## Gilded Rose

Wojciech Gołuński

https://github.com/dosia0804/Refaktoring-projekt

## Oryginał:

https://github.com/emilybache/GildedRose-Refactoring-Kata/tree/master/py thon

Rozwiązywanie kata refaktoryzacyjnego

## Sposób testowania - Text-Based Approval Testing

Ograniczenia kata - Nie można zmieniać klasy Item oraz zmiennej Items

```
def update_quality(self):
   for item in self.items:
       if item.name != "Aged Brie" and item.name != "Backstage passes to a TAFKAL80ETC concert":
            if item.quality > 0:
                    item.quality = item.quality - 1
            if item.quality < 50:
                item.quality = item.quality + 1
                    if item.sell in < 11:</pre>
                        if item.quality < 50:
                            item.quality = item.quality + 1
                    if item.sell_in < 6:</pre>
                        if item.quality < 50:
                            item.quality = item.quality + 1
       if item.name != "Sulfuras, Hand of Ragnaros":
            item.sell_in = item.sell_in - 1
       if item.sell in < 0:
            if item.name != "Aged Brie":
                    if item.quality > 0:
                        if item.name != "Sulfuras, Hand of Ragnaros":
                            item.quality = item.quality - 1
                    item.quality = item.quality - item.quality
                if item.quality < 50:
                    item.quality = item.quality + 1
```

Kod przed refaktoryzacją:

- -jedna metoda
- -do 6 zagłębień pętli i instrukcji warunkowych
- -poważne naruszenie single responsibility principle
- -gigantyczna ilość rozgałęzień kodu w jednej funkcji
- -powielanie tych samych warunków w rozgałęzieniach

```
def update_items(self):
    for item in self.items:
        self.update_sellin(item)
        self.update quality(item)
def update sellin(self, item):
    if item.name != "Sulfuras, Hand of Ragnaros":
        item.sell_in -= 1
def update quality(self, item):
    if item.name == "Sulfuras, Hand of Ragnaros":
        self.update_legendary_quality(item)
    elif item.name == "Aged Brie":
        self.update_cheese_quality(item)
   elif item.name == "Backstage passes to a TAFKAL80ETC concert":
        self.update_backstage_pass_quality(item)
    elif "Conjured" in item.name:
        self.update_conjured_quality(item)
        self.update_default_quality(item)
def modify quality(self, item, value):
    if item.quality + value > 50:
        item.quality = 50
    elif item.quality + value < 0:
        item.quality = 0
        item.quality+=value
def update_legendary_quality(self, item):
def update_cheese_quality(self, item):
    if item.sell_in < 0:</pre>
        self.modify_quality(item, 2)
        self.modify_quality(item, 1)
```

```
def update_backstage_pass_quality(self, item):
   if item.sell in < 10:
       self.modify quality(item, 2)
   elif item.sell in < 5:</pre>
        self.modify_quality(item, 3)
   elif item.quality < 0:
        self.modify quality(item, -item.quality)
        self.modify_quality(item, 1)
def update default quality(self, item):
   if item.sell in < 0:
       self.modify_quality(item, -2)
        self.modify quality(item, -1)
def update_conjured_quality(self, item):
   if item.sell in < 0:
        self.modify quality(item, -4)
        self.modify_quality(item, -2)
```

## Kod po refaktoryzacji:

- -Rozdzielenie długiej metody na 9 za pomocą extract method
- -Możliwość prostego dodania conjured item
- -Tylko jeden poziom zagnieżdżenia instrukcji warunkowych
- -Poprawiona czytelność
- -Każda funkcja jest odpowiedzialna tylko za jedną rzecz
- -Przechodzi test bazowany na tekście
- -Nie zostały naruszone ograniczenia kata