Code Manager

Manual

09.12.2021
Version Prototype

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

Either Import the Unity package in your project or download the tool from GitHub: https://github.com/Morchul/CodeManager place the downloaded folder under Assets/Plugins/ in your project.

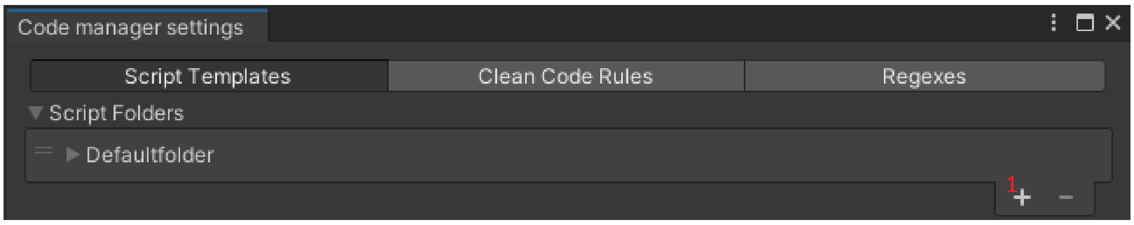
5 Features

5.1 Script Templates

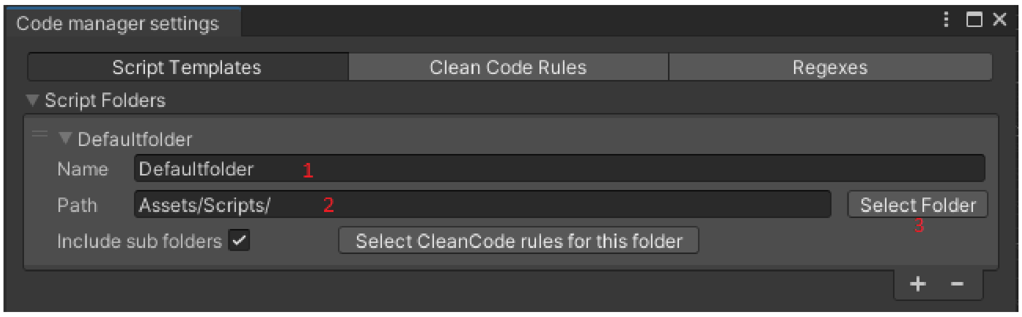
With Script Templates you create templates which can contain placeholders. In the settings you can give every placeholder a value. You then can create a new script from one of these templates and during script creation the placeholders will be replaced through your defined values. With this you can use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

5.1.1 Define script folders

1. Open the Code Manager settings under: Code Manager => 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.



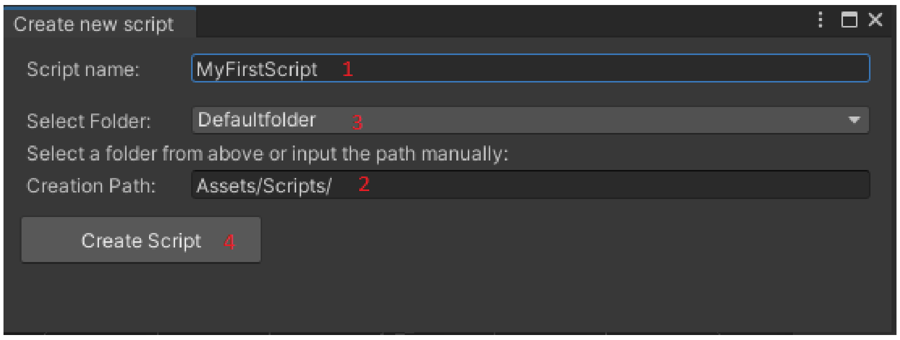
5.1.2 Create script templates

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 the path where the script should be created (2). You can either select one of the script folders (3) which will set the path or write the path manually. The folders to select are the script folders created in the settings.
4. Press the Create Script (4) button to create the new script.

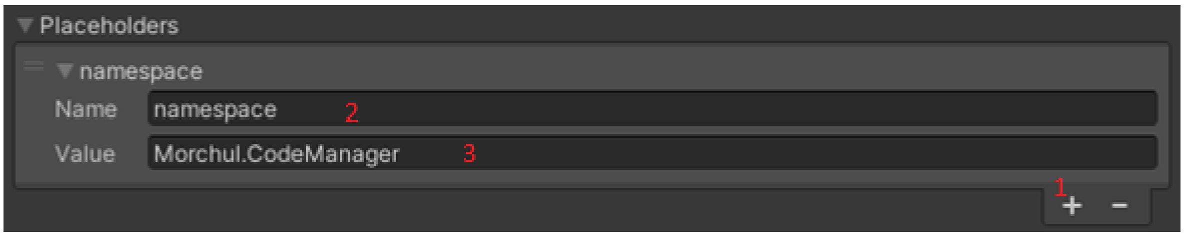


You can also open the Script create window by right click on a folder in your Project and navigate to the menu: Create => Script from template. The path will then automatically be filled with the path of the folder where you right clicked.

5.1.4 Placeholders

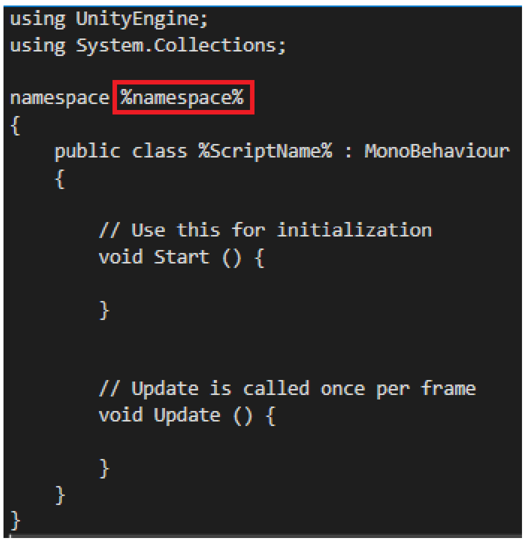
To create a new placeholder, open the Code Manager settings under: Code Manager => Settings. Open the Placeholders list and add a new item (1).

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)

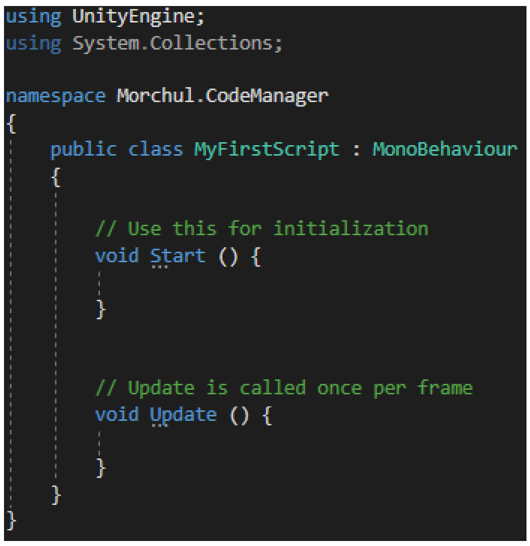


Name: namespace (2)
Value: Morchul.CodeManager (3)

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:

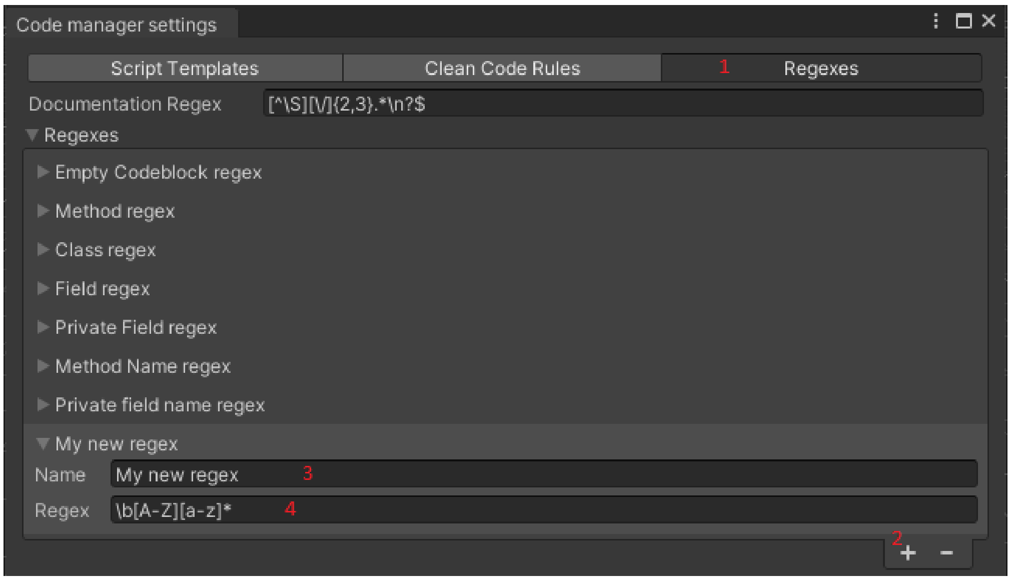
- **Unwanted Code**: You can define a Regex for code you do not want. If this regex matches anything, it will be displayed to you in the clean code console as clean code violation.
- **Code Documentation**: You can define a Regex for code you want to have documented. If this regex matches something the 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 Code Manager settings under: Code Manager => 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.



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

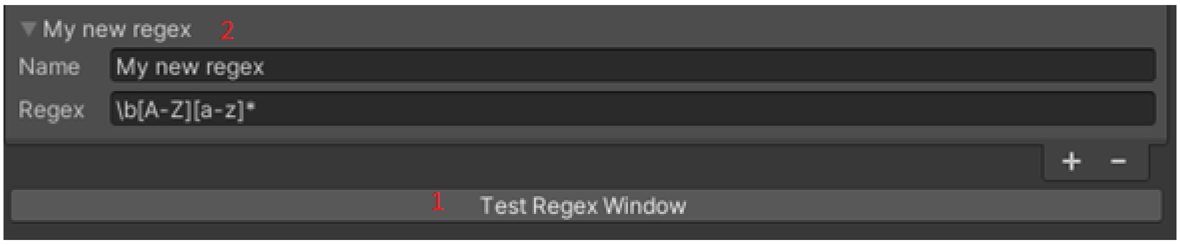
Name: My new regex

Regex: \b[A-Z][a-z]*

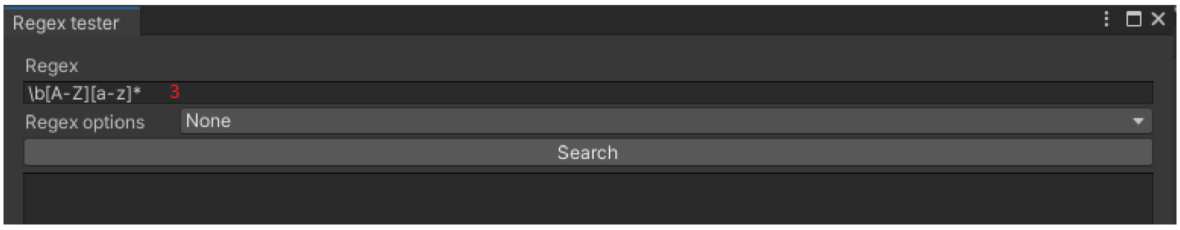
5.2.2 Test Regex Window

With the Test Regex Window, you can check your regexes and simple update them.

To open the Test Regex Window, press the “Test Regex Window” (1) Button under the Regexes List. It is important which regex in the Regexes list is selected when you press the button. The selected Regex (2) will be automatically filled in the Regex field (3) when the window opens. Also, if you close the window and you made any changes to the Regex the Test Regex Window will ask if you want to replace the regex with the new one. If you press yes, the selected regex will be replaced by the new regex.

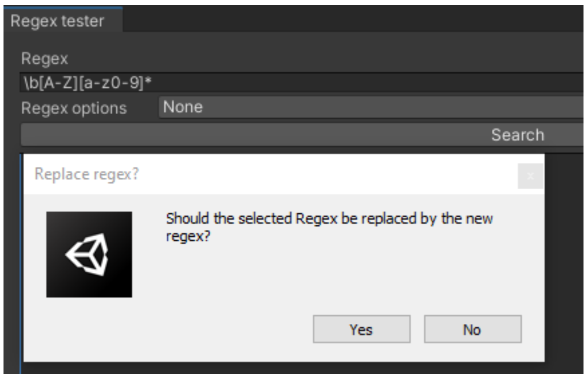


Name: My new regex
Regex: \b[A-Z][a-z]*



```
\b[A-Z][a-z]*
```

Regex options: None

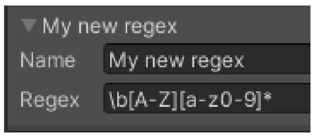


```
\b[A-Z][a-z0-9]*
```

Regex options: None

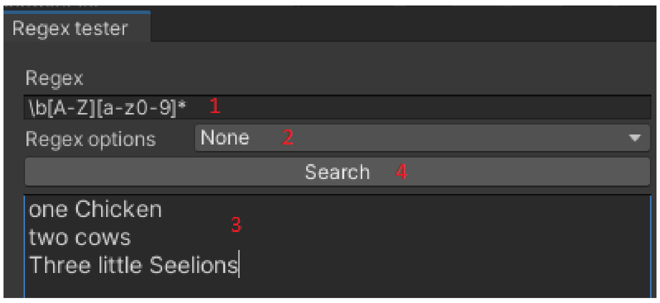
Replace regex?

Should the selected Regex be replaced by the new regex?



Name: My new regex
Regex: \b[A-Z][a-z0-9]*

In the Test Regex Window, you can input a regex (1), select regex options (2) and input some text (3). A search will be started automatically if the regex or text have changed. You can also press the Search button (4) to start a search.

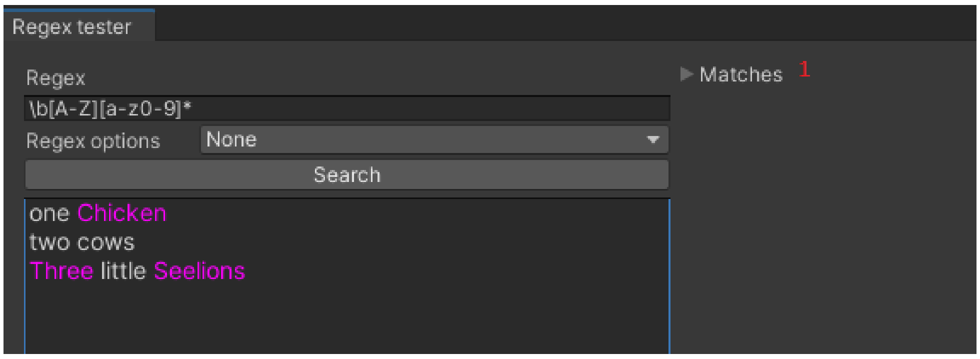


```
\b[A-Z][a-z0-9]*
```

Regex options: None

```
one Chicken
two cows
Three little Seelions
```

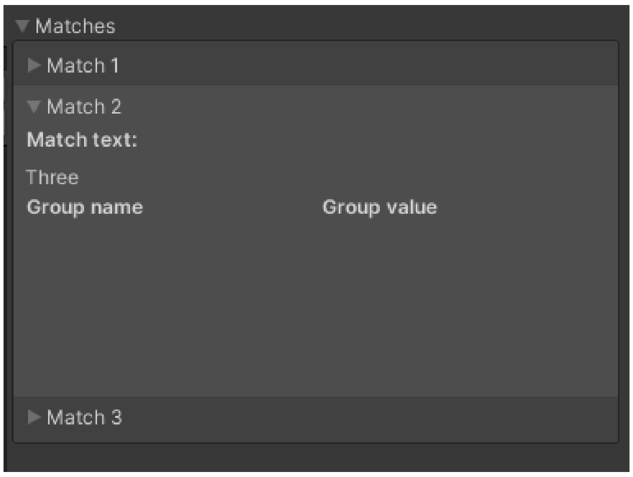
If a match was found, you will see a Matches List appearing on the right side (1). The text will also take some color where the match was found. The color changes will only appear after the text field loses focus.



Regex
```
\b[A-Z][a-z0-9]*
```
Regex options: None

```
one Chicken
two cows
Three little Seelions
```

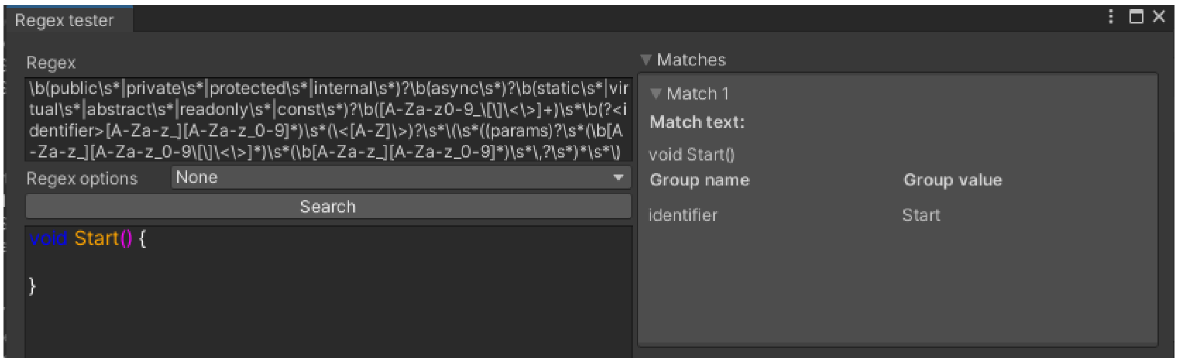
If you expand the Matches list, you find every match which was found in the text. Expand a match to see which text exactly matched and which groups were found is groups were specified. If groups were specified, you also see a difference in the color highlighting.



Match text:

Three

Group name	Group value



Regex
```
\b(public\s*|private\s*|protected\s*|internal\s*)?\b(async\s*)?\b(static\s*|vir
tual\s*|abstract\s*|readonly\s*|const\s*)?\b([A-Za-z0-9_\[\]\<\>]+)\s*\b(?<i
dentifier>[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 options: None

```
void Start() {

}
```

Match text:

void Start()

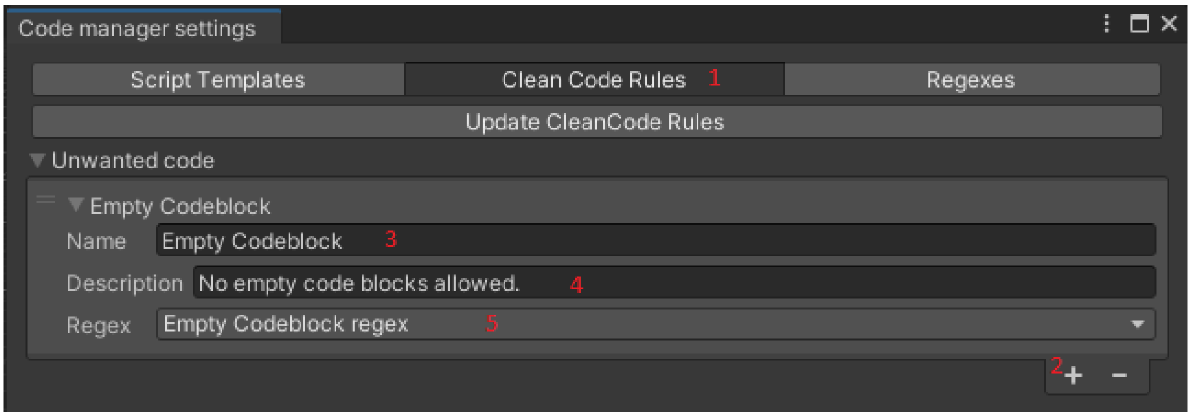
Group name	Group value
identifier	Start

5.2.3 Create Clean Code Rules

There are three different types of clean code rules. Unwanted code, code documentation and code guideline.

Unwanted Code:

1. Open the Code Manager Settings under: Code Manager => Settings.
2. Select the Clean Code Rules tab (1), open the Unwanted Code list and add a new item (2)
3. Enter the name of the unwanted code rule in the Name field (3). Enter a description in the Description field to explain the rule (4). Finally, select a Regex you have created which matches the unwanted code (5).



Name: Empty Codeblock
Description: No empty code blocks allowed.
Regex: Empty Codeblock regex

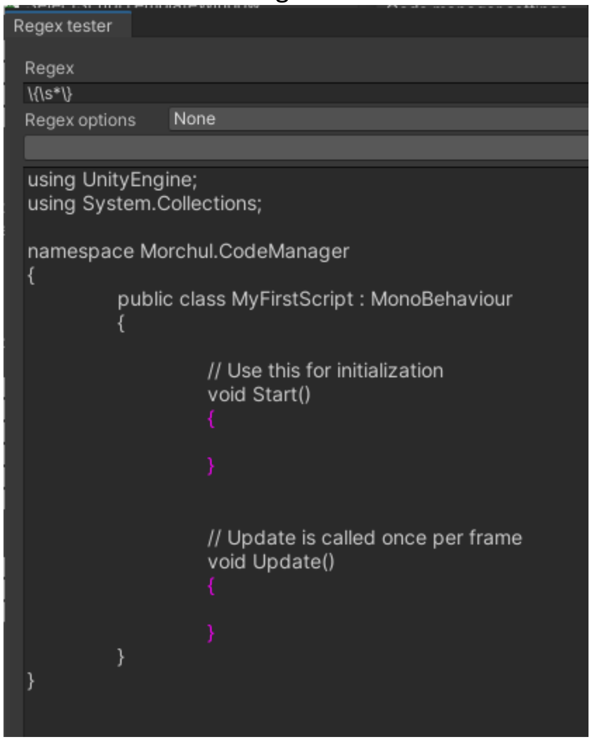
Every time the regex of the unwanted code will find a match in a script a clean code violation will be created.

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.

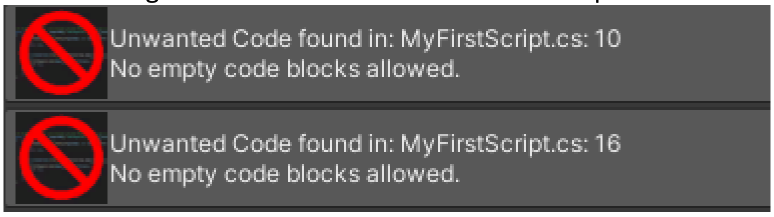


Regex tester

Regex
\{\s*\}
Regex options None

```
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()
		{

		}
	}
}
```

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

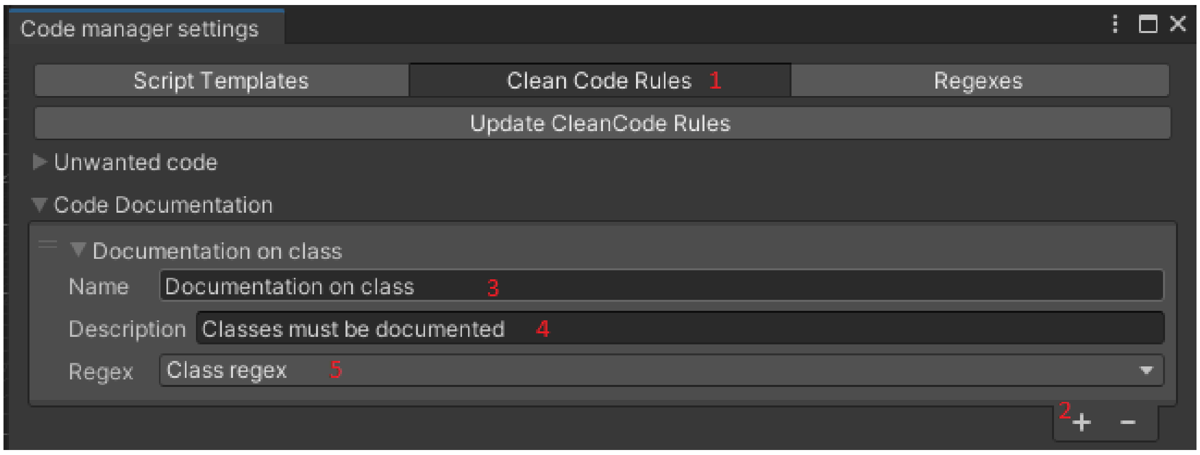


Unwanted Code found in: MyFirstScript.cs: 10
No empty code blocks allowed.

Unwanted Code found in: MyFirstScript.cs: 16
No empty code blocks allowed.

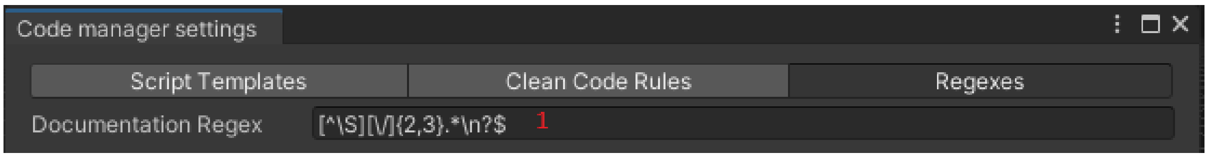
Code Documentation:

1. Open the Code Manager Settings under: Code Manager => Settings.
2. Select the Clean Code Rules tab (1), open the Code Documentation list and add an item (2)
3. Enter the name of the code documentation rule in the Name field (3). Enter a description in the Description field to explain the rule (4). Finally, select a Regex you have created which matches the code which should be documented (5).



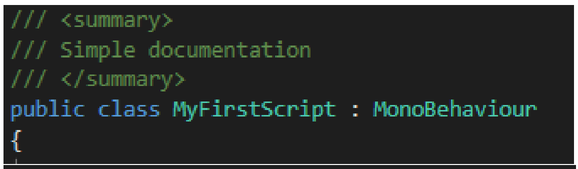
Name: Documentation on class 3
Description: Classes must be documented 4
Regex: Class regex 5

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.

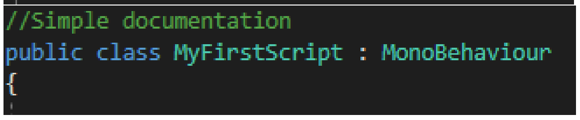


Documentation Regex: [^\S][\/]{2,3}.*\n?$ 1

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
{
```

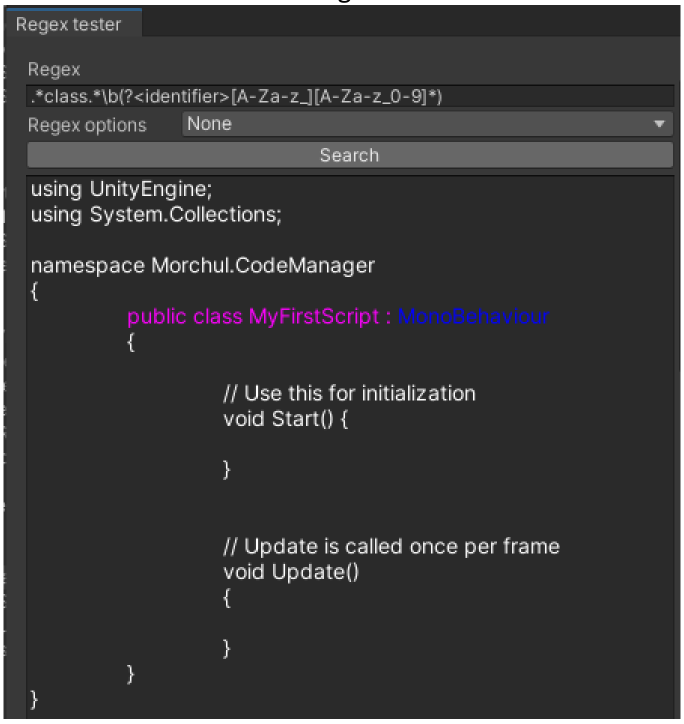
Code Documentation Example:

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?$

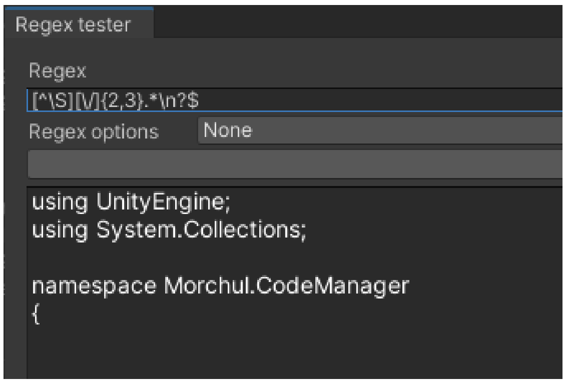
The Code documentation regex found a match:



Regex
.*class.*\b(?<identifier>[A-Za-z_][A-Za-z_0-9]*)
Regex options None

```
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()
                {

                }
        }
}
```

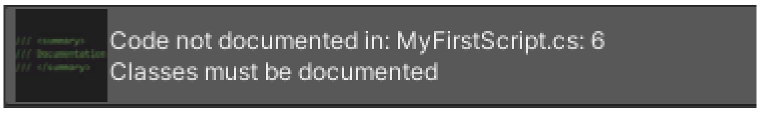
Now the code before will be tested with the documentation regex



Regex
[^\S][\/]{2,3}.*\n?$
Regex options None

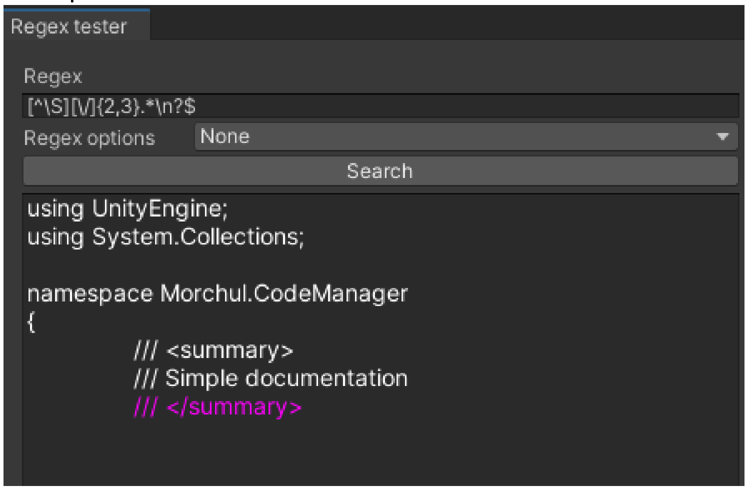
```
using UnityEngine;
using System.Collections;

namespace Morchul.CodeManager
{
```

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



Code not documented in: MyFirstScript.cs: 6
Classes must be documented

Example of documented code



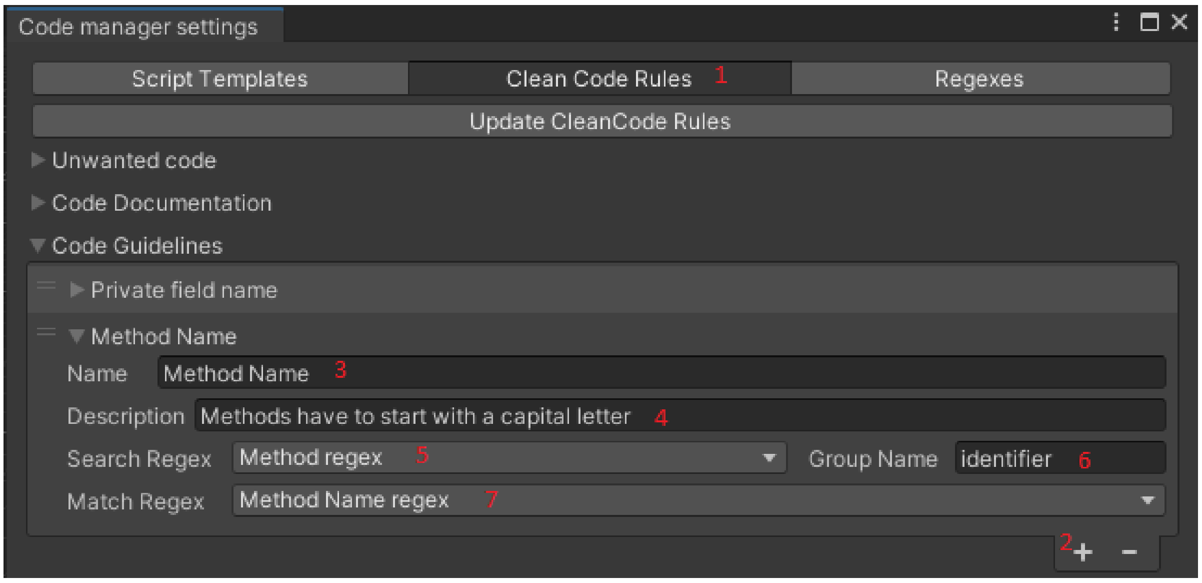
```
[^\S][\/]{2,3}.*\n?$
```

Regex options: None

```
using UnityEngine;
using System.Collections;

namespace Morchul.CodeManager
{
            /// <summary>
            /// Simple documentation
            /// </summary>
```

Code Guideline:

1. Open the Code Manager Settings under: Code Manager => Settings.
2. Select the Clean Code Rules tab (1), open the Code Guideline list and add a new item (2)
3. Enter the name of the code guideline rule in the Name field (3). Enter a description in the Description field to explain the rule (4). Select a search Regex you have created which matches the code where the code guideline rule applies (5). 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 (6). Finally, select the Match Regex (7) with which the group value will be tested.



Name: Method Name 3
Description: Methods have to start with a capital letter 4
Search Regex: Method regex 5
Group Name: identifier 6
Match Regex: Method Name regex 7

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.

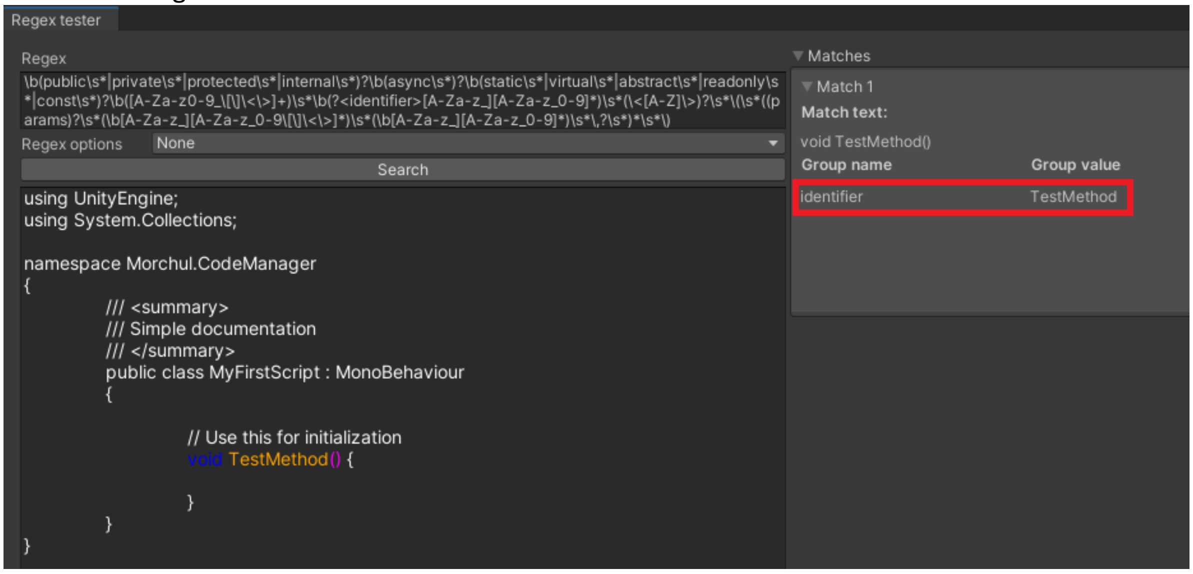
Code Guideline Example:

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



```
\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 options: None

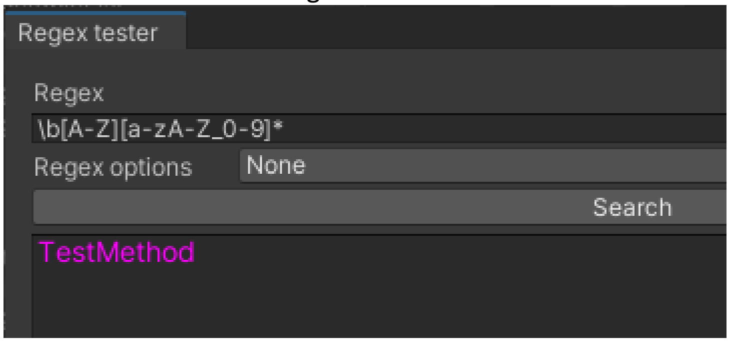
```
using UnityEngine;
using System.Collections;

namespace Morchul.CodeManager
{
        /// <summary>
        /// Simple documentation
        /// </summary>
        public class MyFirstScript : MonoBehaviour
        {

                // Use this for initialization
                void TestMethod() {

                }
        }
}
```

Match text:

void TestMethod()

Group name	Group value
identifier	TestMethod

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.



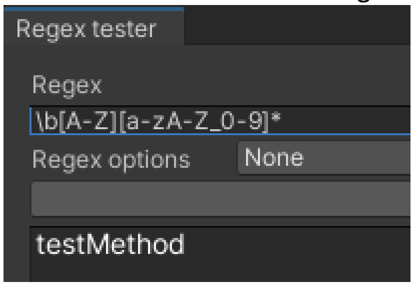
```
\b[A-Z][a-zA-Z_0-9]*
```

Regex options: None

TestMethod

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

If there would be no match e.g.:



Regex tester

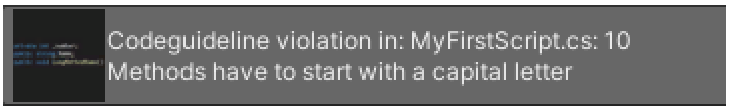
Regex

```
\b[A-Z][a-zA-Z_0-9]*
```

Regex options: None

testMethod

A clean code violation would be created:

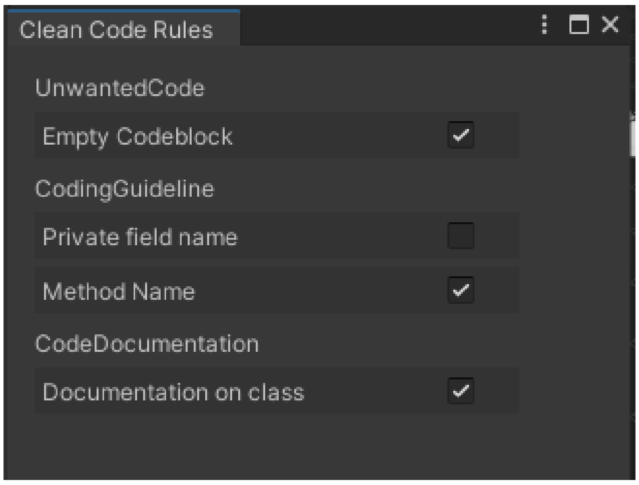


Codeguideline violation in: MyFirstScript.cs: 10
Methods have to start with a capital letter

5.2.4 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 => 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. If you have set Include sub folders the scans will also be done in sub folders for the selected rules.



UnwantedCode

Empty Codeblock	✓

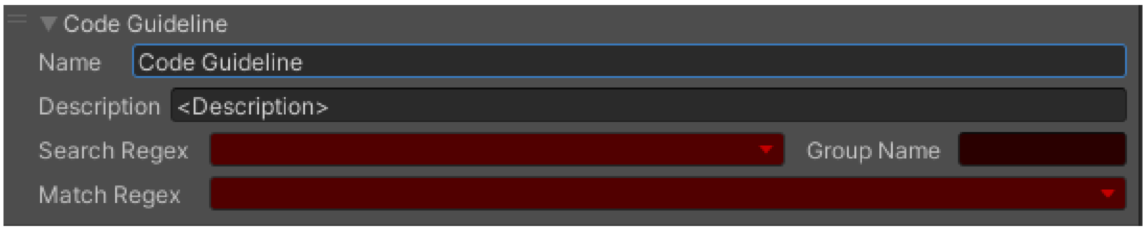
CodingGuideline

Private field name	
Method Name	✓

CodeDocumentation

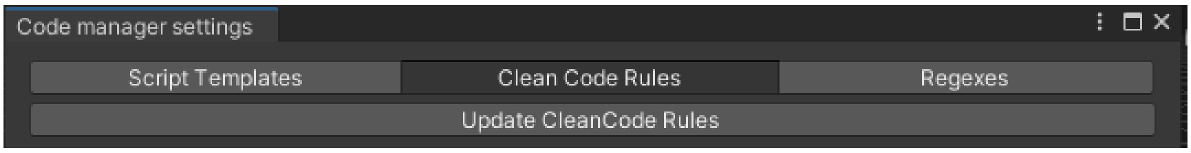
Documentation on class	✓

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:



Name: Code Guideline
Description: <Description>
Search Regex:
Group Name:
Match Regex:

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

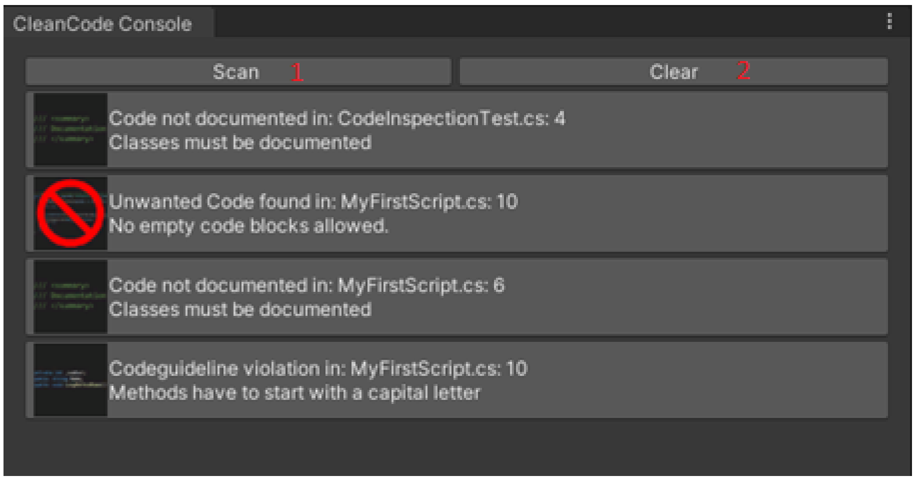


5.2.5 Clean Code Console

You can open the Clean Code Console under: Code Manager => Clean Code => Console. Or Ctrl + M
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.



Code not documented in: CodeInspectionTest.cs: 4
Classes must be documented

Unwanted Code found in: MyFirstScript.cs: 10
No empty code blocks allowed.

Code not documented in: MyFirstScript.cs: 6
Classes must be documented

Codeguideline violation in: MyFirstScript.cs: 10
Methods have to start with a capital letter

5.3 Code Inspector

The Code Manager contains a Tool to find and replace parts of an 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