# Exercises: High-Quality Methods

This document defines the **in-class exercises** assignments for the ["High-Quality Code" course @ Software University](https://softuni.bg/courses/high-quality-code).

## .NET Reference Source

Examine **at least three** examples in the .NET Reference source code (<http://referencesource.microsoft.com>). Look at their usage of methods. Document your findings. If you find things you don't like, document them too.

See the example below:

|  |  |  |
| --- | --- | --- |
| **Class / Method** | **Link** | **Notes** |
| System.IO.File | <http://referencesource.microsoft.com/#mscorlib/system/io/file.cs> | Methods are clearly written, not too long, have good names… |
| System.IO.File | <http://referencesource.microsoft.com/#mscorlib/system/io/file.cs> | Exception handling – the framework uses helper classes to throw exceptions |
| … | … | … |

Note that some things you might not like in the code may have been part of the Microsoft source code conventions (for example, Microsoft used to have a convention to write opening curly braces on the same line, not on their own line). However, feel free to document anything you like and don't like with the code.

Some things to look for:

* Method length – should not be too long
* Cohesion and coupling
* Consistent level of abstraction
* Duplicated code
* Correct usage of parameters – method does not depend on external parameters and does not change its parameters
* Exception handling

You can additionally look for things you have already learned:

* Code formatting
* Clear naming – variables, methods, namespaces, etc.
* Documentation and comments
* Variable usage
* Deep nesting of control structures
* Straight-line code

## Refactoring Bad Code

Find and document **at least three** examples of bad methods usage. You can look at popular source control repositories, such as **GitHub** (<https://github.com>), or **CodePlex** (<https://codeplex.com>), or you can look at curated examples of bad code: **GovnoKod** (<http://govnokod.ru>), **Bad Programming** (<http://badprogramming.com>), or **Reddit BadCode** (<http://reddit.com/r/badcode>).

The examples may be from the same code snippet (if it's long enough), or from different code snippets. You may also use the already provided code snippet.

Try and refactor the code where possible.

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Example** | **Problem** | **Solution** |
| 1 | <http://pastebin.com/tJyw8LjT> | Enormous method | This is hardly fixable. Extract several methods, use more meaningful names (readMsg -> the method does not only read a message, it performs many other tasks) |
| … | … | … | … |

## Is Recursion Always Necessary?

Write two implementations of the Fibonacci sequence: one with a loop, and one using recursion. Compare their performance. You can look at the **System.Diagnostics.Stopwatch** class for a way to calculate the time a method takes to run.

**Note:** If you copy the recursive implementation from the Internet, make sure it does not use an array to store values that have already been calculated. This is sometimes done as an optimization, but we need to compare the performance of pure recursion vs. loops.

Pass a few parameters and see how long it takes for the method to finish. Fill in the table below. Write "hangs" if the execution does not finish within 45-60 seconds.

You can optionally pass some more intermediate values and make a plot: place the parameter on the X axis, and the execution time on the Y axis.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Function** | **n = 10** | **n = 50** | **n = 100** | **n = 1000** | **n = 10 000** | **n = 100 000** | **n = 1 000 000** | **n = 10 000 000** |
| **Loop** |  |  |  |  |  |  |  |  |
| **Recursive** |  |  |  |  |  |  |  |  |

You may optionally repeat this to compare a loop and recursive implementation of factorial.