

# 1 Benennung

## 1.1 Sprechende Namen verwenden

Benutze sprechende Namen für alle Bezeichner (Klassen-, Variablen- und Methodennamen).

Vermeide Mehrdeutigkeiten. Vermeide Abkürzungen. Einfache verändernde Methoden sollten in der Form `setzeIrgendetwas()` benannt werden. Einfache sondierende Methoden sollten in der Form `gibIrgendetwas()` benannt werden. Sondierende Methoden mit einem booleschen Ergebnis sind oft mit `istIrgendetwas()` benannt, beispielsweise `istLeer()`.

## 1.2 Klassennamen beginnen mit einem Großbuchstaben

## 1.3 Klassennamen sind Hauptwörter im Singular

## 1.4 Methoden- und Variablennamen beginnen mit einem Kleinbuchstaben

Benutze für alle drei – für Klassen-, Methoden- und Variablennamen – Großbuchstaben in der Mitte, um zusammengesetzte Namen lesbarer zu machen, etwa `anzahlEintraege`.

## 1.5 Konstanten werden in GROSSBUCHSTABEN geschrieben

Verwende Unterstriche, um zusammengesetzte Konstantennamen lesbarer zu gestalten: `OBERE_GRENZE`.

# 2 Layout

## 2.1 Eine Stufe der Einrückung besteht aus vier Leerzeichen

## 2.2 Alle Anweisungen innerhalb eines Blocks sind um eine Stufe eingerückt

## 2.3 Klammern für Klassen und Methoden stehen allein auf einer Zeile

Die geschweiften Klammern für den Block einer Klasse oder einer Methode stehen jeweils auf einer eigenen Zeile und befinden sich auf derselben Einrückungsstufe. Beispielsweise:

```
public int gibAlter()
{
```

```
    Anweisungen
}
```

## **2.4 Für alle anderen Blöcke steht die öffnende Klammer am Ende einer Zeile**

Alle anderen Blöcke beginnen mit einer geschweiften Klammer am Ende der Zeile mit dem einleitenden Schlüsselwort für den Block. Die schließende Klammer steht auf einer eigenen Zeile und ist genauso weit eingerückt wie das einleitende Schlüsselwort für den Block. Beispielsweise:

```
while (Bedingung) {
    Anweisungen
}

if (Bedingung) {
    Anweisungen
}
else {
    Anweisungen
}
```

## **2.5 In Kontrollstrukturen immer Klammern verwenden**

Verwende immer geschweifte Klammern in `if`-Anweisungen und Schleifen, auch wenn der Rumpf nur aus einer einzelnen Anweisung besteht.

## **2.6 Vor der öffnenden Klammer für den Block einer Kontrollstruktur steht ein Leerzeichen**

## **2.7 Operatoren links und rechts mit einem Leerzeichen absetzen**

## **2.8 Immer eine Leerzeile zwischen Methoden (und Konstruktoren)**

Benutze eine Leerzeile, um logische Abschnitte im Quelltext voneinander abzusetzen,; also mindestens zwischen Methoden immer eine Leerzeile, aber auch zwischen logischen Abschnitten innerhalb einer Methode.

# **3 Dokumentation**

## **3.1 Jede Klasse wird mit einem Klassenkommentar eingeleitet**

Der Klassenkommentar enthält mindestens  
eine allgemeine Beschreibung der Klasse  
Namen des/der Autoren  
eine Versionsnummer

Jede Person, die zur Klasse beigetragen hat, sollte als Autor aufgeführt sein oder in anderer Form erwähnt werden.

Eine Versionsnummer kann eine einfache Nummer oder ein Datum sein.

Wichtig ist, dass ein Leser an ihr erkennen kann, dass zwei Versionen sich unterscheiden und welche von ihnen die neuere ist.

### **3.2 Jede Methode hat einen Methodenkommentar**

### **3.3 Kommentare sind im Javadoc-Format**

Klassen- und Methodenkommentare müssen von Javadoc erkannt werden können. Deshalb müssen sie mit der Zeichenfolge `/**` beginnen.

### **3.4 Implementierungskommentare nur falls notwendig**

Kommentare in den Anweisungen einer Klasse sollten eingefügt werden, wenn der Quelltext nicht leicht zu verstehen ist.

## **4 Restriktion bei der Sprachbenutzung**

### **4.1 Reihenfolge der Deklaration: Datenfelder, Konstruktoren, Methoden**

Die Elemente einer Klassendefinition erscheinen (falls vorhanden) in folgender Reihenfolge: Paketzugehörigkeit; import-Anweisungen; Klassenkommentar; Klassenkopf; Definition der Datenfelder; Konstruktoren; Methoden.

### **4.2 Datenfelder dürfen nicht public erklärt werden**

### **4.3 Immer einen Zugriffsmodifikator verwenden**

Deklariere alle Datenfelder und Methoden entweder als `private`, `public` oder `protected`.

### **4.4 Klassen einzeln importieren**

Import-Anweisungen, die jede Klasse einzeln benennen, sind dem Import von ganzen Paketen vorzuziehen. Beispielsweise

```
import java.util.ArrayList;
import java.util.HashSet;
```

ist besser als

```
import java.util.*;
```

**4.5 Immer einen Konstruktor deklarieren (auch wenn er leer ist)**

**4.6 Immer einen expliziten super-Aufruf im Konstruktor einfügen**

Füge in einer Subklasse immer den Aufruf `super()` explizit ein, auch wennes ohne ginge.

**4.7 Alle Datenfelder im Konstruktor initialisieren**

## 5 Programmiermuster

**5.1 Iteratoren für Sammlungen benutzen**

Benutze eine `for-each`-Schleife, um über eine Sammlung zu iterieren. Wenn die Sammlung während der Iteration geändert werden muss, verwende einen Iterator, keinen `int`-Index.

Entnommen aus:

Java lernen mit BlueJ

Eine Einführung in die objektorientierte Programmierung

David J. Barnes, Michael Kölling