Naming Guide

Inhalt

[1 Git, Github 1](#_Toc119614122)

[1.1 Git Bash 1](#_Toc119614123)

[1.2 VIM Commands 2](#_Toc119614124)

[1.3 Editor Commands 2](#_Toc119614125)

[2 C# 3](#_Toc119614126)

[2.1 Using 3](#_Toc119614127)

[2.2 Class Variables 3](#_Toc119614128)

[2.3 Methods 4](#_Toc119614129)

[2.4 Method Variables 4](#_Toc119614130)

# Git, GitHub

**Project-URL :** <https://github.com/KevinvonBallmoos/ProjectTextingSpree/blob/master/Projekt%20Texting%20Spree.docx>

Before typing in any commands, be sure that you are in your Project Folder and you have selected the right branch.  
(e.g. master, develop, test)

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## Git Bash

|  |  |
| --- | --- |
| Initialize Git | git init |
| Add all Files | add . |
| Commit Files | git commit -m “Commit” |
| Point Git to existing Repo URL | git remote add origin <project url> |
| Verify Git Repo | git remote -v |
| Push changes | git push origin <branch> |
| Switch branch | git switch -c <branch> |

## VIM Commands

|  |  |
| --- | --- |
| Exit Vim | :wq |
| Exit without Save | :q! |

Additional Commands are listed here (Link): [***Vim-Cheat-Sheet***](https://vim.rtorr.com/)

Similar Commands apply to other Editors in Bash.

## Editor Commands

Be sure to be in the bottom line, else the commands won’t work.  
If you are on another line, press ESC or CTRL + C to get to the bottom line.

|  |  |
| --- | --- |
| Exit | :wq |

# C#

Below, this document describes the naming of the variables, methods and other code syntax. The Class Code Conventions includes Examples to all Points.

Adjust namespaces, according to Folder structure.

Curly Brackets are below the Code and not on the same Line.

## 2.1 Using

|  |  |
| --- | --- |
| Nr. | Description |
| 1 | Usings are sorted alphabetically |
| 2 | Usings of other Libraries comes first. Then follows usings of other own Classes. |
| 3 | Unused usings have to be removed. |

## 2.2 Class Variables

|  |  |
| --- | --- |
| Nr. | Description |
| 1 | They are either private, public or protected. |
| 2 | Should a private field be visible from the Inspector and serializable as well, then the [SerializeField] can be added. |
| 3 | Variables are written in lowerCamelCaseand with an underscore to begin with. |
| 4 | Use of type var is not allowed. |
| 5 | Readonly or const variables are written in Upper CamelCase, without an underscore. 1 |
| 6 | They have to be sorted. |

## 

## 2.3 Methods

|  |  |
| --- | --- |
| Nr. | Description |
| 1 | Method Names are written in UpperCamelCase, and start with a Verb. |
| 2 | Method Parameters are written in lowerCamelCasewithout an underscore to begin with. |

## 2.4 Method Variables

|  |  |
| --- | --- |
| Nr. | Description |
| 1 | Variables are written in lowerCamelCasewithout an underscore to begin with. |
| 2 | Use of type var is allowed and recommended in different situations.  When defining variables with var make sure to initialize them. |

## 2.5 Miscellaneous

|  |  |
| --- | --- |
| Nr. | Description |
| 1 | Loops or Ifs can be simplified when the body is no longer than 1 Line. |
| 2 | … |

## 2.6 Comments

|  |  |
| --- | --- |
| Nr. | Description |
| 1 | Every Class need to be commented, with Class Task, Author and Date of Creation |
| 2 | Class Variables and Getters can be commented, after they are sorted. 2 Slashes -> // |
| 3 | Every Method, except Getters are commented above the Method with 3 Slashes -> ///. This creates an automatic comment section for parameters or return values. |
| 4 | Code like loops or statements can be commented only if its unclear what’s happening. |
| 5 | With #region, the Code can be separated and provides more readability. |

1

Difference const and readonly   
Const is initialized at compile time, while readonly can be initialized at compile time or runtime.