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, must be removed. |

## 2.2 Class Variables

|  |  |
| --- | --- |
| **Nr.** | **Description** |
| 1 | The Access Modifiers private, public or protected are mandatory. |
| 2 | Should a private field be visible from the Inspector and serializable as well, then the [SerializeField] attribute can be added. |
| 3 | Public variables which should not be visible from the Inspector, get the [NonSerialized] attribute. |
| 4 | Use of type var is not allowed. |
| 5 | The naming convention is generally lowerCamelCase. |
| 6 | Readonly or const variables are written in UpperCamelCase, without an underscore. 1 |
| 7 | They must be sorted, grouped and named (single or as group)   1. Logger class (even if not used later) 2. All class variables 3. Serialized 4. NonSerialized (public) 5. Public 6. Private 7. Static (public, private) 8. Const / Readonly   The Order is the same in the Code Conventions |

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

## 

## 2.3 Methods

|  |  |
| --- | --- |
| **Nr.** | **Description** |
| **1** | Method Names are written in Upper CamelCase 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** | Use of System.LINQ is recommended |

## 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 that are single lined, are commented above the Method with 3 Slashes -> ///. This creates an automatic comment section for parameters and return values. |
| **4** | Code like loops or statements can be commented only if its unclear what’s happening. |
| **5** | With #region / #endregion, the Code can be separated and provides more readability. |

## 2.7 Logging

|  |  |
| --- | --- |
| **Nr.** | **Description** |
| **1** | The first class-variable must be the Logger class. Even if its not used to log Entries. Its just to look uniform. |
| **2** | Instantiate the Logger Class. (Don’t forget the using directive)  The Constructor takes the Name of the Class as parameter |
| **3** | Call the LogEntry Function, when you want to log.  This takes 3 Parameters: Type, Message, Line Number |

## 2.8 Exception Handling

|  |  |
| --- | --- |
| **Nr.** | **Description** |
| **1** | Use Try catch to catch Exceptions |
| **2** | Use using to dispose Code that’s not needed anymore |
| **3** | Use the Logger Class to log Exceptions |