Karlsruher Institut für Technologie

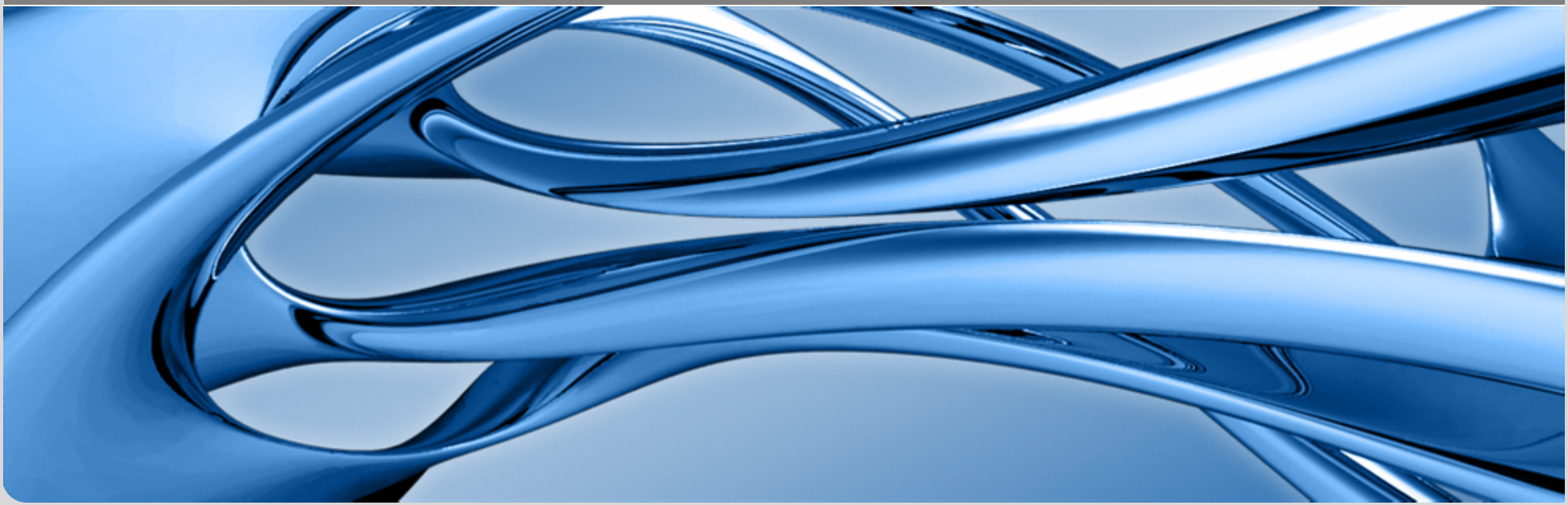
SDQ Software Design and Quality

Tutorium 3

Kontrollstrukturen

Christian Zielke | 13. November 2018

CHAIR FOR SOFTWARE DESIGN AND QUALITY



Gliederung

Gliederung

Was sind Kontrollstrukturen?

- Kontrollstrukturen lassen euch im Programm “springen”
- Programmteile werden unter bestimmten Bedingungen ausgeführt
- “if” (und “switch”) für Ausführung versch. Programmteile
- Schleifen für mehrmalige Ausführung von Programmteilen

if-Statement

- Das if-Statement verwirklicht das **Wenn-Dann-Prinzip in Java** (und generell in Programmiersprachen) $\rightarrow$ Fallunterscheidung
- Es existiert eine Kurzschreibweise (Bedingungsoperator):
  `variable = WENN ? DANN : SONST`
- Macht den Code aber schnell unverständlich

Syntax

- Nur `if`:

```
1       if (Bedingung) {
2         Anweisungen
3       }
```

- `if` gefolgt von `else`

```
1       if (Bedingung) {
2         Anweisungen
3       } else {
4         Anweisungen
5       }
```

- `if` gefolgt von **beliebig vielen** `else-if` und am Ende einem `else`

```
1       if (Bedingung) {
2         Anweisungen
3       } else if (Bedingung) {
4         Anweisungen
5       } else {
6         Anweisungen
7       }
```

Beispiele

- Nur `if`:

```
1     boolean debug = true;
2     if (debug) {
3       System.out.println("DEBUGAUSGABE");
4     }
```

- `if` gefolgt von `else`

```
1     if (a >= b) {
2       System.out.println(a + " ist groesser-gleich " + b);
3     } else {
4       System.out.println(a + " ist kleiner als " + b);
5     }
```

Beispiele

- `if` gefolgt von **beliebig vielen** `else-if` und am Ende einem `else`

```
1       if (a < 10) {
2         System.out.println("Kleiner 10");
3       } else if (10 <= a && a < 100){
4         System.out.println("Zwischen 10 und 100");
5       } else {
6         System.out.println("Groesser 100");
7       }
```

switch - case

- Bietet eine Alternative zur if-else Verschachtelung
- Nur für `char`, `byte`, `short`, `int`, `enum` oder `String` anwendbar

```
1       switch (Ausdruck) {
2         case Wert: Anweisung break;
3         ...
4         case Wert: Anweisung break;
5         default: Anweisung break;
6       }
```

- Werte müssen konstant sein (keine Variablen)
- default und break sind optional
- ohne break wird auch der nächste case mit ausgelöst, bis zum nächsten break

Beispiel

```
1   int x = 8;
2   int y = 4;
3   char operator = '+';
4   switch (operator) {
5     case '+' : System.out.println(x + y); break;
6     case '-' : System.out.println(x - y); break;
7     case '*' : System.out.println(x * y); break;
8     case '/' : System.out.println(x / y); break;
9     default: System.out.println("Kein gueltiges Zeichen");
10  }
```

Schleifen - Grundlagen

- Schleifen ermöglichen das häufige Wiederholen von einem Codeabschnitt
- Haben eine Abbruchbedingung
- Gefahr: Endlosschleifen bei unaufmerksamen Programmieren

while-Schleife

- Einfachste Form der Schleife
- Am **Anfang** jedes Durchlaufs wird die Ausführbedingung geprüft
  $\rightarrow$ kopfgesteuert
- Syntax:

```
while (Bedingung) {
    Schleifenanweisungen
}
```

- **Beispiel:** Zählen von übrigen Einträgen einer Liste

```
1    int entryCount = 0;
2    while (list.hasNext()) {
3      list.next();
4      entryCount++;
5    }
```

do-while-Schleife

- Am **Ende** jedes Durchlaufs wird die Ausführbedingung geprüft
  $\rightarrow$ fußgesteuert
- Die Schleifenanweisungen werden **mindestens einmal** ausgeführt
- Syntax:

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

- **Beispiel:** Würfeln, bis eine 6 geworfen wurde

```
1    int cast = 0;
2    do {
3      cast = random(1, 6);
4    } while (cast != 6);
```

for-Schleife

- Kopfgesteuerte **Zählschleife**, die eine Zählvariable verändert
- Syntax:

```
for (Init; Bedingung; Intervall) {
    Schleifenanweisungen
}
```

- Die `Initialisierung` wird vor Start der Schleife ausgeführt
- Die `Bedingung` wird vor jedem Schleifendurchlauf geprüft
- Die `Intervall`-Anweisung wird am Ende jedes Durchlaufs durchgeführt (Inkrement, Dekrement, sonstige Berechnungen)
- **Beispiel:** Summe von Zweierpotenzen berechnen

```
1    int sum = 0;
2    for (int i = 0; i <= 5; i++) {
3      sum += Math.pow(2, i);
4    }
```

break-Anweisung

- Bricht eine Schleife sofort ab
- Code innerhalb des aktuellen Schleifendurchlaufs, welcher nach dem `break` kommt, wird nicht mehr ausgeführt
- Es wird nur die **innerste** Schleife verlassen (bei Schleifenschachtelungen)
- `break` sparsam und nur gezielt verwenden, da es den Code unübersichtlich macht → Besser über Abbruchbedingung
- **Beispiel:** Zählen von übrigen Einträgen einer Liste (max. 10)

```
1     int entryCount = 0;
2     while (list.hasNext()) {
3       list.next();
4       entryCount++;
5       if (entryCount >= 10) {
6         break;
7       }
8     }
```

continue-Anweisung

- Bricht den aktuellen Schleifendurchlauf ab und springt zum Nächsten
- Nächster Durchlauf beginnt bei Prüfung der Abbruchbedingung
- `continue` sind manchmal nützlich, sollten aber ebenfalls zwecks Übersichtlichkeit wenig verwendet werden
- **Beispiel:** Summe von positiven Zahlen

```
1   int sumCount = 0;
2   int sum = 0;
3   int number = 0;
4
5   while (sumCount != 10) {
6     number = random(-10, 10);
7     if(number < 0) {
8       continue;
9     }
10    sum += number;
11    sumCount++;
12  }
```

Konsoleeingaben

- Mittels java.util.Scanner
- Mögliche Exceptions werden noch nicht behandelt.
- **Beispiel:**

```
1   import java.util.Scanner;
2
3   class Example {
4     public static void main(String args[]) {
5
6       Scanner scanner = new Scanner(System.in);
7       System.out.print("Gib eine Zahl ein: ");
8       String s = scanner.nextLine();
9     }
10  }
```

Übungsaufgaben

Positiv oder Negativ

Schreibe ein Programm, dass für eine gegeben Zahl überprüft, ob diese kleiner, größer oder gleich null ist und dies auf dem Terminal ausgibt.

Übungsaufgaben

Arbeitstage

- Schreibe ein enum mit Wochentagen
- Schreibe ein Programm, dass für einen gegebenen Wochentag entscheidet, ob dieser ein Arbeitstag ist, oder nicht und dies auf dem Terminal ausgibt

Übungsaufgaben

Gerade Zahlen

Schreibe ein Programm, dass alle geraden Zahlen zwischen 1 und 20 ausgibt. Verwende zuerst eine while- und dann eine for-Schleife.

Übungsaufgaben

Kleines Einmaleins

Schreibe ein Programm, dass das kleine Einmaleins auf dem Terminal ausgibt. Die Ausgabe soll so aussehen:

1x1 = 1

1x2 = 2

...

10x10 = 100

Übungsaufgaben

Intervallüberprüfung

Schreibe ein Programm, dass für eine Zahl $x \in [0, 99]$ überprüft in welchem Teilintervall [0,9] [10,19] ... [90,99] sie liegt.
Verwende switch/case.