Karlsruher Institut für Technologie

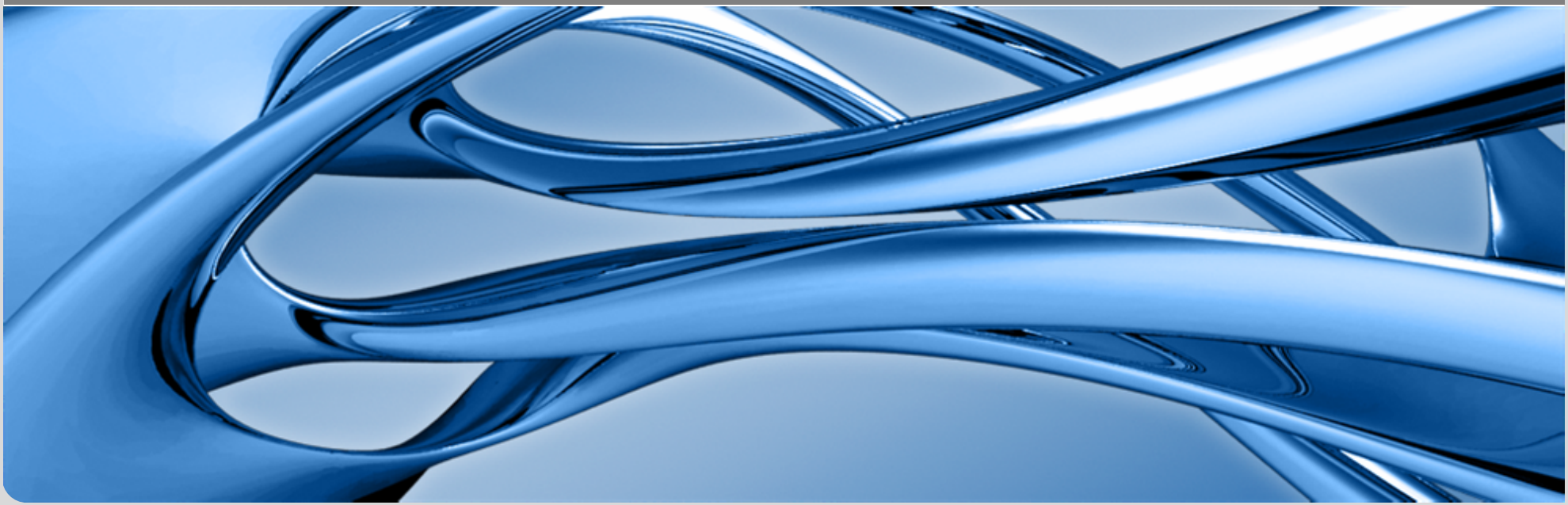
SDQ Software Design and Quality

Tutorium 2

Datentypen, Operatoren und Variablen

Christian Zielke | 6. November 2018

CHAIR FOR SOFTWARE DESIGN AND QUALITY



Gliederung

1. Allgemeines
   - Übungsblätter
2. Wiederholung
3. Datentypen
   - Primitive Datentypen
   - enum

Gliederung

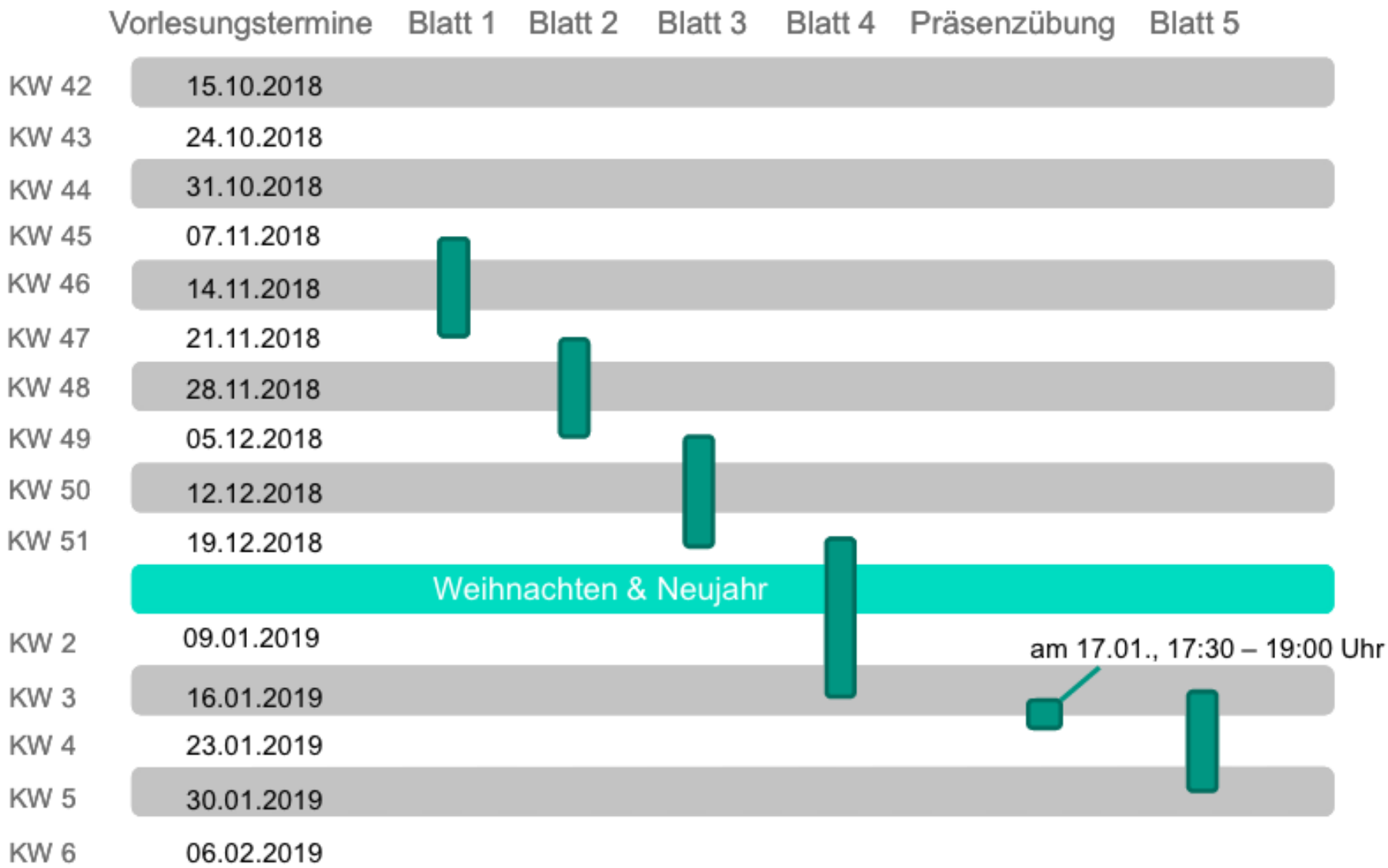
4 Operationen und Präzedenz
- Operationen und Präzedenz
- Arithmetische Operatoren
- Vergleichsoperatoren
- Der “==” Operator
- Operatoren auf ganze Zahlen
- Operatoren auf Wahrheitswerte
- Operatoren auf Wahrheitswerte und Ganzzahlen
- Verkürzende Operatoren

Gliederung

5 Variablen und Objekte
- Variablen
- Konstanten
- Objekte

6 Übungsaufgaben
- Mathe Aufgaben
- Fußball
- Hörsaal

Übungsblätter Übersicht



Was kennen/können wir schon?

- .java Dateien erstellen und kompilieren
- Klassen und Objekte
- main Methode
- Konsolenausgabe
- Variablen und primitive Datentypen
- Grundrechenarten

Primitive Datentypen

Datentyp	Erklärung	Größe	Wertebereich
byte	kleine Ganzzahl	8 bit	-128 bis + 127
short	Ganzzahl	16 bit	-32768 bis + 32767
int	Standard Ganzzahl	32 bit	-2^{31} bis $+2^{31}-1$
long	Große Ganzzahl	64 bit	-2^{63} bis $+2^{63}-1$
float	Ungenaue Fließkommazahl	32 bit	+/-$1,4E-45$ bis +/-$3,4E+38$
double	Genauere Fließkommazahl	64 bit	+/-$4,9E-324$ bis +/-$1,7E+308$
boolean	Wahrheitswert	1 bit	true, false
char	Unicode Zeichen	16 bit	

Umwandeln von Datentypen

- Problemlose Umwandlung in größere Datentypen

```
1       byte b = 42;
2       int i = b;
```

- Umwandlung in kleinere Datentypen nur mit explizitem cast

```
1       int i = 42;
2       byte b = (byte) i;
```

- Achtung: Beim cast kann sich die Zahl Ändern

```
1       int i = 128;
2       byte b = (byte) i; // b = -128
3       double d = 10.5;
4       int x = (int) d; // x = 10
```

Spezialfälle bei Gleitkommazahlen

Eine Zahl außerhalb des Wertebereichs führt zum **Überlauf**:

```
1   double d = -1E308 - 1E308; //Ausgabe: -Infinity
2   double e =  1E308 + 1E308; //Ausgabe:  Infinity
```

Eine Zahl zu nahe an der 0 führt zum **Unterlauf**.

```
1   double f = 6E-324 - 5E-324; //Ausgabe: 0.0
```

Weitere Besonderheiten:

- Teilen durch 0 ergibt `Infinity`
- `0.0/0.0` ergibt `NaN` (Not a Number)

enum

Der enum-Datentyp speichert **Aufzählungen**:

- enum <Name> {WERT1, WERT2, ...}
- Beispiel:

```
1   enum Weekday { MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY };
```

- ein enum kann/sollte in einer eigenen Datei angelegt sein
- es müssen alle Werte vorher bekannt sein
- Eine enum-Variable kann einen der aufgezählten Werte annehmen:

```
1   Weekday day = Weekday.THURSDAY;
```

Präzedenz

- Die **Präzedenz** gibt an, wie stark der Operator zwei verknüpfte Elemente bindet
- Beispiel: Punkt vor Strich (hier hat * eine höhere Präzedenz als +)
- Wird in Zahlen angegeben:
  1 - Hohe Präzendenz, 10 - Niedrige Präzedenz

Arithmetische Operatoren

Arithmetische Operatoren kann man auf alles anwenden, mit dem man rechnen kann:

Liste

Operator	Bezeichung	Präzedenz
+x, -x	Vorzeichen (unäres Plus/Minus)	1
x*y, x/y, x%y	Multiplikation, Division, Modulo	2
x+y, x-y	Addition, Subtraktion	3

- Division von Ganzzahlen werden nach dem Komma abgeschnitten
- Division durch 0 erzeugt meist einen Laufzeitfehler

Vergleichsoperatoren

Will man zwei Werte miteinander vergleichen, benutzt man Vergleichsoperatoren:

Liste

Operator	Bezeichung	Präzedenz
x<y, x<=y, x>y, x>=y	Größenvergleiche	5
x==y, x!=y	(Un-)Gleichheit	6

Der “==” Operator und seine Tücken

“==” bei String

- **Falsch:**

```
1     boolean match = ("Hallo!" == "Hallo!");
```

- **Richtig:**

```
1     boolean match = "Hallo!".equals("Hallo!");
```

Der “==” Operator und seine Tücken

“==” bei Gleitkommazahlen

- **Falsch:**

```
1    //Ergebnis: 0.04999997 -> false
2    boolean match = (0.03f + 0.02f == 0.05f);
```

- **Richtig:**

```
1    boolean match =
2    Math.abs((0.03f + 0.02f) - 0.05f)) < 1E-6f;
```

Operatoren auf ganze Zahlen

Liste

Operator	Bezeichung	Präzedenz
$\sim$x	Bitweises Komplement (NOT)	1
x << y	Linksshift	4
x >> y	Rechtshift	4
x >>> y	Rechtshift (Vorzeichen ignoriert)	4

Operatoren auf Wahrheitswerte

Liste

Operator	Bezeichung	Präzedenz
!x	NOT	1
x && y	Sequentielles AND	10
x || y	Sequentielles OR	11

Kurzauswertung

x && y sowie x || y werden kurz ausgewertet:

- x && y: Ist x `false` → Ganzer Ausdruck ist `false`
  → Auswertung wird abgebrochen
- x || y: Ist x `true` → Ganzer Ausdruck ist `true`
  → Auswertung wird abgebrochen

Operatoren auf Wahrheitswerte und Ganzzahlen

Liste

Operator	Bezeichung	Präzedenz
x & y	Bitweises UND	7
x ^ y	Bitweises XOR	8
x | y	Bitweises OR	9

Verkürzende Operatoren

Liste

Operator	Beschreibung	Ersetzt
x++, x- -	Inkrement, Dekrement	x = x + 1, x = x - 1
a = x++, a = y- -	Postinkrement, -dekrement	a = x; x = x + 1
a = ++x, a = - -y	Präinkrement, -dekrement	x = x + 1; a = x
x += 5, x *= 5, ...	Schnellere Berechnung	x = x + 5, x = x * 5, ...

Kurze Fragerunde

```
1       int i = 10;
2       int j = i++;
3       int c = --j;
4       i = ++c;
```

- Welchen Wert hat i?
  - 10

```
1       boolean a = true;
2       boolean b = false;
3       boolean c = true;
4       boolean d;
5       d = !d;
6       d = !a;
7       d = a && b;
8       d = !a || !c;
```

- Welchen Wert hat d?
  - Keinen, Compilerfehler

Kurze Fragerunde

```
1       int i = 6 ^ 4;
```

- Welchen Wert hat i?
  - 2

```
1       boolean q = true;
2       boolean w = false;
3       boolean t = true;
4       boolean b;
5       b = !( q || w ) && t || !w ^!q;
```

- Welchen Wert hat b?
  - true

Kurze Fragerunde

```
1       int i = 10 << 1;
```

- Welchen Wert hat i?
  - 20

```
1       boolean a = false;
2       boolean b;
3       b = (5 % 2 < 2) & (!!a | (5 >> 2 > 2 >> 3));
```

- Welchen Wert hat b?
  - true

Variablen

- Syntax:

```
1     Datentyp <name> = Initialwert;
2     Datentyp <name1>, <name2>;
3     Datentyp <name3>;
4     <name3> = Wert;
```

- Wertzuweisung mittels **=**
- Variablen stehen für Speicheradressen.

Konstanten

- Syntax:

```
1      final Datentyp <name> = Initialwert;
```

- Konstantendeklaration mit Schlüsselwort **final**
- Wert kann nur einmal zugewiesen werden, sonst Compilerfehler.

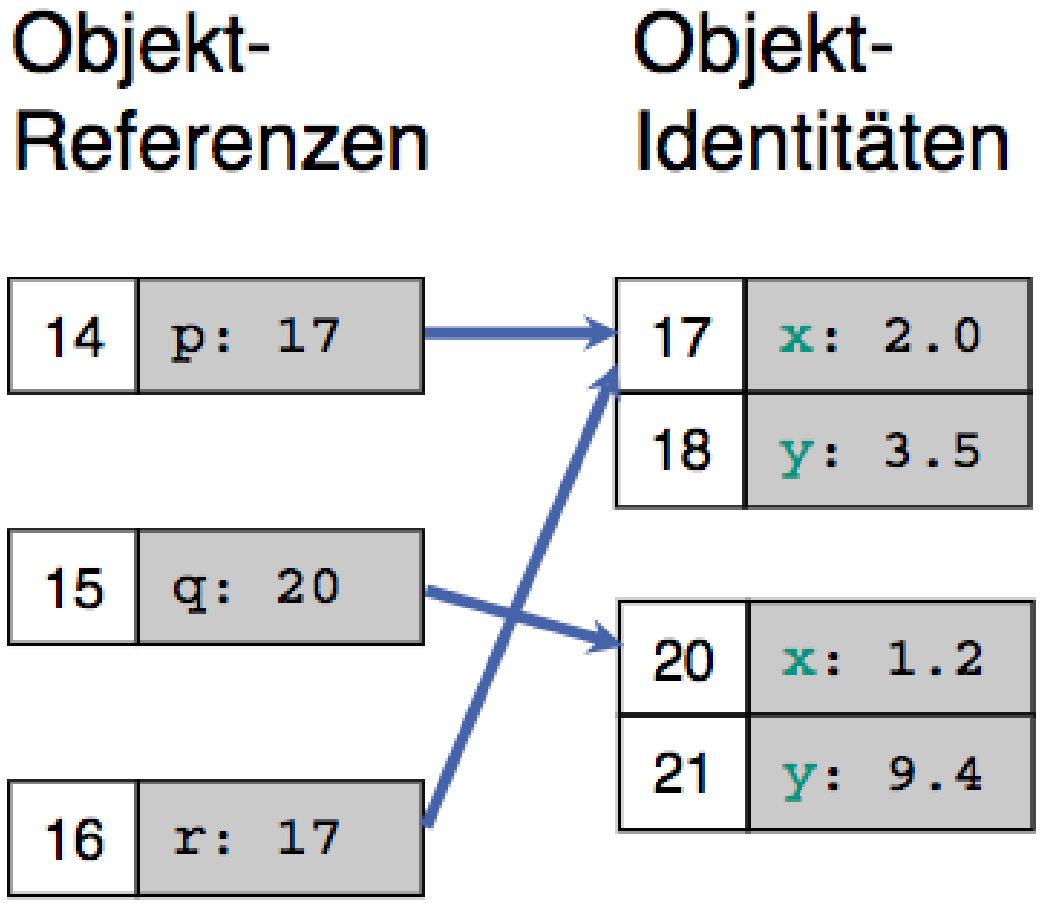
Objekte

- Syntax:

```
1    Klassenname <name> = new Klassenname();
2    Klassenname <name1>, <name2>;
3    KlassennameX <name3>;
4    <name3> = new KlassennameX();
```

  - Datentypen sind hier Klassen.
  - Variablen zeigen auf erstes Objektattribut.
  - Mehrere Variablen können auf gleiches Objekt zeigen.
  - **null** bedeutet "kein Objekt".

Objekte


Objekt-
Referenzen
Objekt-
Identitäten
14
p: 17
15
q: 20
16
r: 17
17
x: 2.0
18
y: 3.5
20
x: 1.2
21
y: 9.4

Aufgabe: Kugel

- Schreibe ein Java Programm, das folgende Größen einer Kugel mit Radius r berechnet:
  - Durchmesser
  - Umfang
  - Oberfläche
  - Volumen
- Zur Erinnerung:
  - $V = \frac{4}{3} * \pi * r^3$
  - $O = 4 * \pi * r^2$

```
Durchmesser: 12.0
Umfang: 37.69911184307752
Oberfläche: 452.3893421169302
Volumen: 904.7786842338603
```

Aufgabe: Spieler

- Modelliere einen Fußballspieler mit den Attributen:
  - Name
  - Alter
  - Position
- mögliche Positionen sind:
  - Torwart (goalkeeper)
  - Abwehr (defense)
  - Mittelfeld (midfield)
  - Stürmer (striker)

Aufgabe: Hörsaal Modellierung

Modelliere mit Klassen und Objekten einen Hörsaal. Gehe dazu wie folgt vor:

1. Überlege dir welche Bestandteile ein Hörsaal hat. (Stuhl, Tafel, ...)
2. Lege dann die Klassen mit ihren Attributen an.

Verwende möglichst viele der dir bekannten Datentypen. (z.B. enum)