# Exercises: Introduction to High-Quality Code

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

## Good Code

Examine one of the code examples below (or both). **You do not need to download anything**, just browse the code:

* Microsoft .NET Framework Reference Source:

<http://referencesource.microsoft.com/#mscorlib,namespaces>

You can examine some of the classes you are familiar with, for example:

* + **System.Collections.Generic.List<T>**: <http://referencesource.microsoft.com/#mscorlib/system/collections/generic/list.cs,cf7f4095e4de7646>
  + **System.Text.StringBuilder**: <http://referencesource.microsoft.com/#mscorlib/system/text/stringbuilder.cs,adf60ee46ebd299f>
  + **System.DateTime**:  
    <http://referencesource.microsoft.com/#mscorlib/system/datetime.cs,df6b1eba7461813b>
  + **System.Random**:  
    <http://referencesource.microsoft.com/#mscorlib/system/random.cs,bb77e610694e64ca>

You can also look at another class (or set of classes).

* OpenPOS: Open Source Point-Of-Sale System:

<http://openpos.codeplex.com/SourceControl/latest>

You can examine any class (or set of classes you wish). Some good examples to look at are:

* + Namespace **Trunk.OpenPOS.OpenPOSData.Models** (all classes)
  + Namespace **Trunk.OpenPOS.OpenPOSInfrastructure.Intefaces** (all interfaces)
  + Namespace **Trunk.OpenPOS.OpenPOSModules.Sales.Services**: CashupService.cs, PaymentService.cs

**You do not need to understand the architecture of the system.** Just look at the classes.

Try to understand what their purpose is, what practices have been used, etc.

Document all things you can find in a table. Follow the format below

|  |  |  |
| --- | --- | --- |
| **Feature** | **Comment** | **Example (optional)** |
| Comments in the code | Well-written comments, not too many; explain tricky points in the code | // A StringBuilder is internally represented as a linked list of blocks each of which holds  // a chunk of the string. It turns out string as a whole can also be represented as just a chunk,  // so that is what we do. |
| Code formatting | Well-formatted code, indentations help explain the code purpose  The code is separated into regions: variables at the top, followed by constructors, followed by methods | - |
| … | … | … |
| **Overall comment** | How does this code make me feel? What makes it a really well-written code? Can I understand what is going on? Can I change something on it easily? | |

## Bad Code

Look at the following example of bad (or not-so-good, if you wish ☺) code:

* <https://github.com/crazycat2701/Nhom19-TrangSuc/>

Look at some classes (for example, **Trangsuc.Models**, **Trangsuc.Controllers**) and fill a table, similar to the one in the previous problem. This time, highlight all problems / bad practices you find. Add a comment below the table explaining your overall feeling about the code: is it easy to **read**, **understand**, **extend** and **manage**?

**Hint 1:** Use the course outline for ideas (for example, code formatting, naming,)

**Hint 2:** Use the OOP principles for more ideas (does the project follow a clear structure, is there any encapsulation, etc.)

**You do not need to understand the architecture of the system.** Just look at the classes.

You can use other repositories if you wish.

## Improving Code Quality

Now that you have identified some issues in the code, try to suggest some improvements, based on your current knowledge. You can add a new column to the table from the previous problem, or make a new table, like so:

|  |  |  |
| --- | --- | --- |
| **Feature** | **Comment** | **Improvents** |
| Code comments written in Vietnamese | The comments could not be understood by many programmers | Rewrite the comments in English |
| Unhelpful / meaningless variable names | Some variables are named inappropriately – tt, vmp, etc. | Change variables:  tt -> numberOfClients  … |
| Very long methods | … | … |
| … | … | … |