



Java Einführung

ABLAUFSTEUERUNG

Kapitel 3 und 4

Inhalt dieser Einheit



Merkmale und **Syntax** der
verschiedenen Kontrollstrukturen:

- `if else`
- `switch`
- `while`
- `do while`
- `for`
- `break, continue`

EXKURS: Rekursion

Kontrollstrukturen

- Alle Kontrollstrukturen eines Programms basieren auf Zustandsüberprüfungen, die die Werte `true` oder `false` liefern.

z.B. PIN-Überprüfung des ATM:

Wenn der richtige PIN eingegeben wurde, dann wird die Abhebung erlaubt, andernfalls verweigert.

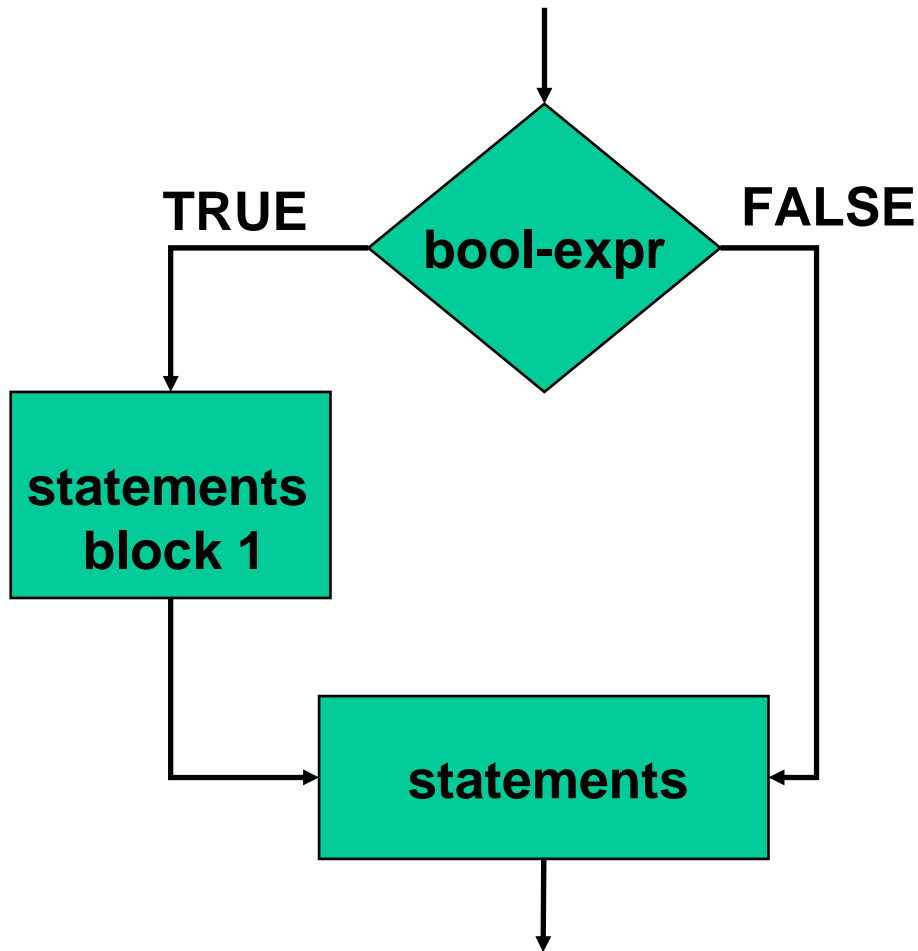
Kontrollstrukturen

Verschiedene Arten von Kontrollstrukturen

- Verzweigungen: `if else, switch`
- Iterationen: `while, do while, for`
- Sprunganweisungen: `continue, break`
- Exception Handling*:
`(try | catch | finally) throw`

*wird später behandelt!

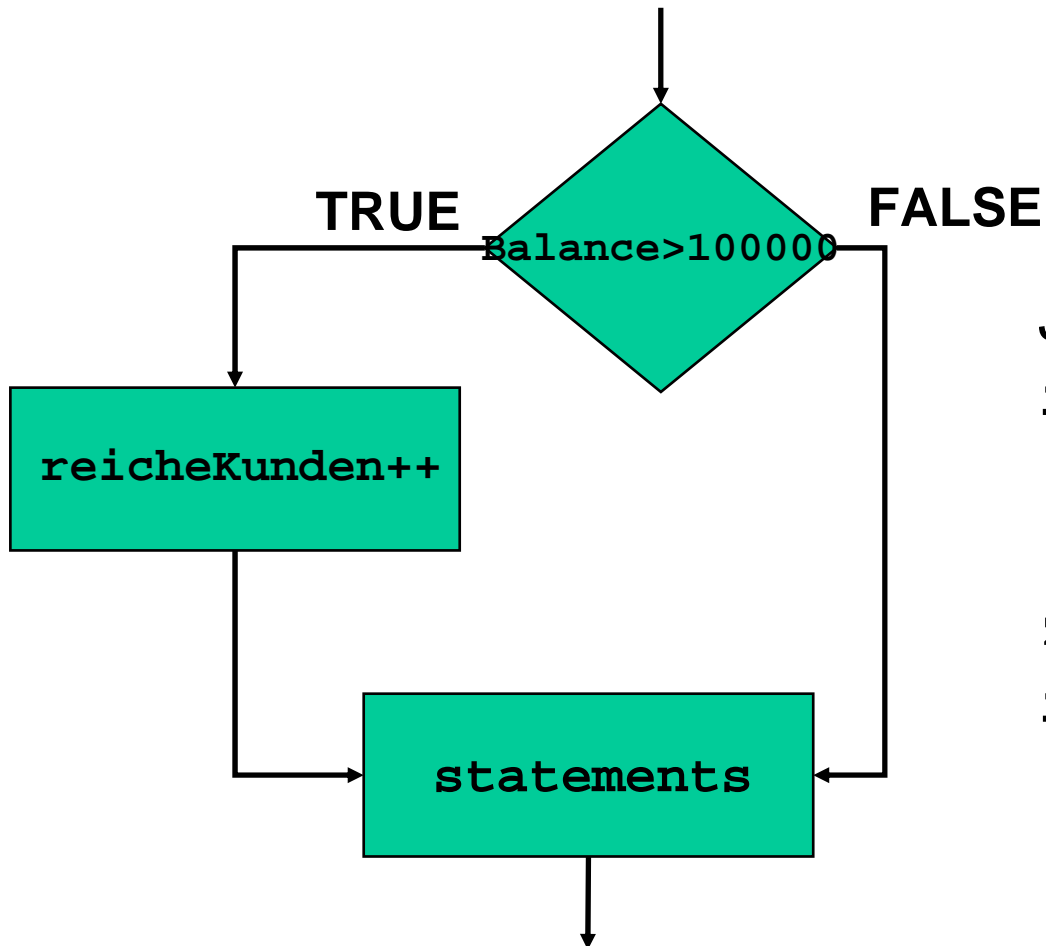
if



In der hier dargestellten Anwendung wird eine Bedingung überprüft (*bool-expr*). Ist diese erfüllt (*true*), dann wird der unter *statement1* beschriebene Programmcode ausgeführt, anschließend wird das Programm mit *statement* fortgesetzt. Ist die Bedingung nicht erfüllt (*false*), dann wird das Programm sofort mit *statement* fortgesetzt.

if

Syntax und Beispiel



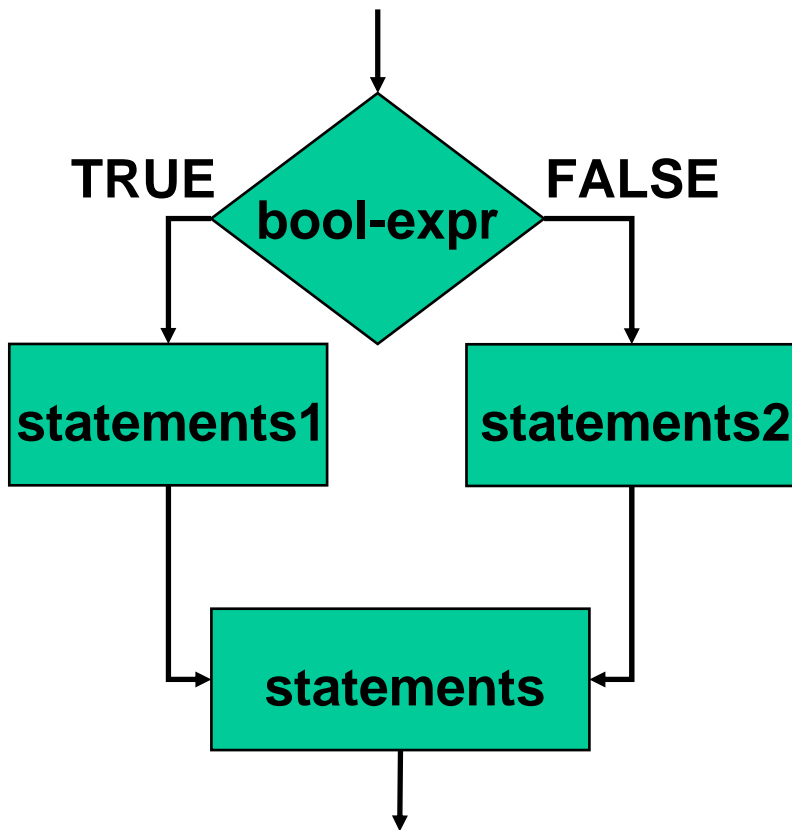
JavaSyntax:

```
if (bool-expr) {  
    statements1;  
}
```

z.B:

```
if (balance > 100000) {  
    reicheKunden++;  
}
```

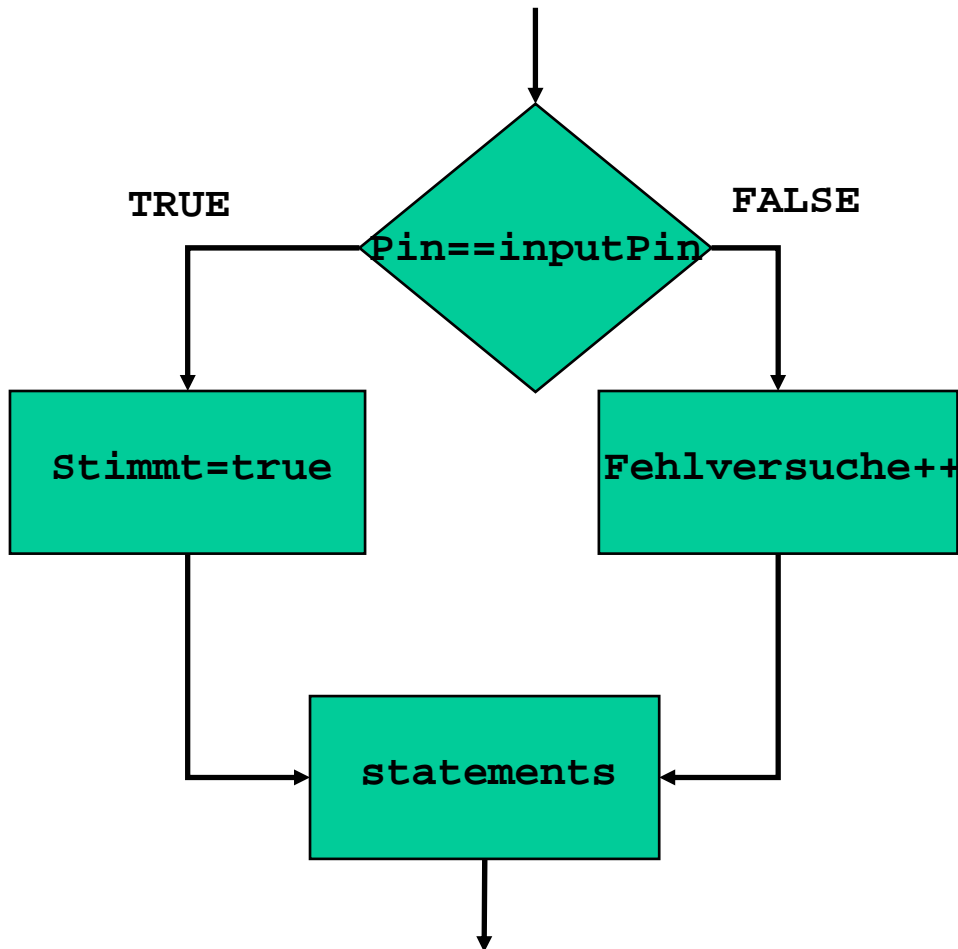
if-else



In der hier dargestellten Anwendung wird eine Bedingung überprüft (*bool-expr*). Ist diese erfüllt (*true*), dann wird der unter *statement1* beschriebene Programmcode ausgeführt, anschließend wird das Programm mit *statement* fortgesetzt. Ist die Bedingung nicht erfüllt (*false*), wird der unter *statement2* beschriebene Programmcode ausgeführt, anschließend wird das Programm mit *statement* fortgesetzt.

if-else

Syntax und Beispiel



JavaSyntax:

```
if (bool-expr) {  
    statements1;  
} else {  
    statements2;  
}
```

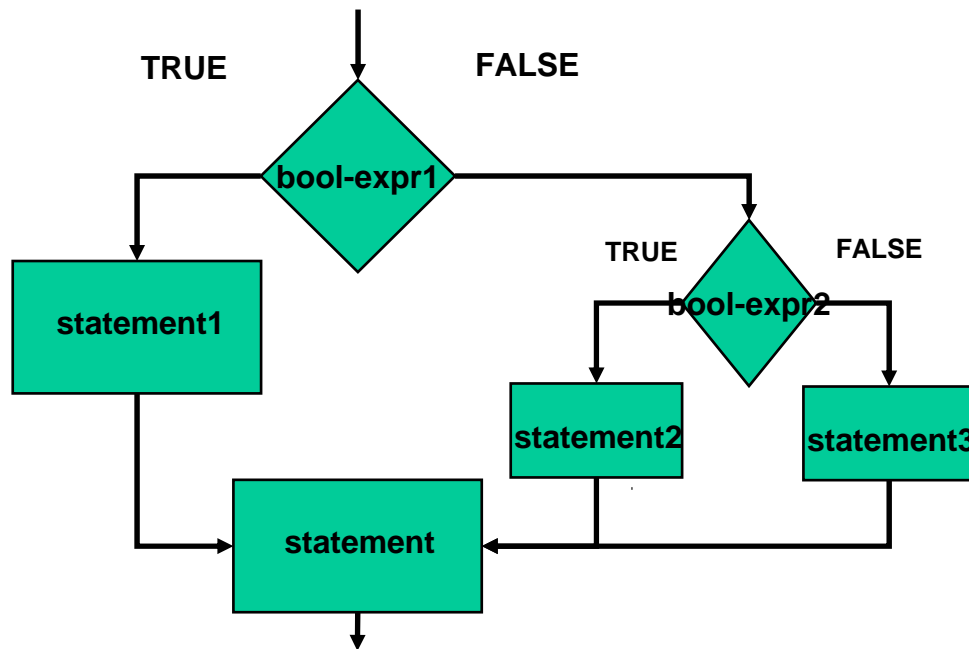
z.B:

```
if (pin==inputPin){  
    stimmt = true;  
} else {  
    fehlversuch++;  
}
```

if-else

Beispiel

In der hier dargestellten Anwendung wird eine Bedingung überprüft (*bool-expr1*). Ist diese erfüllt (*true*), dann wird der unter *statement1* beschriebene Programmcode ausgeführt, anschließend wird das Programm mit *statement* fortgesetzt. Ist die Bedingung *bool-expr1* nicht erfüllt (*false*), dann wird eine weitere Bedingung (*bool-expr2*) überprüft.

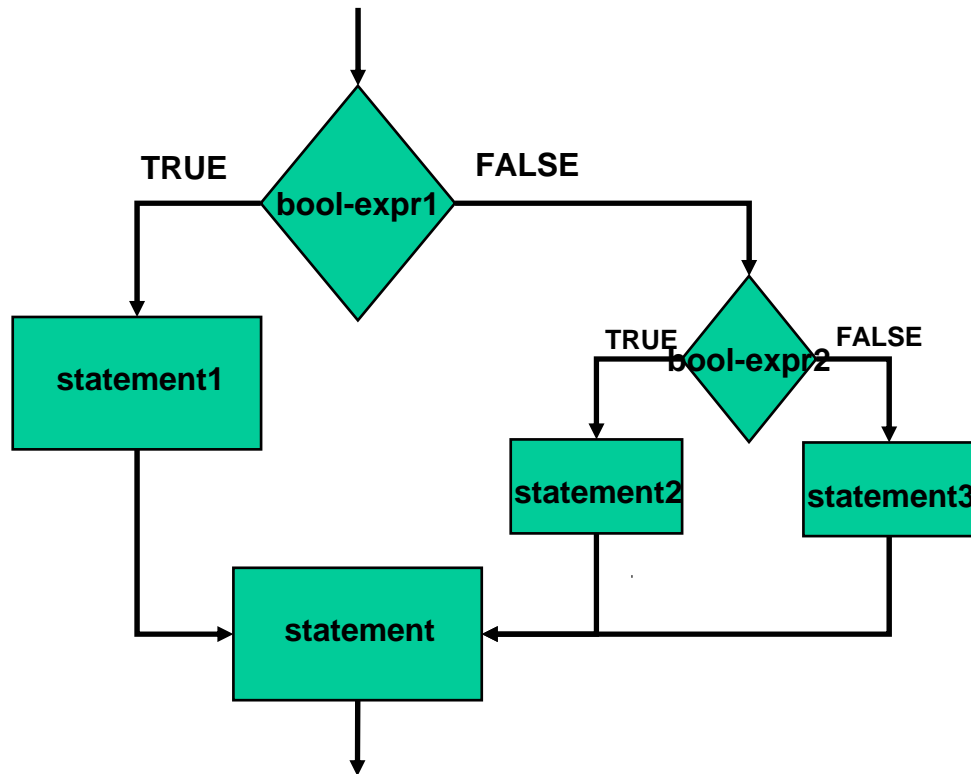


Ist diese Bedingung erfüllt, dann wird der unter *statement2* beschriebene Programmcode ausgeführt, anschließend wird das Programm mit *statement* fortgesetzt. Andern-falls wird der unter *statement3* beschriebene Programmcode ausgeführt und das Programm mit *statement* fortgesetzt.

if-else

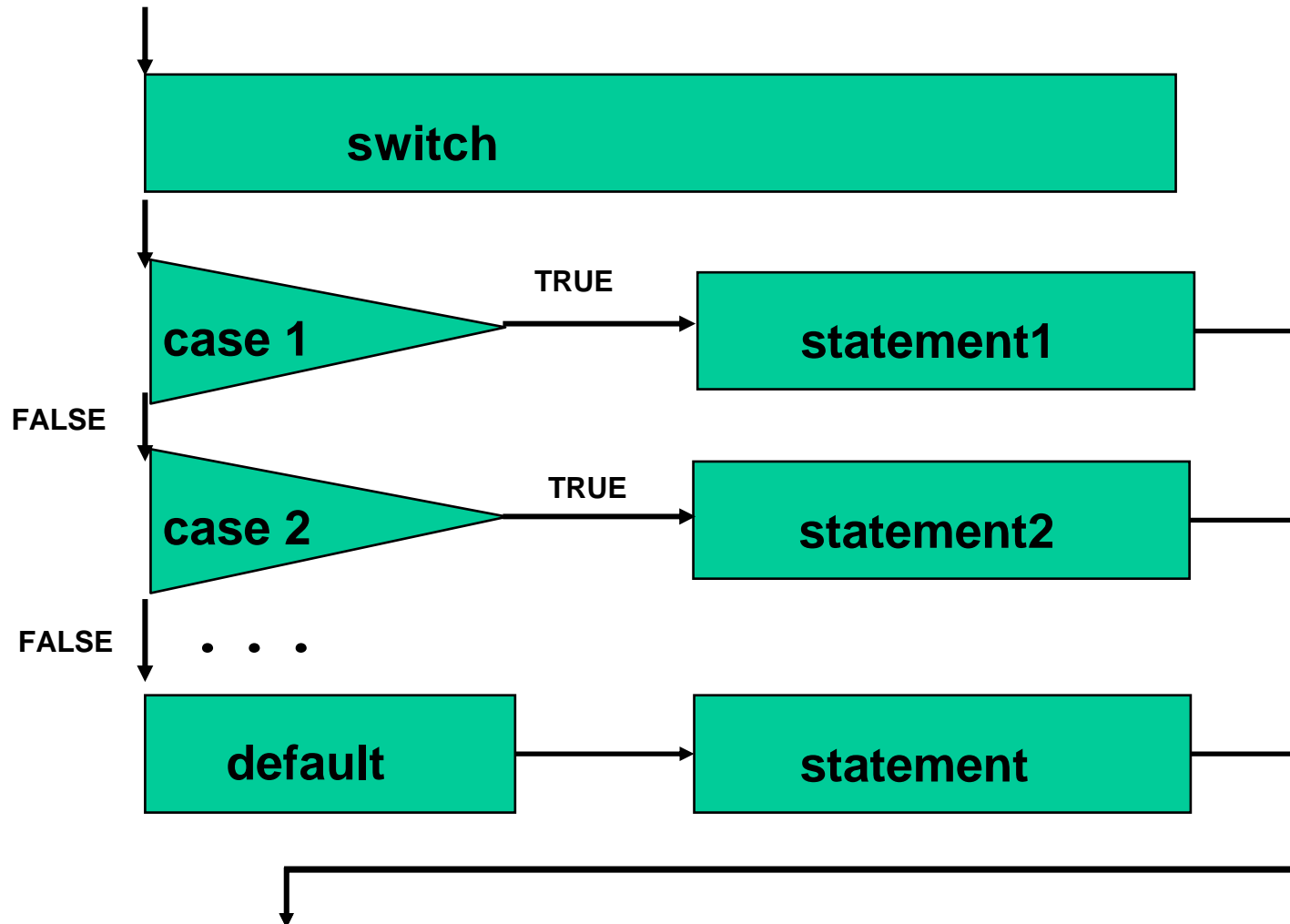
Beispiel

if-else kann beliebig tief verschachtelt werden.



```
...  
if (bool-expr1) {  
    statement1;  
} else if (bool-expr2) {  
    statement2;  
} else {  
    statement3;  
}  
statement;  
...
```

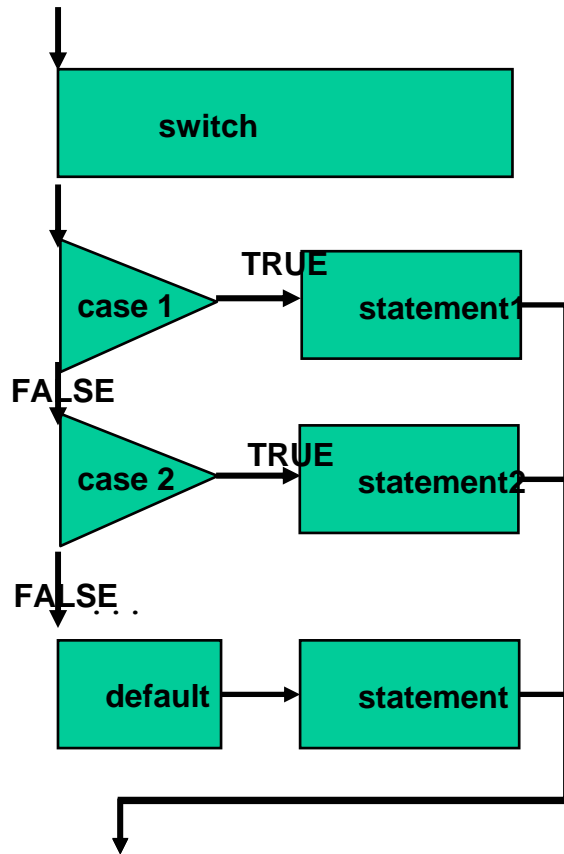
switch



switch (forts.)

1. An die switch – Anweisung wird eine Variable (integral selector) übergeben (z.B. `int`)
2. `case1` überprüft ob der Wert der Variable mit dem angegebenen Wert übereinstimmt.
3. Ist dies der Fall, dann wird die Anweisung in `statement1` ausgeführt.
4. Dann überprüft `case2` ob der Wert der Variable mit dem angegebenen Wert übereinstimmt.
5. Ist dies der Fall, dann wird die Anweisung in `statement2` ausgeführt.
6. Dann wird der Vergleich der Variablen in `case3` analog fortgesetzt.
7. Ist keine der angegebenen Bedingungen erfüllt, dann werden die Anweisungen vom Default-Zweig ausgeführt. (Der Default-Zweig ist optional)

switch - Syntax



```
switch(integral-selector) {  
  case integral-value1:  
    statement1; break;  
  case integral-value2:  
    statement2; break;  
  ...  
  default: statement;  
}
```

(**break** verhindert, dass alle restlichen Statements ausgeführt werden.)

switch (Bsp.)

```
int month = 12;
switch (month) {
    case 1: System.out.println("Jan"); break;
    case 2: System.out.println("Feb"); break;
    ...
    case 12: System.out.println("Dec"); break;
    default: System.out.println("error"); break;
}
```

(implizit: Wenn `month == 1`, dann gibt „Jan“ aus!)

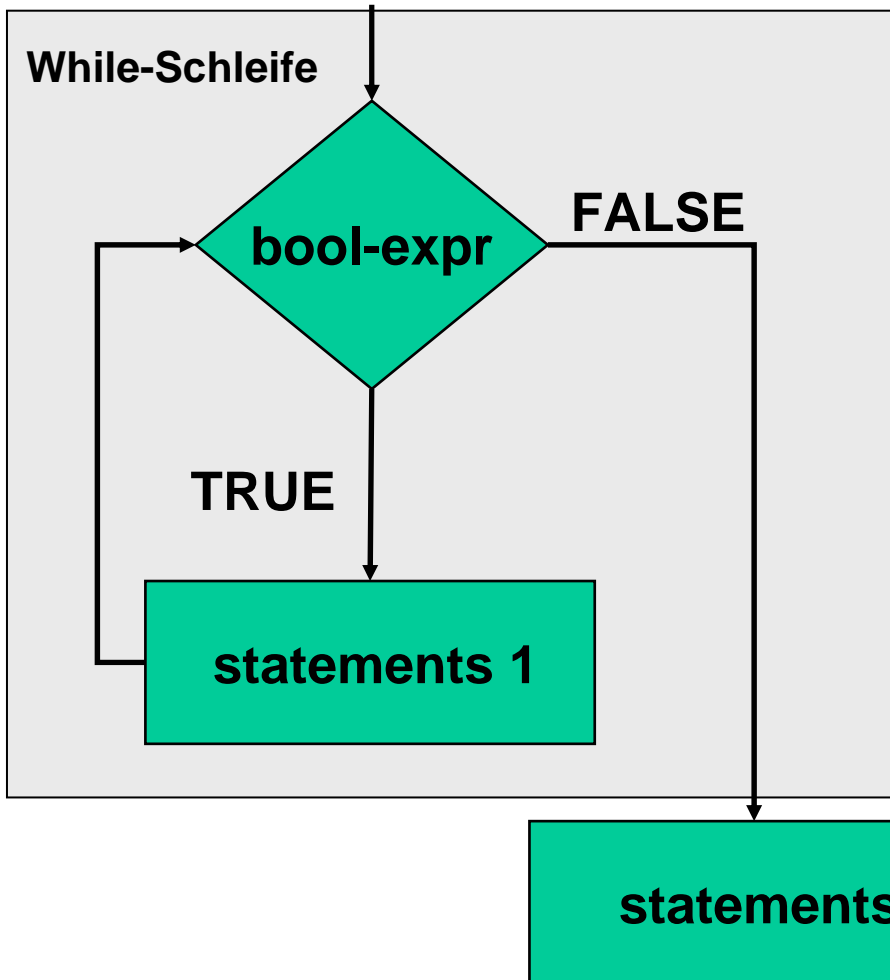
Iterationen

- Mit Iterationen (=Schleifen) kann der Programmfluss von Anweisungen, die mehrmals hintereinander durchgeführt werden sollen, gesteuert werden.
- Die Schleifen werden solange durchlaufen, bis ein Abbruchkriterium erfüllt ist.
- Das Abbruchkriterium befindet sich am Kopf oder am Fuß der Schleife.

z.B. PIN-Eingabe des Kunden:

Fordere den Kunden zur Eingabe der PIN auf, bis die richtige PIN bzw. 3* eine falsche PIN eingegeben wurden.

while



Die Anweisungen der while-Schleife werden ausgeführt, **solange** die bool-expr erfüllt (=true) ist. Die Überprüfung der Bedingung findet am Anfang der Schleife statt.

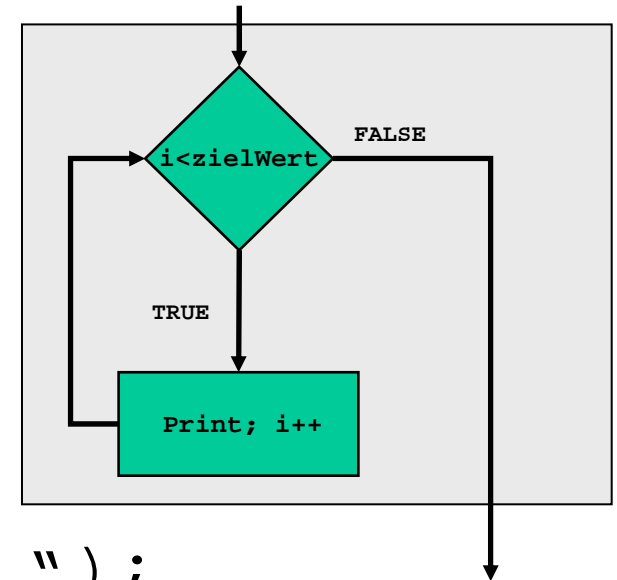
JavaSyntax:

```
while (bool-expr) {  
    statements1;  
}
```

while

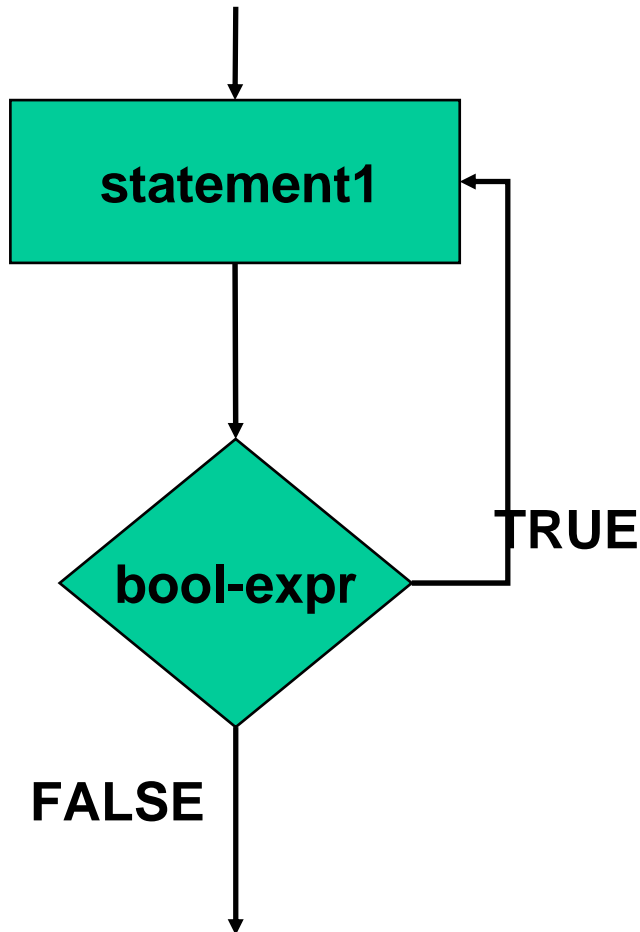
Beispiel

```
int zielWert = 100;
int i = 1;
while (i < zielWert) {
    System.out.print("i=" + i + ", ");
    i++;
}
```



Ausgabe: i=1,i=2,i=3,...,i=99

do-while



Die Anweisungen der do-while-Schleife werden ausgeführt, bis die bool-expr nicht mehr erfüllt ist. Die **Überprüfung der Bedingung findet am Ende der Schleife statt**. Sie wird somit mindestens einmal durchlaufen.

JavaSyntax:

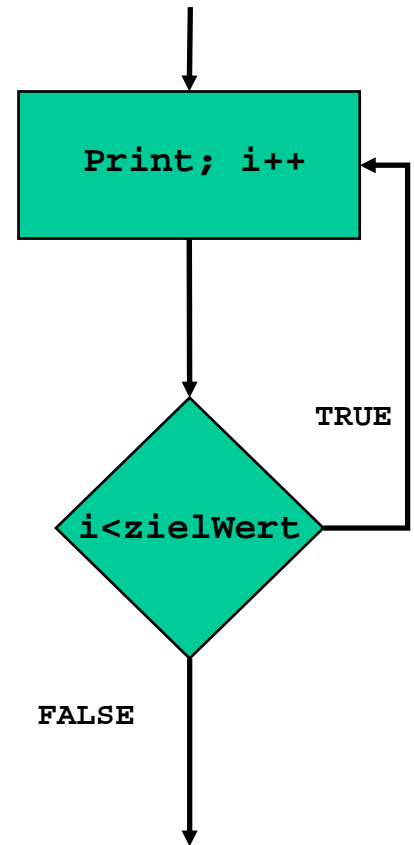
```
do {  
    statement1;  
} while (bool-expr)
```

do-while

Beispiel

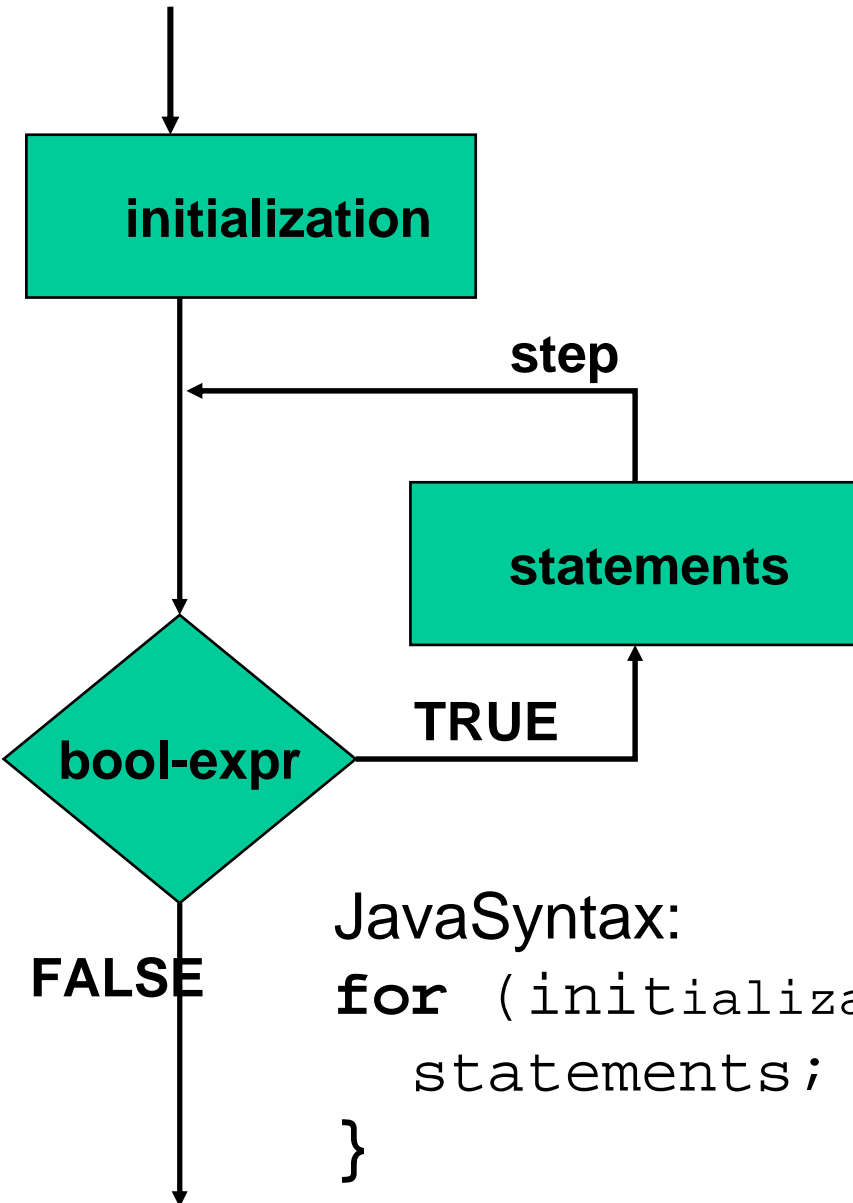
```
int zielWert = 1000;  
int i = 1;  
do {  
    System.out.println("i=" + i + ", ");  
    i++;  
} while (i < zielWert);
```

Ausgabe: i=1,
i=2,
i=3, ...
i=999,



for

Die Anweisungen der for-Schleife werden ausgeführt, solange die bool-expr erfüllt ist. Sie wird idR eingesetzt, um eine bekannte, d.h. im vorhinein festgelegte Anzahl von Schritten mit einem Zähler durchzuführen.



JavaSyntax:

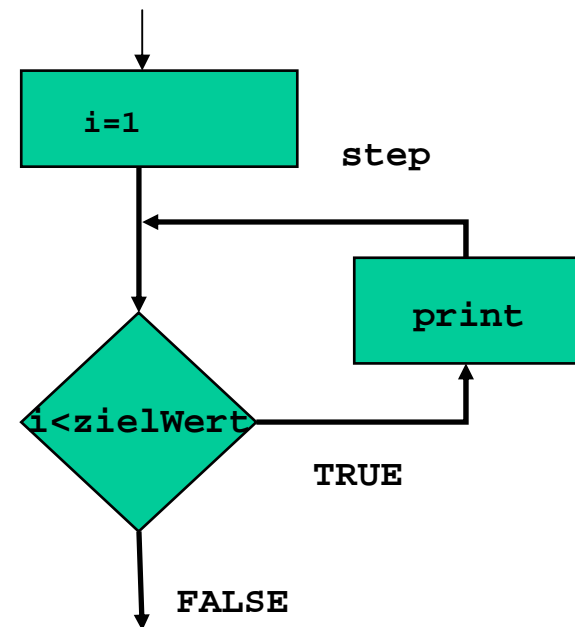
```
for (initialization; bool-expr; step) {  
    statements;  
}
```

for

Beispiel

```
int zielWert = 4;  
for (int i = 1; i < zielWert; i++) {  
    System.out.println(i+" , ");  
}
```

Ausgabe: 1,
2,
3,



For (Ergänzung)

Java Syntax bei Verwendung von mehreren Zählvariablen (gleichen Typs!):

```
for (initialization; bool-expr; step) {  
    statements; }
```

```
for (int i=0, j=10; i<=10; i++,j--) {  
    System.out.println("i=" + i + " i inkrementiert");  
    System.out.println("j=" + j + " j dekrementiert");  
}
```

Ausgabe: i=0 i inkrementiert
 j=10 j dekrementiert
 i=1 i inkrementiert
 i=9 j dekrementiert ...

Endlosschleifen

Problemfall: Endlosschleifen

(Abbruchbedingung kann nie erfüllt werden!)

```
for (int i=2; i>=1; i = i + 2) {  
    System.out.println(i);  
}
```

```
i = 10
```

```
while (i >= 1){  
    System.out.println(i);  
    i++;  
}
```

Endlosschleifen 2

Soll eine Endlosschleife programmiert werden:

```
while (true) {  
    // Code  
}
```

Iterationen Überblick

	Abfrage	Durchlauf	Anzahl der Durchläufe
while	zu Beginn	0, 1 oder mehrmals	unbestimmt
do-while	am Ende	1 oder mehrmals	unbestimmt
for	zu Beginn	0, 1 oder mehrmals	festgelegt

for

Beispiel Fakultät

Berechnen der Fakultät $n! = 1 * 2 * 3 * \dots * n$ einer Zahl mittels For-Schleife:

```
class Fakultaet{
    public static void main(String[] arg){
        int n = 20;

        long f = 1;
        for(long i = 2; i<=n; i++){
            f = f * i;
        }
        System.out.println(n + "! = "+ f);
    }
}
```

Sprunganweisungen

Sprunganweisungen können in allen Schleifen verwendet werden:

`break` bricht die Schleife ab

`continue` setzt die Schleife mit dem nächsten Durchgang fort, d.h. der Rest dieses einen Durchgangs wird übersprungen.

Um Lesbarkeit und Nachvollziehbarkeit des Codes zu gewährleisten empfehlen wir, Sprunganweisungen nicht einzusetzen!

Sprunganweisungen Beispiel

```
class Sprung {  
  
    public static void main (String[] args) {  
        int i=0, j=0;  
        while(i<10){  
            i++;  
            System.out.println("1. " + i);  
            if(i==7) break;  
        }  
        while(j<10){  
            j++;  
            if(j<7) continue;  
            System.out.println("2. " + j);  
        }  
    }  
}
```

Ausgabe:

1. 1
1. 2
1. 3
1. 4
1. 5
1. 6
1. 7
2. 7
2. 8
2. 9
2. 10

Nach dieser Einheit sollten Sie ...

- die Möglichkeiten bedingter Anweisungen kennen,
- die verschiedenen Schleifen kennen,
- ihre Eignung für spezielle Anforderung kennen,
- und alle Konstrukte in eigenen Programmen anwenden können.

Lernkontrolle

Schreiben Sie ein kleines Programm, das zu arabischen Zahlen die entsprechende römische Zahl ausgibt. Verwenden Sie dazu verschiedenartige Kontrollstrukturen.

Anleitung: Beginnen Sie zuerst mit einer vereinfachten Problemstellung (I, V, X) und erweitern Sie diese später.

Lernkontrolle

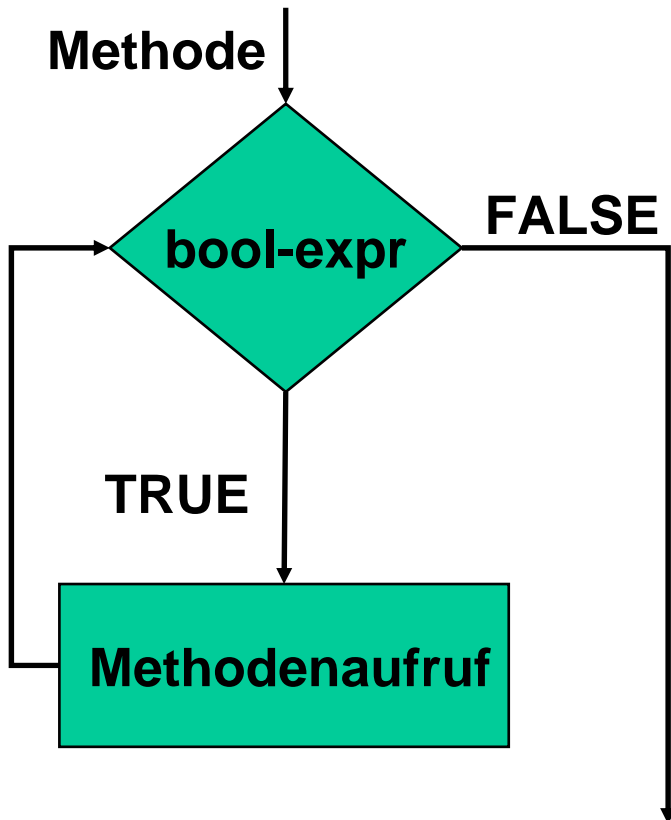
Fibonacci Sequence

Versuchen Sie die Ausgabe dieses Programms zu ermitteln.

```
class Fibonacci {
    public static void main (String[] args) {
        /** Ausgeben der Fibonacci Zahlen kleiner 50 */
        int lo = 1;
        int hi = 1;
        System.out.println(lo);
        while (hi < 50) {
            System.out.println(hi);
            hi = lo + hi;
            lo = hi - lo;
        }
    }
}
```

Exkurs: Rekursion*

* benötigt Methoden, die später durchgenommen werden



Die Rekursion stellt eine Alternative zur Verwendung von Schleifen dar. Rekursion bezeichnet den Aufruf der eigenen Methode mit anderem Parameter. Analog zu den Schleifen darf dieser Aufruf nur dann durchgeführt werden, wenn eine bestimmte Bedingung erfüllt ist.

Rekursion-Beispiel

Berechnen der Fakultät einer Zahl mittels Rekursion
(siehe Fakultaet.java):

```
static int fakulRekursiv(int i) {  
    if (i > 1) return i * fakulRekursiv(i-1);  
    else return 1;  
}
```

(Kapitel 14 in „Sprechen Sie Java“)

Rekursion-Beispiel (forts.)

