Java für C++-Programmierer

HS 2024, Peter Bühler

Fabian Suter, 5. November 2024

0.0.1

1 Basics

1.1 Grundsätzlich

Diese Zusammenfassung dient zur Hilfe beim Programmieren mit Java anhand von C++-Vorwissen. Es gibt einige wichtige Unterschiede, **Java:**

- kennt keine Zeiger / Pointer
- kennt keine Funktionen (reine OO-Sprache)
- kennt kein Überladen von Operatoren
- kennt keine Destruktoren (Garbage-Collector gibt Speicher frei)
- kennt keine Mehrfachvererbung
- stellt eine umfangreiche Klassenbibliothek zur Verfügung
- -Programme sind plattformunabhängig
- ist weniger hardwarenah wie C++
- ...

1.2 Sourcecode Hello world Java

```
public class HelloWorld{
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

1.3 Java Runtime

Der Compiler erzeugt anders als in C++ **Bytecode**, welcher anschliessend auf einer Java Virtual MAchine (JVM) laufen kann. Dadurch wird Java plattformunabhängig, da jede Plattform eine JVM hat, sei es Windows, Android oder MacOS.

2 Datentypen

Ein Datentyp besteht aus *Werten* und *Operationen*. Java besitzt zwei generelle Datentypen, namentlich **Primitive Datentypen** und **Referenzdatentypen**.

2.1 Primitive Datentypen

Sie haben plattformunabhängig den gleichen Wertebereich, es gibt keine `unsigned`-Typen. Für bool'sche Werte gibt es den Datentyp `boolean`.

Typ	Beschr.	Beispiele
boolean	Bool'scher Wert	true, false
char	Textzeichen (UTF16)	'a', 'B', etc.
byte	Ganzzahl (8 Bit)	-128 bis 127
byte	Ganzzahl (16 Bit)	-32'768 bis 32'767
byte	Ganzzahl (32 Bit)	-2^{31} bis $2^{31}-1$
byte	Ganzzahl (64 Bit)	-2^{63} bis $2^{63}-1$, 1L
byte	Gleitkommazahl (32 Bit)	0.1f, 2e4f
byte	Gleitkommazahl (64 Bit)	0.1, 2e4

2.2 Referenzdatentypen

2.2.1 Arrays

Arrays speichern wie in C++ mehrere Elemente mit selbem Datentyp. Der Zugriff erfolgt über Index, die Elemente liegen nebeneinander im Speicher. Die Grösse des Arrays muss bei der Deklaration festgelegt werden und ist später nicht mehr änderbar.
Die Anzahl der Elemente kann in Java direkt mit `length` abgerufen werden, siehe auch **3.2**.
Falls der Index ungültig ist, wird eine **ArrayIndexOutOfBoundsException** geworfen.

2.3 Typumwandlung

Implizit:
Klein zu Gross,
kein Cast-Operator erf.

```
int a = 4711;

float b = a;

double c = b;
```

Explizit:
Gross zu Klein,
Cast-Operator erf.

```
int y = 0x11223344;
short x = (short) y; // x = 0x3344

double w = 4.7;
int v = (int) w; // v = 4
```

3 Code-Snippets

3.1 Fakultät

```
public class Factorial {
    public static void main(String[] args) {
        int n = 12;
        int p = 1;
        int i = 1;
        while (i <= n) {
            p = p * i;
            ++i;
        }
        System.out.println(p);
    }
}
```

3.2 Array-Loop

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

for(int i=0; i < array.length; i++){
    System.out.printf("Index %d : %d\n", i, array[i]);
}
```

```
// Index 0 : 1
// Index 1 : 2
// Index 2 : 3

int[][] m = new int[2][3];

// int[Zeile][Spalte]
// m.length    => Anzahl Zeilen
// m[0].length => Anzahl Spalten
```