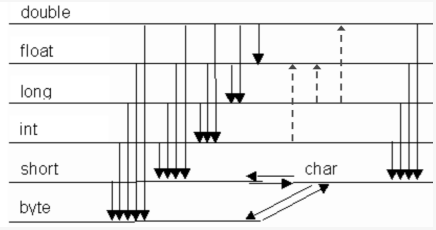
Typ Konversion



schwarze Pfeile: expliziet
rote Pfeile: impliziet, evt Genauigkeits-verlust
alles andere impliziet

Switch-Statement

```
switch (Ausdruck) {
case Wert1:
      Anw eis ungen;
      break;
case Wert2:
      Anw eis ungen;
      break;
default:
      Anw eis ungen;
}
```

Wenn Ausdruck Wert1 entspricht, wird Anweisung 1 ausgeführt...

Typ Polymorphismus

```
car extends vehicle
@Override
drive(){}
Vehicle v = new Car();
//calls drive of Car not Vehicle
```

Typ Polymorphismus (cont)

```
> v.drive();
```

Klasse hat eigenen Typ plus alle Typen der Superklassen.
verschiedene "Brillen"
Dynamic Dispatch = dynamischer Typ zur Laufzeit entscheidet über aufgerufene Methoden.

Rekursion

```
public boolean groupSum(int
start, int[] nums, int target) {
   if (start >= nums.l ength)
return (target == 0);
   if (group Sum (start + 1,
nums, target - nums[s tart]))
return true;
   if (group Sum (start + 1,
nums, target)) return true;
    return false;}
Collec tio n<S tri ng> validP -
are nth ese s(int nofPairs){
    Col lec tio n<S tri ng> list
= new ArrayL ist <>();
   if (nofPairs == 0) {list.a -
dd ("");} else {
       for (int k = 0; k <
nofPairs; k++) {
            Col lec tio n<S -
tri ng> infixes = validP are -
nth ese s(k);
            Col lec tio n<S -
tri ng> suffixes = validP are -
nth ese s(n ofPairs - k - 1);
           for (String infix :
infixes) {
               for (String
suffix : suffixes) {
                    lis t.a -
dd( " (" + infix + " )" +
suffix );}}}}
    return list;}
```

Generics

allg. Form

class Pair<T,U>

TypeBound (Bedingungen für Type)

class Node<T extends Comparable<T>>
Node<Person>//ok Node<Student>//Error

Multiple TypeBounds (mit & anhängen)

class Node<T extends Person & Compar-able<Person>>

generische Methode

public <T extends Person> T set (T element){...}
<T>: Bedingungen für Input
T : Rückgabetyp

Wildcard-Typ (Nutzen: Typ**argument** nicht relevant)

Node<?> undef --> kein Lesen & schreiben ausser: Object o = undef.getValue();

Lambdas

```
(p1, p2) -> p1.get Age() - p2.get
p -> p.getAge() >= 18
people.so rt((p1, p2) -> p1.get Na
people.so rt( Com par ato r.c omp
rs on: :ge tFi rst Name));
Utils.r em ove All (pe ople, perso
people.so rt( Com par ato r.c omp
```

Syntax: (Parameter) -> {Body}

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

StreamAPI

```
people
.stream()
.filter(p -> p.getAge() >= 18)
.map(p -> p.getL ast Name())
.sorted()
.forEa ch( Sys tem.ou t:: pri -
ntln);
people.st rea m().ma pTo Int(p -
> p.getA ge())
.avera ge( ).i fPr ese nt( Sys -
tem.ou t:: pri ntln);
Map<St ring, Intege r> totalA -
geB yCity =
    peo ple.st ream()
   .co llect(
          Col lec tor s.g rou -
pin gBy (Pe rso n:: get City,
          Col lec tor s.s umm -
ing Int (Pe rso n:: get Age)));
```

endliche Quelle: `IntStr eam.ra nge(1, 100)`

unendl. Quelle: `Random random = new R andom(); Stream.ge ner ate (ra n do m:: nex tIn t).l im it(100)`

FileReader/Writer

```
private static void
reverteText() throws
FileNotFoundException,
IOException{
   try (FileR eader reader = new
FileRe ade r("i npu t.t xt");
          Fil eWriter writer =
new FileWr ite r("o utp ur.t -
xt ")){
           int value =
reader.re ad();
            String text = " ";
            while (value >=0){
                 text = (char) -
value + text;
                  value =
reader.re ad();}
```

FileReader/Writer (cont)

```
>      writer.write(text);}}
```

Sichtbarkeit

public	alle Klassen
protected	Package und Sub-Klassen
private	nur innerhalb Klasse
(keines)	innerhalb Package

Datentypen

byte	8 bit (2^7 bis 2^7-1)
short	16 bit
int	32 bit
long	64 bit (1L)
float	32 bit (0.1f)
double	64 bit

Operator-Prio

+,-,++,--,! (unär)

*, /, %

+, - (binär)

<, <=, >, >=

==, !=

&&

||

Rundungsfehler

0.1+0.1+0.1 != 0.3

Problem: `x == y` double/floa t

Lösung `Math.abs(x - y) < 1e-6`

Integer Literal

binär	0b10 = 2
oktal	010 = 8
hex	0x10 = 16
opt.	1000 = 1_000

Arithmetik

```
1 / 0 --> ArithmeticException: /
by zero
1 / 0.0 --> Infinity
-1.0 / 0 --> -Infinity
0 / 0.0 --> NaN
```

Arithmetik Overloading

```
int operate(int x, int y) { ...
}
double operat e(int x, double y)
{ ... }
double operat e(d ouble x,
double y) { ... }
```

operate(1, 2); --> meth 1
operate(1.0, 2); --> meth 3
operate(1, 2.0); --> meth 2

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Overloading

```
class Graphic {
  void moveTo(Graphic other)
    // Method 1
  }
}

Graphic g = new Circle();
Circle c = new Circle();
```

```
class Circle extends Graphic {
  void moveTo(Graphic other) {
    // Method 2
  }

  void moveTo(Circle other) {
    // Method 3
  }
}
```

g.moveTo(c)	Compiler: nur Methode 1 → Laufzeit: Overriding durch Methode 2
c.moveTo(g)	Compiler: Overloading, nur Methode 2 passt für Argument g
c.moveTo(c)	Compiler: Overloading, Methode 3 spezifischer
g.moveTo(g)	Compiler: nur Methode 1 → Laufzeit: Overriding durch Methode 2

Bedingungsoperator

```
max = left > right ? left :
right;
```

wenn left>right wird max = left, sonst max=right

Package prio

own class (inc. nested class)

single type imports -> import p2.A;

class in same package

import on demand -> import p2.*

Enum

```
public enum Color {
      BLU E(1), RED(2);
      private final int code;
      private Color(int code) {
            thi s.code = code;}
      public int getCol orV -
alue() {
            return code;}}
```

Methodenreferenz

```
people.so rt( thi s:: com par eByAge)
int compar eBy Age (Person p1, Person p2){return p1.age - p2.age;}
```

```
people.so rt( Sor ter ::c omp are ByAge)
static int compar eBy Age (Person p1, Person p2){return p1.age - p2.ag
e;}
```

```
Sorter sorter = new Sorter();
people.sort(sorter::compareByAge);
```

Serialisieren

```
OuputStream fos = new
FileOutputStream("serial.bin");
try (Objec tOu tpu tStream
stream = new Object Out put -
Str eam (fos)){
      str eam.wr ite Obj ect -
(pe rson);
}
```

needs Serializable interface
serialisiert alle direkt & inderekt referenz. Obj

Input/Output

```
try (BufferedInputStream
inputBuffer = new bis(new
FileInputStream(pathSource));
Buffer edO utp utS tream outputB
= new bos(new FileOu tpu tSt -
rea m(p ath Tar get))) {
      byte[] byteBuffer = new
byte[4 096];
     int value = inputB uff -
er.r ea d(b yte Buf fer);
      while (value >= 0) {
            out put B.w rit -
e(b yte Buffer, 0, value);
            value = inputB uff -
er.r ea d(b yte Buf fer);}}
```

Input/Output (cont)

```
> catch (FileNotFoundException e) {
    System.out.println("File not found:"+e);
} catch (IOException e) {
    System.out.println("reading Error:"+e);}
```

String Pooling

```
String a = "OO", b = "Prog", c =
"OOProg";
//true
a == " OO"; c == " OO" + " Pro -
g"; (a+b).e qu als(c);
//false
a + b == c;
```

String Literals gepoolt. True Fälle --> es werden keine neuen Objekte erstellt
Integer-Pooling -128 bis 127

Collections

Array	`int[]`
List	`List<T ); `
Set	`Set<T>`
Map	`Map<U, ;`

```
Iterat or< T> it = a.iter ator();
.}
```

```
Collec tio ns.s or t(O bje
ct o)
```

needs C

Sponsored by CrosswordCheats.com
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Collections (cont)

```
for (Map.E ntr y<C har acter, Double>-entry: map.entry, Set())
```

LIST/SET: add(), remove()
LIST: get(index)
MAP: put(key, value), remove(key),getKey()

Interface vs. Abstract Class

Methoden	implizit public, abstract	evt abstract, nie private
Variablen	KONSTANTEN (impl. public static final)	normal, Constructor mit super()
Vererbung	`implements a,b`	`extends a`

Abstract Method (in abs. cl.):
`public abstract void report();`
kein Rumpf{}
in interface: `void report();`
I. Methode mit Rumpf: `default void report (){...}`
`interface I1 extends I2,I3`
`class C2 extends C1 implements I1, I2`
wenn I1&I2 nun gleiche Methoden haben (signatur) --> C2 muss ine der beiden überschreiben. Zugriff möglich mit `I1.super.`*`methode`*`(); I2.super.`*`methode`*`();`

Unchecked vs Checked Exceptions

Unchecked	Checked
Error	Try/catch oder Methodenkopf
RunTime-Exceptions	mögl. catch-Block: `e.prin tSt ack Trace()`

Unchecked vs Checked Exceptions (cont)

-> RunTime, NullPointer, IllegalArgument, IndexOutOfBounds

eigene exception
```
class myE extends Exception { myE() {}
MyE(String message) {
super( mes sage);} }
```

Junit

```
@Before
public void setUp() {...}
@Test (timeout = 500, expected = Except ion.class)
public void testGo odN ame(){
    ass ert Equ als (ex pected, actual);
    ass ert Tru e(c ond iti - on);}
```

@After -> tearDown()
EdgeCases testen (Grenzwerte, null,...)

equals

```
@Override
public boolean equals (Object obj) {
      if(this == obj) {return true;}
     if (obj == null) {return false;}
     if (getCl ass() != obj.ge - tCl ass()) {
            return false;}
      Student other = (Stude - nt)obj;
      return regNumber == other.r eg Num ber;}}
```

HashCode überschreiben!
` x.equals(y) → x.hashCode() == y.hash-Code()
Grund: inkonsistenz bei Hashing. obj wird nicht gefunden, obwohl in Hash-Tabelle, x.contains(y) --> nutzt hashCode() & equals()

HashCode()

```
@Override
public int hashCode() {
    return 31* firstN ame.ha - shC ode()
     + 31 * lastNa me.h as hCo - de();
}
```

Funktionsschnittstelle

```
@FunctionalInterface
interface Compar ato r<T> {
   int compare(T first, T second);
}
```

genau eine Methode --> java.util.function

RegEx

Pattern pattern = Pattern.compile("*reg*")	
vor ? optionaler Teil	([0]?[0-9]|2[0-3])
(){}*+?| als norm text	* \(\\ ...
Gruppennamen	(?<NAME>)

Matcher matcher = pattern.matcher(string);
if (matcher.matches()){String one = matche-r.group("NAME")}

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com