösung zeigt beispielsweise der Quellcode 4.7.

Aufgabe 4.22

Zur Lösung s. Quellcode 4.8.

Aufgabe 4.23

(a) Bei dieser Aufgabe wird die gleiche Lösung wie in Quellcode 4.9 genommen. Dabei wird die Schleife durch einen statischen Teil ersetzt.

```java
package task0418;

public class JavaProgramm {

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		int i = 1;

		while(i <= 5) {
			int a = i;
			int wert = 8;

			if (wert % a == 2) {
				wert += a;
			} else {
				a += 3;
			}
			/*
			 * Zusatz fuer Aufgabe 4.19 a)
			 */
			System.out.println("Fuer a = " + i + " erhaelt man:
			    a = " + a + " und wert = " + wert);
			i++;
		}
	}
}
```

Quellcode 4.4: *Lösung zur Aufgabe 4.18: JavaProgramm.java*

(b) Der Quellcode 4.9 berechnet die Quadrat- und Kubikzahlen.

(c) Ein Test von 0 bis 1000 lief anstandslos. Demzufolge funktioniert dieses Programm auch mit der Zahl 0.

Aufgabe 4.24

(a) Zu dieser Übungsaufgabe liegt keine Lösung vor.

(b) Der Quellcode wird hier aus Quellcode 4.10 genommen und dabei wird der Schleifenanfang statisch genutzt.

(c) Bei diesem Programm lassen sich sowohl Anfangs- als auch Endpunkt des Intervalls wählen.

```java
package task0417;

public class JavaProgramm {

	public static void main(String[] args) {
		int summe = 0;
		int zaehler = 1;

		while (zaehler < 4) {
			summe -= zaehler;
			zaehler++;

			// Zusatz fuer Aufgabe 4.19 a)
			System.out.println("zaehler = " + zaehler + ", summe
				= " + summe);
		}
	}
}
```

Quellcode 4.5: *Lösung zur Aufgabe 4.19: JavaProgramm.java*

```java
package task0420;

public class Eingabe {


	public static void main(String[] args) {
		System.out.println(args[0] + " " + args[1]);
	}
}
```

Quellcode 4.6: *Lösung zur Aufgabe 4.20: Eingabe.java*

Aufgabe 4.25

(a) Ein statisches Programm mit der Ausgabe Hallo, mein Name ist ... gibt den Text direkt gefolgt von `args[0]` aus. Quellcode 4.10 zeigt jedoch ein dynamisches Programm.

(b) Ein mögliches dynamische Programm, mit dem beliebig lange Namen ausgeben werden können:

(c) Das erweiterte Programm ProgrammFlaeche wäre:

Aufgabe 4.26

(a) Zur Musterlösung s. Quellcode 4.13.

```java
package task0420;

public class Eingabe2 {


	public static void main(String[] args) {
		String out = "";

		for (String s : args) {
			out += s + " ";
		}
		System.out.println(out);
	}
}
```

Quellcode 4.7: *Alternative Lösung zur Aufgabe 4.20: Eingabe2.java*

```java
package task0421;

public class ProgrammFlaeche {


	public static void main(String[] args) {
		double laenge = 4;
		double breite = 3;
		double flaeche = laenge * breite;

		System.out.println("Die Flaeche des Rechtecks betraegt: " +
		    flaeche);
	}
}
```

Quellcode 4.8: *ProgrammFlaeche.java*

(b) Zu dieser Übungsaufgabe ist keine Musterlösung notwendig.

(c) Zu dieser Übungsaufgabe liegt keine Lösung vor.

(d) Zu dieser Übungsaufgabe liegt keine Lösung vor.

```java
package task0422;

public class Table {

	public static void main(String[] args) {
		int from = 1;
		int to = 3;

		/*
		 * Mit \t kann man Tabs in seiner Ausgabe erreichen.
		 */
		System.out.println("n\tn^2\tn^3");
		System.out.println("++++++++++++++++++++");
		for (int i = from; i <= to; i++) {
			System.out.println(i + "\t" + i*i + "\t" + i*i*i);
		}
	}
}
```

Quellcode 4.9: *Table.java*

```java
package task0423;

public class Numbers {

	public static void main(String[] args) {
		int from = 1;
		int to = 100;

		for (int i = from; i <= to; i++) {
			String out = i + "";

			if (i % 3 == 0) {
				out += "\t*";
			}
			System.out.println(out);
		}
	}
}
```

Quellcode 4.10: *Numbers.java*

```java
package task0424;

public class EinProgramm {

	public static void main(String[] args) {
		String name = "";

		for (int i = 0; i < args.length; i++) {
			name += args[i] + " ";
		}

		System.out.println("Hallo, mein Name ist " + name);
	}
}
```

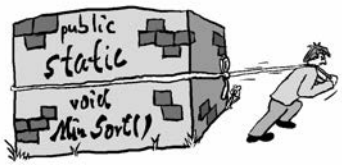
Quellcode 4.11: *EinProgramm.java*

```java
package task0424;

public class ProgrammFlaeche {

	public static void main(String[] args) {
		double laenge = Double.parseDouble(args[0]);
		double breite = Double.parseDouble(args[1]);
		double flaeche = laenge * breite;

		System.out.println("Die Flaeche des Rechtecks betraegt: " +
		    flaeche);
	}
}
```

Quellcode 4.12: *ProgrammFlaeche.java*

```java
package task0425;

public class Plot {

	public static void main(String[] args) {
		double xAnfang = 2;
		double xEnde = 5;
		double step = 0.1;
		double x;
		double y;

		for (x = xAnfang; x < xEnde; x += step) {
			y = 4 * x * x + 5 * x - 3;

			System.out.println("Fuer x = " + x + " ergibt sich:
			    y = " + y);
		}
	}
}
```

Quellcode 4.13: *Plot.java*



Kapitel 5

Funktionen

Aufgabe 5.1

Das „Sortieren durch Minimumsuche“ für die Zahlenfolge 19, 18, 15, 13, 6, 8.

Durchlauf	*restfolge*	a_k	*ergebnisfolge*
1	19, 18, 15, 13, 6, 8	6	6
2	19, 18, 15, 13, 8	8	6, 8
3	19, 18, 15, 13	13	6, 8, 13
4	19, 18, 15	15	6, 8, 13, 15
5	19, 18	18	6, 8, 13, 15, 18
6	19	19	6, 8, 13, 15, 18, 19

Tabelle 5.1: *Sortieren durch Minimumsuche mit Hilfsfolge*

Aufgabe 5.2

(a) Der Pseudocode 5.1 für die Maximumsuche sucht bei jedem Durchlauf das jeweils größte Element a_k.

```
1 Weise restfolge die gegebene Folge zu;
2
3 Solange restfolge nicht leer ist, fuehre aus:
4         suche ein Element a[k] mit dem groesstem Wert in restfolge;
5         fuege a[k] an ergebnisfolge an;
6         entferne a[k] aus restfolge;
7 Gib ergebnisfolge zurueck;
```

Pseudocode 5.1: *Die Maximumsuche als Pseudocode*

(b) Nachfolgend wird die Tabelle für das Sortieren nach Maximumsuche dargestellt. Hier werden die Elemente nun absteigend sortiert.

Durchlauf	`restfolge`	a_k	`ergebnisfolge`
1	19, 18, 15, 13, 6, 8	19	19
2	18, 15, 13, 6, 8	18	19, 18
3	15, 13, 6, 8	15	19, 18, 15
4	13, 6, 8	13	19, 18, 15, 13
5	6, 8	8	19, 18, 15, 13, 8
6	6	6	19, 18, 15, 13, 8, 6

Tabelle 5.2: *Sortieren durch Maximumsuche mit Hilfsfolge*

Aufgabe 5.3

Das „Sortieren durch Minimumsuche ohne Hilfsfolge“ .

Durchlauf	`folge`	a_k
1	19, 18, 15, 13, 6, 8	6
2	6, 18, 15, 13, 19, 8	8
3	6, 8, 15, 13, 19, 18	13
4	6, 8, 13, 15, 19, 18	15
5	6, 8, 13, 15, 19, 18	18
6	6, 8, 13, 15, 18, 19	19
7	6, 8, 13, 15, 18, 19	

Tabelle 5.3: *Sortieren durch Minimumsuche ohne Hilfsfolge*

Aufgabe 5.4

Das „Sortieren durch Maximumsuche ohne Hilfsfolge“.

Durchlauf	folge	a_k
1	19, 18, 15, 13, 6, 8	19
2	19, 18, 15, 13, 6, 8	18
3	19, 18, 15, 13, 6, 8	15
4	19, 18, 15, 13, 6, 8	13
5	19, 18, 15, 13, 6, 8	8
6	19, 18, 15, 13, 8, 6	6
7	19, 18, 15, 13, 8, 6	

Tabelle 5.4: *Sortieren durch Maximumsuche ohne Hilfsfolge*

Aufgabe 5.5

(a) Der Algorithmus enthält folgende Schritte:

- Suche das Element der Folge mit dem größten Wert. Vertausche das letzte Element der Folge mit diesem Element.
- Suche in der Restfolge bis zum vorletzten Element ein Element mit größtem Wert. Vertausche das vorletzte Element der Folge mit diesem Element.
- Führe dieses Verfahren mit der Restfolge bis zum drittletzten Element, viertletzten Element und so weiter bis zum zweiten Element aus.

(b) s. Struktogramm innerhalb der Abbildung 5.1

(c) Auf die Zahlenfolge $19, 18, 15, 13, 6, 8$ angewendet, ergibt sich folgende Darstellung:

Durchlauf	folge	a_k
1	19,18,15,13,6,8	19
2	8,18,15,13,6,19	18
3	8,6,15,13,18,19	15
4	8,6,13,15,18,19	13
5	8,6,13,15,18,19	8
6	6,8,13,15,18,19	6
7	6,8,13,15,18,19	

Tabelle 5.5: *Darstellung des Algorithmus*

Aufgabe 5.6

Zu dieser Übungsaufgabe liegt keine Lösung vor.

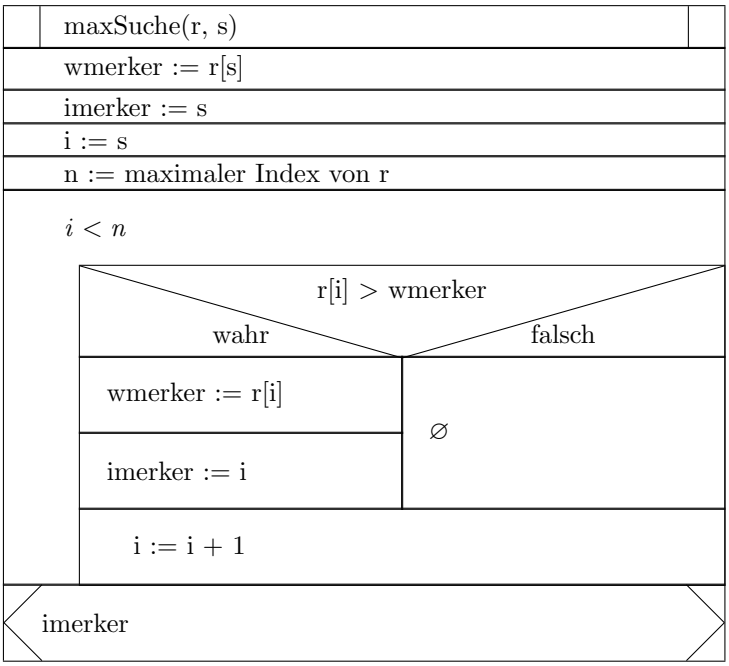


Abbildung 5.1: *Struktogramm zur Aufgabe 5.5b*

Aufgabe 5.7

(a) Der Quellcode 5.2 ist entsprechend um Funktionen erweitert.

(b) Die Funktion `static double qFunktion(double x)` liegt innerhalb des Quellcodes 5.3 vor.

Aufgabe 5.8

Der Algorithmus „Sortieren mit Maximumsuche ohne Hilfsfolge“– s. Quellcode 5.4.

Aufgabe 5.9

Die folgende Tabelle zeigt die Gültigkeitsbereiche. Ein „j“ zeigt dabei an, dass die Variable im betrachteten Block sichtbar ist, ein „n“ dagegen, dass diese nicht sichtbar ist.

Aufgabe 5.10

(a) Siehe Funktion `minSort(int[] values, int[] solution)` innerhalb des Quellcodes 5.5.

```java
package task0507;

public class Table {


	public static void main(String[] args) {
		int from = 1;
		int to = 3;

		// Mit \t kann man Tabs in seiner Ausgabe erreichen.

		System.out.println("n\tn^2\tn^3");
		System.out.println("++++++++++++++++++++");
		for (int i = from; i <= to; i++) {
			System.out.println(i + "\t" + hochZwei(i) + "\t" +
			    hochDrei(i));
		}
	}

	static int hochZwei(int n) {
		return n * n;
	}

	static int hochDrei(int n) {
		return hochZwei(n) * n;
	}
}
```

Quellcode 5.2: *Table.java*

	r	*s*	*wmerker*	*imerker*	*i*	*n*	*a*	*k*	*merker*
minSuche2	j	j	j	j	j	j	n	n	n
while-Schleife in minSuche2	j	j	j	j	j	j	n	n	n
main-Methode	n	n	n	n	j	j	j	j	n
while-Schleife 1	n	n	n	n	j	j	j	j	j
while-Schleife 2	n	n	n	n	j	j	j	j	n

Tabelle 5.6: *Die Gültigkeitsbereiche der Variablen aus Quellcode 5.3*

(b) Siehe Funktion main(String[] args) innerhalb des Quellcodes 5.5.

```java
package task0507;

public class Plot {


	public static void main(String[] args) {
		//...

	}

	public static double qFunktion(double x) {
		return 4 * x * x + 5 * x - 3;
	}
}
```

Quellcode 5.3: *Plot.java*

```java
package task0508;

public class MaxSort {

	/**
	 * Gibt den Index eines Elements von r mit dem groesstem Wert ab
	     Index s.
	 * @param r Die verschiedenen Elemente.
	 * @param s Der Index ab dem betrachtet wird.
	 * @return Der Index des groesstem Wertes von s--r.length;
	 */
	static int maxSort(int[] r, int s) {
		int index = s;

		for (int i = s + 1; i < r.length; i++) {
			if (r[i] > r[index]) {
				index = i;
			}
		}
		return index;
	}


	public static void main(String[] args) {
		int[] a = {11, 7, 8, 3, 15, 13, 9, 19, 18, 10, 4};

		for (int i = 0; i < a.length; i++) {
			int j = maxSort(a, i);
			int merker = a[i];
			a[i] = a[j];
			a[j] = merker;
		}

		for (int i = 0; i < a.length; i++) {
			System.out.println(a[i]);
		}
	}
}
```

Quellcode 5.4: *MaxSort.java*

```java
package task0510;

public class MinSort {

	//Gibt den Index des niedrigsten Wertes der Folge zurueck.

	static int getMinIndex(int[] values, int endIndex) {
		int minIndex = 0;

		for (int i = 1; i <= endIndex; i++) {
			if (values[i] < values[minIndex]) {
				minIndex = i;
			}
		}
		int tmp = values[endIndex];

		values[endIndex] = values[minIndex];
		values[minIndex] = tmp;

		return endIndex;
	}

	/**
	 * Sortiert ein Array mit Hilfe eines Hilfsarrays.
	 * @param values Das Eingabearray.
	 * @param solution Das Hilfsarray.
	 */
	static void minSort(int[] values, int[] solution) {
		for (int i = values.length - 1; i >= 0; i--) {
			int minIndex = getMinIndex(values, i);

			solution[i] = values[minIndex];

		}
	}



	public static void main(String[] args) {
		int[] a = {7, 4, 434, 65, 57, 96, 12, 45, 9, 2, 2265};
		int[] b = new int[a.length];

		minSort(a, b);
		for (int i = 0; i < b.length; i++) {
			System.out.println(b[i]);
		}
	}
}
```

Quellcode 5.5: *MinSort.java*

Kapitel 6

Rekursion

Aufgabe 6.1

Durchlauf	*folge*1	*folge*2	*folge*
1	15, 18, 19	6, 8, 13	6
2	15, 18, 19	8, 13	6, 8
3	15, 18, 19	13	6, 8, 13
4	15, 18, 19		6, 8, 13, 15, 18, 19

Tabelle 6.1: *Ablauf des Mischvorgangs für zwei Folgen.*

Aufgabe 6.2

i	$i1$	$i2$	$f[0]$	$f[1]$	$f[2]$	$f[3]$	$f[4]$	$f[5]$
0	0	0	undef.	undef.	undef.	undef.	undef.	undef.
1	0	1	6	undef.	undef.	undef.	undef.	undef.
2	0	2	6	8	undef.	undef.	undef.	undef.
3	0	3	6	8	13	undef.	undef.	undef.

Tabelle 6.2: *Die Variablenwerte vor jedem Durchlauf der while-Schleife.*

Aufgabe 6.3

Die Folge der übergebenen Werte:

Aufruf	Parameterwert
1	11,7,8,3,15,13,9,19,18,10,4
2	11,7,8,3,15
3	11,7
4	11
5	7
6	8,3,15
7	8
8	3,15
9	3
10	15
11	13,9,19,18,10,4
12	13,9,19
13	13
14	9,19
15	9
16	19
17	18,10,4
18	18
19	10,4
20	10
21	4

Tabelle 6.3: *Übergabewerte*

Aufgabe 6.4

(a) Die iterative Lösung für die Summe von 1–*n* vermittelt Quellcode 6.1.

(b) Die rekursive Lösung für die Summe von 1–*n* zuerst als Pseudocode 6.2.

```
1 summe(n)
2
3 Wenn n groesser als 1 ist, dann:
4         Gib n + summe(n - 1) zurueck.
5 Sonst:
6         Gib n zurueck;
```

Pseudocode 6.2: *Eine Funktion, welche die Summe von 1–n rekursiv berechnet.*

```java
package task0604;

public class SummeIterativ {

	/**
	 * Berechnet die Summe von 1 bis n iterativ.
	 * @param n Die groesste Zahl.
	 * @return Die Summe von 1 + 2 + ... + n.
	 */
	static int summe(int n) {
		int sum = 0;

		for (int i = 1; i <= n; i++) {
			sum += i;
		}
		return sum;
	}


	public static void main(String[] args) {
		System.out.println(summe(10));
	}

}
```

Quellcode 6.1: *SummeIterativ.java*

Anschließend wird diese Funktion als Java-Funktion betrachtet – s. Quellcode 6.3.

Aufgabe 6.5

(a) Die rekursive Java-Funktion `static void mischen(...)` vermittelt Quellcode 6.4.

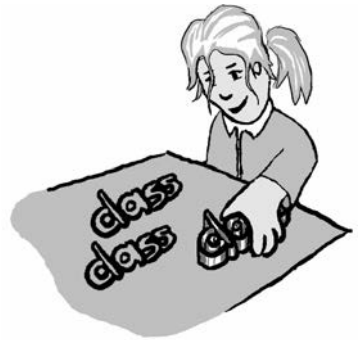
(b) Das Hauptprogramm steht mit der Funktion im Programm `MergeSort` in Quellcode 6.4.

```java
package task0604;

public class SummeRekursiv {

	/**
	 * Berechnet die Summe von 1 bis n rekursiv.
	 * @param n Die groesste Zahl.
	 * @return Die Summe von 1 + 2 + ... + n.
	 */
	static int summe(int n) {
		if (n > 0) {
			return n + summe(n - 1);
		} else {
			return n;
		}
	}


	public static void main(String[] args) {
		System.out.println(summe(10));
	}
}
```

Quellcode 6.3: *SummeRekursiv.java*

```java
package task0605;

public class MergeSort {

	static void mischen(int[] folge1, int i1, int[] folge2, int i2, int
	    [] ergebnisfolge, int i3) {
		if (i1 > folge1.length - 1) {
			if (i2 > folge2.length - 1) {
				return;
			} else {
				ergebnisfolge[i3] = folge2[i2];
				mischen(folge1, i1, folge2, i2 + 1,
				    ergebnisfolge, i3 + 1);
			}
		} else if (i2 > folge2.length - 1) {
			if (i1 > folge1.length - 1) {
				return;
			} else {
				ergebnisfolge[i3] = folge1[i1];
				mischen(folge1, i1 + 1, folge2, i2,
				    ergebnisfolge, i3 + 1);
			}
		} else {
			if (folge1[i1] <= folge2[i2]) {
				ergebnisfolge[i3] = folge1[i1];
				mischen(folge1, i1 + 1, folge2, i2,
				    ergebnisfolge, i3 + 1);
			} else {
				ergebnisfolge[i3] = folge2[i2];
				mischen(folge1, i1, folge2, i2 + 1,
				    ergebnisfolge, i3 + 1);
			}
		}
	}

	public static void main(String[] args) {
		int[] testfolge1 = {3, 7, 8, 11, 15};
		int[] testfolge2 = {4, 9, 10, 13, 18, 19};
		int[] testErgebnis = new int[testfolge1.length + testfolge2.
		    length];

		mischen(testfolge1, 0, testfolge2, 0, testErgebnis, 0);
		for (int i = 0; i < testErgebnis.length; i++) {
			System.out.println(testErgebnis[i]);
		}
	}
}
```

Quellcode 6.4: *MergeSort.java*



Kapitel 7

Klassen und Objekte

Aufgabe 7.1

Eine Funktion, die alle Studierenden, die älter als ein gegebenes Alter *j* sind, ausgibt.

```
1 suche(d0, d1, ..., dn, j)
2
3 i := 0;
4 Solange i <= n fuehre aus:
5         Wenn di.gibAlter() >= j, dann
6                 Gib di zurueck;
```

Pseudocode 7.1: *Die Funktion* suche$(d_0, d_1, \ldots, d_n, j)$

Aufgabe 7.2

Nach „`int geburtsjahr;` “ ist ein „`String[] pruefungen;` “ einzufügen.

Aufgabe 7.3

Die Klasse Ampel – s. Quellcode 7.2.

Aufgabe 7.4

(a) Siehe Klassendeklaration Stuhl innerhalb des Quellcodes 7.3.

(b) Siehe Methode hatArmlehnen() innerhalb des Quellcodes 7.3.

```java
package task0703;

public class Ampel {

	boolean roteLampe;
	boolean grueneLampe;
}
```

Quellcode 7.2: *Ampel.java*

```java
package task0704;

public class Stuhl {

	String farbeSitzflaeche;
	double hoehe;
	boolean armlehnen;

	public boolean hatArmlehnen() {
		return this.armlehnen;
	}
}
```

Quellcode 7.3: *Stuhl.java*

Aufgabe 7.5

(a) Ein Schrank hat z. B. folgende Eigenschaften:

- Höhe,
- Breite und
- Tiefe.
- Material aus dem er gebaut ist.
- Hat er Türen?
- Sind diese offen oder geschlossen?
- Kann man die Türen abschließen?
- Hat er eine Beleuchtung?
- Ist diese an oder aus?
- Hat er einen Spiegel?
- ...

Diese Liste kann beliebig weitergeführt werden. Zur Demonstration werden lediglich diese Eigenschaften genutzt.

```java
package task0705;

public class Schrank {

	int breite;
	int hoehe;
	int tiefe;

	String material;

	boolean tueren;
	boolean offen;
	boolean schloss;
	boolean beleuchtung;
	boolean beleuchtungAn;
	boolean spiegel;

	public Schrank() {

	}

	public boolean hatTueren() {
		return this.tueren;
	}

	public boolean istOffen() {
		return this.offen;
	}

	// Und so weiter...

}
```

Quellcode 7.4: *Schrank.java*

(b) Die aus Aufgabenteil a bekannten Eigenschaften Höhe, Breite, Tiefe und Material lassen sich als Attribute deklarieren. Für die anderen Eigenschaften werden zusätzlich Methoden entworfen. Folgende Klasse Schrank wäre möglich – s. Quellcode 7.4.

Aufgabe 7.6

Die Klasse Studierende um einen zweiten Konstruktor erweitert – s. Quellcode 7.5.

Aufgabe 7.7

(a) Siehe den Konstruktor innerhalb des Quellcodes 7.6.

(b) Siehe Methoden schalteRot, schalteGruen und druckeZustand innerhalb des Quellcodes 7.6.

Aufgabe 7.8

Das Programm Programm – s. Quellcode 7.7.

Aufgabe 7.9

(a) s. Quellcode 7.8

(b) s. Quellcode 7.8

(c) s. Quellcode 7.8

Aufgabe 7.10

(a) s. Quellcode 7.9

(b) s. Quellcode 7.9

Aufgabe 7.11

(a) s. Quellcode 7.10

(b) s. Quellcode 7.10

(c) s. Quellcode 7.10

Aufgabe 7.12

Die Klasse Suchprogramm – s. Quellcode 7.11.

Aufgabe 7.13

Das Testprogramm Programm der Klasse Vektor – s. Quellcode 7.12.

Aufgabe 7.14

Die Referenztabelle:

Aufruf	`eineListe`	`einElement`
1	Element mit Referenz auf Thorsten Meier	Objekt von Hans Otto
2	Element mit Referenz auf Monika Schmidt	Objekt von Hans Otto
3	Element mit Referenz auf Monika Schneider	Objekt von Hans Otto
4	`null`	Objekt von Hans Otto

Tabelle 7.1: *Referenztabelle*

Aufgabe 7.15

Quellcode 7.4 im Buch enthält bereits die Methode `Studierende gibWert()`.

Aufgabe 7.16

s. Quellcode 7.13

```java
package task0706;

public class Studierende {

	// --- Attribute ---
	String studname;
	int matrikelnummer;
	int geburtsjahr;
	String[] pruefungen;

	// --- Konstruktor(en) ---
	public Studierende(String name, int nummer, int jahr) {
		studname = name;
		matrikelnummer = nummer;
		geburtsjahr = jahr;
	}

	public Studierende(int jahr) {
		if (jahr > 1950) {
			this.geburtsjahr = jahr;
		} else {
			this.geburtsjahr = 0;
		}
	}

	// --- Methoden ---
	public String gibStudname() {
		return studname;
	}

	void setzeStudname(String name) {
		studname = name;
	}

	public int gibMatrikelnummer() {
		return matrikelnummer;
	}

	void setzeMatrikelnummer(int nummer) {
		matrikelnummer = nummer;
	}

	int gibGeburtsjahr() {
		return geburtsjahr;
	}

	void setzeGeburtsjahr(int jahr) {
		geburtsjahr = jahr;
	}

	int gibAlter() {
		int aktJahr = Datum.gibJahreszahl();
		return aktJahr - geburtsjahr;
	}
}
```

Quellcode 7.5: *Studierende.java*

```java
package task0707;

public class Ampel {

	boolean roteLampe;
	boolean grueneLampe;

	public Ampel() {
		this.roteLampe = false;
		this.grueneLampe = false;
	}

	public void schalteRot(boolean b) {
		this.roteLampe = b;
	}

	public void schalteGruen(boolean b) {
		this.grueneLampe = b;
	}

	public void druckeZustand() {
		String statusRot;
		String statusGruen;

		if(this.roteLampe) {
			statusRot = "an";
		} else {
			statusRot = "aus";
		}

		if(this.grueneLampe) {
			statusGruen = "an";
		} else {
			statusGruen = "aus";
		}

		System.out.println("Die Rote Lampe ist " + statusRot + ".");
		System.out.println("Die Gruene Lampe ist " + statusGruen + "
		    .");
	}
}
```

Quellcode 7.6: *Ampel.java*

```java
package task0708;

import task0707.Ampel;

public class Programm {


	public static void main(String[] args) {
		Ampel eineAmpel = new Ampel();

		for (int i = 0; i < 20; i++) {
			if (i % 2 == 0) {
				eineAmpel.schalteGruen(false);
				eineAmpel.schalteRot(true);
			} else {
				eineAmpel.schalteGruen(true);
				eineAmpel.schalteRot(false);
			}
			System.out.println(i + "-ter Zustand:");
			eineAmpel.druckeZustand();
		}
	}

}
```

Quellcode 7.7: *Programm.java*

```java
package task0709;

public class Test {

	int[] a;

	public Test(int n) {
		this.a = new int[n];
	}

	public void invers() {
		for (int i = 0; i < this.a.length / 2; i++) {
			int mem = this.a[i];
			this.a[i] = this.a[this.a.length - i];
			this.a[this.a.length - 1] = mem;
		}
	}
}
```

Quellcode 7.8: *Test.java*

```java
package task0710;

import task0706.Studierende;

public class Test {

	Studierende[] a;

	public Test(int n) {
		this.a = new Studierende[n];
	}

	public void invers() {
		for (int i = 0; i < this.a.length / 2; i++) {
			Studierende mem = this.a[i];
			this.a[i] = this.a[this.a.length - i - 1];
			this.a[this.a.length - 1] = mem;
		}
	}
}
```

Quellcode 7.9: *Test.java*

```java
package task0711;

public class Vektor {

	float[] values;

	public Vektor(float[] values) {
		this.values = values;
	}

	public float gibWert(int i) {
		if (i >= 0 && i < this.values.length) {
			return this.values[i];
		} else {
			return -100000F;
		}
	}

	public int gibMinimum() {
		int min = 0;

		for (int i = 1; i < this.values.length; i++) {
			if (this.values[i] < this.values[min]) {
				min = i;
			}
		}
		return min;
	}
}
```

Quellcode 7.10: *Vektor.java*

```java
package task0712;

import task0706.Studierende;

class Suchprogramm {

	static Studierende suche(Studierende[] d, int m) {
		int i = 0;
		int n = d.length;

		while ((i < n) && (d[i].gibMatrikelnummer() != m)) {
			i = i + 1;
		}
		if (i < n) {
			return d[i];
		} else {
			return null;
		}
	}

	static void druckeNamen(Studierende[] d, int m) {
		int i = 0;

		while (i < d.length) {
			if (d[i].gibMatrikelnummer() <= m) {
				System.out.println(d[i].gibStudname());
			}
		}
	}

	public static void main(String[] args) {
		Studierende[] testd = new Studierende[3];
		testd[0] = new Studierende("Thorsten Meier", 88188, 1980);
		testd[1] = new Studierende("Monika Schmidt", 88633, 1981);
		testd[2] = new Studierende("Monika Schneider", 88755, 1980);
		int m = 88633;
		boolean gefunden = (suche(testd, m) != null);

		if (gefunden) {
			System.out.println(m + " gefunden");
		} else {
			System.out.println(m + " nicht gefunden");
		}
	}

}
```

Quellcode 7.11: *Suchprogramm.java*

```java
package task0713;

import task0711.Vektor;

public class Programm {


        public static void main(String[] args) {
                float[] values = {2F, 3F, -1.7F};
                Vektor testvektor = new Vektor(values);

                System.out.println("Wert 0 ist: " + testvektor.gibWert(0));
                System.out.println("Wert 1 ist: " + testvektor.gibWert(1));
                System.out.println("Wert 2 ist: " + testvektor.gibWert(2));
                System.out.println("Der niedrigste Wert steht an Stelle: " +
                     testvektor.gibMinimum());
        }
}
```

Quellcode 7.12: *Programm.java*

```java
package task0715;

import task0706.Studierende;

public class Liste {



        static Studierende lSuche(Liste eineListe, int m) {
                if (this.gibWert().gibMatrikelnummer == m) {
                        return this.gibWert();
                } else if (this.gibNaechstes() == null) {
                        return null;
                } else {
                        lSuche(this.gibNaechstes(), m);
                }
        }
}
```

Quellcode 7.13: *Liste.java*

Erweiterte Programmierkonzepte

Kapitel 8

Andere Programmierstile

Aufgabe 8.1

Zu dieser Übungsaufgabe liegt keine Lösung vor.

Aufgabe 8.2

Zu dieser Übungsaufgabe liegt keine Lösung vor.

Kapitel 9

Objektorientierte Programmierung

Aufgabe 9.1

(a) Die Reihenfolge der aufgerufenen Objekte ist:

- `dListe`
- `ListenVerwalter` von Thorsten Meier
- `Studierende` Thorsten Meier
- `ListenVerwalter` von Monika Schmidt
- `Studierende` Monika Schmidt
- `ListenVerwalter` von Monika Schneider
- `Studierende` Monika Schneider
- `Neuelement`

(b) Die Methode `Studierende lSuche(int m)` könnte gemäß Quellcode 9.1 aussehen.

```
1 Studierende lSuche(int m) {
2         do {
3                 if (this.gibWert().gibMatrikelnummer() == m) {
4                         return this.gibWert();
5                 }
6         } while (this.gibNaechstes() != null)
7         return null;
8 }
```

Quellcode 9.1: *Die Methode `Studierende lSuche(int m)`*

```java
public class Einrichtungsgegenstand {

        String name;
        int produktionsjahr;
        double preis;
}
```

Quellcode 9.2: *Einrichtungsgegenstand.java*

```java
public Einrichtungsgegenstand(String name, int produktionsjahr, double preis
    ) {
        this.name = name;
        this.produktionsjahr = produktionsjahr;
        this.preis = preis;
}

public Einrichtungsgegenstand() {}
```

Quellcode 9.3: *Einrichtungsgegenstand.java*

Aufgabe 9.2

(a) Die Klasse Einrichtungsgegenstand – s. Quellcode 9.2.

(b) Die zwei Konstruktoren der erweiterten Klasse – s. Quellcode 9.3.

(c) Die Klasse Einrichtungsgegenstand um das Prinzip der Sichtbarkeit erweitert – s. Quellcode 9.4.

(d) Die Methode getVerkaufspreis() – s. Quellcode 9.5. Zum Abschluss wird die komplette Klasse Einrichtungsgegenstand noch einmal angesehen – s. Quellcode 9.6.

(e) Um die Klasse Einrichtungsgegenstand zu testen, wird die Klasse Einrichtungshaus mit einer main-Methode implementiert. Diese könnte wie folgt aussehen[1] – s. Quellcode 9.7.

Aufgabe 9.3

(a) Zu dieser Übungsaufgabe liegt keine Lösung vor.

(b) Die Klassen `Tisch`, `Sitzgelegenheit` und `Stuhl` in Java implementiert – s. Quellcodes 9.8, 9.9 und 9.10.

(c) Die überschreibenden Methoden finden sich in den jeweiligen Klassen.

[1] Da im Buch an dieser Stelle zweimal von gegenstandEins die Rede ist, wird der zweite Gegenstand zur Vereinfachung gegenstandZwei genannt.

```
1  private String name;
2  private int produktionsjahr;
3  private double preis;
4
5  public void setName(String name) {
6          this.name = name;
7  }
8
9  public String getName() {
10         return this.name;
11 }
12
13 public void setProduktionsjahr(int produktionsjahr) {
14         this.produktionsjahr = produktionsjahr;
15 }
16
17 public int getProduktionsjahr() {
18         return this.produktionsjahr;
19 }
20
21 public void setPreis(double preis) {
22         this.preis = preis;
23 }
24
25 public double getName() {
26         return this.preis;
27 }
```

Quellcode 9.4: *Einrichtungsgegenstand.java*

```
1  public double getVerkaufspreis() {
2          return (1.15 * this.preis);
3  }
```

Quellcode 9.5: *Einrichtungsgegenstand.java*

Aufgabe 9.4

(a) Die Klasse `Einrichtungsverwaltung` – s. Quellcodes 9.11.

(b) Die Liste wird in der `main`-Methode erstellt[2].

(c) Die Auflistung der Gegenstände wird mit der Methode `listContent()` der Klasse `Einrichtungsverwaltung` realisiert.

[2] Die einzufügenden Gegenstände stehen im Aufgabenteil e) der Aufgabe

```java
package task0802;

public class Einrichtungsgegenstand {

        public String name;
        public int produktionsjahr;
        public double preis;

        public Einrichtungsgegenstand(String name, int produktionsjahr,
            double preis) {
                this.name = name;
                this.produktionsjahr = produktionsjahr;
                this.preis = preis;
        }

        public Einrichtungsgegenstand() {}

        public void setName(String name) {
                this.name = name;
        }

        public String getName() {
                return this.name;
        }

        public void setProduktionsjahr(int produktionsjahr) {
                this.produktionsjahr = produktionsjahr;
        }

        public int getProduktionsjahr() {
                return this.produktionsjahr;
        }

        public void setPreis(double preis) {
                this.preis = preis;
        }

        public double getPreis() {
                return this.preis;
        }

        public double getVerkaufspreis() {
                return (1.15 * this.preis);
        }
}
```

Quellcode 9.6: *Einrichtungsgegenstand.java*

```java
package task0802;

public class Einrichtungshaus {

	public static void main(String[] args) {
		Einrichtungsgegenstand gegenstandEins = new
		    Einrichtungsgegenstand("kleinerTisch", 2004, 569.99);
		Einrichtungsgegenstand gegenstandZwei = new
		    Einrichtungsgegenstand();

		gegenstandZwei.setName("runderStuhl");
		gegenstandZwei.setProduktionsjahr(2003);
		gegenstandZwei.setPreis(80.00);
	}
}
```

Quellcode 9.7: *Einrichtungshaus.java*

```java
package task0803;

import task0802.Einrichtungsgegenstand;

public class Tisch extends Einrichtungsgegenstand {

	private int anzahlBeine;

	// Fuer Aufgabe 8.4

	public Tisch(String name, int produktionsJahr, double preis) {
		super(name, produktionsJahr, preis);
	}

	public void setAnzahlBeine(int anzahlBeine) {
		this.anzahlBeine = anzahlBeine;
	}

	public int getAnzahlBeine() {
		return this.anzahlBeine;
	}

	public double getVerkaufspreis() {
		return 1.2 * this.preis;
	}
}
```

Quellcode 9.8: *Tisch.java*

```java
package task0803;

import task0802.Einrichtungsgegenstand;

public class Sitzgelegenheit extends Einrichtungsgegenstand {

        // Fuer Aufgabe 8.4

        public Sitzgelegenheit(String name, int produktionsJahr, double
            preis) {
                super(name, produktionsJahr, preis);
        }

        boolean stoffbesatz;
}
```

Quellcode 9.9: *Sitzgelegenheit.java*

```java
package task0803;

public class Stuhl extends Sitzgelegenheit {

        boolean armlehne;

        // Fuer Aufgabe 8.4

        public Stuhl(String name, int produktionsJahr, double preis) {
                super(name, produktionsJahr, preis);
        }

        public double getVerkaufspreis() {
                return 1.16 * this.preis;
        }
}
```

Quellcode 9.10: *Stuhl.java*

```java
package task0804;

import task0802.Einrichtungsgegenstand;
import task0803.Stuhl;
import task0803.Tisch;

public class Einrichtungsverwaltung {

	Einrichtungsgegenstand content;
	Einrichtungsverwaltung next;

	public void setContent(Einrichtungsgegenstand content) {
		this.content = content;
	}

	public Einrichtungsgegenstand getContent() {
		return this.content;
	}

	public void insert(Einrichtungsgegenstand content) {
		if (this.next == null) {
			this.next = new Einrichtungsverwaltung();
			this.next.setContent(content);
		} else {
			this.next.insert(content);
		}
	}

	public void setNext(Einrichtungsverwaltung next) {
		this.next = next;
	}

	public Einrichtungsverwaltung getNext() {
		return this.next;
	}

	public void listContent() {
		String type = this.content.getClass().getSimpleName();
		String name = this.content.getName();
		int year = this.content.getProduktionsjahr();
		double price = this.content.getPreis();

		System.out.println(type + " - Name: " + name + ", Jahr: " +
		    year +
				", Preis: " + price + " Euro.");
		if (this.next != null) {
			this.next.listContent();
		}
	}

	public static void main(String args[]) {
		Einrichtungsverwaltung list = new Einrichtungsverwaltung();
		list.setContent(new Tisch("kleinerTisch", 2004, 569.99));
		list.insert(new Stuhl("runderStuhl", 2003, 80.00));

		list.listContent();
	}
}
```

Quellcode 9.11: *Einrichtungsverwaltung.java*

Kapitel **10**

Klassenbibliotheken

Aufgabe 10.1

(a) Besonderheiten verschiedener Browser:

Safari Keinerlei Besonderheiten.

Firefox Keinerlei Besonderheiten.

Opera Keinerlei Besonderheiten.

Konqueror Keinerlei Besonderheiten.

Internet Explorer Keinerlei Besonderheiten.

(b) Quellcode 10.1 vermittelt die Ausgabe der Tabelle.

Aufgabe 10.2

(a) Siehe Quellcode 10.2.

(b) Siehe Quellcode 10.2.

(c) Siehe Quellcode 10.3.

Aufgabe 10.3

Das Programm `EinProgramm` (s. Quellcode 10.4), welches zur Laufzeit eine Eingabe ausgibt.

```
1  package task0901;
2
3  import java.applet.Applet;
4  import java.awt.Graphics;
5
6  public class Table extends Applet {
7
8          public void paint(Graphics g) {
9                  int xPos = 50;
10                 int yPos = 50;
11
12                 g.drawString("n\t n^2\t n^3", xPos, yPos);
13                 for (int i = 1; i < 4; i++) {
14                         yPos += 25;
15
16                         g.drawString(i + "\t" + i * i + "\t" + i * i * i,
                               xPos, yPos);
17                 }
18         }
19 }
```

Quellcode 10.1: *Table.java*

Aufgabe 10.4

(a) Das Programm `Flaeche`, welches sich nun komplett über die Tastatur steuern lässt – s. Quellcode 10.5.

(b) Zu dieser Übungsaufgabe liegt keine Lösung vor.

Aufgabe 10.5

(a) Um die Klassen als Paket nutzen zu können, müssen die Klassen in das Verzeichnis `./studierende/` verschoben werden.

(b) Alle in diesem Verzeichnis befindlichen Klassen erhalten als ersten Befehl „`package studierende;` “. Die Klassen, die dieses Paket nun nutzen sollen, binden es dann über „`import studierende.*;` “ ein[1].

[1] Das Paket `studierende` muss nach den Java-Code-Conventions klein geschrieben werden.

```java
package task0902;

import java.util.GregorianCalendar;

public class Datum {

	private int jahr;
	private int monat;
	private int tag;

	private static GregorianCalendar now = new GregorianCalendar();

	public Datum() {
		this(now.get(GregorianCalendar.YEAR),
				now.get(GregorianCalendar.MONTH + 1),
				now.get(GregorianCalendar.DAY_OF_MONTH));
	}

	public Datum(int jahr, int monat, int tag) {
		this.jahr = jahr;
		this.monat = monat;
		this.tag = tag;
	}

	public int anzahlMonate() {
		now = new GregorianCalendar();
		int diffJahr = now.get(GregorianCalendar.YEAR) - this.jahr;
		diffJahr = Math.abs(diffJahr);
		int diffMonat = now.get(GregorianCalendar.MONTH) + 1 - this.
		    monat;
		diffMonat = Math.abs(diffMonat);

		int res = diffJahr * 12 + diffMonat;

		return res;
	}
}
```

Quellcode 10.2: *Datum.java*

```
1  package task0902;
2
3  public class Testprogramm {
4
5
6          public static void main(String[] args) {
7                  Datum einDatum = new Datum(1989, 5, 15);
8                  System.out.println(einDatum.anzahlMonate());
9          }
10
11 }
```

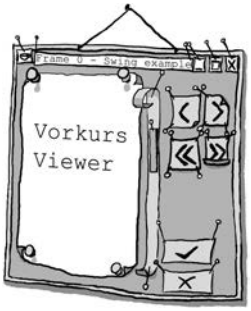
Quellcode 10.3: *Testprogramm.java*

```java
package task0903;

import java.io.*;

public class EinProgramm {

	String name;

	public EinProgramm() {
		this.name = "";
	}

	public void query() {
		System.out.println("Wie ist dein Name?");
	}

	public void answer() {
		BufferedReader input = new BufferedReader(new
		    InputStreamReader(System.in));

		try {
			name = input.readLine();
		} catch (IOException e) {
			e.printStackTrace();
		}

		System.out.println("Hallo " + name + "!");
	}


	public static void main(String[] args) {
		EinProgramm stream = new EinProgramm();

		stream.query();
		stream.answer();
	}
}
```

Quellcode 10.4: *EinProgramm.java*

```java
package task0904;

import java.io.*;

public class Flaeche {

	double[] numbers;

	public Flaeche() {
		this.numbers = new double[2];
	}

	public void query() {
		System.out.println("Geben sie eine Kantenlaenge ein.");
	}

	public void answer(int x) {
		BufferedReader input = new BufferedReader(new
		    InputStreamReader(System.in));

		try {
			this.numbers[x] = Double.parseDouble(input.readLine
			    ());
		} catch (IOException e) {
			e.printStackTrace();
		}
	}

	public double getSize() {
		return this.numbers[0] * this.numbers[1];
	}

	public void exit() {
		BufferedReader input = new BufferedReader(new
		    InputStreamReader(System.in));
		String s = "";

		System.out.println("Wollen sie abbrechen? (ja)");
		try {
			s = input.readLine();
		} catch (IOException e) {
			e.printStackTrace();
		}

		if ("ja".equals(s)) {
			System.exit(0);
		}
	}


	public static void main(String[] args) {
		Flaeche stream = new Flaeche();

		while (true) {
			stream.query();
			stream.answer(0);
			stream.query();
			stream.answer(1);
			System.out.println("Die Flaeche betraegt " + stream.
			    getSize());
			stream.exit();
		}
	}
}
```

Quellcode 10.5: *Flaeche.java*



Kapitel **11**

Grafikprogrammierung mit Swing

Aufgabe 11.1

(a)
- *AWT* ist das Grafikpaket aus Java 1.0. Es bietet grundlegende Funktionen zur GUI-Gestaltung sowie elementare Zeichenfunktionen.
- *Swing* ist das aktuelle Grafikpaket von Java und seit Version 1.1 verfügbar. Swing zeichnet sich durch ein einheitliches Erscheinungsbild auf allen Betriebssystemen aus. Es bietet weiterhin die Möglichkeit ein *Look & Feel* zu bestimmen.
- *Java 2D* ist eine Java-Bibliothek zur Nutzung hardwarebeschleunigter 2D-Grafikbefehle.
- *Drag und Drop* zur Unterstützung des Datenaustauschs zwischen verschiedenen Programmen.
- *Accessibility* sind Erweiterungen für Sehbehinderte.

(b) *Schwergewichtige Komponenten* nutzen Objekte des Betriebssystems, *leichtgewichtige Komponenten* dagegen sind komplett in Java realisiert.

Aufgabe 11.2

(a) siehe Quellcode 11.1

(b) siehe Quellcode 11.1

(c) siehe Quellcode 11.1

Aufgabe 11.3

(a) siehe Quellcode 11.2

(b) siehe Quellcode 11.2

Aufgabe 11.4

Der Countdown wird mit der von Java bereitgestellten Timer-API implementiert und in drei Klassen aufgeteilt.

(a) siehe Quellcodes 11.3, 11.4 und 11.5

(b) siehe Quellcodes 11.3, 11.4 und 11.5

(c) siehe Quellcodes 11.3, 11.4 und 11.5

Aufgabe 11.5

(a) siehe Quellcodes 11.8

(b) siehe Quellcodes 11.7

(c) Eine Ampel, die sich auf Knopfdruck umstellen lässt, besteht zum Beispiel aus folgenden drei Klassen:

`TrafficLight.java` Dient als Controller-Klasse, die die Ereignisse verarbeitet und somit alles steuert (s. Quellcode 11.6).

`Controls.java` Beinhaltet nur die `JButton`-Objekte (s. Quellcode 11.7).

`DrawPanel.java` Das `JPanel`, auf welchem gezeichnet wird (s. Quellcode 11.8).

Aufgabe 11.6

Damit die Klasse `DrawPanel` die Farben nicht überschreibt, wird diese in einem weiteren `Vector` abgespeichert. In diesem Beispiel wird gleichzeitig die Verwendung von sogenannten *generics* demonstriert. Hiermit kann eine allgemeine Datenstruktur für `Object`-Objekte auf etwas Spezielleres abgeleitet werden. Im Vektor `Vector<Color>` werden nur Objekte des Typs `Color` gespeichert. Weiterhin wurde die Methode `getRandomColor()` erstellt, die hier von Vorteil ist – s. Quelltext 11.9.

Aufgabe 11.7

(a) siehe Quellcode 11.10

(b) siehe Quellcode 11.10

Aufgabe 11.8

Da die Lösung hierzu sehr umfangreich ist, steht sie nur auf der Internetseite zum Download zur Verfügung (Quellcode „Repeater“).

```java
package task1002;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

import javax.swing.JFrame;

public class DrawFrame extends JFrame {

	public DrawFrame(String title) {
		super(title);
		System.out.println("Width:");
		int x = Integer.parseInt(this.readLine());
		System.out.println("Height:");
		int y = Integer.parseInt(this.readLine());
		this.setSize(x, y);
		this.setVisible(true);
	}

	public DrawFrame() {
		this("Ein Java-Swing Fenster");
	}

	public String readLine() {
		String input = "";
		BufferedReader keyboard = new BufferedReader(new
		    InputStreamReader(System.in));

		try {
			input = keyboard.readLine();
		} catch (IOException e) {
			e.printStackTrace();
		}
		return input;
	}


	public static void main(String[] args) {
		DrawFrame frameEins = new DrawFrame();
		DrawFrame frameZwei = new DrawFrame("Ein anderer Titel");


		// frameEins.setSize(400, 300);
		frameEins.setLocation(50, 50);

		// frameZwei.setSize(200, 350);
		frameZwei.setLocation(600, 200);
	}
}
```

Quellcode 11.1: *DrawFrame.java*

```java
package task1003;

import java.awt.Dimension;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class DrawFrame extends JFrame {

	private JLabel labelCount;

	public DrawFrame() {
		super("Ein Java-Swing-Fenster");

		JPanel drawPanel = new JPanel();
		drawPanel.setPreferredSize(new Dimension(300, 50));

		JButton testButton = new JButton("Test-Button");
		drawPanel.add(testButton);

		JLabel labelText = new JLabel("Countdown: ");
		drawPanel.add(labelText);

		labelCount = new JLabel("1000");
		drawPanel.add(labelCount);

		this.add(drawPanel);

		this.setLocation(300, 300);
		this.pack();

		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		this.setVisible(true);
	}

	public void startCountDown(int from, int to) {
		for (int i = from; i >= to; i--) {
			this.labelCount.setText(Integer.toString(i));
			try {
				Thread.sleep(1000);
			} catch (InterruptedException e) {
				e.printStackTrace();
			}
		}
	}

	public static void main(String[] args) {
		DrawFrame frame = new DrawFrame();
		frame.startCountDown(1000, 0);
	}
}
```

Quellcode 11.2: *DrawFrame.java*

```java
package task1004;

import java.util.TimerTask;

public class Countdown extends TimerTask {

	private int value;
	private CountdownPanel parent;

	public Countdown(int startValue, CountdownPanel parent) {
		this.value = startValue;
		this.parent = parent;

		this.printValue();
	}

	@Override
	public void run() {
		if (this.value > 0) {
			this.value--;
			this.printValue();
		}
	}

	private void printValue() {
		this.parent.getView().setText(String.valueOf(this.value));
	}

	public int getValue() {
		return this.value;
	}
}
```

Quellcode 11.3: *Countdown.java*

```java
package task1004;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Timer;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;

@SuppressWarnings("serial")
public class CountdownPanel extends JPanel implements ActionListener {

	private JButton control;
	private JLabel view;
	private boolean active;
	private Timer t;
	private Countdown c;
	private int tmp;

	public CountdownPanel() {
		this.control = new JButton("Control");
		this.view = new JLabel();
		this.active = false;
		this.t = new Timer();
		this.tmp = 1000;;

		this.control.addActionListener(this);

		this.add(this.control);
		this.add(this.view);
	}

	public JLabel getView() {
		return this.view;
	}

	public void actionPerformed(ActionEvent e) {
		if (e.getSource() == this.control) {
			this.toggleTimer();
		}
	}

	private void toggleTimer() {
		if (this.active) {
			this.c.cancel();
			this.tmp = this.c.getValue();
		} else {
			this.c = new Countdown(this.tmp, this);
			this.t.scheduleAtFixedRate(this.c, 0, 1000);
		}
		this.active = !this.active;
	}

}
```

Quellcode 11.4: *CountdownPanel.java*

```java
package task1004;

import javax.swing.JFrame;

@SuppressWarnings("serial")
public class GUI extends JFrame {

	private CountdownPanel p;

	public GUI() {
		this.p = new CountdownPanel();

		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		this.setTitle("Timer-Test");
		this.add(this.p);
		this.pack();
		this.setVisible(true);
	}


	public static void main(String[] args) {
		@SuppressWarnings("unused")
		GUI tmp = new GUI();
	}

}
```

Quellcode 11.5: *GUI.java*

```java
package task1005;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JFrame;

public class TrafficLight extends JFrame implements ActionListener {

	private Controls c;
	private DrawPanel p;

	public TrafficLight() {
		super("Ampel");

		this.c = new Controls(this);
		this.p = new DrawPanel();

		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		this.setLayout(new BorderLayout());
		this.add(this.p, BorderLayout.CENTER);
		this.add(this.c, BorderLayout.PAGE_END);

		this.setSize(new Dimension(800, 500));
		this.setVisible(true);
	}


	public static void main(String[] args) {
		TrafficLight tmp = new TrafficLight();
	}

	public void actionPerformed(ActionEvent e) {
		if (e.getActionCommand().equals("car")) {
			this.p.setCar(true);
		} else if(e.getActionCommand().equals("passenger")) {
			this.p.setCar(false);
		}
	}

}
```

Quellcode 11.6: *TrafficLight.java*

```java
package task1005;

import java.awt.GridLayout;

import javax.swing.JButton;
import javax.swing.JPanel;

public class Controls extends JPanel {

	private TrafficLight controller;
	private JButton car;
	private JButton passenger;

	public Controls(TrafficLight controller) {
		this.controller = controller;
		this.car = new JButton("Autofahrer");
		this.passenger = new JButton("Fussgaenger");

		this.car.setActionCommand("car");
		this.car.addActionListener(this.controller);

		this.passenger.setActionCommand("passenger");
		this.passenger.addActionListener(this.controller);

		this.setLayout(new GridLayout(1, 2));
		this.add(this.car);
		this.add(this.passenger);
	}

}
```

Quellcode 11.7: *Controls.java*

```java
package task1005;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;

import javax.swing.JPanel;

public class DrawPanel extends JPanel {

	private boolean isCar;
	private Color top;
	private Color bottom;
	private final int lightSize = 50;

	public DrawPanel() {
		this.isCar = true;

		this.setColors();
		this.setPreferredSize(new Dimension(800, 500));
	}

	protected void paintComponent(Graphics g) {
		g.setColor(this.top);
		g.fillOval(this.getWidth() / 2 - this.lightSize / 2,
				200, this.lightSize, this.lightSize);

		g.setColor(this.bottom);
		g.fillOval(this.getWidth() / 2 - this.lightSize / 2,
				200 + this.lightSize + 5, this.lightSize,
				    this.lightSize);
	}

	public void setCar(boolean b) {
		this.isCar = b;
		this.setColors();
		this.repaint();
	}

	private void setColors() {
		if (this.isCar) {
			this.top = Color.GRAY;
			this.bottom = Color.GREEN;
		} else {
			this.top = Color.RED;
			this.bottom = Color.GRAY;
		}
	}
}
```

Quellcode 11.8: *DrawPanel.java*

```
1  package task1006;
2
3  import java.awt.*;
4  import java.awt.event.*;
5  import javax.swing.*;
6  import java.awt.geom.Line2D;
7  import java.util.Vector;
8
9  public class DrawPanel extends JPanel implements MouseListener,
       MouseMotionListener {
10
11         private Vector drawnGeom;
12         private Vector<Color> colors;
13
14         public DrawPanel() {
15                 super();
16                 addMouseListener(this);
17                 addMouseMotionListener(this);
18                 drawnGeom = new Vector();
19                 this.colors = new Vector<Color>();
20         }
21
22         public void mousePressed(MouseEvent e) {
23                 setFirstLinePoint(e.getPoint());}
24
25         public void mouseClicked(MouseEvent e) {}
26
27         public void mouseReleased(MouseEvent e) {}
28
29         public void mouseEntered(MouseEvent e) {}
30
31         public void mouseExited(MouseEvent e) {}
32
33         public void mouseDragged(MouseEvent e) {
34                 setSecondLinePoint(e.getPoint());}
35
36         public void mouseMoved(MouseEvent e) {}
37
38         private void setFirstLinePoint(Point p) {
39                 drawnGeom.addElement(new Line2D.Double(p, p));
40                 this.colors.add(this.getRandomColor());
41                 this.repaint();
42         }
43
44         private void setSecondLinePoint(Point p) {
45                 ((Line2D.Double) drawnGeom.lastElement()).setLine(((Line2D.
                       Double)
46                           drawnGeom.lastElement()).getP1(), p);
47                 this.repaint();
48         }
49
50         private Color getRandomColor() {
51                 int r = (int) (Math.random() * 256);
52                 int g = (int) (Math.random() * 256);
53                 int b = (int) (Math.random() * 256);
54                 return new Color(r, g, b);
55         }
56
57         protected void paintComponent(Graphics g) {
58                 super.paintComponent(g);
59                 for (int i = 0; i < this.drawnGeom.size(); i++) {
60                         Line2D.Double line = (Line2D.Double) this.drawnGeom.
                               elementAt(i);
61                         g.setColor(this.colors.elementAt(i));
62                         g.drawLine((int) line.x1, (int) line.y1, (int) line.
                               x2, (int) line.y2);
```

```java
package task1007;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.geom.Line2D;
import java.awt.geom.Point2D;
import java.util.Vector;

public class DrawPanel extends JPanel implements MouseListener {
	private Point last;
	private Point act;
	private Vector<Line2D.Double> drawnGeom;
	private Vector<Color> colors;

	public DrawPanel() {
		this.drawnGeom = new Vector<Line2D.Double>();
		this.colors = new Vector<Color>();
		this.last = null;
		this.addMouseListener(this);
	}

	private void startLine(Point p) {
		this.colors.add(this.getRandomColor());
		if (this.last == null) {
			this.last = p;
			this.drawnGeom.add(new Line2D.Double(p, p));
		} else {
			this.addLine(p);
		}
	}

	private void addLine(Point p) {
		this.drawnGeom.lastElement().setLine(this.drawnGeom.
		    lastElement().getP1(), p);
		this.drawnGeom.add(new Line2D.Double(p, p));
		this.repaint();
		if (p.equals(this.last)) {
			this.endLine();
		}
	}

	private void endLine() { this.last = null; }

	protected void paintComponent(Graphics g) {
		super.paintComponent(g);
		Graphics2D g2d = (Graphics2D) g;
		g2d.setStroke(new BasicStroke(5.0f));
		for (int i = 0; i < this.drawnGeom.size(); i++) {
			Line2D.Double l = this.drawnGeom.elementAt(i);
			g2d.setColor(this.colors.elementAt(i));
			g2d.draw(l);
		}
	}

	private Color getRandomColor() {
		int r = (int) (Math.random() * 256);
		int g = (int) (Math.random() * 256);
		int b = (int) (Math.random() * 256);
		return new Color(r, g, b);
	}

	public void mouseClicked(MouseEvent e) { this.startLine(e.getPoint()
	    ); }

	public void mouseEntered(MouseEvent e) {}
```

Kapitel 12

Programmieren in C++

Aufgabe 12.1

Zu dieser Übungsaufgabe liegt keine Lösung vor.

Aufgabe 12.2

siehe Quellcode 12.1

```
1 // Vorkurs Informatik
2
3 #include <iostream>
4
5 int main(int argc, char* argv[]){
6   std::cout << "Viel Spass mit C++" << std::endl;
7   return 0;
8 }
```

Quellcode 12.1: *basic2.cpp*

Aufgabe 12.3

siehe Quellcode 12.2

```
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 int main(int argc, char* argv[]){
7         int start = -1;
```

```
8         int end = 8;
9         double inkrement = 1.5;
10        double value = start + inkrement;
11        while (value < end){
12                cout << value << endl;
13                value = value + inkrement;
14        }
15        return 0;
16 }
```

Quellcode 12.2: *inkrement.cpp*

Aufgabe 12.4

(a) siehe Quellcode 12.3

(b) siehe Quellcode 12.4

```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  int main(int argc, char* argv[]){
7         int a[] = {11,5,8,3,15,13,9,19,18,10,4};
8         int minimum = a[0];
9         int i = 1;
10        int n = sizeof(a) / sizeof(a[0]);
11        while (i < n){
12                if(a[i] < minimum) minimum = a[i];
13                i = i + 1;
14        }
15        cout << "Minimum:" << minimum << endl;
16        return 0;
17 }
```

Quellcode 12.3: *minimum.cpp*

```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  int main(int argc, char* argv[]){
7         int a[] = {7, 2, 9, 3, 2, 5};
8         int minimum = a[0];
```

```
 9         int i = 0;
10         int j = 0;
11         int n = sizeof(a) / sizeof(a[0]);
12         while (i < n){
13                 while (j < n){
14                         if((a[i] == a[j]) && (i!=j)){
15                                 cout << "Mehrfache Werte vorhanden" << endl;
16                                 return 0;
17                         }
18                         j = j + 1;
19                 }
20                 j = 0;
21                 i = i + 1;
22         }
23         cout << "Keine mehrfachen Werte" << endl;
24         return 0;
25 }
```

Quellcode 12.4: *doppelteWerte.cpp*

Aufgabe 12.5

siehe Quellcode 12.5

```
 1 #include <iostream>
 2 #include <string>
 3 #include <vector>
 4
 5 using namespace std;
 6
 7 int main(int argc, char* argv[]){
 8         vector<int> einVektor(11);
 9         einVektor[0] = 11;
10         einVektor[1] = 5;
11         einVektor[2] = 8;
12         einVektor[3] = 3;
13         einVektor[4] = 15;
14         einVektor[5] = 13;
15         einVektor[6] = 9;
16         einVektor[7] = 19;
17         einVektor[8] = 18;
18         einVektor[9] = 10;
19         einVektor[10] = 4;
20         unsigned int index = 1;
21         unsigned int laengeVektor = einVektor.size();
22         int minimum = einVektor[0];
23         while (index < laengeVektor){
```

```
24                  if(einVektor[index] < minimum) minimum = einVektor[index];
25                  index = index + 1;
26          }
27          cout << "Minimum:" << minimum << endl;
28          return 0;
29  }
```

Quellcode 12.5: *MinimumVector.cpp*

Aufgabe 12.6

(a) Zu dieser Übungsaufgabe liegt keine Lösung vor.

(b) Zu dieser Übungsaufgabe liegt keine Lösung vor.

(c) Zu dieser Übungsaufgabe liegt keine Lösung vor.

Aufgabe 12.7

(a) Zu dieser Übungsaufgabe liegt keine Lösung vor.

(b) Zu dieser Übungsaufgabe liegt keine Lösung vor.

Aufgabe 12.8

(a) Zu dieser Übungsaufgabe liegt keine Lösung vor.

(b) Zu dieser Übungsaufgabe liegt keine Lösung vor.

Kapitel 13

Modellgestützte Softwareentwicklung

Aufgabe 13.1

Zu dieser Übungsaufgabe liegt keine Lösung vor.

Aufgabe 13.2

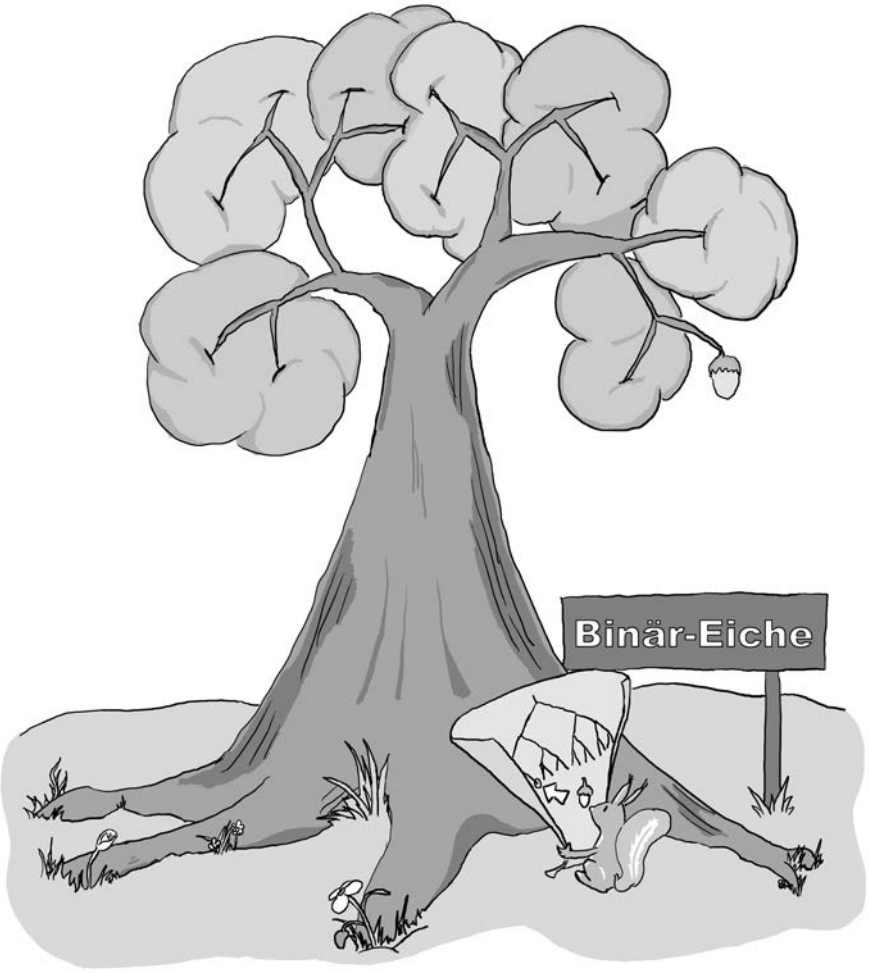
Zu dieser Übungsaufgabe liegt keine Lösung vor.

Aufgabe 13.3

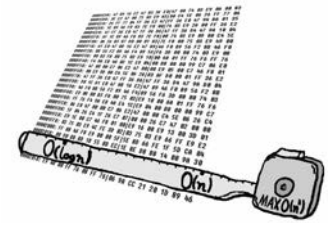
Zu dieser Übungsaufgabe liegt keine Lösung vor.

Aufgabe 13.4

Zu dieser Übungsaufgabe liegt keine Lösung vor.



Algorithmen und Datenstrukturen



Kapitel 14

Asymptotische Aufwandsanalyse

Aufgabe 14.1

- richtig
- richtig
- falsch
- falsch

Aufgabe 14.2

- $O(n^2)$
- $O(n \cdot \log(n))$
- $O(\log(n))$

Aufgabe 14.3

(a) $O(n)$

(b) $O(n)$

Kapitel 15

Sortieren

Aufgabe 15.1

$$T(n) = T(n-1) + O(1)$$

Aufgabe 15.2

(a) Gegeben ist die Funktion $S(n)$ mit

$$S(n) = \left(\frac{n \cdot (n+1)}{2}\right)^2,$$

welche die Summe der ersten n Dreierpotenzen $1^3 + 2^3 + \cdots + n^3$ für alle $n \geq 1$ berechnen soll. Dieses wird mit Hilfe der vollständigen Induktion bewiesen.

Beweis. **Induktionsanfang** $n = 1$.

$$1^3 = \left(\frac{1 \cdot (1+1)}{2}\right)^2 \tag{15.1}$$

$$= 1 \tag{15.2}$$

Auf Grund der Summe der Dreierpotenzen muss $S(n) = S(n-1) + n^3$ sein. Somit wird mit folgendem Induktionsschritt fortgefahren.

Induktionsschritt $(n-1) \to n$.

$$S(n-1) + n^3 = S(n) \tag{15.3}$$

$$\left(\frac{n \cdot (n-1)}{2}\right)^2 + n^3 = \left(\frac{n \cdot (n+1)}{2}\right)^2 \tag{15.4}$$

$$= \left(\frac{n \cdot (n-1) + 2 \cdot n}{2}\right)^2 \tag{15.5}$$

$$= \left(\frac{n \cdot (n-1)}{2} + \frac{2 \cdot n}{2}\right)^2 \tag{15.6}$$

$$= \left(\frac{n \cdot (n-1)}{2} + n\right)^2 \tag{15.7}$$

$$= \left(\frac{n \cdot (n-1)}{2}\right)^2 + 2 \cdot \frac{n \cdot (n-1)}{2} \cdot n + n^2 \tag{15.8}$$

$$= \left(\frac{n \cdot (n-1)}{2}\right)^2 + \underbrace{n^2 - n^2}_{0} + n^3 \tag{15.9}$$

$$= \left(\frac{n \cdot (n-1)}{2}\right)^2 + n^3 \tag{15.10}$$

□

Es wird bewiesen, dass die Summe der ersten n Dreierpotenzen $S(n)$ als

$$S(n) = \left(\frac{n \cdot (n+1)}{2}\right)^2$$

definiert ist.

(b) Zu dieser Übungsaufgabe liegt keine Lösung vor.

(c) Zu dieser Übungsaufgabe liegt keine Lösung vor.

Aufgabe 15.3

Behauptung: $\exists c > 0 : T(n) = c \cdot n \; \forall n > 0$.
Es gilt:

$$T(n) = T(n-1) + O(1)$$
$$T(1) = c_1$$

Induktionsanfang: $T(1) = c_1$. Behauptung erfüllt für jedes $c > c_1$.
Induktionsvoraussetzung: Behauptung gilt für $1, 2, \ldots, n-1$.

Induktionsschluss: Dies zeigt, dass die Behauptung auch für n gilt:

$$\begin{aligned} T(n) &= T(n-1) + O(1) \\ &< c \cdot (n-1) + c_2 \qquad \text{(Induktionsvoraussetzung)} \\ &= c \cdot n + (c_2 - c) \\ &< c \cdot n \qquad \text{, falls} c > c_2\text{, dann ist} c_2 - c < 0. \end{aligned}$$

Mit einer Konstante $c = \max(c_1, c_2)$ ist die Behauptung erfüllt.

Aufgabe 15.4

(a) siehe Quellcode 15.1

(b) siehe Quellcode 15.1

(c) siehe Quellcode 15.1

(d) siehe Quellcode 15.1

```java
package task1304;

public class MinSort {

	/**
	 * Liefert den Index des niedrigsten Wertes.
	 */
	static int getMinIndex(int[] values, int endIndex) {
		int minIndex = 0;

		for (int i = 1; i <= endIndex; i++) {
			if (values[i] < values[minIndex]) {
				minIndex = i;
			}
		}
		int mem = values[endIndex];

		values[endIndex] = values[minIndex];
		values[minIndex] = mem;

		return endIndex;
	}

	/**
	 * Sortiert mit einem Hilfsarray.
	 */
	static void minSort(int[] values, int[] solution) {
		for (int i = values.length - 1; i >= 0; i--) {
			int minIndex = getMinIndex(values, i);

			solution[i] = values[minIndex];

		}
	}

	private int zahl(int a, int b, int m, int i) {
		return (((a * i) + b) % m);
	}

	public static void main(String[] args) {
		int x = 1731673;
		int y = 139187;
		int m = 51898279;
		int n = 1000;
		int[] a = new int[n];
		int[] b = new int[a.length];

		MinSort application = new MinSort();

		for(int i = 0; i < a.length; i++) {
			a[i] = application.zahl(x, y, m, i);
		}

		minSort(a, b);
		for (int i = 0; i < b.length; i++) {
			System.out.println(b[i]);
		}
	}
}
```

Quellcode 15.1: *MinSort.java*

Kapitel 16

Mengen

Aufgabe 16.1

Die Bedingung ist vier Mal erfüllt.

Aufgabe 16.2

Der neue Algorithmus `entferne(s)` (s. Pseudocode 16.1) muss nun für jedes *s* das gesamte Array durchsuchen.

```
1  Algorithmus entferne(s) {
2          i := 0;
3
4          Solange i < fuellstand, {
5                  Wenn S[i] = s, dann {
6                          S[i] := S[fuellstand - 1];
7                          fuellstand := fuellstand - 1;
8                  }
9                  Wenn S[i] nicht = s, dann {
10                         i := i + 1;
11                 }
12         }
13 }
```

Pseudocode 16.1: *entferne(s).*

Aufgabe 16.3

(a) Klasse UnsortArraySetAdmin, s. Quellcode 16.2

(b) Klasse SetAdminProgramm, s. Quellcode 16.3

Aufgabe 16.4

Suche nach 21: Nicht gefunden - Tabelle 16.1 Suche nach 4: Gefunden - Tabelle 16.2

Suche im Intervall 0–10:	3	4	7	8	9	10	11	13	15	18	19
Suche im Intervall 6–10:							11	13	15	18	19
Suche im Intervall 9–10:										18	19
Suche im Intervall 10–10:											19

Tabelle 16.1: *Binäre Suche*

Suche im Intervall 0–10:	3	4	7	8	9	10	11	13	15	18	19
Suche im Intervall 0–4:	3	4	7	8	9						
Suche im Intervall 0–1:	3	4									
Suche im Intervall 1–1:		4									

Tabelle 16.2: *Binäre Suche*

```java
package task1403;

public class UnsortArraySetAdmin {

	private int fuellstand;
	private int[] menge;

	public UnsortArraySetAdmin(int maxn) {
		this.fuellstand = 0;
		this.menge = new int[maxn];
	}

	public boolean suche(int s) {
		for (int i = 0; i < this.fuellstand; i++) {
			if (this.menge[i] == s) {
				return true;
			}
		}
		return false;
	}

	public boolean fuegeEin(int s) {
		if (this.fuellstand >= this.menge.length) {
			return false;
		} else if (this.suche(s)) {
			return false;
		} else {
			this.menge[this.fuellstand] = s;
			this.fuellstand++;
			return true;
		}
	}

	public boolean entferne(int s) {
		for (int i = 0; i < this.fuellstand; i++) {
			if (this.menge[i] == s) {
				this.menge[i] = this.menge[this.fuellstand -
					1];
				this.fuellstand--;
				return true;
			}
		}
		return false;
	}
}
```

Quellcode 16.2: *UnsortArraySetAdmin.java*

```
1  package task1403;
2
3  public class SetAdminProgramm {
4
5
6          public static void main(String[] args) {
7                  int length = 10;
8                  UnsortArraySetAdmin eineMenge = new UnsortArraySetAdmin(
                       length);
9
10                 // Alle geraden Elemente werden eingefuegt.
11                 for (int i = 0; i < length; i++) {
12                         if (i % 2 == 0) {
13                                 eineMenge.fuegeEin(i);
14                         }
15                 }
16
17                 // Danach werden alle Elemente gesucht.
18                 for (int i = 0; i < length; i++) {
19                         if (eineMenge.suche(i)) {
20                                 System.out.println(i + ": gefunden.");
21                         } else {
22                                 System.out.println(i + ": nicht gefunden.");
23                         }
24                 }
25         }
26 }
```

Quellcode 16.3: *SetAdminProgramm.java*

Aufgabe 16.5

siehe Pseudocode 16.4

```
1 j := fuellstand - 1;
2
3 Solange j > i : {
4         S[j] := S[j - 1]
5         j := j - 1;
6 }
```

Pseudocode 16.4: *verschiebe().*

Aufgabe 16.6

Die Werte für den Algorithmus entferne finden sich in Tabelle 16.3.

Durchlauf	j	Array
0	6	3,4,7,8,9,10,11,13,15,18,19
1	7	3,4,7,8,9,10,13,13,15,18,19
2	8	3,4,7,8,9,10,13,15,15,18,19
3	9	3,4,7,8,9,10,13,15,18,18,19
4	10	3,4,7,8,9,10,11,13,15,19,19

Tabelle 16.3: *Der Algorithmus* entferne.

Aufgabe 16.7

(a) s. Quellcode 16.5

(b) s. Quellcode 16.5

Aufgabe 16.8

Die folgenden Knoten werden für die einzelnen Teilaufgaben durchlaufen:

(a) $10 \rightarrow 7 \rightarrow 9 \rightarrow 8$.

(b) $10 \rightarrow 7$.

(c) $10 \rightarrow 7 \rightarrow 4 \rightarrow 3 \rightarrow$ nil.

(d) $10 \rightarrow 15 \rightarrow 19 \rightarrow 18$.

Aufgabe 16.9

Ein ausgeglichener binärer Baum:

Abbildung 16.1: *Der binäre Suchbaum zur Aufgabe 14.9.*

Aufgabe 16.10

Ein binärer Suchbaum, in dem die Suchzeit im ungünstigsten Fall etwa gleich der Anzahl der gegebenen Zahlen, also $O(n)$, ist, hat die Gestalt einer sortierten linearen Liste. Der zugehörige Baum entspricht der Abbildung 16.2.

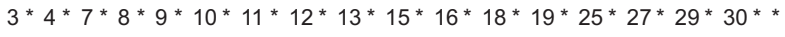


Abbildung 16.2: *Der binäre Suchbaum zur Aufgabe 14.10.*

Aufgabe 16.11

(a) s. Quellcode 16.6

(b) Um den Baum aus Abbildung 14.10 nachzubilden, können beide Konstruktoren verschachtelt und der Baum direkt erstellt werden (s. Quellcode 16.7).

Aufgabe 16.12

(a) $i(79) = 79 \mod 14 = 9$

(b) $i(178) = 178 \mod 14 = 10$

Aufgabe 16.13

(a) $i(140) = 140 \mod 14 = 0$. Durch lineares Sondieren wird somit Position 0 im Array erreicht.

(b) $i(79) = 79 \mod 14 = 9$. Position 9 ist noch frei. Demzufolge ist kein Sondieren erforderlich.

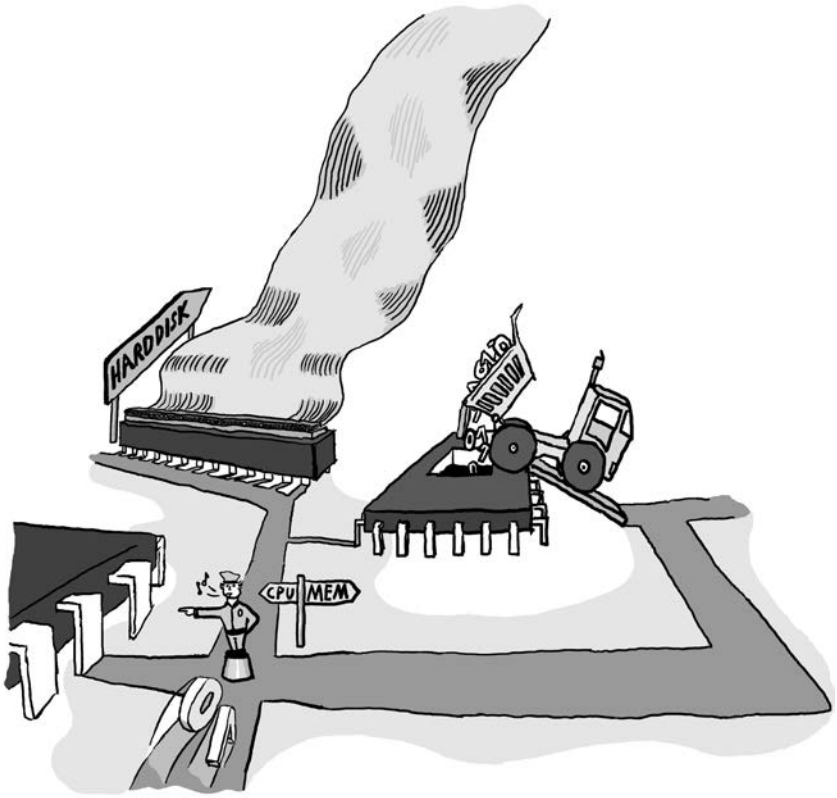
```java
package task1407;

public class SortArraySetAdmin {

	private int fuellstand;
	private int[] menge;

	public SortArraySetAdmin(int maxn) {
		this.fuellstand = 0;
		this.menge = new int[maxn];
	}

	public boolean suche(int s, int l, int r) {
		if (l >= r) {
			return false;
		} else {
			int m = (l + r) / 2;
			if (this.menge[m] == s) {
				return true;
			} else if (this.menge[m] < s) {
				this.suche(s, m + 1, r);
			} else {
				this.suche(s, l, m - 1);
			}
		}
		return false;
	}

	public boolean suche(int s) {
		return this.suche(s, 0, this.fuellstand);
	}

	public boolean fuegeEin(int s) {
		if (this.fuellstand == this.menge.length) {
			return false;
		} else if (!this.suche(s)) {
			this.menge[this.fuellstand - 1] = s;
			this.fuellstand++;
			return true;
		}
		return false;
	}

	public boolean entferne(int s) {
		if (this.suche(s)) {
			for (int i = 0; i < this.menge.length; i++) {
				if (this.menge[i] == s) {
					this.trimArray(i);
					return true;
				}
			}
		}
		return false;
	}

	/**
	 * Loescht eine Position aus der Menge und laesst die weiteren
	     Elemente
	 * nachruecken.
	 */
	private void trimArray(int position) {
		for (int i = position; i < this.menge.length; i++) {
			this.menge[i] = this.menge[i + 1];
		}
		this.fuellstand--;
	}
```

```java
package task1411;

public class Baum {

	private Object content;

	private Baum leftNode;
	private Baum rightNode;


	public Baum(Object o, Baum l, Baum r) {
		this.content = o;
		this.leftNode = l;
		this.rightNode = r;
	}


	public Baum(Object o) {
		this(o, null, null);
	}

	public void setContent(Object o) {
		this.content = o;
	}

	public void setLeftNode(Baum b) {
		this.leftNode = b;
	}

	public void setRightNode(Baum b) {
		this.rightNode = b;
	}
}
```

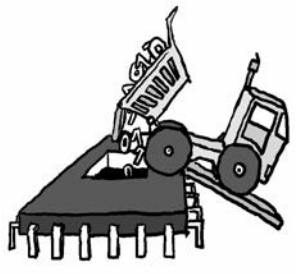
Quellcode 16.6: *Baum.java*

```
1 package task1411;
2
3 public class TestBaum {
4
5         public static void main(String[] args) {
6                 @SuppressWarnings("unused")
7                 Baum dBaum = new Baum("Referenz_1", new Baum("Referenz_2"),
                      new Baum("Referenz_3"));
8         }
9 }
```

Quellcode 16.7: *TestBaum.java*



Vom Programm zum Rechner



Kapitel 17

Hardware und Programmierung

Zu diesem Kapitel liegen keine Übungsaufgaben vor.

Kapitel 18

Rechnerarchitektur und Maschinensprache

Aufgabe 18.1

(a) Die wesentlichen Komponenten der von-Neumann-Rechnerarchitektur sind:

- **Speicher**,
- **Prozessor** und
- **Ein-/Ausgabe**.

Diese Komponenten sind durch Kommunikationswege verbunden. Für diese Kommunikationswege gibt es wieder zwei verschiedene Architekturen:

- Die Einzelverbindungsarchitektur und
- die Busarchitektur.

(b) Das Programm befindet sich bei der von-Neumann-Rechnerarchitektur im Speicher.

Aufgabe 18.2

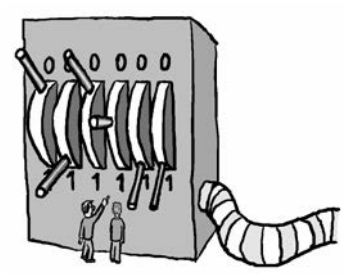
(a) In diesem Beispielprogramm wird r_2 zehn Mal um 2 erhöht. Nach der Ausführung des Programms steht in Register a_{10} eine 20.

(b) Das Maschinenprogramm, das die Summe von 1–n ausrechnet, könnte so aussehen:

setze r_1, 1;
setze r_2, n;
setze r_3, 0;
X addiere r_3, r_3, r_2;
subtrahiere r_2, r_2, r_1;

springe X, r_2;
speichere a_{10}, r_3;

Als Alternativlösung siehe Gleichung 3.1 auf Seite 21.



Kapitel 19

Schaltungen

Aufgabe 19.1

(a)

$$\begin{aligned}(11011)_2 &= (1\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0)_{10}\\ &= (27)_{10}\end{aligned}$$

(b)

$$\begin{aligned}(3021)_4 &= (3\cdot 4^3 + 0\cdot 4^2 + 2\cdot 4^1 + 1\cdot 4^0)_{10}\\ &= (201)_{10}\end{aligned}$$

(c)

$$(29)_{10} = \begin{cases}(11101)_p & \text{für } p = 2,\\ (35)_p & \text{für } p = 8,\\ (1\mathrm{D})_p & \text{für } p = 16.\end{cases}$$

Aufgabe 19.2

s. Quellcode 19.1

Aufgabe 19.3

siehe Tabelle 19.1.

```java
package task1702;

public class Programm {


	public static void main(String[] args) {
		System.out.println(Programm.stelle(2, 1, 4));
	}

	/**
	 *
	 * i: gesuchte Stelle
	 * p: Basis
	 * x: Eingabezahl
	 * rueckgabe: Wert der i-ten Stelle in der p-adischen Form von x
	 */
	public static int stelle(int i, int p, int x) {
		if (x > 0) {
			return (int) ((x / Math.pow((double) p, (double) i))
				% i);
		} else {
			return -1;
		}
	}
}
```

Quellcode 19.1: *Programm.java*

a	b	c	$g(a,b,c)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Tabelle 19.1: *Die Wertetabelle von $g(a,b,c)$.*

Aufgabe 19.4

Die KNF, die der Wertetabelle 19.1 entnommen werden, lautet:

$$(\neg a \wedge \neg b \wedge \neg c) \vee (\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge c) \vee (a \wedge b \wedge \neg c) \vee (a \wedge b \wedge c)$$

Aufgabe 19.5

x	y	$A = x \wedge \neg y$	$B = \neg x \wedge y$	$A \vee B$
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

Tabelle 19.2: *Die Funktionstabelle für* $h(x,y) = (x \wedge \neg y) \vee (\neg x \wedge y)$.

Wie aus Tabelle 19.2 entnehmbar ist, liefert diese Funktion `true` zurück wenn *entweder x oder y* `true` ist. Diese Funktion nennt man *XOR*-Funktion.

Aufgabe 19.6

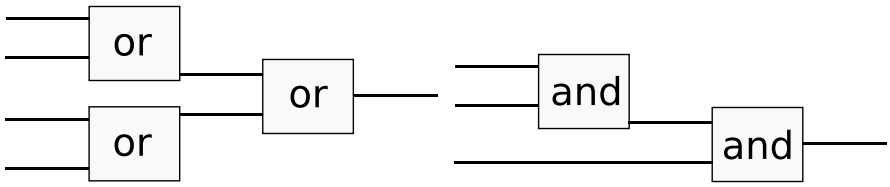
Die Funktionstabelle für die Funktion $f(x,y,z,w) = (x+y)+(z+w)$.

x	y	z	w	$A = x+y$	$B = z+w$	$A+B$
0	0	0	0	0	0	0
0	0	0	1	0	1	1
0	0	1	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	1	1	1
0	1	1	0	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	0	1
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	0	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

Tabelle 19.3: *Funktionstabelle*

Aufgabe 19.7

s. Abbildung 19.1



(a) *OR-Gatter mit fan-in 4* **(b)** *AND-Gatter mit fan-in 3*

Abbildung 19.1: *Logische Gatter*

Aufgabe 19.8

s. Abbildung 19.2

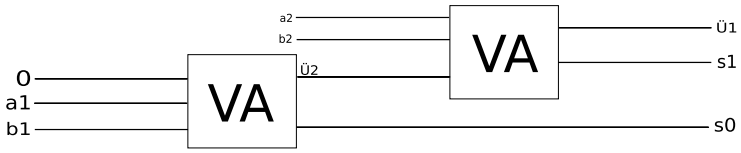


Abbildung 19.2: *VA für 2-stellige Binärzahlen.*

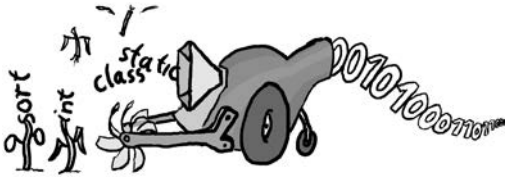
Aufgabe 19.9

s. Abbildung 19.4 Wenn beide Werte 0 sind, ist der Ausgangswert 1. Andernfalls ist der Aus-

Zustand	a	b_{alt}	b_{neu}
0	0	0	1
1	0	1	0
2	1	0	0
3	1	1	0

Tabelle 19.4: *Die Wertetabelle für Aufgabe 17.9.*

gangswert stets 0.



Kapitel 20

Formale Sprachen und Compiler

Aufgabe 20.1

Die Worte, die mit dieser Sprache gebildet werden können, haben für alle $n \in \mathbb{N}$ die Gestalt:

$$\underbrace{a \ldots a}_{n\text{-mal}}\underbrace{b \ldots b}_{n\text{-mal}}$$

Aufgabe 20.2

Mit dieser Sprache lassen sich beliebig viele a, gefolgt von einem ba, gefolgt von beliebig vielen b, bilden.

Aufgabe 20.3

Folgende Grammatik G notwendig, um diese Sprache zu erstellen:

$$G = (N, T, P, S)$$

mit

$$\begin{aligned} N &= \{S\} \\ T &= \{a, b, c\} \\ P &= \{S \to c, S \to aSb\} \\ S &= \{S\} \end{aligned}$$

Aufgabe 20.4

- Die Grammatik ist eine Chomsky-Typ-2-Grammatik.

- Die Grammatik ist eine Chomsky-Typ-3-Grammatik.

Aufgabe 20.5

```
hole r1, a1;
hole r2, a2;
multipliziere r1, r1, r2;
hole r2, a3;
addiere r1, r1, r2;
```

Aufgabe 20.6

(a) Der Zustandsübergangsgraph

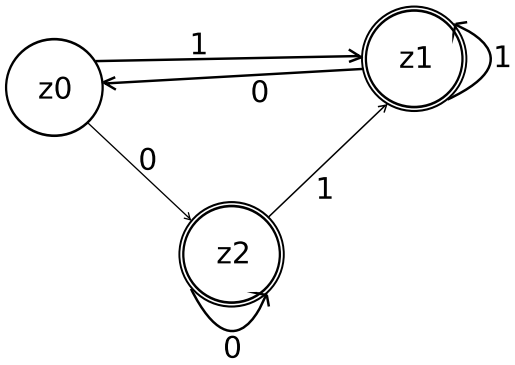


Abbildung 20.1: *Zustandsübergangsgraph.*

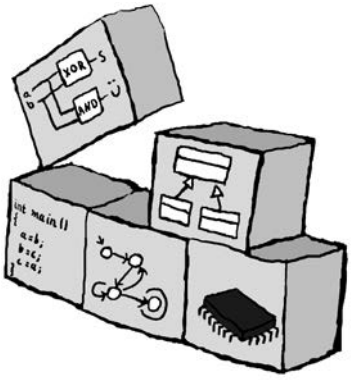
(b)
- `01100` wird akzeptiert.
- `11110` wird nicht akzeptiert.
- `001100` wird akzeptiert.
- `010101` wird akzeptiert.
- `01010` wird nicht akzeptiert.

(c) Die akzeptierte Sprache des Automaten sind alle Werte, die nicht auf `10` enden.

$G = \{N, T, P, S\}$ mit

$T = \{0, 1\}$
$N = \{X, Y\}$
$P = \{X \to 1Y, 0X, 0, 1; Y \to 1Y, 0X\}$
$S = \{X\}$

Anhang



Kapitel A

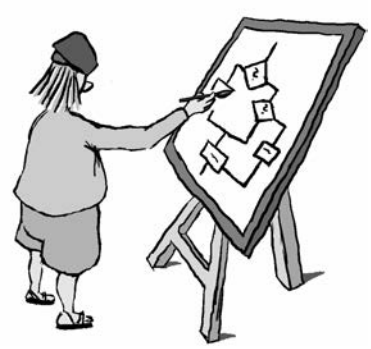
Schlüsselwörter im Sprachumfang von Java

Zu diesem Kapitel liegen keine Übungsaufgaben vor.

Kapitel **B**

Grundlagen der Java-Programmierumgebung

Zu diesem Kapitel liegen keine Übungsaufgaben vor.



Kapitel C

Literaturverzeichnis

Zu diesem Kapitel liegen keine Übungsaufgaben vor.