

Hochschule -

Fakultät IV – Technische Informatik

Modul: Programmieren 1

Prüfer: -

Semester: 1

Entwicklungsarbeit

von

Sebastian Schramm Matrikel-Nr. -

14. März 2021

Inhaltsverzeichnis

1	Kapitel 1	5
1.1	Aufgabenstellung	5
1.2	Anforderungsdefinition	5
1.3	Entwurf	5
1.4	Quellcode	5
1.4.1	Main.java	5
1.5	Testdokumentation	5
1.6	Benutzungshinweise	5
1.7	Anwendungsbeispiel	6
2	Kapitel 2	6
2.1	Entwurf	6
3	Kapitel 3	6
3.1	Teilaufgabe 1	6
3.1.1	Aufgabenstellung	6
3.1.2	Anforderungsdefinition	6
3.1.3	Entwurf	6
3.1.4	Quelltext	7
3.1.4.1	Typkonvertierungen.java	7
3.1.5	Testdokumentation	10
3.1.6	Benutzungshinweise	10
3.1.7	Anwendungsbeispiel	10
3.2	Teilaufgabe 2	12
3.2.1	Aufgabenstellung	12
3.2.2	Anforderungsdefinition	12
3.2.3	Entwurf	12
3.2.4	Quelltext	12
3.2.4.1	Wertebereiche.java	12
3.2.5	Testdokumentation	13
3.2.6	Benutzungshinweise	13
3.2.7	Anwendungsbeispiel	13
4	Kapitel 4	13
4.1	Teilaufgabe 1	13
4.1.1	Aufgabenstellung	13
4.1.2	Anforderungsdefinition	13
4.1.3	Entwurf	14
4.1.4	Quellcode	14
4.1.4.1	Referenzen.java	14
4.1.4.2	Punkt.java	15
4.1.5	Testdokumentation	15
4.1.6	Benutzungshinweise	15
4.1.7	Anwendungsbeispiel	15
4.2	Teilaufgabe 2	16
4.2.1	Aufgabenstellung	16
4.2.2	Anforderungsdefinition	16
4.2.3	Entwurf	16
4.2.4	Quellcode	17
4.2.4.1	Matrizen.java	17
4.2.5	Testdokumentation	18
4.2.6	Benutzungshinweise	19
4.2.7	Anwendungsbeispiel	19
5	Kapitel 5	19
5.1	Teilaufgabe 1	19
5.1.1	Aufgabenstellung	19

5.1.2	Anforderungsdefinition	19
5.1.3	Entwurf	20
5.1.4	Quelltext	20
5.1.4.1	Nebeneffekte.java	20
5.1.5	Testdokumentation	21
5.1.6	Benutzungshinweise	21
5.1.7	Anwendungsbeispiel	21
5.2	Teilaufgabe 2	21
5.2.1	Aufgabenstellung	21
5.2.2	Anforderungsdefinition	21
5.2.3	Entwurf	21
5.2.4	Quelltext	21
5.2.4.1	Operatoren.java	21
5.2.5	Testdokumentation	23
5.2.6	Benutzungshinweise	23
5.2.7	Anwendungsbeispiel	23
6	Kapitel 6	25
6.1	Teilaufgabe 1	25
6.1.1	Aufgabenstellung	25
6.1.2	Anforderungsdefinition	25
6.1.3	Entwurf	25
6.1.4	Quelltext	27
6.1.4.1	Matrizen.java	27
6.1.5	Testdokumentation	30
6.1.6	Benutzungshinweise	30
6.1.7	Anwendungsbeispiel	30
6.2	Teilaufgabe 2	31
6.2.1	Aufgabenstellung	31
6.2.2	Anforderungsdefinition	31
6.2.3	Entwurf	31
6.2.4	Quelltext	32
6.2.4.1	Sprunganweisungen.java	32
6.2.5	Testdokumentation	33
6.2.6	Benutzungshinweise	33
6.2.7	Anwendungsbeispiel	33
7	Kapitel 7	34
7.1	Aufgabenstellung	34
7.2	Anforderungsdefinition	34
7.3	Entwurf	34
7.4	Quelltext	36
7.4.1	Main.java	36
7.4.2	Viereck.java	36
7.4.3	KonvexesViereck.java	36
7.4.4	Trapez.java	39
7.4.5	Parallelogramm.java	40
7.4.6	Rhombus.java	41
7.4.7	Rechteck.java	41
7.4.8	Quadrat.java	41
7.5	Testdokumentation	42
7.6	Anwendungsbeispiel	42
8	Kapitel 8	42
8.1	Aufgabenstellung	42
8.2	Anforderungsdefinition	42
8.3	Entwurf	42
8.4	Quelltext	43
8.4.1	Main.java	43

8.4.2	GeradeZahl.java	43
8.4.3	OddException.java	44
8.5	Testdokumentation	44
8.6	Benutzungshinweise	44
8.7	Anwendungsbeispiel	44
9	Kapitel 10	45
9.1	Aufgabenstellung	45
9.2	Anforderungsdefinition	45
9.3	Entwurf	45
9.4	Quelltext	45
9.4.1	Main.java	45
9.4.2	IO.java	47
9.4.3	MyException.java	50
9.5	Testdokumentation	50
9.6	Benutzungshinweise	51
9.7	Anwendungsbeispiel	51

1 Kapitel 1

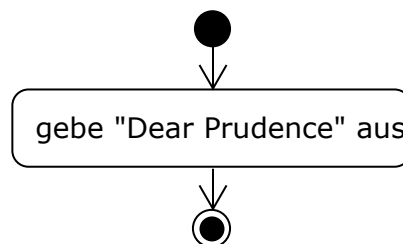
1.1 Aufgabenstellung

Um uns mit Java und der Konsole vertraut zu machen sollten wir zuerst ein Java Programm in der Konsole schreiben, welches "Dear Prudence" ausgibt. Anschließend mit Javac Kompilieren und mit Java ausführen. Wenn wir dies geschafft haben, sollte wir als nächstes in NetBeans ein neues Projekt erstellen und dort das selbe Programm erneut Programmieren.

1.2 Anforderungsdefinition

1. Das Programm soll "Dear Prudence" auf der Konsole ausgeben.

1.3 Entwurf



1.4 Quellcode

1.4.1 Main.java

```
1 package chapter_01;
2
3 /**
4  * Klasse mit der Main-Methode
5  * @author sebastian
6  *
7  */
8 public class Main {
9
10     /**
11      * Die Main Methode
12      * Gibt "Dear Prudence" aus
13      * @param args
14      */
15     public static void main(String[] args) {
16         System.out.println("Dear Prudence");
17     }
18 }
```

1.5 Testdokumentation

Wenn das Programm gestartet wird, sollte "Dear Prudence" auf der Konsole ausgegeben werden. Dies war der Fall.

1.6 Benutzungshinweise

Navigieren Sie in der Kommandozeile zum Ordner, wo sich die Java Datei befindet. Danach führen sie "javac Main.java" auf. Jetzt können Sie das Programm mit "java main" starten. In der Konsole sollte nun "Dear Prudence" angezeigt werden.

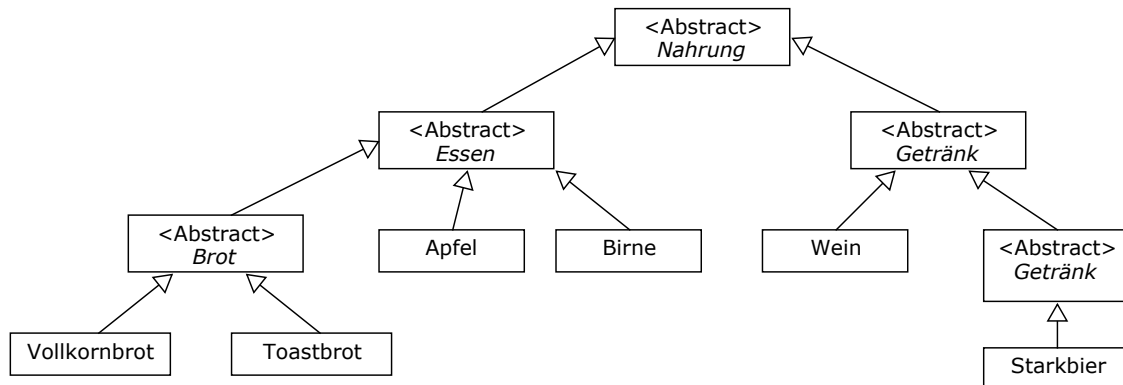
1.7 Anwendungsbeispiel

Nach dem Aufruf von java Main, sollten wir folgendes sehen:

```
1 [sebastian@laptop bin]$ java Main
2 Dear Prudence
3 [sebastian@laptop bin]$
```

2 Kapitel 2

2.1 Entwurf



3 Kapitel 3

3.1 Teilaufgabe 1

3.1.1 Aufgabenstellung

In der ersten Teilaufgabe sollten wir uns mit der Typkonvertierung befassen. Dafür schreiben wir ein kleines Programm, welches die primitiven Datentypen erweiternd und einschränkend Konvertiert.

3.1.2 Anforderungsdefinition

1. Zu jedem Primitiven Datentypen eine erweiternde und einschränkende Konvertierung durchführen.

3.1.3 Entwurf

Für die Aufgabe habe ich mich entschieden für jeden Primitive Datentyp eine Methode zu entwerfen, in der letztendlich die Konvertierung für den entsprechenden Datentyp stattfinden wird.

Typkonvertierung
<pre> +main() static -convertByte(byte: _byte) static -convertShort(short: _short) static -convertInt(int: _int) static -convertLong(long: _long) static -convertChar(char: _char) static -convertFloat(float: _float) static -convertDouble(double: _double) static </pre>

3.1.4 Quelltext

3.1.4.1 Typkonvertierungen.java

```
1 package chapter_03;
2
3 /**
4  * Klasse mit der Main-Methode
5  * und der einzelnen Typkonvertierungen
6  * @author Sebastian
7  */
8 public class Typkonvertierungen {
9
10     public static void main(String[] args) {
11         /*
12          * Rund die einzelnen Methoden auf, mit entsprechenden Werten
13          */
14         convertByte((byte) -128);
15         convertShort((short) 34);
16         convertInt(98987);
17         convertLong(987987987);
18
19         convertChar('a');
20
21         convertFloat(15.0f);
22         convertDouble(1.7976931348623157E308);
23     }
24
25     /**
26      * Eine erweiternde Konvertierung von Byte zu Double
27      * @param _byte
28      */
29     private static void convertByte(byte _byte) {
30         short newShort = _byte;
31         int newInt = _byte;
32         long newLong = _byte;
33         float newFloat = _byte;
34         double newDouble = _byte;
35
36         System.out.println("-----");
37         System.out.println("Byte erweiternd");
38         System.out.println("Byte   " + _byte);
39         System.out.println("Short  " + newShort);
40         System.out.println("Int    " + newInt);
41         System.out.println("Long   " + newLong);
42         System.out.println("Float  " + newFloat);
43         System.out.println("Double " + newDouble);
44         System.out.println("\nChar   " + (char) newInt); //Char wird hier separat
45             ausgegeben
46         System.out.println("-----");
47     }
48
49     /**
50      * Eine einschränkende Konvertierung von Short zu Byte
51      * Eine erweiternde Konvertierung von Short zu Double
52      * @param _short
53      */
54     private static void convertShort(short _short) {
55         byte newByte = (byte) _short;
56         int newInt = _short;
57         long newLong = _short;
58         float newFloat = _short;
59         double newDouble = _short;
```

```
60     System.out.println("Short einschränkend");
61     System.out.println("Short " + _short);
62     System.out.println("Byte " + newByte);
63
64     System.out.println("Short erweiternd");
65     System.out.println("Short " + _short);
66     System.out.println("Int " + newInt);
67     System.out.println("Long " + newLong);
68     System.out.println("Float " + newFloat);
69     System.out.println("Double " + newDouble);
70     System.out.println("\nChar " + (char) newInt); //Char wird hier separat
        ausgegeben
71     System.out.println("-----");
72 }
73
74 /**
75  * Eine einschränkende Konvertierung von Int zu Byte
76  * Eine erweiternde Konvertierung von Int zu Double
77  * @param _int
78  */
79 private static void convertInt(int _int) {
80     short newShort = (short) _int;
81     byte newByte = (byte) _int ;
82
83     long newLong = _int;
84     float newFloat = _int;
85     double newDouble = _int;
86
87     System.out.println("Int einschränkend");
88     System.out.println("Int " + _int);
89     System.out.println("Short " + newShort);
90     System.out.println("Byte " + newByte);
91
92     System.out.println("Int erweiternd");
93     System.out.println("Int " + _int);
94     System.out.println("Long " + newLong);
95     System.out.println("Float " + newFloat);
96     System.out.println("Double " + newDouble);
97     System.out.println("\nChar " + (char) _int); //Char wird hier separat
        ausgegeben
98     System.out.println("-----");
99 }
100
101 /**
102  * Eine einschränkende Konvertierung von Long zu Byte
103  * Eine erweiternde Konvertierung von Long zu Double
104  * @param _long
105  */
106 private static void convertLong(long _long) {
107     int newInt = (int) _long;
108     short newShort = (short) _long;
109     byte newByte = (byte) _long;
110
111     float newFloat = _long;
112     double newDouble = _long;
113
114     System.out.println("Long einschränkend");
115     System.out.println("Long " + _long);
116     System.out.println("Int " + newInt);
117     System.out.println("Short " + newShort);
118     System.out.println("Byte " + newByte);
119
120     System.out.println("Long erweiternd");
```



```

121     System.out.println("Long   " + _long);
122     System.out.println("Float  " + newFloat);
123     System.out.println("Double " + newDouble);
124     System.out.println("\nChar   " + (char) newInt); //Char wird hier separat
        ausgegeben
125     System.out.println("-----");
126 }
127
128 /**
129  * Eine einschränkende Konvertierung von Char zu Byte
130  * Eine erweiternde Konvertierung von Char zu Double
131  * @param _char
132  */
133 private static void convertChar(char _char) {
134     int newInt = _char;
135     short newShort = (short) _char;
136     byte newByte = (byte) _char;
137
138     long newLong = _char;
139     float newFloat = _char;
140     double newDouble = _char;
141
142     System.out.println("Char einschränkend");
143     System.out.println("Char   " + _char);
144     System.out.println("Long   " + newLong);
145     System.out.println("Int     " + newInt);
146     System.out.println("Short  " + newShort);
147     System.out.println("Byte   " + newByte);
148
149     System.out.println("Char erweiternd");
150     System.out.println("Char   " + _char);
151     System.out.println("Long   " + newLong);
152     System.out.println("Float  " + newFloat);
153     System.out.println("Double " + newDouble);
154     System.out.println("-----");
155 }
156
157 /**
158  * Eine einschränkende Konvertierung von Float zu Byte
159  * Eine erweiternde Konvertierung von Float zu Double
160  * @param _float
161  */
162 private static void convertFloat(float _float) {
163     long newLong = (long) _float;
164     int newInt = (int) _float;
165     short newShort = (short) _float;
166     byte newByte = (byte) _float;
167
168     double newDouble = _float;
169
170     System.out.println("Float einschränkend");
171     System.out.println("Float   " + _float);
172     System.out.println("Long   " + newLong);
173     System.out.println("Int     " + newInt);
174     System.out.println("Short  " + newShort);
175     System.out.println("Byte   " + newByte);
176
177     System.out.println("Float erweiternd");
178     System.out.println("Float   " + _float);
179     System.out.println("Double " + newDouble);
180     System.out.println("\nChar   " + (char) newInt); //Char wird hier separat
        ausgegeben
181     System.out.println("-----");

```

```

182     }
183
184     /**
185      * Eine einschränkende Konvertierung von Double zu Byte
186      * @param _double
187      */
188     private static void convertDouble(double _double) {
189         float newFloat = (float) _double;
190         long newLong = (long) _double;
191         int newInt = (int) _double;
192         short newShort = (short) _double;
193         byte newByte = (byte) _double;
194
195         System.out.println("Double einschränkend");
196         System.out.println("Double " + _double);
197         System.out.println("Float " + newFloat);
198         System.out.println("Long " + newLong);
199         System.out.println("Int " + newInt);
200         System.out.println("Short " + newShort);
201         System.out.println("Byte " + newByte);
202         System.out.println("\nChar " + (char) newInt); //Char wird hier separat
                ausgegeben
203         System.out.println("-----");
204     }
205
206 }

```

3.1.5 Testdokumentation

Nach den Aufruf des Programms, sollten alle Typkonvertierungen auf der Konsole ausgegeben werden. Dies ist auch geschehen.

3.1.6 Benutzungshinweise

Keine Besonderen Benutzungshinweise. Man navigiere zu dem Ordner von sich die Compilierte Datei mit dem Namen "Typkonvertierungen.class" befindet und führt anschließend `java Typkonvertierungen` aus.

3.1.7 Anwendungsbeispiel

Nach dem man das Programm gestartet hat (aufgrund der Formatierung, werden einige Zeichen bei Char nicht dargestellt), sollte folgende Ausgabe erscheinen:

```

1 [sebastian@laptop bin]$ java Typkonvertierungen
2 -----
3 Byte erweiternd
4 Byte -128
5 Short -128
6 Int -128
7 Long -128
8 Float -128.0
9 Double -128.0
10
11 Char
12 -----
13 Short einschränkend
14 Short 34
15 Byte 34
16 Short erweiternd
17 Short 34
18 Int 34
19 Long 34
20 Float 34.0

```

```
21 Double 34.0
22
23 Char  "
24 -----
25 Int einschränkend
26 Int    98987
27 Short -32085
28 Byte  -85
29 Int erweiternd
30 Int    98987
31 Long   98987
32 Float  98987.0
33 Double 98987.0
34
35 Char
36 -----
37 Long einschränkend
38 Long   987987987
39 Int    987987987
40 Short -32749
41 Byte   19
42 Long erweiternd
43 Long   987987987
44 Float  9.8798797E8
45 Double 9.87987987E8
46
47 Char
48 -----
49 Char einschränkend
50 Char   a
51 Long   97
52 Int    97
53 Short  97
54 Byte   97
55 Char erweiternd
56 Char   a
57 Long   97
58 Float  97.0
59 Double 97.0
60 -----
61 Float einschränkend
62 Float  15.0
63 Long   15
64 Int    15
65 Short  15
66 Byte   15
67 Float erweiternd
68 Float  15.0
69 Double 15.0
70
71 Char
72 -----
73 Double einschränkend
74 Double 1.7976931348623157E308
75 Float  Infinity
76 Long   9223372036854775807
77 Int    2147483647
78 Short  -1
79 Byte   -1
80
81 Char
82 -----
83 [sebastian@laptop bin]$
```

3.2 Teilaufgabe 2

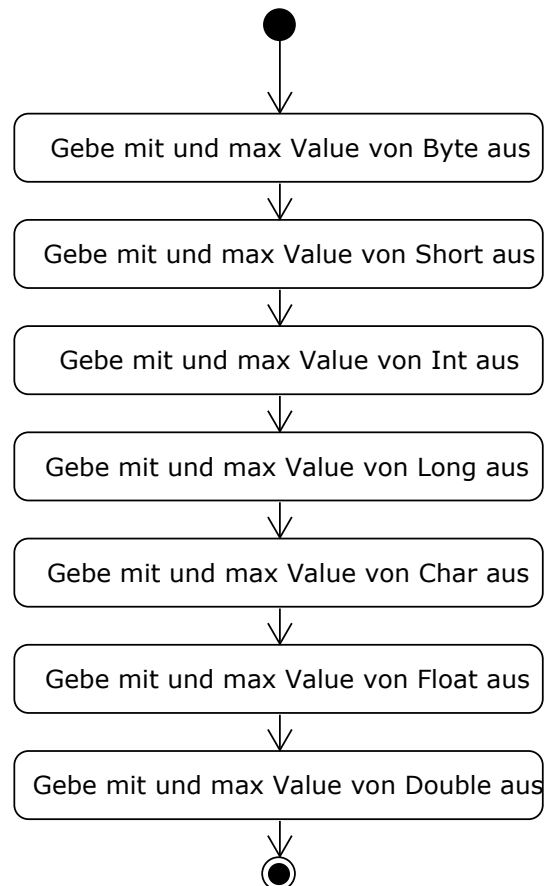
3.2.1 Aufgabenstellung

In der Zweiten Teilaufgabe sollen wir uns mit den Wertebereichen der Primitiven Datentypen befassen und deren Wertebereiche in einem Programm ausgeben.

3.2.2 Anforderungsdefinition

1. Zu jedem primitiven Datentypen den Max und Min-Wert ausgeben.

3.2.3 Entwurf



3.2.4 Quelltext

3.2.4.1 Wertebereiche.java

```
1 package chapter_03;
2
3 /**
4  * Klasse mit der Main-Methode
5  * und gibt die Wertebereiche der primitiven Datentypen aus
6  * @author Sebastian
7  */
8 public class Wertebereiche {
9
10     public static void main(String[] args) {
11         //Min und Max Value von Byte
12         System.out.println("Byte min " + Byte.MIN_VALUE + " | Byte max " + Byte.
13             MAX_VALUE);
```

```

13 //Min und Max Value von Short
14 System.out.println("Short min " + Short.MIN_VALUE + " | Short max " + Short.
    MAX_VALUE);
15 //Min und Max Value von Integer
16 System.out.println("Integer min " + Integer.MIN_VALUE + " | Integer max " +
    Integer.MAX_VALUE);
17 //Min und Max Value von Long
18 System.out.println("Long min " + Long.MIN_VALUE + " | Byte Long " + Long.
    MAX_VALUE);
19
20 //Min und Max Value von Char
21 System.out.println("Char min \u0000 | Char max \uffff");
22
23 //Min und Max Value von Float
24 System.out.println("Float min " + Float.MIN_VALUE + " | Float max " + Float.
    MAX_VALUE);
25 //Min und Max Value von Double
26 System.out.println("Double min " + Double.MIN_VALUE + " | Double max " + Double
    .MAX_VALUE);
27 }
28
29 }

```

3.2.5 Testdokumentation

Nach dem Start des Programms sollten die Min und Max werte der einzelnen Datentypen ausgegeben werden. Anschließend wurden die Werten mit den in der Java Dokumentation stehenden Werten verglichen. Alles hat soweit gestimmt.

3.2.6 Benutzungshinweise

Keine Besonderen Benutzungshinweise. Man navigiere zu dem Ordner von sich die Compilierte Datei mit dem Namen "Wertebereiche.class" befindet und führt anschließend java Wertebereiche aus.

3.2.7 Anwendungsbeispiel

Nach dem man das Programm gestartet hat (aufgrund der Formatierung, werden einige Zeichen bei Char nicht dargestellt), sollte folgende Ausgabe erscheinen:

```

30 [sebastian@laptop bin]$ java Wertebereiche
31 Byte min -128 | Byte max 127
32 Short min -32768 | Short max 32767
33 Integer min -2147483648 | Integer max 2147483647
34 Long min -9223372036854775808 | Byte Long 9223372036854775807
35 Char min  | Char max
36 Float min 1.4E-45 | Float max 3.4028235E38
37 Double min 4.9E-324 | Double max 1.7976931348623157E308
38 [sebastian@laptop bin]$

```

4 Kapitel 4

4.1 Teilaufgabe 1

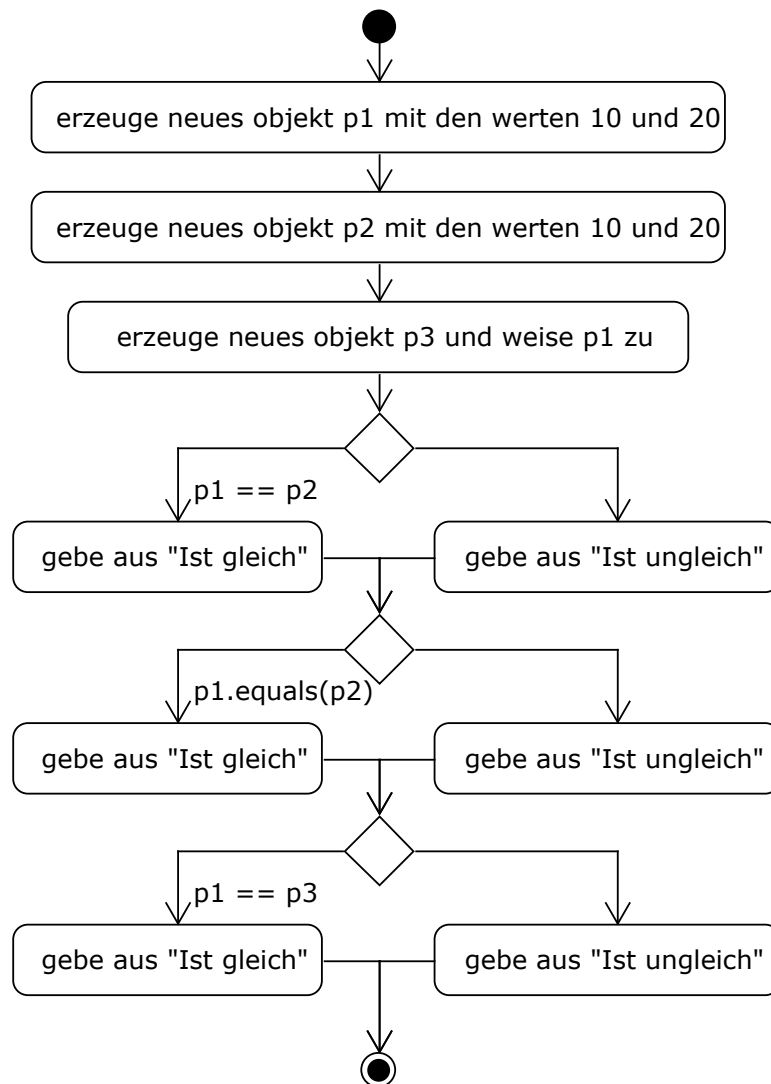
4.1.1 Aufgabenstellung

Wir sollen ein Programm schreiben welches Prüft ob zwei Referenzen gleich sind.

4.1.2 Anforderungsdefinition

1. Prüfe ob zwei Referenzen gleich sind.

4.1.3 Entwurf



4.1.4 Quellcode

4.1.4.1 Referenzen.java

```

1 package chapter_04;
2
3 /**
4  * Klasse mit der Main-Methode
5  * und prüft ob Zwei Referenzen gleich sind
6  * @author Sebastian
7  *
8  */
9 public class Referenzen {
10
11     public static void main(String[] args) {
12         /*
13          * Es werden zwei identische Objekte erzeugt
14          * mit den selben Werten.
15          * Zuletzt wird noch ein drittes erzeugt mit einer
16          * Referenz auf das erste
17          */
18         Punkt p1 = new Punkt(10, 20);

```

```
19 Punkt p2 = new Punkt(10, 20);
20 Punkt p3 = p1;
21
22 //Hier wird geprüft ob p1 und p2 die selbe Adresse hat.
23 if (p1 == p2)
24     System.out.println("Ist gleich");
25 else
26     System.out.println("Ist ungleich");
27
28 //Hier wird geprüft ob der Inhalt der selbe ist
29 if (p1.equals(p2))
30     System.out.println("Ist gleich");
31 else
32     System.out.println("Ist ungleich");
33
34 //Hier wird geprüft ob p3 und p1 gleich sind
35 if (p3 == p1)
36     System.out.println("Ist gleich");
37 else
38     System.out.println("Ist ungleich");
39 }
40
41 }
```

4.1.4.2 Punkt.java

```
1 package chapter_04;
2
3 /**
4  * Punkt Klasse
5  * Hier werden nur Zwei Punkte gespeichert
6  * @author Sebastian
7  *
8  */
9 @SuppressWarnings("unused")
10 public class Punkt {
11     private static int x = 0;
12     private static int y = 0;
13
14     public Punkt(int x, int y) {
15         this.x = x;
16         this.y = y;
17     }
18 }
```

4.1.5 Testdokumentation

Nach dem Start des Programms sollten die ersten beiden Bedingungen falsch sein und die dritte wahr, dies war auch der Fall.

4.1.6 Benutzungshinweise

Navigieren Sie in der Kommandozeile zum dem Ordner, wo sich die Java Datei befindet. Danach führen sie "javac Referenzen.java" auf. Jetzt können Sie das Programm mit "java Referenzen" starten.

4.1.7 Anwendungsbeispiel

Nach dem Aufruf von java Referenzen, sollten wir nun folgendes sehen:

```
1 [sebastian@laptop bin]$ java Referenzen
2 Ist ungleich
3 Ist ungleich
```

```
4 | Ist gleich
5 | [sebastian@laptop bin]$
```

4.2 Teilaufgabe 2

4.2.1 Aufgabenstellung

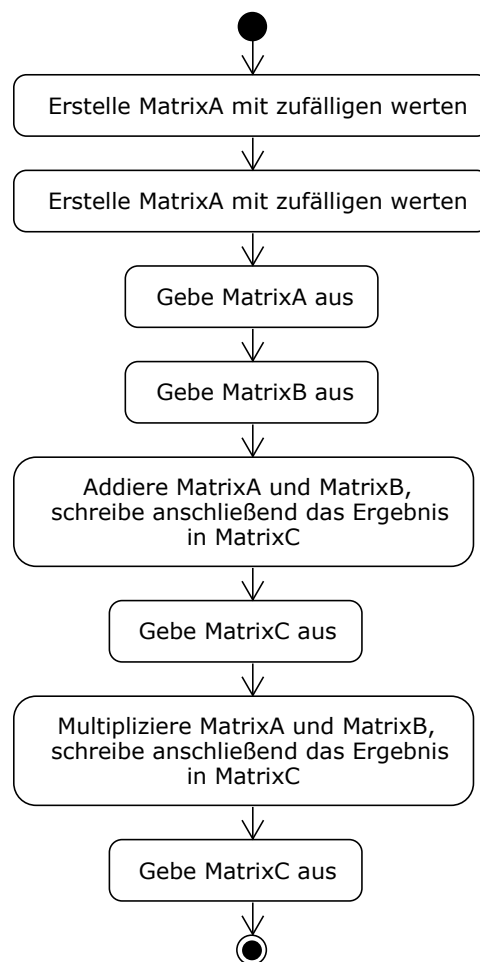
Es soll ein Programm erstellt werden welches zwei 2x2 Matrizen miteinander multiplizieren und addieren kann. Wobei die Elemente der beiden Matrizen bei der Erzeugung zufällig initialisiert werden sollen.

4.2.2 Anforderungsdefinition

1. Addiere zwei 2x2 Matrizen.
2. Multipliziere zwei 2x2 Matrizen.
3. Initialisiere die Matrizen mit zufälligen zahlen.

4.2.3 Entwurf

Es sollen zunächst 2 2x2 Matrizen erzeugt welche anschließend einer Methode übergeben werden, welche die beiden Matrizen mit zufälligen Werten initialisiert. Danach sollen beide Matrizen einmal ausgegeben werden. Nach der Ausgabe werden zwei Methoden aufgerufen, eine für die Multiplikation und eine für die Addition, diese liefern jeweils eine neue Matrix zurück mit den ausgerechneten Werten. Diese Matrix wird als MatrixC Temporär gespeichert und anschließend ausgegeben.



4.2.4 Quellcode

4.2.4.1 Matrizen.java

```
1 package chapter_04;
2
3 import java.util.Random;
4
5 /**
6  * Klasse mit der Main-Methode
7  * Addiert und Multipliziert Matrizen
8  * @author Sebastian
9  *
10 */
11 public class Matrizen {
12
13     public static void main(String[] args) {
14         int matrixA[][];
15         int matrixB[][];
16
17         /*
18          * Initialisierungsmethode wird mit dem Wert n aufgerufen.
19          * Anschliessend wird diese Matrix erzeugt und mit
20          * zufällig generierten Zahlen befüllt.
21          */
22         matrixA = initialize(2);
23         matrixB = initialize(2);
24
25         /*
26          * Zuerst werden die Beiden Matrizen A und B jeweils ausgegeben
27          */
28         System.out.println("Matrix A:");
29         printMatrix(matrixA);
30         System.out.println("Matrix B:");
31         printMatrix(matrixB);
32
33         /*
34          * Anschliessend werden die Matrizen hier Addiert
35          */
36         System.out.println("Addition von A und B:");
37         printMatrix(addition(matrixA, matrixB));
38
39         /*
40          * Und hier Multipliziert
41          */
42         System.out.println("Multiplikation von A und B:");
43         printMatrix(multiplikation(matrixA, matrixB));
44     }
45
46     /**
47      * Initialisierung des Arrays
48      * @param n Die gröSse der nxn Matrix
49      * @return matrix
50      */
51     private static int[][] initialize(int n) {
52         int matrix[][] = new int[n][n];
53
54         /*
55          * Bei der Initialisierung wird einmal durch das gesamt Array dutch iteriert.
56          * Dabei werden dann mit Math.random() zufällige Zahlen rein geschrieben.
57          */
58
59         Random rand = new Random();
60         for (int i = 0; i < matrix.length; ++i)
61             for (int l = 0; l < matrix[i].length; ++l)
62                 matrix[i][l] = rand.nextInt(200) - 100;
```

```

61     return matrix;
62 }
63
64 /**
65  * Addition der beiden Matrizen A und B
66  * @param matrixA
67  * @param matrixB
68  * @return Gibt ein neues Array mit den Addierten Werten zurück
69  */
70 private static int[][] addition(int matrixA[][], int matrixB[][]) {
71     int matrixAd[][] = new int[matrixA.length][matrixA[0].length]; //Es wird ein
72     neues Temporäres Array angelegt
73
74     for (int i = 0; i < matrixA.length; ++i) {
75         for (int n = 0; n < matrixA[i].length; ++n) {
76             matrixAd[i][n] = matrixA[i][n] + matrixB[i][n];
77         }
78     }
79     return matrixAd;
80 }
81
82 /**
83  * Multiplikation der beiden Matrizen A und B
84  * @param matrixA
85  * @param matrixB
86  * @return Gibt ein neues Array mit den Multiplizierten Werten zurück
87  */
88 private static int[][] multiplikation(int matrixA[][], int matrixB[][]) {
89     int matrixMult[][] = new int[matrixB.length][matrixB[0].length];
90
91     for (int HmatrixB = 0; HmatrixB < matrixB.length; ++HmatrixB)
92         for (int WmatrixB = 0; WmatrixB < matrixB[HmatrixB].length; ++WmatrixB)
93             for (int WmatrixA = 0; WmatrixA < matrixB.length; ++WmatrixA)
94                 matrixMult[HmatrixB][WmatrixB] += matrixA[HmatrixB][WmatrixA] * matrixB[
95                     WmatrixA][WmatrixB];
96
97     return matrixMult;
98 }
99
100 /**
101  * Hier wird die Matrix ausgegeben
102  * @param matrix
103  */
104 private static void printMatrix(int matrix[][]) {
105     for (int y[]: matrix) {
106         for (int x: y)
107             System.out.print(x + "\t");
108         System.out.println();
109     }
110     System.out.println();
111 }
112 }

```

4.2.5 Testdokumentation

Das Programm hat nach dem Aufruf Zwei 2x2 Matrizen erstellt und initialisiert, anschließend miteinander Addiert und Multipliziert. Dabei wurde das Ergebnis mit einem Matrizen Rechner im Internet verglichen und die Ergebnisse haben übereingestimmt.

4.2.6 Benutzungshinweise

Navigieren Sie in der Kommandozeile zum dem Ordner, wo sich die Java Datei befindet. Danach führen sie "javac Matrizen.java" auf. Jetzt können Sie das Programm mit "java Matrizen" starten. Nach dem das Programm gestartet ist, können Sie die größe der Matrix angeben.

4.2.7 Anwendungsbeispiel

Nach dem Aufruf von java Matrizen, sollten wir nun folgendes sehen:

```
1 [sebastian@laptop bin]$ java Matrizen
2 Matrix A:
3 70  50
4 16  52
5
6 Matrix B:
7 80  75
8 11  33
9
10 Addition von A und B:
11 150 125
12 27  85
13
14 Multiplikation von A und B:
15 6150 6900
16 1852 2916
17 [sebastian@laptop bin]$
```

5 Kapitel 5

5.1 Teilaufgabe 1

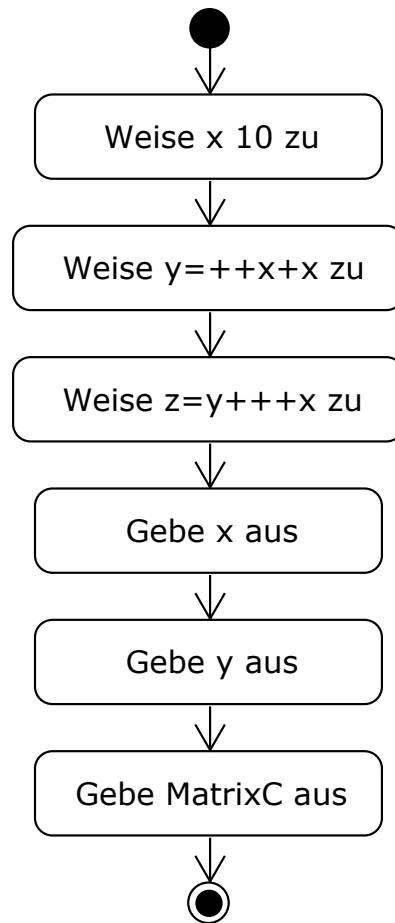
5.1.1 Aufgabenstellung

In der ersten Teilaufgabe sollen wir ein Kleines simples Programm schreiben, welches die Nebeneffekte in Java verdeutlicht.

5.1.2 Anforderungsdefinition

1. Nebeneffekte verdeutlichen.

5.1.3 Entwurf



5.1.4 Quelltext

5.1.4.1 Nebeneffekte.java

```
1 package chapter_05;
2
3 /**
4  * Klasse mit der Main-Methode
5  * @author Sebastian
6  *
7  */
8 public class Nebeneffekte {
9
10     public static void main(String[] args) {
11         int x = 10;
12         int y = ++x+x;
13         int z = y+++--x;
14         System.out.println("Der Wert von x lautet: " + x);
15         System.out.println("Der Wert von y lautet: " + y);
16         System.out.println("Der Wert von z lautet: " + z);
17     }
18
19 }
```

5.1.5 Testdokumentation

Nach dem Start sollte x 10, y 23 und z 32 betragen, dies war auch der Fall.

5.1.6 Benutzungshinweise

Keine Besonderen Benutzungshinweise. Das Programm muss lediglich nur ausgeführt werden.

5.1.7 Anwendungsbeispiel

Nach dem man das Programm gestartet hat, sollte folgende Ausgabe erscheinen:

```
1 [sebastian@laptop bin]$ java Nebeneffekte
2 Der Wert von x lautet: 10
3 Der Wert von y lautet: 23
4 Der Wert von z lautet: 32
5 [sebastian@laptop bin]$
```

5.2 Teilaufgabe 2

5.2.1 Aufgabenstellung

In der zweiten Teilaufgabe sollten wir ein Programm schreiben welches sämtliche Operatoren, die Java beinhaltet veranschaulichen.

5.2.2 Anforderungsdefinition

1. Verwende alle Operatoren in Java.

5.2.3 Entwurf

Es werden jeweils einzelne Methoden erstellt in denen die entsprechenden Operatoren ausgeführt werden.

Operatoren
+main() static -arithmetisch() static -inkrement() static -vergleiche() static -boolische() static -bitshiftign() static -zuweisung() static

5.2.4 Quelltext

5.2.4.1 Operatoren.java

```
1 package chapter_05;
2
3 @SuppressWarnings("unused")
4 public class Operatoren {
5     //Schreiben Sie ein Programm, welches alle Operatoren in Java verwendet.
6     /**
7      * Klasse mit der Main-Methode
8      * Dieses Programm sollte alle Operatoren,
9      * die in Java existieren verdeutlichen
```

```

10  * @param args
11  */
12  public static void main(String[] args) {
13      arithmetisch();
14      inkrement();
15      vergleiche();
16      boolesche();
17      bitshifting();
18      zuweisung();
19  }
20
21  private static void arithmetisch() {
22      System.out.println("Arithmetische Operatoren:");
23      System.out.println("23 + 34 = " + (23 + 34)); // Addition
24      System.out.println("54 - 32 = " + (54 - 32)); // Subtraktion
25      System.out.println("12 * 30 = " + 12 * 30);    // Multiplikation
26      System.out.println("56 / 12 = " + 56 / 12);    // Division
27      System.out.println("74 % 2 = " + 74 % 2);      // Teiler rest, Modulo-Operation,
                                                    errechnet den Rest einer Division
28      int i;
29      System.out.println("int i = +3 = " + (i = +3)); // positives Vorzeichen
30      int n;
31      System.out.println("int n = -i = " + (n = -i)); //negatives Vorzeichen
32  }
33
34  private static void inkrement() {
35      int x = 10;
36      System.out.println("\nInkrement Operatoren:");
37      System.out.println("x = " + x);
38      System.out.println("x++ = " + x++); //Postinkrement: Weist zuerst zu, dann
                                                    hochzählen
39      System.out.println("x = " + x);
40      System.out.println("++x = " + ++x); //Preinkrement: Zählt erst hoch, dann
                                                    zuweisen
41      System.out.println("x = " + x);
42      System.out.println("x-- = " + x--); //Postinkrement: Weist zuerst zu, dann
                                                    hochzählen
43      System.out.println("x = " + x);
44      System.out.println("--x = " + --x); //Preinkrement: Zählt erst hoch, dann
                                                    zuweisen
45      System.out.println("x = " + x);
46  }
47
48  private static void vergleiche() {
49      System.out.println("\nVergleichs Operatoren:");
50      System.out.println("37 == 2 = " + (37 == 2)); // gleich
51      System.out.println("1 != 2 = " + (1 != 2));   // ungleich
52      System.out.println("13 > 3 = " + (13 > 3));   // größer
53      System.out.println("23 < 2 = " + (23 < 2));   // kleiner
54      System.out.println("23 >= 23 = " + (23 >= 23)); // größer oder gleich
55      System.out.println("45 <= 44 = " + (45 <= 44)); // kleiner oder gleich
56  }
57
58  private static void boolesche() {
59      System.out.println("\nBoolesche Operatoren:");
60      System.out.println("!true = " + !true);        // Negation
61      System.out.println("true && true = " + (true && true)); // Und, true 2, genau
                                                    dann wenn alle Argumente true sind
62      System.out.println("true || false = " + (true || false)); // Oder, true, wenn
                                                    mindestens ein Operand true ist
63      System.out.println("true ^ true = " + (true ^ true));   // Xor, true, wenn
                                                    genau ein Operand true ist
64  }

```

```

65
66 private static void bitshifting() {
67     int bit = ~0b10111011 & 0xff;
68     System.out.println("\nBitweise Operatoren:");
69     System.out.println("0b10111011 = ~0b" + Integer.toString(bit, 2)); //Invertiert
        die Bits
70     System.out.println("0b10111011 = ~0b01000100"); //Invertiert die Bits
71     System.out.println("0b10101010 & 0b11111111 = " + Integer.toString(0b10101010 &
        0b11111111, 2)); // Verundet die Bits
72     System.out.println("0b10101010 | 0b01101001 = " + Integer.toString(0b10101010 |
        0b00101001, 2)); // Verodert die Bits
73     System.out.println("0b10101010 ^ 0b11111111 = " + Integer.toString(0b10101010 ^
        0b11111111, 2)); // Exklusives oder
74     System.out.println("0b10101010 >> 2 = " + Integer.toString(0b10101010 >> 2, 2))
        ; // Rechtssshift
75     System.out.println("0b10101010 >>> 1 = " + Integer.toString(0b10101010 >>> 1,
        2)); // Rechtssshift mit Nullen auffüllen
76     System.out.println("0b10101010 << 1 = " + Integer.toString(0b10101010 << 1, 2))
        ; // Linksverschiebung
77 }
78
79 private static void zuweisung() {
80     int a = 20;
81     System.out.println("\nZuweisung Operatoren:");
82     System.out.println("int a = 20"); // Einfache zuweisung
83     System.out.println("a += 10 => " + (a += 10)); // Addiert ein wert zu der
        Variable
84     System.out.println("a -= 20 => " + (a -= 20)); // Subtrahiert ein wert zu
        der Variable
85     System.out.println("a *= 7 => " + (a *= 7)); // Dividiert die Variable durch
        den angegebenen Wert und weist ihn zu
86     System.out.println("a /= 5 => " + (a /= 5)); // Multipliziert die Variable
        durch den angegebenen Wert und weist ihn zu
87     System.out.println("a %= 5 => " + (a %= 5)); // Ermittelt den Rest und weist
        ihn zu
88     System.out.println("a &= 12 => " + (a &= 12)); // Eine bitweise Verundung
89     System.out.println("a |= 10 => " + (a |= 10)); // Bitweise Veroderung
90     System.out.println("a ^= 30 => " + (a ^= 30)); // Exklusives oder auf Bit
        ebene
91     System.out.println("a <= 3 => " + (a <= 3)); // Linksverschiebung
92     System.out.println("a >= 1 => " + (a >= 1)); // Rechtsverschiebung
93     System.out.println("a >>= 2 => " + (a >>= 2)); // Rechtsverschiebung und
        Auffüllen mit Nullen
94 }
95
96 }

```

5.2.5 Testdokumentation

Es wurden alle Berechnungen korrekt ausgeführt.

5.2.6 Benutzungshinweise

Keine Besonderen Benutzungshinweise. Das Programm muss lediglich nur ausgeführt werden.

5.2.7 Anwendungsbeispiel

Nach dem man das Programm gestartet hat, sollte folgende Ausgabe erscheinen:

```

1 [sebastian@laptop bin]$ java Operatoren
2 Arithmeschie Operatoren:
3 23 + 34 = 57
4 54 - 32 = 22

```

```
5 | 12 * 30 = 360
6 | 56 / 12 = 4
7 | 74 % 2 = 0
8 | int i = +3 = 3
9 | int n = -i = -3
10 |
11 | Inkrement Operatoren:
12 | x = 10
13 | x++ = 10
14 | x = 11
15 | ++x = 12
16 | x = 12
17 | x-- = 12
18 | x = 11
19 | --x = 10
20 | x = 10
21 |
22 | Vergleichs Operatoren:
23 | 37 == 2 = false
24 | 1 != 2 = true
25 | 13 > 3 = true
26 | 23 < 2 = false
27 | 23 >= 23 = true
28 | 45 <= 44 = false
29 |
30 | Boolische Operatoren:
31 | !true = false
32 | true && true = true
33 | true || false = true
34 | true ^ true = false
35 |
36 | Bitweise Operatoren:
37 | 0b10111011 = ~0b01000100
38 | 0b10101010 & 0b11111111 = 10101010
39 | 0b10101010 | 0b01101001 = 10101011
40 | 0b10101010 ^ 0b11111111 = 1010101
41 | 0b10101010 >> 2 = 101010
42 | 0b10101010 >>> 1 = 1010101
43 | 0b10101010 << 1 = 101010100
44 |
45 | Zuweisungs Operatoren:
46 | int a = 20
47 | a += 10 => 30
48 | a -= 20 => 10
49 | a *= 7 => 70
50 | a /= 5 => 14
51 | a %= 5 => 4
52 | a &= 12 => 4
53 | a |= 10 => 14
54 | a ^= 30 => 16
55 | a <<= 3 => 128
56 | a >>= 1 => 64
57 | a >>>= 2 => 16
58 | [sebastian@laptop bin]$
```


6 Kapitel 6

6.1 Teilaufgabe 1

6.1.1 Aufgabenstellung

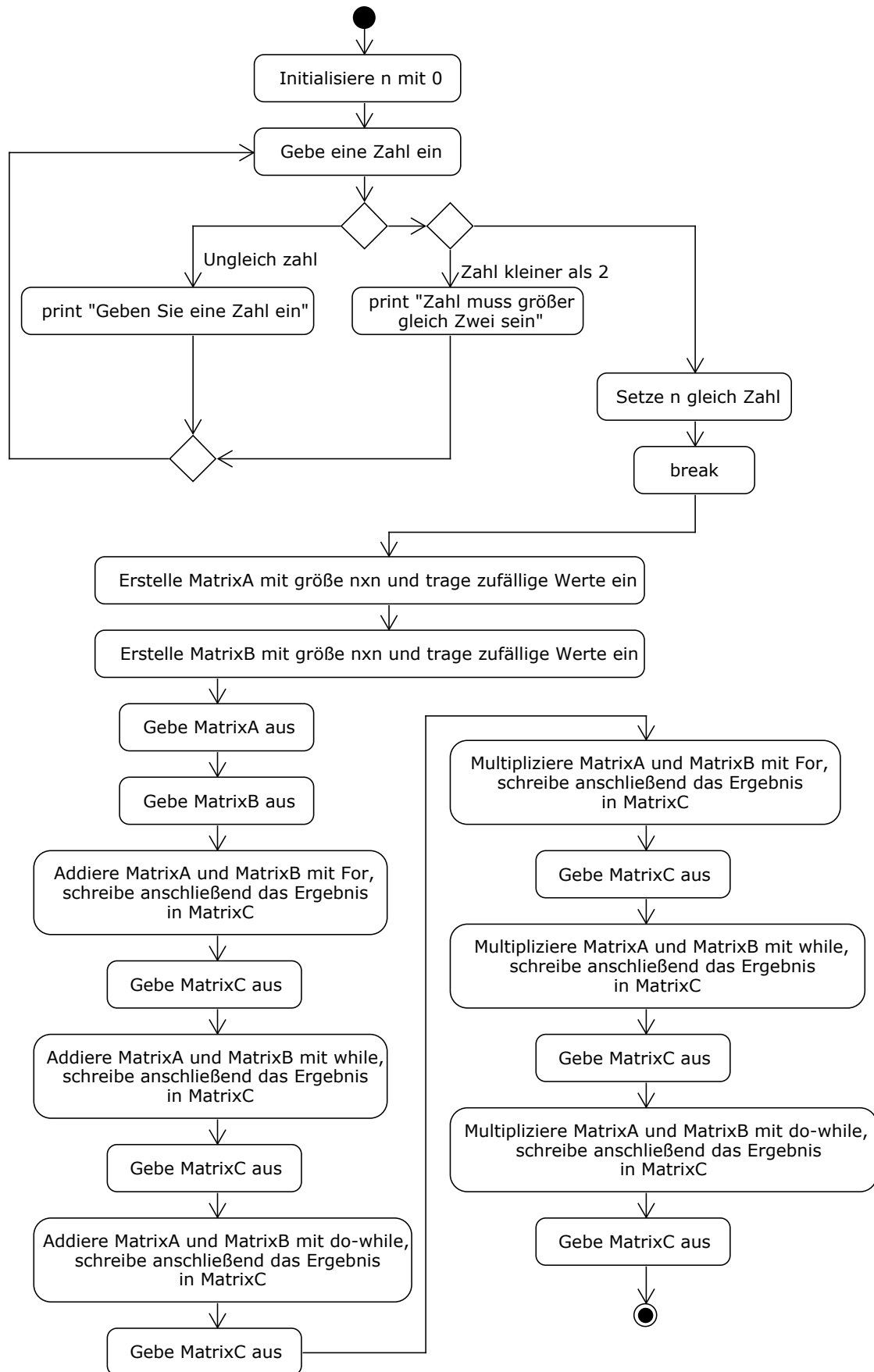
Im Grunde ist es die Selbe Aufgabe wie aus Kapitel 4, Teilaufgabe 2. Doch jetzt solle es auch für $n \times n$ Matrizen funktionieren. Die Größe gibt an, was der Nutzer einträgt. Zusätzlich soll noch die Multiplikation der Matrizen auch mit `while` und `do-while` gelöst werden.

6.1.2 Anforderungsdefinition

1. Unser Input für die Größe der $n \times n$ Matrix.
2. Multiplikation mit `for`, `while`, und `do-while`.

6.1.3 Entwurf

Zuerst wird der Nutzer aufgefordert eine Zahl einzugeben. Anschließend wird in einer `do while` Schleife geprüft, ob die Nutzereingabe valide ist. Es wird einmal geprüft, ob es sich um Buchstaben handelt und ob die Zahl kleiner als 2 ist. Wenn der Fall nicht eintritt, wird die Schleife beendet und es werden zwei $n \times n$ Matrizen erzeugt. Der Rest ist identisch wie aus Kapitel 4, nur dass hier noch für die Berechnung eine weitere `while` und `do while` Schleife verwendet wird.



6.1.4 Quelltext

6.1.4.1 Matrizen.java

```
1 package chapter_06;
2
3 import java.util.Random;
4 import java.util.Scanner;
5 /**
6  * Klasse mit der Main-Methode
7  * Addiert und Multipliziert Matrizen
8  * @author Sebastian
9  *
10 */
11 public class Matrizen {
12
13     public static void main(String[] args) {
14         int[][] matrixA;
15         int[][] matrixB;
16
17         //Hier können sie die GrösSe definieren, z.B. 2,3 oder 5
18         System.out.println("Dieses Programm berechnet eine zufällig erstellte nxn
19             Matrix");
20         System.out.print("Geben sie n an: ");
21         Scanner sc = new Scanner(System.in);
22
23         //Prüft ob der Userinput eine Zahl ist und ob diese gröSSer als Eins ist
24         int n = 0;
25         boolean isInt;
26         do {
27             isInt = sc.hasNextInt();
28             if (!isInt) {
29                 System.out.println("Es dürfen nur Zahlen verwendet werden");
30                 sc.next();
31             } else if ((n = sc.nextInt()) < 2)
32                 System.out.println("Die Zahl muss gröSSer gleich 2 sein");
33             else
34                 break;
35         } while (true);
36
37         /*
38          * Initialisierungsmethode wird mit dem Wert n aufgerufen.
39          * AnschliesSend wird diese Matrix erzeugt und mit
40          * zufällig generierten Zahlen befüllt.
41          */
42         matrixA = initialize(n);
43         matrixB = initialize(n);
44
45         /*
46          * Zuerst werden die Beiden Matrizen A und B jeweils ausgegeben
47          */
48         System.out.println("Matrix A:");
49         printMatrix(matrixA);
50         System.out.println("Matrix B:");
51         printMatrix(matrixB);
52
53         /*
54          * AnschliesSend werden die Matrizen hier Addiert
55          */
56         System.out.println("Addition von A und B:");
57         printMatrix(addition(matrixA, matrixB));
58
59         /*
60          * Und hier Multipliziert
61          */
62         System.out.println("Multiplikation von A und B:");
63         System.out.println("For Schleife");
```

```

60     printMatrix(multiplikationFor(matrixA, matrixB));
61     System.out.println("While Schleife");
62     printMatrix(multiplikationWhile(matrixA, matrixB));
63     System.out.println("Do-While Schleife");
64     printMatrix(multiplikationDoWhile(matrixA, matrixB));
65
66     sc.close();
67 }
68
69 /**
70  * Initialisierung des Arrays
71  * @param n Die gröSse der nxn Matrix
72  * @return matrix
73  */
74 private static int[][] initialize(int n) {
75     int[][] matrix = new int[n][n];
76     /*
77      * Bei der Initialisierung wird einmal durch das gesamt Array durch iteriert.
78      * Dabei werden dann mit Math.random() zufällige Zahlen rein geschrieben.
79      */
80
81     Random rand = new Random();
82     for (int i = 0; i < matrix.length; ++i)
83         for (int l = 0; l < matrix[i].length; ++l)
84             matrix[i][l] = rand.nextInt(200) - 100;
85
86     return matrix;
87 }
88
89 /**
90  * Addition der beiden Matrizen A und B
91  * @param matrixA
92  * @param matrixB
93  * @return Gibt ein neues Array mit den Addierten Werten zurück
94  */
95 private static int[][] addition(int[][] matrixA, int[][] matrixB) {
96     int[][] matrixAd = new int[matrixA.length][matrixA[0].length]; //Es wird ein
97     //neues Temporäres Array angelegt
98
99     for (int i = 0; i < matrixA.length; ++i) {
100         for (int n = 0; n < matrixA[i].length; ++n) {
101             matrixAd[i][n] = matrixA[i][n] + matrixB[i][n];
102         }
103     }
104
105     return matrixAd;
106 }
107
108 /**
109  * Multiplikation der beiden Matrizen A und B
110  * @param matrixA
111  * @param matrixB
112  * @return Gibt ein neues Array mit den Multiplizierten Werten zurück
113  */
114 private static int[][] multiplikationFor(int[][] matrixA, int[][] matrixB) {
115     int[][] matrixMult = new int[matrixB.length][matrixB[0].length];
116
117     //Hier die Variante mit For Schleifen
118     for (int HmatrixB = 0; HmatrixB < matrixB.length; ++HmatrixB)
119         for (int WmatrixB = 0; WmatrixB < matrixB[HmatrixB].length; ++WmatrixB)
120             for (int WmatrixA = 0; WmatrixA < matrixB.length; ++WmatrixA)
121                 matrixMult[HmatrixB][WmatrixB] += matrixA[HmatrixB][WmatrixA] * matrixB[
122                     WmatrixA][WmatrixB];

```

```
121
122     return matrixMult;
123 }
124
125 /**
126  * Multiplikation der beiden Matrizen A und B
127  * @param matrixA
128  * @param matrixB
129  * @return Gibt ein neues Array mit den Multiplizierten Werten zurück
130  */
131 private static int[][] multiplikationWhile(int[][] matrixA, int[][] matrixB) {
132     int[][] matrixMult = new int[matrixB.length][matrixB[0].length];
133
134     int HmatrixB = 0;
135     int WmatrixB = 0;
136     int WmatrixA = 0;
137
138     //Hier die Variante mit While Schleifen
139     while (HmatrixB < matrixB.length) {
140         WmatrixB = 0;
141         while (WmatrixB < matrixB[HmatrixB].length) {
142             WmatrixA = 0;
143             while (WmatrixA < matrixB[HmatrixB].length) {
144                 matrixMult[HmatrixB][WmatrixB] += matrixA[HmatrixB][WmatrixA] * matrixB[
145                     WmatrixA][WmatrixB];
146                 ++WmatrixA;
147             }
148             ++WmatrixB;
149         }
150         ++HmatrixB;
151     }
152     return matrixMult;
153 }
154
155 /**
156  * Multiplikation der beiden Matrizen A und B
157  * @param matrixA
158  * @param matrixB
159  * @return Gibt ein neues Array mit den Multiplizierten Werten zurück
160  */
161 private static int[][] multiplikationDoWhile(int[][] matrixA, int[][] matrixB) {
162     int[][] matrixMult = new int[matrixB.length][matrixB[0].length];
163
164     int HmatrixB = 0;
165     int WmatrixB = 0;
166     int WmatrixA = 0;
167
168     //Hier die Variante mit Do-While Schleifen
169     do {
170         WmatrixB = 0;
171         do {
172             WmatrixA = 0;
173             do {
174                 matrixMult[HmatrixB][WmatrixB] += matrixA[HmatrixB][WmatrixA] * matrixB[
175                     WmatrixA][WmatrixB];
176                 ++WmatrixA;
177             } while (WmatrixA < matrixB[HmatrixB].length);
178             ++WmatrixB;
179         } while (WmatrixB < matrixB[HmatrixB].length);
180         ++HmatrixB;
181     } while (HmatrixB < matrixB.length);
182
183     return matrixMult;
184 }
```

```

182     }
183
184     /**
185      * Hier wird die Matrix ausgegeben
186      * @param matrix
187      */
188     private static void printMatrix(int[][] matrix) {
189         for (int[] y : matrix) {
190             for (int x: y)
191                 System.out.print(x + "\t");
192             System.out.println();
193         }
194         System.out.println();
195     }
196
197 }

```

6.1.5 Testdokumentation

Bei Zahlen die kleiner als 2 waren, sowie Buchstaben, kam es zu einer entsprechenden Fehlermeldung. Anschließend konnte man den Wert erneut eintippen. Bei den Ergebnissen der Matrizen wurden die Werte mit einem Onlinerechner verglichen, hierbei hat alles übereingestimmt.

6.1.6 Benutzungshinweise

Nach dem aufrufen des Programms, wird der Nutzer aufgefordert eine Zahl einzugeben. Diese muss größer als ein sein.

6.1.7 Anwendungsbeispiel

Nach dem man das Programm gestartet hat, sollte folgende Ausgabe erscheinen:

```

1  [sebastian@laptop bin]$ java Matrizen
2  Dieses Programm berechnet eine zufällig erstellte nxn Matrix
3  Geben sie n an: -45
4  Die Zahl muss gröSSer gleich 2 sein
5  test
6  Es dürfen nur Zahlen verwendet werden
7  5
8  Matrix A:
9  78  88  96  91  50
10 17  16  14  17  77
11 34  1 45  21  42
12 63  1 92  76  57
13 84  59 13  85  46
14
15 Matrix B:
16 38  36  45  22  18
17 96  4  0  93  79
18 86  38  92  50  92
19 1 54  63  71  10
20 4 62  91  31  55
21
22 Addition von A und B:
23 116 124 141 113 68
24 113 20  14  110 156
25 120 39  137 71  134
26 64  55  155 147 67
27 88  121 104 116 101
28
29 Multiplikation von A und B:
30 For Schleife

```

```
31 19959 14822 22625 22711 20848
32 3711 6900 10131 6156 7263
33 5447 6676 10815 5884 7351
34 10706 13406 21274 13242 13572
35 10243 11196 14517 15446 10749
36
37 While Schleife
38 19959 14822 22625 22711 20848
39 3711 6900 10131 6156 7263
40 5447 6676 10815 5884 7351
41 10706 13406 21274 13242 13572
42 10243 11196 14517 15446 10749
43
44 Do-While Schleife
45 19959 14822 22625 22711 20848
46 3711 6900 10131 6156 7263
47 5447 6676 10815 5884 7351
48 10706 13406 21274 13242 13572
49 10243 11196 14517 15446 10749
50 [sebastian@laptop bin]$
```

6.2 Teilaufgabe 2

6.2.1 Aufgabenstellung

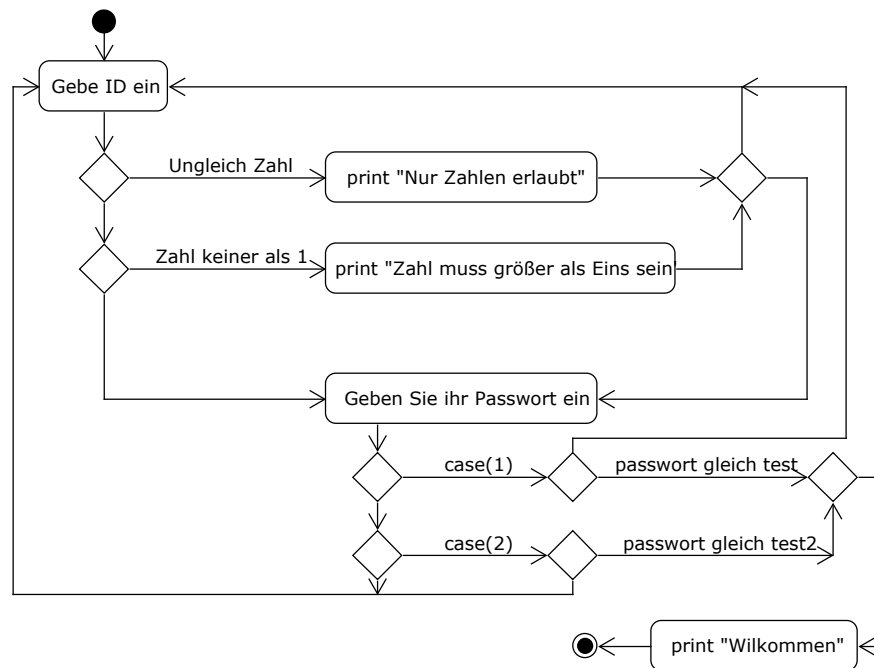
In der zweiten Teilaufgabe sollten wir ein Programm schreiben welches Sprunganweisungen in Java Sinnvoll verdeutlichen.

6.2.2 Anforderungsdefinition

1. Verwendung von Sprunganweisungen.
2. Mindestens ein switch-Anweisung.

6.2.3 Entwurf

Im folgenden ist ein Entwurf für ein einfaches Login. Zuerst wird der Nutzer aufgefordert seine ID anzugeben. Dabei wird geprüft ob es sich um eine Zahl handelt und ob diese Zahl größer gleich Eins ist. Anschließend wird in einem switch überprüft ob eine solche ID existiert und ob das angegebene Passwort auch das richtige ist. Wenn alles Stimmt wird eine Willkommens Nachricht ausgegeben. Andernfalls wird der Nutzer aufgefordert alles erneut einzutippen.



6.2.4 Quelltext

6.2.4.1 Sprunganweisungen.java

```

1 package chapter_06;
2
3 import java.util.Scanner;
4
5 /**
6  * Klasse mit der Main-Methode
7  * @author Sebastian
8  *
9  */
10 public class Sprunganweisungen {
11
12     public static void main(String[] args) {
13         login();
14     }
15
16     /**
17      * Kleine einfache Implementierung von Nutzern, mithilfe
18      * einer switch-Anweisung
19      * @param userID ID die beim login eingegeben wurde
20      * @param userPw Password was bei login eingegeben wurde
21      * @return Wenn die ID und das Passwort übereinstimmen,
22      * wird true zurück gegeben
23      */
24     private static boolean userData(int userID, String userPw) {
25         return switch (userID) {
26             case 1 -> userPw.equals("hallo");
27             case 112 -> userPw.equals("das");
28             case 124 -> userPw.equals("ist");
29             case 345 -> userPw.equals("nicht");
30             case 653 -> userPw.equals("geheim");
31             default -> false;
32         };
33     }
34 }

```



```
35  /**
36   * Hier befindet sich das Login feld
37   */
38  private static void login() {
39      int id;
40      Scanner sc = new Scanner(System.in);
41      do {
42          System.out.println("Willkommen...!");
43          System.out.print("ID      : ");
44
45          /**
46           * Eine kleine Abfrage die Prüft, ob die eingegebene
47           * ID nur aus Zahlen besteht
48           */
49          do {
50              if (!sc.hasNextInt()) {
51                  System.out.println("Error, es dürfen nur Zahlen enthalten sein.");
52                  sc.next();
53              } else if ((id = sc.nextInt()) < 1)
54                  System.out.println("Die Zahl muss gröSSer gleich 1 sein");
55              else
56                  break;
57              System.out.print("ID      : ");
58          } while (true);
59
60          System.out.print("Passwort: ");
61          /**
62           * Wenn ein Nutzer mit dem angegebenen Passwort
63           * nicht existiert, wird die ID zurückgesetzt
64           * und eine Fehlermeldung wird ausgegeben
65           */
66          if(!userData(id, sc.next())) {
67              System.out.println("Ihre Angaben sind leider falsch, versuchen Sie es
68                  erneut.");
69              break;
70
71          /**
72           * Die Schleife wird solange durchlaufen, bis sich ein nutzer
73           * erfolgreich angemeldet hat.
74           */
75          } while (true);
76
77          System.out.println("Juhu, Sie haben sich eingeloggt");
78          sc.close();
79      }
80  }
```

6.2.5 Testdokumentation

Bei Zahlen die kleiner als 1 waren, sowie Buchstaben, kam es zu einer entsprechenden Fehlermeldung. Anschließend konnte man den Wert erneut eintippen.

6.2.6 Benutzungshinweise

Nach dem aufrufen des Programms, wird der Nutzer aufgefordert seine NutzerID anzugeben, sowie anschließend sein Passwort. Bei inkorrekt eingaben, wird man erneut aufgefordert die Daten einzutippen.

6.2.7 Anwendungsbeispiel

Bei Erfolgreicher Anmeldung:

```
1 [sebastian@laptop bin]$ java Sprunganweisungen
2 Willkommen...!
3 ID      : 1
4 Passwort: hallo
5 Juhu, Sie haben sich eingeloggt
6 [sebastian@laptop bin]$
```

Bei inkorrektter Anmeldung:

```
1 [sebastian@laptop bin]$ java Sprunganweisungen
2 Willkommen...!
3 ID      : 12
4 Passwort: qwert
5 Ihre Angaben sind leider falsch, versuchen Sie es erneut.
6 ID      : -1
7 Passwort: hallo
8 Die Zahl muss gröSS er gleich 1 sein
9 ID      :
10 [sebastian@laptop bin]$
```

7 Kapitel 7

7.1 Aufgabenstellung

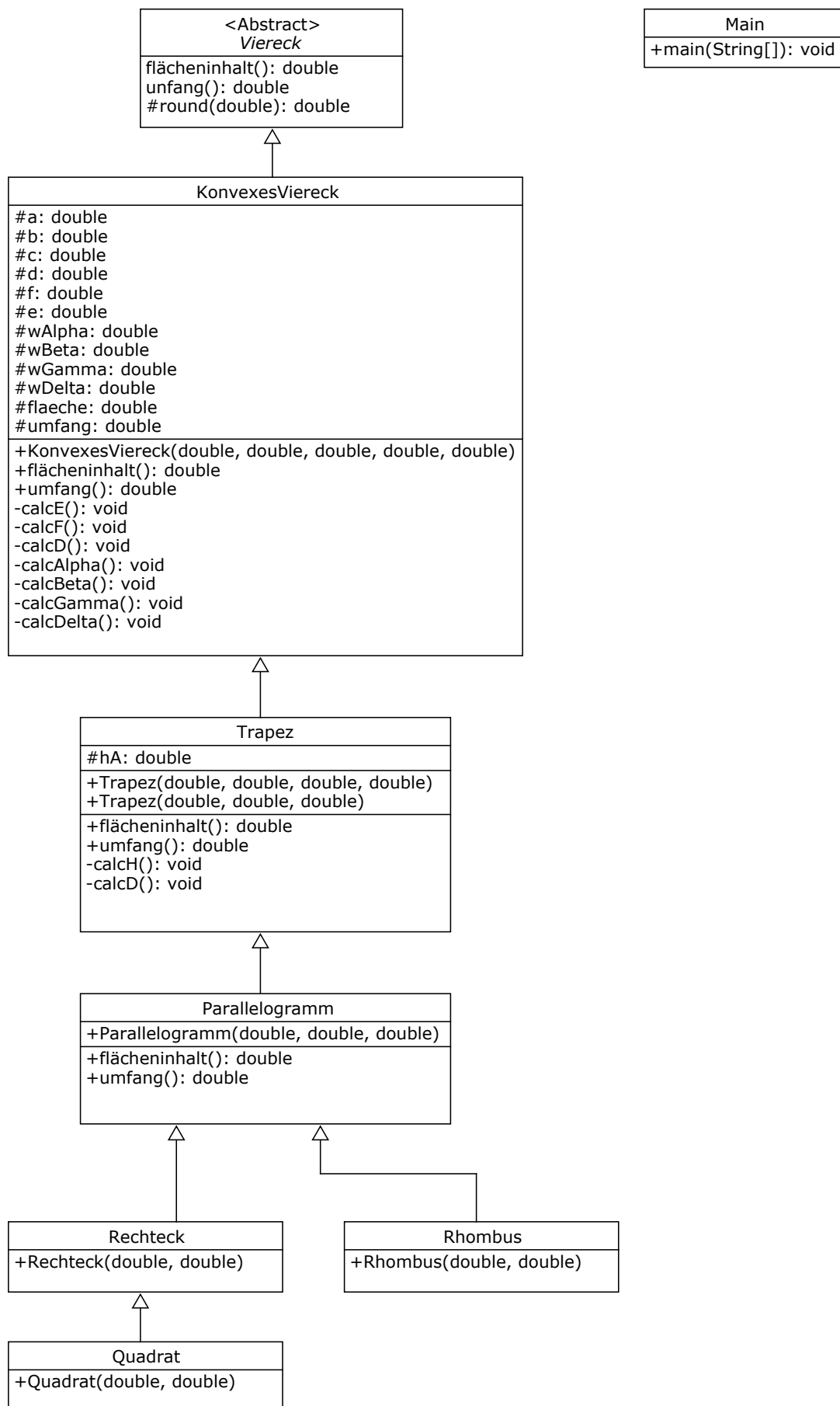
Wie sollen 7 Klassen Definieren, dabei sollen wir uns überlegen welche davon Instanzierbar sind und welche nicht. Anschließend soll jede Klasse sinnvolle Methoden und Attribute enthalten.

7.2 Anforderungsdefinition

1. Definiere folgende Klassen: Viereck, konvexes Viereck, Trapez, Parallelogramm, Rhombus, Rechteck, Quadrat
2. Definiere sinnvolle Methoden und Attribute

7.3 Entwurf

Für den Entwurf wurde ein Klassendiagramm angefertigt. Wobei hier die Klasse Viereck Abstrakt ist und die Wesentlichen Methoden beinhaltet. Beim Konvexen Viereck werden die wesentlichen Attribute eines Vierecks definiert, dazu auch deren Berechnung. Zudem muss hier auch die Methode für den Flächeninhalt und Umfang überschrieben werden. Das Trapez erbt vom Konvexen Viereck, Parallelogramm vom Trapez, Rhombus vom Parallelogramm, Rechteck von Parallelogramm und Quadrat vom Rechteck.



7.4 Quelltext

7.4.1 Main.java

```
1 package chapter_07;
2
3 import chapter_07.figures.*;
4
5 public class Main {
6
7     public static void main(String[] args) {
8
9         KonvexesViereck konvexesViereck = new KonvexesViereck(60, 80, 50, 60, 70);
10        System.out.println(konvexesViereck.umfang() + " " + konvexesViereck.
11            flächeninhalt());
12
13        Trapez trapez = new Trapez(10, 5, 5, 80);
14        System.out.println(trapez.umfang() + " " + trapez.flächeninhalt());
15
16        Trapez trapez2 = new Trapez(10, 20, 10);
17        System.out.println(trapez2.umfang() + " " + trapez2.flächeninhalt());
18
19        Parallelogramm parallelogramm = new Parallelogramm(10, 20, 20);
20        System.out.println(parallelogramm.umfang() + " " + parallelogramm.
21            flächeninhalt());
22
23        Rhombus rhombus = new Rhombus(8, 30);
24        System.out.println(rhombus.umfang() + " " + rhombus.flächeninhalt());
25
26        Rechteck rechteck = new Rechteck(5, 10);
27        System.out.println(rechteck.umfang() + " " + rechteck.flächeninhalt());
28
29        Quadrat quadrat = new Quadrat(5);
30        System.out.println(quadrat.umfang() + " " + quadrat.flächeninhalt());
31    }
32 }
```

7.4.2 Viereck.java

```
1 package chapter_07.figures;
2
3 abstract class Viereck {
4
5     abstract double flächeninhalt();
6     abstract double umfang();
7
8     protected double round(double value) {
9         return Math.round(value * 10000.0) / 10000.0;
10    }
11 }
```

7.4.3 KonvexesViereck.java

```
1 package chapter_07.figures;
2
3 public class KonvexesViereck extends Viereck {
4     protected double a = 0;
```

```

5      protected double b = 0;
6      protected double c = 0;
7      protected double d = 0;
8
9      protected double f = 0;
10     protected double e = 0;
11
12     protected double wAlpha = 0;
13     protected double wBeta = 0;
14     protected double wGamma = 0;
15     protected double wDelta = 0;
16
17     protected double flaeche;
18     protected double umfang;
19
20     /**
21      * Konstruktor, hier werden die Attribute initialisiert
22      * @param a
23      * @param b
24      * @param c
25      * @param winkelBeta
26      * @param winkelGamma
27      */
28     public KonvexesViereck(double a, double b, double c, double winkelBeta, double
        winkelGamma) {
29         this.a = a;
30         this.b = b;
31         this.c = c;
32
33         this.wBeta = winkelBeta;
34         this.wGamma = winkelGamma;
35
36         calcE();
37         calcF();
38         calcD();
39         calcAlpha();
40         calcDelta();
41
42         try {
43             if (wAlpha + wBeta + wGamma + wDelta < 359.99)
44                 throw new Exception("Error, die Summe der Winkel ist under 360 Grad
                    " + (wAlpha + wBeta + wGamma + wDelta));
45         } catch (Exception exception) {
46             exception.printStackTrace();
47             System.exit(0);
48         }
49
50     }
51
52     @Override
53     /**
54      * Gibt de flächeninhalt zurück
55      */
56     public double flächeninhalt() {
57         if (flaeche != 0)
58             return flaeche;
59         else
60             //Wenn der Flächeninhalt 0 ist, wird die fläche berechnet
61             return flaeche = (Math.sqrt(((a+f+d)/2) * ((a+f+d)/2-a) * ((a+f+d)/2-f)
                * ((a+f+d)/2-d)) + Math.sqrt(((b+c+f)/2) * ((b+c+f)/2-b) * ((b+c+f)
                /2-c) * ((b+c+f)/2-f));
62     }
63

```

```
64  @Override
65  /**
66   * Gibt den umfang zurück
67   */
68  public double umfang() {
69      if (umfang != 0)
70          return umfang;
71      else
72          //Wenn der Umfang 0 ist, wird der umfang berechnet
73          return umfang = a + b + c + d;
74  }
75
76  /**
77   * Berechnet die Diagonale E
78   */
79  private void calcE() {
80      if (wBeta != 0)
81          e = Math.sqrt(a*a + b*b - 2 * a * b * Math.cos(Math.toRadians(wBeta)));
82      else
83          e = Math.sqrt(c*c + d*d - 2 * c * d * Math.cos(Math.toRadians(wDelta)));
84      ;
85
86  /**
87   * Berechnet die Diagonale F
88   */
89  private void calcF() {
90      if (wAlpha != 0)
91          f = Math.sqrt(a*a + d*d - 2 * a * b * Math.cos(Math.toRadians(wAlpha)));
92      ;
93      else
94          f = Math.sqrt(b*b + c*c - 2 * b * c * Math.cos(Math.toRadians(wGamma)));
95      ;
96  }
97
98  /**
99   * Berechnet die Seite D
100  */
101  private void calcD() {
102      double tmp = wBeta - Math.toDegrees(Math.acos((f*f + b*b - c*c) / (2 * f *
103          b)));
104      d = Math.sqrt(a*a + f*f - 2 * a * f * Math.cos(Math.toRadians(tmp)));
105  }
106
107  /**
108   * Berechnet den Winkel Alpha
109   */
110  private void calcAlpha() {
111      wAlpha = round(Math.toDegrees(Math.acos((a*a + d*d - f*f) / (2 * a * d))));
112  }
113
114  /**
115   * Berechnet den Winkel Beta
116   */
117  private void calcBeta() {
118      wBeta = round(Math.toDegrees(Math.acos((a*a + b*b - e*e) / (2 * a * b))));
119  }
120
121  /**
122   * Berechnet den Winkel Gamma
123   */
124  private void calcGamma() {
```

```

123         wGamma = round(Math.toDegrees(Math.acos((b*b + c*c - f*f) / (2 * b * c))));
124     }
125
126     /**
127     * Berechnet den Winkel Delta
128     */
129     private void calcDelta() {
130         wDelta = round(Math.toDegrees(Math.acos((c*c + d*d - e*e) / (2 * c * d))));
131     }
132 }

```

7.4.4 Trapez.java

```

1 package chapter_07.figures;
2
3 public class Trapez extends KonvexesViereck {
4
5     protected double hA;
6
7     /**
8     * Konstruktor, hier werden die Attribute initialisiert
9     * @param a
10    * @param b
11    * @param c
12    * @param winkelBeta
13    */
14    public Trapez(double a, double b, double c, double winkelBeta) {
15        super(a, b, c, winkelBeta, 180 - winkelBeta);
16        try {
17            if (c > a)
18                throw new Exception("Seite C muss kleiner als seine A sein");
19            if (wAlpha > 90.00 || wBeta > 90.00 || wGamma <= 90.00 || wDelta <=
20                90.00)
21                throw new Exception("Alpha oder Beta ist gröSSer als 90 Grad");
22        } catch (Exception exception) {
23            exception.printStackTrace();
24            System.exit(0);
25        }
26        calcH();
27        calcD();
28    }
29
30    /**
31    * Konstruktor für ein Rechtwinkliges Trapez
32    * @param a
33    * @param b
34    * @param c
35    */
36    public Trapez (double a, double b, double c) {
37        super(a, b, c, 90, 90);
38
39        calcH();
40        calcD();
41    }
42
43    @Override
44    /**
45    * Gibt de flächeninhalt zurück
46    */
47    public double flächeninhalt() {
48        if (flaeche != 0)

```

```
48         return flaeche;
49     else
50         //Wenn der Flächeninhalt 0 ist, wird die fläche berechnet
51         return flaeche = ((a + c) * hA) / 2;
52     }
53
54     @Override
55     /**
56      * Gibt den umfang zurück
57      */
58     public double umfang() {
59         if (umfang != 0)
60             return umfang;
61         else
62             //Wenn der Umfang 0 ist, wird der umfang berechnet
63             return umfang = a + b + c + d;
64     }
65
66     /**
67      * Berechnet die Höhe
68      */
69     private void calcH() {
70         hA = b * Math.sin(Math.toRadians(wBeta));
71     }
72
73     /**
74      * Berechnet die Seite D
75      */
76     private void calcD() {
77         d = Math.sqrt(Math.pow(a - c - Math.sqrt(b*b - hA*hA), 2) + hA*hA);
78     }
79
80 }
```

7.4.5 Parallelogramm.java

```
1 package chapter_07.figures;
2
3 public class Parallelogramm extends Trapez {
4
5     /**
6      * Konstruktor, hier werden die Attribute initialisiert
7      * @param a
8      * @param b
9      * @param winkelAlpha
10     */
11     public Parallelogramm(double a, double b, double winkelAlpha) {
12         super(a, b, a);
13         wAlpha = winkelAlpha;
14         wBeta = 180 - winkelAlpha;
15     }
16
17     @Override
18     /**
19      * Gibt de flächeninhalt zurück
20      */
21     public double flächeninhalt() {
22         if (flaeche != 0)
23             return flaeche;
24         else
25             //Wenn der Flächeninhalt 0 ist, wird die fläche berechnet
```



```
26         return flaeche = a * b * Math.sin(Math.toRadians(wAlpha));
27     }
28
29     @Override
30     /**
31      * Gibt den umfang zurück
32      */
33     public double umfang() {
34         if (umfang != 0)
35             return umfang;
36         else
37             //Wenn der Umfang 0 ist, wird der umfang berechnet
38             return umfang = 2*a + 2*b;
39     }
40 }
```

7.4.6 Rhombus.java

```
1 package chapter_07.figures;
2
3 public class Rhombus extends Parallelogramm {
4
5     /**
6      * Konstruktor, hier werden die Attribute initialisiert
7      * @param a
8      * @param hA
9      */
10    public Rhombus(double a, double winkelAlpha) {
11        super(a, a, winkelAlpha);
12    }
13 }
```

7.4.7 Rechteck.java

```
1 package chapter_07.figures;
2
3 public class Rechteck extends Parallelogramm {
4
5     /**
6      * Konstruktor, hier werden die Attribute initialisiert
7      * @param a
8      * @param b
9      */
10    public Rechteck(double a, double b) {
11        super(a, b, 90);
12    }
13 }
```

7.4.8 Quadrat.java

```
1 package chapter_07.figures;
2
3 public class Quadrat extends Rechteck {
4
5     /**
6      * Konstruktor, hier werden die Attribute initialisiert
```

```

7      * @param a
8      */
9      public Quadrat(double a) {
10         super(a, a);
11     }
12 }

```

7.5 Testdokumentation

Mittels WhiteBox Test wurden verschiedene Formen getestet und bei fast allen Testdurchläufen kam das richtige Ergebnis heraus.

7.6 Anwendungsbeispiel

Nach dem man das Programm gestartet hat, sollte folgende Ausgabe erscheinen:

```

1  [sebastian@laptop bin]$ java Main
2  223.2733105547887 2808.779545976002
3  35.17638090205041 49.99999999999999
4  50.0 120.0
5  32.0 45.256
6  30.0 50.0
7  20.0 25.0
8  [sebastian@laptop bin]$

```

8 Kapitel 8

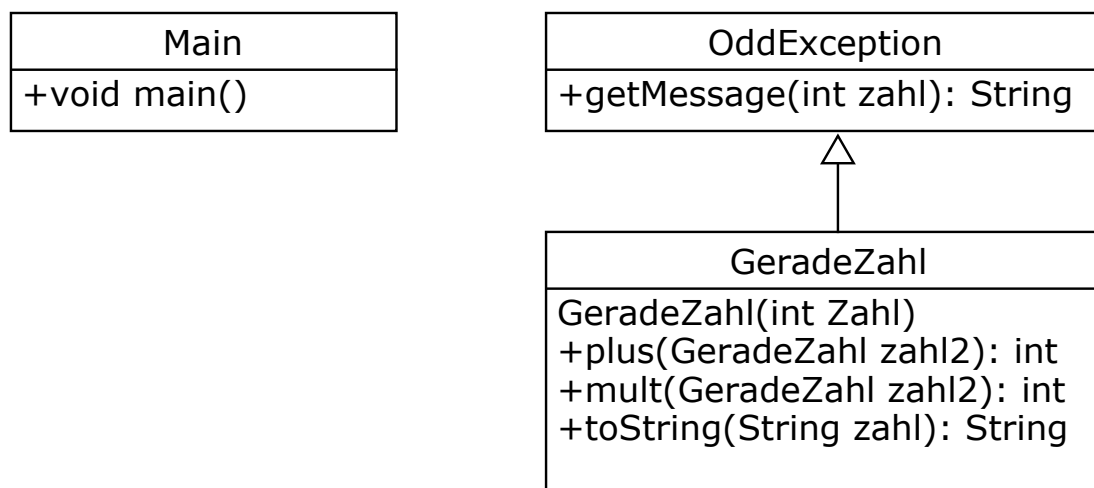
8.1 Aufgabenstellung

Wir sollen eine Klasse schreiben die Nur Gerade Zahlen annimmt, andernfalls soll eine Exception geworfen werden. Zusätzlich kommen noch zwei Methoden in diese Klasse für die Multiplikation und Addition.

8.2 Anforderungsdefinition

1. Erstelle eine Klasse GeradeZahl.
2. Erstelle eine Methode für die Multiplikation und Addition.
3. Bei ungeraden Zahlen soll eine Exception geworfen werden.

8.3 Entwurf



8.4 Quelltext

8.4.1 Main.java

```
1 package chapter_08;
2
3 /**
4  * Klasse mit der Main-Methode
5  * @author sebastian
6  */
7 public class Main {
8
9     public static void main(String[] args) throws OddException {
10         GeradeZahl zahl1 = new GeradeZahl(10);
11         GeradeZahl zahl2 = new GeradeZahl(21);
12         GeradeZahl zahl3 = new GeradeZahl(30);
13         GeradeZahl zahl4 = new GeradeZahl(13);
14
15         System.out.println("Zahl 1 = " + zahl1.mult(zahl2));
16         System.out.println("Zahl 2 = " + zahl2);
17         System.out.println("Zahl 3 = " + zahl3.mult(zahl2));
18         System.out.println("Zahl 4 = " + zahl4.plus(zahl1));
19     }
20 }
```

8.4.2 GeradeZahl.java

```
1 package chapter_08;
2
3 public class GeradeZahl {
4     private int zahl1;
5
6     /**
7      * Konstruktor, hier wird geprüft ob es sich bei der Zahl um eine gerade Zahl
8      * handelt
9      * @param zahl1
10     */
11     public GeradeZahl(int zahl1) throws OddException {
12         this.zahl1 = zahl1;
13
14         //Wenn die Zahl ungerade ist, wird eine Exception geworfen
15         if (zahl1%2 == 1) throw new OddException(zahl1);
16     }
17
18     /**
19      * Addiert Zwei GeradeZahl objekte miteinander
20      * @param zahl2
21      * @return Gibt eine int Zahl zurück
22     */
23     public int plus(GeradeZahl zahl2) {
24         return zahl1 + zahl2.getZahl();
25     }
26
27     /**
28      * Multipliziert Zwei GeradeZahl objekte miteinander
29      * @param zahl2
30      * @return Gibt eine int Zahl zurück
31     */
32     public int mult(GeradeZahl zahl2) {
33         return zahl1 * zahl2.getZahl();
34     }
35 }
```

```
35  /**
36   * @return Liefert die Zahl zurück
37   */
38  public int getZahl() {
39      return zahl1;
40  }
41
42  /**
43   * Bei einem Print wird diese Methode ausgeführt
44   * @return Liefert einen String mit der Zahl zurück
45   */
46  @Override
47  public String toString() {
48      return "" + getZahl();
49  }
50 }
```

8.4.3 OddException.java

```
1  package chapter_08;
2
3  public class OddException extends Exception {
4      int zahl;
5
6      public OddException(int zahl) {
7          this.zahl = zahl;
8      }
9
10     // @Override
11     public String getMessage() {
12         return "Error, " + zahl + " ist keine Gerade Zahl! Die Zahl wurde um Eins
13             erhöht.";
14     }
15
16 }
```

8.5 Testdokumentation

Bei einer Ungeraden Zahl wurde eine Exception geworfen und wie erwarte die Zahl anschließend um eins erhöht.

8.6 Benutzungshinweise

Keine Besonderen Benutzungshinweise. Das Programm muss lediglich nur ausgeführt werden.

8.7 Anwendungsbeispiel

Bei einer ungeraden Zahl sollte diese Ausgabe erscheinen.

```
1  [sebastian@laptop bin]$ java Main
2  Exception in thread "main" chapter_08.OddException: Error, 21 ist keine Gerade Zahl
3      ! Die Zahl wurde um Eins erhöht.
4  at chapter_08.GeradeZahl.<init>(GeradeZahl.java:14)
5  at chapter_08.Main.main(Main.java:11)
6  [sebastian@laptop bin]$
```

Und bei einer Geraden diese.

```

1 [sebastian@laptop bin]$ java Main
2 Zahl 1 = 200
3 Zahl 2 = 20
4 Zahl 3 = 600
5 Zahl 4 = 24
6 [sebastian@laptop bin]$

```

9 Kapitel 10

9.1 Aufgabenstellung

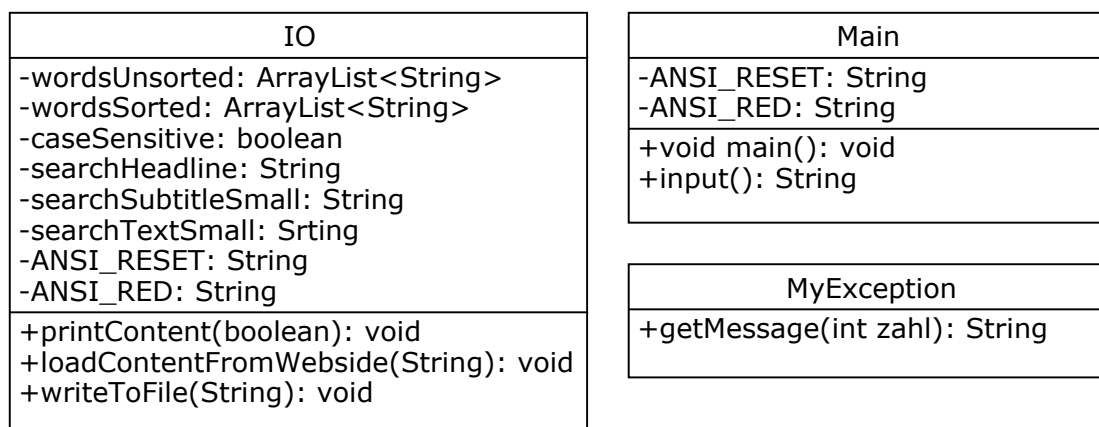
Wir sollen ein Programm schreiben Welches ein Artikel von der Tagesschau herunterlädt und anschließend in einer Datei abspeichert. Der Nutzer kann angeben, welche Datei er einlesen möchte, anschließend kann er die gezählten Wörter in Alphabetischer Reihenfolge ausgeben lassen. Zusätzlich soll über ein Startparameter entschieden werden, ob Groß- und Kleinschreibung beachtet wird.

9.2 Anforderungsdefinition

1. Ein Artikel der Tagesschau mit mehr als 500 Wörtern soll runter geladen werden.
2. Die Wörter sollen in einer geeigneten Collection abgelegt werden,
3. Die Wörter sollen Sortiert ausgegeben können.
4. Bei der Ausgabe der Worte wird zusätzlich die Häufigkeit mit angegeben.
5. Beim Start soll entschieden werden ob Groß- oder Kleinschreibung.

9.3 Entwurf

Zum Entwurf wurde ein Klassendiagramm gewählt welches die Methoden und deren Attribute der jeweiligen Klassen veranschaulicht. Die IO Klasse befasst sich mit dem Einlesen einer Datei, sowie auch das anlegen neuer Dateien. Des weiterem kann die Klasse auch einen Artikel von der Tagesschau heruntergeladen werden, hierfür muss nur ein Link angegeben werden. Wenn ein Artikel heruntergeladen wurden, kann dieser eingelesen werden und anschließend Sortiert ausgegeben werden.



9.4 Quelltext

9.4.1 Main.java

```
1 package chapter_10;
2
3 import java.util.Scanner;
4
5 /**
6  * Klasse mit der Main-Methode
7  * @author sebastian
8  */
9 public class Main {
10     private static final String ANSI_RESET = "\u001B[0m";
11     private static final String ANSI_RED = "\u001B[31m";
12
13     public static void main(String[] args) throws MyException {
14         boolean caseSensitive;
15         /*
16          * Prüft ob bei dem aufruf des Programmes ein Parameter übergeben wurde.
17          * Wenn dies der fall ist und der Parameter true oder false ist, wird
18          * dieser
19          * in einer Variable gespeichert. Andernfalls wird eine exception geworfen.
20          */
21         if (args.length != 0 && (args[0].equalsIgnoreCase("true") || args[0].
22             equalsIgnoreCase("false"))) {
23             caseSensitive = Boolean.parseBoolean(args[0]);
24         } else
25             throw new MyException();
26
27         IO io = new IO(caseSensitive);
28         int option;
29
30         Scanner sc = new Scanner(System.in);
31         do {
32             //Hier werden die auswahlmöglichkeiten ausgegeben
33             System.out.println("Bitte wähle einer der Folgenden Optionen aus:");
34             System.out.println("1) Einen Artikel von der Tagesschau herunterladen."
35                 );
36             System.out.println("2) Eine Datei laden.");
37             System.out.println("3) Wörter ausgeben.");
38             System.out.println("0) Programm Beenden.");
39             do {
40                 /*
41                  * Wenn der Nutzer einen Buchstaben oder eine Zahl auSSerhalb des
42                  * wertebereiches
43                  * angibt wird eine entsprechende Fehlermeldung angegeben.
44                  */
45                 if (!sc.hasNextInt()) {
46                     System.out.println("Es sind nur Zahlen erlaubt");
47                     sc.next();
48                 } else if ((option = sc.nextInt()) > 4 || option < 0)
49                     System.out.println("Zahl auSSerhalb des Bereiches!");
50                 else
51                     break;
52             } while (true);
53
54             //Bei einer erfolgreichen eingabe, wird eine dieser Optionen ausgeführt
55             switch (option) {
56                 case 1:
57                     System.out.println("Bitte gebe den Link an:");
58                     io.loadContentFromWebsite(input());
59                     break;
60                 case 2:
61                     System.out.println("Bitte geben Sie den Datei namen an:");
62                     io.readFromFile(input());
63                     break;
```

```

60         case 3:
61             io.printContent(true);
62             break;
63     }
64     } while (option > 0);
65
66     sc.close();
67 }
68
69 //Der Input wird hier zu einem String zusammengeführt
70 public static String input() {
71     Scanner sc = new Scanner(System.in);
72     String tmp;
73     tmp = sc.next();
74     tmp += sc.nextLine();
75     return tmp;
76 }
77 }

```

9.4.2 IO.java

```

1 package chapter_10;
2
3 import java.io.*;
4 import java.net.URL;
5 import java.nio.charset.Charset;
6 import java.nio.charset.StandardCharsets;
7 import java.util.ArrayList;
8 import java.util.Collections;
9 import java.util.Scanner;
10
11 public class IO{
12     private static ArrayList<String> wordsUnsorted;
13     private static ArrayList<String> wordsSorted;
14     private final boolean caseSensitive;
15
16     private static final String searchHeadline = "class=\"meldungskopf__headline--
17         text\"";
18     private static final String searchSubtitleSmall = "class=\"meldung_";
19     private static final String searchTextSmall = "textabsatz columns twelve\"";
20
21     private static final String ANSI_RESET = "\u001B[0m";
22     private static final String ANSI_RED = "\u001B[31m";
23
24     public IO(boolean caseSensitive){
25         this.caseSensitive = caseSensitive;
26     }
27
28     /**
29      * Listet die einzelnen wörter aus dem ausgewählten Artikel auf und gibt
30      * zusätzlich der Anzahl an
31      * @param sorted Noch keinen nutzen
32      */
33     public void printContent(boolean sorted) {
34         //Zuerst wird geprüft ob eine Datei eingelesen wurde
35         if (wordsSorted != null) {
36             ArrayList<String> tmpWords = new ArrayList<>(wordsSorted);
37             //Der erste eintrag wird in einen String geschrieben
38             String currentWord = tmpWords.get(0);
39             int countWords = 0;
40
41             for (int i = 0; i < tmpWords.size(); ++i) {

```

```

41      /*
42      * Falls das Wort übereinstimmt mit dem zwischengespeichertem word,
        wird der
43      * Counter um eins erhöht und der aktuelle eintrag gelöscht
44      */
45      if (caseSensitive ? currentWord.equals(tmpWords.get(i)) :
        currentWord.equalsIgnoreCase(tmpWords.get(i))) {
46          ++countWords;
47          tmpWords.remove(i--);
48      /*
49      * Falls die Wörter unterschiedlich sind, wird das aktuelle Word
        ausgegeben plus
50      * deren Anzahl, anschliessend wird aktuelle word mit dem neuen
        ersetzt und der
51      * counter zurückgesetzt
52      */
53      } else {
54          System.out.println(currentWord + " " + countWords + "x");
55          currentWord = tmpWords.get(i);
56          countWords = 1;
57      }
58    }
59    } else
60        //Ausgabe, falls keine Datei eingelesen wurde
61        System.out.println(ANSI_RED + "Es wurde noch keine Datei Geladen!" +
        ANSI_RESET);
62    }
63
64    /**
65     * Lädt einen Artikel von der Tagesschau herunter
66     * @param link Link von dem Artikel
67     */
68    public void loadContentFromWebsite(String link) {
69        ArrayList<String> webContent = null;
70        Scanner sc = null;
71
72        try {
73            System.out.println("Loading Content");
74            /*
75             * Baut eine verbindung zu Seite auf und lädt den Inhalt herunter
76             */
77            sc = new Scanner(new URL(link).openStream());
78            webContent = new ArrayList<>();
79            String line;
80
81            //Wenn inhalt gefunden wurde, wird die Schleife ausgeführt
82            while (sc.hasNextLine()) {
83                /*
84                 * Jetzt gehen Zeile für Zeile durch den Inhalt.
85                 * Da wir uns nur für den Artikel interessieren suchen wir nach
                    drei verschiedenen HTML
86                 * Tags, einmal für dei beiden Überschriften und da wo sich der
                    Text befinden.
87                 * Wenn wir diese Zeile gefunden haben entfernen wir anschliessend
                    alle HTML
88                 * Tags aus dieser Zeile, sodass wir nur noch den Text über haben
89                 */
90                line = sc.nextLine();
91                if (line.contains(searchTextSmall)) {
92                    line = sc.nextLine();
93                    //System.out.println(line.stripLeading().replaceAll("&",
                        "&").replaceAll("\\<.*?\\>\\h*", ""));

```



```

94         webContent.add(line.stripLeading().replaceAll("&", "&").
95             replaceAll("\\<.*?\\>\\h*", ""));
96     }
97     if (line.contains(searchHeadline) || line.contains(
98         searchSubtitleSmall)) {
99         //System.out.println(line.stripLeading().replaceAll
100             ("\\<.*?\\>\\h*", ""));
101         webContent.add(line.stripLeading().replaceAll("\\<.*?\\>\\h*",
102             ""));
103     }
104 }
105 System.out.println("Loading done");
106 } catch (IOException e) {
107     System.out.println(ANSI_RED + "Es konnte keine Verbindung hergestellt
108         werden. Bitte überprüfen Sie die angegebene address, oder ihre
109         Internetverbindung!" + ANSI_RESET);
110 } finally {
111     if (sc != null)
112         sc.close();
113 }
114
115 writeToFile(webContent);
116 }
117
118 /**
119  * Hier wird der Artikel in eine txt gespeichert
120  * @param content Text, der von der Webseite runtergeladen wurde
121  */
122 public void writeToFile(ArrayList<String> content) {
123     FileWriter out = null;
124
125     if (content != null && content.size() != 0)
126         try {
127             //Setzt den Dateinamen fest
128             out = new FileWriter(content.get(0).replaceAll("[^a-zA-ZäöüÄÖÜSS%\\
129                 h-]", "") + ".txt", Charset.forName("UTF-8"));
130
131             //Fügt jede Zeile vom content hinzu und fügt ein Zeilenumbruch am
132             ende hinzu, zudem werden die Umlaute ersetzt
133             for (String s : content)
134                 out.append(s.replace("ü", "ue")
135                     .replace("ö", "oe")
136                     .replace("ä", "ae")
137                     .replace("ß", "ss")
138                     .replaceAll("Ü(?=[a-zA-ZäöüSS ])", "Ue")
139                     .replaceAll("Ö(?=[a-zA-ZäöüSS ])", "Oe")
140                     .replaceAll("Ä(?=[a-zA-ZäöüSS ])", "Ae")
141                     + "\n");
142             System.out.println("Data written do file called: " + content.get(0)
143                 .replaceAll("[^a-zA-ZäöüÄÖÜSS%\\h-]", "") + ".txt\n");
144         } catch (IOException e) {
145             System.out.println("Datei konnte nicht erstellt werden");
146         } finally {
147             if (out != null) {
148                 try {
149                     out.close();
150                 } catch (IOException e) {
151                     e.printStackTrace();
152                 }
153             }
154         }
155     }
156     else

```

```

147         System.out.println(ANSI_RED + "Error, kein Inhalt gefunden!" +
148                               ANSI_RESET);
149     }
150     public void readFromFile(String filename) {
151         //Lädt die Datei
152         try (Scanner sc = new Scanner(new FileReader(filename + ".txt"))) {
153             wordsUnsorted = new ArrayList<>();
154             wordsSorted = new ArrayList<>();
155
156             while (sc.hasNextLine()) {
157                 /*
158                  * Läuft durch jede Zeile und entfernt alle unerwünschten Zeichen
159                  * AnschlieSSen wird bei Jeden Leerzeichen ein split gemacht
160                  */
161                 String[] line = sc.nextLine().replaceAll("ä", "").replaceAll("[^a-
162                                     zA-Z\\h-]", "").replaceAll("[\\h]-", " ").split(" ");
163                 for (String words : line)
164                     /*
165                      * Hier wird noch kurz geprüft der String befüllt ist
166                      * AnschlieSSend wird er in einer Collection gespeichert
167                      */
168                     if (!words.equals("")) {
169                         wordsUnsorted.add(words);
170                         wordsSorted.add(words);
171                     }
172                 //Zum schluss wird die Collection noch Alphabetisch sortiert
173                 Collections.sort(wordsSorted);
174                 Collections.sort(wordsSorted, String.CASE_INSENSITIVE_ORDER);
175                 System.out.println("Datei wurde Erfolgreich geladen\n");
176             } catch (IOException e) {
177                 System.out.println(ANSI_RED + "Error, es wurde noch keine Datei geladen
178                                     !" + ANSI_RESET);
179             }
180         }

```

9.4.3 MyException.java

```

1 package chapter_10;
2
3 public class MyException extends Exception {
4
5     @Override
6     public String getMessage() {
7         return "Error, kein gültiger start Parameter, nur true oder false ist
8             erlaubt!";
9     }
10
11 }

```

9.5 Testdokumentation

- Startparameter:
Bei einem inkorrekten Startparameter wird eine Exception geworfen.
- Menüauswahl:
Bei falscher Eingabe wurde eine entsprechende Fehlermeldung ausgegeben.

- Artikel herunterladen:
Wenn der Link nicht existiert kommt eine Fehlermeldung.
- Datei Laden:
Wenn keine Datei gefunden wird mit den Namen kommt eine Fehlermeldung.
- Wörter ausgeben:
Wenn keine Datei geladen wurde, kommt eine entsprechende Fehlermeldung.

9.6 Benutzungshinweise

Bei dem Aufruf des Programmes muss mit true oder false angegeben werden, ob Groß- und Kleinschreibung zu beachten ist. Anschließend kann man in dem Menü auswählen ob man einen Artikel von der Tagesschau herunterladen möchte, eine Datei einlesen oder die Wörter in Alphabetischer Reihenfolge ausgibt.

9.7 Anwendungsbeispiel

```
1  [sebastian@laptop bin]$ java Main
2  Exception in thread "main" chapter_10.MyException: Error, nur true oder false
   erlaubt
3  at chapter_10.Main.main(Main.java:21)
4  [sebastian@laptop bin]$
5  [sebastian@laptop bin]$ java Main
6  Bitte wähle einer der Folgenden Optionen aus:
7  1) Einen Artikel von der Tagesschau herunterladen.
8  2) Eine Datei laden.
9  3) Wörter ausgeben.
10 0) Programm Beenden.
11 1
12 Bitte gebe den Link an:
13 https://www.tagesschau.de/inland/gesellschaft/johnsonjohnson-103.html
14 Loading Content
15 Loading done
16 Data written do file called: Warum punktet der neue Impfstoff.txt
17
18 Bitte wähle einer der Folgenden Optionen aus:
19 1) Einen Artikel von der Tagesschau herunterladen.
20 2) Eine Datei laden.
21 3) Wörter ausgeben.
22 0) Programm Beenden.
23 a
24 Es sind nur Zahlen erlaubt
25 b
26 Es sind nur Zahlen erlaubt
27 2
28 Bitte geben Sie den Datei namen an:
29 Warum punktet der neue Impfstoff
30 Datei wurde Erfolgreich geladen
31
32 Bitte wähle einer der Folgenden Optionen aus:
33 1) Einen Artikel von der Tagesschau herunterladen.
34 2) Eine Datei laden.
35 3) Wörter ausgeben.
36 0) Programm Beenden.
37 3
38 ab 4x
39 aber 4x
40 abgeschwaechten 1x
41 aehnliche 1x
42 aehnliches 1x
```

43 | aendern 1x
44 | Alle 2x
45 | allen 2x
46 | allerdings 1x
47 | Allergien 1x
48 | allergischen 1x
49 | als 2x
50 | also 1x
51 | alsoauch 1x
52 | Altersgruppe 1x
53 | am 1x
54 | amp 1x
55 | an 3x
56 | anderen 3x
57 | anhaelt 1x
58 | ansteckend 1x
59 | Antikoerper 2x
60 | Anwendung 1x
61 | Anzahl 1x
62 | Arzneimittel-Agentur 1x
63 | Arzneimittelhersteller 2x
64 | AstraZeneca 4x
65 | AstraZeneca-Impfstoff 1x
66 | Auch 3x
67 | auch 6x
68 | auf 2x
69 | aus 3x
70 | ausreichende 1x
71 | Ausserdem 1x
72 | basierte 1x
73 | bauen 1x
74 | Bauplan 2x
75 | Bei 8x
76 | bei 8x
77 | beide 1x
78 | beiden 1x
79 | bekommen 1x
80 | beobachteten 1x
81 | bereits 4x
82 | Berichte 1x
83 | Beschwerden 1x
84 | bildet 2x
85 | BioNTechPfizer 5x
86 | BioNTechPfizernbspbei 1x
87 | bisher 1x
88 | bisherigen 1x
89 | bleibt 1x
90 | Brasilien 2x
91 | britischen 1x
92 | Coronavirus 5x
93 | Coronavirus-Erbgutes 1x
94 | Dabei 1x
95 | dafuer 1x
96 | Damit 1x
97 | damit 3x
98 | Darauf 1x
99 | darauf 1x
100 | darueber 1x
101 | Das 4x
102 | das 5x
103 | dass 4x
104 | Datenlage 2x
105 | Dazu 1x

106 dem 4x
107 den 14x
108 denen 1x
109 denn 1x
110 dennoch 1x
111 Der 5x
112 der 25x
113 des 7x
114 deuten 1x
115 Deutschland 4x
116 Die 3x
117 die 23x
118 dies 1x
119 Diese 3x
120 diese 1x
121 direkt 1x
122 Dort 1x
123 dort 1x
124 Dosis 1x
125 drei 2x
126 ebenfalls 1x
127 ein 4x
128 Ein-Mal-Dosis 1x
129 eine 5x
130 einen 3x
131 einer 2x
132 eines 1x
133 eineSchutzwirkung 1x
134 einfachen 1x
135 eingeräumt 1x
136 Einschränkung 1x
137 Einstichstelle 1x
138 Empfehlung 2x
139 empfiehlt 1x
140 empfindlicher 1x
141 entscheidet 1x
142 entsprechende 1x
143 Entwicklung 1x
144 er 1x
145 Erbmaterials 1x
146 Ergebnis 1x
147 Ergebnisse 1x
148 Ergebnissen 1x
149 Erkrankungen 1x
150 erreicht 1x
151 Erste 1x
152 ersten 2x
153 es 5x
154 etwas 1x
155 EU-Arzneimittelbehörde 1x
156 Europäische 1x
157 Europageändert 1x
158 Exakte 1x
159 Faellen 1x
160 Fall 1x
161 flächendeckend 1x
162 fuer 3x
163 funktioniert 1x
164 Gefrierschranktemperaturenstabil 1x
165 gegeben 1x
166 gegen 6x
167 gegenueber 1x
168 gehoeren 1x

169 geimpft 6x
170 Geimpfte 1x
171 geimpfte 1x
172 Geimpften 1x
173 gelagert 1x
174 Gelenkschmerzen 1x
175 genau 1x
176 generelle 1x
177 genetische 2x
178 geringer 1x
179 geringeren 1x
180 getestet 2x
181 gibt 2x
182 gleiche 1x
183 Grad 2x
184 groesseren 1x
185 Grossbritannien 1x
186 grosser 1x
187 Grund 1x
188 haeufigsten 1x
189 haltbar 1x
190 handelt 1x
191 hat 7x
192 hatteBioNTechbekannt 1x
193 hatten 2x
194 heisst 1x
195 hier 1x
196 hindass 1x
197 hingegen 1x
198 hoch 1x
199 hohe 1x
200 ihre 1x
201 ihrer 1x
202 im 2x
203 Immunsystem 2x
204 Impfdosis 1x
205 impfen 1x
206 Impfkommision 2x
207 Impfstoff 11x
208 Impfstoffe 4x
209 Impfstoffen 3x
210 Impfstoffs 1x
211 Impfung 4x
212 In 3x
213 in 20x
214 Infektionen 1x
215 Information 1x
216 Inhaltsstoffe 1x
217 Israel 1x
218 ist 11x
219 Jaehrige 1x
220 Jahren 4x
221 jeden 1x
222 Johnson 18x
223 kam 1x
224 keine 1x
225 klangen 1x
226 klinischen 2x
227 Koch-Institut 1x
228 koennen 2x
229 koennte 1x
230 Koerper 1x
231 kommenden 1x

232 konnten 1x
233 Kopf- 1x
234 Kuehlschranktemperaturen 1x
235 Laendern 1x
236 lag 2x
237 Lagerung 1x
238 lange 3x
239 lassen 1x
240 laufen 2x
241 laut 2x
242 liegen 1x
243 Mal 2x
244 manche 1x
245 mehrere 1x
246 mehreren 1x
247 meisten 1x
248 Menschen 6x
249 menschliche 1x
250 Minus 2x
251 Minusgraden 1x
252 mit 5x
253 mithilfe 1x
254 Moderna 6x
255 Monaten 1x
256 mRNA-Impfstoffen 2x
257 Muedigkeit 1x
258 muessen 1x
259 muss 1x
260 Mutation 1x
261 Mutationen 2x
262 mutierten 1x
263 nach 3x
264 nachgebaut 1x
265 Nachteile 1x
266 neben 1x
267 Nebenwirkungen 2x
268 neuartigen 1x
269 neue 2x
270 nicht 4x
271 niedriger 1x
272 noch 4x
273 noetig 1x
274 normalen 2x
275 Notfallzulassung 1x
276 Nun 1x
277 Nur 1x
278 nur 4x
279 Ob 1x
280 ob 1x
281 Oberflaechenproteine 1x
282 ObTransport 1x
283 oder 4x
284 Personen 1x
285 Praeparat 2x
286 Praeparate 1x
287 Praeparats 1x
288 Prioritaet 1x
289 Probanden 2x
290 Prozent 6x
291 pruefen 1x
292 punktet 1x
293 reagiert 2x
294 Reaktionen 1x

295 RKI 1x
296 Robert 1x
297 schaedlich 1x
298 Schmerzen 1x
299 schnell 1x
300 Schuettelfrost 1x
301 schuetzen 1x
302 schuetzt 1x
303 Schutz 1x
304 Schutzwirkung 3x
305 Schwere 1x
306 schwere 1x
307 schweren 1x
308 sehr 1x
309 sein 1x
310 sich 6x
311 Sicht 1x
312 sie 2x
313 sind 7x
314 so 1x
315 sogenannten 1x
316 soll 1x
317 sollten 1x
318 sondern 1x
319 Staendigen 1x
320 StiKo 2x
321 Studien 5x
322 Studienvorliegen 1x
323 Suedafrika 3x
324 suedafrikanischen 1x
325 Symptomen 1x
326 tatsaechlich 1x
327 Teil 1x
328 Teile 2x
329 Teils 1x
330 Todesfaelle 1x
331 Traegerviren 1x
332 Traegervirus 2x
333 transportiert 1x
334 trotz 1x
335 ueber 1x
336 uebertragen 1x
337 um 1x
338 und 15x
339 ungefaehrlich 1x
340 Unklar 1x
341 unklar 1x
342 Unternehmen 1x
343 unterschiedliche 1x
344 untersucht 1x
345 Untersuchungen 1x
346 urspruengliche 1x
347 USA 1x
348 Variante 1x
349 Varianten 1x
350 Vektorimpfstoff 1x
351 VektorimpfstoffenbspDas 1x
352 verbessert 1x
353 Vergleich 1x
354 vergleichbar 1x
355 verhindert 1x
356 verimpft 2x
357 Verlaeufer 1x


```
358   verpackt 1x
359   Vertraeglichkeit 1x
360   verwenden 1x
361   viele 1x
362   Vier 1x
363   vier 2x
364   Viren 2x
365   Virus 1x
366   von 21x
367   vor 2x
368   Vor- 1x
369   Vorteil 2x
370   Vorteile 1x
371   war 1x
372   Warum 1x
373   weitertragen 1x
374   Welche 3x
375   welche 1x
376   Welcher 1x
377   wenigen 1x
378   weniger 2x
379   wenn 2x
380   werden 8x
381   Wie 3x
382   wie 6x
383   wieder 1x
384   wird 4x
385   Wirken 1x
386   wirksam 1x
387   Wirksamkeit 4x
388   Wirkstoffe 1x
389   wirkt 1x
390   Wirkungsweise 1x
391   wo 1x
392   Wochen 2x
393   worden 2x
394   wurde 3x
395   Zahlen 1x
396   zeigen 2x
397   zeigte 1x
398   Zelle 1x
399   Zellen 1x
400   Zu 1x
401   zu 3x
402   zudem 1x
403   zufolge 2x
404   zugelassen 1x
405   zugelassenen 2x
406   zunaechst 1x
407   zurueckgenommenNun 1x
408   Zusammenhang 1x
409   zwei 1x
410   zweiten 1x
411   Bitte wähle einer der Folgenden Optionen aus:
412   1) Einen Artikel von der Tagesschau herunterladen.
413   2) Eine Datei laden.
414   3) Wörter ausgeben.
415   0) Programm Beenden.
416   0
417   [sebastian@laptop bin]$
```