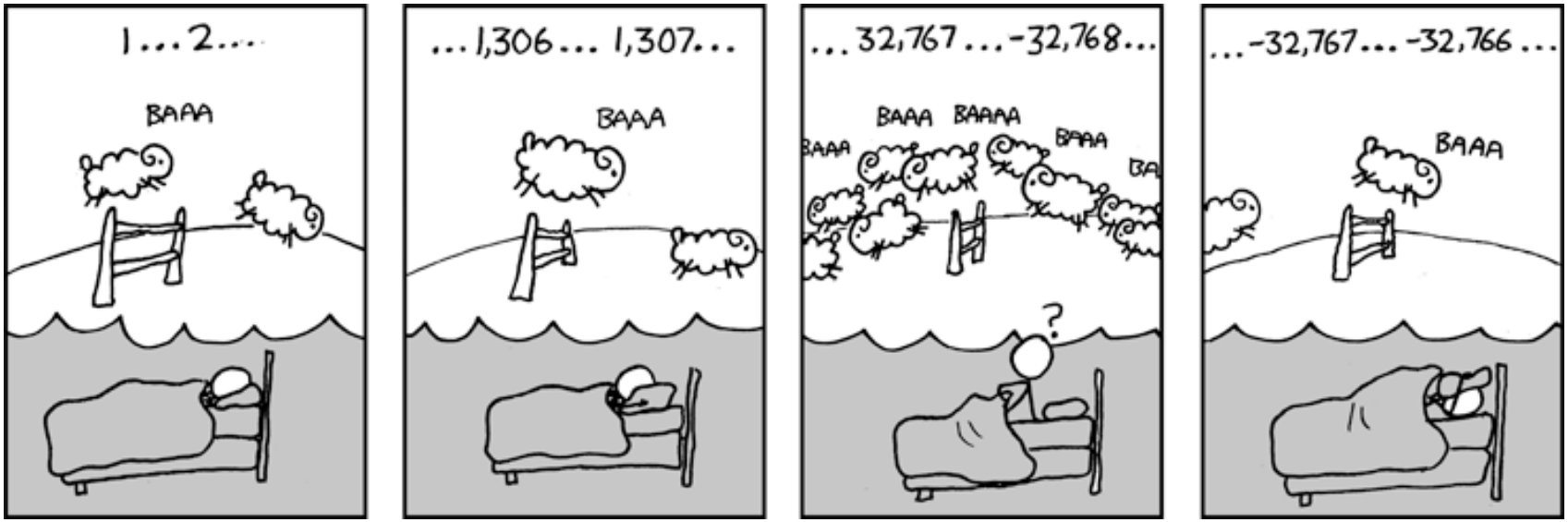
Universität Stuttgart
MINT-Kolleg Baden-Württemberg

Variablen, Wertebereiche

Vorkurs Informatik

Universität Stuttgart
Karlsruher Institut für Technologie
F. Schweiner
MINT-KOLLEG BADEN-WÜRTTEMBERG

Getting started ...



©Randall Munroe, `http://xkcd.com/571/`

Outline

(nach Folien von L. Vettin, V. Weidler, W. Kessler)

Outline

Rechnen mit Java

- `+`, `-`, `*` funktionieren wie erwartet.
- Bei der Auswertung wird die Rangfolge der Operatoren berücksichtigt (Punkt vor Strich, Klammern, . . .)
- Kompliziertere Operationen (Wurzelziehen, etc.) der Mathematik befinden sich in der von Java zur Verfügung gestellten Sammlung `Math`*:

```
Math.pow(3,2); // 3 hoch 2 (engl. to the power of)
Math.sqrt(16); // Wurzel aus 16 (engl. square root)
Math.sin(Math.PI); // Sinus von Pi
```

*`http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html`

Division in Java

Das Ergebnis einer Division hängt von den Datentypen der beteiligten Variablen/Zahlen ab:

- Dezimaldivision (mindestens ein `double`):

```
double x = 9.0/4; // = 2.25
```

- Ganzzahldivision (zwei `int`):

```
int x = 9/4; // = 2
```

- Rest der Ganzzahldivision (Modulo):

```
int x = 9 % 4; // = 1
```

Stolperfallen

- Wo findet Ganzzahldivision statt?

```
int x = 4; int y = 7; double z = 3.0;
int a; double b;

a = 4/6;
b = 4/6.0;
a = x/5;
b = y/x;
b = x/z;
b = x/(1.0*y);
a = x/z;
```

Stolperfallen

- Wo findet Ganzzahldivision statt?

```
int x = 4; int y = 7; double z = 3.0;
int a; double b;

a = 4/6; // ja
b = 4/6.0; // nein
a = x/5; // ja
b = y/x; // ja - und erst dann wird das Ergebnis
         // in ein double umgewandelt!
b = x/z; // nein, da eine double-Variable beteiligt
b = x/(1.0*y); // nein
a = x/z; // nein - aber dann kommt es zu einem Fehler, weil
         // ein double in ein int gespeichert werden soll
```

- Seien Sie bei Division IMMER wachsam!

Division in Java

- Der Divisionsrest ist bei negativen Zahlen ebenfalls negativ!

```
int x = 9 % 4; // = 1
int x = -9 % 4; // = -1
```

- Gerade Zahlen haben nach dem Teilen durch 2 keinen Rest

```
int x = 8 % 2; // = 0
```

- Ungerade Zahlen haben nach dem Teilen durch 2 Rest 1 oder -1

```
int x = 9 % 2; // = 1
int x = -9 % 2; // = -1
```

Outline

Wertebereiche von Datentypen

- Datentypen für Zahlen haben einen Wertebereich.
- Der Wertebereich hängt mit der Art der Speicherung zusammen (siehe Einheit “Darstellung von Informationen”).
- Beispiele von Wertebereich für ganze Zahlen:
  `byte`: $-2^7 \ldots 2^7 - 1$
  `int`: $-2^{31} \ldots 2^{31} - 1$
  `long`: $-2^{63} \ldots 2^{63} - 1$
- Beispiele von Wertebereich für Dezimalzahlen (je $+$ und -):
  `float`: $1.40239846 \cdot 10^{-45} \ldots 3.40282347 \cdot 10^{38}$
  `double`: $4.9406564584124654 \cdot 10^{-324} \ldots 1.7976931348623157 \cdot 10^{308}$

Überlauf bei ganzen Zahlen

- Ganze Zahlen können "überlaufen", wenn der Zahlenwert den Wertebereich der Variablen übersteigt:

```
int x = Integer.MAX_VALUE;
System.out.println(x); // 2147483647
x=x+1;
System.out.println(x); // -2147483648
```

- Es gibt *keinen* Fehler und *keine* Warnung!
- Es wird wird einfach von vorne weiter gerechnet!

Überlauf bei Dezimalzahlen

- Dezimalzahlen können "überlaufen" und auch "unterlaufen" (zu klein, um dargestellt zu werden).

```
double x = Double.MAX_VALUE;
System.out.println(x); // 1.79769313486231570E308
double y = x+x;
System.out.println(y); // Infinity
double z = y-x;
System.out.println(z); // Infinity
```

- Es gibt *keinen* Fehler und *keine* Warnung!
- Es wird wird einfach der Wert $+\infty/-\infty$ (bei Überlauf) oder 0 (bei Unterlauf) eingesetzt und weitergerechnet.

Einschub: Überlauf in der realen Welt

- Die europäische Rakete Ariane 5 wurde beim Erstflug am 4. Juni 1996 zerstört.
- Grund war die Umwandlung einer 64-Bit-Gleitkommazahl in eine vorzeichenbehaftete 16-Bit-Ganzzahl in der Steuersoftware.
- Überlauf führte zu starker Neigung der Rakete nach 37 Sekunden Flug und Zerstörung nach weiteren 3 Sekunden.
- Verlust 290 Millionen Euro, keine Personenschäden.
- Steuersoftware war von Ariane 4 übernommen und hatte dort funktioniert – aber in Ariane 5 traten größere Werte auf!

Siehe: `http://www.deutschlandfunk.de/der-absturz-der-ariane-5.676.de.html?dram:article_id=25637`

Rundungsfehler

- Es kommt aufgrund der Verwendung des Binärsystems und der limitierten Speichergenauigkeit zu Rundungsfehlern:

```
System.out.println( 0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1 );
// = 0.7999999999999999
System.out.println(2.05 - 0.05); // = 1.9999999999999998
```

- Das exakte Ergebnis ($\frac{1}{10_{10}} = \frac{1}{1010_2} = 0,0\overline{0011}_2$) wird durch eine Rundung auf eine der beiden benachbarten darstellbaren Binärzahlen abgebildet.
- Bei vielen Operationen mit kleinen Zahlen können in ungünstigen Fällen Rundungsfehler sehr stark anwachsen.

Outline

Dateinamen

```
public class Test {
    public static void main (String[] args) {
        System.out.println("This is a test.");
    }
}
```

- Dateien mit Javacode haben die Endung `.java`
- Die Dateien liegen bei Eclipse im Eclipse-Workspace
- Der Dateiname muss IMMER dem entsprechen, was hinter `public class` steht, d. h. hier `Test.java`.
- Dateinamen müssen mit Großbuchstaben beginnen, erlaubt sind Groß-, Kleinbuchstaben und Ziffern, aber keine Leerzeichen.
- Programm compilieren: `javac <Dateiname>.java`
- Programm ausführen: `java <Dateiname>`

Regeln für Variablennamen

- Variablennamen bestehen aus Groß- und Kleinbuchstaben, Ziffern und einigen erlaubten Sonderzeichen (z. B. _).
- Sie dürfen nicht mit einer Ziffer beginnen.
- Sie dürfen keine Leerzeichen enthalten.
- Groß- und Kleinschreibung wird unterschieden.
- Reservierte Namen (Schlüsselwörter wie `int`, `double`, ...) dürfen nicht verwendet werden.

Empfehlungen für Variablennamen

- Nur A-Z, a-z, 0-9 und _ verwenden.
- Aussagekräftige Variablennamen wählen, z. B. `sum` oder `name` statt `a` oder `x23`.
- Konvention: Variablennamen beginnen mit Kleinbuchstaben.
- Konvention: Variablennamen verwenden lowerCamelCase, d. h. an das erste Wort werden alle anderen Wörter direkt angehängt, je beginnend mit Großbuchstaben, z. B. `sumOfAllNumbers`, `firstNmame`.
- Konvention: Variablennamen in Englisch!

Empfehlungen für Kommentare

- Kommentare sind Erklärungen für den Programmierer.
- Kommentare sollen nicht nur beschreiben was passiert, sondern auch *warum* gerade das passiert.
- Beispiel für schlechte Kommentare:

```
t = t + 1;  // erhoehe t um eins
s = s + v;  // zaehle v zu s dazu
d = s / t; // teile s durch t
```

Empfehlungen für Kommentare

- Kommentare sind Erklärungen für den Programmierer.
- Kommentare sollen nicht nur beschreiben was passiert, sondern auch *warum* gerade das passiert.
- Beispiel für schlechte Kommentare:

```
t = t + 1;  // erhoehe t um eins
s = s + v;  // zaehle v zu s dazu
d = s / t; // teile s durch t
```

- Beispiel für gute Kommentare:

```
t = t + 1;  // erhoehe die Anzahl Klausurteilnehmer
s = s + v;  // addiere die Note zur Summe aller Noten
d = s / t; // berechne den neuen Durchschnitt
```

Empfehlungen für Kommentare

- Am besten Variablennamen so wählen, dass Kommentare überflüssig sind!

```
t = t + 1;  // erhoehe die Anzahl Klausurteilnehmer
s = s + v;  // addiere die Note zur Summe aller Noten
d = s / t; // berechne den neuen Durchschnitt
```

- besser

```
numberOfParticipants = numberOfParticipants + 1;
sumOfGrades = sumOfGrades + grade;
averageGrade = sumOfGrades / numberOfParticipants;
```

Empfehlungen für Einrückungen und Leerzeichen

- Einrückungen und Leerzeichen spielen für Java keine Rolle.
- Sinnvolle Verwendung von Einrückungen und Leerzeichen erhöht die Übersichtlichkeit und hilft Fehler zu vermeiden.
- Beispiel für schlechte Einrückungen und Leerzeichen:

```
int a = 1  ; int b= 2
        ; b = a+ b; int
          f = c
* 2;
```

Empfehlungen für Einrückungen und Leerzeichen

- Einrückungen und Leerzeichen spielen für Java keine Rolle.
- Sinnvolle Verwendung von Einrückungen und Leerzeichen erhöht die Übersichtlichkeit und hilft Fehler zu vermeiden.
- Beispiel für schlechte Einrückungen und Leerzeichen:

```
int a = 1  ; int b= 2
        ; b = a+ b; int
          f = c
* 2;
```

- Beispiel für gute Einrückungen und Leerzeichen:

```
int a = 1;
int b = 2;
b = a + b;
int f = c * 2;
```

Outline

Zusammenfassung: Variablen

- In Java lässt sich wie gewohnt mit $+$, $-$, $*$ rechen.
- Das Ergebnis einer Division hängt von den Datentypen der beteiligten Variablen/Zahlen ab
- Kompliziertere Operationen haben einen eigenen Namen, zum Beispiel `Math.sqrt()` oder `Math.pow()`
- Datentypen für Zahlen haben einen beschränkten Wertebereich. Vorsicht bei "**Überlauf**", "**Unterlauf**" und **Rundungsfehlern**.
- Vermeiden Sie Fehler durch die Verwendung von
  - **sinnvollen** Variablennamen.
  - **Kommentaren** mit Erläuterungen.
  - übersichtlichen **Einrückungen** und **Leerzeichen**

Universität Stuttgart
MINT-Kolleg Baden-Württemberg

Vielen Dank!

Frank Schweiner

E-Mail frank.schweiner@mint-kolleg.de
Telefon +49 (0) 711 685-84326
Fax —

Universität Stuttgart
MINT-Kolleg Baden-Württemberg

Azenbergstr. 12
70174 Stuttgart