# Exercises: Naming Identifiers in the Code

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

## Writing New Code – the Good and the Bad Way

Log into the Software University Judge System. Select one of the past exams in the categories Programming Basics or Advanced C#:

* Programming Basics: <https://judge.softuni.bg/Contests/#!/List/ByCategory/1/Programming-Basics>
* Advanced C#: <https://judge.softuni.bg/Contests/#!/List/ByCategory/9/Advanced-CSharp>

Select one exam and solve one problem from it. It is better to choose a problem which: 1) you haven’t solved before, and 2) needs at least 30 lines of code to solve.

**Intentionally** use bad naming **first** – variable names, class names, method names, etc., for example **class Program**, **int a**, **string[] s**, etc.. Once the judge system gives you **at least half of the points** for the problem, refactor all names.

You should observe that choosing good and bad names takes almost the same time. What really eats up your time is trying to understand what your original intent was when you were writing the code.

Also, this is a simple project and you were working alone on it. What would happen if an entire system was written using bad naming? What if you were a part of a large international team where each member wrote code in their own language?

## Refactoring Your Own Code

Find one of your earliest projects, possibly one in which you didn't care about proper naming. Refactor the project so that it follows all requirements, conventions, recommendations and best practices for naming.

If your code is too good to find something like this ☺, you may use a class found on the internet.

## \* Code Review

Work in pairs. Use the code which you refactored. Show the initial and refactored codes to your teammate. Ask them for a review. It may be written or verbal.

A code review is a comment on your code. It should include things like: how much has the code quality improved after you refactored the code, is the code better to read and understand, are there any additional things you can refactor.

Switch roles. Give your teammate a code review for their own code.

## .NET Framework Source Code Naming

Look at the .NET Framework Reference Source, located at <http://referencesource.microsoft.com/> (you may want to examine the **mscorlib** assembly more closely). Look at all identifier names (files, assemblies, namespaces, classes, structures, properties, methods, local variables, parameters, etc.).

You should observe that the code is well-written and easy to read, understand and maintain.

**Optional:** Nobody's perfect and Microsoft also has its pitfalls. If you come across some example(s) of bad naming, document them. You can use the table below as a guide:

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Name** | **Corrected name** | **Reason** |
| System.MarshalByRefObject.InvokeMember ([http://referencesource.microsoft.com/#mscorlib/system/ marshalbyrefobject.cs,83](http://referencesource.microsoft.com/#mscorlib/system/marshalbyrefobject.cs,83)) | Type **t** = GetType(); | resultType | It is better to understand that "t" is actually a type and it comes as a result of the GetType() operation |
| … | … | … | … |

Something you consider bad code, may actually be a part of the code convention (for example, underscores before private field names). Nevertheless, document anything you are sure breaks the principles of high-quality code.

## Refactoring Others' Code

Find a code written by somebody else that has bad naming. You can search open source repositories, such as **GitHub** (<http://github.com>) or **CodePlex** (<http://codeplex.com>). If you want some specific examples of badly written code, you can look at one of the following sites: **GovnoKod** (<http://govnokod.ru>) **Bad Programming** (<http://badprogramming.com>), or **BadCode at reddit** (<https://www.reddit.com/r/badcode>).