

Hochschule -

Fakultät IV – Technische Informatik

Modul: Programmieren 1

Professor: -

Entwicklungsarbeit

von

Sebastian Schramm Matrikel-Nr. -

7. Dezember 2020

Inhaltsverzeichnis

1	Kapitel 1	3
1.1	Aufgabenstellung	3
1.2	Anforderungsdefinition	3
1.3	Entwurf	3
1.4	Quellcode	3
1.4.1	Main.java	3
1.5	Testdokumentation	3
1.6	Benutzungshinweise	3
1.7	Anwendungsbeispiel	4
2	Kapitel 3	4
2.1	Teilaufgabe 1	4
2.1.1	Aufgabenstellung	4
2.1.2	Anforderungsdefinition	4
2.1.3	Entwurf	4
2.1.4	Quelltext	4
2.1.4.1	Typkonvertierungen.java	4
2.1.5	Testdokumentation	7
2.1.6	Benutzungshinweise	7
2.1.7	Anwendungsbeispiel	7
2.2	Teilaufgabe 2	9
2.2.1	Aufgabenstellung	9
2.2.2	Anforderungsdefinition	9
2.2.3	Entwurf	9
2.2.4	Quelltext	9
2.2.4.1	Wertebereiche.java	9
2.2.5	Testdokumentation	10
2.2.6	Benutzungshinweise	10
2.2.7	Anwendungsbeispiel	10
3	Kapitel 4	10
3.1	Teilaufgabe 1	10
3.1.1	Aufgabenstellung	10
3.1.2	Anforderungsdefinition	10
3.1.3	Entwurf	10
3.1.4	Quellcode	10
3.1.4.1	Referenzen.java	10
3.1.4.2	Punkt.java	11
3.1.5	Testdokumentation	11
3.1.6	Benutzungshinweise	11
3.1.7	Anwendungsbeispiel	11
3.2	Teilaufgabe 2	12
3.2.1	Aufgabenstellung	12
3.2.2	Anforderungsdefinition	12
3.2.3	Entwurf	12
3.2.4	Quellcode	12
3.2.4.1	Matrizen.java	12
3.2.5	Testdokumentation	14
3.2.6	Benutzungshinweise	14
3.2.7	Anwendungsbeispiel	14
4	Kapitel 5	14
4.1	Teilaufgabe 1	14
4.1.1	Aufgabenstellung	14
4.1.2	Anforderungsdefinition	14
4.1.3	Entwurf	14

4.1.4	Quelltext	14
4.1.4.1	Nebeneffekte.java	14
4.1.5	Testdokumentation	15
4.1.6	Benutzungshinweise	15
4.1.7	Anwendungsbeispiel	15
4.2	Teilaufgabe 2	15
4.2.1	Aufgabenstellung	15
4.2.2	Anforderungsdefinition	15
4.2.3	Entwurf	15
4.2.4	Quelltext	15
4.2.4.1	Operatoren.java	15
4.2.5	Testdokumentation	17
4.2.6	Benutzungshinweise	17
4.2.7	Anwendungsbeispiel	17
5	Kapitel 6	18
5.1	Teilaufgabe 1	18
5.1.1	Aufgabenstellung	18
5.1.2	Anforderungsdefinition	19
5.1.3	Entwurf	19
5.1.4	Quelltext	19
5.1.4.1	Matrizen.java	19
5.1.5	Testdokumentation	22
5.1.6	Benutzungshinweise	22
5.1.7	Anwendungsbeispiel	22
6	Kapitel 6	23
6.1	Teilaufgabe 2	23
6.1.1	Aufgabenstellung	23
6.1.2	Anforderungsdefinition	23
6.1.3	Entwurf	23
6.1.4	Quelltext	23
6.1.4.1	Sprunganweisungen.java	23
6.1.5	Testdokumentation	24
6.1.6	Benutzungshinweise	24
6.1.7	Anwendungsbeispiel	24

1 Kapitel 1

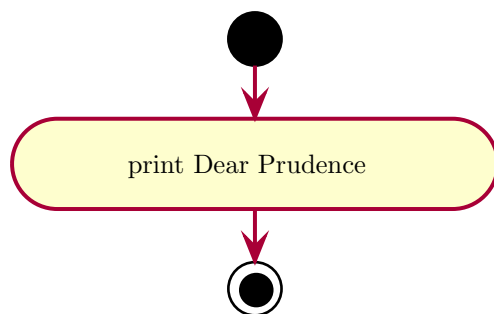
1.1 Aufgabenstellung

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

?

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. Welches alle primitive Datentypen erweiternd und einschränkend Konvertiert.

2.1.2 Anforderungsdefinition

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

2.1.3 Entwurf

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("Bouble  " + newDouble);
44     System.out.println("\nChar   " + (char) newInt); //Char wird hier separat
        ausgegeben
45     System.out.println("-----");
46 }
47
48 /**
49  * Eine einschraenkende 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 einschraenkend");
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("Bouble  " + newDouble);
70     System.out.println("\nChar    " + (char) newInt); //Char wird hier separat
        ausgegeben
71     System.out.println("-----");
72 }
73
74 /**
75  * Eine einschraenkende 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 einschraenkend");
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("Bouble  " + newDouble);
97     System.out.println("\nChar    " + (char) _int); //Char wird hier separat

```

```

    ausgegeben
98     System.out.println("-----");
99 }
100
101 /**
102  * Eine einschaenkende 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 einschaenkend");
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("Bouble   " + newDouble);
124     System.out.println("\nChar    " + (char) newInt); //Char wird hier separat
    ausgegeben
125     System.out.println("-----");
126 }
127
128 /**
129  * Eine einschaenkende 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 einschaenkend");
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("Bouble   " + newDouble);
154     System.out.println("-----");
155 }
156
157 /**
158  * Eine einschaenkende 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 einschraenkend");
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("Bouble " + newDouble);
180      System.out.println("\nChar " + (char) newInt); //Char wird hier separat
181      ausgegeben
182      System.out.println("-----");
183  }
184  /**
185   * Eine einschraenkende 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 einschraenkend");
196       System.out.println("Bouble " + _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
203       ausgegeben
204       System.out.println("-----");
205   }
206 }

```

2.1.5 Testdokumentation

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, 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  Bouble -128.0
10
11 Char
12 -----
13 Short einschraenkend
14 Short   34
15 Byte    34
16 Short erweiternd
17 Short   34
18 Int     34
19 Long    34
20 FLoat   34.0
21 Bouble  34.0
22
23 Char    "
24 -----
25 Int einschraenkend
26 Int     98987
27 Short  -32085
28 Byte   -85
29 Int erweiternd
30 Int     98987
31 Long    98987
32 FLoat   98987.0
33 Bouble  98987.0
34
35 Char
36 -----
37 Long einschraenkend
38 Long    987987987
39 Int     987987987
40 Short  -32749
41 Byte    19
42 Long erweiternd
43 Long    987987987
44 FLoat   9.8798797E8
45 Bouble  9.87987987E8
46
47 Char
48 -----
49 Char einschraenkend
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 Bouble  97.0
60 -----
61 Float einschraenkend
62 FLoat   15.0
63 Long    15
64 Int     15

```

```

65 Short 15
66 Byte 15
67 Float erweiternd
68 FLoat 15.0
69 Bouble 15.0
70
71 Char
72 -----
73 Double einschraenkend
74 Bouble 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 welle die Wertebereiche der primitieven Datentypen ausgibt.

2.2.2 Anforderungsdefinition

1. Zu jedem primitieven 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 primitieven 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
15         System.out.println("Short min " + Short.MIN_VALUE + " | Short max " + Short.
16             MAX_VALUE);
17         //Min und Max Value von Integer
18         System.out.println("Integer min " + Integer.MIN_VALUE + " | Integer max " +
19             Integer.MAX_VALUE);
20         //Min und Max Value von Long
21         System.out.println("Long min " + Long.MIN_VALUE + " | Byte Long " + Long.
22             MAX_VALUE);
23
24         //Min und Max Value von Char
25         System.out.println("Char min " + Character.MIN_VALUE + " | Char max " +
26             Character.MAX_VALUE);
27     }
28 }

```

```

22 //System.out.println("\u0000 | \uffff");
23
24 //Min und Max Value von Float
25 System.out.println("Float min " + Float.MIN_VALUE + " | Float max " + Float.
    MAX_VALUE);
26 //Min und Max Value von Double
27 System.out.println("Double min " + Double.MIN_VALUE + " | Double max " + Double
    .MAX_VALUE);
28 }
29
30 }

```

2.2.5 Testdokumentation

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:

```

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

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  * Classe 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          * Inizialisierungsmethode 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         /*
32          * Anschliessend werden die Matrizen hier Addiert
33          */
34         System.out.println("Addition von A und B:");
35         printMatrix(addition(matrixA, matrixB));
36
37         /*
38          * Und hier Multipliziert
39          */
40         System.out.println("Multiplikation von A und B:");
```

```

39     printMatrix(multiplikation(matrixA, matrixB));
40 }
41
42 /**
43  * Initialisierung des Arrays
44  * @param n
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 gesamte Array 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
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
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[
                    WmatrixA][WmatrixB];
91
92     return matrixMult;
93 }
94
95 /**
96  * Hier wird die Matrix ausgegeben
97  * @param matrix
98  */
99 private static void printMatrix(int matrix[][]) {

```

```

100     for (int y[]: matrix) {
101         for (int x: y)
102             System.out.print(x + "\t");
103         System.out.println();
104     }
105     System.out.println();
106 }
107
108 }

```

3.2.5 Testdokumentation

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 For Schleife
16 6150 6900
17 1852 2916
18 [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

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

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      boolische();
17      bitshifting();
18      zuweisung();
19  }
20
21  private static void arithmetisch() {
22      System.out.println("Arithmeschie 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);      // Teilerrest, 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  @SuppressWarnings("unused")
59  private static void boolische() {
60      System.out.println("\nBoolische Operatoren:");
61      System.out.println("!true = " + !true);      // Negation
62      System.out.println("true && true = " + (true && true)); // Und, true 2, genau
// dann wenn alle Argumente true sind
63      System.out.println("true || false = " + (true || false)); // Oder, true, wenn
// mindestens ein Operand true ist
64      System.out.println("true ^ true = " + (true ^ true));    // Xor, true, wenn
// genau ein Operand true ist

```

```

65     }
66
67     private static void bitshifting() {
68         int bit = 0b01000100;
69         System.out.println("\nBitweise Operatoren:");
70         System.out.println("0b10111011 = ~0b01000100"); //Invertiert die Bits
71         System.out.println("0b10101010 & 0b11111111 = " + Integer.toString(0b10101010 &
72             0b11111111, 2)); // Verundet die Bits
73         System.out.println("0b10101010 | 0b01101001 = " + Integer.toString(0b10101010 |
74             0b00101001, 2)); // Verodert die Bits
75         System.out.println("0b10101010 ^ 0b11111111 = " + Integer.toString(0b10101010 ^
76             0b11111111, 2)); // Exklusives oder
77         System.out.println("0b10101010 >> 2 = " + Integer.toString(0b10101010 >> 2, 2))
78             ; // Rechtssshift
79         System.out.println("0b10101010 >>> 1 = " + Integer.toString(0b10101010 >>> 1,
80             2)); // Rechtssshift mit Nullen auffüllen
81         System.out.println("0b10101010 << 1 = " + Integer.toString(0b10101010 << 1, 2))
82             ; // Linksverschiebung
83     }
84
85     private static void zuweisung() {
86         int a = 20;
87         System.out.println("\nZuweisungs Operatoren:");
88         System.out.println("int a = 20"); // Einfache zuweisung
89         System.out.println("a += 10 => " + (a += 10)); // Addiert ein wert zu der
90             Variable
91         System.out.println("a -= 20 => " + (a -= 20)); // Subtrahiert ein wert zu
92             der Variable
93         System.out.println("a *= 7 => " + (a *= 7)); // Dividiert die Variable durch
94             den angegebenen Wert und weist ihn zu
95         System.out.println("a /= 5 => " + (a /= 5)); // Multipliziert die Variable
96             durch den angegebenen Wert und weist ihn zu
97         System.out.println("a %= 5 => " + (a %= 5)); // Ermittelt den Rest und weist
98             ihn zu
99         System.out.println("a &= 12 => " + (a &= 12)); // Eine bitweise verundung
100         System.out.println("a |= 10 => " + (a |= 10)); // Bitweise veroderung
101         System.out.println("a ^= 30 => " + (a ^= 30)); // Exklusives oder auf
102             Bitebene
103         System.out.println("a <= 3 => " + (a <= 3)); // Linksverschiebung
104         System.out.println("a >= 1 => " + (a >= 1)); // Rechtsverschiebung
105         System.out.println("a >>= 2 => " + (a >>= 2)); // Rechtsverschiebung und
106             Auffüllen mit Nullen
107     }
108 }

```

4.2.5 Testdokumentation

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 nxn 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 nxn 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  * Classe 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         int n = sc.nextInt();
22
23         /*
24          * Inizialisierungsmethode wird mit dem Wert n aufgerufen.
25          * Anschliessend wird diese Matrix erzeugt und mit
26          * zufällig generierten Zahlen befüllt.
27          */
28         matrixA = initialize(n);
29         matrixB = initialize(n);
30
31         /*
32          * Zuerst werden die Beiden Matrizen A und B jeweils ausgegeben
33          */
34         System.out.println("Matrix A:");
35         printMatrix(matrixA);
36         System.out.println("Matrix B:");
37         printMatrix(matrixB);
38
39         /*
40          * Anschliessend werden die Matrizen hier Addiert
41          */
42         System.out.println("Addition von A und B:");
43         printMatrix(addition(matrixA, matrixB));
44
45         /*
46          * Und hier Multipliziert
47          */
48         System.out.println("Multiplikation von A und B:");
49         System.out.println("For Schleife");
50         printMatrix(multiplikationFor(matrixA, matrixB));
51         System.out.println("While Schleife");
52         printMatrix(multiplikationWhile(matrixA, matrixB));
53         System.out.println("Do-While Schleife");
54         printMatrix(multiplikationDoWhile(matrixA, matrixB));
55     }
56 }
```

```

53     sc.close();
54 }
55
56 /**
57  * Initialisierung des Arrays
58  * @param n
59  * @return matrix
60  */
61 private static int[][] initialize(int n) {
62     int matrix[][] = new int[n][n];
63     /*
64      * Bei der Initialisierung wird einmal durch das gesamt Array durch itteriert.
65      * Dabei werden dann mit Math.random() zufällige Zahlen rein geschrieben.
66      */
67     for (int i = 0; i < matrix.length; ++i)
68         for (int l = 0; l < matrix[i].length; ++l)
69             matrix[i][l] = (int) (Math.random() * 100);
70
71     return matrix;
72 }
73
74 /**
75  * Addition der beiden Matrizen A und B
76  * @param matrixA
77  * @param matrixB
78  * @return
79  */
80 private static int[][] addition(int matrixA[][], int matrixB[][]) {
81     int matrixAd[][] = new int[matrixA.length][matrixA[0].length]; //Es wird ein
82     //neues Temporäres Array angelegt
83
84     for (int i = 0; i < matrixA.length; ++i) {
85         for (int n = 0; n < matrixA[i].length; ++n) {
86             matrixAd[i][n] = matrixA[i][n] + matrixB[i][n];
87         }
88     }
89
90     return matrixAd;
91 }
92
93 /**
94  * Multiplikation der beiden Matrizen A und B
95  * @param matrixA
96  * @param matrixB
97  * @return
98  */
99 private static int[][] multiplikationFor(int matrixA[][], int matrixB[][]) {
100     int matrixMult[][] = new int[matrixB.length][matrixB[0].length];
101
102     //Hier die Variante mit For Schleifen
103     for (int HmatrixB = 0; HmatrixB < matrixB.length; ++HmatrixB)
104         for (int WmatrixB = 0; WmatrixB < matrixB[HmatrixB].length; ++WmatrixB)
105             for (int WmatrixA = 0; WmatrixA < matrixB.length; ++WmatrixA)
106                 matrixMult[HmatrixB][WmatrixB] += matrixA[HmatrixB][WmatrixA] * matrixB[
107                     WmatrixA][WmatrixB];
108
109     return matrixMult;
110 }
111
112 /**
113  * Multiplikation der beiden Matrizen A und B
114  * @param matrixA
115  * @param matrixB

```

```

114     * @return
115     */
116 private static int[][] multiplikationWhile(int matrixA[], int matrixB[]) {
117     int matrixMult[][] = new int[matrixB.length][matrixB[0].length];
118
119     int HmatrixB = 0;
120     int WmatrixB = 0;
121     int WmatrixA = 0;
122
123     //Hier die Variante mit While Schleifen
124     while (HmatrixB < matrixB.length) {
125         WmatrixB = 0;
126         while (WmatrixB < matrixB[HmatrixB].length) {
127             WmatrixA = 0;
128             while (WmatrixA < matrixB.length) {
129                 matrixMult[HmatrixB][WmatrixB] += matrixA[HmatrixB][WmatrixA] * matrixB[
130                     WmatrixA][WmatrixB];
131                 ++WmatrixA;
132             }
133             ++WmatrixB;
134         }
135         ++HmatrixB;
136     }
137     return matrixMult;
138 }
139 /**
140  * Multiplikation der beiden Matrizen A und B
141  * @param matrixA
142  * @param matrixB
143  * @return
144  */
145 private static int[][] multiplikationDoWhile(int matrixA[], int matrixB[]) {
146     int matrixMult[][] = new int[matrixB.length][matrixB[0].length];
147
148     int HmatrixB = 0;
149     int WmatrixB = 0;
150     int WmatrixA = 0;
151
152     //Hier die Variante mit Do-While Schleifen
153     do {
154         WmatrixB = 0;
155         do {
156             WmatrixA = 0;
157             do {
158                 matrixMult[HmatrixB][WmatrixB] += matrixA[HmatrixB][WmatrixA] * matrixB[
159                     WmatrixA][WmatrixB];
160                 ++WmatrixA;
161             } while (WmatrixA < matrixB.length);
162             ++WmatrixB;
163         } while (WmatrixB < matrixB[HmatrixB].length);
164         ++HmatrixB;
165     } while (HmatrixB < matrixB.length);
166
167     return matrixMult;
168 }
169 /**
170  * Hier wird die Matrix ausgegeben
171  * @param matrix
172  */
173 private static void printMatrix(int matrix[][]) {
174     for (int y[]: matrix) {

```

```

175         for (int x: y)
176             System.out.print(x + "\t");
177         System.out.println();
178     }
179     System.out.println();
180 }
181
182 }

```

5.1.5 Testdokumentation

5.1.6 Benutzungshinweise

Nach dem aufrufen des Programmes, 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: 5
4  Matrix A:
5  56  64  80  51  83
6  28  21  53  57  4
7  31  65  76  25  17
8  23  14  36  38  1
9  12  59  78  21  54
10
11 Matrix B:
12 22  39  22  22  95
13 47  51  16  19  73
14 91  43  20  42  15
15 44  25  72  19  6
16 46  7  58  14  18
17
18 Addition von A und B:
19 78  103 102  73  178
20 75  72  69  76  77
21 122 108 96  67  32
22 67  39  108 57  7
23 58  66  136 35  72
24
25 Multiplikation von A und B:
26 For Schleife
27 17582 10744 12342 7939 12992
28 9118  5895  6348  4380  5402
29 12535 8536  6028  5822  9286
30 6158  4116  4244  3020  3993
31 13543 7734  7412  5816  7715
32
33 While Schleife
34 17582 10744 12342 7939 12992
35 9118  5895  6348  4380  5402
36 12535 8536  6028  5822  9286
37 6158  4116  4244  3020  3993
38 13543 7734  7412  5816  7715
39
40 Do-While Schleife
41 17582 10744 12342 7939 12992
42 9118  5895  6348  4380  5402
43 12535 8536  6028  5822  9286

```

```
44 6158 4116 4244 3020 3993
45 13543 7734 7412 5816 7715
46 [sebastian@laptop bin]$
```

6 Kapitel 6

6.1 Teilaufgabe 2

6.1.1 Aufgabenstellung

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

6.1.2 Anforderungsdefinition

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

6.1.3 Entwurf

6.1.4 Quelltext

6.1.4.1 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     private static int id;
12     private static String pw;
13
14     public static void main(String[] args) {
15         login();
16     }
17
18     /**
19      * Kleine einfache Implementierung von Nutzern, mithilfe
20      * einer switch-Anweisung
21      * @param userID
22      * @param userPw
23      * @return
24      */
25     private static boolean userData(int userID, String userPw) {
26         switch (userID) {
27             case 1:
28                 return userPw.equals("hallo")? true:false;
29             case 112:
30                 return userPw.equals("das")? true:false;
31             case 124:
32                 return userPw.equals("ist")? true:false;
33             case 345:
34                 return userPw.equals("nicht")? true:false;
35             case 653:
36                 return userPw.equals("geheim")? true:false;
37         }
38         return false;
39     }
}
```



```

40
41 /**
42  * Hier befindet sich das Login feld
43  */
44 private static void login() {
45     Scanner sc = new Scanner(System.in);
46     do {
47         System.out.println("Willkommen...!");
48         System.out.print("ID      : ");
49
50         /*
51          * Eine kleine Abfrage die Prüft, ob die eingegebene
52          * ID nur aus Zahlen besteht
53          */
54         while (!sc.hasNextInt()) {
55             System.out.println("Error, es dürfen nur Zahlen enthalten sein.");
56             sc.nextLine();
57         }
58         id = sc.nextInt();
59
60         System.out.print("Passwort: ");
61         /*
62          * Wenn ein Nutzer mit dem angegebenen Passwort
63          * nicht existiert, wird die ID zurückgesetzt
64          * und eine Fehlermeldung wird ausgegeben
65          */
66         if(!userData(id, sc.next())) {
67             id = 0;
68             System.out.println("Ihre Angaben sind leider falsch, versuchen Sie es
69                             erneut.");
70         }
71         /*
72          * Die Schleife wird solange durchlaufen, bis sich ein nutzer
73          * erfolgreich angemeldet hat.
74          */
75     } while (id == 0);
76
77     System.out.println("Juhu, Sie haben sich eingeloggt");
78     //    sc.close();
79 }
80
81
82
83 }

```

6.1.5 Testdokumentation

6.1.6 Benutzungshinweise

Nach dem aufrufen des Programmes, wird der nutzer aufgefordert seine NutzerID anzugebe,j sowie anschließend sein Passwort. Bei inkorrekt eingaben, wird man erneut aufgeforder die Daten einzutippen.

6.1.7 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 Willkommen...!
7 ID      :
8 [sebastian@laptop bin]$
```