Code Manager

Manual

01.12.2021
Version 1.0

Silvan Ott

Content

1 Overview

Code Manager has two main features. The First Feature is ‘Script Templates’. With Script Templates you can create generic templates from which you can create scripts in the before defined folder structure. With the second main feature ‘Clean Code’ you can create clean code rules and then scan the folders for your clean code rule violations. You can create clean code rules with regexes to find either unwanted- or undocumented code or define coding guidelines.

2 Requirements

There are no requirements for Code Manager, but it is recommended to have a basic understanding of Regexes, else at the end are some links to learn the basics of Regex.

3 Compatibility

Code Manager was developed in Unity 2020.3.3f1 on Windows 10. Older Unity versions might work but are not verified.

4 Setup

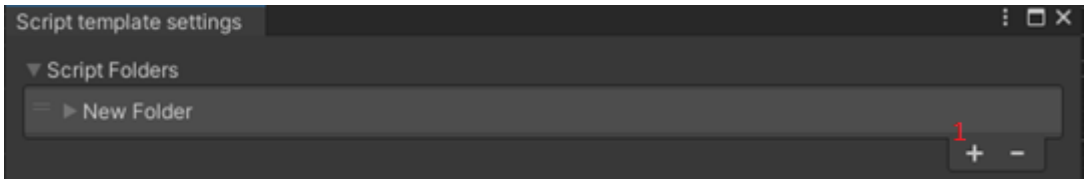
5 Features

5.1 Script Templates

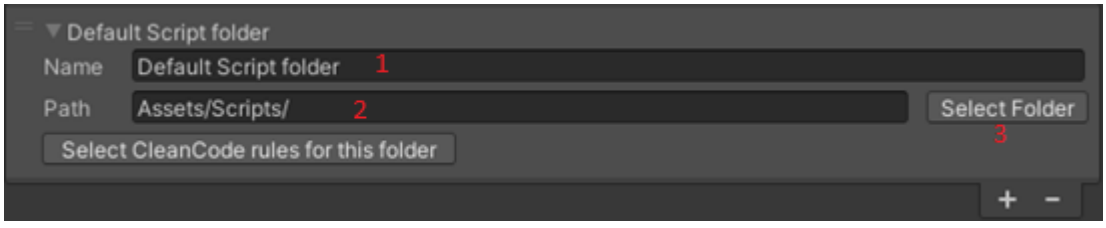
With Script Templates you create templates which can contain placeholders. In the settings you can give every placeholder a value. You then can create a new script from one of these templates and during script creation the placeholders will be replaced through your defined values. With this you can use the same templates in multiple projects, and you just must adjust some placeholder values. Script Templates also lets you define script folders. By script creation you can choose one of the script folders the script will then be created in this folder.

5.1.1 Define script folders

1. Open the Script Templates settings under: Code Manager => Script Templates => Settings.
2. Open the Script Folders List and add a new Item by pressing the “+” (1) button.



3. Expand the new Item.
    - In the Name field (1) you can give your script folder a name. (This does not have to match the real folder and is just for identification)
    - In the Path field (2) you now can either write the path to the folder or select the folder in the explorer by pressing the Select Folder (3) button. The Folder must be under Assets/ and at the end of the path must be a “/” so the path points into the folder.



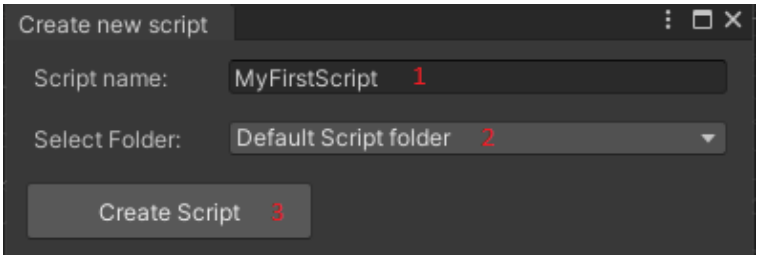
5.1.2 Create script templates

1. Click the menu: Code Manager => Script Templates => New script template.
2. The explorer will open. Enter the name of the new script template and press Save.

- The script template must be under the folder Assets/ScriptTemplates/
- The template must be a .txt
- You do not have to use the menu to create a template. You can create a .txt by your own and simply put it somewhere under Assets/ScriptTemplates/

5.1.3 Create new script

1. Open the Script Create window under: Code Manager => Script Template => New script or with the shortcut Ctrl + T
2. Here you can see all templates you have created select the template from which you want to create your script.
3. A new window will open where you can input the script name (1) and select the folder where you want to create the script (2). The folders to select are the script folders created in the settings.
4. Press the Create Script (3) button to create the new script.



5.1.4 Placeholders

To create a new placeholder, open the Script Templates settings under: Code Manager => Script Templates => Settings. Open the Placeholders list and add a new item (1).

Write the name of the placeholder in the Name field (2) and which value the placeholder should have after creation in the Value field (3)

To use a placeholder in the template, write the Name of the placeholder surrounded by "%". After you create a new script with this template the placeholders will be replaced.

```
using UnityEngine;
using System.Collections;

namespace %namespace%
{
    public class %ScriptName% : MonoBehaviour
    {

        // Use this for initialization
        void Start () {

        }


        // Update is called once per frame
        void Update () {

        }
    }
}
```

```
using UnityEngine;
using System.Collections;

namespace Morchul.CodeManager
{
    public class MyFirstScript : MonoBehaviour
    {

        // Use this for initialization
        void Start () {

        }


        // Update is called once per frame
        void Update () {

        }
    }
}
```

How you can see there is already a placeholder in the default template: "%ScriptName%" this is a default placeholder and will be replaced through the name of the script.

It is in your own responsibility that a placeholder name never occurs twice in the settings. If so one placeholder just will be ignored.

List of default placeholders	
Placeholder name	**Value**
ScriptName	The name of the script

5.2 Clean Code

Clean Code is a tool to keep your scripts without any unwanted- or undocumented code and let you define coding guidelines. All these with Regexes. These three types (unwanted code, code documentation and code guideline) are called clean code rules and work the following:

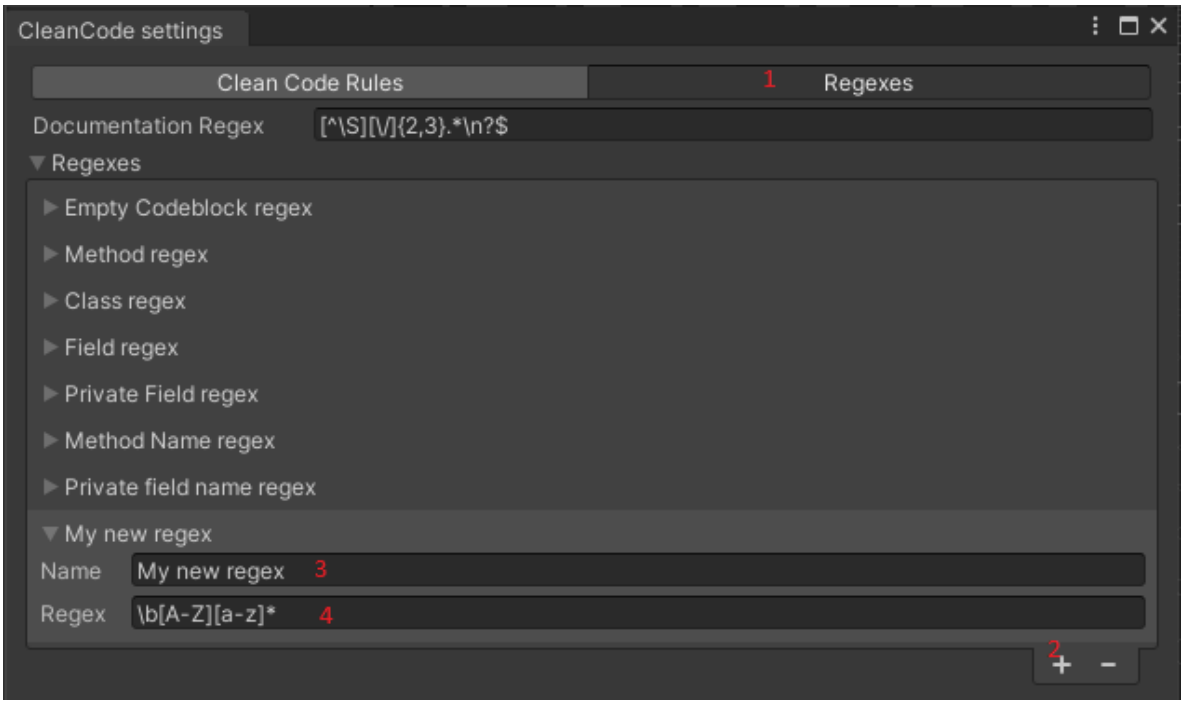
- **Unwanted Code**: You can define a Regex for code you do not want. If this regex matches anything, it will be displayed to you in the clean code console as clean code violation.
- **Code Documentation**: You can define a Regex for code you want to have documented. If this regex matches something the Code before this match will be tested with a Documentation regex. This documentation regex should check if there is a documentation. If there is no documentation a clean code violation will be displayed in the clean code console.
- **Code Guidelines**: You define a search regex which has at least one group name in it. If this search regex matches something the value of the group specified by the group name will be tested with the match regex you can also define by yourself. If this test does not match a clean code violation will be displayed in the clean code console.

There is a clean code console in which all clean code violation will be displayed. By click on them you will be directed to the place where the clean code violation occurred.

For each created script folder, you can choose between all clean code rules you created for which rules you want to scan all scripts in the folder.

5.2.1 Create Regexes

1. Open the Clean Code settings under: Code Manager => Clean Code => Settings.
2. Change to the Regexes Tab (1), open the Regexes list, and add a new item (2)
3. Enter the name of the regex In the Name field (3) and the regex in the Regex field (4). If the Regex is not valid a message will appear beneath the Regex field.

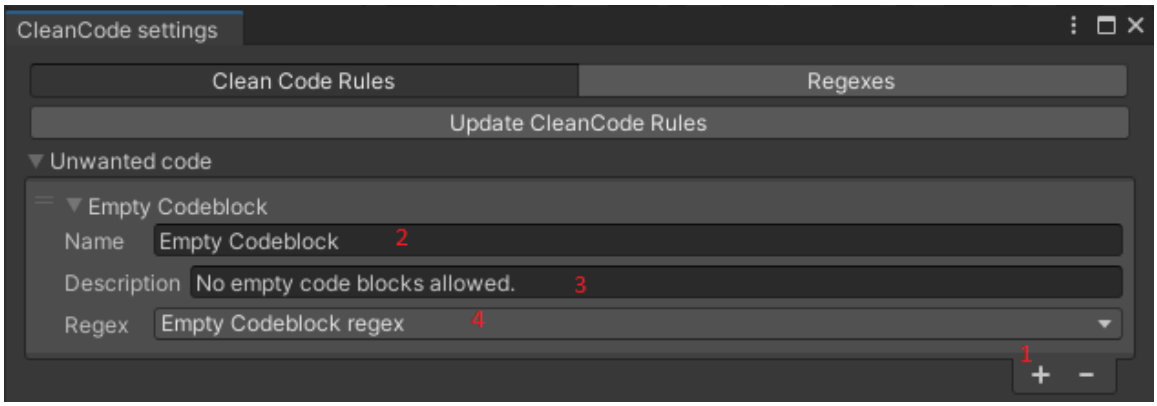


5.2.2 Create Clean Code Rules

There are three different types of clean code rules. Unwanted code, code documentation and code guideline.
The match UI shown in the examples are not a part of Code Manager and made with the website regex101. The link to regex101 is under 6.1 Regex.

Unwanted Code:

1. Open the Clean Code Settings under: Code Manager => Clean Code => Settings.
2. Open the Unwanted Code list and add a new item (1)
3. Enter the name of the unwanted code rule in the Name field (2). Enter a description in the Description field to explain the rule (3). Finally, select a Regex you have created which matches the unwanted code (4).



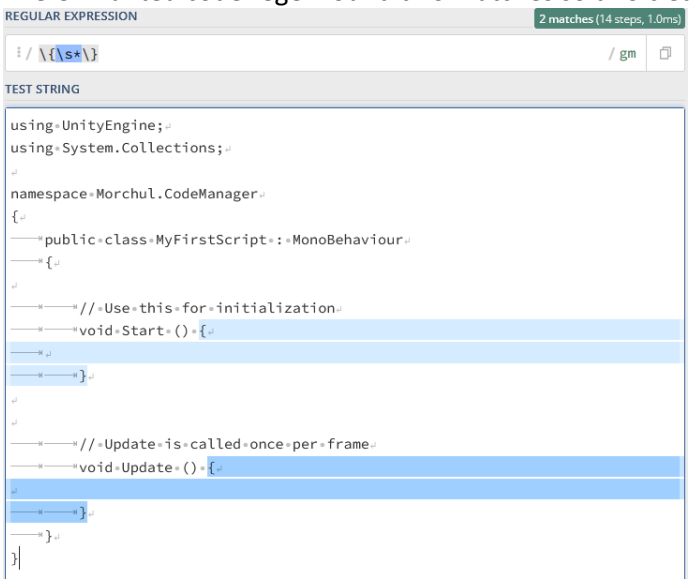
Every time the regex of the unwanted code will find a match in a script a clean code violation will be created.

Unwanted Code Example:

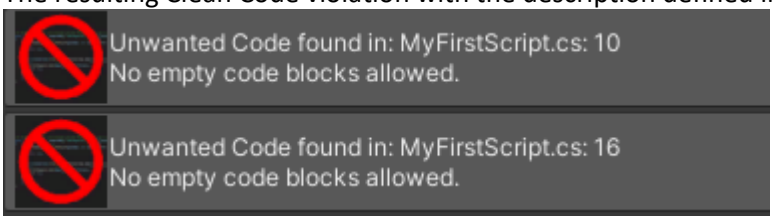
Unwanted Code	
Name	Empty Codeblock
Description	No empty code blocks allowed
Regex	Empty Codeblock regex

Regex	
Name	Empty Codeblock regex
Regex	\{\s*\}

The Unwanted code regex found two matches so two clean code violations will be created.

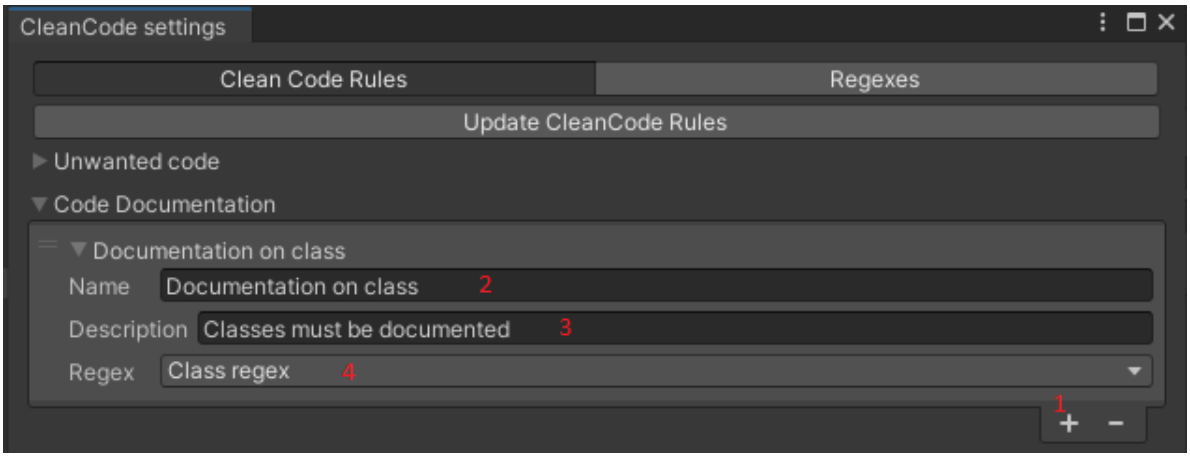


The resulting Clean Code violation with the description defined in the Unwanted Code:



Code Documentation:

1. Open the Clean Code Settings under: Code Manager => Clean Code => Settings.
2. Open the Code Documentation list and add a new item (1)
3. Enter the name of the code documentation rule in the Name field (2). Enter a description in the Description field to explain the rule (3). Finally, select a Regex you have created which matches the code which should be documented (4).



Every time the code documentation regex will find a match, the code before this match will be tested with the Documentation Regex found at the top of the Regexes Tab (1). If the code before does not match the Documentation Regex a clean code violation will be created.

The default Documentation Regex matches a documentation if the last line before the code to document starts with two or more “/”.

```
/// <summary>
/// Simple documentation
/// </summary>
public class MyFirstScript : MonoBehaviour
{
```

```
//Simple documentation
public class MyFirstScript : MonoBehaviour
{
```

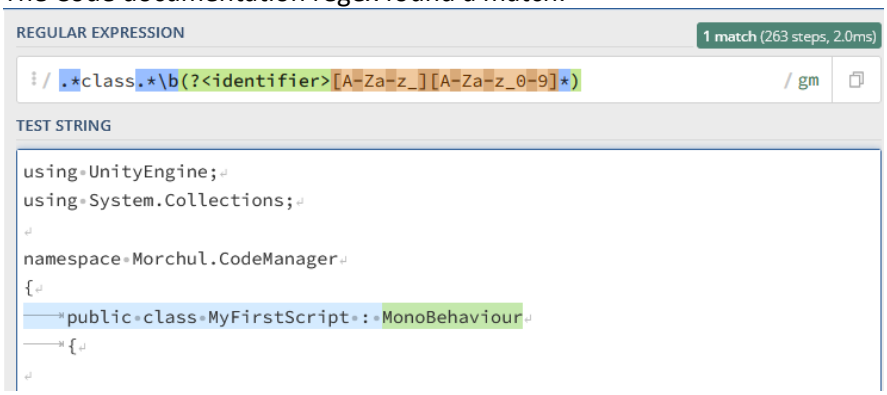
Code Documentation Example:

Code Documentation	
Name	Documentation on class
Description	Classes must be documented
Regex	Class Regex

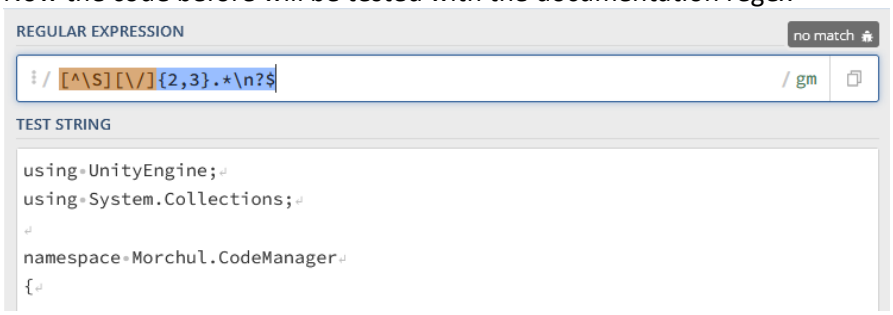
Regex	
Name	Class Regex
Regex	.*class.*\b(?<identifier>[A-Za-z_][A-Za-z_0-9]*)

Documentation Regex	[^\S][\/]{2,3}.*\n?$

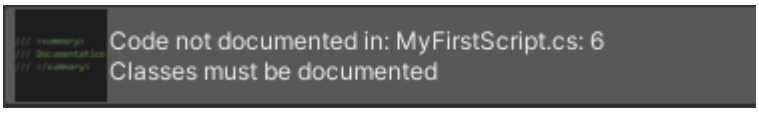
The Code documentation regex found a match:



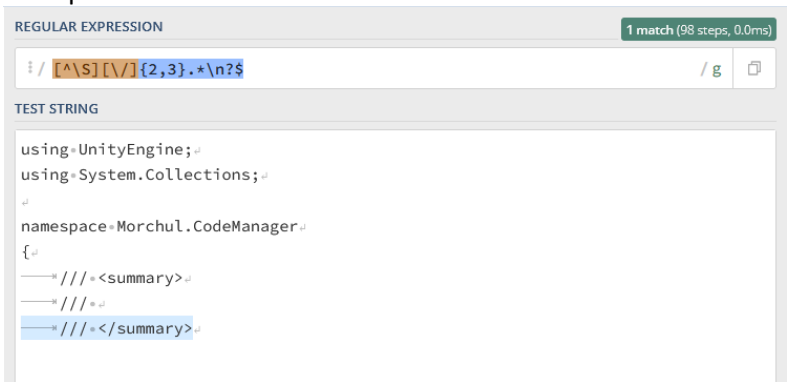
Now the code before will be tested with the documentation regex



Now match was found so there is no documentation and a clean code violation will be created

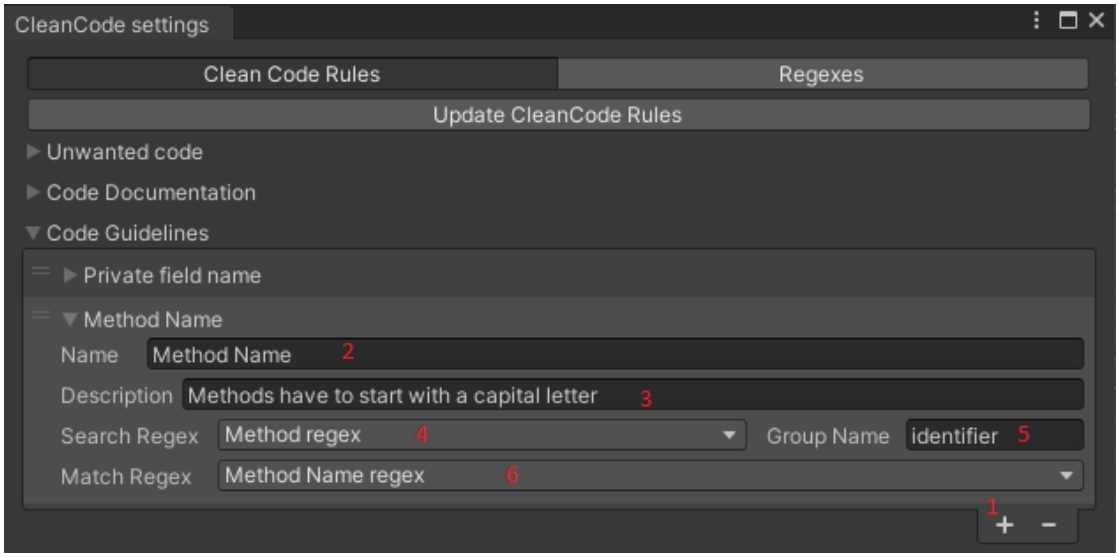


Example of documented code



Code Guideline:

1. Open the Clean Code Settings under: Code Manager => Clean Code => Settings.
2. Open the Code Guideline list and add a new item (1)
3. Enter the name of the code guideline rule in the Name field (2). Enter a description in the Description field to explain the rule (3). Select a search Regex you have created which matches the code where the code guideline rule applies (4). This regex should contain a group name. Write the group name you want to check for your code guideline rule in the Group Name field (5). Finally, select the Match Regex (6) with which the group value will be tested.



Code guideline rules work the following: First, all will be searched with the Search Regex. If the Search Regex finds a match the value of the defined group will be tested with the Match Regex and if there is no match a clean code violation will be created.
A short example: The Search Regex would match any method declaration. With the group you then can select only the name of the method and with the Match Regex you check if the method name corresponds to your code guideline rule.

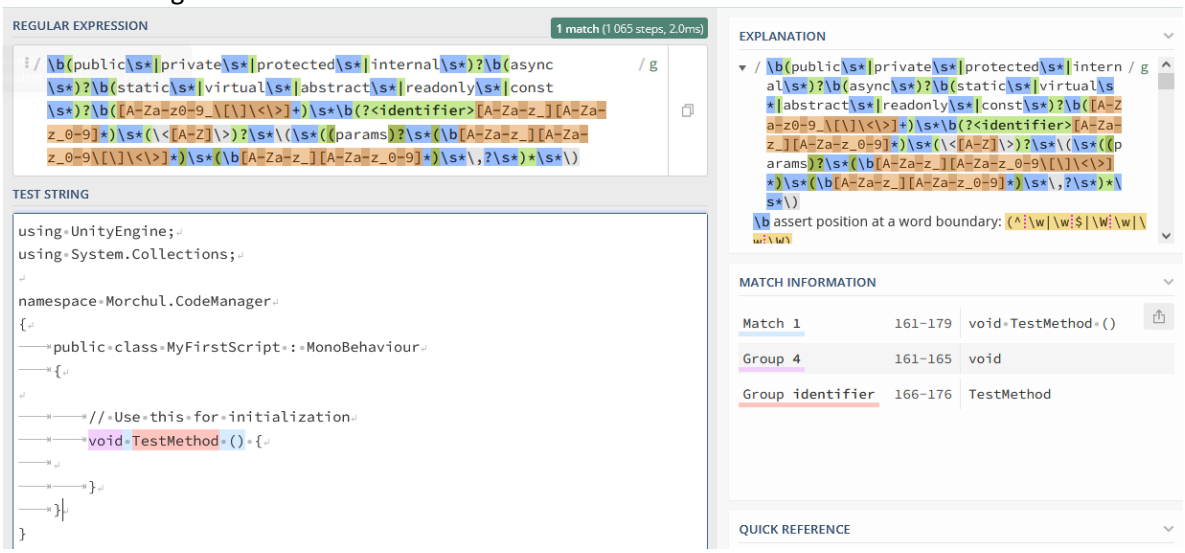
Code Guideline Example:

Code Guideline	
Name	Method Name
Description	Methods have to start with a capital letter
Search Regex	Method regex
Group Name	identifier
Match Regex	Method Name regex

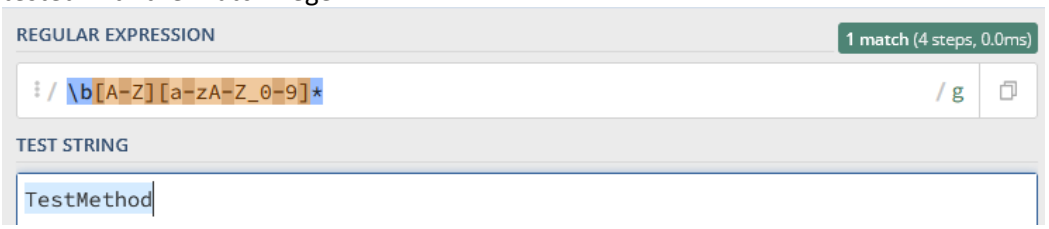
Regex	
Name	Method Regex
Regex	\b(public\s*|private\s*|protected\s*|internal\s*)?\b(async\s*)?\b(static\s*|virtual\s*|abstract\s*|readonly\s*|const\s*)?\b([A-Za-z0-9_\[\]\<\>]+)\s*\b(?<identifier>[A-Za-z_][A-Za-z_0-9]*)\s*(\<[A-Z]\>)?\s*\(\s*((params)?\s*(\b[A-Za-z_][A-Za-z_0-9\[\]\<\>]*)\s*(\b[A-Za-z_][A-Za-z_0-9]*)\s*\,?\s*)*\s*\)

Regex	
Name	Method Name Regex
Regex	\b[A-Z][a-zA-Z_0-9]*

The Search Regex found a match

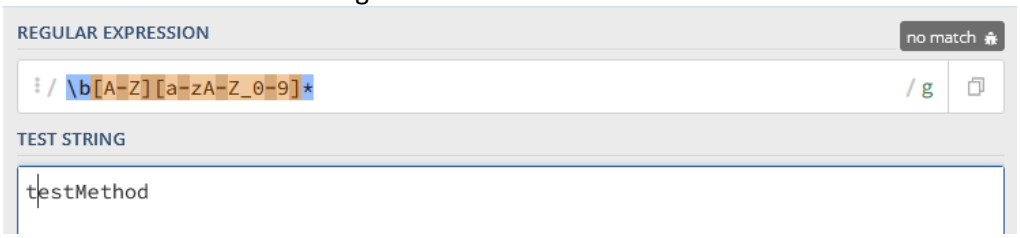


As we can see on the right the value of our group identifier is “TestMethod”. Now this value will be tested with the Match Regex.



There is one match, so the value corresponds to the code guideline.

If there would be no match e.g.:

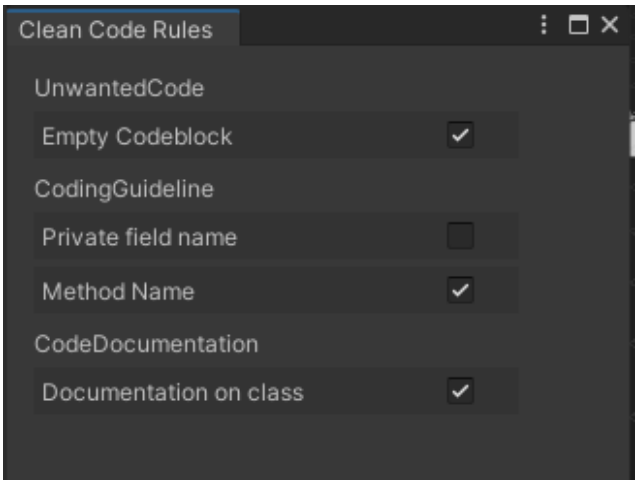


A clean code violation would be created:

5.2.3 Select Clean Code Rules

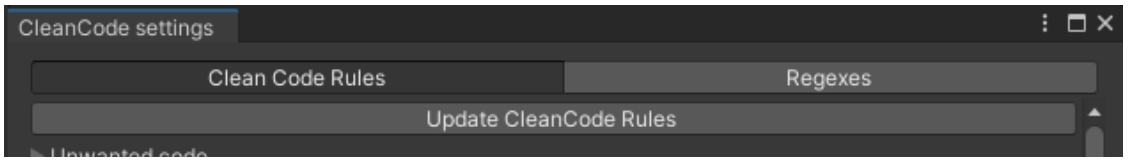
Now after the creation of Clean Code rules you must choose for which clean code rules should be scanned. You do this on each script folder individual.

1. Open Script Templates settings under: Code Manager => Script Templates => Settings.
2. Expand the Script Folders list and the folder you want to edit.
3. Press the Button “Select CleanCode rules for this folder”
4. Select all Clean Code Rules for which you want to scan in this folder (No subfolders)



If you do not see your Clean Code rule to select. There can be two reasons.
First, your Clean Code rule is not valid. This is indicated by red fields in your clean code rule:

Second, your Clean Code rules are not updated. The Clean Code rules will be updated if you close the Clean Code settings window or if you press the “Update CleanCode Rules” button at the top in the Clean Code Settings.

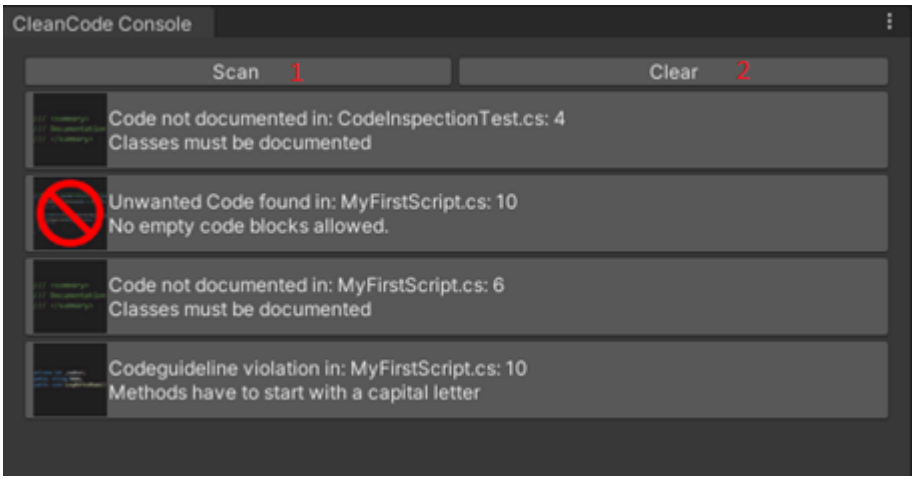


5.2.4 Clean Code Console

You can open the Clean Code Console under: Code Manager => Clean Code => Console.
The console has two Buttons: Scan (1) to scan all folders for Clean Code violations and Clear (2) to clear the console.

All clean code violations will be displayed in the console. A clean code violation contains the type of the violation, the file name and line index where the violation occurred and the description of the rule which was broken.

By clicking on a clean code violation, the script will be opened at the place of the clean code violation.



5.3 Code Inspector

The Code Manager contains a Tool to find and replace parts of a either plain text or text in a file. This tool can be accessed by code in your own project. The main classes are CodeInspector, CodeInspection and CodePiece.

There are some examples under:
Plugins/Morchul/CodeManager/Examples/CodeInspectorExamples.cs

6 Additional information

6.1 Regex

Regexes are a very important part in the Code Manager, and it is important that the user has some basic knowledge about it. Here are some links to learn simple Regex and what are they capable of:

https://en.wikipedia.org/wiki/Regular_expression (Definition of Regex)

https://docs.microsoft.com/en-us/dotnet/api/system.text.regularexpressions.regex?view=net-5.0
https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-language-quick-reference
https://www.softwaretestinghelp.com/csharp-regex-tutorial/
https://regex101.com/ (Online Regex Tester)

Demonstration (Scene)

Description of components
Schema
Providers
API
Code examples
Third-party support