2.3 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

2.3.1 Wiederholung mit fester Anzahl - Zählschleife

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>
}
```

Der Schleifenkopf lässt sich wie folgt erläutern:

`<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.)

Eine bestimmte Variable wird als Lauf- oder auch Zählvariable definiert und ihr wird ein Anfangswert zugewiesen. Diese Laufvariable wird nach jedem Durchlauf erhöht oder verringert. Die Anweisungen im Schleifenrumpf werden durchgeführt, bis die Laufvariable einen Endwert erreicht hat. Dieser wird im Schleifenkopf festgelegt.
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;
}
```

2.3.2 Abweisende Wiederholung

Die abweisende Schleife ist ein Sprachelement zur Umsetzung der algorithmischen Grundstruktur des abweisenden Zyklus (kopfgesteuerte Wiederholung). Möchten wir also, dass ein Anweisungsblock nur ausgeführt bzw. solange ausgeführt wird, wie eine bestimmte Bedingung erfüllt ist, hilft die kopfgesteuerte Wiederholungsanweisung.

Beispiel:

```
Solange zahl ungleich Null
   rechne: Ergebnis = 20/zahl;
   verringere Zahl um 1;
```

Die Schleifenbedingung wird vor Eintritt in den Anweisungsblock überprüft und die im Schleifenrumpf angegebenen Anweisungen werden so lange ausgeführt, wie die Bedingung wahr (`true`) ist.

Ist die Bedingung bereits vor dem ersten Eintritt in den Anweisungsblock falsch (`false`), wird der Schleifenrumpf nicht durchlaufen.

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

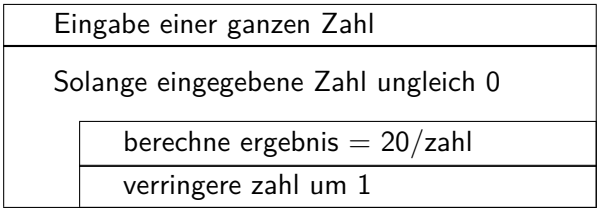
Da die abweisende Schleife in den meisten Programmiersprachen mit dem Schlüsselwort `while` beschrieben wird, nennt man sie meist auch `while`-Schleife.

Beispiel kopfgesteuerte Wiederholung:

```
int zahl = input();

while(zahl != 0){
  ergebnis = 20/zahl;
  zahl--;
}
```

Struktogramm



2.3.3 Nicht-abweisende Schleife

Wollen wir hingegen, dass eine Sequenz mindestens einmal ausgeführt wird, müssen wir die Bedingung nach einem Durchlauf prüfen.
Im Gegensatz zur abweisenden Schleife wird also bei der nicht-abweisenden Schleife die Bedingung am Ende geprüft.
Beispiel:

```
wiederhole
   zahl = zahl*2
      bis zahl größer als 100;
```

Die nicht-abweisende Schleife wird auch als fußgesteuerte Wiederholung, oder `do-while`-Schleife oder `repeat-until`-Schleife bezeichnet.

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

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

Beispiel fußgesteuerte Wiederholung:

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

Struktogramm