Kontrollstrukturen

Programmierung und Softwareentwicklung
Hörsaalübung

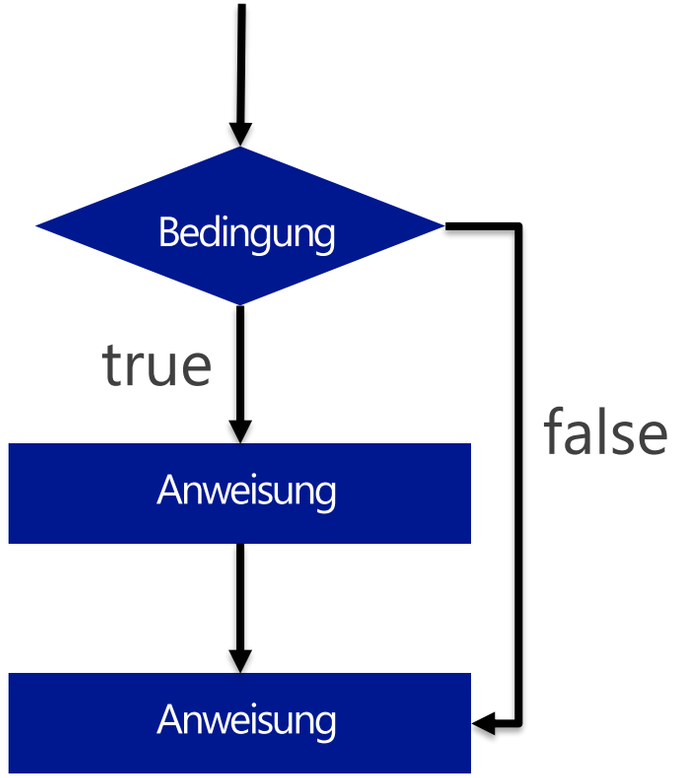
Kontrollflussstrukturen

Verzweigungen und Schleifen

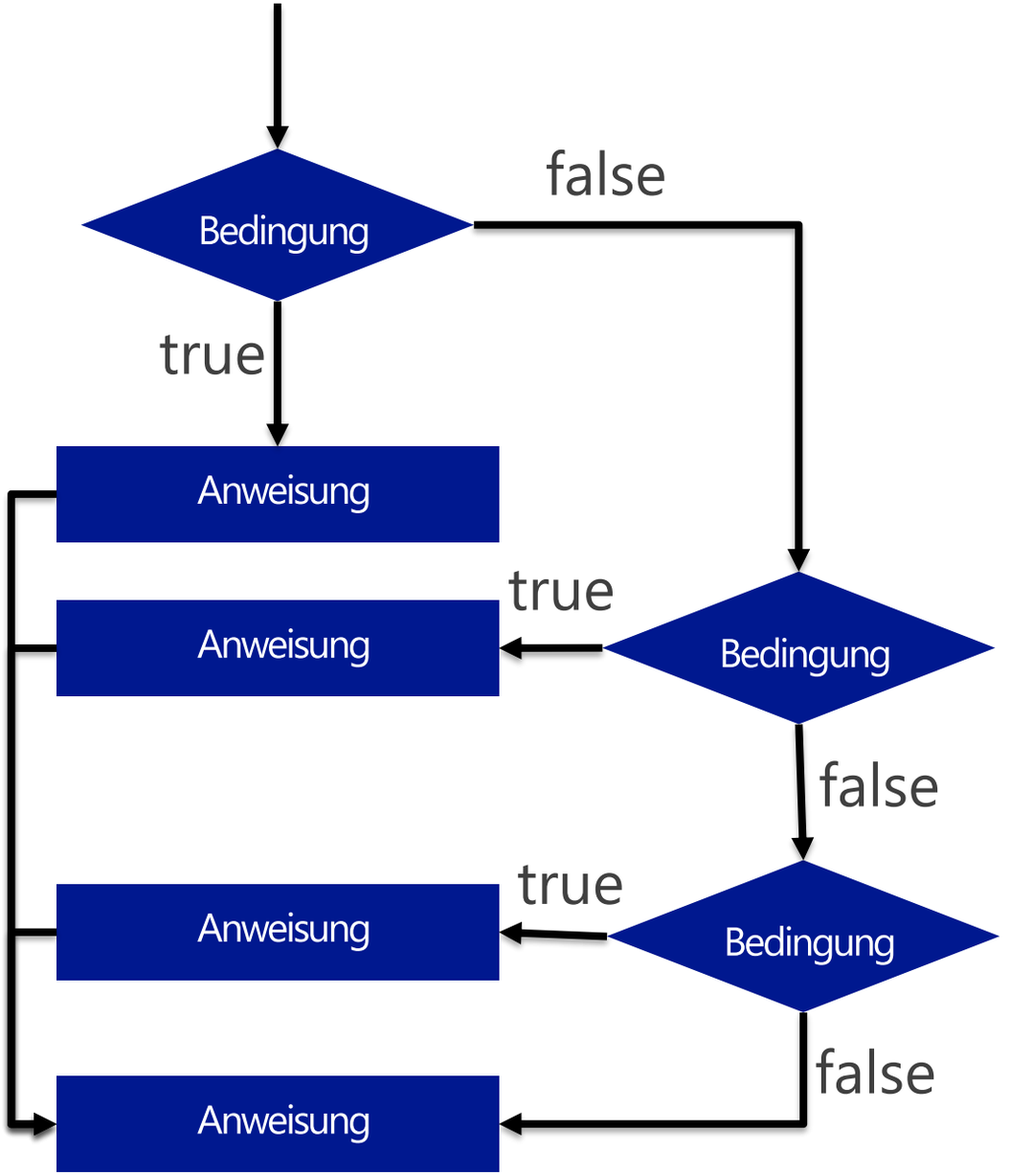
Verzweigungen

Verzweigungen

- Verzweigen den Programmverlauf
  - Ermöglicht bedingten Programmfluss
- Ausführung von Codeteilen kann optional werden
- Verzweigungen
  - If-else
  - Switch-case



If-Verzweigung



If-Verzweigung (II)

- Verzweigt den Programmfluss nach beliebiger boolscher Bedingung
- `else if` Zweig kann 0- bis *-mal vorkommen
- `else` Zweig kann 0- oder 1-mal vorkommen
- Bedingungen können logisch verknüpft werden

```
if (Bedingung A) {
    // Tritt ein, wenn die A wahr ist
} else if (Bedingung B) { // optional
    // Tritt ein, wenn A falsch, aber B wahr ist
} else if (Bedingung C) { // optional
    // Tritt ein, wenn A und B falsch, C aber wahr ist
} else { // optional
    // Tritt ein, falls A, B und C falsch waren
}
```

If-Verzweigung – Beispiel

```
if(a > 0 && a <= 2){
    System.out.println("Der Wert von a ist 1 oder 2!");
}else if(a == 3){
    System.out.println("Der Wert von a ist genau 3!");
}else{
    System.out.println("Die Variable a ist größer " +
    "3 oder kleiner gleich 0!");
}
```

Switch-Case

- Verzweigt Programmfluss nach Wert einer übergebenen Variablen
  - Erlaubt: primitive Variablen und String
- Beliebig viele Cases möglich
- `default` Case 0- oder 1-mal
  - Entspricht `else` Zweig

```
switch(a){
case 1: System.out.println("Der Wert von a ist 1!");
        break;
case 2: System.out.println("Der Wert von a ist 2!");
        break;
case 3: System.out.println("Der Wert von a ist 3!");
        break;
default: System.out.println("Der Wert von a ist größer " +
         "3 oder kleiner gleich 0!");
        break;
}
```

Switch-Case

Schlüsselwort `break`

- Wird benötigt, um Verzweigung nach Case zu beenden
- Sonst direktes Ausführen des nächsten Cases, bis zu einem `break`

```
switch(name) {
    case "Jiaozi": System.out.println("Delicious steemed dumpling");
    case "Zongzi": System.out.println("Steamed rice stuffed with different "
        + "fillings and wrapped in bamboo leaves");
    default: System.out.println("Generic food description");
}
```

Aufgabe 1

Schreiben Sie eine Operation `compare3Numbers`, die drei Variablen vom Typ `int` als Argumente übergeben bekommt und zuerst die größte, dann die mittlere und als letztes die kleinste Zahl auf der Konsole ausgibt.

Aufgabe 2

Schreiben Sie eine Abfrage `isLeapYear`, die eine Jahreszahl als Variable vom Typ `int` übergeben bekommt und ermittelt, ob es sich bei dem übergebenen Jahr um ein Schaltjahr handelt oder nicht.

Handelt es sich um ein Schaltjahr, wird `true` zurückgeliefert, wenn nicht `false`. Ein Jahr ist ein Schaltjahr, wenn es durch 4 teilbar ist. Es ist kein Schaltjahr, wenn es durch 100, aber nicht durch 400 teilbar ist. Der Rückgabetyp der Methode ist `boolean`.

Überprüfen Sie am Anfang der Abfrage, dass der Wert des Parameters `year` zwischen 1470 und 2970 liegt. Falls der Wert nicht in diesem Bereich liegt, soll `false` zurückgegeben werden.

Schleifen

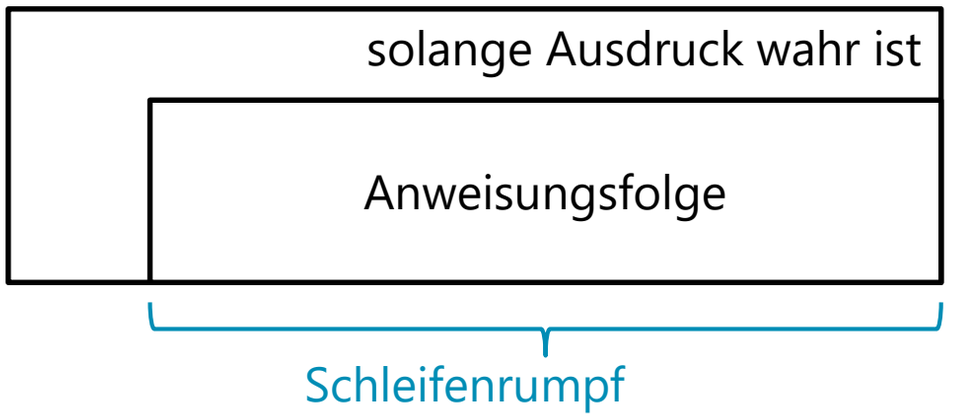
Schleifen (I)

Schleifen erlauben es, einen Code-Block mehrfach zu durchlaufen. Die Anzahl der Durchläufe wird dabei erst zur Laufzeit festgelegt.

Schleifen (II)

Wiederholte Ausführung einer Anweisungsfolge (im sogenannten Schleifenrumpf) so lange, bis ein Kriterium für den Abbruch erfüllt ist

Schleifen mit Vorabprüfung



Schleifen mit Prüfung am Ende

Schleifen in Java (I)

```
while (Ausdruck a) {
    Anweisungsfolge
}
```

```
int x = 10;
while (x > 0) {
        f(x);
        x--;
}
```

`a` ist Boolescher Ausdruck

Zu Beginn wird `a` ausgewertet:

- Anweisungsfolge ausgeführt, falls `a` wahr ist
- Schleife wird beendet, sonst

Wiederholung solange, bis der Ausdruck zu falsch ausgewertet wird bzw. die Schleifenausführung explizit abgebrochen wird (dazu später mehr)

Schleifen in Java (II)

```
for (Initialisierung; Ausdruck; Aktualisierung) {
        Anweisungsfolge
}
```

```
for (int x = 10; x > 0; x--) {
        f(x);
}
```

Zählschleife

Äquivalente `while`-Schleife:

```
Initialisierung;
while (Ausdruck) {
        Anweisungsfolge
        Aktualisierung;
}
```

Schleifen in Java (III)

```
do {
        Anweisungsfolge
} while (Ausdruck);
```

```
int x = 10;
if (x > 0) {
        do {
                f(x);
                x--;
        } while (x > 0);
}
```

Äquivalente `while`-Schleife:

```
Anweisungsfolge;
while (Ausdruck) {
        Anweisungsfolge
}
```

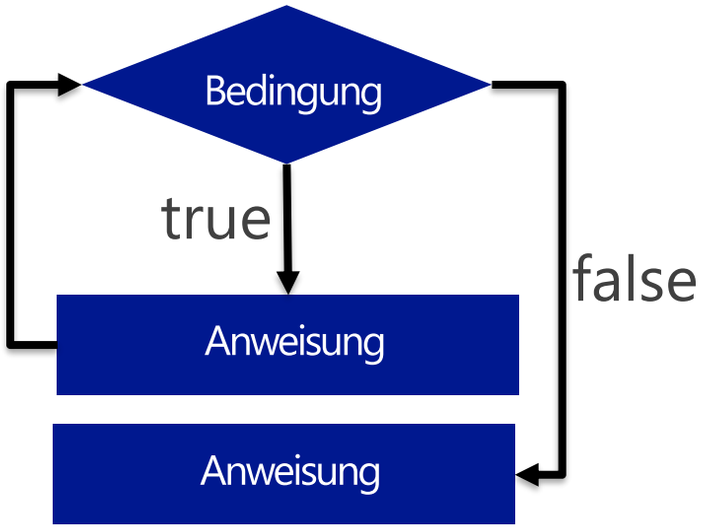
Schleifen im Vergleich (I)

`for`-Schleife

- Initialisierung
- Abbruchbedingung
- Schleifenschritt

sinnvoll wenn Anzahl der Durchläufe bekannt

```
for(int i = 0; i < 10; ++i){
    System.out.println("Der Wert von i ist " + i);
}
```



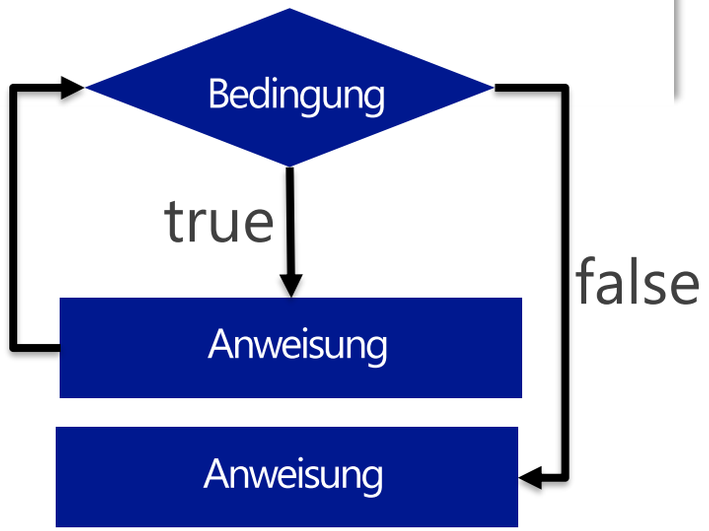
Schleifen im Vergleich (II)

`while`-Schleife

- Abbruchbedingung

Wird durchlaufen bis Bedingung falsch ist

```
int i = 0;
while(i < 10){
    System.out.println("Der Wert von i ist " + i);
    ++i;
}
```



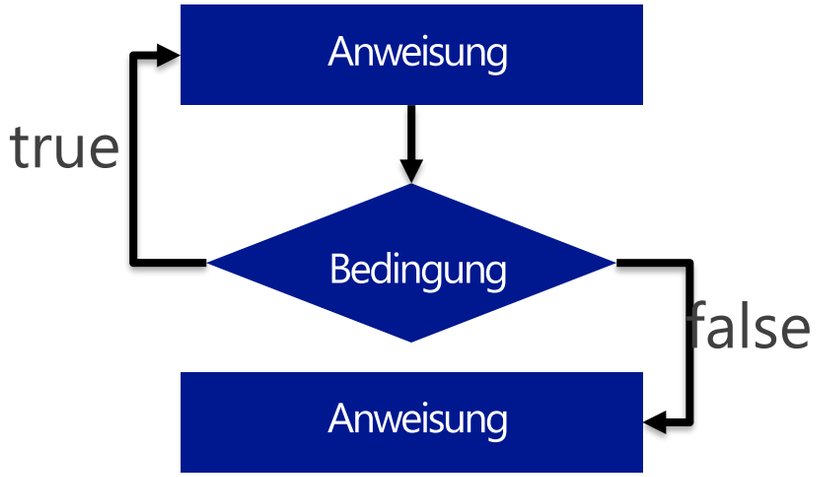
Schleifen im Vergleich (III)

`do-while`-Schleife

- Abbruchbedingung

Wird durchlaufen bis Bedingung falsch ist, aber mindestes einmal

```
int i = 0;
do {
    System.out.println("Der Wert von i ist " + i);
    ++i;
} while( i < 10 );
```



Terminierung von Schleifen

Eine Schleife terminiert, wenn sie nach endlich vielen Schritten beendet wird. Dies erfolgt entweder dadurch, dass:

- die boolesche Bedingung nicht mehr erfüllt ist
- die `break`-Anweisung aufgerufen wird

Mittels `continue` wird nicht die ganze Schleife, sondern nur der aktuelle Durchlauf des Schleifenrumpfs beendet.

`break` und `continue` verringern die Lesbarkeit von Programmen!

Schleifen

Schlüsselwörter

continue

- Schleifendurchlauf wird abgebrochen
- Geht direkt in den nächsten Schleifendurchlauf über

```
for(int i = 0; i < 10; ++i){
    if(i % 2 == 0)
        continue;

    System.out.println(i + " ist eine ungerade Zahl");
}
```

Schleifen

Schlüsselwörter

break

- Aktuelle Schleife wird beendet

```
int i = 0;
while(true) {
    i++;
    if(i % 7 == 0)
        break;
}
System.out.println(i + " ist durch 7 teilbar");
```

Schleifen

Wenn Schleifen verschachtelt sind, beeinflussen Schlüsselwörter standardmäßig die jeweils innerste Schleife.

Gezieltes Abbrechen

```
outer: for(int i = 0; i != 10; ++i) {
    for(int j = 0; j != 10; ++j) {
        if(i == 2 && j == 3)
            break outer;
    }
}
```

Schleifen in Java

```java
for (int i = 0; i < 50; i++) {
        System.out.println(i);
}
```

```java
int i = 0;
while (i < 50) {
        System.out.println(i);
        i++;
}
```

Initialisierung der Schleifenvariablen

Abbruchbedingung der Schleife

Schleifenschritt

ACHTUNG: Bei der **`for`**-Schleife endet die Lebendigkeit der Schleifenvariablen mit der Schleife!

```java
int i = 0;
if (i < 50) {
        do {
                System.out.println(i);
                i++;
        } while (i < 50);
}
```

Continue & Break

Befehl	Beschreibung
continue	• continue bricht den aktuellen Schleifendurchgang ab • Nächste Schleifeniteration wird angestoßen, d.h. Schleifenbedingung wird ausgewertet • Bei for-Schleifen wird der Schleifenschritt ausgeführt
break	• break bricht die unmittelbar umschließende Schleife vollständig ab • Erster Befehl nach der Schleife wird ausgeführt

```
for (int i = 0; i < 50; i++) {
        if (i % 2 == 0) {
                continue;
        }
        System.out.println(i);
}
```

```
int i = 0;
while (i < 50) {
        if (i % 2 == 0) {
                continue;
        }
        System.out.println(i);
        i++
}
```

i++;

Schleifen – Aufgaben (I)

Forme die folgenden `for`-Schleifen in äquivalente `while`-Schleifen um:

```
for(int i = 0; i != X; ++i) {
    result *= i;
}
```

```
for(int i = 1; i != X; ++i) {
    for(int j = 1; j != i; ++j) {
        if(i % j == 0) {
            System.out.println(j + " " + i);
        }
    }
}
```

Schleifen – Aufgaben (I)

Formen Sie die folgende `while`-Schleife in eine äquivalente `for`-Schleife um:

```
int i = 23;
int j = 0;
while(i < 42) {
    j = i;
    while(j > 23) {
        System.out.println(i + " " + j);
        --j;
    }
    ++i;
}
```

Schleifen – Aufgaben (I) – Lösungen (I)

```
int i = 0;
while(i != X) {
    result *= i;
    ++i;
}
```

```
int i = 1;
int j = 1;
while(i != X) {
    j = 1;
    while(j != i) {
        if(i % j == 0) {
            System.out.println(j + " " + i);
        }
        ++j;
    }
    ++i;
}
```

Schleifen – Aufgaben (I) – Lösungen (II)

```
for(int i = 23; i < 42; ++i) {
    for(j = i; j > 23; --j) {
        System.out.println(i + " " + j);
    }
}
```

Schleifen – Aufgaben (II)

Schreiben Sie für die folgenden mathematischen Funktionen jeweils eine Java Operation, welche die benötigten Werte als Argumente übergeben bekommt und das Ergebnis zurückgibt.

- $\sum_{i=1}^{b} {}^{1}/_{i}$
- $\prod_{i=1}^{n} i$
- $\sum_{i=0}^{n-1} \left(\prod_{j=1}^{x-i} j \right)$

Schleifen – Aufgaben (II) – Lösungen (I)

```java
public double f1(int b) {
    double result = 0;
    for(int i = 1; i <= b; ++i) {
        result += 1.0 / (double) i;
    }
    return result;
}
```

```java
public int f2(int n) {
    int result = 1;
    for(int i = 1; i <= n; ++i) {
        result *= i;
    }
    return result;
}
```

Schleifen – Aufgaben (II) – Lösungen (II)

```java
public int f3(int n, int x) {
    int result = 0;
    for(int i = 0; i < n; ++i) {
        int innerResult = 1;
        for(int j = 1; j < (x - i); ++j) {
            innerResult *= j;
        }
        result += innerResult;
    }
    return result;
}
```