3. Java

Java

Java = C + Klassen (?)

(Fast) wie in C (I)

Primitive Datentypen:

- `boolean` (`true`, `false`)
- `char` (`'A'`, `'\n'`, `'\\'`, `'\''`, `'\101'`)
- Numerisch:
  - Ganzzahlig (mit Vorzeichen):
    - `byte` [8bit]
    - `short` [16bit]
    - `int` [32bit] (`29`, `035`, `0X1d`)
    - `long` [64bit] (`29L`, `035l`, `0x1DL`)
  - Fließkomma:
    - `float` [32bit] (`18.F`, `1.8e1F`)
    - `double` [64bit] (`18.`, `1.8e1`, `.18E2`)

(Fast) wie in C (II)

Variable:

```
int i, j, k;
  int diff = max - min;
double[] coordinates;
Counter counter;
```

Lokale Variablen sind nicht automatisch initialisiert.

Konstanten:

```
final int DIMENSION = 3;
```

(Fast) wie in C (III)

Operatoren:

arithmetisch:
`+ - * / %`

inkr./dekr.:
`++ --` (Prä-/Postfix)

Vergleich:
`> >= < <= == !=`

(Fast) wie in C (IV)

Operatoren:

logisch:
`&& || !`

bitweise logisch:
`& | ^ << >> >>>`

if-then-else:
condition ? trueValue : falseValue

Zuweisung:
`= += -= *= /= %=`
`&= |= ^= <<= >>= >>>=`

Array-Zugriff:
`a[i]`

(Fast) wie in C (V)

Kontrollfluß:

`{...}`

`if-else`

`switch-case-default`

`for`

`while`, `do-while`

`return`

(Fast) wie in C (VI)

Kontrollfluß:

`break`

`continue`

(möglicherweise mit Label)

```
search:
  while (...) {
    while (...) {
      ...
      if (...) {
        break search;
      }
    }
  }
```

(Fast) wie in C (VII)

Kommentare:

`/* Kommentar */`

speziell: `/** Dokumentation */`

`// Kommentar (bis Zeilenende)`

Geschachtelte Kommentare sind nicht möglich.

Objekte und Klassen (I)

Java-Programm:

Menge von Klassen.

Klasse:

Felder
(Daten der Objekte)

Methoden
(Funktionen, die die Daten manipulieren)

Objekte und Klassen (II)

```
class Counter {

  // FELDER
  public long value;

  // METHODEN
  public long getValue() {
    return value;
  }

  public void incr() {
    value++;
  }

}
```


Objekte und Klassen (III)

```
Counter ctr;
// Lokale Var.: nicht initialisiert!

ctr = new Counter();
// Felder in Objekten:
// initialisiert mit 0, '\000',
// false, null.

System.out.println(ctr.value);

ctr.incr();
ctr.incr();
System.out.println(ctr.value);
System.out.println(ctr.getValue());

ctr.value = 15;
```

Referenzen

Auf Objekte wird ausschließlich über Referenzen zugegriffen.

Java	C, C++
`Counter ctr;`	`Counter *ctr;`
`ctr.value`	`ctr->value`
	`(*ctr).value`
`null`	`NULL`

(aber nicht unbedingt so implementiert).

Konstruktoren (I)

```
class Counter {

  // FELDER
  long value;

  // KONSTRUKTOREN
  // beachte: kein Typ, kein "return",
  // Name gleich Klassenname.
  Counter() {
    value = 1;
  }

  Counter(long initialValue) {
    value = initialValue;
  }

  // METHODEN (wie bisher)
}
```


Konstruktoren (II)

```
class Counter {

  // FELDER
  long value;

  // KONSTRUKTOREN
  Counter(long initialValue) {
    value = initialValue;
  }

  Counter() {
    this(1);
  }

  // METHODEN (wie bisher)
}
```


this

```
class Counter {

  // FELDER
  long value;

  // KONSTRUKTOREN (wie bisher)
  // METHODEN (wie bisher)

  public Counter
           largerCtr(Counter ctr) {
    if (value < ctr.getValue()) {
      return ctr;
    } else {
      return this;
    }
  }

}
```

Klassenvariable/-methoden

```
class Counter {

  // FELDER
  long value;
  static long lastInitialValue;

  // KONSTRUKTOREN
  Counter(long initialValue) {
    value = initialValue;
    lastInitialValue = initialValue;
  }

  Counter() {
    this(lastInitialValue);
  }

  // METHODEN (wie bisher)
}
```


main

```
class TestCounter {
  public static void
              main(String[] args) {
    Counter c1, c2;
    c1 = new Counter(100);
    c2 = c1;
    c1.incr();
    System.out.println(c2.getValue());
  }
}
```

```
% javac ⟨dateiname⟩.java
% java TestCounter
```


Arrays

Arrays verhalten sich im wesentlichen wie andere Objekte.

```
int[] values;

values = new int[3];
values[0] = 8;
values[2] = 17;

int[] primes = { 2, 3, 5, 7 };

primes = new int[] { 2,3,5,7,11,13 };

int[][] pascalsTriangle = {
          { 1 },
          { 1, 1 },
          { 1, 2, 1 },
          { 1, 3, 3, 1 }
};
```


Packages

Package = Modul.

```
package counting;
// nur einmal pro Datei erlaubt!

package de.uni-sb.studcs.eva.counting;
```

Zugriff:

```
counting.Counter ctr;

import counting.Counter;
Counter ctr;

import counting.*;
Counter ctr;
```