UNIVERSITÄT
LEIPZIG

Einführung in die Objekt-Orientierte Modellierung und Programmierung

Wintersemester 2025/2026

Dirk Zeckzer

Institut für Informatik

Teil VIII

Kontrollstrukturen

Java: Bedingte Anweisungen

if

```
1  int a;
2  int b;
3  int c;
4
5  if (a < b) {
6    c = a;
7  } else {
8    c = b;
9  }
```

```
1  int a;
2  int b;
3  int c;
4
5  if ( ! (a >= b) ) {
6    c = a;
7  } else {
8    c = b;
9  }
```

Java: Bedingte Anweisungen

switch

```
 1  int a;
 2  int b;
 3  int c;
 4
 5  if (a == 1) {
 6      c = b;
 7
 8  } else if (a == 2) {
 9      c = b * b;
10
11  } else if (a == 3) {
12      c = b * b * b;
13
14  } else {
15      c = 1;
16  }
```

```
 1  int a;
 2  int b;
 3  int c;
 4  switch (a) {
 5    case 1:
 6      c = b;
 7      break;
 8    case 2:
 9      c = b * b;
10      break;
11    case 3:
12      c = b * b * b;
13      break;
14    default:
15      c = 1;
16  }
```

Java: Bedingte Anweisungen

switch

```
 1  int a;
 2  int b;
 3  int c;
 4  switch (a) {
 5    case 1:
 6      c = b;
 7      break;
 8    case 2:
 9      c = b * b;
10      break;
11    case 3:
12      c = b * b * b;
13      break;
14    default:
15      c = 1;
16  }
```

```
 1  int a;
 2  int b;
 3  int c = 1;
 4  switch (a) {
 5    case 3:
 6      c *= b;
 7    case 2:
 8      c *= b;
 9    case 1:
10      c *= b;
11      break;
12    default:
13      c = 1;
14  }
```

Java: Bedingte Anweisungen

switch

```java
int monat;
int quartal;
switch (monat) {
  case 1:
  case 2:
  case 3:
    quartal = 1;
    break;
  case 4:
  case 5:
  case 6:
    quartal = 2;
    break;
  case 7:
  case 8:
  case 9:
    quartal = 3;
    break;
  case 10:
  case 11:
  case 12:
    quartal = 3;
    break;
  default:
    System.err.println("Monat falsch");
}
```

```java
int monat;
int quartal;
switch (monat) {
  case 1, 2, 3:
    quartal = 1;
    break;
  case 4, 5, 6:
    quartal = 2;
    break;
  case 7, 8, 9:
    quartal = 3;
    break;
  case 10, 11, 12:
    quartal = 3;
    break;
  default:
    System.err.println("Monat falsch");
}
```

Java: Bedingte Anweisungen

- Datentypen für `switch`
  - `int`, `Integer`
  - `byte`, `Byte`
  - `short`, `Short`
  - `char`, `Character`
  - `String`
  - `enum` Werte (kommt später)
- `long`, `Long` können nicht verwendet werden für
  - `switch`
  - Arrays

Java: Schleifen

while

```
 1  int[] werte = new int[7];
 2
 3  int index = 0;
 4  while (index < werte.length) {
 5    System.out.println("werte["
 6                       + index
 7                       + "] = "
 8                       + werte[index]
 9                      );
10    ++index;
11  }
```

Ausgabe:

werte[0] = 0
werte[1] = 0
werte[2] = 0
werte[3] = 0
werte[4] = 0
werte[5] = 0
werte[6] = 0

Java: Schleifen

for

```
 1  int[] werte = new int[7];
 2
 3  for (int index = 0;
 4        index < werte.length;
 5        ++index) {
 6    System.out.println("werte["
 7                       + index
 8                       + "] = "
 9                       + werte[index]
10                      );
11  }
```

Java: Schleifen

for

```java
int[] werte = new int[7];

for (int index = 0;
      index < werte.length;
      ++index) {
  System.out.println("werte["
                     + index
                     + "] = "
                     + werte[index]
                    );

}
```

while

```java
int[] werte = new int[7];

int index = 0;
while (index < werte.length) {

  System.out.println("werte["
                     + index
                     + "] = "
                     + werte[index]
                    );
  ++index;
}
```

Java: Schleifen

`for`, vereinfacht

```
1  int[] werte = new int[7];
2
3  for (int wert : werte) {
4    System.out.println("Wert = " + wert);
5  }
```

Ausgabe:
Wert = 0
Wert = 0
Wert = 0
Wert = 0
Wert = 0
Wert = 0
Wert = 0

Java

continue

```
1   for (int i = 0;
2        i < 10;
3        ++i) {
4     if (i % 2 == 0) {
5       continue;
6     }
7     System.out.println( i );
8   }
```

Ausgabe:

1
3
5
7
9

Java

break

```
1   int[] werte = new int[7];
2
3   for (int index = 0;
4        index < werte.length;
5        ++index) {
6     werte[index] = 10 - index;
7   }
8
9   int index;
10  for (index = 0;
11       index < werte.length;
12       ++index) {
13    if (werte[index] == 5) {
14      break;
15    }
16  }
17
18  if (index < werte.length) {
19    System.out.println( "werte["
20                        + index
21                         + "] = "
22                         + werte[index]
23                       );
24  }
```

Ausgabe:
werte[5] = 5

Java

`break`: Example using methods

```
1   int[] werte = new int[7];
2
3   for (int index = 0;
4        index < werte.length;
5        ++index) {
6     werte[index] = 10 - index;
7   }
8
9   int index;
10  for (index = 0;
11       index < werte.length;
12       ++index) {
13    if (werte[index] == 5) {
14      break;
15    }
16  }
17
18  if (index < werte.length) {
19    System.out.println( "werte["
20                        + index
21                         + "] = "
22                         + werte[index]
23                       );
24  }
```

```
1   public class Loops {
2
3     private int[] werte = new int[7];
4
5     public Loops() {
6       fill();
7     }
8
9     private void fill() {
10      for (int index = 0;
11           index < werte.length;
12           ++index) {
13        werte[index] = 10 - index;
14      }
15    }
```

Java

`break`: Example using methods

```
1   int[] werte = new int[7];
2
3   for (int index = 0;
4        index < werte.length;
5        ++index) {
6     werte[index] = 10 - index;
7   }
8
9   int index;
10  for (index = 0;
11       index < werte.length;
12       ++index) {
13    if (werte[index] == 5) {
14      break;
15    }
16  }
17
18  if (index < werte.length) {
19    System.out.println( "werte["
20                        + index
21                         + "] = "
22                         + werte[index]
23                       );
24  }
```

```
17    public int search(
18      final int value
19    ) {
20      int index;
21      for (index = 0;
22           index < werte.length;
23           ++index) {
24        if (werte[index] == value) {
25          break;
26        }
27      }
28
29      return index;
30    }
31
32    public void print(
33      int found
34    ) {
35      if (found < werte.length) {
36        System.out.println("werte["
37                           + found
38                           + "] = "
39                           + werte[found]
40        );
41      }
42    }
43  }
```

Java

`break`: Example using methods

```java
public class Suche {

  public static void main(
    String[] args
  ) {
    Loops loops = new Loops();
    int found = loops.search(5);
    loops.print(found);
    int found = loops.search(0);
    loops.print(found);
    int found = loops.search(10);
  }
}
```

```java
public class Loops {

  public Loops() {
    ...
  }

  public int search(
    final int value
  ) {
    ...
  }

  public void print(
    int found
  ) {
    ...
  }
}
```