Louis Potterie

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

1. What is this code about?

The code is about an inventory management of an inn. The code updates quality and sell in of the products present in the inventory (represented in a list) at the end of each day. Some items followed specific rules for their quality and sell in characteristics.

2. Can you clearly identify the name of all the goods stored in the Guilded Rose?

We can easily identify goods stored for the moment with the List creation, but we do not have idea of all references that could be in this inventory.

Current goods:

- +5 Dexterity Vest
- Aged Brie
- Elixir of the Mongoose
- Sulfuras, Hand of Ragnaros
- Backstage passes to a TAFKAL80ETC concert
- Conjured Mana Cake

3. What happens when the day is over?

At the end of each day, the quality is update following the rules in the UpdateQuality function (that is different for each item) and the Sell in decrease by 1 for all items except the Sulfuras item.

4. What happens to cheese when the day is over?

The cheese when the day is over will improve his quality. When sellIn > 0 => cheese quality +1. When sellIn < 0 => cheese quality + 2

5. What happens when a concert ticket goes over its expiration day?

When a concert ticket goes over its expiration day the quality fall over to 0.

6. What makes this code hard to read?

Louis Potterie

The code is hard to read with the multiple if statement that will influence the sell in and the quality of the product. Also, item name is not clearly given because the code works with an Items List and last point UpdateQuality() uses a For loop and not a For each loop.

7. Do you think the rules are clear enough so that you could rework the entire solution

from scratch?

The rules are not enough clear to rework from scratch, I think a flowchart that represents all "If" possibility could be useful to rework from scratch. But we do not have a flowchart, so we will modify code step by step after to have created unit test and to be sure to respect the same logic.

Question: What is the benefit of adding tests here?

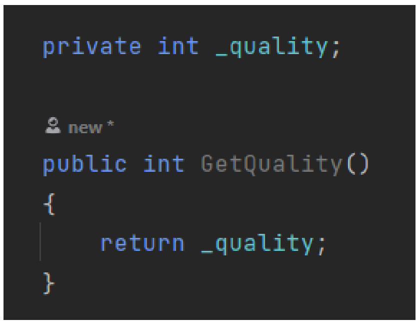
The goal of these tests is to check the functioning and behavior of the program in detail, to check if all the rules is well implemented and if all is secure with limit use case. And in a second time to verify that our code modifications have not affect the initial code logic.

Part 4:

The conjured functionality is added to the project.

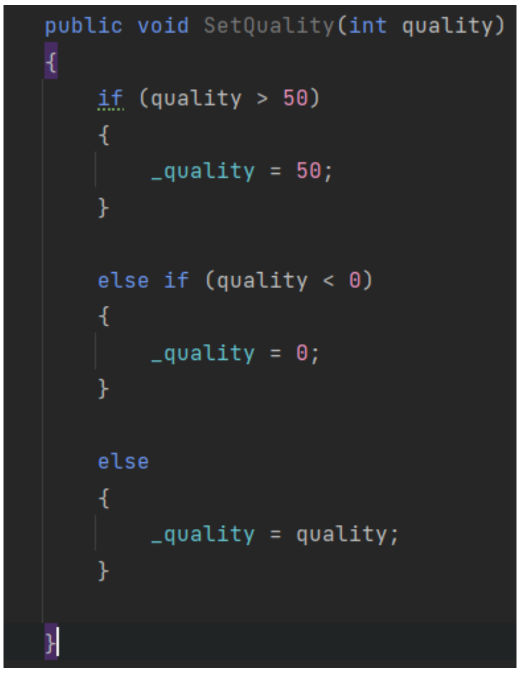
Evolution proposals:

First, I proposed to create conditional setter to be sure when a new item is added to the inventory (Items List), it respects rules about the Quality or Sell In characteristics. To illustrate my remark, I drafted the potential Getter / Setter of the Item Class:



```
private int _quality;

new *
public int GetQuality()
{
    return _quality;
}
```



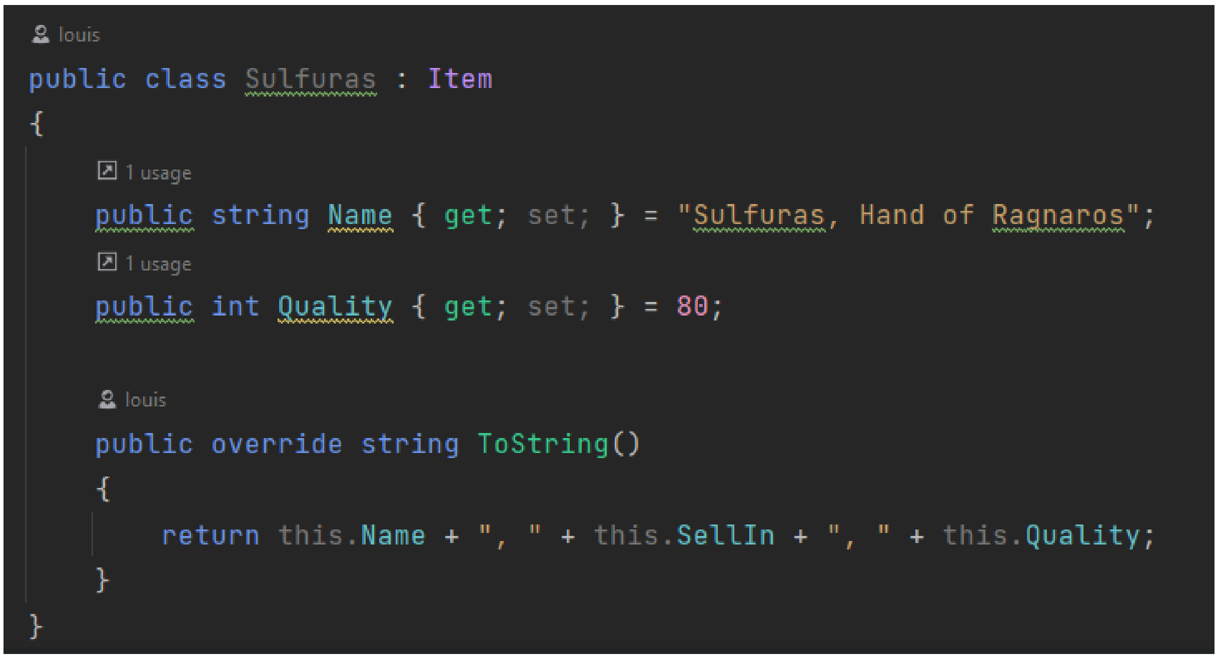
```
public void SetQuality(int quality)
{
    if (quality > 50)
    {
        _quality = 50;
    }

    else if (quality < 0)
    {
        _quality = 0;
    }

    else
    {
        _quality = quality;
    }

}
```

In a second time, we can create a class by type of Item inheriting from the Item class as following:

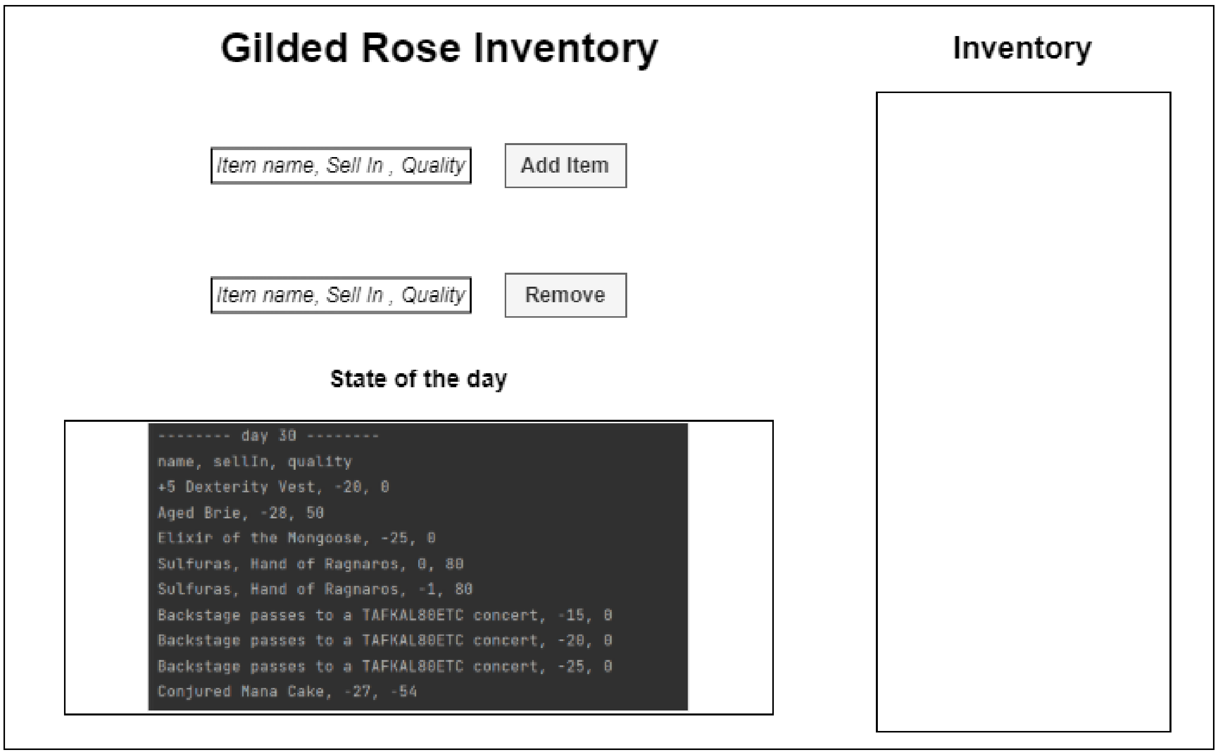


```
louis
public class Sulfuras : Item
{
    1 usage
    public string Name { get; set; } = "Sulfuras, Hand of Ragnaros";
    1 usage
    public int Quality { get; set; } = 80;

    louis
    public override string ToString()
    {
        return this.Name + ", " + this.SellIn + ", " + this.Quality;
    }
}
```

To create a class by type of item allow to directly apply their specific properties according to the class and will clarify the code because Items will be call by their Name explicitly.

We can also add a Graphical interface for a better management and visualization of the inventory. A first might look like as following:



```
-------- day 30 --------
name, sellIn, quality
+5 Dexterity Vest, -20, 0
Aged Brie, -28, 50
Elixir of the Mongoose, -25, 0
Sulfuras, Hand of Ragnaros, 0, 80
Sulfuras, Hand of Ragnaros, -1, 80
Backstage passes to a TAFKAL80ETC concert, -15, 0
Backstage passes to a TAFKAL80ETC concert, -20, 0
Backstage passes to a TAFKAL80ETC concert, -25, 0
Conjured Mana Cake, -27, -54
```

We have only two buttons, one for add item to the inventory with his characteristics and another button to remove an item by a triplet Name, Quality and Sell In.

At the bottom, we can find a console that will display the state inventory for the current day.

At the right, we will display all items in the inventory.