

3. Tutorium

Enums, Strings, Konstruktoren, Methoden, Code Stil

Tutorium 14

Péter Bohner | 16.11.2022



Tutorium 14

-11-16

www.kit.edu

Inhaltsverzeichnis 1. Wiederholung 2. Datentypen III Strings 3.1 Vergleiche 4. Konstruktoren 5. Methoden 6. Sauber Programmieren Datentypen III Strings Konstruktoren Methoden Sauber Programmieren 00000 00000 000000000000 000 16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium

Ende

0





Wiederholung

16.11.2022



Mit welchem Operator berechnet man den Rest der Ganzzahldivision?

Strings

00000

Datentypen III

Péter Bohner: Tutorium 14

Konstruktoren

00000

Methoden

000000000000

Sauber Programmieren

000

Ende

Programmieren Tutorium

Tutorium 14 ─Wiederholung

─Wiederholung



Mit welchem Operator berechnet man den Rest der Ganzzahldivision?





Mit welchem Operator berechnet man den Rest der Ganzzahldivision?

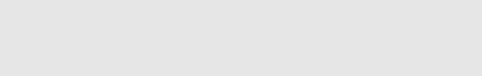
Beispiel: 5 % 3 ergibt 2 Modulo: %



─Wiederholung ─Wiederholung

Tutorium 14

Mit welchem Operator berechnet man den Rest der Ganzzahldivision? Modulo: % Beispiel: 5 % 3 ergibt 2



Datentypen III Strings Konstruktoren Methoden 00000 00000 000000000000 000 00

Ende

Wiederholung

Sauber Programmieren

16.11.2022 Programmieren Tutorium Péter Bohner: Tutorium 14



Mit welchem Operator berechnet man den Rest der Ganzzahldivision? Modulo: % Beispiel: 5 % 3 ergibt 2 y = ++x entspricht...

─Wiederholung

Tutorium 14 ─Wiederholung

Modulo: % Beispiel: 5 % 3 eroibt 2 v = ++x entapricht...

Mit welchem Operator berechnet man den Rest der Ganzzahldivision?



00

00000

000



Ende

- 16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium



Mit welchem Operator berechnet man den Rest der Ganzzahldivision?

Strings

00000

Modulo: % Beispiel: 5 % 3 ergibt 2

Datentypen III

y = ++x entspricht... x = x + 1; y = x;

Methoden

000000000000

Sauber Programmieren

000

Konstruktoren

00000

Tutorium 14 ─Wiederholung

─Wiederholung

Mit welchem Operator berechnet man den Rest der Ganzzahldivision? Modulo: % Beispiel: 5 % 3 ergibt 2 v = ++x entapricht...

Ende

16.11.2022

Wiederholung

00

Péter Bohner: Tutorium 14 Programmieren Tutorium



Mit welchem Operator berechnet man den Rest der Ganzzahldivision?

Modulo: % Beispiel: 5 % 3 ergibt 2

y = ++x entspricht...

x = x + 1; y = x;

Welches Schlüsselwort steht für eine Referenz auf "kein Objekt"?

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 000000000000 000

Tutorium 14 ─Wiederholung ─Wiederholung

Mit welchem Operator berechnet man den Reat der Ganzzahldivision? Modulo: % Beispiel: 5 % 3 ergibt 2 v = ++x entapricht...

x = x + 1; y = x; Welches Schlüsselwort steht für eine Referenz auf "kein Objekt"?

16.11.2022



```
Mit welchem Operator berechnet man den Rest der Ganzzahldivision?
```

Modulo: % Beispiel: 5 % 3 ergibt 2

y = ++x entspricht...

x = x + 1; y = x;Welches Schlüsselwort steht für eine Referenz auf "kein Objekt"?

00000

00

null





Datentypen III

Péter Bohner: Tutorium 14

Strings

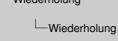
000000000000

000





Tutorium 14 ─Wiederholung

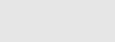


















Wiederholung



```
Mit welchem Operator berechnet man den Rest der Ganzzahldivision?
```

Modulo: % Beispiel: 5 % 3 ergibt 2

y = ++x entspricht...

x = x + 1; y = x;Welches Schlüsselwort steht für eine Referenz auf "kein Objekt"?

null

Was passiert bei folgender Code-Ausführung: Rectangle rec = null; rec.a = 20f;?

Strings Methoden Sauber Programmieren Ende Wiederholung Datentypen III Konstruktoren 00000 00000 000000000000 000

Tutorium 14 ─Wiederholung ─Wiederholung

Mit welchem Operator berechnet man den Rest der Ganzzahldivision? Modulo: % Beispiel: 5 % 3 ergibt 2 v = ++x entapricht...



```
Mit welchem Operator berechnet man den Rest der Ganzzahldivision?
```

Modulo: % Beispiel: 5 % 3 ergibt 2

y = ++x entspricht... x = x + 1; y = x;

Welches Schlüsselwort steht für eine Referenz auf "kein Objekt"?

null Was passiert bei folgender Code-Ausführung: Rectangle rec = null; rec.a = 20f;?

NullPointerException

Strings Methoden Sauber Programmieren Ende Wiederholung Datentypen III Konstruktoren 00000 00000 000000000000 000 00

Tutorium 14 ─Wiederholung

Mit welchem Operator berechnet man den Rest der Ganzzahldivision? Modulo: % Beispiel: 5 % 3 ergibt 2 v = ++x entapricht... Welches Schlüsselwort steht für eine Referenz auf "kein Obiekt" Was passiert bei folgender Code-Ausführung: Rectangle rec = null: rec.a = 20f:1

─Wiederholung

16.11.2022

Péter Bohner: Tutorium 14



Programmieren Tutorium

Eine Liste der möglichen Werte des Typs wird explizit angegeben Schema: enum Name { Liste der Werte }

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 000000000000 000

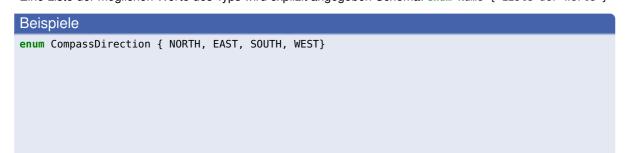
☐ Datentypen III Aufzählungs-Datentyp

Tutorium 14

Eine Liste der möglichen Werte des Typs wird explizit angegeben Schema: eeue Name { Liste der Werte }



Eine Liste der möglichen Werte des Typs wird explizit angegeben Schema: enum Name { Liste der Werte }



Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 000000000000 000

Tutorium 14 ☐ Datentypen III Aufzählungs-Datentyp

Eine Liste der möglichen Werte des Typs wird explizit angegeben Schema: enun Name { Liste der Werte } enum CompassDirection (NORTH, EAST, SOUTH, WEST)



Eine Liste der möglichen Werte des Typs wird explizit angegeben Schema: enum Name { Liste der Werte }

```
Beispiele

enum CompassDirection { NORTH, EAST, SOUTH, WEST}

Color.java

enum Color {
    RED, GREEN, BLUE
}

Enums können in eine eigene Java-Datei oder innerhalb einer Klasse definiert werden.
```

Tutorium 14

Datentypen III

Aufzählungs-Datentyp

Eine Late der nöglichen Werke das Tips mit exploit angegleben Schemit erem kans (Lists der imrite)
Bioglobe
man Capssalforstan (1607), DAT, 1507), WET)
Citripies
mit ()
m



Eine Liste der möglichen Werte des Typs wird explizit angegeben Schema: enum Name { Liste der Werte }

```
Beispiele

enum CompassDirection { NORTH, EAST, SOUTH, WEST}

Color.java

enum Color {
    RED, GREEN, BLUE
}

Enums können in eine eigene Java-Datei oder innerhalb einer Klasse definiert werden.
```

Operationen: Müssen vom Benutzer angegeben werden analog zu Klassenmethoden

Tutorium 14

Datentypen III

Aufzählungs-Datentyp

Konstanten



Variablem mit festem, unveränderlichen Wert werden in Java durch das Schlüsselwort final gekennzeichnet:

☐ Datentypen III └─Konstanten

Tutorium 14

Variablem mit festem, unveränderlichen Wert werden in Java durch das Schlüsselwort fizzal gekennzeichnet:

Wiederholung

Datentypen III

Strings

Konstruktoren

00000

Methoden

000000000000

Sauber Programmieren 000

Ende

Programmieren Tutorium

00000 16.11.2022 Péter Bohner: Tutorium 14

Konstanten



Variablem mit festem, unveränderlichen Wert werden in Java durch das Schlüsselwort final gekennzeichnet:

lokale Variablen und Attribute

final Typ variablenName;

oder mit Initialisierung:

final Typ variablenName = Wert;

16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium

Tutorium 14

Datentypen III

Konstanten

Verlation mit lander, verwächsche Weit werden in Jans durch das Gelfüssekent f lant gelennsachset.

John Michaelm und Abschag

Taut Typ verzächnisse,
den erm blieberung der Sterken,
den erm blieberung.

Taut Typ verzächnisse * Wert;

Konstanten



Variablem mit festem, unveränderlichen Wert werden in Java durch das Schlüsselwort final gekennzeichnet:

lokale Variablen und Attribute

final Typ variablenName;

oder mit Initialisierung: final Typ variablenName = Wert;

Klassenkonstanten

Schema: static final Typ NAME = Wert;

Beispiel: static final float GRAVITATION_EARTH = 9.80665f;

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 000000000000 000

16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium Tutorium 14 ☐ Datentypen III └─Konstanten

Variablem mit festem, unveränderlichen Wert werden in Java durch das Schlüsselwort ninst gekennzeichnet final Typ variablenName; odermit Initialisierung: final Typ variablenName - Wert; Schema: static final Typ NAME = Wert;
Beispiel: static final float GRAVITATION_EARTH = 9.886657

Wiederholung Zeichenkette



String

- Zusammensetzung von chars
- Ist kein primitiver Datentyp, sondern eine Klasse
- Außerdem sind ein paar Besonderheiten z.B. beim Vergleichen zu beachten
- Operation: Konkatenation

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende •0000 00000 000000000000 000

Tutorium 14 Strings

Wiederholung Zeichenkette

lat kein primitiver Datentyp, sondern eine Klasse

 Außerdem sind ein paar Besonderheiten z.B. beim Veroleichen zu beachten Operation: Konkatenation

16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium

Konkatenation



Strings werden mit dem + Operator konkateniert (verkettet):

```
String s1 = "Dies ist ";
String s2 = "eine Konkatenation";
String s3 = s1 + s2 + "!";
String s4 = "Dies ist + keine Konkatenation!";
System.out.println(s3);
System.out.println(s4);
```

Ausgabe:

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 000000000000 000

16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium Tutorium 14 Strings

String 41 - "Dies 541 "; String 64 - "Diet 167 + beine Horkstenstion!" System.out.println(43);

Strings werden mit dem + Operator konkateniert (verkettet)

System.out.println(s4);

Ausgabe:

Konkatenation



Strings werden mit dem + Operator konkateniert (verkettet):

```
String s1 = "Dies ist ";
String s2 = "eine Konkatenation";
String s3 = s1 + s2 + "!";
String s4 = "Dies ist + keine Konkatenation!";
System.out.println(s3);
System.out.println(s4);
```

Ausgabe:

Dies ist eine Konkatenation!
Dies ist + keine Konkatenation!

 Wiederholung
 Datentypen III
 Strings
 Konstruktoren
 Methoden
 Sauber Programmieren
 Ende

 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○

Tutorium 14
Strings
Konkatenation

Drings weeke rid dan - Operator lookstational (perfecting)

- 1000 4.1 - 1000 6.1 - 1000

+ keine Konkatenatio



Was wird Ausgegeben?

```
class References {
    public static void main(String[] args) {
       String s1 = "Vorsicht mit ";
       String s2 = s1;
       s2 = s2 + "Strings";
       System.out.println("s1: " + s1);
       System.out.println("s2: " + s2);
```

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren 00000 00000 000000000000 000

Tutorium 14 Strings Was wird Ausgegeben? class References (public static wold main(String() args) String 61 = "Yersicht mit "; Übungen G2 = G2 + "Strings"; System.out.println("G1: " + G1); hystem out printing (CD) (1 + GD)

Ende



Was wird Ausgegeben?

```
class References {
   public static void main(String[] args) {
      String s1 = "Vorsicht mit ";
      String s2 = s1;
      s2 = s2 + "Strings";
      System.out.println("s1: " + s1);
      System.out.println("s2: " + s2);
   }
}
```

```
$ java References
s1: Vorsicht mit
s2: Vorsicht mit Strings
```

Tutorium 14
Strings
L
Übungen

5 java References al: Vorsicht mit a2: Vorsicht mit Strings

Warum ist das so?



Konkatenation mit +

16.11.2022

■ Bei der Nutzung von + wird ein neues Objekt erstellt

Péter Bohner: Tutorium 14

- Die Referenz zu "s1" wird dabei gelöscht
- Deshalb ändert sich nur "s2"

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 000000000000 000

Programmieren Tutorium

Tutorium 14 Strings

─Warum ist das so?

Bei der Nutzung von + wird ein neues Objekt erstellt

 Die Referenz zu "s1" wird dabei gelöscht Deshalb ändert sich nur "s2"

Warum ist das so?



Tutorium 14

Strings ─Warum ist das so?

 Bei der Nutzung von + wird ein neues Objekt erstellt Die Referenz zu "s1" wird dabei gelöscht Deshalb ändert sich nur "s2" String ist smutable, also unveränderbar

Konkatenation mit +

Bei der Nutzung von + wird ein neues Objekt erstellt

Strings

- Die Referenz zu "s1" wird dabei gelöscht
- Deshalb ändert sich nur "s2"

immutable

Datentypen III

String ist immutable, also unveränderbar

00000 00000 000000000000 000

Methoden

Konstruktoren

Programmieren Tutorium

Ende

Sauber Programmieren

Wiederholung



Was wird Ausgegeben?

```
class References2 {
   public static void main(String[] args) {
      String s1 = "Vorsicht mit ";
      String s2 = s1;
      s2.concat("Strings");
      System.out.println("s1: " + s1);
      System.out.println("s2: " + s2);
   }
}
```

e

Tutorium 14

Strings

Übungen

Was with Ausgraphen?

Class Reference? {
patts code and instituting; angl {
patts code and instituting; angl {
patts code and institution; or ;

Class Reference; code;

Class Reference; code;

Application of anisotic (20.1 * 40);

}

}

11/34 16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium



Was wird Ausgegeben?

```
class References2 {
   public static void main(String[] args) {
        String s1 = "Vorsicht mit ";
        String s2 = s1;
        s2.concat("Strings");
        System.out.println("s1: " + s1);
        System.out.println("s2: " + s2);
   }
}
```

```
$ java References2
s1: Vorsicht mit
s2: Vorsicht mit
```

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende

Tutorium 14

Strings

Übungen





Was wird Ausgegeben?

```
class References2 {
   public static void main(String[] args) {
      String s1 = "Vorsicht mit ";
      String s2 = s1;
      s2.concat("Strings");
      System.out.println("s1: " + s1);
      System.out.println("s2: " + s2);
   }
}
```

```
$ java References2
s1: Vorsicht mit
s2: Vorsicht mit
```

String ist immutable: .concat() verändert String nicht, es wird ein neuer String zurückgegeben

 Wiederholung
 Datentypen III
 Strings
 Konstruktoren
 Methoden
 Sauber Programmieren
 Ende

 0
 00
 0000
 0000
 000000000000
 000
 0
 0

Tutorium 14
—Strings
—Übungen

Was with Anapopular?

Lakes Section and Control of the Control of

Strings vergleichen



Vergleiche von primitiven Datentypen (==, >=, ...)

- Können direkt verglichen werden
- Es wird direkt der im Speicher hinterlegte Wert verglichen

Tutorium 14
Strings
Vergleiche
Strings vergleichen

Vergleiche von primitiven Datentypen (---, ---,

« Körnen diest anglichen werden

« Es wird diest der im Specimer hörenligte Wert weglichen

Strings vergleichen



Vergleiche von primitiven Datentypen (==, >=, ...)

- Können direkt verglichen werden
- Es wird direkt der im Speicher hinterlegte Wert verglichen

Vergleiche von Strings

- == vergleicht nur die im Speicher hinterlegten Werte beider Variablen
- Bei Objekten werden also nur die Referenzen verglichen
- Attributwerte werden ignoriert

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende
0 00 0000 0000 000000000 000 0

Tutorium 14
Strings
Vergleiche
Strings vergleichen

Strings vergleichen



Programmieren Tutorium

Vergleiche von primitiven Datentypen (==, >=, ...)

- Können direkt verglichen werden
- Es wird direkt der im Speicher hinterlegte Wert verglichen

Vergleiche von Strings

- == vergleicht nur die im Speicher hinterlegten Werte beider Variablen
- Bei Objekten werden also nur die Referenzen verglichen
- Attributwerte werden ignoriert
- Benutzung der Funktion equals(String string)
- Für Klasse String ist equals bereits implementiert
- Später: Nutzung von equals für eigene Objekte

Wiederholung	Datentypen III	Strings	Konstruktoren	Methoden	Sauber Programmieren	Ende
0	00	00000 •0	00000	00000000000	000	0

Tutorium 14

Strings

Vergleiche

Strings vergleichen

Später: Nutzung von equals für eigene Objekte

Anwendung

Wiederholung



Sauber Programmieren

000

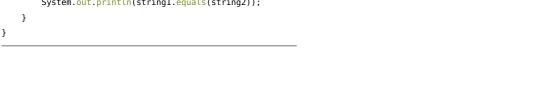
```
class StringComparator {
    public static void main(String[] args) {
       String string1 = "string";
       String string2 = new String("string");
       System.out.println(string1 == string2);
       System.out.println(string1.equals(string2));
```

Strings

00000

0

Datentypen III



Konstruktoren

00000



Tutorium 14 -Strings └─Vergleiche —Anwendung

class StringComparator (
public static wold main(String() args) (String stringt = "string"; String stringt = new String("string"); System.out.println(strings -- strings); tystem.out.println(string1.equals(string2)))

16.11.2022 Programmieren Tutorium Péter Bohner: Tutorium 14

Methoden

000000000000

Anwendung



```
class StringComparator {
   public static void main(String[] args) {
      String string1 = "string";
      String string2 = new String("string");
      System.out.println(string1 == string2);
      System.out.println(string1.equals(string2));
   }
}
```

```
$ java StringComparator
false
true
```



Tutorium 14

Strings
Vergleiche
Anwendung

Konstruktoren - Einführung



- Dienen der Initialisierung eines Objekts (Setzen von Anfangswerten der Attribute)
- legen damit den Startzustand eines Objekts fest
- Können Anfangswerte des Objekts als Parameter erhalten





Tutorium 14

 legen damit den Startzustand eines Obiekts fest Können Anfangswerte des Obiekts als Parameter erhalten

Dienen der Initialisierung eines Obiekts (Setzen von Anfangswerten der Attribute)

Konstruktoren - Einführung

- 1. parameter ist der aussagekräftiger Bezeichner des übergebenen Arguments
- 2. Parameter-List kann leer sein: keine Argumente werden übergeben

Wiederholung

Datentypen III

00000

Strings

•0000

Konstruktoren

Methoden 000000000000

Sauber Programmieren

000

Ende

Programmieren Tutorium

16.11.2022 Péter Bohner: Tutorium 14

Konstruktoren - Einführung



- Dienen der Initialisierung eines Objekts (Setzen von Anfangswerten der Attribute)
- legen damit den Startzustand eines Objekts fest
- Können Anfangswerte des Objekts als Parameter erhalten

Schema:

```
KlassenName(Parameter-Liste) {
    // Konstruktor-Rumpf
```

Parameter-Liste: Datentyp₁ parameter₁, Datentyp₂ parameter₂, ..., Datentyp_n parameter_n



Konstruktoren - Einführung

Tutorium 14

 Dienen der Initialisierung eines Obiekta (Setzen von Anfangswerten der Attribute) legen damit den Startzustand eines Obiekts fest Können Anfangswerte des Obiekts als Parameter erhalten

KlassenName(Parameter-Liste) {

Parameter-Liste: Datentyp₁ parameter₁, Datentyp₂ parameter₂, ..., Datentyp₆ parameter,

- 1. parameter ist der aussagekräftiger Bezeichner des übergebenen Arguments
- 2. Parameter-List kann leer sein: keine Argumente werden übergeben

Wiederholung

Datentypen III

Strings

00000

Konstruktoren •0000

Methoden 000000000000 Sauber Programmieren 000

Ende

Programmieren Tutorium

Konstruktoren - Einführung



- Dienen der Initialisierung eines Objekts (Setzen von Anfangswerten der Attribute)
- legen damit den Startzustand eines Objekts fest
- Können Anfangswerte des Objekts als Parameter erhalten

Schema:

```
KlassenName(Parameter-Liste) {
    // Konstruktor-Rumpf
}
Parameter-Liste: Datentyp1 parameter1, Datentyp2 parameter2,..., Datentypn parametern
```

Aufruf: new KlassenName(aktuelle Parameter);

١ |

Tutorium 14

Konstruktoren

└─Konstruktoren - Einführung

Dienen der Initialisierung eines Objekts (Setzen von Anfangswerten der Attribute)
 legen damit den Startzustand eines Objekts fest
 Können Anfangswerte des Objekts als Parameter erhalten

KlassenName(Parameter-Liste) {
// Konstruktor-Rumpf
}

Parameter-Liste: Datentyp; parameter; Datentyp; parameter; ..., Datentyp; parameter;
Aufnut: new Xlassenkame(aktuelle Parameter);

- 1. parameter ist der aussagekräftiger Bezeichner des übergebenen Arguments
- 2. Parameter-List kann leer sein: keine Argumente werden übergeben

4 16.11.2022 Péter Bohner: Tutorium 14

Programmieren Tutorium

Konstruktor - Beispiel



```
Bisher:
```

```
class Person {
    String name;
    int age;
Person person = new Person();
person.name = "Hans Peter";
person.age = 68;
```

Wiederholung

16.11.2022

Datentypen III

Péter Bohner: Tutorium 14

Strings

00000

00

Konstruktoren

00000

Methoden

000000000000

Sauber Programmieren

000

Ende

Programmieren Tutorium

Tutorium 14 └─ Konstruktoren

Konstruktor - Beispiel

Person person - new Person(); person.como = "Hans Peter"; person.ogo = 68;

Bisher: class Person (String name; int age;







Konstruktor - Beispiel

Datentypen III

Strings

Wiederholung



Sauber Programmieren

```
Bisher:
                                                    Mit Konstruktor:
                                                    class Person {
                                                         String name;
class Person {
                                                         int age;
   String name;
   int age;
                                                         Person(String name, int age) {
                                                             this.name = name;
                                                             this.age = age;
Person person = new Person();
person.name = "Hans Peter";
person.age = 68;
                                                    Person person = new Person("Hans Peter", 68);
```



Ende

Tutorium 14 └─ Konstruktoren Bisher: String name; int age; Person(String name, int age) { this.name = name; this.name = age; Konstruktor - Beispiel Person person - new Person() person.com = "Hans Peter"; person.oge = 60; Person person - new Person("Hans Peter", 68);

Methoden

Konstruktoren

Konstruktor - Beispiel

16.11.2022

Péter Bohner: Tutorium 14



Programmieren Tutorium

```
Bisher:
                                                          Mit Konstruktor:
                                                          class Person {
                                                              String name;
 class Person {
                                                              int age;
     String name;
     int age;
                                                              Person(String name, int age) {
                                                                  this.name = name:
                                                                  this.age = age;
 Person person = new Person();
 person.name = "Hans Peter";
 person.age = 68;
                                                          Person person = new Person("Hans Peter", 68);
                         this erlaubt Unterscheidung zwischen Parameter und Attribut
Wiederholung
                 Datentypen III
                                   Strings
                                                 Konstruktoren
                                                                   Methoden
                                                                                       Sauber Programmieren
                                                                                                              Ende
                                   00000
                                                 00000
                                                                   000000000000
                                                                                       000
```

Tutorium 14 Konstruktoren Konstruktor - Beispiel Konstruktor - Beispiel Konstruktor - Beispiel Konstruktor - Beispiel

Mehrere Konstruktoren



- festlegen mehrer Konstuktoren möglich
- müssen unterschiedliche Parameteranzahl und/oder unterschiedliche Parametertypen habe

Konstruktoren

00000

Methoden

000000000000

Sauber Programmieren Ende 000

Tutorium 14

└─ Konstruktoren Mehrere Konstruktoren







Wiederholung

Strings

00000

Datentypen III

Mehrere Konstruktoren



• festlegen mehrer Konstuktoren möglich

Péter Bohner: Tutorium 14

müssen unterschiedliche Parameteranzahl und/oder unterschiedliche Parametertypen habe

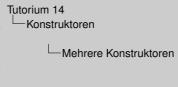
Ok

```
Person(String forname, String lastname) { .. }
Person(String name, int age) { .. }
```

```
Person(String name) { .. }
```

16.11.2022

Programmieren Tutorium





Mehrere Konstruktoren



- festlegen mehrer Konstuktoren möglich
- müssen unterschiedliche Parameteranzahl und/oder unterschiedliche Parametertypen habe

```
Ok
```

```
Person(String forname, String lastname) { .. }
Person(String name, int age) { .. }
Person(String name) { .. }
```

Nicht Ok

```
Person(String forname, int age) { .. }
```

Person(String lastname, int age) { .. }

Wiederholung Methoden Sauber Programmieren Ende Datentypen III Strings Konstruktoren 00000 00000 000000000000 000

Tutorium 14 Mehrere Konstruktoren

 festlegen mehrer Konstuktoren möglich müssen unterschiedliche Parameteranzahl und/oder unterschiedliche Parametertvoer habe Person(String formame, String lastname) { .. } Person(String name, int age) { .. } Person(String name) { .. } Person(String formane, int age) { .. } Person(String lastname, int age) { .. }

Mehrere Konstruktoren - Beispiel



```
Intuition:
class Person {
   String name;
   int age;
   Person(String name, int age) {
       this.name = name;
       this.age = age;
   Person(String name) {
       this.name = name;
       this.age = 0;
```

Tutorium 14

Konstruktoren

Mehrere Konstruktoren - Beispiel

Mediatric Maria Perus (
Maria

Wiederholung

Datentypen III

Mehrere Konstruktoren - Beispiel



Programmieren Tutorium

```
Intuition:
                                                           Besser:
class Person {
    String name;
                                                           class Person {
   int age;
                                                               String name;
                                                               int age;
   Person(String name, int age) {
       this.name = name:
                                                               Person (String name, int age) {
                                                                   this.name = name;
       this.age = age;
                                                                   this.age = age;
    Person(String name) {
       this.name = name;
                                                               Person (String name) {
       this.age = 0;
                                                                   this(name, 0);
```

Tutorium 14

Konstruktoren

Mehrere Konstruktoren - Beispiel



- ist kein Konstruktor angegeben, wird ein leerer Konstuktor automatisch ergänzt
- KlassenName() { }

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 0000 000000000000 000

Programmieren Tutorium

Tutorium 14 └─ Konstruktoren Default-Konstruktor

ist kein Konstruktor angegeben, wird ein leerer Konstuktor automatisch ergänzt



- ist kein Konstruktor angegeben, wird ein leerer Konstuktor automatisch ergänzt
- KlassenName() { }
- dieser initialisiert Attribute auf ihre Default-Werte



Tutorium 14 └─ Konstruktoren Default-Konstruktor

 ist kein Konstruktor angegeben, wird ein leerer Konstuktor automatisch ergänzt KlassenName() { } dieser initialisiert Attribute auf ihre Default-Werte

Programmieren Tutorium



- ist kein Konstruktor angegeben, wird ein leerer Konstuktor automatisch ergänzt
- KlassenName() { }
- dieser initialisiert Attribute auf ihre Default-Werte
- sobald ein Konstuktor angegeben wird, verschwindet der Default-Konstruktor



Tutorium 14

☐ Default-Konstruktor

 ist kein Konstruktor angegeben, wird ein leerer Konstuktor automatisch ergänzt KlassenName() { }

dieser initialisiert Attribute auf ihre Default-Werte

sobald ein Konstuktor angegeben wird, verschwindet der Default-Konstruktor

16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium



- ist kein Konstruktor angegeben, wird ein leerer Konstuktor automatisch ergänzt
- KlassenName() { }
- dieser initialisiert Attribute auf ihre Default-Werte
- sobald ein Konstuktor angegeben wird, verschwindet der Default-Konstruktor

Тур	Default-Werte		
boolean	false		
byte, short, int	0		
long	0L		
float	0.0f		
double	0.0		
char	'\u0000'		
Obiekt-Referenz	null		

Wiederholung	Datentypen III	Strings	Konstruktoren	Methoden	Sauber Programmieren	Ende
0	00	00000	0000	00000000000	000	0

16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium Tutorium 14

☐ Default-Konstruktor

 ist kein Konstruktor angegeben, wird ein leerer Konstuktor automatisch ergänzt KlassenName() { }

 dieser initialisiert Attribute auf ihre Default-Werte sobald ein Konstuktor angegeben wird, verschwindet der Default-Konstruktor



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



Methoden - Einführung

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

Tutorium 14 └─ Methoden

1. paramter ist der aussagekräftiger Bezeichner des übergebens Arguments

•0000000000

000

Sauber Programmieren

16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium



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

Schema:

```
Rückgabetyp methodenName(Parameterliste) {
   // Methodenrumpf
```

```
Wiederholung
                     Datentypen III
                                           Strings
                                                            Konstruktoren
                                                                                  Methoden
                                                                                                           Sauber Programmieren
                                           00000
                                                            00000
                                                                                  •0000000000
                                                                                                           000
```

Ende

└─ Methoden Methoden - Einführung

Tutorium 14

Methoden realisieren das dynamische Verhalten von Objekten und führen Berechnungen durch. Rückgabetyp methodenName(ParameterListe) {

1. paramter ist der aussagekräftiger Bezeichner des übergebens Arguments



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

```
Schema:
```

```
Rückgabetyp methodenName(Parameterliste) {
   // Methodenrumpf
```

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

Methoden - Einführung

Tutorium 14

Methoden realisieren das dynamische Verhalten von Objekten und führen Berechnungen durch. Rückgabetyp methodenName(ParameterListe) {

Der Rückgebetyp volld steht dabei für keine Rückgebel

1. paramter ist der aussagekräftiger Bezeichner des übergebens Arguments

Wiederholung

Datentypen III

00000

Strings

00000

Konstruktoren

Methoden •0000000000 Sauber Programmieren

000

Ende

Programmieren Tutorium

16.11.2022

Péter Bohner: Tutorium 14



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

```
Schema:
```

```
Rückgabetyp methodenName(Parameterliste) {
    // Methodenrumpf
.
```

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

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

0

Tutorium 14

9
Hermony

Methoden - Einführung

Promiserials Daintys, promoter, Daintys, promoter, Daintys, promoter, Daintys, promoter, Daintys, promoter,

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

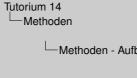
1. paramter ist der aussagekräftiger Bezeichner des übergebens Arguments

16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium



Der Methodenkopf besteht aus dem Rückgabetyp und der Methodensignatur.





Der Methodenkoof besteht aus dem Rückgabetvo und der Methodensignatur.

Methoden - Aufbau



Programmieren Tutorium

Der Methodenkopf besteht aus dem Rückgabetyp und der Methodensignatur.

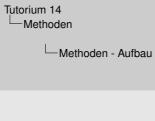
00

Péter Bohner: Tutorium 14

Die Methodensignatur besteht aus:

16.11.2022

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 00000000000 000



Der Methodenkoof besteht aus dem Rückgabetvo und der Methodensignatur. Die Methodensionatur besteht aus:



Der Methodenkopf besteht aus dem Rückgabetyp und der Methodensignatur.

Die Methodensignatur besteht aus:

dem Namen der Methode

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 00000000000 000 00

Tutorium 14 └─ Methoden Methoden - Aufbau

Der Methodenkoof besteht aus dem Rückpabetvo und der Methodensionatur Die Methodensionatur besteht aus:

dem Namen der Methode



Programmieren Tutorium

Der Methodenkopf besteht aus dem Rückgabetyp und der Methodensignatur.

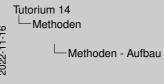
Die Methodensignatur besteht aus:

Péter Bohner: Tutorium 14

- dem Namen der Methode
- der Anzahl der Parameter

16.11.2022

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 00000000000 000 00



Der Methodenkoof besteht aus dem Rückpabetvo und der Methodensionatur Die Methodensionatur besteht aus:

dem Namen der Methode

der Anzahl der Parameter



Der Methodenkopf besteht aus dem Rückgabetyp und der Methodensignatur.

Die Methodensignatur besteht aus:

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

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 00000000000 000 00

└─ Methoden Methoden - Aufbau

Tutorium 14

Der Methodenkoof besteht aus dem Rückpabetvo und der Methodensionatur Die Methodensionatur besteht aus:

dem Namen der Methode

 der Anzahl der Parameter der Reihenfolge der Parameter

16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium



Der Methodenkopf besteht aus dem Rückgabetyp und der Methodensignatur.

- dem Namen der Methode



Tutorium 14 └─ Methoden

Methoden - Aufbau

Der Methodenkoof besteht aus dem Rückpabetvo und der Methodensionatur Die Methodensionatur besteht aus:

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

Die Methodensignatur besteht aus:

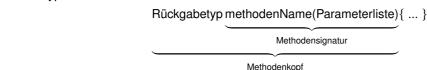
- der Anzahl der Parameter
- der Reihenfolge der Parameter
- der Typen der Parameter



Der Methodenkopf besteht aus dem Rückgabetyp und der Methodensignatur.

Die Methodensignatur besteht aus:

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



Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 00000000000 000 00

-Methoden

Tutorium 14

-Methoden - Aufbau

Die Methodensionatur besteht aus: dem Namen der Methode der Anzahl der Parameter

 der Reihenfolge der Parameter der Typen der Parameter

Rückgebetyp methodenName(Parameterliste) (...

Der Methodenkoof besteht aus dem Rückpabetvo und der Methodensionatur

Methoden - Zugriff auf Attribute



In Methoden kann auch auf Attribute des Objekts zugeriffen werden:



```
Tutorium 14
 └─ Methoden
                                                                                                                                           In Methoden kann auch auf Attribute des Obiekts zugeniffen werden:
            Methoden - Zugriff auf Attribute
```

Methoden - Zugriff auf Attribute



In Methoden kann auch auf Attribute des Objekts zugeriffen werden:

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

```
Wiederholung
                     Datentypen III
                                           Strings
                                                            Konstruktoren
                                                                                   Methoden
                                                                                                            Sauber Programmieren
                                                                                                                                        Ende
                                           00000
                                                            00000
                                                                                   00000000000
                                                                                                            000
```

Tutorium 14 └─ Methoden In Methoden kann auch auf Attribute des Obiekts zugeriffen werden: class Person (String name = "Swen"; void greet(String name) (System.out.println("Mallo " + name + ", ich heiße " + this.name + "."); Methoden - Zugriff auf Attribute

16.11.2022

Methoden - Zugriff auf Attribute



In Methoden kann auch auf Attribute des Objekts zugeriffen 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

Tutorium 14

Methoden

Methoden - Zugriff auf Attribute

Methoden - Zugriff auf Attribute

Starting and Attribute

Starting and Attribute

Starting and Attribute



Methoden - Aufruf

Tutorium 14 └─ Methoden

1. Bei Aufruf in selber Klasse nicht nötig: implizit this genutzt

Methoden können nur auf Objekten aufgerufen werden!

Datentypen III

Methoden können nur auf Objekten aufgerufen werden!

Ende Programmieren Tutorium

Wiederholung

16.11.2022

Strings 00000 Péter Bohner: Tutorium 14

Konstruktoren 00000

Methoden 00000000000

Sauber Programmieren 000

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

- Methoden Aufruf

Tutorium 14 └─ Methoden

 Methoden k\u00f6nnen nur auf Objekten aufgerufen werden! Dazu wird die Syntax obliekt .eethode() verwendet.

1. Bei Aufruf in selber Klasse nicht nötig: implizit this genutzt

- Sauber Programmieren Ende 000

Wiederholung

Datentypen III



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

```
Beispiel

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

Tutorium 14

Methoden

Methoden - Aufruf

Methoden können nur auf Cöpisten aufgerufen werden!
 Dizus wird de Syntax stjekt .nertode!) werwendst.

 Bodspiel
 Person person = new Person!);
 person.grees! ("Peter");

Bei Aufruf in selber Klasse nicht nötig: implizit this genutzt

22/34 16.11.2022 Péter Bohner: Tutorium 14

utorium 14 Programmieren Tutorium



- 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.
```

Tutorium 14

Methoden

Methoden - Aufruf

Mathodox Morris no and Cigisten adjusted session
 Box vide di Systex signi, avantuici) varendat.

| Dozsold
| Dozsold di Systex signi, avantuici) varendat.
| Dozsold
| Dozsold di Systex signi, avantuici) |
| principal ("Nota"); |
| princi

Bei Aufruf in selber Klasse nicht nötig: implizit this genutzt

22/34 16.11.2022 Péter Bohner: Tutorium 14

Programmieren Tutorium

Methoden - Rückgabe



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



└─ Methoden └─Methoden - Rückgabe

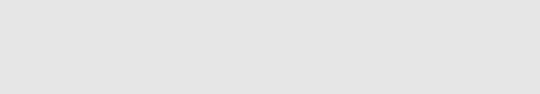
Tutorium 14

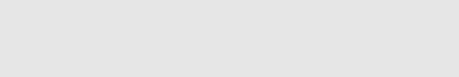


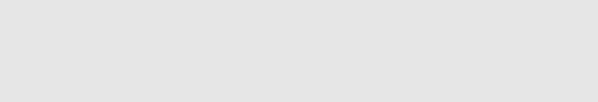
Wird ein anderer Rückgabetyp als vold gewählt, muss die Methode einen Wert vom Rückgabetyp mit dem

Schlüsselwort return zurückgeben.









00

000

Programmieren Tutorium

Methoden - Rückgabe



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

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

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 000000000000 000 00

Tutorium 14 └─ Methoden Wird ein anderer Rückgabetyp als volld gewählt, muss die Methode einen Wert vom Rückgabetyp mit dem Schlüsselwort return zurückgeben. int sum(int a, int b) {
 return a + b; └─Methoden - Rückgabe

Methoden - return bei void



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

Tutorium 14 └─ Methoden Methoden - return bei void

return kann aber auch bei Methoden mit Rückgabetyp volid benutzt werden: z.B. um eine Methode gezielt zu verlassen

1. Geschweifte Klammern beim if setzten. Nur für auf Folien oke

Wiederholung

Datentypen III

00000

Strings

00000

Konstruktoren

000000000000

000

Sauber Programmieren

Ende

Methoden

Methoden - return bei void



Programmieren Tutorium

return kann aber auch bei Methoden mit Rückgabetyp 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
    }
}
```

Tutorium 14

Methoden

Methoden - return bei void

Methoden - return bei void

James Strategies (State)

James Strategies

1. Geschweifte Klammern beim if setzten. Nur für auf Folien oke

1/34 16.11.2022 Péter Bohner: Tutorium 14

Klassenmethoden



Tutorium 14 └─ Methoden

─Klassenmethoden

Methoden die mit dem Modifier static beginnen nennt man Klassenmethoden.





Methoden die mit dem Modifier status beginnen nernt man Klassenmethoden.

2. Zugriff auf diese Analog zum Methode: Klasse.variable

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 000000000000 000

16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium

Klassenmethoden



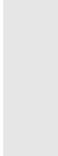
Methoden die mit dem Modifier static beginnen nennt man Klassenmethoden. Diese gehören zu der Klasse statt zu einem Objekt.

Strings

00000

Konstruktoren

00000



Ende

Methoden

Klassenmethoden

Tutorium 14

- 1. Mit static kann man auch Attribute als Klassenvariable definieren
- Zugriff auf diese Analog zum Methode: Klasse.variable

Methoden die mit dem Modifier statzic beginnen nernt man Klassenmethoden. Diese oehönen zu der Klasse statt zu einem Obiekt.

0

Wiederholung

Datentypen III

000000000000

Sauber Programmieren

000

Methoden

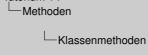


Methoden die mit dem Modifier static beginnen nennt man Klassenmethoden. Diese gehören zu der Klasse statt zu einem Objekt.

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



Tutorium 14





- 1. Mit static kann man auch Attribute als Klassenvariable definieren
- 2. Zugriff auf diese Analog zum Methode: Klasse.variable



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 Ausfühung: c = 13
```

:

Tutorium 14

Methoden

Klassenmethoden

Methoden die mit dem Modifier ktatic beginnen nernt man Klassenmethoden.
Diese gehömen zu der Klasse stat zu einem Chybit.
Class stem {
 static lat sunjist s , lest b) {
 restric lat sunjist s , lest b) {
 restric a t , lest b) {
 restrict a t , l

- 1. Mit static kann man auch Attribute als Klassenvariable definieren
- 2. Zugriff auf diese Analog zum Methode: Klasse.variable



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 Ausfühung: c = 13
Es wird kein Objekt benötigt, um die Methode auszuführen.
```



Tutorium 14

Methoden die mit dem Modifier status beginnen nernt man Klassenmethoden. Diese gehören zu der Klasse statt zu einem Obiekt. static int sum(int a, int b) { int c = Math.sum(5, 0): // Nach Auxfühung: c = 13 Es wird kein Obiekt benötiot, um die Methode auszuführen.

- 1. Mit static kann man auch Attribute als Klassenvariable definieren
- 2. Zugriff auf diese Analog zum Methode: Klasse.variable



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 Ausfühung: c = 13
Es wird kein Objekt benötigt, um die Methode auszuführen.
Schema: Klasse.methodenName();
```

Tutorium 14

Methoden

Klassenmethoden



- 1. Mit static kann man auch Attribute als Klassenvariable definieren
- 2. Zugriff auf diese Analog zum Methode: Klasse.variable

734 16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium



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 Ausfühung: c = 13
Es wird kein Objekt benötigt, um die Methode auszuführen.
Schema: Klasse.methodenName();
```

this kann in Klassenmethoden nicht genutzt werden

Wiederholung	Datentypen III	Strings	Konstruktoren	Methoden	Sauber Programmieren	Ende
0	00	00000	00000	00000000000	000	0

Tutorium 14

Methoden

Klassenmethoden

Methoden die mit dans Modifier stanls beginnen neuer mas Nassermethoden.
Dese gerbeit zu der Visious unt zu einem Objekt.

stanls die stanlste zu, der stanlste zu eine Objekt.

stanlste zu einestiet zu, des Si (

stanlste zu einestiet zu einestiet

- Mit static kann man auch Attribute als Klassenvariable definieren
- 2. Zugriff auf diese Analog zum Methode: Klasse.variable

25/

i/34 16.11.2022 Péter Bohner: Tutorium 14

Programmieren Tutorium

Hilfsmethoden



Was sind Hilfsmethoden?

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

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 000000000000 000 00

Programmieren Tutorium



└─ Methoden

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

Was sind Hilfsmethoden?



Hilfsmethoden

Tutorium 14

16.11.2022

Péter Bohner: Tutorium 14

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)

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 000000000000 000

16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium Tutorium 14 └─ Methoden Hilfsmethoden

Was sind Hilfsmethoden? Methoden, die häufig verwendete Funktionalität ausführen Können statisch oder nicht statisch sein

 hängt nicht von Attributen einer Klasse ab • static int sum(int a, int b)

Hilfsmethoden



Programmieren Tutorium

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)

Methoden Sauber Programmieren Ende Wiederholung Datentypen III Strings Konstruktoren 00000 00000 000000000000 000

Tutorium 14 ☐ Hilfsmethoden

Was sind Hilfsmethoden? Methoden, die häufig verwendete Funktionalität ausführen Können statisch oder nicht statisch sein

• static int sum(int a, int b)

gliedern innerhalb einer Klasse Funktionaltät.

hängt von Attributen ab

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 Rückgabetyp unterscheiden, sind nicht möglich.

Tutorium 14

Methoden

Überladen

As More natives Methods in the adjoint hims indicate, were in due to the Changes, Membridge united the Annual four Personal resistance of the Egyptical parameteristics. On the time (see a, see a) { ... } see that (see a, see a) { ... } see that (see a, see a) { ... } See the see and see all se

Methoden gleichen Namers, die sich nur im Rückgabetyp unterscheiden, sind nicht möglich



Aufgabe

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

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 000000000000

Tutorium 14 └─ Methoden

Methoden - Übung

am besten die Aufgabe live vormachen. Input der Tutanden in Eclipse umsetzen. Danach MusterlösungMethode muss nicht unbedingt static sein



Sauber Programmieren

Aufgabe

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

Tipp

Wiederholung

Mit dem Modulo Operator % erhaltet ihr den Rest der Ganzzahldivision.

Strings

Datentypen III

00 00000 00000 **000000000000000** 000 0

Konstruktoren

Ende

2-11-1

Tutorium 14

Methoden

└─Methoden - Übung

am besten die Aufgabe live vormachen. Input der Tutanden in Eclipse umsetzen. Danach MusterlösungMethode muss nicht unbedingt static sein

Methoden



Aufgabe

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

Tipp

- Mit dem Modulo Operator % erhaltet ihr den Rest der Ganzzahldivision.
- Der Rückgabetyp euer Methode muss boolean sein.

2-11-16

Tutorium 14

Methoden

└─Methoden - Übung

Authorized and Methode die eine Clanzzahl an nimmt und zurück gibt de diese gerade ist.

Tipp

M M dem Modulo Operator % erhaltet ihr den Rest der Ganzzahldriviston.

Der Rückgibehige voer Methode musis hoot aan sien.

am besten die Aufgabe live vormachen. Input der Tutanden in Eclipse umsetzen. Danach MusterlösungMethode muss nicht unbedingt static sein



Aufgabe

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

Tipp

- Mit dem Modulo Operator % erhaltet ihr den Rest der Ganzzahldivision.
- Der Rückgabetyp euer Methode muss boolean sein.

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

Methoden Sauber Programmieren Ende Wiederholung Datentypen III Strings Konstruktoren 00000 00000 000000000000

Tutorium 14 -Methoden Mt dem Modulo Operator % erhaltet ihr den Rest der Ganzzahldivision Methoden - Übung Der Rückgabetyo euer Methode muss boolean sein. static boolean isEven(int number) {
 return (number % 2 == 0);

am besten die Aufgabe live vormachen. Input der Tutanden in Eclipse umsetzen. Danach MusterlösungMethode muss nicht unbedingt static sein



beschreibt Zweck der Methode

00000

Methoden

Ende

Tutorium 14 └─ Methoden

Methoden - JavaDoc

16.11.2022

Wiederholung

Datentypen III

Péter Bohner: Tutorium 14

Strings 00000 00

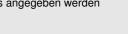
Konstruktoren

00000000000

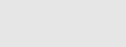
Sauber Programmieren 000

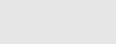
Programmieren Tutorium





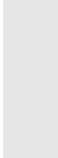
beschreibt Zweck der Methode







- beschreibt Zweck der Methode
- beschreibt in welchem Kontext die Methode verwendet werden kann



Tutorium 14 └─ Methoden beschreibt Zweck der Methode beschreibt in welchem Kontext die Methode verwendet werden kann Methoden - JavaDoc

1. Beispiel Spezialfälle: Wurzel ziehen, Methode verarbeitet keine negativen werte: muss angegeben werden

00

Wiederholung

urlsruher Institut für Technologie

Sauber Programmieren

000

beschreibt Zweck der Methode

Datentypen III

beschreibt in welchem Kontext die Methode verwendet werden kann

Strings

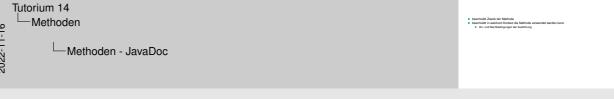
00000

00

Vor- und Nachbedingungen der Ausführung



Ende



1. Beispiel Spezialfälle: Wurzel ziehen, Methode verarbeitet keine negativen werte: muss angegeben werden

Methoden

00000000000

Konstruktoren

Wiederholung



Sauber Programmieren

000

beschreibt Zweck der Methode

Datentypen III

beschreibt in welchem Kontext die Methode verwendet werden kann

Strings

00000

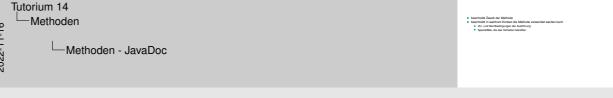
00

- Vor- und Nachbedingungen der Ausführung
- Spezialfälle, die das Verhalten betreffen





Ende



1. Beispiel Spezialfälle: Wurzel ziehen, Methode verarbeitet keine negativen werte: muss angegeben werden

Methoden

00000000000

Konstruktoren

Wiederholung



- beschreibt Zweck der Methode
- beschreibt in welchem Kontext die Methode verwendet werden kann

Strings

00

00000

- Vor- und Nachbedingungen der Ausführung
- Spezialfälle, die das Verhalten betreffen

Datentypen III

alle Parameter werden beschrieben (Zweck und Wertebereich)



Ende



Sauber Programmieren

000



beschreibt in welchen Korteol die Methode verwendet werden kann
 Vor- und Nachbedingungen der Austähnung
 Spezialfälle, die das Verhalten beterften
 alle Parameter weden beschrieben (Zweck und Werlebereich)

beschreibt Zweck der Methode

1. Beispiel Spezialfälle: Wurzel ziehen, Methode verarbeitet keine negativen werte: muss angegeben werden

Konstruktoren

00000

Methoden



- 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 Rückgabetyp wird beschrieben (Inhalt)



Tutorium 14

Methoden

* translati and for March

* translation

1. Beispiel Spezialfälle: Wurzel ziehen, Methode verarbeitet keine negativen werte: muss angegeben werden



- beschreibt Zweck der Methode
- beschreibt in welchem Kontext die Methode verwendet werden kann

00000

- Vor- und Nachbedingungen der Ausführung
- Spezialfälle, die das Verhalten betreffen
- alle Parameter werden beschrieben (Zweck und Wertebereich)
- der Rückgabetyp 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) {
Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende
```

00000

Tutorium 14

Methoden

- Methoden - JavaDoc

M

1. Beispiel Spezialfälle: Wurzel ziehen, Methode verarbeitet keine negativen werte: muss angegeben werden

34 16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium

16.11.2022

Péter Bohner: Tutorium 14



Programmieren Tutorium

```
* 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;
Wiederholung
               Datentypen III
                                Strings
                                            Konstruktoren
                                                            Methoden
                                                                               Sauber Programmieren
                                                                                                    Ende
                                00000
                                            00000
                                                            00000000000
                                                                               000
```

```
Tutorium 14

Methoden

- Attion the abstract value of an out value.
- If the argument is a designate, the required to returned.
- If the argument is the argument to returned.
- If the argument is a designate, the required to returned.
- If the argument is a designate in the argument is returned.
- If the argument is a designate in the argument is a designate in the argument is a designate in the argument is a designate of the argument is a designate or state. (If note one value, which is expertant.)
- If the argument is a designate or state is the argument is a designate or state. (If note one value, which is expertant.)
- If the argument is a designate or state is the argument is a designate or state. (If note one value, which is expertant.)
- If the argument is a designate or state is the argument is a designate or state. (If note one value, which is expertant.)
- If the argument is a designate or state is a designate or state. (If note one value, which is expertant.)
- If the argument is a designate or state is a designate or state. (If note one value, which is expertant.)
- If the argument is a designate or state. (If note one value, which is expertant.)
- If the argument is a designate or state is a designate or state. (If note one value, which is expertant.)
- If the argument is a designate or state. (If note one value, which is expertant.)
- If the argument is a designate or state. (If note one value, which is expertant.)
- If the argument is a designate or is a designate or is a designate or in the argument.
- If the argument is a designate or is a designate or in the argument is a designate or in the argument.
- If the argument is a designate or is a designate or in the argument is a designate or in th
```



```
Negativ Beispiel
```

```
public class DemoBad{public static void main(String[]
args){System.out.println("Hello world!");System.
out.println("auto-generated text");}}
```

Wiederholung Datentypen III Strings Konstruktoren Methoden Sauber Programmieren Ende 00000 00000 000000000000 •00

Tutorium 14 -Sauber Programmieren Sauber Programmieren

Negativ Beispiel

public class Temolod(public static wold main(String[]

args){System.out.println("Hello world");System.



Sauber Programmieren

Ende

```
Tutorium 14
Sauber Programmieren
Sauber Programmieren
```

Negativ Beispiel

Wiederholung

```
public class DemoBad{public static void main(String[]
args){System.out.println("Hello world!");System.
out.println("auto-generated text");}}
```

Warum sauber programmieren?

Datentypen III

- erhöht Softwarequalität, Lesbarkeit und Wiederverwendbarkeit
- Kalanda Falidha a dawa da Oan ay ay la ad i

Strings

Kosten der Entwicklung und Wartung der Software werden reduziert

00 0000 00000 00000 000000000 00 0

Methoden

Konstruktoren



Aber wie?

- Namen und Bezeichner sind aussagekräftig und in Englisch (Bennenungskonventionen beachten)
- Kommentare existieren und sind sinnvoll (JavaDoc)
- Einheitliche Sprache in Kommentaren (nur Englisch oder Deutsch)
- Quelltext enthält sinnvolle Einrückungen und Leerzeichen



Tutorium 14
Sauber Programmieren
Sauber Programmieren

Abor wie?

• Namen und Bezeichner sind aussagskräftig und in Englisch (Bernenungskonventionen beachten)

• Kommentare existieren und sind simvoll (Java/Doc)

• Einheitliche Sprache in Kommentaren (nur Englisch oder Deutsch)

Sauber Programmieren

4 16.11.2022 Péter Bohner: Tutorium 14 Programmieren Tutorium



Aber wie?

- Namen und Bezeichner sind aussagekräftig und in Englisch (Bennenungskonventionen beachten)
- Kommentare existieren und sind sinnvoll (JavaDoc)
- Einheitliche Sprache in Kommentaren (nur Englisch oder Deutsch)
- Quelltext enthält sinnvolle Einrückungen und Leerzeichen

Unterstützung

- Checkstyle!
- Automatische Überprüfung, abgaberelevant!
- Auch nach Programmieren sinnvoll, vor allem in Team Projekten

Datentypen III Methoden Sauber Programmieren Ende Wiederholung Strings Konstruktoren 00000 00000 000000000000 000

Tutorium 14 Sauber Programmieren Sauber Programmieren

 Namen und Bezeichner sind aussagekräftig und in Englisch (Bernenungskonventionen beachte Einheitliche Sprache in Kommentaren (nur Englisch oder Deutsch) Quelitext enthält sinnvolle Einrückungen und Leerzeichen Automatische Überprüfung, abgeberelevent! Auch nach Programmieren sinnvoll, vor allem in Team Projekte

Sauber Programmieren - Beispiel



```
/**
 * The class DemoNice looks way better!
 * @author Gregor Lucka
 * @see DemoBad
public class DemoNice {
    /**
     * This main-method greets the world.
     * @params args This methods does not need any arguments.
   public static void main(String[] args) {
        System.out.println("Hello World!");
```



Tutorium 14 Sauber Programmieren Sauber Programmieren - Beispiel

public class benobice (This main-method greets the world.
 Aparams args This methods does not need any arguments. public static wold main(String() args) (

Ende

