# Exercises: Using Variables, Data, Expressions and Constants

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

## Document and Refactor Bad Code

Find **at least five** examples of bad variable 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.

Document these examples. Where possible, try to improve the code. You can look at an example comment below:

**Code:** <http://pastebin.com/RGAyFyGx>

|  |  |  |
| --- | --- | --- |
| **No.** | **Bad usage** | **Improvement Suggestion** |
| 1 | Settings\_Load(): bad naming: app1, …, app6; ico1, …, ico6 | Refactor variables to have more meaningful names.  Note: The code snippet is insufficient to provide better names |
| 2 | Public field Properties.Settings s | Remove the public field; either encapsulate it in a property or pass it as a method parameter where needed |
| 3 | … | … |

## Refactor Your Own Code

Find one of your exam problems or teamwork assignments and **refactor the variable usage** in it.

Some things to look for:

* Variable names – undescriptive, not meaningful names, misleading names
* Variable scope and span – too large scope, variables which have been initialized long before their first usage
* Variable lifetime – unnecessarily large variable lifetime; consider changing fields to method parameters
* Global variables – usually static variables which can be changed from too many places
* Variable visibility – ensure that you have provided the lowest visibility possible
* Cohesion and coupling – consider lowering the visibility, removing global variables, etc. to reduce the coupling between methods and classes
* Code formatting: ensure the code is well-formatted and can be easily read