

4. Tutorium

JavaDoc, Methoden, Arrays

Tutorium 14

Péter Bohner | 23.11.2022



Inhaltsverzeichnis

1. Übungsblatt 1

2. Wiederholung

3. JavaDoc

4. Methoden

5. Arrays

6. Aufgabe

Übungsblatt 1
○

Wiederholung
○○○

JavaDoc
○○○○○

Methoden
○○○○○○○○○○○○○

Arrays
○○○○○○○○○○○

Aufgabe
○○○○○

Ende
○

Übungsblatt 1

- Alle 24 von euch haben Abgegeben, super!
- **WENN SICH ARTEMIS BEI IRGENDWAS BESCHWERT, DANN GIBT ES AUTOMATISCH 0 PUNKTE!**
→ Kompilierfehler, Mandatory Test fails nicht ignorieren
- Dateien: Groß- und Kleinschreibung beachten! Dateiname muss genau der Klassenname sein (.java nicht vergessen) Dateien nur im src/ Ordner ablegen.
- Aufgabenstellungen genau lesen: **Wer lesen kann ist klar im Vorteil!**
- Beachtet bitte Namenskonventionen und richtige Einrückung (4 Spaces)
- Aufgabe A: bei allen gut
- Aufgabe B: isEmpty() ignoriert kein Whitespace, ist also falsch. isBlank() ist richtig
- Aufgabe C: Nicht vergessen den Scanner zu schließen.
- Aufgabe D: Keine Initialisierungen nur Deklarationen, keine main-Methode, Adresse nicht als String, Jede Klasse in eigene Datei.

Übungsblatt 1	Wiederholung	JavaDoc	Methoden	Arrays	Aufgabe	Ende
●	○○○	○○○○○	○○○○○○○○○○○○	○○○○○○○○○○	○○○○○	○

Wiederholung

Mit welchem Schlüsselwort werden Variablen unveränderbar gemacht?

final

Füllt die gegebene Tabelle aus ($x = z = \text{true}$).

Ausdruck	y = ?	Ergebnis
$x \ \&\& \ y == y \ \&\& \ x$	true	true
$(z \wedge z) \ \&\& \ y$	false/true	false
$y == \text{false} \ \ y == \text{true}$	false/true	true
$!((z \ \ z) \ \& \ !y)$	false	false

Wiederholung

Welche Ausgabe erzeugt der folgende Code:

```
1 String s1 = "Hallo ";  
2 String s2 = s1;  
3 s2 = s2 + "Welt";  
4 System.out.println("s1: " + s1 + ", s2: " + s2);
```

Ausgabe: `s1: Hallo , s2: Hallo Welt` da Strings immutable sind

Womit vergleicht man Strings?

`string1.equals(string2)`

Sei `int i = 1`. Welchen Wert hat `i` nach der Ausführung folgendes Ausdrucks? `(1 == 1) || (i++ == 5)`

1, wegen der Kurzschlussauswertung wird die zweite Bedingung nie ausgeführt

Mit welchem Schlüsselwort verlässt man einen case in einem switch-statement?

`break`

Wiederholung

Wie oft wird ein do-while-Schleife mindestens ausgeführt?

Einmal, bevor die Schleifenbedingung zum ersten Mal geprüft wird

Was trifft idealerweise auf eine for-Schleife zu?

- Anzahl der Iterationen bekannt
- Die Schrittanweisung wird nach jeder Rumpfausführung ausgeführt
- for-Schleifen können verschachtelt werden

Mit welchem Schlüsselwort wird eine Schleifeniteration übersprungen?

continue

Wie ist die Syntax von einem Konstruktor einer Klasse?

Klassenname(Parameterliste)

Eine Klasse kann nur einen Konstruktor haben.

Falsch, man kann die Parameterliste überladen

Mit welchem Schlüsselwort unterscheidet man zwischen Attributen und anderen Variablen gleichen Namens? **this**

JavaDoc - Was ist das?

Javadoc...

- ... beschreibt Klassen, Methoden, Konstruktoren und Felder
- ... wird direkt in den Code geschrieben
- ... erleichtert das Verständnis

Regeln und Ziele

- **einheitlich** auf Deutsch oder auf Englisch
- vollständige und aussagekräftige Beschreibung

Javadoc zur Java API

JavaDoc für Klassen

- beschreibt, welchen Zweck die Klasse erfüllt und welche Funktionalität sie aufweist
- mögliche Verwendungszwecke können erwähnt werden
- der Autor wird angegeben
- die Versionsnummer wird angegeben

Syntax

```
/**  
 * Kurze aber aussagekräftige Beschreibung der Klasse.  
 * @author Autor  
 * @version versionsNummer  
 */  
class ClassName {
```


JavaDoc für Klassen - Beispiel

```
/**  
 * Modelliert einen zweidimensionalen Vektor.  
 * Der Vektor besteht aus einer x- und y-Komponente.  
 * @author Peter Bohner  
 * @author uqknc  
 * @version 1.0  
 */  
class Vector2D {  
    double x;  
    double y;  
}
```

JavaDoc für Konstruktoren

- beschreibt Zweck der Konstruktors
- beschreibt, in welchem Kontext der Kontruktor verwendet werden kann
 - Spezialfälle, die das Verhalten betreffen
- alle Parameter werden beschrieben (Zweck und eventuell Wertebereich)

Syntax

```
/**  
 * Kurze aber aussagekräftige Beschreibung von Zweck und Kontext des Konstruktors.  
 * Sonderfälle werden beschrieben.  
 * @param parameter Beschreibung des Parameters  
 */  
KlassenName(Datentyp parameter, ...) {
```

JavaDoc für Konstruktoren - Beispiel

```
/**
 * Konstruiert einen zwei dimensionalen Vektor mit zwei übergebenen Komponenten.
 *
 * @param x die x-Komponente des Vektors
 * @param y die y-Komponente des Vektors
 */
Vector2D(double x, double y) {
    this.x = x;
    this.y = y;
}
```

Methoden - Einführung

Methoden realisieren das dynamische Verhalten von Objekten und führen Berechnungen durch.

Schema:

```
Rückgabetyyp methodenName(Parameterliste) {
    // Methodenrumpf
}
```

Der Rückgabetyyp **void** steht dabei für keine Rückgabe!

- Parameterliste: Datentyp₁ parameter₁, Datentyp₂ parameter₂, ..., Datentyp_n parameter_n

Methoden - Aufbau

Der Methodenkopf besteht aus dem Rückgabety und der Methodensignatur.

Die Methodensignatur besteht aus:

- dem Namen der Methode
- der Anzahl der Parameter
- der Reihenfolge der Parameter
- der Typen der Parameter

Rückgabety `methodName(Parameterliste){ ... }`
└──────────────────────────────────┘
 Methodensignatur
└──────────────────────────────────┘
 Methodenkopf

Methoden - Zugriff auf Attribute

In Methoden kann auch auf Attribute des Objekts zugegriffen werden:

```
class Person {  
    String name = "Sven";  
  
    void greet(String name) {  
        System.out.println("Hallo " + name + ", ich heiße " + this.name + ".");  
    }  
}
```

this erlaubt Unterscheidung zwischen Parameter und Attribut

Methoden - Aufruf

- Methoden können nur auf Objekten aufgerufen werden!
 - Dazu wird die Syntax `objekt.methode()` verwendet.

Beispiel

```
Person person = new Person();  
person.greet("Peter");
```

Ausgabe:

```
Hallo Peter, ich heiße Sven.
```

Methoden - Rückgabe

Wird ein anderer Rückgabetypp als **void** gewählt, muss die Methode einen Wert vom Rückgabetypp mit dem Schlüsselwort **return** zurückgeben.

Beispiel

```
int sum(int a, int b) {  
    return a + b;  
}
```


Methoden - return bei void

return kann aber auch bei Methoden mit Rückgabetyt **void** benutzt werden:
z.B. um eine Methode gezielt zu verlassen

Beispiel

```
class Job {  
    boolean started = false;  
    void startJob() {  
        if (started)  
            return;  
        started = true;  
        // Do the Job  
    }  
}
```

Klassenmethoden

Methoden die mit dem Modifier **static** beginnen nennt man Klassenmethoden. Diese gehören zu der Klasse statt zu einem Objekt.

```
class Math {  
    static int sum(int a, int b) {  
        return a + b;  
    }  
}
```

```
int c = Math.sum(5, 8); // Nach Ausführung: c = 13
```

Es wird **kein** Objekt benötigt, um die Methode auszuführen.

Schema: Klasse.methodenName();

this kann in Klassenmethoden nicht genutzt werden

Hilfsmethoden

Was sind Hilfsmethoden?

- Methoden, die häufig verwendete Funktionalität ausführen
- Können statisch oder nicht statisch sein

statisch

- Klassenmethoden
- hängt nicht von Attributen einer Klasse ab
- `static int sum(int a, int b)`

nicht statisch

- gliedern innerhalb einer Klasse Funktionalität aus
- hängt von Attributen ab
- `int sum()` (Summe zweier Attribute)

Überladen

Es können mehrere Methoden mit dem gleichen Namen existieren, wenn sich diese in den Datentypen, Reihenfolge und/oder der Anzahl ihrer Parameter(d.h. in ihrer Signatur) unterscheiden.

Ok

```
int sum (int a, int b) { ... }  
double sum (double a, double b) { ... }  
int sum (int a, int b, int c) { ... }
```

Nicht Ok

```
int sum (int a, int b) { ... }  
long sum (int a, int b) { ... }  
int sum (int a, int c) { ... }
```

Methoden gleichen Namens, die sich nur im Rückgabebetyp unterscheiden, sind nicht möglich.

Methoden - Übung

Aufgabe

Schreibe eine Methode die eine Ganzzahl an nimmt und zurück gibt ob diese gerade ist.

Tipp

- 1 Mit dem Modulo Operator % erhaltet ihr den Rest der Ganzzahldivision.
- 2 Der Rückgabetyt euer Methode muss **boolean** sein.

Lösung

```
static boolean isEven(int number) {  
    return (number % 2 == 0);  
}
```

Methoden - JavaDoc

- beschreibt Zweck der Methode
- beschreibt in welchem Kontext die Methode verwendet werden kann
 - Vor- und Nachbedingungen der Ausführung
 - Spezialfälle, die das Verhalten betreffen
- alle Parameter werden beschrieben (Zweck und Wertebereich)
- der Rückgabetyt wird beschrieben (Inhalt)

Syntax

```
/**  
 * Kurze aber aussagekräftige Beschreibung von Zweck und Kontext der Methode.  
 * @param parameter Beschreibung des Parameters  
 * @return Beschreibung Rückgabewert  
 */  
RückgabeDatentyp methodenName(DatenTyp parameter) {
```

Methoden - JavaDoc

```

/**
 * Returns the absolute value of an int value.
 * If the argument is not negative, the argument is returned.
 * If the argument is negative, the negation of the argument is returned.
 *
 * Note that if the argument is equal to the value of Integer.MIN_VALUE,
 * the most negative representable int value, the result is that same value, which is negative.
 *
 * @param a the argument whose absolute value is to be determined
 * @return the absolute value of the argument.
 */
static int abs(int a) {
    return (a < 0) ? -a : a;
}

```

Arrays - Einführung

- sind Folgen von n Elementen **desselben** Datentyps
- Zugriff auf einzelne Elemente erfolgt über Index
 - Index des ersten Elements: 0
 - Index des letzten Elements: $n - 1$
- Arrays werden intern wie Objekte behandelt
 - Referenzierung des Arrays
- einfachste Datenstruktur

Arrays - Deklaration

- Deklariere Array vom Datentyp „Typ“ mit dem Bezeichner „name“:
 - `Typ[] name;`
- Reserviere im gleichen Schritt Speicher für N Elemente und initialisiere diese mit ihren Default Werten:
 - `Typ[] name = new Typ[N];`
- Deklariere ein Array und fülle es direkt mit spezifizierten Werten auf:
 - `Typ[] name = { wert1, wert2, ..., wertn};`
 - Array hat damit automatisch die Größe n .

Achtung!

Die Größe eines Arrays ist nach anfordern des Speicherplatzes **fest!**

Arrays - Beispiele

```
double[] a = new double[8];
```

```

a =  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
      0    1    2    3    4    5    6    7

```

```
String[] b = new String[5];
```

```

b =  null  null  null  null  null
      0    1    2    3    4

```

```
int[] c = {3, -5, 9, 11};
```

```

c =   3   -5   9   11
      0    1    2    3

```

Arrays - Zugriff

- Zugriff auf ein Element des initialisierten Arrays *a* erfolgt mit:
 - `a[index]`
 - Für Index gilt: $0 \leq \text{index} < \text{a.length}$
 - **Sonst:** `ArrayIndexOutOfBoundsException`

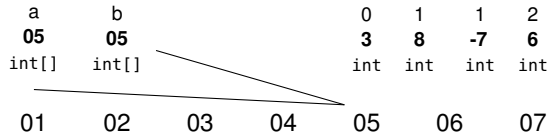
Beispiel

```
int[] a = {3, 7, 1, -5, 9};  
System.out.println(a[3]); // Ausgabe: -5  
a[3] = 11;  
System.out.println(a[3]); // Ausgabe: 11  
  
a[5] = 7; // ArrayIndexOutOfBoundsException
```

Arrays - Aufbau im Speicher

Arrays werden in Java wie Objekte behandelt:

- `int[] a = {3, 8, 6};`
 - erstellt ein Array und weist der Variable `a` die Referenz auf das Array zu
- `int[] b = a;`
 - `a` und `b` referenzieren dasselbe Array
- `b[1] = -7;`
 - Der Ausdruck `a[1]` wertet zu -7 aus.



Arrays - Iteration

Iteriere über Array `int[] a = {3, 8, 6};` um Summe der Elemente zu bestimmen.

Möglichkeit 1

- Iteriere mit `for`-Schleife über mögliche Indizes
- greife im Schleifenrumpf über Index auf Elemente zu

Beispiel:

```
int sum = 0;
for (int i = 0; i < a.length; i++) {
    sum += a[i];
}
```

Möglichkeit 2

- Nutze for-each Schleife
- Führe Anweisung für jedes Element des Arrays aus
- Schema:
 - `for` (Typ element : array) { }
 - „Typ“ ist Datentyp der Elemente des Arrays

Beispiel:

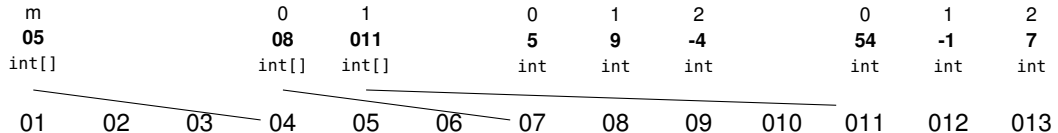
```
int sum = 0;
for (int summand : a) {
    sum += element;
}
```

Mehrdimensionale Arrays

- sind „Arrays von Arrays“
- häufiger Anwendungsfall: Matrizen
- Beispiel:

■ `int[][] m = new int [2][3];`

- erzeugt 2-dimensionales Array
- `m` ist ein Array der Größe 2 von Arrays der Größe 3 vom Datentyp `int`



■ `int[][] a = new int [3][];` erzeugt Array mit 3 Referenzen des Typs `int[]`

Mehrdimensionale Arrays - Iteration

Iterieren über Mehrdimensionale Arrays meist mit for-Schleifen:

```
for (int i = 0; i < m.length; i++) {  
    for (int j = 0; j < m[i].length; j++) {  
        // Tue etwas mit Element m[i][j]  
    }  
}
```

Arrays - Aufgabe

Aufgabe

Vervollständige den unten stehenden Code, um das Produkt von zwei Elementen mit gleichem Index aus den untenstehenden **gleich langen** Arrays zu berechnen.

Speichere das Ergebnis in einem weiteren Array und gebe das Ergebnis auf der Konsole aus.

Code

```
class ArrayMultiplier {  
    public static void main(String[] args) {  
        int[] a = {-1, 5, 3, 5, 8};  
        int[] b = {4, 3, 8, 1, -1};  
        // Ergänze deinen Code hier  
    }  
}
```


Arrays - Lösung

```
class ArrayMultiplier {  
    public static void main(String[] args) {  
        int[] a = {-1, 5, 3, 5, 8};  
        int[] b = {4, 3, 8, 1, -1};  
  
        int[] result = new int[5];  
  
        for (int i = 0; i < a.length; i++) {  
            int product = a[i] * b[i];  
            result[i] = product;  
            System.out.println(a[i] + " * " + b[i] + " = " + product);  
        }  
    }  
}
```

Aufgabe

Buch

Modelliert eine Klasse `Book`, die die Attribute `Titel`, `Autor`, `Erscheinungsjahr` und `Id` hat. Die `Id` soll unveränderbar sein. Schreibt einen Konstruktor, der alle Attribute übergeben bekommt und diese setzt. Außerdem soll ein Konstruktor definiert werden, der immer das Jahr 2020 als Erscheinungsdatum setzt.

Für Schnelle

Schreibt ein ausführbares Programm, mit dem pro Konstruktor jeweils ein Objekt erstellt wird und `Titel` und `Erscheinungsjahre` der Bücher jeweils in getrennten Zeilen ausgegeben werden.

```
/**  
 * Modelliert ein Buch zur Vereinfachung der Digitalisierung von Büchern.  
 * Ein Buch besteht aus einer eindeutigen Identifikationsnummer,  
 * dem Erscheinungsjahr, einem Titel und dem Namen des Autors.  
 *  
 * @author Leon Wittemund  
 * @version 1.0  
 */  
class Book {  
    final int id;  
    int year;  
    String title;  
    String author;
```

```
/**  
 * Erstellt Instanzen der Klasse Buch  
 *  
 * @param id eindeutige Identifikationsnummer  
 * @param year Erscheinungsjahr des Buches  
 * @param title Buchtitel  
 * @param author Buchautor  
 */  
Book(int id, int year, String title, String author) {  
    this.id = id;  
    this.year = year;  
    this.title = title;  
    this.author = author;  
}
```

```
/**  
 * Erstellt Instanzen der Klasse Buch,  
 * wobei das Erscheinungsjahr auf das Jahr 2020 gesetzt wird.  
 *  
 * @param id eindeutige Identifikationsnummer  
 * @param title Buchtitel  
 * @param author Buchautor  
 */  
Book(int id, String title, String author) {  
    this(id, 2020, title, author);  
}  
}
```

Lösung

```

/**
 * Die Klasse BookApp ermöglicht es,
 * die Funktionen der Klasse Book benutzen zu können.
 *
 * @author Leon Wittemund
 * @version 1.0
 */
class BookApp {
    /**
     * Methode, mit der die Funktionen der Klasse Book getestet werden.
     * @param args übergebene Argumente
     */
    public static void main(String[] args) {
        Book bookA = new Book(1, "Java ist auch eine Insel", "Christian Ullenboom");
        Book bookB = new Book(20, 2009, "Clean Code", "Robert Cecil Martin");
        System.out.println(bookA.title + ": " + bookA.year);
        System.out.println(bookB.title + ": " + bookB.year);
    }
}

```

Bis zum nächsten Tutorium am 30.11.2022!

Übungsblatt 1
○

Wiederholung
○○○

JavaDoc
○○○○○

Methoden
○○○○○○○○○○○○○

Arrays
○○○○○○○○○○○

Aufgabe
○○○○○

Ende
●