

Hochschule -

Fakultät IV – Technische Informatik

Modul: Programmieren 1

Professor: -

Entwicklungsarbeit

von

Sebastian Schramm Matrikel-Nr. -

17. Januar 2021

Inhaltsverzeichnis

1	Kapitel 1	4
1.1	Aufgabenstellung	4
1.2	Anforderungsdefinition	4
1.3	Entwurf	4
1.4	Quellcode	4
1.4.1	Main.java	4
1.5	Testdokumentation	4
1.6	Benutzungshinweise	4
1.7	Anwendungsbeispiel	5
2	Kapitel 3	5
2.1	Teilaufgabe 1	5
2.1.1	Aufgabenstellung	5
2.1.2	Anforderungsdefinition	5
2.1.3	Entwurf	5
2.1.4	Quelltext	5
2.1.4.1	Typkonvertierungen.java	5
2.1.5	Testdokumentation	9
2.1.6	Benutzungshinweise	9
2.1.7	Anwendungsbeispiel	9
2.2	Teilaufgabe 2	10
2.2.1	Aufgabenstellung	10
2.2.2	Anforderungsdefinition	10
2.2.3	Entwurf	11
2.2.4	Quelltext	11
2.2.4.1	Wertebereiche.java	11
2.2.5	Testdokumentation	12
2.2.6	Benutzungshinweise	12
2.2.7	Anwendungsbeispiel	12
3	Kapitel 4	12
3.1	Teilaufgabe 1	12
3.1.1	Aufgabenstellung	12
3.1.2	Anforderungsdefinition	12
3.1.3	Entwurf	13
3.1.4	Quellcode	13
3.1.4.1	Referenzen.java	13
3.1.4.2	Punkt.java	14
3.1.5	Testdokumentation	15
3.1.6	Benutzungshinweise	15
3.1.7	Anwendungsbeispiel	15
3.2	Teilaufgabe 2	15
3.2.1	Aufgabenstellung	15
3.2.2	Anforderungsdefinition	15
3.2.3	Entwurf	16
3.2.4	Quellcode	16
3.2.4.1	Matrizen.java	16
3.2.5	Testdokumentation	18
3.2.6	Benutzungshinweise	18
3.2.7	Anwendungsbeispiel	18
4	Kapitel 5	19
4.1	Teilaufgabe 1	19
4.1.1	Aufgabenstellung	19
4.1.2	Anforderungsdefinition	19
4.1.3	Entwurf	19

4.1.4	Quelltext	19
4.1.4.1	Nebeneffekte.java	19
4.1.5	Testdokumentation	20
4.1.6	Benutzungshinweise	20
4.1.7	Anwendungsbeispiel	20
4.2	Teilaufgabe 2	20
4.2.1	Aufgabenstellung	20
4.2.2	Anforderungsdefinition	20
4.2.3	Entwurf	20
4.2.4	Quelltext	20
4.2.4.1	Operatoren.java	20
4.2.5	Testdokumentation	22
4.2.6	Benutzungshinweise	22
4.2.7	Anwendungsbeispiel	23
5	Kapitel 6	24
5.1	Teilaufgabe 1	24
5.1.1	Aufgabenstellung	24
5.1.2	Anforderungsdefinition	24
5.1.3	Entwurf	24
5.1.4	Quelltext	25
5.1.4.1	Matrizen.java	25
5.1.5	Testdokumentation	28
5.1.6	Benutzungshinweise	28
5.1.7	Anwendungsbeispiel	28
5.2	Teilaufgabe 2	29
5.2.1	Aufgabenstellung	29
5.2.2	Anforderungsdefinition	29
5.2.3	Entwurf	29
5.2.4	Quelltext	29
5.2.4.1	Sprunganweisungen.java	29
5.2.5	Testdokumentation	30
5.2.6	Benutzungshinweise	30
5.2.7	Anwendungsbeispiel	31
6	Kapitel 8	31
6.1	Aufgabenstellung	31
6.2	Anforderungsdefinition	31
6.3	Entwurf	31
6.4	Quelltext	31
6.4.1	Main.java	31
6.4.2	GeradeZahl.java	32
6.4.3	OddException.java	33
6.5	Testdokumentation	33
6.6	Benutzungshinweise	33
6.7	Anwendungsbeispiel	33

1 Kapitel 1

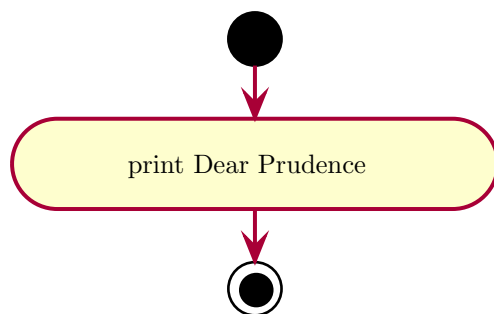
1.1 Aufgabenstellung

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

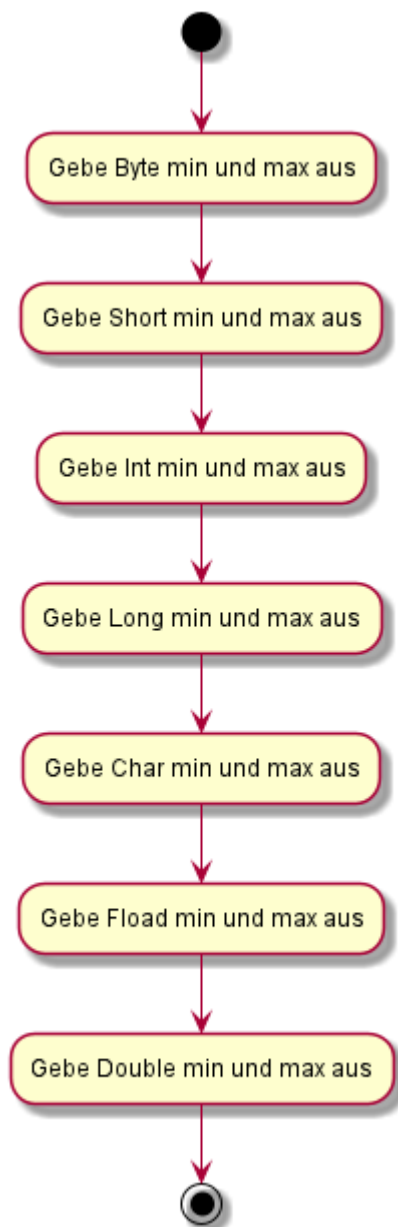
2.2.1 Aufgabenstellung

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.
        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 }

```

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

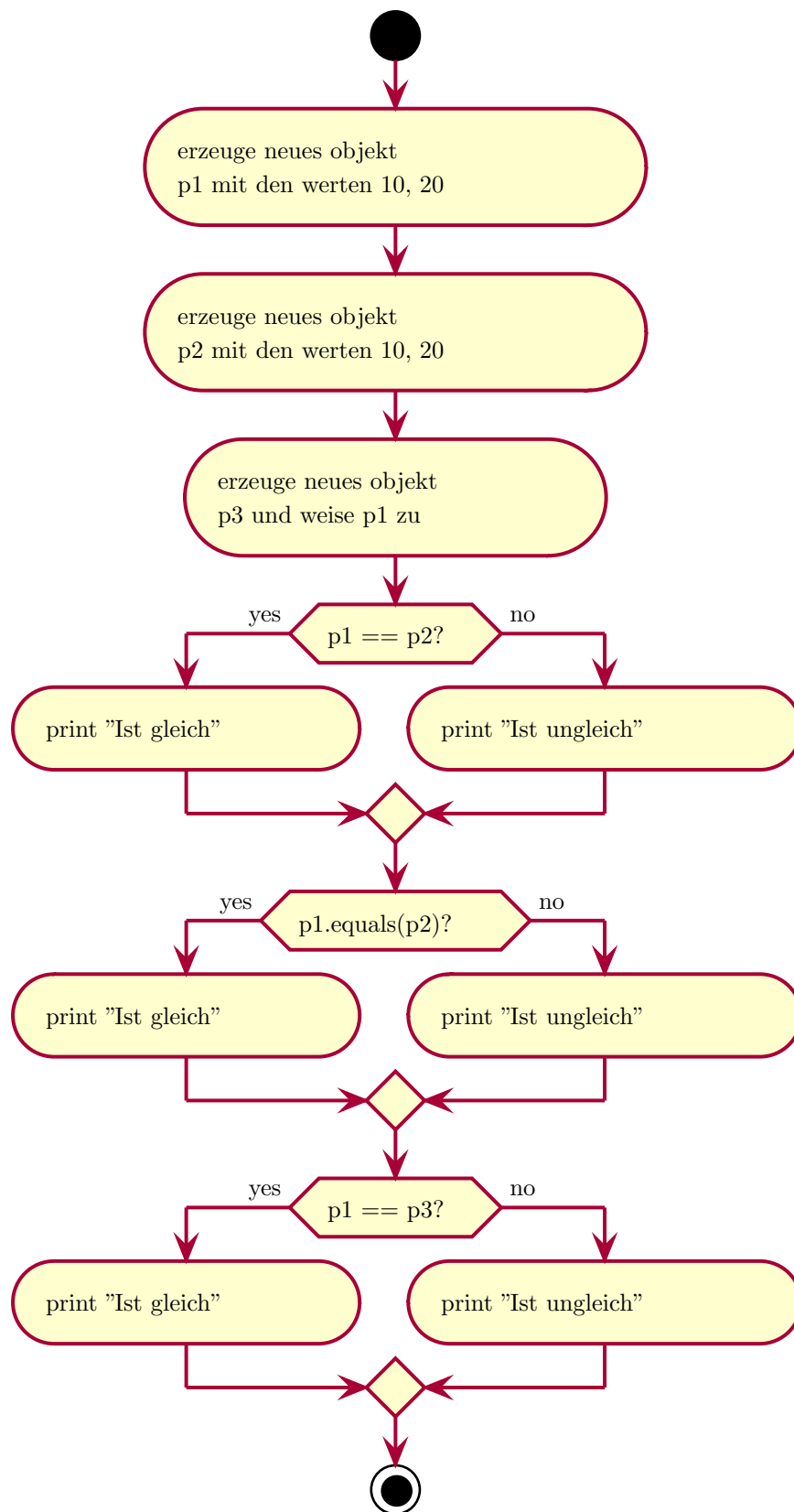
3.1.1 Aufgabenstellung

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

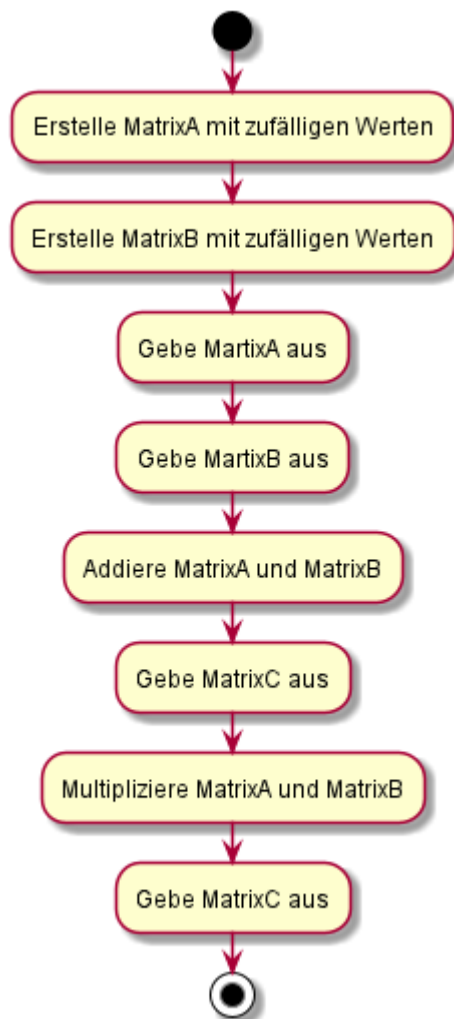
3.2.1 Aufgabenstellung

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

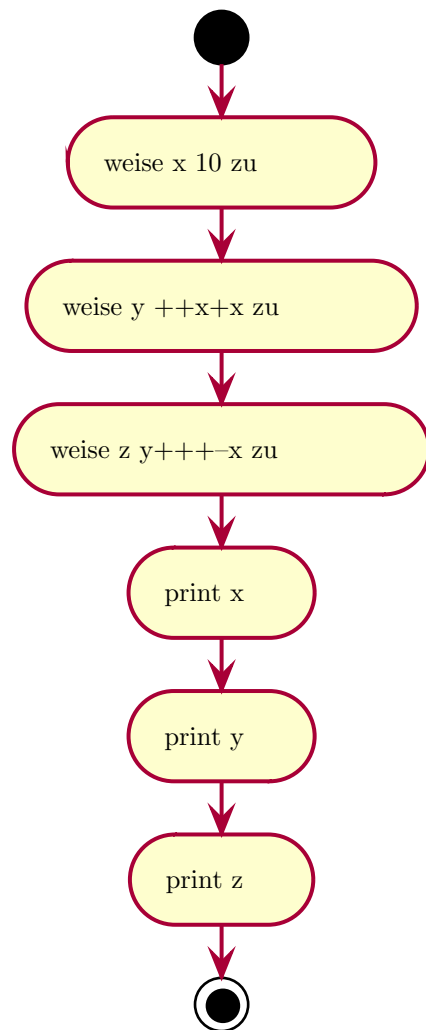
4.1.1 Aufgabenstellung

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

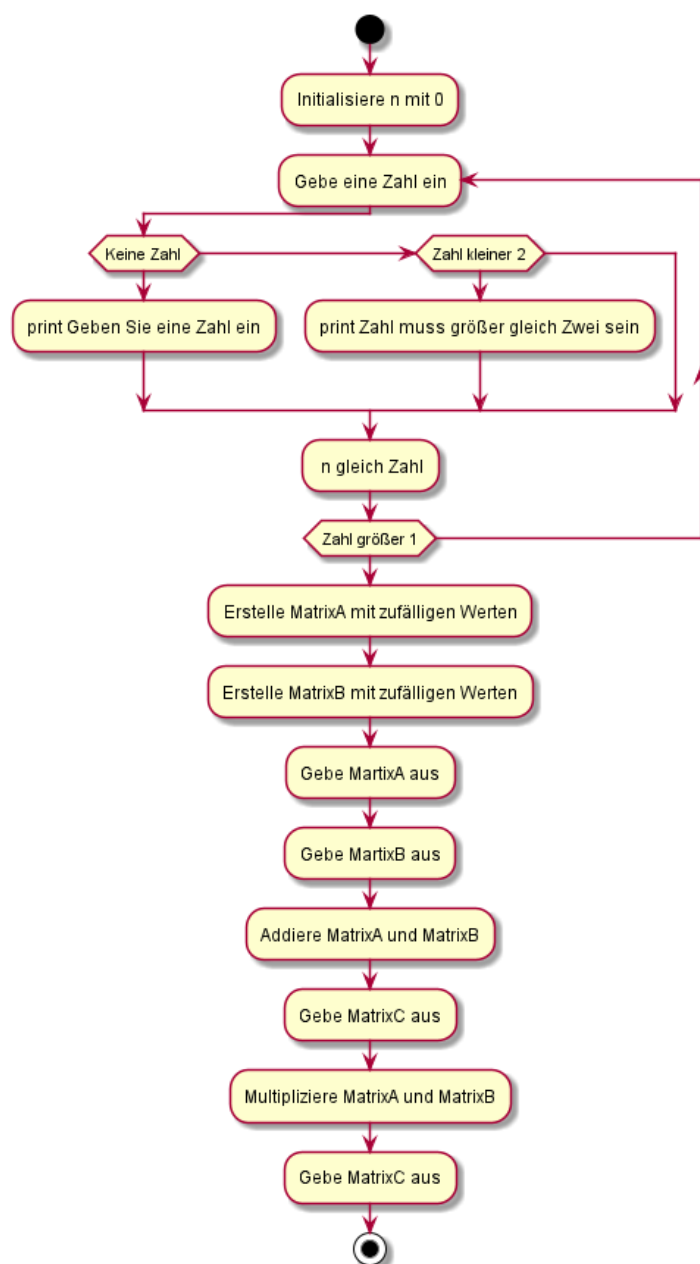
5.1.1 Aufgabenstellung

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 inkorrektener 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 grösser gleich 1 sein
9 ID      :
10 [sebastian@laptop bin]$
```

6 Kapitel 8

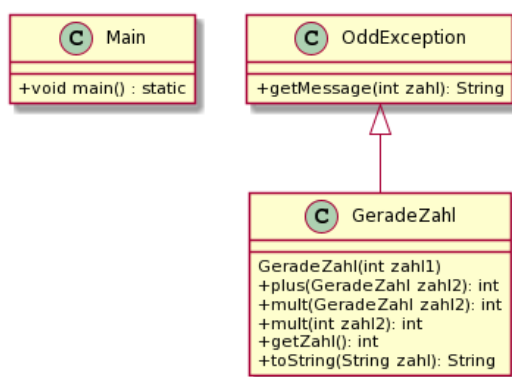
6.1 Aufgabenstellung

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.

6.2 Anforderungsdefinition

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

6.3 Entwurf



6.4 Quelltext

6.4.1 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 IllegalAccessException {
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 }

```

6.4.2 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) {
12         OddException oddexception = new OddException();
13         this.zahl1 = zahl1;
14
15         //Wenn die Zahl ungerade ist, wird eine Exception geworfen
16         try {
17             if (zahl1%2 == 1) throw oddexception;
18         } catch ( OddException a ) {
19             //Bei einer Exception wird eine ausgabe getätigt, zusätzlich wird die
20             Zahl um eins erhöht
21             System.out.println(a.getMessage(this.zahl1++));
22         }
23     }
24
25     /**
26      * Addiert Zwei GeradeZahl objekte miteinander
27      * @param zahl2
28      * @return Gibt eine int Zahl zurück
29     */
30     public int plus(GeradeZahl zahl2) {
31         return zahl1 + zahl2.getZahl();
32     }
33
34     /**
35      * Multipliziert Zwei GeradeZahl objekte miteinander
36      * @param zahl2
37      * @return Gibt eine int Zahl zurück
38     */

```



```

37     public int mult(GeradeZahl zahl2) {
38         return zahl1 * zahl2.getZahl();
39     }
40
41     /**
42      * Multipliziert ein GeradeZahl objekte mit einer int Zahl
43      * @param zahl2
44      * @return Gibt eine int Zahl zurück
45      */
46     public int mult(int zahl2) {
47         return zahl1 * zahl2;
48     }
49
50     /**
51      * @return Liefert die Zahl zurück
52      */
53     public int getZahl() {
54         return zahl1;
55     }
56
57     /**
58      * Bei einem Print wird diese Methode ausgeführt
59      * @return Liefert einen String mit der Zahl zurück
60      */
61     @Override
62     public String toString() {
63         return "" + getZahl();
64     }
65 }

```

6.4.3 OddException.java

```

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

```

6.5 Testdokumentation

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

6.6 Benutzungshinweise

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

6.7 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]$
```