

Ich kann Programmieren. Warum bekomme ich Regeln auferlegt?

Guter Quelltext „hält länger“ und ist leichter zu warten. Diese Konvention beschäftigt sich mit dem Aussehen und der Struktur von Quelltext. Diese haben großen Einfluss auf die Lesbarkeit und Verständnis und damit auf die Wartung und Lebensdauer von früher investierter Arbeitszeit. Grob gesagt ist Ziel jeder einzelnen Regel, „schlechtes Verhalten“ zu erschweren und „gutes Verhalten“ zu fördern. Daran muss sich auch im Laufe der Zeit jede aufgeschriebene Regel messen lassen und der Guide ggf. angepasst werden.

Grundlegende Regeln zum Programmieren

1. Optimierte den Code für den Leser, nicht für den Schreiber.
2. Jede Regel soll in der Bilanz einen positiven Effekt haben.
3. Wertschätze den Standard, aber vergöttere ihn nicht.
4. Sei konstant und konsequent im Umgang mit Regeln.
5. Wenn etwas „Ungewöhnliches“ passiert, hinterlasse Hinweise für den Leser.
6. Verhindere Konstrukte die überraschend oder gefährlich sind (z.B. side effects).
7. Halte den globalen Namensraum schmal.
8. Optimierte deinen Quelltext auf Kosten der Lesbarkeit erst dann, wenn es Anlass dazu gibt. (Dazu zählen beispielsweise Laufzeit, Speicherbedarf und Flexibilität.)

Datei-Grundlagen

- Alle Quell-Dateien werden in UTF-8 Codiert.
- Whitespaces:
 - Nur Zeilenende („Enter“) und Leerzeichen () sind im Quelltext erlaubt.
 - Alle anderen „unsichtbaren“ Zeichen sind im Quelltext nur durch ihre Ersetzung mit Fluchtzeichen (Tab = \t) erlaubt.
 - Tabulatoren („Tab(s)“) sind als Zeichen explizit nicht erlaubt, da es keinen einheitlichen Standard bei Editoren für die Tab-Breite gibt.
 - Viele Editoren können allerdings Tabs automatisch in Leerzeichen übersetzen, sodass man beim editieren zwar wie gewohnt Tabs verwenden kann, aber die Datei nur die Leerzeichen-Ersetzung enthält.
 - Editoreinstellung: Tabs werden in 4 Leerzeichen übersetzt
- Nicht-ASCII-Zeichen dürfen ausdrücklich im Quellcode erscheinen. Eine Ersetzung ist aufgrund der schlechteren Lesbarkeit zu vermeiden (z. B. "µs" statt "\u03bcs").
- Aber: Bezeichner bitte nur in ASCII

FALSCH	RICHTIG
<code>public class Tür {</code>	<code>public class Tuer {</code>

Formatierung von Klassen (objektorientierte Sprachen)

- Die Member einer Klasse sollten *einer* Ordnung folgen. Diese kann selbst gewählt werden, sollte sich aber nach inhaltlichen und nicht syntaktischen Kriterien richten, d. h. keine Sortierung nach static-Membern o. ä.

FALSCH	RICHTIG
<pre>public static final int STANDARD_TUER = 0; private int typ; public static final int SCHIEBE_TUER = 1; private static int instanceCounter; private boolean klinke_vorhanden; public static final int DREH_TUER = 2; private int oeffnung; protected static final float VERSION=1.0f;</pre>	<pre>protected static final float VERSION=1.0f; public static final int STANDARD_TUER = 0; public static final int SCHIEBE_TUER = 1; public static final int DREH_TUER = 2; private static int instanceCounter; private int typ; private int oeffnung; private boolean klinke_vorhanden;</pre>

- Überladene Methoden/Konstruktoren dürfen nicht getrennt werden.

FALSCH	RICHTIG
<pre>public void oeffne() { ... } public void schliesse() { ... } public void oeffne(double winkel) { ... } public void schliesse(double winkel) { ... }</pre>	<pre>public void oeffne() { ... } public void oeffne(double winkel) { ... } public void schliesse() { ... } public void schliesse(double winkel) { ... }</pre>

Klammersetzung

- Es wird dringend empfohlen die geschweiften Klammern immer zu setzen!
- Warum?** - Oftmals ist es möglich eine einzelne Anweisung statt eines Blocks direkt hinter eine Bedingung zu schreiben. Das ist sehr kompakt, kann aber dadurch auch zu schwer erkennbaren Fehlern führen. Insbesondere bei nachträglichen Bearbeitungen und fehlerhaften Formatierungen ist dann nicht klar, was innerhalb und was außerhalb des Blocks ist.

One Statement per Line

- Eine Zeile enthält nur einen Befehl bzw. Bedingung.
- Mathematische Ausdrücke bilden eine Ausnahme, sofern diese erkennbar oder dokumentiert sind. Die Dokumentation kann mit Verweis auch extern existieren.

Zeilenende

- Aufgrund der Beschränkung der Zeilenbreite einiger Editoren, bzw. aufgrund Unterschiedlicher Arbeitsbedingungen (Monitor- /Auflösungsbreite) sollen Zeilen nicht mehr als 100 Zeichen enthalten (Projektindividuell auch 80).

Bezeichner

- Variablen-, Klassen- und Funktionsnamen und andere Bezeichner haben stets ihre Bedeutung als Bezeichnung. Ausnahmen sind:
 - Laufvariablen, sofern sie nicht in Begleitung anderer anonymer Variablen sind oder
 - technische oder mathematische Bezeichner, deren Bedeutung sich z. B. aus Formeln und damit anderer Dokumentation ergibt.

Exceptions

- Ausnahmen dürfen auch nur als solche auftreten und dürfen nicht Teil der Programmlogik sein.
- Wenn eine Methode Exceptions werfen kann, dann muss auch der Header darauf hinweisen (Indikator *throws*).
- Java: Wirft ein Statement wahrscheinliche eine RuntimeException, dann muss diese vor Ort auch behandelt werden, oder als Nicht-RuntimeException weitergeleitet werden.
- C/C++: Exceptions sind im Allgemeinen zu vermeiden und wahrscheinlich auftretende Exceptions direkt zu behandeln, da der Compiler nicht ihr potentiellies Auftreten prüft.
- Wahrscheinliche Exception (Java-Beispiel):
 - `Double.parseDouble(String s) throws NumberFormatException`

Acknowledgements

- Datei „Acknowledgements.txt“ im Softwarepaket hinzufügen
- Template: „Dieser Quellcode entstand im Rahmen eines von [Geldgeber/Auftraggeber] geförderten Projektes [Projektname/Akronym, Projektnummer/Förderkennzeichen].“
- Autor(en) nennen (inklusive Konzeptgeber, etc.)
- Außerdem Englische Übersetzung

Lizenz

- WICHTIG: Eine Softwarelizenz ist immer anzugeben, da Dritte sonst weder die kompilierte Software, noch den Quelltext verwenden dürfen! Es ist immer (am besten automatisiert) die Lizenz anzugeben, unter der dieser Quelltext (und später auch das Kompilat) steht. Fehlt eine Lizenz und das führt bei späteren Kunden zu Schadensersatzforderungen, so kann man durch Rechtemängelhaftung auch finanziell zur Verantwortung gezogen werden.

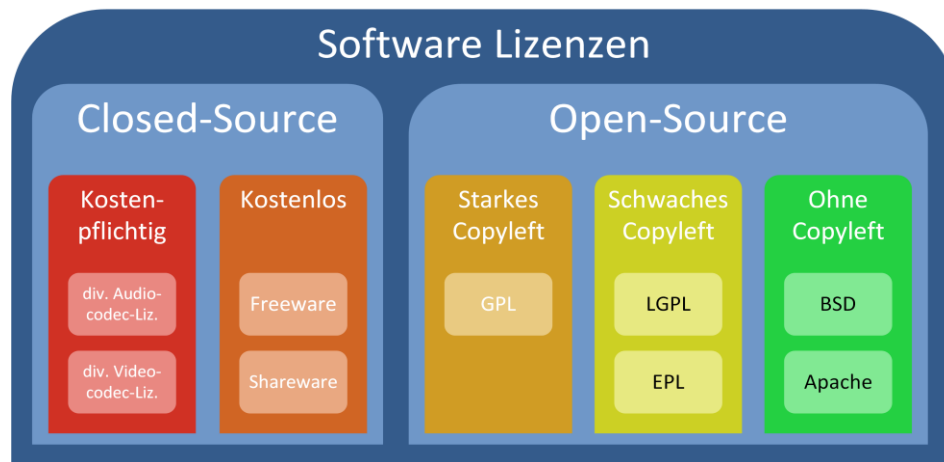


Abbildung 1: Übersicht Softwarelizenzen

- Sofern keine expliziten Gründe dagegen sprechen, ist die Fachgebiets-Lizenz zu verwenden (BSD 3-clause license – siehe Anhang). Diese ist auch kompatibel zur GPL-Lizenz, man kann die Software also gleichzeitig unter beiden Lizenzen veröffentlichen.
- Umgang mit Fremdlizenzen (siehe Abb. 1):
 - **Copyright-Lizenzen** (z.B. GPL, LGPL, Mozilla Public License), vor allem solche mit starkem Copyleft, sind im Allgemeinen kritisch wenn es um die Weitergabe der Software sowie des (erweiterten) Quelltextes geht. Die Lizenz wird hier u. U. **vererbt**, d. h. man muss bei deren Weitergabe oft die eigene Software unter die gleiche Lizenz stellen sowie den Quelltext veröffentlichen. Die Verwendung sollte **immer mit dem jeweiligen Betreuer/Projektpartner absprechen**.
 - **Proprietäre Software(-Pakete)** verbieten in der Regel die kostenfreie Nutzung oder stellen sie unter kritische Auflagen und sind damit noch schwieriger zu verwenden als GPL-lizenzierte Software. Im Allgemeinen sind sie zu vermeiden, bzw. auf Ersatz zu überprüfen. Jede Verwendung muss mit dem Betreuer/Projektpartner geklärt werden.
- Häufige Pflichten bei Verwendung von Freien- und Open-Source-Softwarelizenzen (FOSS-Lizenzen):
 - Bei Verbreitung sind **ALLE beteiligten Urheber** zu nennen, also auch die Urheber aller verwendeten Bibliotheken. (Mitliefern des Quelltexts umgeht das Problem.)
 - Bei Verbreitung ist der **Lizenztext** einer Bibliothek oder eines Programms mitzuliefern.
 - Bei Verbreitung ist der **Haftungsausschluss** (engl. Disclaimer, meist in Großbuchstaben und Teil des Lizenztexts) mitzuliefern.
 - LGPL - obwohl nur mit „schwachem“ Copyleft versehen, ist hier dennoch Vorsicht geboten:
 - Der LGPL-Teil muss austauschbar bleiben.
 - Das Debuggen und Reverse Engineering vom LGPL-Teil darf nicht untersagt werden.
 - Durch dynamisches Linken kann so eine Trennung erfolgen, aber: auch dynamisch verlinkte Bibliotheken können den Status „abgeleitet“ erzeugen und ggf. das Copyleft auslösen.

Fachgebiets-Lizenz (BSD)

Copyright (c) <YEAR>

TU Berlin, Institut für Werkzeugmaschinen und Fabrikbetrieb

Fachgebiet Industrielle Automatisierungstechnik

Authors: <AUTHORS>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

DISCLAIMER

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.