

Session 3: Commenting, data types, casting

Softwaretechnologie: Java I

Nils Reiter

`nils.reiter@uni-koeln.de`

October 26, 2022

Section 1

Exercise 2



▼ Hausaufgabe 02 (Verpflichtend)

Beendet am: Gestern, 23:55

Arbeitsanweisung

Siehe beiliegende Datei README.md.

Dateien

exercise-02.zip Download

Terminplan

Startzeit 19. Okt 2022, 13:00

Beendet am Gestern, 23:55

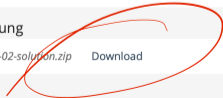
Verbleibende Bearbeitungsdauer **Die Zeit ist abgelaufen.**

Ihre Einreichung

Abgegebene Dateien Sie haben noch keine Datei abgegeben.

Musterlösung

exercise-02-solution.zip Download



Exercise 01: Operators

```
1 public class Operators {
2     public static void main(String[] args) {
3         int i;
4         int j;
5         boolean b;
6         i = 5;
7
8         j = 5 + 5; // 10
9         System.out.println(j);
10
11        j = 5 - 5 + 5 * 2; // 10
12        System.out.println(j);
13
14        j = i - (10 * i) + 10; // -35
15        System.out.println(j);
16
17        b = (i + 1) > (i * 1); // true
18        System.out.println(b);
19    }
20 }
```

Exercise 02: Functions

```
1 public class Functions {
2     public static void main(String[] args) {
3         System.out.println(cube(5));
4         System.out.println(compare("7",5));
5     }
6
7     static int cube(int x) {
8         // calculate x to the power of 3 the old fashioned way
9         return x*x*x;
10    }
11
12    static boolean compare(String s, int i) {
13        // get int value from within the string
14        int j = Integer.valueOf(s);
15
16        // compare the two int values and return the result
17        return j == i;
18    }
19 }
```

Section 2

Commenting

Comments

- ▶ Ignored by the compiler
- ▶ Information for us humans

Comments

- ▶ Ignored by the compiler
- ▶ Information for us humans

Two types

1 `// This comment ends when the line ends`

2

3 `/* This comments ends with */`

4

5 `/*`

6 `We can include text that spans`

7 `multiple lines`

8 `*/`

`int a = 5;`

*nicht
ausgelassen*

Comments

Example

```
1 public class Example {
2
3     public static void main(String[] args) {
4         // stores how much users want to withdraw
5         int amount = 1500;
6
7         /* the next lines are supposed to calculate
8            the third root of amount, I took the idea from
9            http://www...
10        */
11        int temp = 3;
12        amount = amount / temp;
13        // TODO: Implement me!
14    }
15 }
```

Commenting

- ▶ No fixed rules what to comment

Commenting

- ▶ No fixed rules what to comment
- ▶ Helpful: Your intentions, complex expressions, non-trivial functions
- ▶ Avoid commenting trivial things
- ▶ Keep comments up to date
- ▶ Don't make ASCII art in comments

Section 3

Data Types

Data Types

- ▶ Java: Strong typing
 - ▶ All variables and literals in Java have types
 - ▶ Types are known at compile-time
 - ▶ Benefit: Compiler can prevent type-related errors
 - ▶ E.g., it's a compile error to subtract a String

Primitive Data Types and Objects

- ▶ Two kinds of types
 - ▶ Primitive data types: Built into the language
 - ▶ Type names are reserved keywords in Java
 - ▶ Convention: Lower cased

Primitive Data Types and Objects

- ▶ Two kinds of types
 - ▶ Primitive data types: Built into the language
 - ▶ Type names are reserved keywords in Java
 - ▶ Convention: Lower cased
 - ▶ Non-primitive data types (“reference types”): Established in the library
 - ▶ Type names are defined by library authors
 - ▶ Convention: Upper cased
 - ▶ Reference types can also be defined by us (in the form of classes, to be discussed later)

Primitive Data Types

Keyword	Full name	Values
<code>boolean</code>	Binary value	<code>true</code> , <code>false</code>

Primitive Data Types

Keyword	Full name	Values
<code>boolean</code>	Binary value	<code>true</code> , <code>false</code>
<code>byte</code>	1 Byte (= 8 bit)	-128 to 127
<code>short</code>	short integer (16 bit)	-32 768 to 32 767
<code>int</code>	Integer (32 bit)	-2 147 483 648 to 2 147 483 647
<code>long</code>	long integer (64 bit)	-9 223 372 036 854 775 808 to 9 223 372 036 854 775 807

Primitive Data Types

Keyword	Full name	Values
<code>boolean</code>	Binary value	<code>true</code> , <code>false</code>
<code>byte</code>	1 Byte (= 8 bit)	-128 to 127
<code>short</code>	short integer (16 bit)	-32 768 to 32 767
<code>int</code>	Integer (32 bit)	-2 147 483 648 to 2 147 483 647
<code>long</code>	long integer (64 bit)	-9 223 372 036 854 775 808 to 9 223 372 036 854 775 807
<code>char</code>	Character in UTF-16	<code>'\u0000'</code> to <code>'\uffff'</code> ($65536 = 2^{16}$ symbols)

Primitive Data Types

Keyword	Full name	Values
<code>boolean</code>	Binary value	<code>true</code> , <code>false</code>
<code>byte</code>	1 Byte (= 8 bit)	-128 to 127
<code>short</code>	short integer (16 bit)	-32 768 to 32 767
<code>int</code>	Integer (<u>32 bit</u>)	-2 147 483 648 to 2 147 483 647
<code>long</code>	long integer (64 bit)	-9 223 372 036 854 775 808 to 9 223 372 036 854 775 807
<code>char</code>	Character in UTF-16	<code>'\u0000'</code> to <code>'\uffff'</code> ($65536 = 2^{16}$ symbols)
<code>float</code>	Decimal numbers (<u>32 bit</u>)	$\pm 1.4 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$
<code>double</code>	Decimal numbers (<u>64 bit</u>)	$\pm 4.9 \times 10^{-324}$ to $\pm 1.8 \times 10^{308}$

Table: All primitive data types in Java

Bits and Bytes

▶ 1 Bit:

0

 or


1

▶ 1 Byte = 8 Bit:

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

 -

1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---



Primitive Types in Memory

	0	1	2	3	4	5	6	7	8	9
1x										
2x										
3x										
4x										
5x										
6x										
7x										

```
1 int myIntVariable = 32785;  
2 // compiler knows that myIntVariable  
3 // goes from bits 29 to 60  
4 byte myByteVariable = 4;  
5 // myByteVariable: bits 61 to 68
```

Primitive Types in Memory

	0	1	2	3	4	5	6	7	8	9
1x										
2x										0
3x	0	0	0	0	0	0	0	0	0	0
4x	0	0	0	0	0	1	0	0	0	0
5x	0	0	0	0	0	0	1	0	0	0
6x	1									
7x										

Handwritten annotations: A red arrow points to the '1' in row 6x, column 0. A red bracket groups the '1's in row 4x, column 5 and row 5x, column 6, with the number '30' written next to it. A red arrow points to the '0' in row 2x, column 9. A red arrow points to the '0' in row 3x, column 9.

```

1 int myIntVariable = 32785;
2 // compiler knows that myIntVariable
3 // goes from bits 29 to 60
4 byte myByteVariable = 4;
5 // myByteVariable: bits 61 to 68

```

Primitive Types in Memory

	0	1	2	3	4	5	6	7	8	9
1x										
2x										0
3x	0	0	0	0	0	0	0	0	0	0
4x	0	0	0	0	0	1	0	0	0	0
5x	0	0	0	0	0	0	1	0	0	0
6x	1	0	0	0	0	0	1	0	0	
7x										

```

1 int myIntVariable = 32785;
2 // compiler knows that myIntVariable
3 // goes from bits 29 to 60
4 byte myByteVariable = 4;
5 // myByteVariable: bits 61 to 68

```

Place Value Systems

“Stellenwertsysteme”


$$7 = 100 + 1 \cdot 10 + 2 \cdot 1$$

- ▶ What's the value of 712 (in the decimal system)?

Place Value Systems

“Stellenwertsysteme”

- ▶ What's the value of 712 (in the decimal system)?

$$7 * 100 + 1 * 10 + 2 * 1$$

Place Value Systems

“Stellenwertsysteme”

- ▶ What's the value of 712 (in the decimal system)?
 $7 * 100 + 1 * 10 + 2 * 1$
- ▶ What's the value of x , if x has four places?

Place Value Systems

“Stellenwertsysteme”

- ▶ What's the value of 712 (in the decimal system)?

$$7 * 100 + 1 * 10 + 2 * 1$$

- ▶ What's the value of x , if x has four places?

$$x_3 * 1000 + x_2 * 100 + x_1 * 10 + x_0 * 1$$

↑
Ziffer

Place Value Systems

“Stellenwertsysteme”

- ▶ What's the value of 712 (in the decimal system)?

$$7 * 100 + 1 * 10 + 2 * 1$$

- ▶ What's the value of x , if x has four places?

$$x_3 * 1000 + x_2 * 100 + x_1 * 10 + x_0 * 1$$

- ▶ Even more generic (with n being the number of digits):

$$\begin{aligned}
 x &= x_n * 10^n + x_{n-1} * 10^{n-1} + \dots + x_0 * 10^0 = 1 \\
 &= \sum_{i=0}^n x_i 10^i
 \end{aligned}$$

Place Value Systems

Decimal and others

- ▶ Decimal system uses base 10, but other bases could be used as well

$$\boxed{1 \mid 7 \mid 3 \mid 4} \\ i=4 \quad i=3 \quad i=2 \quad i=1 \quad i=0$$

Place Value Systems

Decimal and others

- ▶ Decimal system uses base 10, but other bases could be used as well

value of x with n places, interpreted to the base b =

$$\sum_{i=0}^n x_i b^i$$

Examples

$$15_{b=10} = 1 * 10^1 + 5 * 10^0 = 15_{b=10}$$

$$15_{b=8} = 1 * 8^1 + 5 * 8^0 = 13_{b=10}$$

$$\text{IV} = 4$$

Place Value Systems

Decimal and others

- Decimal system uses base 10, but other bases could be used as well

value of x with n places, interpreted to the base $b = \sum_{i=0}^n x_i b^i$

8er: 0-7

3er: 0-2

Examples

$$15_{b=10} = 1 * 10^1 + 5 * 10^0 = 15_{b=10}$$

$$15_{b=8} = 1 * 8^1 + 5 * 8^0 = 13_{b=10}$$

$$15_{b=3} = 1 * 3^1 + 5 * 3^0$$

Place Value Systems

Decimal and others

- ▶ Decimal system uses base 10, but other bases could be used as well

$$\text{value of } x \text{ with } n \text{ places, interpreted to the base } b = \sum_{i=0}^n x_i b^i$$

Examples

$$15_{b=10} = 1 * 10^1 + 5 * 10^0 = 15_{b=10}$$

$$15_{b=8} = 1 * 8^1 + 5 * 8^0 = 13_{b=10}$$

$$15_{b=3} = \text{undefined symbol } 5$$

Place Value Systems

Decimal and others

- ▶ Decimal system uses base 10, but other bases could be used as well

$$\text{value of } x \text{ with } n \text{ places, interpreted to the base } b = \sum_{i=0}^n x_i b^i$$

Examples

$$15_{b=10} = 1 * 10^1 + 5 * 10^0 = 15_{b=10}$$

$$15_{b=8} = 1 * 8^1 + 5 * 8^0 = 13_{b=10}$$

$$15_{b=3} = \text{undefined symbol } 5$$

$$11_{b=2} = 1 * 2^1 + 1 * 2^0 = 3_{b=10}$$

Place Value Systems

Decimal and others

- ▶ Decimal system uses base 10, but other bases could be used as well

$$\text{value of } x \text{ with } n \text{ places, interpreted to the base } b = \sum_{i=0}^n x_i b^i$$

Examples

$$15_{b=10} = 1 * 10^1 + 5 * 10^0 = 15_{b=10}$$

$$15_{b=8} = 1 * 8^1 + 5 * 8^0 = 13_{b=10}$$

$$15_{b=3} = \text{undefined symbol } 5$$

$$\underline{11}_{b=2} = \underline{1 * 2^1} + \underline{1 * 2^0} = \underline{3}_{b=10}$$

$$\underline{1010}_{b=2} = \underline{1 * 2^3} + \underline{0 * 2^2} + \underline{1 * 2^1} + \underline{0 * 2^0} = \underline{10}_{b=10}$$

Place Value Systems

Maximal Numbers

- ▶ Highest number depends on the number of places
- ▶ For 3 places
 - ▶ Decimal ($b = 10$): $999_{b=10} = 1000_{b=10} - 1$
 - ▶ Octal ($b = 8$): $777 = 511_{b=10} = 1000_{b=8} - 1$
 - ▶ Binary ($b = 2$): $111 = 7_{b=10} = 1000_{b=2} - 1$

Place Value Systems

Maximal Numbers

- ▶ Highest number depends on the number of places
- ▶ For 3 places
 - ▶ Decimal ($b = 10$): $999_{b=10} = 1000_{b=10} - 1$
 - ▶ Octal ($b = 8$): $777 = 511_{b=10} = 1000_{b=8} - 1$
 - ▶ Binary ($b = 2$): $111 = 7_{b=10} = 1000_{b=2} - 1$
- ▶ More general
 - ▶ Maximal value with n places to base b : $b^n - 1$
 - ▶ I.e.: 4 bits can distinguish 16 different states

Place Value Systems

Maximal Numbers

- ▶ Highest number depends on the number of places
- ▶ For 3 places
 - ▶ Decimal ($b = 10$): $999_{b=10} = 1000_{b=10} - 1$
 - ▶ Octal ($b = 8$): $777 = 511_{b=10} = 1000_{b=8} - 1$
 - ▶ Binary ($b = 2$): $111 = 7_{b=10} = 1000_{b=2} - 1$
- ▶ More general
 - ▶ Maximal value with n places to base b : $b^n - 1$
 - ▶ I.e.: 4 bits can distinguish 16 different states

Maximal
 $15 = 2^4 - 1$

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Integral Data Types

Keyword	Full name	Values
<code>byte</code>	1 Byte (= 8 bit)	-128 to 127
<code>short</code>	short integer (16 bit)	-32 768 to 32 767
<code>int</code>	Integer (32 bit)	-2 147 483 648 to 2 147 483 647
<code>long</code>	long integer (64 bit)	-9 223 372 036 854 775 808 to 9 223 372 036 854 775 807

- ▶ Integral data types defined over the number of places they occupy
- ▶ Shorter types consume less memory
- ▶ I.e.: If you know a value will never be higher than 127, use a byte
 - ▶ E.g., For calendar weeks

Integral Data Types

Literals

- ▶ By default: full numbers within expressions are of type `int`

```
1 int myIntValue = 27; // literal int value assigned to an int variable
2 byte myByteValue = 27; // literal int value assigned to a byte variable
3 long myLongValue = 27; // literal int assigned to a long variable
4
5 long myLargeLongValue = 2700000000000000000L;
6 // append L to enforce a long literal
7 long mySmallLongValue = 27L; // also works for small numbers
```

- ▶ Why can we assign an int literal to a byte/long/short variable?
→ Implicit casting (see below)!

Section 5

Exercise

Nur
Aufgabe 3c,
also die Funktion
isOdd()