Datum:

1 Wiederholungsanweisungen

Angenommen man möchte in Java eine unbestimmte Anzahl an Anweisungen (**Sequenz**) mehrfach hintereinander ausführen, ist das im Allgemeinen möglich, wenn man diese Sequenz so häufig im Quellcode aufschreibt, wie sie wiederholt ausgeführt werden soll.
Das ist die einfachste Lösung, bei der wirklich nicht viel Überlegung dahinter steht.

Man stelle sich aber jetzt mal vor, diese zu wiederholende Sequenz besteht aus 40 Einzelanweisungen und muss der Korrektheit wegen sieben Mal ausgeführt werden. Dann haben wir unseren Quellcode ganz schnell mal um 320 Zeilen erweitert.
Bei kleineren Projekten könnte man das ja verkraften, aber wollen wir das? **Nein!** Natürlich nicht.

Also brauchen wir einen work-around - eine Möglichkeit eine Sequenz mehrfach auszuführen, ohne diese mehrfach im Quellcode anzugeben.
Dafür gibt es die **Wiederholungsanweisung**.

Java bietet uns drei verschiedene Wiederholungsanweisungen, die in unterschiedlichen Fällen Anwendung finden.

Wiederholung mit fester Anzahl Abweisende Wiederholung Nicht-Abweisende Wiederholung

1.1 Wiederholung mit fester Anzahl

Weiß man, dass eine Sequenz eine festgelegte Anzahl wiederholt wird, so nutzt man die `for`-Anweisung. Diese hat immer die folgende Form:

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

Die geforderten Parameter haben die folgende Bedeutung:

`<Init>`	Deklaration einer ganzzahligen Zählvariable und der Zuweisung eines Anfangswerts (z.B. `int i = 0`)
`<Bedingung>`	Solange die Bedingung, abhängig von der Zählvariablen, erfüllt ist, wird die im inneren der `for`-Schleife angegebene Sequenz ausgeführt (z.B. `int i= 0; i < 5` - Die Sequenz wird fünf mal ausgeführt - wurde hingegen `int i=0; i <= 5` angegeben, so wird die Schleife sechs mal ausgeführt)
`<Update>`	Das Update der Zählvariablen erfolgt nach jedem Durchlauf entsprechend der angegebenen Zuweisung (z.B. `i++`, also i wird um eins erhöht. Bei `i--` wird i um eins verringert.)

Zu beachten ist, dass die <Bedingung> immer vor der Ausführung der Sequenz überprüft wird.

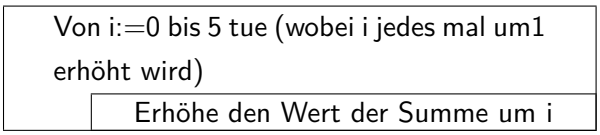
Beispiel Abbruchbedingung einbezogen:

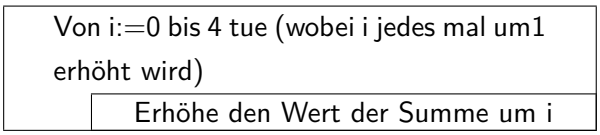
```
/**
* Berechnet die Summe aller ganzen
    Zahlen von 0 bis 5.
*/
sum = 0;
for(i=0; i<=5; i++) {
  sum = sum + i;
}
```

Beispiel Abbruchbedingung ausgeschlossen:

```
/**
* Berechnet die Summe aller ganzen
    Zahlen von 0 bis 4.
*/
sum = 0;
for(i=0; i<5; i++) {
  sum = sum + i;
}
```

Die zugehörigen Struktogramme verdeutlichen den minimalen Unterschied in der <Bedingung>.





1.2 Abweisende Wiederholung

```
while(<Bedingung>){
  <Anweisungen>
}
```

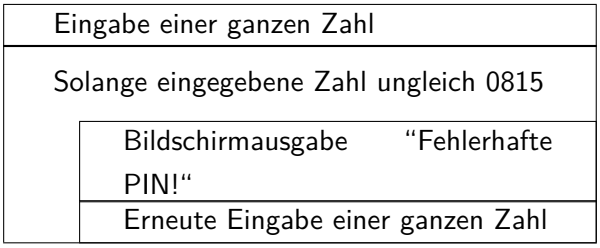
Wir überprüfen also **bevor** die Sequenz ausgeführt wird, ob die Bedingung erfüllt ist. Dabei kann es also passieren, dass die Sequenz garnicht ausgeführt wird.

Beispiel kopfgesteuerte Wiederholung:

```
int pin = input();

while(pin != 0815){
  System.out.println("Fehlerhafte
      PIN!");
  pin = input();
}
```

Struktogramm



1.3 Nicht-abweisende Schleife

Die in 1.2 erwähnte Wiederholungsanweisung prüft wie es der Name sagt bevor die Sequenz ausgeführt wird. Es kann nun aber natürlich auch mal passieren, dass eine solche Sequenz aber mindestens einmal ausgeführt werden soll, bevor die Abbruchbedingung überprüft wird. Um diese Anforderung zu erfüllen nutzt man die entsprechende *Fußgesteuerte* Wiederholung.

```
do{
  <Anweisungen>
} while(<Bedingung>)
```

Bei dieser wird die Sequenz einmal ausgeführt und die Bedingung wird **nach** jeder Wiederholung überprüft.

Beispiel fußgesteuerte Wiederholung:

```
do{
  System.out.println(number);
  number--;
} while(number > 0);
```

Struktogramm