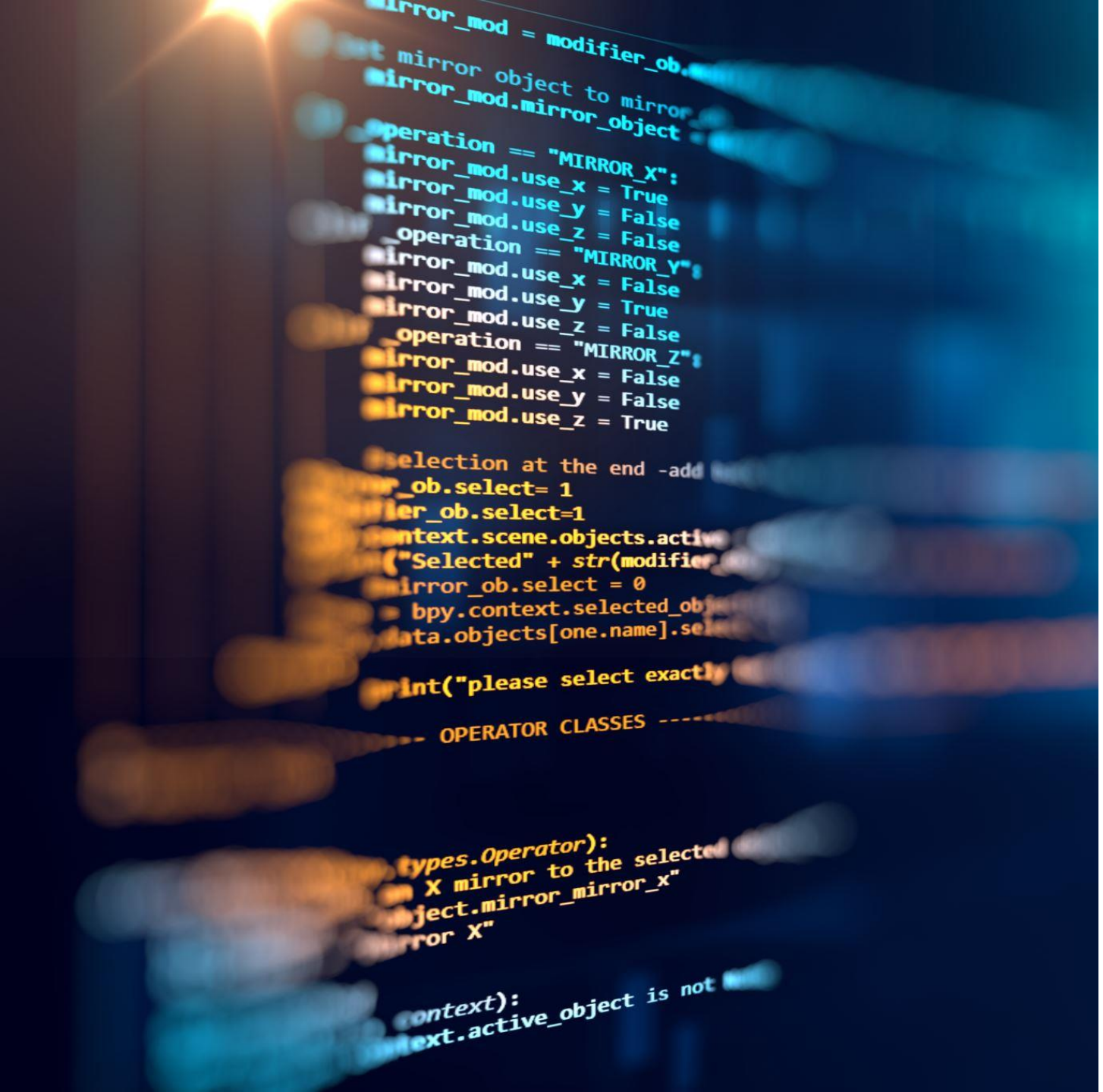
C#
The Gilded Rose

Rasmus Lystrøm
Associate Professor
ITU



Agenda

Refactoring kata

The Gilded Rose

Pointers

Code Coverage

Resources

Refactoring kata

Kata: Japanese word, literally meaning "*form*".

~ any basic form, routine, or pattern of behavior that is practiced to various levels of mastery

Practice makes perfect

Gilded Rose background

Hi and welcome to team Gilded Rose. As you know, we are a small inn with a prime location in a prominent city ran by a friendly innkeeper named Allison. We also buy and sell only the finest goods.

Unfortunately, our goods are constantly degrading in quality as they approach their sell by date.

We have a system in place that updates our inventory for us. It was developed by a no-nonsense type named Leeroy, who has moved on to new adventures.

Your task is to add the new feature to our system so that we can begin selling a new category of items. First an introduction to our system:

Gilded Rose specification

- All items have a **SellIn** value which denotes the number of days we have to sell the item
- All items have a **Quality** value which denotes how valuable the item is
- At the end of each day our system lowers both values for every item

Pretty simple, right? Well this is where it gets interesting:

- Once the sell by date has passed, **Quality** degrades twice as fast
- The **Quality** of an item is never negative
- "*Aged Brie*" actually increases in **Quality** the older it gets
- The **Quality** of an item is never more than 50
- "*Sulfuras*", being a legendary item, never has to be sold or decreases in **Quality**
- "*Backstage passes*", like *aged brie*, increases in **Quality** as it's **SellIn** value approaches; **Quality** increases by 2 when there are 10 days or less and by 3 when there are 5 days or less but **Quality** drops to 0 after the concert

We have recently signed a supplier of conjured items. This requires an update to our system:

- "*Conjured*" items degrade in **Quality** twice as fast as normal items

Implement “Conjured”

```
public void UpdateQuality()
{
    for (var i = 0; i < _items.Count; i++)
    {
        if (_items[i].Name ≠ "Aged Brie" && _items[i].Name ≠ "Backstage passes to a TAFKAL80ETC concert")
        {
            if (_items[i].Quality > 0)
            {
                if (_items[i].Name ≠ "Sulfuras, Hand of Ragnaros")
                {
                    _items[i].Quality = _items[i].Quality - 1;
                }
            }
        }
        else
        {
            if (_items[i].Quality < 50)
            {
                _items[i].Quality = _items[i].Quality + 1;

                if (_items[i].Name == "Backstage passes to a TAFKAL80ETC concert")
                {
                    if (_items[i].SellIn < 11)
                    {
                        if (_items[i].Quality < 50)
                        {
                            _items[i].Quality = _items[i].Quality + 1;
                        }
                    }

                    if (_items[i].SellIn < 6)
                    {
                        if (_items[i].Quality < 50)
                        {
                            _items[i].Quality = _items[i].Quality + 1;
                        }
                    }
                }
            }
        }

        if (_items[i].Name ≠ "Sulfuras, Hand of Ragnaros")
        {
            _items[i].SellIn = _items[i].SellIn - 1;
        }

        if (_items[i].SellIn < 0)
        {
            if (_items[i].Name ≠ "Aged Brie")
            {
                if (_items[i].Name ≠ "Backstage passes to a TAFKAL80ETC concert")
                {
                    if (_items[i].Quality > 0)
                    {
                        if (_items[i].Name ≠ "Sulfuras, Hand of Ragnaros")
                        {
                            _items[i].Quality = _items[i].Quality - 1;
                        }
                    }
                }
                else
                {
                    _items[i].Quality = _items[i].Quality - _items[i].Quality;
                }
            }
            else
            {
                if (_items[i].Quality < 50)
                {
                    _items[i].Quality = _items[i].Quality + 1;
                }
            }
        }
    }
}
```

"make the change easy (warning: this may be hard), then make the easy change"

Kent Beck, 2012

Refactoring

But first: **MAKE THEN CHANGE SAFE!**

Ensure 100% code coverage in **`GildedRose.cs`**
Test **`Main`** to verify that in generates the same output
(`dotnet run > output.txt`)

Now fix!

Extract method → Polymorphism

Code Coverage

```
dotnet add package coverlet.collector
dotnet add package coverlet.msbuild

dotnet test /p:CollectCoverage=true
```

```
+---------------------+------+---------+--------+
| Module              | Line | Branch | Method |
+---------------------+------+---------+--------+
| GildedRose          | 0%   | 0%      | 0%     |
+---------------------+------+---------+--------+
```

.NET Watcher

From the tests folder:

```
dotnet watch test /p:CollectCoverage=true
```

Resources

Martin Fowler on Refactoring:
https://martinfowler.com/articles/preparatory-refactoring-example.html

Sandy Metz on solving the Guilded Rose in Ruby:
https://youtu.be/8bZh5LMaSmE

The original Gilded Rose by Terry Hughes and Bobby Johnson:
https://github.com/NotMyself/GildedRose

More katas: https://kata-log.rocks/