JAVA I

Grundlagen der Programmierung

Termin 1 von 5:

Imperative Sprachelemente

Letzte Änderung: 07.03.2017

Ihr Dozent: Steffen Kemena

- 2007 + 2008 Hiwi-Dozent „Java“ FH-Kiel (Wirtschaftsinformatik)
- 2008 – 2011 Webentwickler bei SealMedia in Kiel
- 2012 – 2014 Java-Entwickler in Hamburg
- Seit 2015 Zurück in Kiel, Softwareentwickler bei NetUSE
- Weitere Infos unter http://www.leycarno.com

Die Kursteilnehmer

- **Wer sind Sie? Wer bist du?**
- **Wissensstand: Java? Programmiersprachen? Komplett planlos?**
- **Welche Erwartungen gibt es an den Dozenten und diesen Kurs?**

Inhalte des Kurses - Kurzübersicht

Termin 1:	**Imperative Sprachkonzepte**
Termin 2:	**Grundlagen der Objektorientierung**
Termin 3:	**Arrays, Collections und Aufzählungen**
Termin 4:	**Das Design-Pattern „MVC“**
Termin 5:	**Persistenz mit Dateien und JSON**

Termin 1 - Imperative Sprachkonzepte

1. **Was ist Java?**
2. **Bezeichner, Variablen und Datentypen**
3. **Ausdrücke und Operatoren**
4. **Kontrollstrukturen**
5. **Kommentare und Eingaben mit der Tastatur**

1. Was ist Java

- Erste JavaVersion **1995** von Sun Microsystems (Seit 2010 Oracle)

- Die JVM (Java Virtual Maschine) ermöglicht **Plattformunabhängigkeit**

- Eine komplett **Objektorientierte** Programmiersprache

- „***Einfach***“ im Vergleich zu **C++** oder **C#** durch reduzierten Umfang

- **Typsicher** (im Gegensatz zu z.B. *PHP*) und **rubust** (geringe Fehleranfälligkeit)

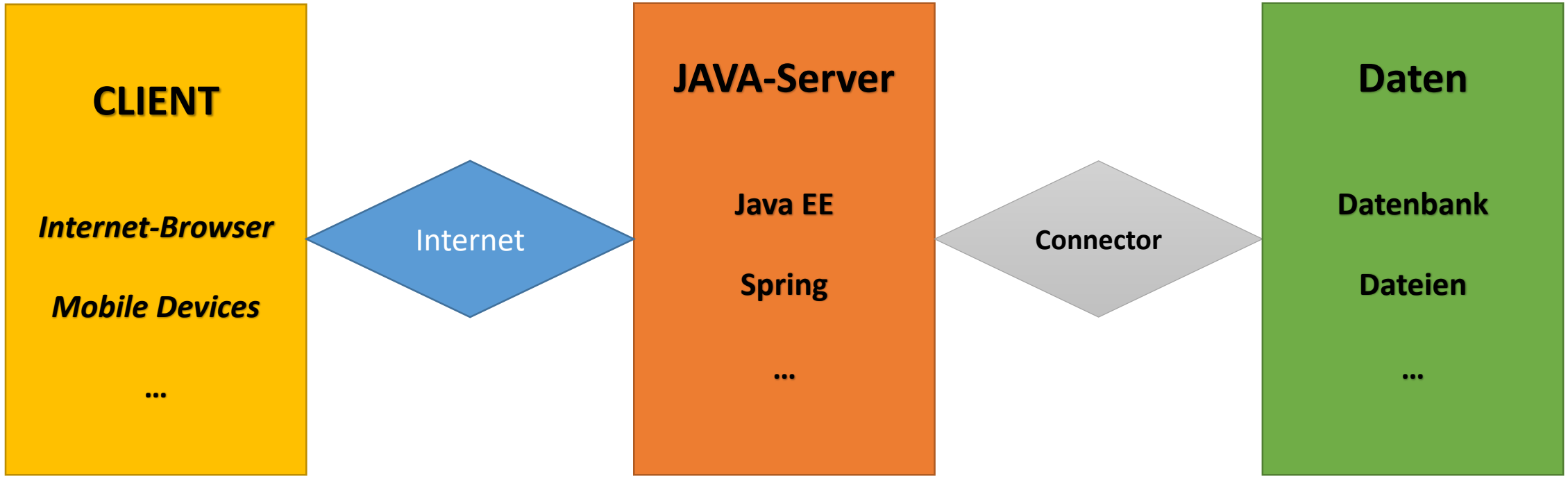
1. Was ist Java

Vergleich/Bezug zu anderen Sprachen

- **JavaScript** hieß früher ***LiveScipt*** und hat NICHTS mit JAVA zu tun
- **Smalltalk** (eine der ältesten Sprachen) ist ein konzeptuelles Vorbild
- **C++** ist sehr ähnlich, aber komplizierter und Fehleranfälliger
- **C#** jüngere Sprache, die (u.a.) von JAVA „gelernt“ hat
  dadurch durchdachter, aber noch nicht so weit verbreitet
- **PHP** Für Webanwendungen einfacher und gebräuchlicher
  Jedoch Fehleranfälliger und langsamer *(mit PHP7 etwas besser)*

1. Was ist Java

Webanwendungen (SERVER-Anwendungen im speziellen)



1. Was ist Java

RESTfull (*REpresentational State Transfer)* **Services**

mit JSON (JavaScript Object Notation)

1. Was ist Java

Mobile Devices & More

1. Was ist Java

Desktopanwendungen

2. Bezeichner, Variablen und Datentypen

2.1. Bezeichner

- **Was sind Bezeichner?**

  Kurz*:* *Wann immer der Programmierer selbst Namen vergibt, handelt es sich dabei um einen Bezeichner…*

- **Die Regeln für Bezeichner:**

  - Beliebig viele ***Unicode*** *(engl. Zeichensatz)* Zeichen und Zahlen
  - Erstes Zeichen = ***Buchstabe***, ***Unterstrich*** oder ***Dollarzeichen***
  - ***Keine Schlüsselwörter*** (im Kurs durchgängig blau dargestellt)
  - ***CaseSensitive*** = `anumber` ist etwas anderes als `aNumber` !

2. Bezeichner, Variablen und Datentypen

2.2. Bezeichner - **Konventionen**

- Es ist anzuraten, ausschließlich **englische** Bezeichner zu nutzen!
- Zusammengesetzte Wörter im „**CamelCase**“ – Format
- ***Variablen*** (und ***Methoden***) beginnen mit einem **Kleinbuchstaben**

„Gute Bezeichner erzählen was sie sind und was sie tun!“

2. Bezeichner, Variablen und Datentypen

2.3. Variablen zuweisen/definieren

- Definition:
```
int aNumber;
```
- Initialisierung:
```
aNumber = 10;
```

- Explizite Initialisierung:
```
int anotherNumber = 20;
```

- Mehrere Variablen eines selben Datentyps gleichzeitig definieren:
```
int age, size, weight;
```

- Und auch explizit:
```
int height = 10, width = 20;
```

2. Bezeichner, Variablen und Datentypen

2.4. Die einfachen („primitiven“) Datentypen

- Wahrheitswerte

```
boolean isReady = true;
```

- Ganzzahlen

```
int pos = 123, neg = -987;
```

- Fließkommazahlen *(englischer Punkt statt deutschem Komma!)*

```
float number = 12.34f;
```

- Zeichen *(nur genau EIN Zeichen in „einfachen“ Anführungszeichen)*

```
char letter = 'A';
```

2. Bezeichner, Variablen und Datentypen

2.4. Die 8 einfachen („primitiven“) Datentypen - Übersicht

Bezeichnung	Datentyp/Schlüsselwort	Wertebereich
Wahrheitswerte	`boolean`	`false, true`
Zeichen	`char`	`a…z, A…Z, 0…9, !"§$%&/()...` Alle möglichen Zeichen
Ganze Zahlen	`byte`	`-128 … 127`
	`short`	`-32.768 … 32.767`
	`int`	`-2.147.483.648 … 2.147.483.647`
	`long`	`-9.223.372.036.854.775.808 … 9.223.372.036.854.775.807`
Fließkommazahlen/ Gleitkommazahlen	`float`	`~1,4*10^-45 … 3,4*10^38` *Genauigkeit ca. 7 Stellen*
	`double`	`~4,9*10^-324 … 1,8*10^308` *Genauigkeit ca. 15 Stellen*

2. Bezeichner, Variablen und Datentypen

2.5. String und Ausgabe *(„Auf den Schirm!“)*

String ist eine **Standard-Klasse**. Es ist KEIN „einfacher/primitiver“ Datentyp. Trotzdem wird sie so oft genutzt, dass Sie hier erwähnt wird.

```
String string1 = „Hello“;
String string2 = „World“;
String message = string1 + „ “  + string2;
System.out.println(message);  // Ausgabe!
```

WICHTIG: Vergleiche erfolgen nicht mit == sondern mit einer Methode Namens „equals“:

```
if (string1.equals(string2)) ...
```

3. Ausdrücke und Operatoren

3.1. Ausdrücke

- Ein **Ausdruck** ist eine Kombination von **Operatoren** und **Operanden**

- Das **Ergebnis** des Ausdrucks wird „nach links geschrieben“ (***zugewiesen***)

```
int aNumber      =    2          +          8;
Ergebnis         =  Operand1     Operator   Operand2
```

- Jeder Operator hat eine **Priorität**, wodurch die Reihenfolge mit der sie innerhalb eines Ausdrucks abgearbeitet werden, festgelegt ist.

3. Ausdrücke und Operatoren

3.2. Arithmetische Operatoren

Bezeichnung	Operator		Priorität
Positives Vorzeichen	`+`	10 (das Vorzeichen wird weggelassen)	**1**
Negatives Vorzeichen	`-`	-10	
Inkrementieren	`++`	`a++;` äquivalent zu: `a = a + 1;`	
Dekrementieren	`--`	`a--;` äquivalent zu: `a = a - 1;`	
Multiplikation	`*`	`int multiplied = a * b;`	**2**
Division	`/`	`float divided = a / b;`	
Rest	`%`	`int modulo = a % b;`	
Addition	`+`	`int addition = a + b;`	**3**
Subtraktion	`-`	`int subtraction = a - b;`	

3. Ausdrücke und Operatoren

3.3. Relationale Operatoren

Bezeichnung	Operator		Priorität
Kleiner	`<`	`boolean isSmaller = a < b;`	5
Kleiner oder gleich	`<=`	`boolean isSmallerOrEuqal = a <= b;`	
Größer	`>`	`boolean isGreater = a > b;`	
Größer oder gleich	`>=`	`boolean isGreaterOrEqual = a >= b;`	
Gleich	`==`	`boolean isEqual = a == b;`	6
Ungleich	`!=`	`boolean isUnequal = a != b;`	

3. Ausdrücke und Operatoren

3.4. Logische Operatoren

Bezeichnung	Operator		Priorität
NICHT	`!`	`boolean isTrue = !isFalse;`	1
UND („vollständige“ Ausw.)	`&`	`boolean isTrue = justTrue & trueToo;`	7
ODER („vollständige“ Ausw.)	`|`	`boolean isTrue = justTrue | justFalse;`	9
UND („kurze“ Auswertung)	`&&`	`boolean isTrue = justTrue && trueTrue;`	10
ODER („kurze“ Auswertung)	`||`	`boolean isTrue = justTrue || justFalse;`	11
exklusive ODER (XOR)	`^`	`boolean isTrue = justTrue ^ justFalse;` `boolean isFalse = justTrue ^ trueTrue;`	8

Der Unterschied von `&` zu `&&` wird später erläutert

3. Ausdrücke und Operatoren

3.5. Bitoperatoren

Beispiele: **8 << 2 = 32** **8 >> 2 = 2**

0000 1000 (also 8)	**<<**	**2**	**=**	**0010 0000 (also 32)**
0000 1000 (also 8)	**>>**	**2**	**=**	**0000 0010 (also 2)**

Was ist das binäre Zahlensystem?

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
2^7	**2^6**	**1^5**	**2^4**	**2^3**	**2^2**	**2^1**	**2^1**
128	**64**	**32**	**16**	**8**	**4**	**2**	**1**

Beispiele:	**Die Zahl**	**9**	**(= 8 + 1)**	**schreibt man binär**	**0000 1001**
	Die Zahl	**123**	**(= 64 + 32 + 16 + 8 + 2 + 1)**	**schreibt man binär**	**0111 1011**

…eine Erwähnung der Vollständigkeit halber - Thema kommt nicht weiter vor…

3. Ausdrücke und Operatoren

3.6. Zuweisungsoperatoren

Bezeichnung	**Operator**		**Priorität**
Einfache Zuweisung	**=**	Das Ergebnis des Ausdrucks (rechts vom Zuweisungsoperator) wird einer Variablen (links vom Zuweisungsoperator) zugeordnet	**13 (immer die letzte Aktion)**

Kombinierte Zuweisung	*op*=	`int aNumber = aNumber + anotherNumber;`
		Vom Ergebnis her dasselbe wie:
		`int aNumber += anotherNumber;`
		
		Mögliche bei: **`* / % + -`** **und Bitoperatoren…**	

3. Ausdrücke und Operatoren

3.7. Cast-Operator (Priorität = 1)

```
int x = 13;
int y = 5;
long bigResult = x * y;
int result = bigResult; // FEHLER!
int result = (int) bigResult; // OK
```

Problemlos, wenn Ergebnistyp „größer“

Absicht bei „kleineren“ Ergebnissen angeben

```
float f = 10.123
int i = f;          // FEHLER !!!
int i = (int) f;    // OK
```

Umwandlung der Fließkommazahl **10.123** in die Ganzzahl **10** erfolgt NICHT automatisch.

4. Kontrollstrukturen

4.1. Das Ende eines Ausdrucks ;

- Mit einem **Semikolon** wird das Ende eines Ausdrucks definiert

4.2. Block {…}

- Die Grenzen von Klassen, Methoden und anderen Kontrollstrukturen
- Zusammenfassung mehrerer aufeinander folgenden Ausdrücke.
- Alle Variablen innerhalb eines Blocks, sind nur dort gültig.

4. Kontrollstrukturen

4.3. Verzweigungen mit Bedingung (1/2)

Einfachste Form der Bedingung – der Block wird nur ausgeführt, wenn Bedingung „wahr“:

```
if (a < b) {
        System.out.println(„a is smaller than b“);
}
```

Alternative – was soll ich „sonst“, wenn der erste Ausdruck „falsch“ ist:

```
if (a > b) {
        System.out.println(„a is bigger than b“);
} else {
        System.out.println(„a is NOT bigger than b“);
}
```

4. Kontrollstrukturen

4.3. Verzweigungen mit Bedingung (2/2)

```
if (a < b) {
    System.out.println(„a is smaller“);
} else if (a > b) {
    System.out.println(„a is bigger“);
} else {
    System.out.println(„a & b are equal“);
}
```

4. Kontrollstrukturen

4.4. Verzweigung mit dem Bedingungsoperator (Priorität = 12)

Bedingung ? Ausdruck1 : Ausdruck2

Ist die **Bedingung wahr**, wird **Ausdruck1** ausgewertet.

Ansonsten wird **Ausdurck2** ausgewertet.

Verschachteln vermeiden!!!

```
b3 ? b2 ? b1 ? a1 : a2 : a3 : a4;          // ??????
```

4. Kontrollstrukturen

4.6. Verzweigungen abhängig von immer derselben Variable

Die Ausdrücke links und rechts sind äquivalent:

```
if (aNumber == 1) {
        aNumber += 2;
} else if (aNumber == 4) {
         aNumber--;
} else {
        aNumber = 3;
}
```

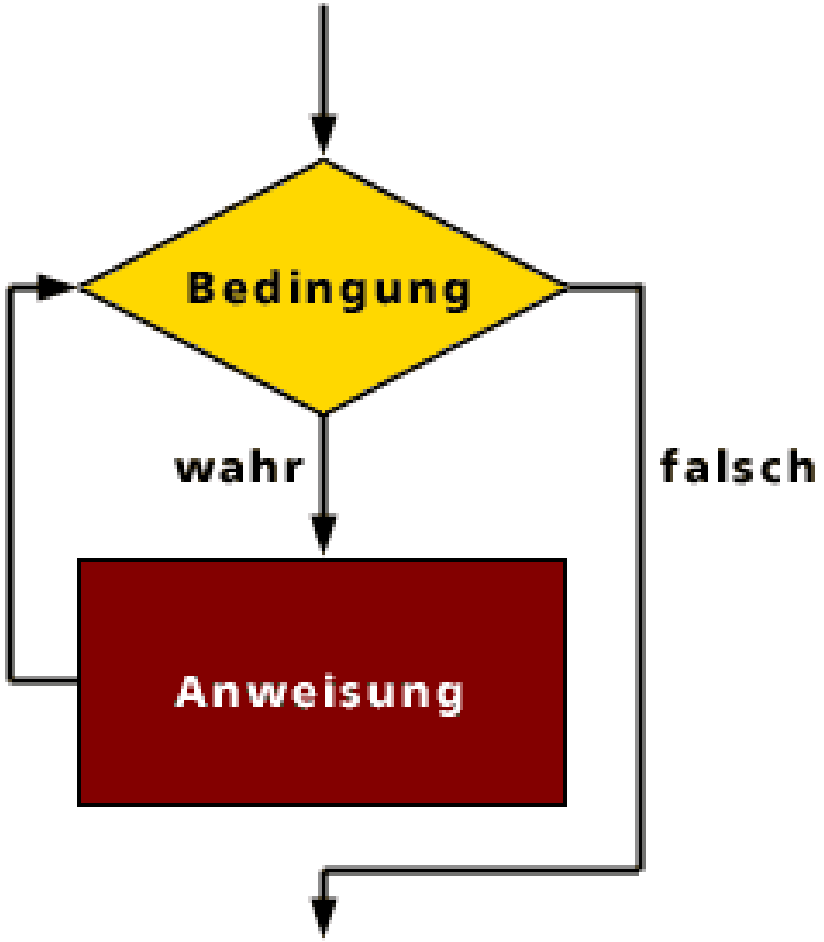
```
switch (aNumber) {
        case 1:
                aNumber += 2;
                break;
        case 4:
                aNumber--;
                break;
        default:
                aNumber = 3;
                break;
}
```

4. Kontrollstrukturen

4.7. Schleifen (1/4) „while“

```
int i = 1;
while (i < 10) {
      System.out.println(i++);
}
```

```
while (Bedingung) {
      Anweisung
}
```

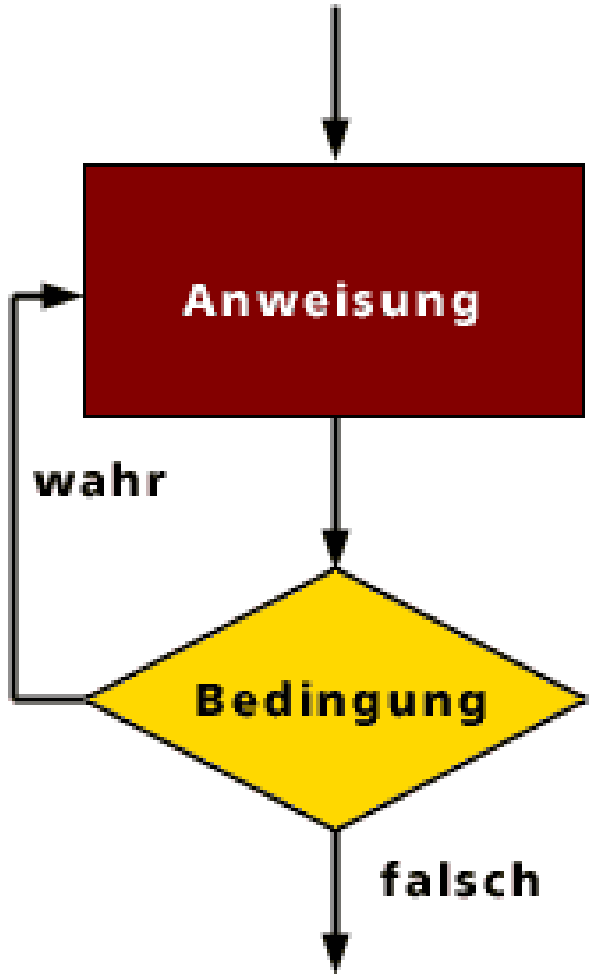


4. Kontrollstrukturen

4.7. Schleifen (2/4) „do“

```
int i = 1;
do {
      System.out.println(i++);
} while (i < 10);
```

```
do {
     Anweisung
} while (Bedingung);
```

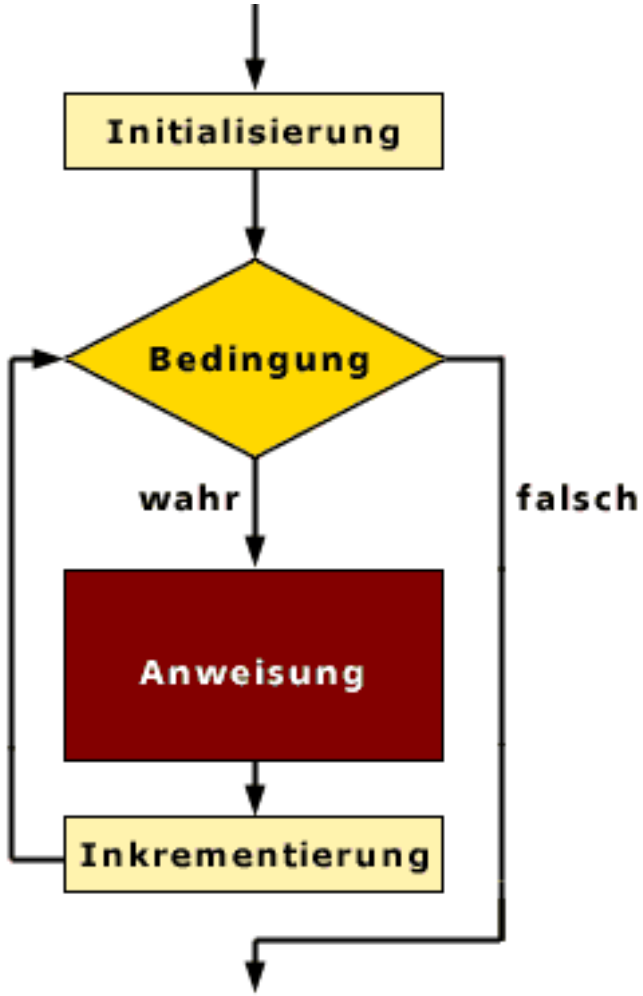


4. Kontrollstrukturen

4.7. Schleifen (3/4) „for“

```
for (int i = 1; i < 10; i++) {
      System.out.println(i);
}
```

```
for (Init; Bedingung; Update){
      Anweisung
}
```



4. Kontrollstrukturen

4.7. Schleifen (4/4) *Sprunganweisungen*

```
boolean justOddNumbers;
int externalMaxValue;
// [...]
for (int i = 1; i < 10; i++) {
      if (justOddNumbers && i % 2 != 0) {
            continue;          // naechster Schleifendurchlauf
      }
      if (i > externalMaxValue) {
            break;             // Schleife komplett abbrechen!
      }
      System.out.println(i);
}
```

Der Unterschied von & und &&

```
boolean x = trueOrFalse;  // „trueOrFalse“ ist wahr ODER falsch...
boolean isUnsafe = x    &&    a > b++;
boolean isSafe   = x    &     a > b++;
```

`&&` = Ausdruck wird **beim ersten false abgebrochen**. `b` wird NICHT inkrementiert!

`&` = Ausdruck wird **vollständig ausgewertet** und `b` ist danach um `1` höher!

```
int x = 0;
while (x < 3) {
        if (false & x++) {
                System.out.println(x);
        }
}
```

```
int x = 0;
while (x < 3) {
        if (false && x++) {
                System.out.println(x);
        }
}
```

Die linke Schleife wird beendet, die rechte läuft endlos weiter…

5. Kommentare und Eingaben mit der Tastatur

5.1. Einzeilige Kommentare:

```
int aNumber = 10; // „aNumber“ wird 10 zugewiesen
```

5.2. Mehrzeilige Kommentare

```
int aNumber = 10; /* Dies ist ein Kommentar über
                     mehrere Zeilen: Der Variablen
                     „aNumber“ wird   das Literal 10
                     zugewiesen... */
```

5. Kommentare und Eingaben mit der Tastatur

5.3. Dokumentationskommentare

```
/**
 * Dies ist eine „Methode“, oder auch „Funktion“.
 * Man könnte sagen ein „Unterprogramm“, das mehrmals im
 * Hauptprogramm aufgerufen werden kann.
 * Mehr über dieses Thema nächste Woche ;-)
 *
 * @var         int          ein Parameter
 * @return      int          der Rückgabe-Wert
 */
int square(int aNumber)     {
    return aNumber * aNumber;
}
```

5. Kommentare und Eingaben mit der Tastatur

5.4. Eingaben

Mit folgendem Aufruf ist es möglich, Eingaben entgegenzunehmen:

```
Scanner scanner = new Scanner(System.in);

String name  = scanner.nextLine();
int input     = scanner.nextInt();
```

Der Aufruf von „**`nextLine()`**“ ist auch notwendig, um *unverarbeitete Eingaben* von nextInt() zu „vergessen“

1. **Schreiben Sie das „Hello World“ Programm so um, dass es nach einen Namen fragt, den der Anwender eingeben darf. (Beispiel: „Hans“) Somit soll das Programm „Hello Hans“ ausgeben.**

2. **Ändern Sie das „Hello World“ Programm noch einmal so ab, dass es solange Namen entgegen nimmt und ausgibt, bis statt einem Namen das Wort „END“ eingegeben wird.**

3. **Schreiben Sie einen kleinen Taschenrechner, der die einfachen Rechenoperationen Addition, Subtraktion, Multiplikation und Division mit zwei Zahlen durchführen kann. Der Anwender soll nacheinander zwei Zahlen und zuletzt das Operationssymbol eingeben. Abschließend erfolgt das Ergebnis oder eine Fehlermeldung.**

4. **Schreibe ein Programm zur Bestimmung des *kleinsten gemeinsamen Vielfaches (kgV)* zweier natürlicher Zahlen. Die Berechnung soll ohne Beteiligung des größten gemeinsamen Teilers erfolgen.** *Beispiel*: Zahl1 = 5, Zahl2 = 7, kgV = 35

5. Es sollen x Flaschen in Kartons verpackt werden. Ein Karton kann n Flaschen aufnehmen. Schreiben Sie ein Programm, das ermittelt, in wie viele Kartons eine bestimmte Anzahl Flaschen verpackt werden kann und wie viele Flaschen übrig sind.

6. Zu vorgegebenen Zahlen x und y, soll festgelegt werden, ob x durch y teilbar ist.

7. Jetzt ist es x Uhr. Wieviel Uhr ist es in n Stunden?

8. Schreiben Sie ein Programm, das die Anzahl von Sekunden im Monat Januar berechnet. Benutzen Sie hierfür aber nicht einfach Literale, sondern mindestens 4 eigene Konstanten.

9. Schreiben Sie ein Programm, das eine Tabelle mit dem kleinen Einmaleins (also 1*1 bis 10*10) angeordnet in je zehn Zeilen, rechtsbündig ausgibt. „Doppelte“erlaubt (Zahlen vertauscht).

10. Schreiben Sie ein Programm, das zu einer Zahl n <= 20 die Fakultät n! ermittelt.

Es gilt: $n! = 1 * 2 * \ldots * (n-1) * n$ und $0! = 1$

Kontakt, Hilfe und GitHub

Unter der folgenden Adresse finden Sie zu den jeweiligen Terminen die PDFs (also auch das aktuelle Dokument in digitaler Form), sowie Code-Beispiele und Musterlösungen:

https://github.com/Leycarno/javacourse

Natürlich können Sie mir bei Fragen jederzeit eine Email schicken:

steffen.kemena@leycarno.com