
Java

Programmierkonventionen

Die wichtigsten Bestandteile der Programmierkonventionen für Java werden hier erläutert. Dies sind:

- Schreibweise von Bezeichnern
- Einrückkonventionen
- Kommentare

Einleitung

- **Spielraum** bei der Gestaltung der **Details**
- **Richtlinien** sollen dazu beitragen, daß die **Programme stets leicht zu verstehen** sind.
- **Lesbarkeit** und **Wartbarkeit** sind wichtig.
- Es gibt zwar keinen "korrekten" Programmierstil, aber die folgenden Konventionen haben sich erfolgreich durchgesetzt.

Bezeichner - 1

- **Bezeichner** müssen (bis auf Laufvariablen in Schleifen, usw. ...) **aussagekräftig** sein.

- Negativbeispiele:

```
public class A ...  
int B ...
```

```
public class Gaius ...  
int Julius ...
```

Bezeichner - 2

- Bei **Bezeichnern**, die **aus mehreren Worten** zusammengesetzt werden, muß der erste Buchstabe eines Wortes jeweils groß geschrieben oder durch "_" getrennt werden.
 - Beispiele:
roemischeZahl, gebeTextEin, gebe_Text_ein
 - Negativbeispiel:
diesisteinschlechterbezeichnerfuereinezahl

Bezeichner - 4

- **Bezeichner für Klassennamen** müssen mit einem Großbuchstaben beginnen.

- Vorsicht bei Bezeichnern, die aus mehr als einem Buchstaben bestehen. Diese können zu **Konflikten mit von Java als Bezeichner belegten Wörtern** führen.

Einrückung - allgemein

- Jede **Ebene** wird **um mindestens 1 Leerzeichen eingerückt**.
- Die Festlegung der Ebenen ist durch die **Einrückkonventionen** gegeben.
- Durch die Einhaltung der Einrückkonventionen wird die **Struktur** der Programme leichter erfaßbar.

Einrücken bei Klassen

```
public class Test {  
    public static void main (<Argumente>) {  
        <Anweisungen>  
    }  
}
```

Methodendeklaration ist eine Ebene tiefer als die Klassendeklaration.

Die Anweisungen sind Ebene tiefer als die Methodendeklaration

Klammern schließen auf der Ebene der zugehörigen Klassen - oder Methodendeklaration

Einrücken bei Schleifen

```
do {  
    <Anweisungen>  
}  
while ( <Bedingung> )
```

```
for ( <Kopf> ) {  
    <Anweisungen>  
}
```

Die Anweisungen in der Schleife sind eine Ebene tiefer als die Identifier für die Schleife.

```
while ( <Ausdruck> ) {  
    <Anweisungen>  
}
```

Klammern sind auf der gleichen Ebene wie die Identifier für die Schleife.

Einrücken bei if-Ausdrücken

```
if <Bedingung> {  
    <Anweisungen>  
}  
else {  
    <Anweisungen>  
}
```

Die Anweisungen sind eine Ebene tiefer als der Bedingungsteil der if-Anweisung.

Klammern sind auf der gleichen Ebene wie die if-Anweisung

Besteht der Teil <Anweisungen> aus nur einer Anweisung, so fordert die Syntax von Java die **Klammern** nicht.

Um die **Übersichtlichkeit** zu bewahren, können die Klammern jedoch stehen!

Kommentare - 1

- /* steht zu **Beginn eines mehrzeiligen Kommentars**.
- */ steht am **Ende eines mehrzeiligen Kommentars**.
- Ein **Leerzeichen** steht nach /* und vor */.
- Die **Zeilen** eines mehrzeiligen Kommentars sollten mit * gekennzeichnet werden.
 - **Beispiel:**

```
/* Dies ist ein
 * mehrzeiliger
 * Kommentar
 */
```

Kommentare - 2

- **Einzeilige Kommentare** werden durch // eingeleitet
- Das Ende eines einzeiligen Kommentars wird nicht gekennzeichnet.
 - Beispiel:

```
// Dies ist ein einzeiliger Kommentar
```

Kommentare - 3

- Für jede Klasse soll zu Beginn kurz die **Funktionalität** beschrieben werden.

- Zusätzlich soll der Kopfkomentar die Informationen über den **Autor**, die **Entwicklungsumgebung**, sowie das **Datum** der **Erstellung** und der **letzten Änderung** enthalten.
 - Der Kopfkomentar ist auch in den **Übungen** zu erstellen!

- Daher ist für jede Klasse ein Kopfkomentar der folgenden Form zu erstellen:

Kommentare - 4

```
/* Dieses Programm zeigt einen Willkommensgruss
 * Autor : Antje Nowack
 * Umgebung : JDK 1.3, Windows 2000
 * Erstellt : 25.10.2001
 * Letzte Aenderung: 30.10.2001
 */
```

```
public class Test{

    public static void main (String [] arguments) {

        <Anweisungen>

    }

}
```

Kommentare - 5

- Nur **sinnvolle Kommentare** sollten eingefügt werden.
- Kommentare sollten **keine redundante Information** enthalten.
 - **Negativbeispiel:**

```
// weise c die Summe von a und b zu  
c = a + b;
```

- Kommentar **vor dem Quelltext**, der kommentiert wird.
- Geht aus den Bezeichnern nicht die Aufgabe der bezeichneten Variablen bzw. Parameter hervor, so sollte diese erläutert werden

Kommentare - 6

- Kommentare dienen dem Verständnis (ebenso wie der übrige Teil der Programmierrichtlinien) des Programmes. Entsprechend **verständlich** sollten die Kommentare verfaßt sein.
 - Beispiel:

```
/* wenn der Rechner nicht benutzt werden kann,  
 * dann soll <Anweisung> ausgeführt werden  
 */  
  
if (Stromausfall || Stecker draussen ||  
     Rechner_kaputt || ... ){  
    <Anweisung>  
}
```

Zusammenfassung - 1

■ Programme müssen gut verständlich sein

- "Wir programmieren nicht für uns, sondern für andere"
- Lesbarkeit und Wartbarkeit
- Programmierkonventionen müssen daher beachtet werden
- dies gilt auch für die Vorlesung "Programmierung"

■ Bezeichnerwahl ist sehr wichtig

- Programme leichter verständlich
- geringere Fehlerhäufigkeit

Zusammenfassung - 2

■ Einrückungen

- helfen, die Übersicht zu behalten
- die Struktur des Programmes schneller zu erfassen

■ Aussagekräftige Kommentare

- helfen, schwierige Stellen zu verstehen
- sind sehr wichtig für die Wartung