Hochschule -

Fakultät IV – Technische Informatik
Modul: Programmieren 1
Professor: -

Entwicklungsarbeit

von

Sebastian Schramm Matrikel-Nr. -

7. Dezember 2020

Inhaltsverzeichnis

1 Kapitel 1

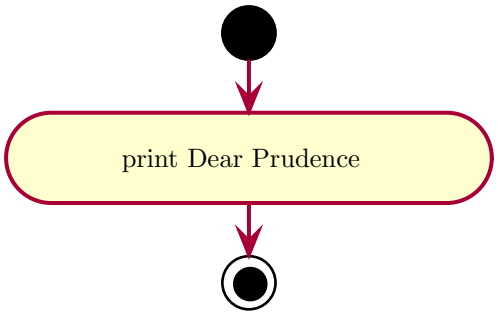
1.1 Aufgabenstellung

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” angeziegt werden.

1.7 Anwendungsbeispiel

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

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

```java
package chapter_03;

/**
 * Klasse mit der Main-Methode
 * und der einzelnen Typkonvertierungen
 * @author Sebastian
 */
public class Typkonvertierungen {

  public static void main(String[] args) {
    /*
     * Rund die einzelnen Methoden auf, mit entsprechenden Werten
     */
    convertByte((byte) -128);
    convertShort((short) 34);
    convertInt(98987);
    convertLong(987987987);

    convertChar('a');

    convertFloat(15.0f);
    convertDouble(1.7976931348623157E308);
  }

  /**
   * Eine erweiternde Konvertierung von Byte zu Double
   * @param _byte
   */
  private static void convertByte(byte _byte) {
    short newShort = _byte;
    int newInt = _byte;
    long newLong = _byte;
    float newFloat = _byte;
    double newDouble = _byte;

    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 einschraenkende 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 einschraenkend");
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 einschraenkende 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 einschraenkend");
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 einschraenkende 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
             ausgegeben
181      System.out.println("---------------------");
182    }
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
             ausgegeben
203      System.out.println("---------------------");
204    }
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
```

```
---------------------
Byte erweiternd
Byte   -128
Short  -128
Int    -128
Long   -128
FLoat  -128.0
Bouble -128.0

Char
---------------------
Short einschraenkend
Short  34
Byte   34
Short erweiternd
Short  34
Int    34
Long   34
FLoat  34.0
Bouble 34.0

Char   "
---------------------
Int einschraenkend
Int    98987
Short  -32085
Byte   -85
Int erweiternd
Int    98987
Long   98987
FLoat  98987.0
Bouble 98987.0

Char
---------------------
Long einschraenkend
Long   987987987
Int    987987987
Short  -32749
Byte   19
Long erweiternd
Long   987987987
FLoat  9.8798797E8
Bouble 9.87987987E8

Char
---------------------
Char einschraenkend
Char   a
Long   97
Int    97
Short  97
Byte   97
Char erweiternd
Char   a
Long   97
FLoat  97.0
Bouble 97.0
---------------------
Float einschraenkend
FLoat  15.0
Long   15
Int    15
```

```
Short  15
Byte   15
Float erweiternd
FLoat  15.0
Bouble 15.0

Char
----------------------
Double einschraenkend
Bouble 1.7976931348623157E308
FLoat  Infinity
Long   9223372036854775807
Int    2147483647
Short  -1
Byte   -1

Char
----------------------
[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

```java
package chapter_03;

/**
 * Klasse mit der Main-Methode
 * und gibt die Wertebereiche der primitieven Datentypen aus
 * @author Sebastian
 */
public class Wertebereiche {

  public static void main(String[] args) {
    //Min und Max Value von Byte
    System.out.println("Byte min " + Byte.MIN_VALUE + " | Byte max " + Byte.
        MAX_VALUE);
    //Min und Max Value von Short
    System.out.println("Short min " + Short.MIN_VALUE + " | Short max " + Short.
        MAX_VALUE);
    //Min und Max Value von Integer
    System.out.println("Integer min " + Integer.MIN_VALUE + " | Integer max " +
        Integer.MAX_VALUE);
    //Min und Max Value von Long
    System.out.println("Long min " + Long.MIN_VALUE + " | Byte Long " + Long.
        MAX_VALUE);

    //Min und Max Value von Char
    System.out.println("Char min " + Character.MIN_VALUE + " | Char max " +
        Character.MAX_VALUE);
```

```
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 Addresse 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 gelich 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:

```
[sebastian@laptop bin]$ java Referenzen
Ist ungleich
Ist ungleich
Ist gleich
[sebastian@laptop bin]$
```

3.2 Teilaufgabe 2

3.2.1 Aufgabenstellung

Wir sollen ein Programm schreiben welches 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

```java
package chapter_04;

/**
 * Classe mit der Main-Methode
 * Addiert und Multipliziert Matrizen
 * @author Sebastian
 *
 */
public class Matrizen {

  public static void main(String[] args) {
    int matrixA[][];
    int matrixB[][];

    /*
     * Inizialisierungsmethode wird mit dem Wert n aufgerufen.
     * AnschlieSSend wird diese Matrix erzeugt und mit
     * zufällig generierten Zahlen befüllt.
     */
    matrixA = initialize(2);
    matrixB = initialize(2);

    /*
     * Zuerst werden die Beiden Matrizen A und B jeweils ausgegeben
     */
    System.out.println("Matrix A:");
    printMatrix(matrixA);
    System.out.println("Matrix B:");
    printMatrix(matrixB);
    /*
     * AnschlieSSend werden die Matrizen hier Addiert
     */
    System.out.println("Addition von A und B:");
    printMatrix(addition(matrixA, matrixB));
    /*
     * Und hier Multipliziert
     */
    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 gesammt Array duch itteriert.
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  Dieses Programm berechnet eine zufällig erstellte nxn Matrix
3  Geben sie n an: 5
4  Matrix A:
5  22  54  23  36  35
6  49  98  67  82  23
7  78  62  36  60  2
8  54  2 74  48  95
9  58  2 86  3 75
10
11 Matrix B:
12 84  7 63  44  84
13 69  55  76  83  97
14 60  98  66  41  4
15 27  6 71  96  24
16 40  8 21  66  38
17
18 Addition von A und B:
19 106 61  86  80  119
20 118 153 143 165 120
21 138 160 102 101 6
22 81  8 145 144 119
23 98  10  107 69  113
24
25 Multiplikation von A und B:
26 9326  5874  10299 12159 9372
27 18032 12975 21262 22427 16732
28 14690 7860  16304 15946 14226
29 14210 8788  13841 16454 9788
30 13251 9562  11270 11482 8332
31 [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 Opteratorn, 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

```java
package chapter_05;

@SuppressWarnings("unused")
public class Operatoren {
  //Schreiben Sie ein Programm, welches alle Operatoren in Java verwendet.
  /**
   * Klasse mit der Main-Methode
   * Dieses Programm solle alle Operatoren,
   * die in Java existieren verdeutlichen
   * @param args
   */
  public static void main(String[] args) {
    arithmetisch();
    inkrement();
    vergleiche();
    boolische();
    bitshifting();
    zuweisung();
  }

  private static void arithmetisch() {
    System.out.println("Arithmeschie Operatoren:");
    System.out.println("23 + 34 = " + (23 + 34)); // Addition
    System.out.println("54 - 32 = " + (54 - 32)); // Subtraktion
    System.out.println("12 * 30 = " + 12 * 30);   // Multiplikation
    System.out.println("56 / 12 = " + 56 / 12);   // Division
    System.out.println("74 % 2  = " + 74 % 2);    // Teilerrest, Modulo-Operation,
        errechnet den Rest einer Division
    int i;
    System.out.println("int i = +3 = " + (i = +3)); // positives Vorzeichen
    int n;
    System.out.println("int n = -i = " + (n = -i)); //negatives Vorzeichen
  }

  private static void inkrement() {
    int x = 10;
    System.out.println("\nInkrement Operatoren:");
    System.out.println("x   = " + x);
    System.out.println("x++ = " + x++); //Postinkrement: Weist zuerst zu, dann
        hochzählen
    System.out.println("x   = " + x);
    System.out.println("++x = " + ++x); //Preinkrement: Zählt erst hoch, dann
        zuweisen
    System.out.println("x   = " + x);
    System.out.println("x-- = " + x--); //Postinkrement: Weist zuerst zu, dann
        hochzählen
    System.out.println("x   = " + x);
    System.out.println("--x = " + --x); //Preinkrement: Zählt erst hoch, dann
        zuweisen
    System.out.println("x   = " + x);
  }

  private static void vergleiche() {
    System.out.println("\nVergleichs Operatoren:");
    System.out.println("37 == 2 = " + (37 == 2)); // gleich
    System.out.println("1 != 2 = " + (1 != 2));   // ungleich
    System.out.println("13 > 3 = " + (13 > 3));   // grösSer
    System.out.println("23 < 2 = " + (23 < 2));   // kleiner
    System.out.println("23 >= 23 = " + (23 >= 23)); // grösSer oder gleich
    System.out.println("45 <= 44 = " + (45 <= 44)); // kleiner oder gleich
  }

  @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, ture 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 &
            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))
            ;         // Rechtsshift
75      System.out.println("0b10101010 >>> 1 = " + Integer.toString(0b10101010 >>> 1,
            2));         // Rechtsshift 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("\nZuweisungs 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
            Bitebene
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

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:

```
[sebastian@laptop bin]$ java Operatoren
Arithmeschie Operatoren:
23 + 34 = 57
54 - 32 = 22
12 * 30 = 360
56 / 12 = 4
74 % 2  = 0
int i = +3 = 3
int n = -i = -3

Inkrement Operatoren:
x   = 10
x++ = 10
x   = 11
++x = 12
x   = 12
x-- = 12
x   = 11
--x = 10
x   = 10

Vergleichs Operatoren:
37 == 2 = false
1 != 2 = true
13 > 3 = true
23 < 2 = false
23 >= 23 = true
45 <= 44 = false

Boolische Operatoren:
!true = false
true && true = true
true || false = true
true ^ true = false

Bitweise Operatoren:
0b10111011 = ~0b01000100
0b10101010 & 0b11111111 = 10101010
0b10101010 | 0b01101001 = 10101011
0b10101010 ^ 0b11111111 = 1010101
0b10101010 >> 2 = 101010
0b10101010 >>> 1 = 1010101
0b10101010 << 1 = 101010100

Zuweisungs Operatoren:
int a = 20
a += 10 => 30
a -= 20 => 10
a *= 7 => 70
a /= 5 => 14
a %= 5 => 4
a &= 12 => 4
a |= 10 => 14
a ^= 30 => 16
a <<= 3 => 128
a >>= 1 => 64
a >>>= 2 => 16
[sebastian@laptop bin]$
```