Was ist der Zweck von gemeinsamen Programmierrichtlinien/ Coding-Guidlines?

Wenn eine Gruppe von Applikationsentwickler an einer App arbeiten, dann kann das zu Problemen führen wenn es keine Richtlinien gibt. Also müssen allgemeine Richtlinien erstellt werden. So Programmiert jeder nach den Richtlinien und nicht mit seinem eigenen Style.

So ist der Programmcode der ganze Gruppe einheitlich.

Was kann/muss eine Richtlinie alles umfassen?

* Die Einrückung muss definiert werden. Der Code soll schön eingerückt werden. Alles soll einheitlich aussehen.
* Kontrollstrukturen: Die Klammern bei if, for, while oder switch müssen immer am gleichen Ort sein. Im ganzen Programmcode müssen die Klammern etweder direkt neben der Kontrollstruktur sein oder unter der Kontrollstruktur.

Das sind nur zwei Beispiele die in Programmierrichtlinien vorkommen können.

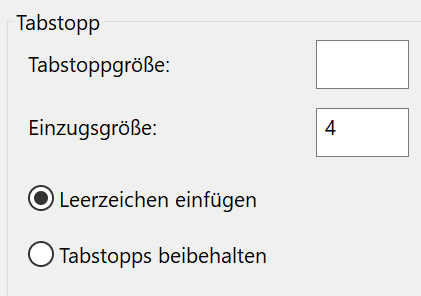
**Programmierrichtlinien**

Die folgenden Programmierrichtlinien wurden erstellt von Nevin Helfenstein und Lino Meyer.

Der Zweck ist die Vereinheitlichung des Programmcodes.

Wenn eine Gruppe von Applikationsentwickler an einer App arbeiten, dann kann das zu Problemen führen wenn es keine Richtlinien gibt. Also müssen allgemeine Richtlinien erstellt werden. So Programmiert jeder nach den Richtlinien und nicht mit seinem eigenen Style.

**Einrückung**

Einrückungen gehören zu den wichtigsten Mitteln um den Code strukturiert zu halten. Wir haben uns für Einrückungen in Höhe von **vier** Leerzeichen entschieden. Das heißt, im VisualStudio muss eingestellt sein, dass ein Tab vier Leerzeichen entspricht. Ausserdem sollte jeder einstellen, dass die Tabs beim Einfügen/Speichern durch Leerzeichen ersetzt werden.

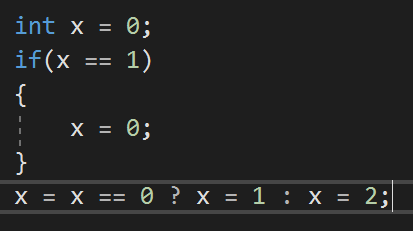
**Kontrollstrukturen**

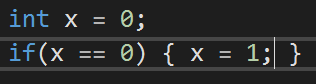
Mit Kontrollstrukturen sind folgende Elemente gemeint:

* if
* for
* while
* switch

Hierbei sollte darauf geachtet werden, dass die geschweiften Klammern immer auf der Höhe der zugehörigen Kontrollstruktur in einer separaten Zeile stehen!

Bei if-Anweisungen ist die kurze Schreibweise erlaubt.

Gutes Beispiel:

Schlechtes Beispiel:

**Funktionsaufrufe**

Ein Funktionsaufruf geschieht immer ohne Leerzeichen zwischen dem Funktionsnamen, der öffnenden Klammer und dem ersten Übergabeparameter. Die einzelnen Parameter werden mit Komma und Leerzeichen voneinander getrennt. Hinter dem letztem Parameter wird ebenfalls kein Leerzeichen mehr eingesetzt.

Gutes Beispiel:

Schlechtes Beispiel:

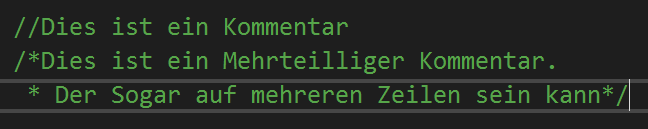
**Kommentare**

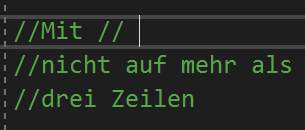
Damit auch andere die eventuell vorhanden Bugs in eurem Code fixen können, muss dieser nicht nur übersichtlich, sondern auch verständlich sein. Daher ist der ein oder andere Kommentar unumgänglich. Hier ist der richtige Mittelweg zu wählen. Zu viel Kommentar verschränkt den Blick auf das wesentliche (den Code), zu wenig Kommentar lässt ihn undurchdringbar erscheinen…

Kommentare sollten in der Regel in englisch erfasst werden, so dass der Sourcecode auch von nicht deutschsprachigen Entwicklern gelesen werden kann.

Kommentare direkt im Sourcecode (keine Funktions- und Methodenbeschreibung) müssen eingerückt werden. So dass sie auf der selben Ebene wie das zu kommentierende Konstrukt stehen. Ein- bis zweizeilige Kommentare werden wie folgt dargestellt:

Gutes Beispiel:



Schlechtes Beispiel:

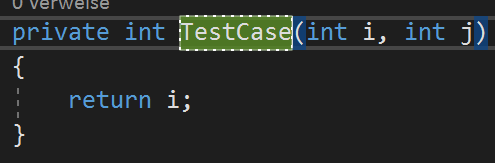
**Namenskonventionen**

**Klassen**  
Klassen sollen selbstsprechende Namen erhalten. Kryptische Abkürzungen sollte man wenn möglich vermeiden. Klassen sollten immer mit einem Großbuchstaben beginnen.

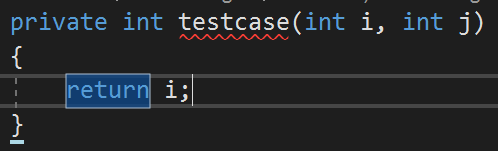
**Funktionen**

Bei Funktionen ist der erste Buchstabe und alle nächsten Anfangsbuchstaben der folgenden Wörter gross geschrieben.

Gutes Beispiel:



Schlechtes Beispiel:

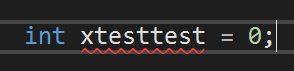


**Variablen**  
Variablen sollten ähnlich den Funktionsnamen im camelStyle (vielen auch als [camelCaps](http://de.wikipedia.org/wiki/CamelCaps" \o "http://de.wikipedia.org/wiki/CamelCaps) oder laOlaStyle bekannt) geschrieben werden. Nur bei Variabeln ist der Anfangsbuchstabe des ersten Wortes klein.

Gutes Beispiel:



Schlechtes Beispiel:



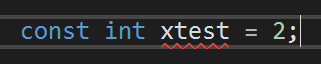
**Konstanten**

Bei Konstanten werden alle Buchstaben groß geschrieben und die einlenen Worte innerhalb de Namens durch Unterstriche getrennt.

Gutes Beispiel:



Schlechtes Beispiel:



Einleitung:

In diesem ÜK haben wir alle notwendigen Sachen gelernt um dieses Projekt zu vervollständigen. Das Ziel ist es die öffentliche API-Schnittstelle der SBB anzuziehen und mit diesen Daten zu arbeiten. Um diese Projektarbeit zu verwirklichen mussten viele Theorieblöcke gemacht werden. Zum einen mussten zuerst die Grundlagen der Programmiersprache C# erlernt werden, mit der das Projekt realisiert wurde. Danach mussten auch noch verschiedene Diagramme besprochen und gemacht werden. Das sind Aktivitätsdiagramme, Use Cas Diagramme und Klassendiagramme. Dies war wichtig um solche Diagramme am Schluss für die Dokumentation zu erstellen.

Zweck dieses Dokumentes:

Dieses Dokument ist keine Dokumentation zu meiner Projektarbeit sondern soll nur das wichtigste meiner Projaktarbeit beschreiben und verständlich machen. Ansonsten sind in diesem Dokument auch noch meine Entwicklungsrichtlinien vorhanden, an denen ich mich den ganzen Programmcode gehalten habe.