1 Style Guides

Die hier vorgestellten Richtlinien werden zum Grossteil von Together schon vorgegeben, was insbesondere durch die Entsteheung aus Java heraus begruendet ist. Dennoch werden sie explizit erwaehnt, da so Unklarheiten bei manuellen Aenderungen vermieden werden.

Es soll mit diesen Richtlinien sichergestellt werden, dass das Projekt vom Verwaltungs- und Wartungsaufwand ein vertretbares Mass einhaelt. Zudem sollen Rueckfragen zwischen einzelnen Personen oder auch Komponenten auf ein (notwendiges) Minimum begrenzt werden.

1.1 Dokumentation

Werden Klassen, Methoden usw. dokumentiert, so sind diese Kommentare im JavaDoc/docc++-Stil zu erzeugen. Unter Together erfolgt dies als Description und wird immer als besonders markierter Kommentar vor dem zu dokumentierenden Bereich eingefuegt.

Zwingend notwendig anzugeben sind Pre-, Postconditions und eine (moeglichst einzeilige) Kurzbeschreibung.

1.2 Packages

Ihre Namen werden komplett klein geschrieben, bis auf den Wortanfang und setzt sich ihr Name aus mehreren Woertern zusammen, dann wird jedes neue Wort am Anfang gross geschrieben.

Beispiel

```
Haltestelle/
Linienplan/
Fahrplan/
ProgrammSchnittstelle/
```

1.3 Klassen

Sie genuegen den oben aufgefuehrtem Schema. Zusaetzlich wird jeder Klassenname mit einem C eingeleitet. Die Namen sind stets Substantive.

Beispiel

```
class CHaltestelle {
} // class CHaltestelle
```

1.4 Attribute

Werden grundsaetzlich kleingeschrieben, bis auf neue Wortanfaenge. Der Zugriff auf selbige ist beschraenkt (private) und darf nur durch setze- bzw. hole-Methoden erfolgen.

Beispiel

```
class CFahrplan {
private
    int a;
    int prioritaet;
public
    void setzenPrioritaet(int a);
    int holenPrioritaet();
} // class CFahrplan
```

1.5 Methoden

Werden ebenso nach dem obigen Schema kodiert, beginnen aber immer mit einem Kleinbuchstaben. Die Namen sind Verben in der Grundform und eventuell angehaengten Substantiven.

Beispiel

```
class CLinienplan {
    void berechenLaenge( void );
} // class CLinienplan
```

1.6 Objekte

Namen haben gleichen Aufbau, wie bei den Klassen, jedoch ohne das einleitende C.

Beispiel

```
CHaltestelle Haltestelle;
```

1.7 Datentypen

Sofern unbedingt notwendig, beginnen die Namen dieser mit einem T und genuegen dem oben schon aufgefuehrten Schema.

Beispiel

```
struct TKnoten = {
}; // struct TKnoten
```

1.8 Codeeinrckung

In jeder Zeile ein Befehl. Eingerckt wird generell mit dem Tabulator, wobei hinter { eine Ebene tiefer und mit } eine Ebene weniger eingerckt wird. Die ffnende Klammer kommt dabei an das Ende des blockeinleitenden Befehles. Die schlieende Klammer gelangt in eine eigene Zeile.

Eingerckt wird auch, wenn ein Befehl direkt von einer vorhergehenden Zeile abhngt, jedoch kein eigener Block benutzt wird.

1.9 Sonstiges

- Pro Klasse je folgende Dateien:
  .cpp
  .hpp
  Dabei darf nie mehr als eine Klasse innerhalb einer Datei definiert oder imlementiert werden.

- Strikte Objektorientierung, bis auf verwendete Bibliotheken und einige wenige Ausnahmen.

- Akurate Verwendung der Kapselungsmglichkeiten durch private, public und protected.

- Kein direkter Zugriff auf Attribute! Entweder indirekter Zugriff per setzenAttribut, holenAttribut oder Manipulation von methodennotwendigen Attributen nur durch diese Methoden. Das heisst auch, Vermeidung von Rueckgaben bei Methoden, welche den Zustand eines Objekte manipulieren.

- Bei Parameterbergabe per Referenz ist immer die zeigerlose Variante mit &, anstelle von * zu verwenden.

- Fehler sind per Exceptions zurueckzugeben.