Konventionen für Java-Quellcode (Formatierung)

Richtlinien und Konventionen zur äußeren Form und Vorschläge zur Namensgebung und Schreibweise in Java-Programmen wurden in dem Dokument Code Conventions for the Java™ Programming Language niedergelegt.

http://www.oracle.com/technetwork/java/codeconvtoc-136057.html

Äußere Gestaltung

Die Form eines Programms ist für den Java-Compiler in gewissen Grenzen gleichgültig. Er kennt nur wenige unverrückbare Regeln zur korrekten Schreibweise und zur Sprachsyntax (Ausdrucksweise und Kombination von Anweisungen). Der Parser orientiert sich *nicht* an Zeilenumbrüchen, die für die Darstellung in einem Quelltexteditor wichtig sind. Für den Compiler-Parser (Sprach-Analysator) sind Token wichtig, die Spracheinheiten voneinander unterscheiden:

- Leerzeichen zwischen Spracheinheiten (z.B. Typdeklaration und Variablennamen),
- Semikolon am Ende einer Anweisung und die
- Beachtung von Klammerpaaren.

Daneben prüft der Compiler natürlich die richtige Schreibweise von Spracheinheiten (Anweisungen, Schlüsselworte). Eine falsch geschriebene Anweisung kann nicht ausgeführt werden und produziert eine Fehlermeldung. Selbstdefinierte Daten und Datentypen (Objekte, Variablen, Methoden) dürfen diese reservierten Schlüsselworte nicht verwenden.

```
abstract    default     if          private      this
boolean     do          implements  protected    throw
break       double      import      public       throws
byte        else        instanceof  return       transient
case        extends     int         short        try
catch       final       interface   static       void
char        finally     long        strictfp     volatile
class       float       native      super        while
const       for         new         switch
continue    goto        package     synchronized
```

```
class myClass
{
    void myMethod( )
    {
    }
}
```

```
class myClass {
    void myMethod( ) {
    }
}
```

Darüber hinaus herrscht Freiheit in der äußeren Gestaltung des Quellcodes. Diese zeigt sich in der unterschiedlichsten Form von Programmen in allen Arten von Publikationen. Besonders über die Anordnung von geschweiften Klammern gibt es unterschiedliche Auffassungen. Jeder Programmierer soll selbst heraus finden, welche Art ihm besser erscheint - versetzt am Ende und Anfang der Zeile oder immer am Anfang einer Zeile. Eingehalten werden sollte immer ein Stil, der die Verschachtelung von Anweisungsblöcken (durch Einrückungen) deutlich zum Ausdruck bringt.

Anerkannte Richtlinien

In Softwarehäusern werden oft detaillierte Richtlinien zur Formatierung von Programmen eingesetzt. Sie dienen zur Vereinheitlichung des Quellcodes bei größeren Projekten, an denen mehrere Programmierer arbeiten. An diese Regeln müssen sich alle Beteiligte halten. Darüber hinaus haben sich einige formale Richtlinien herausgeprägt, die alle Programmierer beachten sollten:

- angemessene Dokumentation (was das bedeutet, wird in der Praxis klar)
- Konsequentes Festhalten an der gewählten bzw. vorgeschriebenen Formatierung.

Benennung

Für die Benennung von Variablen und Methoden haben sich folgende lexikalische Konventionen bewährt:

Klassen-Namen sollten *Substantive* sein, die das Wesen und die Bedeutung der Klasse möglichst kurz und treffend beschreiben. Der erste Buchstabe wird **groß** geschrieben. Sie können auch aus mehreren Worten bestehen, die ohne Leerzeichen und Unterstrich aneinandergehängt werden. Der erste Buchstabe jedes einzelnen Worts wird groß geschrieben.

```
class MeineErsteKlasse
class UniStudent
```

Methoden-Namen sollten *Verben* sein, deren erster Buchstabe **klein** geschrieben wird. Zusammengesetzte Namensbestandteile werden ohne Leerzeichen / Unterstrich aneinandergehängt, wobei der erste Buchstabe jedes Wortes groß geschrieben wird.

```
void gibAus()
int  holMirDasSofortHer()
```

Variablen-Namen sollten möglichst kurz und selbstbeschreibend sein. Der erste Buchstabe wird **klein** geschrieben. Für Zählvariablen werden meist einbuchstabige Namen verwendet: i, j, k, m, für Buchstaben c, d, e.

```
int   i;
char  c;
float eigeneBreite;
```

Konstanten-Namen werden **groß** geschrieben, bei mehreren Bestandteilen wird der Unterstrich verwendet.

```
static final int MIN_WIDTH   = 4;
static final int MAX_WIDTH   = 999;
static final int GET_THE_CPU = 1;
```

Konstanten dürfen in Java übrigens auch verzögert initialisiert werden:

```
static final int MIN_WIDTH;
. . .
MIN_WIDTH = 4;
```

Position der Typdeklarationen

Die Anordnung von Deklarationen in einem Quelltext wird vom Compiler überprüft. Es gibt zwei unterschiedliche Regelsätze für die Positionierung von Typdeklarationen. Hierbei wird unterschieden zwischen den zwei Ebenen, die auch für die Art der Datentypen bedeutsam sind (siehe Kapitel Datentypen):

- Bestandteile auf Klassenebene (Objektvariablen und Methoden)
- Variablen innerhalb eines Anweisungsblocks (lokale Variablen)

a.) Klassenbestandteile

Auf Klassenebene können Attribute und Methoden an beliebiger Stelle deklariert werden. Zwei Arten begegnen in Beispiel-Programmen häufig:

1.) Zuerst alle Attribute, dann die Methodendeklarationen

```
class myClass {

   int a;               // Attribute
   double b;

   void liesEin() {     // Methoden
   }
   void gibAus() {
   }
}
```

2.) Zuerst alle Methodendeklarationen, danach am Ende der Klasse alle Attribute:

```
class myClass {

   void liesEin() {     // Methoden
   }
   void gibAus() {
   }
   int a;               // Attribute
   double b;
}
```

Für den Compiler ist die Position der Attribut-Deklarationen auf Klassenebene gleichgültig. Die menschliche Analyse einer Klasse wird aber erleichtert, wenn die Attribute (die ja in den Methoden referenziert werden) am Beginn der Klasse stehen. In den folgenden Kapiteln wird die erste Variante (erst Attribute, dann Methoden) bevorzugt.

Position von Variablendeklarationen in einer Methode

Prinzipiell können Variablen innerhalb einer Methode an beliebiger Stelle deklariert werden, auf jeden Fall natürlich vor den Anweisungen, die mit ihnen operieren. Es ist nicht zwingend nötig (obwohl empfehlenswert), Variablen ganz am Anfang eines Anweisungsblocks (wie in C) zu deklarieren. Dasselbe gilt natürlich auch für Konstanten.

```
void umfang() {
  int a;                    // Variable
  final double PI = 3.14; // Konstante
  a = 3;
  System.out.println(„Umfang " + (a*PI));
}
```

Beispiel für die hier bevorzugte Formatierung eines Programms

```
class myCircle {

    int ganzZahl;                                   // Attribute
    double kommaZahl;
    String name;

    void methode() {                          // Methode
         ganzZahl  = ganzZahl + 3;
         kommaZahl = kommaZahl + 2.14;
    }
}
```