Ausdrücke und Bedingungen

- Beispiel „Division“
  - Mit ganzen Zahlen
  - Mit Gleitkommazahlen
  - Negativ-Beispiel für Gleitkommazahlen
  - Ganzzahlige Division und Divisionsrest
  - String-Konkatenation
- Beispiel Geburtstag
  - Mit Strings
  - Mit ganzen Zahlen

Division

- Wir möchten zwei ganz Zahlen eingeben und dividieren:

```
print("Geben Sie den Dividenten (Zähler) an: ");
int divident = readInt();

print("Geben Sie den Divisor (Nenner) an: ");
int divisor = readInt();

println("Ergebnis: " + divident / divisor);
```

Division mit Gleitkommazahlen

- Ganze Zahlen (*int*) ignorieren eventuelle Nachkommastellen
- Um genaue Werte zu bekommen, kann man mit Gleitkommazahlen (*double*) arbeiten:

```
print("Geben Sie den Dividenten (Zähler) an: ");
double divident = readDouble();

print("Geben Sie den Divisor (Nenner) an: ");
double divisor = readDouble();

println("Ergebnis: " + divident / divisor);
```

Wir lernen 1/3

- Alle Variablen bekommen bei ihrer Vereinbarung einen Datentyp

```
print("Geben Sie den Dividenten (Zähler) an: ");
int divident = readInt();

print("Geben Sie den Divisor (Nenner) an: ");
int divisor = r
```

```
print("Geben Sie den Dividenten (Zähle
double divident = readDouble();
```

- Java prüft bei Zuweisungen, ob die Datentypen zuweisungskompatibel sind

```
int divisor = readDouble();
```

Wir lernen 2/3

- Gleitkommazahlen kann man mit *readDouble()* von der Konsole einlesen

- Beim Rechnen mit ganzen Zahlen werden eventuelle Nachkommastellen abgeschnitten.
  Das entspricht „Runden gegen 0“

```
Geben Sie den Dividenten (Zähler) an: 20
Geben Sie den Divisor (Nenner) an: 8
Ergebnis: 2
```

Wir lernen 3/3

- Man kann mit Zahlen *und Strings* „rechnen“

```
println("Ergebnis: " + divident / divisor);
```

Ergebnis: *int*

String **+** int: ***String-Konkatenation***
(engl.: to concatenate = aneinanderhängen)

- Es gilt die übliche bekannte Regel „Punkt vor Strich“; im Beispiel
  1. Berechnung des Divisionsergebnisses
  2. Wandeln des *int*-Wertes in einen String
  3. Verknüpfen der beiden Strings zu einem neuen, längeren String
- Die Alternative wäre möglich, aber umständlich zu schreiben:

```
print("Ergebnis: ");
println(divident / divisor);
```

Warum ich Doubles nicht besonders mag: Beispiel für einen Rechenfehler

```
public static void main(String[] args) {

    double d = 2.5 * 2.14;

    println(d);

}
```

Erkenntnisse

- Doubles rechnen genau (= mit Nachkommastellen)
- Aber so genau nun auch wieder nicht...

Konsequenz: Wir nutzen ***ganze Zahlen, wenn immer das geht***!

Problem bei der Division 1/2

- Die Division durch 0 ist in der Mathematik nicht erlaubt
- Das Folgende ist aber ein gültiges Java-Programm, kann also ausgeführt werden, egal welchen Divisor wir dann eingeben:

```
print("Geben Sie den Dividenten (Zähler) an: ");
int divident = readInt();

print("Geben Sie den Divisor (Nenner) an: ");
int divisor = readInt();

println("Ergebnis: " + divident / divisor);
```

Problem bei der Division 2/2

- Es tritt eine „Division by Zero“ auf, das Programm bricht ab
- Wir bekommen eine entsprechende Meldung
  mit Angabe des Grundes und der Position im Programm

```
Geben Sie den Dividenten (Zähler) an: 20
Geben Sie den Divisor (Nenner) an: 0
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at ws17.DivisionByZero.main(DivisionByZero.java:15)
```

Division durch Null

Dateiname

Zeilennummer

Noch eine Division

- Es gibt noch ein anderes Divisionszeichen in Java, das zur Berechnung des Divisionsrests benutzt wird:

```
println("Divisionsrest: " + divident % divisor);
```

Das Zeichen „%“ soll ebenfalls an den Bruchstrich erinnern

- Der Divisionsrest ist der ganzzahlige Teil, der übrig bleibt, wenn der Teil weg ist, der ganzzahlig dividiert werden kann (Beispiele gleich)
- Bei dieser Division darf ebenfalls nicht durch 0 dividiert werden -> Programmabbruch mit Fehlermeldung

Ganzzahl-Division: Wie oft passt x ganz in y?

Beispiele für die ganzzahlige Division

11 / 4	= ?
	→
11 / -4	= ?
	→
-11 / 4	= ?
	→
-11 / -4	= ?
	→

Divisions-Rest (*modulo*)

Was bleibt nach Abzug der ganzzahligen Division?

- Die Wirkung von % ist festgelegt durch die Gleichung

  a % b = a – (a / b) * b

- Beispiele für die Rest-Berechnung bei einer ganzzahligen Division

  11 % 4 = ?

  →

  11 % -4 = ?

  →

  -11 % 4 = ?

  →

  -11 % -4 = ?

  →

Vermeiden des Programmabbruchs

- Wir vermeiden den Programmabbruch durch eine Überprüfung vor der Berechnung der Division

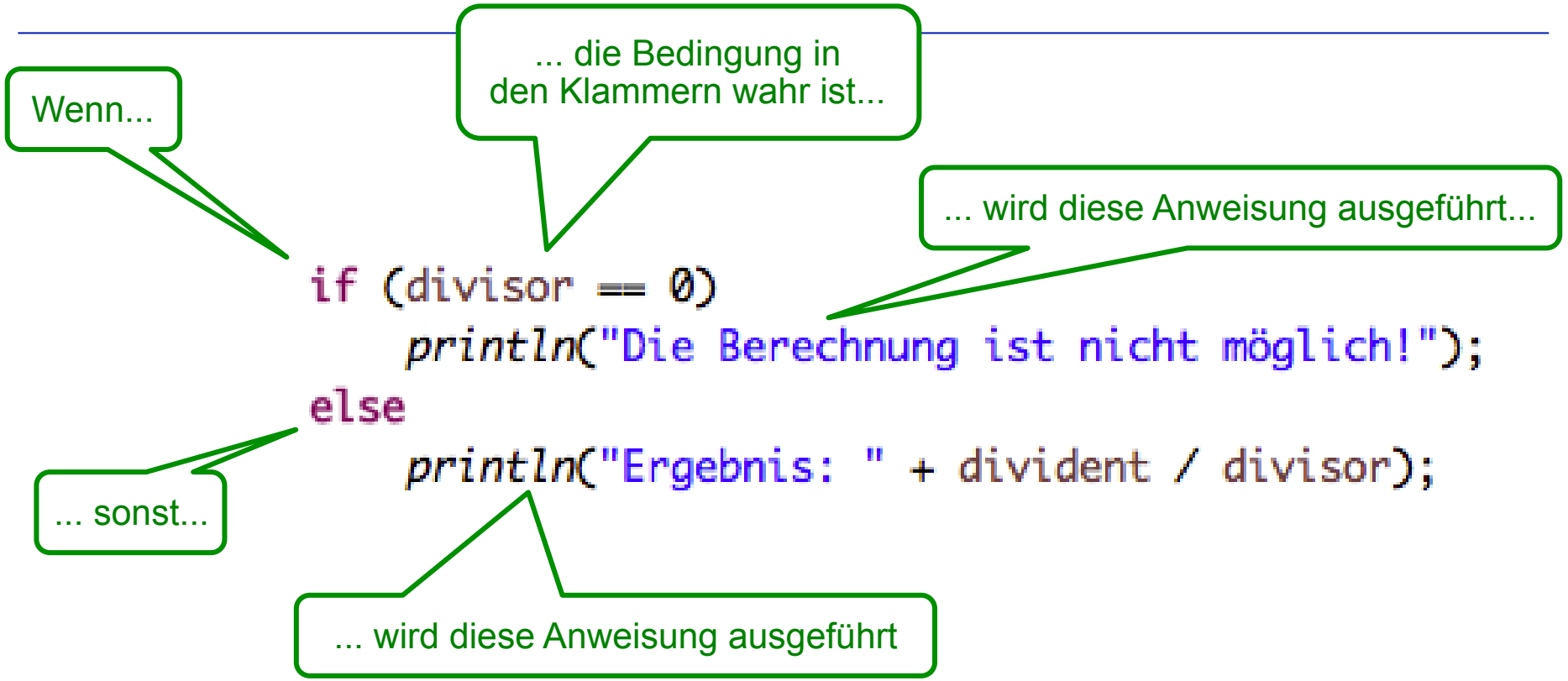
Wenn als Divisor eine 0 eingegeben wurde,
    geben wir eine entsprechende Meldung aus
sonst (alles ist in Ordnung)
    berechnen wir den Ausdruck

- In Java-Schreibweise:

Weil das einfache Gleichheitszeichen für die Zuweisung „verbraucht“ ist, schreiben wir zwei Gleichheitszeichen für den Vergleich zweier Zahlen

```
if (divisor == 0)
    println("Die Berechnung ist nicht möglich!");
else
    println("Ergebnis: " + divident / divisor);
```

Formal


Wenn...
... die Bedingung in den Klammern wahr ist...
... wird diese Anweisung ausgeführt...
if (divisor == 0)
println("Die Berechnung ist nicht möglich!");
else
println("Ergebnis: " + divident / divisor);
... sonst...
... wird diese Anweisung ausgeführt

Der Datentyp *boolean*

- Der Ausdruck in den Klammern der Bedingung muss entweder den Wert *wahr* oder den Wert *falsch* ergeben
- In Java gibt es dafür den Datentyp *boolean*

- Variablen vom Typ *boolean* können nur die beiden Werte *true* (wahr) und *false* (falsch) annehmen

Der „else“-Fall muss nicht genannt werden

- Wir vermeiden den Programmabbruch durch eine Überprüfung vor der Berechnung der Division

  Wenn als Divisor *keine* 0 eingegeben wurde,
  berechnen wir den Ausdruck

- In Java-Schreibweise:

```
if (divisor != 0)
    println("Ergebnis: " + divident / divisor);
```

- Wenn die Bedingung wahr ist, wird die Anweisung ausgeführt, sonst: ---

Beispiel „Geburtsdatum abfragen“

```
println("Bitte geben Sie ein gültiges Geburtsdatum seit dem 1. Januar 1900 ein.");
print("Geben Sie das Datum im Format tt.mm.yyyy ein: ");
String birthday = readString();

// nehmen wir an, es handelt sich um ein gültiges Datum

if (birthday == "02.10.2017")
    println("Das ist ja heute!");
else
    println("Das ist nicht heute.");
```

Das Problem:
Der Gleichheitsoperator tut bei Strings ***nicht*** das, was man erwarten würde

```
Bitte geben Sie ein gültiges Geburtsdatum seit dem 1. Januar 1900 ein.
Geben Sie das Datum im Format tt.mm.yyyy ein: 02.10.2017
Das ist nicht heute.
```

Beispiel „Geburtsdatum abfragen“

```
println("Bitte geben Sie ein gültiges Geburtsdatum seit dem 1. Januar 1900 ein.");
print("Geben Sie das Datum im Format tt.mm.yyyy ein: ");
String birthday = readString();

// nehmen wir an, es handelt sich um ein gültiges Datum

if (birthday == "02.10.2017")
    println("Das ist ja heute!");
else
    println("Das ist nicht heute.");
```

Das Problem:
Der Gleichheitsoperator tut bei Strings nicht das, was man erwarten würde

```
Bitte geben Sie ein gültiges Geburtsdatum seit dem 1. Januar 1900 ein.
Geben Sie das Datum im Format tt.mm.yyyy ein: 02.10.2017
Das ist nicht heute.
```

Richtiger Vergleich zweier Strings

```
println("Bitte geben Sie ein gültiges Geburtsdatum seit dem 1. Januar 1900 ein.");
print("Geben Sie das Datum im Format tt.mm.yyyy ein: ");
String birthday = readString();

// nehmen wir an, es handelt sich um ein gültiges Datum

if (birthday.equals("02.10.2017"))
    println("Das ist ja heute!");
else
    println("Das ist nicht heute.");
```

Strings vergleicht man mit

equals

- Die Funktion *equals* vergleicht zwei Strings auf ihren Inhalt
- Sie liefert genau dann *true* (wahr) als Ergebnis, wenn beide Strings gleich lang sind und die selben Zeichen beinhalten
- Der Operator „==“ angewendet auf Strings vergleicht deren Speicheradresse (nicht Länge und Inhalt)

Beispiele für String-Vergleiche

- "Test7".equals("Test7")

  →

- "Test7".equals("Test8")

  →

- "Test49".equals("Test" + 7 * 7)

  →

- ("T" + "est7").equals("Test" + "7")

  →

- "7Test".equals(3 + 4 + "T" + "e" + "s" + "t")

  →

- "Test7".equals("T" + "e" + "s" + "t" + 3 + 4)

  →

Zurück zum Beispiel

- Wie kann man feststellen, ob es sich um eine gültige Datumsangabe handelt?

```
println("Bitte geben Sie ein gültiges Geburtsdatum seit dem 1. Januar 1900 ein.");
print("Geben Sie das Datum im Format tt.mm.yyyy ein: ");
String birthday = readString();
```

-> Strings sind dafür nicht geeignet

-> Idee: Datum als drei ganze Zahlen einlesen

Definitionsbedarf: „Dangling Else“

- if (*Bedingung1*)
  if (*Bedingung2*)
  Anweisung1
  else
  Anweisung2

Beispiel für zwei mögliche Interpretationen: siehe Formatierung

a)
```
if ( Bedingung1 )
    if ( Bedingung2 )
        Anweisung1
    else
        Anweisung2
```

b)
```
if ( Bedingung1 )
    if ( Bedingung2 )
        Anweisung1
else
    Anweisung2
```

Beispiel und Regelung: „Dangling Else“

```
int wert1 = 5;
int wert2 = 5;

int ausgabe = 10;
if ( wert1 > 10 )
    if ( wert2 > 10 )
        ausgabe = 100;
    else
        ausgabe = 1000;

println( ausgabe );      → ?
```

```
int wert1 = 5;
int wert2 = 5;

int ausgabe = 10;
if ( wert1 > 10 )
    if ( wert2 > 10 )
        ausgabe = 100;
else
    ausgabe = 1000;

println( ausgabe );      → ?
```

Anweisungsfolgen in Java

- Eine ***Anweisungsfolge*** fasst eine Reihe von Anweisungen zu einer neuen " großen" Anweisung zusammen; Beispiel

```
int eingabe = readInt();
if (eingabe != 0) {
        x = 3 / eingabe;
        println( x );
}
```

- Anweisungsfolgen werden durch **{** eingeleitet und durch **}** abgeschlossen

- Anmerkung:
  "Einfache" Anweisungen werden durch ein *Semikolon* abgeschlossen (Anweisungsfolgen nicht)

Beispiel für eine Anweisungsfolge

```
print("Geben Sie Ihren Geburtstag ein: ");
int day = readInt();
print("Geben Sie Ihren Geburtsmonat ein: ");
int month = readInt();

if (month == currentMonth() && day == currentDayOfMonth()) {
    println("Das ist ja heute:");
    println("Herzlichen Glückwunsch!");
}
```

Das *if* bezieht sich auf beliebig viele Anweisungen von der öffnenden bis zur schließenden Klammer

Haben Sie Fragen?