TP#5: February 02nd – Good practices, Refactoring, Testing, API Call

The deadline for this TP is Friday 4th. The refactoring / testing aspect of the subject will be graded. The "API Call" part won't, that's just a bit of help for the final project. Questions answers should be in a specific file in the git repository, not in plein sign in the readme.

Exercise:

Download the worldly famous KATA: "Guilded Rose" (in C#). French requirement here.

Part 1: Questions

You can run the code, but you CANNOT modify it. Do not touch the code. Read it, and answer the following questions:

1. What is this code about?
2. Can you clearly identify the name of all the goods stored in the Guilded Rose?
3. What happens when the day is over?
4. What happens to cheese when the day is over?
5. What happens when a concert ticket goes over its expiration day?
6. What makes this code hard to read?
7. Do you think the rules are clear enough so that you could rework the entire solution from scratch?

Part 2: Tests

Look online for NUnit (and NFluent if you want) and AAA test pattern. Create a project in the solution (ie: GuildedRose.Test). To increase clarity, split your tests into multiple files in a way you see fit / clever.

Write some tests about this code. There should be about ~10 tests that should all pass and cover most of the cases. More tests the better. Question : What is the benefit of adding tests here ?

COMMIT YOU CODE AT THIS POINT. I'll look back in time in your git history to check that you did multiple commits.

Once you think you cover most of the code, we'll start part 3.

Part 3: Refactoring

Only now can you modify the GuildedRose.cs file. Before you rush in, ask yourself how you want to implement this in a clever way. I want different classes, single purpose function, objects. Tests should be green at the end, and you should not have modified them.

Part 4: Up, up, and away

Implement the conjured item property.

Then, find at least 3 ways to expand this code to add more functionalities. Explain how you would carry on with your implementation of the code. Thread carefully, this is harder than you think it is. Scalability is key. You don't need to code these, but you should explain in detail what you would do.

API Call:

If you're fast enough and that most of the Exercise is done, you can come to me and together we'll give a look at the CryptoCompare API, what the parameters in the url are like, how to make a simple API call, how to properly deserialize JSON into proper object, and maybe even how to work with the data. You're expected to have signed up to the API and have a proper API key by this point.