**Version 0.1 -** 08.März 2018

**Code Style**

**Guide**

**Fahrzeug Organisations-System**

**GRUPPE 16:** Reto Mayer, Jonas Rüegge, Marco Sutter, Fabian Wipf

[Dokumentenzweck 2](#_Toc508315600)

[Namenskonventionen 2](#_Toc508315601)

[Java-Namenskonventionen werden eingehalten (L1 - Vorgabe) 2](#_Toc508315602)

[Zusammenfassung Vorgaben PSIT2 3](#_Toc508315603)

[Code ist aufgeräumt (L1) 3](#_Toc508315604)

[Klassen 3](#_Toc508315605)

[Testing 3](#_Toc508315606)

[Wartbarkeit 3](#_Toc508315607)

[Korrektheit 4](#_Toc508315608)

[Organisatorisches Vorgaben 4](#_Toc508315609)

[Standard-Java-Verzeichnisstruktur (L2) 4](#_Toc508315610)

[Eigene Organisatorische Vorgaben 5](#_Toc508315611)

[Glossar 5](#_Toc508315612)

# Dokumentenzweck

Dieses Dokument legt Konventionen fest, bzw. sammelt diese wo vorgegeben, und dient als Code Style Guide für das Projekt Fahrzeug-Organisations-System, kurz FOS. Im Folgenden wird dieses Dokument mit CSG abgekürzt. Als Guide für den CSG dienen neben den Projektvorgaben[[1]](#endnote-1) die Guidelines für Clean Code in Java[[2]](#endnote-2) .

# Namenskonventionen

## Java-Namenskonventionen werden eingehalten (L1 - Vorgabe)

Klassennamen sind Upper Camel Case (MySpecialClass)

Methodennamen, Variablennamen und Feldnamen sind Lower Camel Case (someMethod())

Konstanten sind Grossbuchstaben (ABSOLUTE\_NULL)

Klassennamen sind nach Möglichkeit selbsterklärend und beschreiben die Funktion der jeweiligen Klasse. Variablen und Methodennamen werden geeignet gewählt, d.h. möglichst kurz, eindeutig und Funktions- bzw. Werteerklärend. Beispiele sind im Code-Beispiel Teil des CSG vorhanden.

# Zusammenfassung Vorgaben PSIT2

## Code ist aufgeräumt (L1)

Kein toter Code ist vorhanden

Sinnvolle Kommentare

Verzichten Sie auf Copy & Paste

Ziehen Sie Konstanten Magic Numbers vor

Keine "[ungarische Notation](https://en.wikipedia.org/wiki/Hungarian_notation)"

Keine Magic Numbers (->Konstanten)

## Klassen

#### Klassen haben Verhalten (L2)

#### richtige Funktionen, nicht nur get() und set()

#### Klassenoberfläche ist minimal (L3)

#### Klassen stehen für sich allein (L3)

#### Sinnvolle Abstraktionsebene (L4)

## Testing

* Ein sinnvoller Unit-Test ist vorhanden (L1)
* Logik mit Tests abgedeckt (L2)
* Mock-Objekte statt externe Kollaborateure (L4)
* Wo Möglich: Schnittstellen für GUI Testing

## Wartbarkeit

* Ausnahmen werden behandelt (L1)
* Logging anstelle von System.out (L2)
* Klassen, ENUMs statt primitive Typen (L2)
* Vernünftiger Einsatz von Feldern (L2)
* Interfaces statt konkrete Typen (L3)
* Anwender erhält sinnvolle Fehlermeldung (L3)
* Kollaborateure als Konstruktor-Argumente (L3)
* Javadoc (L4)

## Korrektheit

#### Null-Objekte anstelle von null (L2)

Geben Sie als Rückgabewerte von Methoden nach Möglichkeit sogenannte Null-Objekte zurück und nicht null an sich. Damit verhindern Sie eine lästige NullPointerException.

#### Ressourcen werden geschlossen (L2)

Besonders Datenbankverbindung -> keine manuellen nur über DB Klasse

#### Eingabeüberprüfung (L3)

#### Streams sind seiteneffektfrei (L4)

# Organisatorisches Vorgaben

#### Frühe, häufige Commits (L1)

#### Gute Commits (L2)

#### Angemessene Verwendung von Collections (L3)

#### Vernünftige Objektlebenszeiten (L3)

#### Konfiguration ist ausgelagert (L4)

## Standard-Java-Verzeichnisstruktur (L2)

src/main/java

Platz für Java-Klassen, jeweils in einem Unterverzeichnis korrespondierend zum Package-Namen (die öffentliche Klasse Test im Package com.example kommt nach src/main/java/com/example/Test.java). Speichern Sie jede Klasse in einer eigenen Datei, egal ob sie öffentlich ist oder nicht.

src/main/resources

Speichern Sie hier Konfigurationsdateien oder ähnliches, das nicht zur webapp gehört.

src/main/webapp

Legen Sie in diesem Verzeichnis Ihre Webapplikation ab.

src/main/java

Platz für Test-Klassen. Der Test für die öffentliche Klasse Example im Package com.example kommt nach src/test/java/com/example/ExampleTest.java. Es ist wichtig, dass Tests in demselben Package sind die zu testenden Klassen, damit sie in den Tests auf Methoden und Felder zugreifen können, die package private sind.

src/main/resources

Speichern Sie hier Konfigurationsdateien oder ähnliches für Tests.

# Eigene Organisatorische Vorgaben

# Glossar

|  |  |
| --- | --- |
| CSG | Abkürzung für Code Style Guide (dieses Dokument) |
|  |  |
|  |  |

1. https://olat.zhaw.ch/auth/1%3A1%3A1133812622%3A4%3A0%3Aserv%3Ax/PSIT2/Code%20Review/handbuch.html#review-details [↑](#endnote-ref-1)
2. Clean Code: A Handbook of Agile Software Craftsmanship, [Robert](https://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882?SubscriptionId=AKIAILSHYYTFIVPWUY6Q&tag=duckduckgo-ffsb-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0132350882) C. Martin, ISBN-13: 978-0132350884 [↑](#endnote-ref-2)