Themen für heute

- Der Datentyp `boolean`
- Logische Verknüpfung von Bedingungen
- Die `while` Schleife
- Die `for` Schleife
- Stringverarbeitung
- Invarianten

Der Datentyp `boolean`

```
int x = 5;
System.out.println(x < 10);
```

Gibt aus: `true`

In Java ist `x < 10` ein Ausdruck des Typs `boolean`, genauso wie `x + 12` einer vom Typ `int` ist.

Werte des Typs `boolean` sind (nur) `true` und `false`.

Nach G. Boole (1815–1864), Britischer Logiker.

Logische Verknüpfung

Auf dem Typ `boolean` sind die zweistelligen Verknüpfungen `&&` und `||` erklärt:

- e_1 `&&` e_2 bedeutet: "e_1 und e_2"; "sowohl e_1, als auch e_2"; "e_1 und e_2 beide `true`".
- e_1 `||` e_2 bedeutet: "e_1 oder e_2"; "mindestens eins von beiden, e_1 oder e_2, ist `true`".

Außerdem gibt es die einstellige Operation `!`:

- `!`e bedeutet: "nicht e", "das Gegenteil von e".

&&	true	false
true	true	false
false	false	false

||	true	false
true	true	true
false	true	false

	!
true	false
false	true

Anwendung

Johannas Geburtstag ist der 21.12.

Angenommen, wir haben `int`-Variablen `day` und `month`.

Welcher Boole'sche Ausdruck ist `true` genau dann, wenn die Werte der beiden Variablen Johannas Geburtstag entsprechen?

Antwort

Der Ausdruck

```
day == 21 && month == 12
```

So können wir ihn verwenden:

```
if (day == 21 && month == 12)
     System.out.println("Happy birthday, Johanna!");
```

Anwendung

Wie drücken wir aus, dass die `double` Variable `heat` zwischen `100.0` und `120.0` liegt?

Antwort

```
100.0 <= heat && heat <= 120.0
```

Äquivalent ist

```
!(heat < 100.) && !(heat > 120.)
```

Äquivalent ist auch

```
!(heat < 100. || heat > 120.)
```

De Morgan's Gesetz:

```
!(a && b)  =  !a || !b
!(a || b)  =  !a && !b
```

Andere Boole'sche Ausdrücke

Methoden können einen `boolean` Wert zurückliefern:

Die Klasse `String` enthält die Methode `equals`, die einen `String` Parameter hat und einen `boolean` zurückliefert.

```
String answer;

System.out.println("Koennen Sie mir helfen?");
answer = console.readLine();
if (answer.equals("ja") || answer.equals("Ja")) {
    System.out.println("Danke!");
} else {
    System.out.println("Schade.");
}
```

Stringvergleiche

Es gibt auch die Methode `equalsIgnoreCase`, die Groß/Kleinschreibung ignoriert.

```
if (antwort.equalsIgnoreCase("ja"))
   System.out.println("Danke!");
```

Zum Vergleich von Strings soll man nicht `==` verwenden.

```
String x = "a";
System.out.println("ja" == "ja");
System.out.println("ja" == "j" + "a");
System.out.println("ja" == "j" + x);
```

Gibt aus:

```
true true false
```

Grund: `==` bezeichnet hier Identität, d.h. Repräsentierung an derselben Speicherstelle.

Stringvergleiche

Die Methode `compareTo` vergleicht nach der lexikographischen Ordnung, liefert aber einen `int` zurück:

s_1`.compareTo(`s_2`)` ist

- < 0, wenn s_1 alphabetisch vor s_2 kommt
- $= 0$, wenn s_1 und s_2 gleich sind
- > 0, wenn s_1 alphabetisch nach s_2 kommt.

Beispiele:

`"AAAaaaaaa".compareTo("meinSchluesseldienst")` ist < 0

`"Vorlesung".compareTo("Vorlesen")` ist > 0

Boole'sche Ausdrücke

Wir können auch selber solche Methoden definieren:

In `Bankkonto`:

```
public boolean ueberzogen() {
   return kontostand < 0.0;
}
```

Vewendung:

```
if (meinGiro.ueberzogen())
   System.out.println("Hoffentlich Monatsende.");
```

Boole'sche Variablen

Natürlich kann man auch Variablen des Typs `boolean` deklarieren:

```
boolean mitBedienung;
boolean tischGedeckt;
boolean draussen;

/* Initialisierung von draussen und tischGedeckt */

mitBedienung = !draussen || tischGedeckt;

if (mitBedienung)
   System.out.println("Hier keine Selbstbedienung!");
```

Übungen

Was ist an den folgenden Statements falsch ?

```
if cents > 0 then System.out.println(cents + " Cents");
```

```
if (1 + x > Math.pow(x, Math.sqrt(2)) y = y +x;
```

```
if (x = 1) y++; else if (x = 2) y = y + 2;
```

```
if (x && y == 0) p = new Point(x,y);
```

```
if (1 <= x <= 10) {System.out.println("Danke.");}
```

```
if (!antwort.equalsIgnoreCase("Ja ") ||
            !antwort.equalsIgnoreCase("Nein"))
    System.out.println("Antworten Sie mit Ja oder Nein.");
```

Vergleichen von Objekten

Objekte kann man mit `==` vergleichen.

Das bezeichnet physikalische Identität ("dasselbe").

Oft gibt es eine Methode `equals`, die die Werte der Instanzvariablen vergleicht.

```
Point p = new Point(4,5);
Point q = p;
Point r = new Point(4,5);
System.out.println(p==q);
System.out.println(q==r);
System.out.println(p.equals(q));
System.out.println(q.equals(r));
```

Ausgabe: `true false true true`.

Man schreibe eine `equals` Methode für Bankkontos.

Lösung

```
public boolean equals(Bankkonto konto) {
   return kontostand == konto.getKontostand();
}
```

Schaltjahre

Jedes vierte Jahr ist ein Schaltjahr, es sei denn, die Jahreszahl ist durch hundert teilbar. In diesem Fall liegt ein Schaltjahr nur vor, wenn die Jahreszahl durch 400 teilbar ist.

Beispiel: 2000 war ein Schaltjahr, 1900 war keins.

Man schreibe einen Boole'schen Ausdruck, der `true` ist, genau dann wenn `jahr` ein Schaltjahr ist.

Hilfe: `x % y` ist der Rest der ganzzahligen Division von `x` durch `y`. Z.B.: `12%5`=2.

Lösung

```
jahr % 4 == 0 && !(jahr % 100) == 0 || jahr % 400 == 0
```

Die While-Schleife

`while (` *bedingung* `)`

statement;

führt *statement* solange immer wieder aus, bis *bedingung* falsch wird (ggf. ad infinitum).

- *statement* ist typischerweise ein Block-Statement (Folge von Anweisungen in geschweiften Klammern)
- *statement* sollte in der Lage sein, *bedingung* zu beeinflussen.
- Ist *bedingung* von vornherein falsch, so wird *statement* gar nicht ausgeführt.

Zinsberechnung als Beispiel

Sie haben einen Kredit in Höhe von `auszahlung` (z.B. €240.000) aufgenommen.

Für Tilgung (Abzahlung) und Zinsen bezahlen Sie jährlich `rate` (z.B. €17000) an die Bank.

Der Zinssatz betrage `zinsSatz` Prozent.

Was von der Rate nach Zinsen übrigbleibt, wird zur Tilgung verwendet.

Dadurch reduziert sich die Kreditsumme und nächstes Jahr kann mehr getilgt werden.

Wie lange dauert es, bis der Kredit abbezahlt ist?

Lösung

```
double tilgung, zinsen;
int jahre = 0;
double restschuld = auszahlung;

while (restschuld > 0) {
    zinsen = restschuld * zinsSatz / 100.;
    tilgung = rate - zinsen;
    restschuld = restschuld - tilgung;
    jahre = jahre + 1;
}
```