

Hochschule -

Fakultät IV – Technische Informatik

Modul: Programmieren 1

Professor: -

# Entwicklungsarbeit

von

**Sebastian Schramm** Matrikel-Nr. -

2. Februar 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 3	6
2.1	Teilaufgabe 1 . . . . .	6
2.1.1	Aufgabenstellung . . . . .	6
2.1.2	Anforderungsdefinition . . . . .	6
2.1.3	Entwurf . . . . .	6
2.1.4	Quelltext . . . . .	6
2.1.4.1	Typkonvertierungen.java . . . . .	6
2.1.5	Testdokumentation . . . . .	10
2.1.6	Benutzungshinweise . . . . .	10
2.1.7	Anwendungsbeispiel . . . . .	10
2.2	Teilaufgabe 2 . . . . .	11
2.2.1	Aufgabenstellung . . . . .	11
2.2.2	Anforderungsdefinition . . . . .	11
2.2.3	Entwurf . . . . .	12
2.2.4	Quelltext . . . . .	12
2.2.4.1	Wertebereiche.java . . . . .	12
2.2.5	Testdokumentation . . . . .	13
2.2.6	Benutzungshinweise . . . . .	13
2.2.7	Anwendungsbeispiel . . . . .	13
3	Kapitel 4	13
3.1	Teilaufgabe 1 . . . . .	13
3.1.1	Aufgabenstellung . . . . .	13
3.1.2	Anforderungsdefinition . . . . .	13
3.1.3	Entwurf . . . . .	14
3.1.4	Quellcode . . . . .	14
3.1.4.1	Referenzen.java . . . . .	14
3.1.4.2	Punkt.java . . . . .	15
3.1.5	Testdokumentation . . . . .	16
3.1.6	Benutzungshinweise . . . . .	16
3.1.7	Anwendungsbeispiel . . . . .	16
3.2	Teilaufgabe 2 . . . . .	16
3.2.1	Aufgabenstellung . . . . .	16
3.2.2	Anforderungsdefinition . . . . .	16
3.2.3	Entwurf . . . . .	17
3.2.4	Quellcode . . . . .	17
3.2.4.1	Matrizen.java . . . . .	17
3.2.5	Testdokumentation . . . . .	19
3.2.6	Benutzungshinweise . . . . .	19
3.2.7	Anwendungsbeispiel . . . . .	19
4	Kapitel 5	20
4.1	Teilaufgabe 1 . . . . .	20
4.1.1	Aufgabenstellung . . . . .	20
4.1.2	Anforderungsdefinition . . . . .	20
4.1.3	Entwurf . . . . .	20

4.1.4	Quelltext . . . . .	20
4.1.4.1	Nebeneffekte.java . . . . .	20
4.1.5	Testdokumentation . . . . .	21
4.1.6	Benutzungshinweise . . . . .	21
4.1.7	Anwendungsbeispiel . . . . .	21
4.2	Teilaufgabe 2 . . . . .	21
4.2.1	Aufgabenstellung . . . . .	21
4.2.2	Anforderungsdefinition . . . . .	21
4.2.3	Entwurf . . . . .	21
4.2.4	Quelltext . . . . .	21
4.2.4.1	Operatoren.java . . . . .	21
4.2.5	Testdokumentation . . . . .	23
4.2.6	Benutzungshinweise . . . . .	23
4.2.7	Anwendungsbeispiel . . . . .	24
5	Kapitel 6 . . . . .	25
5.1	Teilaufgabe 1 . . . . .	25
5.1.1	Aufgabenstellung . . . . .	25
5.1.2	Anforderungsdefinition . . . . .	25
5.1.3	Entwurf . . . . .	25
5.1.4	Quelltext . . . . .	26
5.1.4.1	Matrizen.java . . . . .	26
5.1.5	Testdokumentation . . . . .	29
5.1.6	Benutzungshinweise . . . . .	29
5.1.7	Anwendungsbeispiel . . . . .	29
5.2	Teilaufgabe 2 . . . . .	30
5.2.1	Aufgabenstellung . . . . .	30
5.2.2	Anforderungsdefinition . . . . .	30
5.2.3	Entwurf . . . . .	30
5.2.4	Quelltext . . . . .	30
5.2.4.1	Sprunganweisungen.java . . . . .	30
5.2.5	Testdokumentation . . . . .	31
5.2.6	Benutzungshinweise . . . . .	31
5.2.7	Anwendungsbeispiel . . . . .	32
6	Kapitel 7 . . . . .	32
6.1	Aufgabenstellung . . . . .	32
6.2	Anforderungsdefinition . . . . .	32
6.3	Entwurf . . . . .	33
6.4	Quelltext . . . . .	34
6.4.1	Main.java . . . . .	34
6.4.2	Viereck.java . . . . .	34
6.4.3	KonvexesViereck.java . . . . .	34
6.4.4	Trapez.java . . . . .	36
6.4.5	Parallelogramm.java . . . . .	36
6.4.6	Rhombus.java . . . . .	36
6.4.7	Rechteck.java . . . . .	37
6.4.8	Quadrat.java . . . . .	37
6.5	Testdokumentation . . . . .	37
6.6	Benutzungshinweise . . . . .	37
6.7	Anwendungsbeispiel . . . . .	37
7	Kapitel 8 . . . . .	38
7.1	Aufgabenstellung . . . . .	38
7.2	Anforderungsdefinition . . . . .	38
7.3	Entwurf . . . . .	38
7.4	Quelltext . . . . .	38
7.4.1	Main.java . . . . .	38
7.4.2	GeradeZahl.java . . . . .	38

7.4.3	OddException.java . . . . .	39
7.5	Testdokumentation . . . . .	40
7.6	Benutzungshinweise . . . . .	40
7.7	Anwendungsbeispiel . . . . .	40
8	Kapitel 10 . . . . .	40
8.1	Aufgabenstellung . . . . .	40
8.2	Anforderungsdefinition . . . . .	40
8.3	Entwurf . . . . .	41
8.4	Quelltext . . . . .	41
8.4.1	Main.java . . . . .	41
8.4.2	IO.java . . . . .	42
8.4.3	MyException.java . . . . .	45
8.5	Testdokumentation . . . . .	45
8.6	Benutzungshinweise . . . . .	46
8.7	Anwendungsbeispiel . . . . .	46

# 1 Kapitel 1

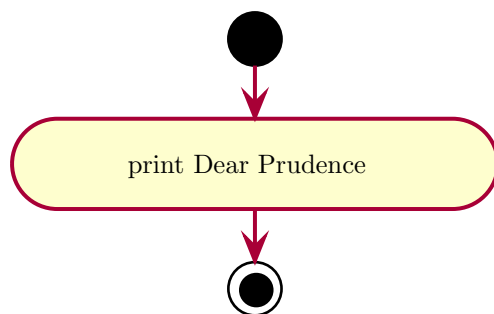
## 1.1 Aufgabenstellung

Wir sollen ein Programm schreiben welches den Text "Dear Prudence" in der Konsole ausgibt. Um uns mit Java vertraut zu machen, sollten wir das erste Programm in der Kommandozeile schreiben. Anschließend mit Javac Kompilieren und mit Java ausführen. Danach öffnen wir unsere IDE, erstellen ein neues Projekt und schreib das selbe Programm diesmal in der IDE.

## 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 werde. Dies war der fall.

## 1.6 Benutzungshinweise

Navigieren Sie in der Kommandozeile zum dem 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 3

### 2.1 Teilaufgabe 1


#### 2.1.1 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.

#### 2.1.2 Anforderungsdefinition

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

#### 2.1.3 Entwurf

 Typkonvertierungen
<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>

#### 2.1.4 Quelltext

##### 2.1.4.1 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
        ausgegeben
45     System.out.println("-----");
46 }
47
48 /**
49  * Eine einschränkende Konvertierung von Short zu Byte
50  * Eine erweiternde Konvertierung von Short zu Double
51  * @param _short
52  */
53 private static void convertShort(short _short) {
54     byte newByte = (byte) _short;
55     int newInt = _short;
56     long newLong = _short;
57     float newFloat = _short;
58     double newDouble = _short;
59
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

```
}
```

### 2.1.5 Testdokumentation

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

### 2.1.6 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.

### 2.1.7 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]$

```

## 2.2 Teilaufgabe 2

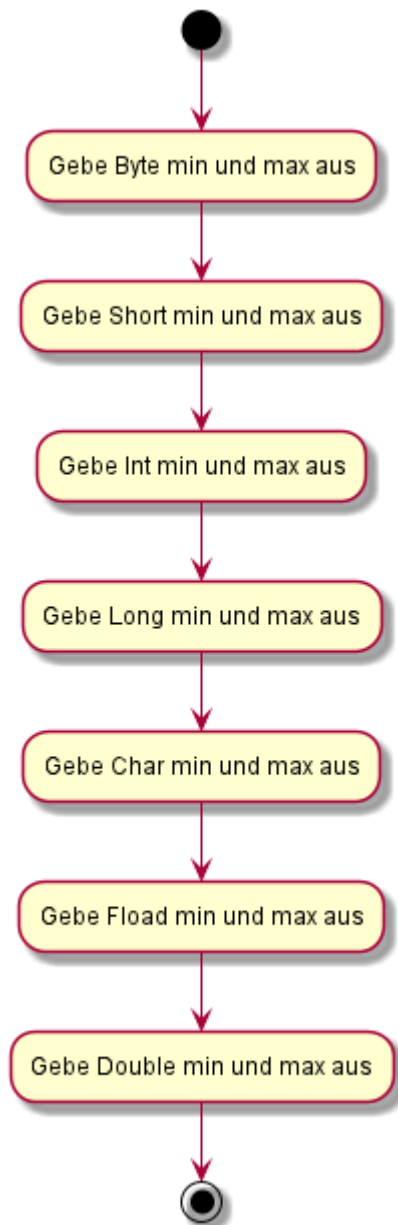
### 2.2.1 Aufgabenstellung

In dieser Teilaufgabe sollen wir ein Programm schreiben welche die Wertebereiche der primitiven Datentypen ausgibt.

### 2.2.2 Anforderungsdefinition

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

### 2.2.3 Entwurf



### 2.2.4 Quelltext

#### 2.2.4.1 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);
14         //Min und Max Value von Short
```

```

14     System.out.println("Short min " + Short.MIN_VALUE + " | Short max " + Short.
15         MAX_VALUE);
16     //Min und Max Value von Integer
17     System.out.println("Integer min " + Integer.MIN_VALUE + " | Integer max " +
18         Integer.MAX_VALUE);
19     //Min und Max Value von Long
20     System.out.println("Long min " + Long.MIN_VALUE + " | Byte Long " + Long.
21         MAX_VALUE);
22
23     //Min und Max Value von Char
24     System.out.println("Char min \u0000 | Char max \uffff");
25
26     //Min und Max Value von Float
27     System.out.println("Float min " + Float.MIN_VALUE + " | Float max " + Float.
28         MAX_VALUE);
29     //Min und Max Value von Double
30     System.out.println("Double min " + Double.MIN_VALUE + " | Double max " + Double
31         .MAX_VALUE);
32 }
33 }

```

### 2.2.5 Testdokumentation

Nach dem Start des Programms sollten die Min und Max werte der einzelnen Datentypen ausgegeben werden, dies war auch der Fall.

### 2.2.6 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.

### 2.2.7 Anwendungsbeispiel

Nach dem man das Programm gestartet hat, 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]$

```

## 3 Kapitel 4

### 3.1 Teilaufgabe 1

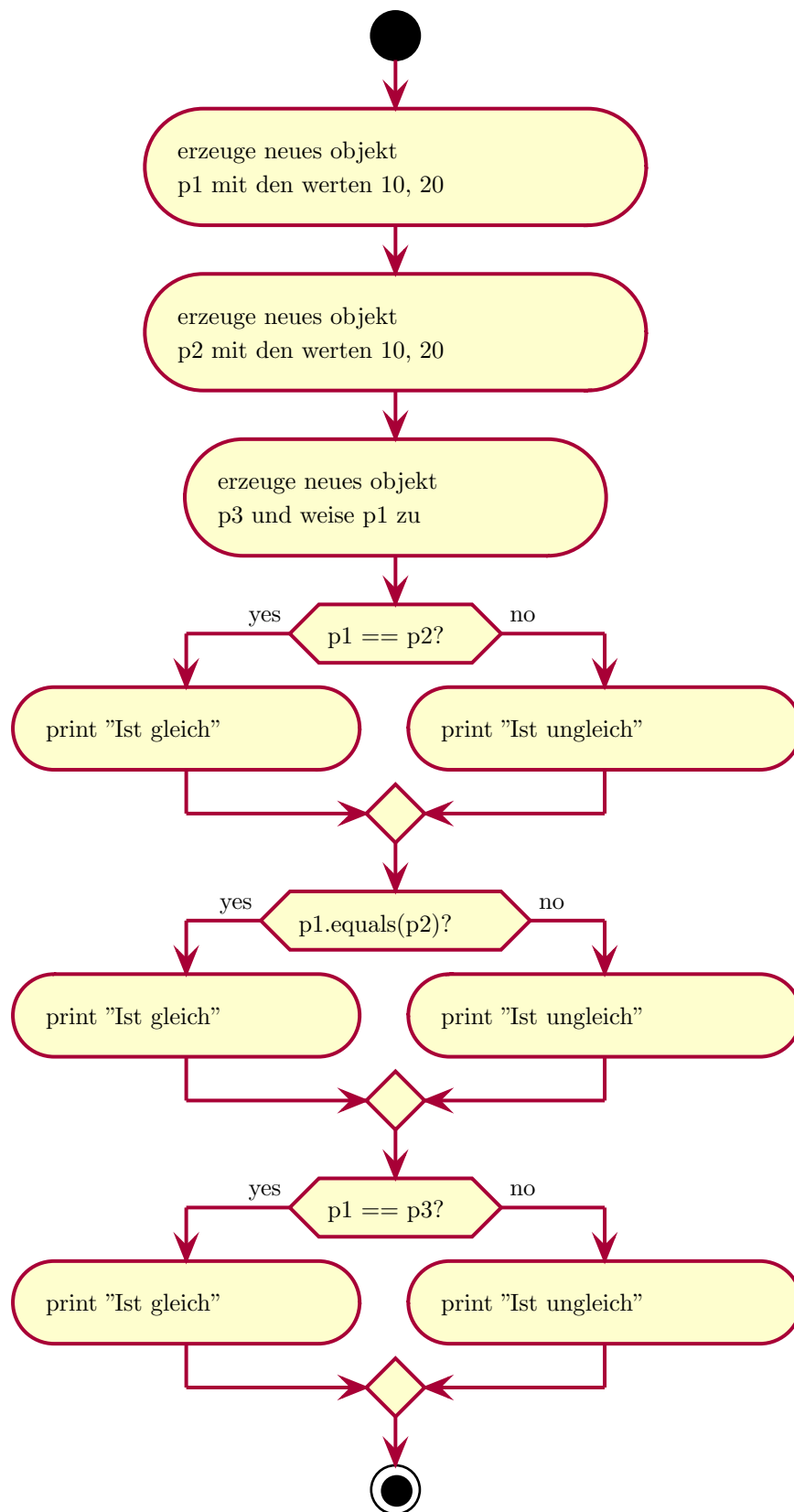
#### 3.1.1 Aufgabenstellung

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

#### 3.1.2 Anforderungsdefinition

1. Prüfe ob zwei Referenzen gleich sind.

### 3.1.3 Entwurf



### 3.1.4 Quellcode

#### 3.1.4.1 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 }

```

### 3.1.4.2 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 int x = 0;
12     private int y = 0;
13
14     public Punkt(int x, int y) {
15         this.x = x;
16         this.y = y;
17     }
18 }

```

### 3.1.5 Testdokumentation

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

### 3.1.6 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.

### 3.1.7 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]$
```

## 3.2 Teilaufgabe 2

### 3.2.1 Aufgabenstellung

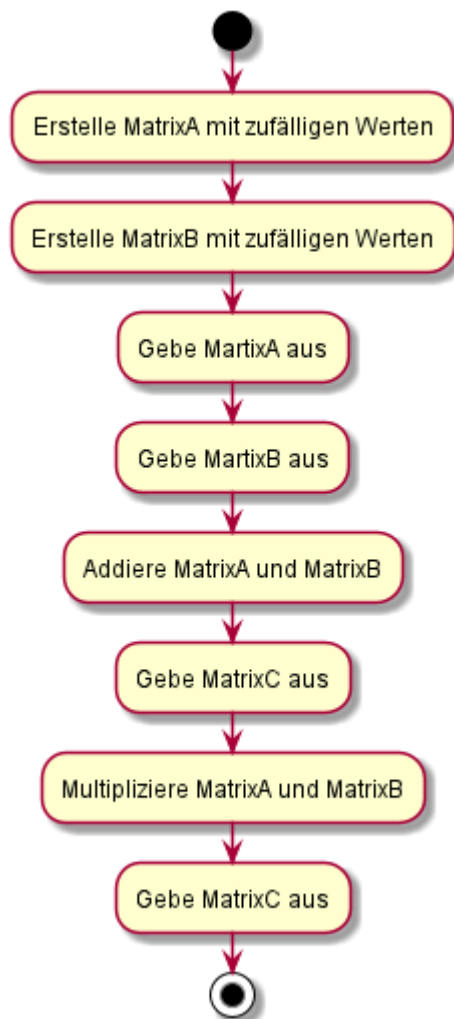
Wir sollen ein Programm schreiben welches 2 nxn Matrizen miteinander Addieren und Multiplizieren kann.

### 3.2.2 Anforderungsdefinition

1. Addiere zwei nxn Matrizen.
2. Multipliziere zwei nxn Matrizen.



### 3.2.3 Entwurf



### 3.2.4 Quellcode

#### 3.2.4.1 Matrizen.java

```
1 package chapter_04;
2
3 /**
4  * Klasse mit der Main-Methode
5  * Addiert und Multipliziert Matrizen
6  * @author Sebastian
7  *
8  */
9 public class Matrizen {
10
11     public static void main(String[] args) {
12         int matrixA[][];
13         int matrixB[][];
14
15         /*
16          * Initialisierungsmethode wird mit dem Wert n aufgerufen.
17          * AnschlieSSend wird diese Matrix erzeugt und mit
18          * zufällig generierten Zahlen befüllt.
19          */
20         matrixA = initialize(2);
21         matrixB = initialize(2);
```

```

22
23  /*
24   * Zuerst werden die Beiden Matrizen A und B jeweils ausgegeben
25   */
26  System.out.println("Matrix A:");
27  printMatrix(matrixA);
28  System.out.println("Matrix B:");
29  printMatrix(matrixB);
30  /*
31   * AnschlieSSend werden die Matrizen hier Addiert
32   */
33  System.out.println("Addition von A und B:");
34  printMatrix(addition(matrixA, matrixB));
35  /*
36   * Und hier Multipliziert
37   */
38  System.out.println("Multiplikation von A und B:");
39  printMatrix(multiplikation(matrixA, matrixB));
40  }
41
42  /**
43   * Initialisierung des Arrays
44   * @param n Die grösSe der nxn Matrix
45   * @return matrix
46   */
47  private static int[][] initialize(int n) {
48      int matrix[][] = new int[n][n];
49      /*
50       * Bei der Initialisierung wird einmal durch das gesamt Array dutch iteriert.
51       * Dabei werden dann mit Math.random() zufällige Zahlen rein geschrieben.
52       */
53      for (int i = 0; i < matrix.length; ++i)
54          for (int l = 0; l < matrix[i].length; ++l)
55              matrix[i][l] = (int) (Math.random() * 100);
56
57      return matrix;
58  }
59
60  /**
61   * Addition der beiden Matrizen A und B
62   * @param matrixA
63   * @param matrixB
64   * @return Gibt ein neues Array mit den Addierten Werten zurück
65   */
66  private static int[][] addition(int matrixA[][], int matrixB[][]) {
67      int matrixAd[][] = new int[matrixA.length][matrixA[0].length]; //Es wird ein
        neues Temporäres Array angelegt
68
69      for (int i = 0; i < matrixA.length; ++i) {
70          for (int n = 0; n < matrixA[i].length; ++n) {
71              matrixAd[i][n] = matrixA[i][n] + matrixB[i][n];
72          }
73      }
74
75      return matrixAd;
76  }
77
78  /**
79   * Multiplikation der beiden Matrizen A und B
80   * @param matrixA
81   * @param matrixB
82   * @return Gibt ein neues Array mit den Multiplizierten Werten zurück
83   */

```

```

84 private static int[][] multiplikation(int matrixA[][], int matrixB[][]) {
85     int matrixMult[][] = new int[matrixB.length][matrixB[0].length];
86
87     for (int HmatrixB = 0; HmatrixB < matrixB.length; ++HmatrixB)
88         for (int WmatrixB = 0; WmatrixB < matrixB[HmatrixB].length; ++WmatrixB)
89             for (int WmatrixA = 0; WmatrixA < matrixB.length; ++WmatrixA)
90                 matrixMult[HmatrixB][WmatrixB] += matrixA[HmatrixB][WmatrixA] * matrixB[
91                     WmatrixA][WmatrixB];
92
93     return matrixMult;
94 }
95
96 /**
97  * Hier wird die Matrix ausgegeben
98  * @param matrix
99  */
100 private static void printMatrix(int matrix[][]) {
101     for (int y[]: matrix) {
102         for (int x: y)
103             System.out.print(x + "\t");
104         System.out.println();
105     }
106     System.out.println();
107 }
108 }

```

### 3.2.5 Testdokumentation

Das Programm hat nach dem Aufruf Zwei 2x2 Matrizen erstellt und initialisiert, anschließend miteinander Addiert und Multipliziert. Dabei kam das Richtige Ergebnis raus.

### 3.2.6 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.

### 3.2.7 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]$

```

## 4 Kapitel 5

### 4.1 Teilaufgabe 1

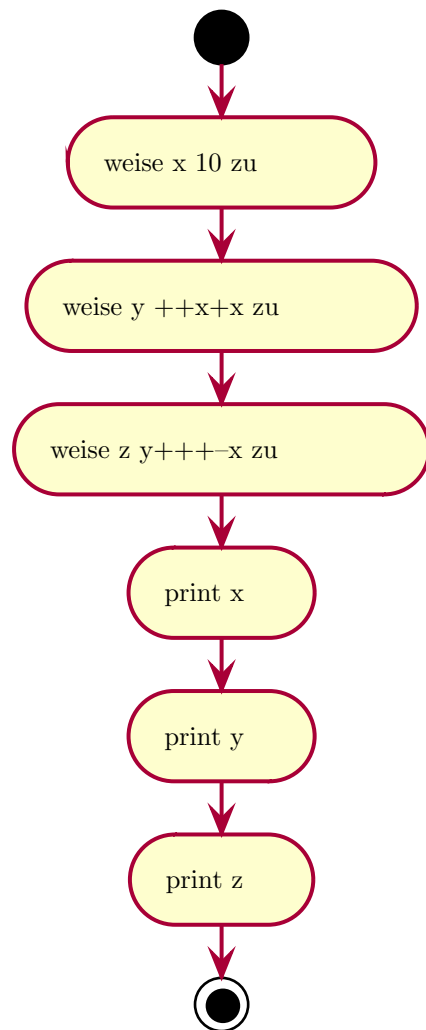
#### 4.1.1 Aufgabenstellung

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

#### 4.1.2 Anforderungsdefinition

1. Nebeneffekte verdeutlichen.

#### 4.1.3 Entwurf



#### 4.1.4 Quelltext

##### 4.1.4.1 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 }

```

#### 4.1.5 Testdokumentation

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

#### 4.1.6 Benutzungshinweise

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

#### 4.1.7 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]$

```

## 4.2 Teilaufgabe 2

### 4.2.1 Aufgabenstellung

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

### 4.2.2 Anforderungsdefinition

1. Verwende alle Operatoren in Java.

### 4.2.3 Entwurf



#### Operatoren

- +main() static
- arithmetisch() static
- inkrement() static
- vergleiche() static
- boolische() static
- bitshifting() static
- zuweisung() static

### 4.2.4 Quelltext

#### 4.2.4.1 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, 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 }

```

#### 4.2.5 Testdokumentation

Es wurden alle Berechnungen korrekt ausgeführt.

#### 4.2.6 Benutzungshinweise

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

#### 4.2.7 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]$
```



## 5 Kapitel 6

### 5.1 Teilaufgabe 1

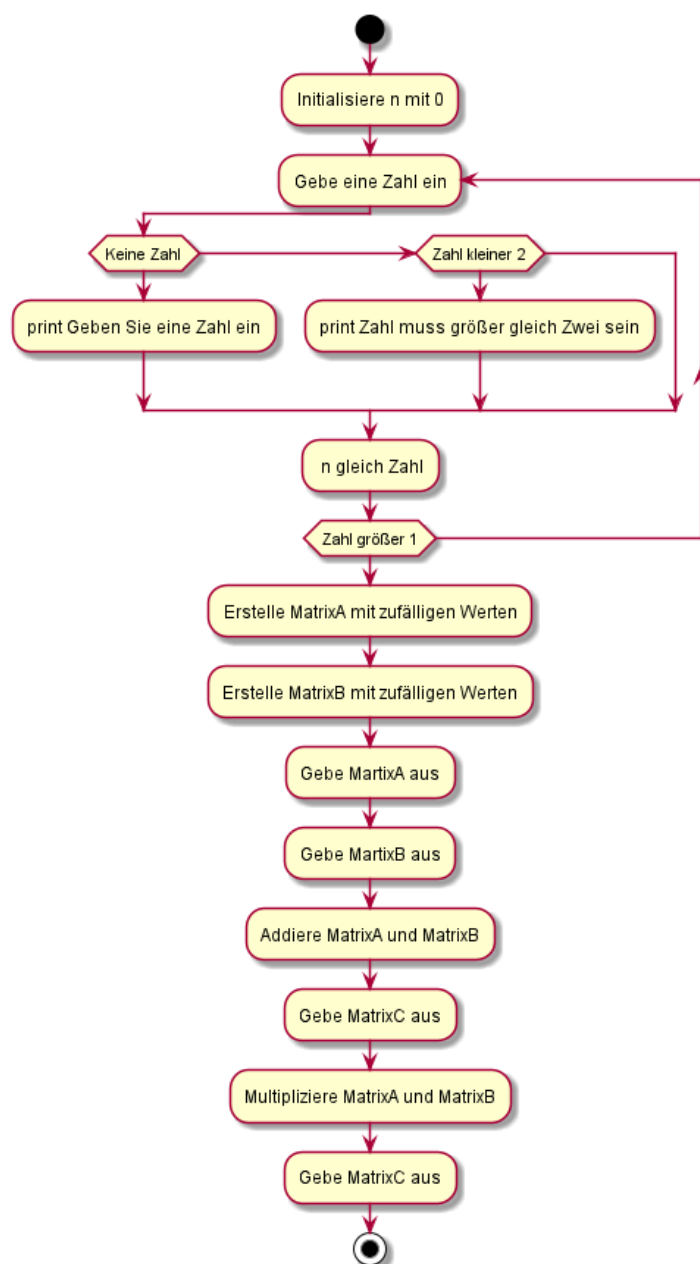
#### 5.1.1 Aufgabenstellung

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 größ gibt an ende der Nutzer ein. Zusätzlich soll noch die Multiplikation der Matrizen auch mit while und do-while gelöst werden.

#### 5.1.2 Anforderungsdefinition

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

#### 5.1.3 Entwurf



## 5.1.4 Quelltext

### 5.1.4.1 Matrizen.java

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

```

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

```

```

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

```

```

182     */
183     private static void printMatrix(int[][] matrix) {
184         for (int[] y : matrix) {
185             for (int x: y)
186                 System.out.print(x + "\t");
187             System.out.println();
188         }
189         System.out.println();
190     }
191 }
192 }

```

### 5.1.5 Testdokumentation

Bei Zahlen die kleiner als 2 waren, sowie Buchstaben, kam es zu einer entsprechenden Fehlermeldung. AnschlieSSend konnte man den Wert erneut eintippen.

### 5.1.6 Benutzungshinweise

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

### 5.1.7 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]$

```

## 5.2 Teilaufgabe 2

### 5.2.1 Aufgabenstellung

In der zweiten Teilaufgabe sollten wir Sprunganweisungen in Java Sinnvoll verdeutlichen.

### 5.2.2 Anforderungsdefinition

1. Verwenden sie Sprunganweisungen.
2. Mindestens ein switch-Anweisung.

### 5.2.3 Entwurf

### 5.2.4 Quelltext

#### 5.2.4.1 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 Passwort 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         /*
63          * Wenn ein Nutzer mit dem angegebenen Passwort
64          * nicht existiert, wird die ID zurückgesetzt
65          * und eine Fehlermeldung wird ausgegeben
66          */
67         if(!userData(id, sc.next())) {
68             System.out.println("Ihre Angaben sind leider falsch, versuchen Sie es
69             erneut.");
70         } else
71             break;
72
73         /*
74          * Die Schleife wird solange durchlaufen, bis sich ein nutzer
75          * erfolgreich angemeldet hat.
76          */
77     } while (true);
78
79     System.out.println("Juhu, Sie haben sich eingeloggt");
80     sc.close();
81 }

```

### 5.2.5 Testdokumentation

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

### 5.2.6 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.

### 5.2.7 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]$
```

## 6 Kapitel 7

### 6.1 Aufgabenstellung

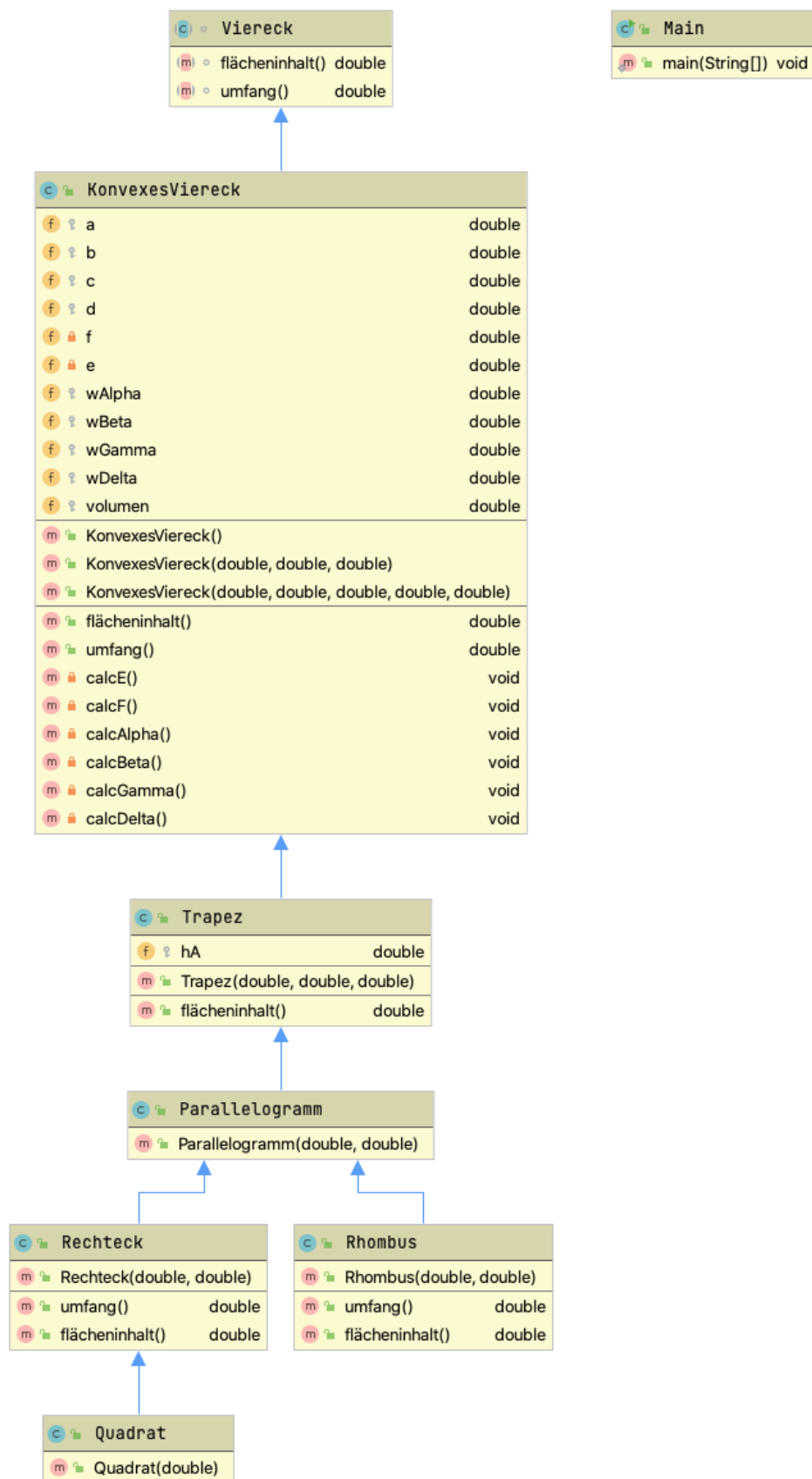
Wie sollen 7 Klassen Definieren, dabei sollen wir uns überlegen welche davon Instanzierbar sind und welche nicht. AnschliesSen soll jede Klasse sinnvolle Methoden und Attribute enthalten.

### 6.2 Anforderungsdefinition

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



## 6.3 Entwurf



Powered by yFiles

## 6.4 Quelltext

### 6.4.1 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         Trapez trapez = new Trapez(15, 5, 5);
9         System.out.println(trapez.umfang() + " " + trapez.flächeninhalt());
10
11         Parallelogramm parallelogramm = new Parallelogramm(20, 5);
12         System.out.println(parallelogramm.umfang() + " " + parallelogramm.
13             flächeninhalt());
14
15         Rhombus rhombus = new Rhombus(5, 3);
16         System.out.println(rhombus.umfang() + " " + rhombus.flächeninhalt());
17
18         Rechteck rechteck = new Rechteck(5, 10);
19         System.out.println(rechteck.umfang() + " " + rechteck.flächeninhalt());
20
21         Quadrat quadrat = new Quadrat(5);
22         System.out.println(quadrat.umfang() + " " + quadrat.flächeninhalt());
23     }
```

### 6.4.2 Viereck.java

```
1 package chapter_07.figures;
2
3 abstract class Viereck {
4
5     abstract double flächeninhalt();
6     abstract double umfang();
7 }
```

### 6.4.3 KonvexesViereck.java

```
1 package chapter_07.figures;
2
3 import static java.lang.Math.PI;
4
5 public class KonvexesViereck extends Viereck {
6     protected double a = 0;
7     protected double b = 0;
8     protected double c = 0;
9     protected double d = 0;
10
11     private double f = 0;
12     private double e = 0;
13
14     protected double wAlpha = 0;
15     protected double wBeta = 0;
16     protected double wGamma = 0;
```

```

17     protected double wDelta = 0;
18
19     protected double volumen;
20
21     public KonvexesViereck() {}
22
23     //     public KonvexesViereck() {
24     //
25     //     }
26
27     //     public KonvexesViereck() {
28     //
29     //     }
30
31     //     public KonvexesViereck() {
32     //
33     //     }
34
35     public KonvexesViereck(double diagonaleF, double diagonaleE, double winkelTheta
36         ) {
37
38     }
39
40     public KonvexesViereck(double a, double b, double c, double d, double
41         winkelAlpha) {
42
43     }
44
45     @Override
46     public double flächeninhalt() {
47         return 0;
48     }
49
50     @Override
51     public double umfang() {
52         return a + b + c + d;
53     }
54
55     private void calcE() {
56         if (wAlpha != 0)
57             f = Math.sqrt(a*a + b*b - 2 * a * b * Math.cos(wBeta) * PI / 180);
58         else
59             f = Math.sqrt(c*c + d*d - 2 * c * d * Math.cos(wDelta) * PI / 180);
60     }
61
62     private void calcF() {
63         if (wAlpha != 0)
64             f = Math.sqrt(a*a + d*d - 2 * a * b * Math.cos(wAlpha) * PI / 180);
65         else
66             f = Math.sqrt(b*b + c*c - 2 * b * c * Math.cos(wGamma) * PI / 180);
67     }
68
69     private void calcAlpha() {
70         wAlpha = Math.acos(((a*a + d*d - f*f) / (2 * a * d)) * PI / 180);
71     }
72
73     private void calcBeta() {
74         wAlpha = Math.acos(((a*a + b*b - e*e) / (2 * a * b)) * PI / 180);
75     }
76
77     private void calcGamma() {
78         wAlpha = Math.acos(((b*b + c*c - f*f) / (2 * b * c)) * PI / 180);
79     }

```

```

78
79     private void calcDelta() {
80         wAlpha = Math.acos(((a*a + d*d - e*e) / (2 * c * d)) * PI / 180);
81     }
82 }

```

#### 6.4.4 Trapez.java

```

1 package chapter_07.figures;
2
3 public class Trapez extends KonvexesViereck {
4     protected double hA;
5
6     public Trapez(double a, double c, double hA) {
7         this.a = a;
8         this.c = c;
9         this.hA = hA;
10    }
11
12    @Override
13    public double flächeninhalt() {
14        return ((a + c) * hA) / 2;
15    }
16
17 }

```

#### 6.4.5 Parallelogramm.java

```

1 package chapter_07.figures;
2
3 public class Parallelogramm extends Trapez {
4
5     public Parallelogramm(double a, double hA) {
6         super(a, 0, hA);
7     }
8 }

```

#### 6.4.6 Rhombus.java

```

1 package chapter_07.figures;
2
3 public class Rhombus extends Parallelogramm {
4
5     public Rhombus(double a, double hA) {
6         super(a, hA);
7     }
8
9     @Override
10    public double umfang() {
11        return a * 4;
12    }
13
14    @Override
15    public double flächeninhalt() {
16        return a * hA;
17    }

```

```
18 }
```

#### 6.4.7 Rechteck.java

```
1 package chapter_07.figures;
2
3 public class Rechteck extends Parallelogramm {
4
5     public Rechteck(double a, double b) {
6         super(a, b);
7         this.b = b;
8     }
9
10    @Override
11    public double umfang() {
12        return (a + b) * 2;
13    }
14
15    @Override
16    public double flächeninhalt() {
17        return a * b;
18    }
19 }
```

#### 6.4.8 Quadrat.java

```
1 package chapter_07.figures;
2
3 public class Quadrat extends Rechteck {
4
5     public Quadrat(double a) {
6         super(a, a);
7     }
8 }
```

### 6.5 Testdokumentation

?

### 6.6 Benutzungshinweise

?

### 6.7 Anwendungsbeispiel

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

```
1 [sebastian@laptop bin]$ java Main
2 20.0 50.0
3 20.0 50.0
4 20.0 15.0
5 30.0 50.0
6 20.0 25.0
7 [sebastian@laptop bin]$
```

## 7 Kapitel 8

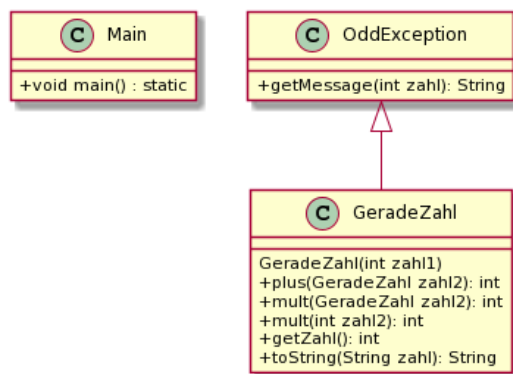
### 7.1 Aufgabenstellung

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.

### 7.2 Anforderungsdefinition

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

### 7.3 Entwurf



### 7.4 Quelltext

#### 7.4.1 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 }
```

#### 7.4.2 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
        handelt
8      * @param zahl1
9      */
10    public GeradeZahl(int zahl1) throws OddException {
11        this.zahl1 = zahl1;
12
13        //Wenn die Zahl ungerade ist, wird eine Exception geworfen
14        if (zahl1%2 == 1) throw new OddException(zahl1);
15    }
16
17    /**
18     * Addiert Zwei GeradeZahl objekte miteinander
19     * @param zahl2
20     * @return Gibt eine int Zahl zurück
21     */
22    public int plus(GeradeZahl zahl2) {
23        return zahl1 + zahl2.getZahl();
24    }
25
26    /**
27     * Multipliziert Zwei GeradeZahl objekte miteinander
28     * @param zahl2
29     * @return Gibt eine int Zahl zurück
30     */
31    public int mult(GeradeZahl zahl2) {
32        return zahl1 * zahl2.getZahl();
33    }
34
35    /**
36     * Multipliziert ein GeradeZahl objekte mit einer int Zahl
37     * @param zahl2
38     * @return Gibt eine int Zahl zurück
39     */
40    public int mult(int zahl2) {
41        return zahl1 * zahl2;
42    }
43
44    /**
45     * @return Liefert die Zahl zurück
46     */
47    public int getZahl() {
48        return zahl1;
49    }
50
51    /**
52     * Bei einem Print wird diese Methode ausgeführt
53     * @return Liefert einen String mit der Zahl zurück
54     */
55    @Override
56    public String toString() {
57        return "" + getZahl();
58    }
59 }

```

### 7.4.3 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 }

```

## 7.5 Testdokumentation

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

## 7.6 Benutzungshinweise

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

## 7.7 Anwendungsbeispiel

```

1 [sebastian@laptop bin]$ java Main
2 Error, 21 ist keine Gerade Zahl! Die Zahl wurde um Eins erhöht.
3 Error, 13 ist keine Gerade Zahl! Die Zahl wurde um Eins erhöht.
4 Zahl 1 = 220
5 Zahl 2 = 22
6 Zahl 3 = 660
7 Zahl 4 = 24
8 [sebastian@laptop bin]$

```

# 8 Kapitel 10

## 8.1 Aufgabenstellung

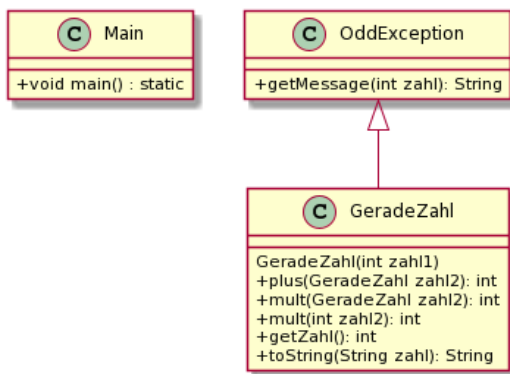
Wir sollen ein Programm schreiben Welches ein Artikel von der Tagesschau herunterlädt und anschließend in einer Datei abspeichert. Der Nutzer kann angeben, welche Datei er einlesen möchte, anschließend kann er die gezählten Wörter in Alphabetischer Reihenfolge ausgeben lassen. Zusätzlich soll über ein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

## 8.2 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.



## 8.3 Entwurf



## 8.4 Quelltext

### 8.4.1 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             System.out.println("2) Eine Datei laden.");
36             System.out.println("3) Wörter ausgeben.");
37             System.out.println("0) Programm Beenden.");
38             do {
39                 /*
40                  * Wenn der Nutzer einen Buchstaben oder eine Zahl außerhalb des
41                  * wertebereiches
42                  * angibt wird eine entsprechende Fehlermeldung angegeben.
```

```

40         */
41         if (!sc.hasNextInt()) {
42             System.out.println("Es sind nur Zahlen erlaubt");
43             sc.next();
44         } else if ((option = sc.nextInt()) > 4 || option < 0)
45             System.out.println("Zahl auSSerhalb des Bereiches!");
46         else
47             break;
48     } while (true);
49
50     //Bei einer erfolgreichen eingabe, wird eine dieser Optionen ausgeführt
51     switch (option) {
52         case 1:
53             System.out.println("Bitte gebe den Link an:");
54             io.loadContentFromWebsite(input());
55             break;
56         case 2:
57             System.out.println("Bitte geben Sie den Datei namen an:");
58             io.readFromFile(input());
59             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 }

```

#### 8.4.2 IO.java

```

1 package chapter_10;
2
3 import java.io.*;
4 import java.net.URL;
5 import java.util.ArrayList;
6 import java.util.Collections;
7 import java.util.Scanner;
8
9 public class IO{
10     private ArrayList<String> wordsUnsorted;
11     private ArrayList<String> wordsSorted;
12     private final boolean caseSensitive;
13
14     private static final String searchHeadline = "span class=\"headline\"";
15     private static final String searchSubtitleSmall = "class=\"subtitle small \"";
16     private static final String searchTextSmall = "class=\"text small\"";
17     private static final String ANSI_RESET = "\u001B[0m";
18     private static final String ANSI_RED = "\u001B[31m";
19
20     public IO(boolean caseSensitive){
21         this.caseSensitive = caseSensitive;

```

```

22     }
23
24     /**
25      * Listet die einzelnen wörter aus dem ausgewählten Artikel auf und gibt
26      * zusätzlich der Anzahl an
27      * @param sorted Noch keinen nutzen
28      */
29     public void printContent(boolean sorted) {
30         //Zuerst wird geprüft ob eine Datei eingelesen wurde
31         if (wordsSorted != null) {
32             ArrayList<String> tmpWords = new ArrayList<>(wordsSorted);
33             //Der erste eintrag wird in einen String geschrieben
34             String currentWord = tmpWords.get(0);
35             int countWords = 0;
36
37             for (int i = 0; i < tmpWords.size(); ++i) {
38                 /*
39                  * Falls das Wort übereinstimmt mit dem zwischengespeicherten word,
40                  * wird der
41                  * Counter um eins erhöht und der aktuelle eintrag gelöscht
42                  */
43                 if (caseSensitive ? currentWord.equals(tmpWords.get(i)) :
44                     currentWord.equalsIgnoreCase(tmpWords.get(i))) {
45                     ++countWords;
46                     tmpWords.remove(i--);
47                 }
48                 /*
49                  * Falls die Wörter unterschiedlich sind, wird das aktuelle Word
50                  * ausgegeben plus
51                  * deren Anzahl, anschliessend wird aktuelle word mit dem neuen
52                  * ersetzt und der
53                  * counter zurückgesetzt
54                  */
55                 } else {
56                     System.out.println(currentWord + " " + countWords + "x");
57                     currentWord = tmpWords.get(i);
58                     countWords = 1;
59                 }
60             }
61         } else
62             //Ausgabe, falls keine Datei eingelesen wurde
63             System.out.println(ANSI_RED + "Es wurde noch keine Datei Geladen!" +
64                 ANSI_RESET);
65     }
66
67     /**
68      * Lädt einen Artikel von der Tagesschau herunter
69      * @param link Link von dem Artikel
70      */
71     public void loadContentFromWebsite(String link) {
72         ArrayList<String> webContent = null;
73         Scanner sc = null;
74
75         try {
76             System.out.println("Loading Content");
77             /*
78              * Baut eine verbindung zu Seite auf und lädt den Inhalt herunter
79              */
80             sc = new Scanner(new URL(link).openStream());
81             webContent = new ArrayList<>();
82             String line;
83
84             //Wenn inhalt gefunden wurde, wird die Schleife ausgeführt
85             while (sc.hasNextLine()) {

```

```

80      /*
81      * Jetzt gehen Zeile für Zeile durch den Inhalt.
82      * Da wir uns nur für den Artikel interessieren suchen wir nach
83      * drei verschiedenen HTML
84      * Tags, einmal für die beiden Überschriften und da wo sich der
85      * Text befinden.
86      * Wenn wir diese Zeile gefunden haben entfernen wir anschließend
87      * alle HTML
88      * Tags aus dieser Zeile, sodass wir nur noch den Text über haben
89      */
90      line = sc.nextLine();
91      if (line.contains(searchHeadline) || line.contains(
92          searchSubtitleSmall) || line.contains(searchTextSmall)) {
93          webContent.add(line.replaceAll("\\<.*?\\>\\h*", ""));
94      }
95      }
96      System.out.println("Loading done");
97      } catch (IOException e) {
98          System.out.println(ANSI_RED + "Es konnte keine Verbindung hergestellt
99          werden. Bitte überprüfen Sie die angegebene address, oder ihre
100          Internetverbindung!" + ANSI_RESET);
101      } finally {
102          if (sc != null)
103              sc.close();
104      }
105      writeToFile(webContent);
106      }
107      /**
108      * Hier wird der Artikel in eine txt gespeichert
109      * @param content Text, der von der Webseite runtergeladen wurde
110      */
111      public void writeToFile(ArrayList<String> content) {
112          FileWriter out = null;
113          if (content != null)
114              try {
115                  //Setzt den Dateinamen fest
116                  out = new FileWriter(content.get(0).replaceAll("[^a-zA-ZäöüÄÖÜSS\\h
117                      -]", "") + ".txt");
118                  //Fügt jede Zeile vom content hinzu und fügt ein Zeilenumbruch am
119                  ende hinzu
120                  for (String s : content)
121                      out.append(s + "\n");
122                  System.out.println("Data written do file called: " + content.get(0)
123                      .replaceAll("[^a-zA-ZäöüÄÖÜSS\\h-]", "") + ".txt\n");
124              } catch (IOException e) {
125                  System.out.println("Datei konnte nicht erstellt werden");
126              } finally {
127                  if (out != null) {
128                      try {
129                          out.close();
130                      } catch (IOException e) {
131                          e.printStackTrace();
132                      }
133                  }
134              }
135          else
136              System.out.println(ANSI_RED + "Error, kein Inhalt gefunden!" +
137                  ANSI_RESET);
138      }

```

```

133
134 public void readFromFile(String filename) {
135     //Lädt die Datei
136     try (Scanner sc = new Scanner(new FileReader(filename + ".txt"))) {
137         wordsUnsorted = new ArrayList<>();
138         wordsSorted = new ArrayList<>();
139
140         while (sc.hasNextLine()) {
141             /*
142              * Läuft durch jede Zeile und entfernt alle unerwünschten Zeichen
143              * AnschliesSen wird bei Jeden Leerzeichen ein split gemacht
144              */
145             String[] line = sc.nextLine().replaceAll("[\\h]-", " ").replaceAll(
146                 "ä", "").replaceAll("[^a-zA-ZäöüÄÖÜSS\\h-]", "").split(" ");
147             for (String words : line)
148                 /*
149                  * Hier wird noch kurz geprüft der String befüllt ist
150                  * AnschliesSend wird er in einer Collection gespeichert
151                  */
152                 if (!words.equals("")) {
153                     wordsUnsorted.add(words);
154                     wordsSorted.add(words);
155                 }
156             //Zum schluss wird die Collection noch Alphabetisch sortiert
157             Collections.sort(wordsSorted);
158             Collections.sort(wordsSorted, String.CASE_INSENSITIVE_ORDER);
159             System.out.println("Datei wurde Erfolgreich geladen\n");
160         } catch (IOException e) {
161             System.out.println(ANSI_RED + "Error, es wurde noch keine Datei geladen
162                 !" + ANSI_RESET);
163         }
164     }

```

### 8.4.3 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 }

```

## 8.5 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.

## 8.6 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.

## 8.7 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 lasdfl
14 Loading Content
15 Es konnte keine Verbindung hergestellt werden. Bitte überprüfen Sie die
   angegebene address, oder ihre Internetverbindung!
16 Error, kein Inhalt gefunden!
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 1
24 Bitte gebe den Link an:
25 https://www.tagesschau.de/inland/bundeslaender-schulen-corona-101.html
26 Loading Content
27 Loading done
28 Data written do file called: Bei Schulen gehen die Länder Sonderwege.txt
29
30 Bitte wähle einer der Folgenden Optionen aus:
31 1) Einen Artikel von der Tagesschau herunterladen.
32 2) Eine Datei laden.
33 3) Wörter ausgeben.
34 0) Programm Beenden.
35 2
36 Bitte geben Sie den Dateinamen an:
37 Test
38 Error, es wurde noch keine Datei geladen!
39
40 Bitte wähle einer der Folgenden Optionen aus:
41 1) Einen Artikel von der Tagesschau herunterladen.
42 2) Eine Datei laden.
43 3) Wörter ausgeben.
44 0) Programm Beenden.
45 3

```

```

46 | Es wurde noch keine Datei Geladen!
47 |
48 | Bitte wähle einer der Folgenden Optionen aus:
49 | 1) Einen Artikel von der Tagesschau herunterladen.
50 | 2) Eine Datei laden.
51 | 3) Wörter ausgeben.
52 | 0) Programm Beenden.
53 | 2
54 | Bitte geben Sie den Dateinamen an:
55 | Bei Schulen gehen die Länder Sonderwege
56 | Datei wurde Erfolgreich geladen
57 |
58 | Bitte wähle einer der Folgenden Optionen aus:
59 | 1) Einen Artikel von der Tagesschau herunterladen.
60 | 2) Eine Datei laden.
61 | 3) Wörter ausgeben.
62 | 0) Programm Beenden.
63 | 3
64 | ab 6x
65 | aber 1x
66 | Abitur 1x
67 | Abschlussklassen 4x
68 | Abschlussprüfungen 1x
69 | Abständen 1x
70 | abzumelden 1x
71 | aktiv 1x
72 | alle 2x
73 | allen 1x
74 | als 2x
75 | also 1x
76 | alten 1x
77 | am 1x
78 | an 6x
79 | andere 1x
80 | anderen 1x
81 | anders 2x
82 | Anfang 1x
83 | angeboten 1x
84 | Anmeldequote 1x
85 | anschlieSSend 1x
86 | Anschluss 1x
87 | anzubieten 1x
88 | Appelle 1x
89 | Auch 1x
90 | auch 5x
91 | auf 5x
92 | aufgehoben 1x
93 | aus 2x
94 | ausgesetzt 2x
95 | Ausnahmen 1x
96 | ausnehme 1x
97 | Baden-Württembergsgrüner 1x
98 | Bayernlässt 1x
99 | bedeute 1x
100 | begrüSSte 1x
101 | Bei 3x
102 | bei 1x
103 | beide 1x
104 | beim 1x
105 | bekommen 1x
106 | berechtigt 1x
107 | beschlossen 1x
108 | beschlossene 1x

```

109 | Beschlusses 1x  
110 | Beschlüsse 1x  
111 | beschränkt 1x  
112 | besonders 2x  
113 | besser 1x  
114 | bestehende 1x  
115 | betonte 1x  
116 | betreffe 1x  
117 | Betreuung 1x  
118 | Betreuungsmöglichkeit 1x  
119 | betroffene 1x  
120 | Bildung 1x  
121 | bis 4x  
122 | bisher 2x  
123 | bisherigen 1x  
124 | bislang 1x  
125 | bisschen 1x  
126 | bleibe 1x  
127 | bleibt 1x  
128 | Blick 1x  
129 | Brandenburg 1x  
130 | Britta 1x  
131 | Bund 2x  
132 | Bund-Länder-Beratungen 1x  
133 | Bund-Länder-Beschlüsse 1x  
134 | Bund-Länder-Beschlüssen 1x  
135 | Bund-Länder-Treffen 1x  
136 | Bundes 1x  
137 | Bundesländer 1x  
138 | CDU 1x  
139 | Chance 1x  
140 | Christian 1x  
141 | Corona-Pandemie 1x  
142 | Corona-Schutz-Verordnung 1x  
143 | CSU 1x  
144 | da 1x  
145 | Dabei 1x  
146 | dabei 1x  
147 | daher 1x  
148 | damit 1x  
149 | dann 1x  
150 | Das 3x  
151 | das 3x  
152 | dass 4x  
153 | dem 5x  
154 | demnach 1x  
155 | Den 1x  
156 | den 5x  
157 | denen 1x  
158 | dennoch 1x  
159 | Der 2x  
160 | der 18x  
161 | Deren 1x  
162 | des 3x  
163 | desto 1x  
164 | deutlich 2x  
165 | Die 6x  
166 | die 19x  
167 | Dienstag 1x  
168 | Dienstagabend 1x  
169 | Dies 2x  
170 | dies 2x  
171 | diese 1x



172	Distanzunterricht 1x
173	Dresden 1x
174	Dreyer 1x
175	eben 1x
176	ebenfalls 2x
177	eher 1x
178	eigene 1x
179	eigentlich 2x
180	Ein 1x
181	ein 2x
182	Eine 1x
183	eine 4x
184	einem 1x
185	einen 3x
186	einer 1x
187	Einfluss 1x
188	eingeschränkt 1x
189	eingeschränkten 2x
190	einiger 1x
191	einmalig 1x
192	einschränkte 1x
193	einzelne 1x
194	Eltern 4x
195	Entscheidung 1x
196	Entscheidungen 1x
197	entschied 1x
198	entsprechenden 1x
199	Er 3x
200	er 1x
201	erfolgen 1x
202	erhalten 1x
203	erklärte 1x
204	erläuterten 1x
205	Ermahnungen 1x
206	Ernst 1x
207	erste 2x
208	erweiterte 1x
209	Es 2x
210	es 6x
211	fast 1x
212	Februar 8x
213	Fernunterricht 1x
214	fest 1x
215	folgen 1x
216	Formblatt 1x
217	fortgesetzt 1x
218	Fragen 1x
219	fünften 1x
220	Für 1x
221	für 11x
222	gar 1x
223	geben 1x
224	gebeten 1x
225	gebracht 1x
226	gehen 4x
227	geht 1x
228	gehört 1x
229	genug 1x
230	geplant 1x
231	gerade 1x
232	gesamte 1x
233	geteilten 1x
234	gibt 1x

235	Grant 1x
236	Grundschulen 3x
237	Grundschüler 2x
238	Grundschülern 1x
239	grundsätzlich 2x
240	haben 3x
241	Hamburgverschärft 1x
242	hart 1x
243	hatten 1x
244	Hause 1x
245	Hendrik 1x
246	hier 2x
247	hoffe 1x
248	hohe 1x
249	hänge 1x
250	Höhe 1x
251	ihre 4x
252	im 9x
253	immer 1x
254	In 1x
255	in 16x
256	Infektionsgeschehen 2x
257	Infektionslage 1x
258	Infektionszahlen 1x
259	Inzidenzwert 1x
260	inzwischen 1x
261	ist 2x
262	Jahrgänge 1x
263	Januar 1x
264	je 2x
265	jedoch 2x
266	Jugendlichen 1x
267	Kabinettsitzung 2x
268	Kanzlerin 1x
269	Kaum 1x
270	Kein 1x
271	keine 2x
272	Kern 1x
273	Kind 1x
274	Kinder 4x
275	Kitas 10x
276	Klasse 2x
277	Klassen 1x
278	Klassenstufen 1x
279	kleinen 1x
280	KMK 1x
281	kommenden 1x
282	Konkret 1x
283	Kretschmann 1x
284	Kritikerin 1x
285	Kultusminister 2x
286	Kultusministerkonferenz 1x
287	Kurs 1x
288	können 1x
289	kündigte 1x
290	lassen 1x
291	liegen 1x
292	Lockdowns 1x
293	Länder 6x
294	Ländern 1x
295	Ländersache 1x
296	machte 1x
297	MainzIn 1x

298 Malu 1x  
 299 manche 1x  
 300 Manuela 1x  
 301 Markus 1x  
 302 MaSSnahme 1x  
 303 MaSSnahmen 1x  
 304 Mecklenburg-Vorpommern 1x  
 305 Mecklenburg-Vorpommerns 1x  
 306 Merkel 1x  
 307 Ministerpräsident 2x  
 308 Ministerpräsidenten 1x  
 309 Ministerpräsidentin 2x  
 310 Ministerpräsidentinnen 1x  
 311 mit 5x  
 312 möglich 1x  
 313 Möglichkeit 1x  
 314 mühsam 1x  
 315 München 1x  
 316 nach 4x  
 317 NDR 1x  
 318 neuen 1x  
 319 nicht 5x  
 320 Niedersachsen 1x  
 321 Niedersachsenbefreit 1x  
 322 noch 4x  
 323 Notbetreuung 1x  
 324 Notfall 1x  
 325 nur 6x  
 326 oder 2x  
 327 offen 1x  
 328 per 1x  
 329 Piwarz 1x  
 330 planen 1x  
 331 Pläne 1x  
 332 positioniert 1x  
 333 Präsenzpflicht 4x  
 334 Präsenzunterricht 6x  
 335 pädagogische 1x  
 336 rechtfertigen 1x  
 337 reduziert 1x  
 338 Regelbetreuung 1x  
 339 Regelbetrieb 1x  
 340 Regeln 2x  
 341 Regelung 1x  
 342 Regelungen 1x  
 343 Regierungschef 1x  
 344 restriktive 2x  
 345 restriktiven 1x  
 346 Rheinland-Pfalz 1x  
 347 Routine 1x  
 348 Runde 1x  
 349 Sachsenhält 1x  
 350 sagte 3x  
 351 schicken 1x  
 352 schon 1x  
 353 schrittweise 1x  
 354 schränkte 1x  
 355 Schulbetrieb 1x  
 356 Schulen 15x  
 357 Schulschließung 1x  
 358 Schwesig 2x  
 359 Schüler 5x  
 360 Schülern 1x

361 sehr 1x  
362 sei 4x  
363 seien 1x  
364 sein 1x  
365 seine 1x  
366 seiner 1x  
367 Senat 1x  
368 sich 3x  
369 sie 1x  
370 siebten 1x  
371 sind 3x  
372 Situation 1x  
373 so 3x  
374 soll 5x  
375 sollte 1x  
376 sollen 2x  
377 Sollten 1x  
378 sollten 4x  
379 Sonderwege 1x  
380 sorgfältig 1x  
381 SPD 3x  
382 SPD-Politikerin 1x  
383 später 1x  
384 stark 1x  
385 statt 1x  
386 stattfinden 1x  
387 steigen 1x  
388 Stephan 1x  
389 Söder 1x  
390 tageweise 1x  
391 Teil 1x  
392 Tisch 1x  
393 Tonne 2x  
394 trotz 1x  
395 tue 1x  
396 umsetzen 1x  
397 Umsetzung 2x  
398 Und 1x  
399 und 18x  
400 unter 1x  
401 Unterricht 1x  
402 unterrichtet 1x  
403 unterschiedlich 1x  
404 unterstrich 1x  
405 Unterstützung 1x  
406 vehemente 1x  
407 vereinbart 1x  
408 verkünden 1x  
409 verlängerten 1x  
410 verlängerter 1x  
411 verschärft 1x  
412 veränderbar 1x  
413 Viele 1x  
414 vom 5x  
415 Von 1x  
416 von 6x  
417 vorgehen 1x  
418 vorsichtig 1x  
419 Vorsitzende 1x  
420 vorzubereiten 1x  
421 war 1x  
422 warum 1x  
423 Wechsel 1x

```
424 Wechselunterricht 3x
425 Wege 1x
426 Weil 1x
427 weil 1x
428 weiter 2x
429 Weitere 1x
430 weitere 1x
431 weiterhin 2x
432 weniger 1x
433 Wenn 1x
434 wenn 2x
435 Wer 1x
436 werde 1x
437 werden 9x
438 wie 1x
439 wieder 4x
440 Winfried 1x
441 wir 1x
442 wird 1x
443 Woche 1x
444 wollen 1x
445 wonach 1x
446 wurde 1x
447 Zahl 1x
448 Zahlen 1x
449 zeigte 1x
450 Zeit 1x
451 zu 5x
452 zugehen 1x
453 zulasse 1x
454 zulässt 1x
455 zum 4x
456 zumindest 1x
457 zur 1x
458 zurück 1x
459 Öffnen 2x
460 Öffnungsschritte 1x
461 Bitte wähle einer der Folgenden Optionen aus:
462 1) Einen Artikel von der Tagesschau herunterladen.
463 2) Eine Datei laden.
464 3) Wörter ausgeben.
465 0) Programm Beenden.
466 0
467 [sebastian@laptop bin]$
```