

Iwona Jaśtak, nr 246849

Moją refaktoryzującą GildedRose-Refactoring-Kata wykonałam w obiektowym języku Java. [Github](#)

1. Testy

Pierwszym krokiem jaki wykonałam było napisanie testów do każdego rodzaju przedmiotu. Testy pomogły mi podczas refaktoryzacji sprawdzać czy moje zmiany nie zmieniły działania początkowego programu. Dodatkowo połączyłam moje repozytorium z serwisem [Travis CI](#).

```
-----
T E S T S
-----
Running com.gildedrose.AgedBrieTest
Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.076 s - in com.gildedrose.AgedBrieTest
Running com.gildedrose.BackstagePassTest
Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.003 s - in com.gildedrose.BackstagePassTest
Running com.gildedrose.ConjuredItemTest
Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.003 s - in com.gildedrose.ConjuredItemTest
Running com.gildedrose.DefaultItemTest
Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.008 s - in com.gildedrose.DefaultItemTest
Running com.gildedrose.SulfurasTest
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.004 s - in com.gildedrose.SulfurasTest

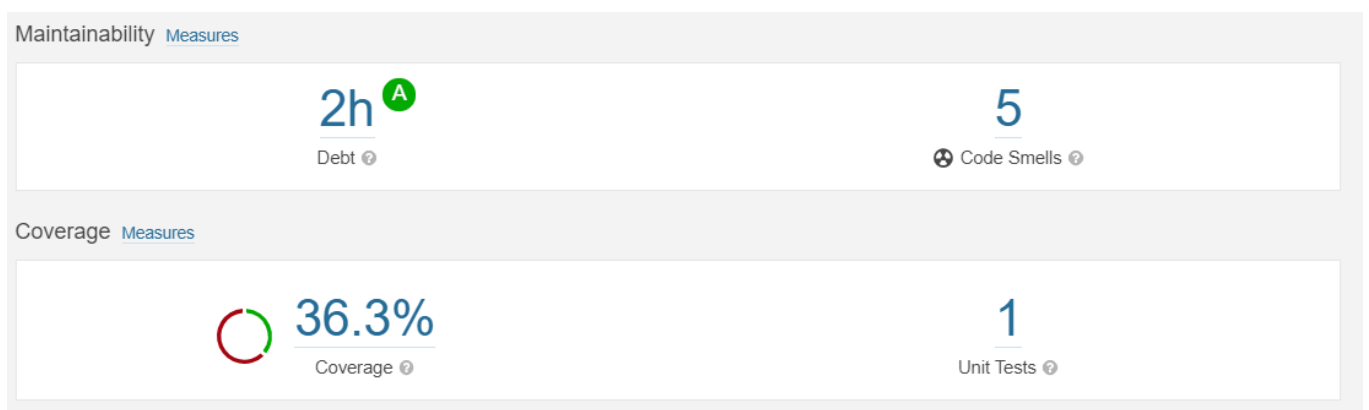
Results:

Tests run: 33, Failures: 0, Errors: 0, Skipped: 0
```

2. Metryki

Do analizy kodu wykorzystałam narzędzie [Sonarcloud](#), który między innymi wylicza złożoność programu, dług technologiczny oraz wykrywa code smelle.

Wyniki z Sonarcloud przed refaktoryzacją:



Pierwszym problemem, który najbardziej rzuca się w oczy jest zbyt wysoki dług technologiczny jak na tak mały program składający się z dwóch klas. Następnym problemem są code smelle pokazane poniżej:

1 / 5 issues

src/.../java/com/gildedrose/GildedRose.java

Refactor this method to reduce its Cognitive Complexity from 79 to the 15 allowed.

Code Smell +23

- 1 +1
- 2 +2 (incl 1 for nesting)
- 3 +1
- 4 +3 (incl 2 for nesting)
- 5 +4 (incl 3 for nesting)
- 6 +4 (incl 3 for nesting)
- 7 +1
- 8 +3 (incl 2 for nesting)
- 9 +4 (incl 3 for nesting)
- 10 +5 (incl 4 for nesting)
- 11 +6 (incl 5 for nesting)
- 12 +5 (incl 4 for nesting)
- 13 +6 (incl 5 for nesting)
- 14 +2 (incl 1 for nesting)
- 15 +2 (incl 1 for nesting)
- 16 +3 (incl 2 for nesting)
- 17 +4 (incl 3 for nesting)
- 18 +5 (incl 4 for nesting)
- 19 +6 (incl 5 for nesting)
- 20 +6 (incl 5 for nesting)
- 21 +1
- 22 +1
- 23 +4 (incl 3 for nesting)

alt + ↑ ↓ to navigate issue locations

Define a constant instead of duplicating this literal "Backstage passes to a TAFKAL80ETC concert" 3 times.

Code Smell +3

Define a constant instead of duplicating this literal "Sulfuras, Hand of Ragnaros" 3 times.

Code Smell +3

Merge this if statement with the enclosing one.

Code Smell +1

Merge this if statement with the enclosing one.

Code Smell +1

5 of 5 shown

gilded-rose-kata src/main/java/com/gildedrose/GildedRose.java See all issues in this file

```

8  iwona...
9
10 public void updateQuality() {
11
12     1 for (int i = 0; i < items.length; i++) {
13         2 if (!items[i].name.equals("Aged Brie"))
14             3 if (items[i].name.equals("Backstage passes to a TAFKAL80ETC concert")) {
15                 4 if (items[i].quality > 0) {
16                     5 if (items[i].name.equals("Sulfuras, Hand of Ragnaros")) {
17                         items[i].quality = items[i].quality - 1;
18                     }
19                     6 if (items[i].name.equals("Conjured Mana Cake")){
20                         items[i].quality = items[i].quality - 1;
21                     }
22                 }
23             } else {
24                 7 if (items[i].quality < 50) {
25                     items[i].quality = items[i].quality + 1;
26                 }
27                 8 if (items[i].name.equals("Backstage passes to a TAFKAL80ETC concert")) {
28                     9 if (items[i].sellIn < 11) {
29                         10 if (items[i].quality < 50) {
30                             items[i].quality = items[i].quality + 1;
31                         }
32                     }
33                     11 if (items[i].sellIn < 6) {
34                         12 if (items[i].quality < 50) {
35                             items[i].quality = items[i].quality + 1;
36                         }
37                     }
38                 }
39             }
40         }
41     }
42     14 if (items[i].name.equals("Sulfuras, Hand of Ragnaros")) {
43         items[i].sellIn = items[i].sellIn - 1;
44     }
45     15 if (items[i].sellIn < 0) {
46         16 if (!items[i].name.equals("Aged Brie")) {
47             17 if (items[i].name.equals("Backstage passes to a TAFKAL80ETC concert")) {
48                 18 if (items[i].quality > 0) {
49                     19 if (items[i].name.equals("Sulfuras, Hand of Ragnaros")) {
50                         items[i].quality = items[i].quality - 1;
51                     }
52                     20 if (items[i].name.equals("Conjured Mana Cake")){
53                         items[i].quality = items[i].quality - 1;
54                     }
55                 }
56             }
57             21 else {
58                 items[i].quality = items[i].quality - items[i].quality;
59             }
60         }
61         22 else {
62             23 if (items[i].quality < 50) {
63                 items[i].quality = items[i].quality + 1;
64             }
65         }
66     }
67 }

```

Dodatkowo by porównać jakość kodu wykorzystałam poniższe metryki:

- złożoność cyklomatyczna (ang. Cyclomatic Complexity) - jest wyliczana na podstawie liczby niezależnych ścieżek w programie
- złożoność poznawcza (ang. Cognitive Complexity) - jest to miara określająca jak kod jest trudny do czytania i zrozumienia.

Na powyższym zrzucie ekranu możemy zauważyć, że jednym z code smelli jest właśnie zbyt duża złożoność poznawcza dla metody `updateQuality()` w klasie `GildedRose`. Jest również przedstawione w jaki sposób jest to wyliczane.

3. Refaktoryzacja

Celem mojej refaktoryzacji było pozbycie się zagnieżdżonych ifów oraz zastąpienie ich polimorfizmem. Wykonałam to w następujących krokach:

1. Utworzenie klasy `DefaultItem` dziedziczącej po `Item` oraz zdefiniowanie działania zwykłego przedmiotu:

```

public class DefaultItem extends Item {
    private static final int MAX_VALUE = 50;
    private static final int MIN_VALUE = 0;

    public DefaultItem(String name, int sellIn, int quality) {
        super(name, sellIn, quality);
    }

    public void updateQuality() {
        decreaseSellIn();
        if(quality < MAX_VALUE && quality >= MIN_VALUE)
            changeQuality();
        if(sellIn < 0)
            changeQuality();

        checkQualityRange();
    }

    protected void checkQualityRange(){
        if(quality < MIN_VALUE)
            quality = 0;
        else if(quality >= MAX_VALUE)
            quality = 50;
    }

    protected void decreaseSellIn(){
        sellIn--;
    }

    protected void changeQuality(){
        quality--;
    }
}

```

2. Usunięcie powtarzających się fragmentów

```

if (!items[i].name.equals("Sulfuras, Hand of Ragnaros")) {
    items[i].quality = items[i].quality - 1;
}

```

oraz wydzielenie ich do osobnej klasy `Sulfuras` dziedziczącej po `DefaultItem` z nadpisaną metodą `updateQuality()`

```

@Override
public void updateQuality(){
    //nothing changes
}

```

3. Usunięcie fragmentu

```

if(items[i].name.equals("Conjured Mana Cake")){
    items[i].quality--;
}

```

oraz wydzielenie go do klasy `ConjuredItem` dziedziczącej po `DefaultItem` z nadpisaną metodą `changeQuality()`

```

@Override
public void changeQuality(){
    quality -= 2;
    checkQualityRange();
}

```

4. Usunięcie fragmentów

```

if (items[i].name.equals("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;
        }
    }
}

```

oraz

```

if (!items[i].name.equals("Backstage passes to a TAFKAL80ETC concert")) {
} else {
    items[i].quality = items[i].quality - items[i].quality;
}

```

oraz wydzielenie go do klasy `BackstagePass` dziedziczącej po `DefaultItem` z nadpisaną metodą `changeQuality()`

```

@Override
public void changeQuality(){
    if(sellIn <= 0)
        quality = 0;
    else if(sellIn <= 5)
        quality += 3;
    else if(sellIn <= 10)
        quality += 2;
}

```

```

else
    quality++;
    checkQualityRange();
}

```

5. Usunięcie pozostałej części

```

if (!items[i].name.equals("Aged Brie")
    && !items[i].name.equals("Backstage passes to a TAFKAL80ETC concert"))
    } else {
        if (items[i].quality < 50) {
            items[i].quality = items[i].quality + 1;
        }
    }

    if (items[i].sellIn < 0) {
        if (!items[i].name.equals("Aged Brie")) {
            } else {
                if (items[i].quality < 50) {
                    items[i].quality = items[i].quality + 1;
                }
            }
        }
    }
}

```

oraz wydzielenie jej do osobnej klasy `AgedBrie` dziedziczącej po `DefaultItem` z nadpisaną metodą `changeQuality()`

```

@Override
public void changeQuality(){
    quality++;
    checkQualityRange();
}

```

6. Zmiana pętli w `updateQuality()` w klasie `GildedRose`

```

public void updateQuality() {
    for (DefaultItem item : items) {
        item.updateQuality();
    }
}

```

7. Skorzystanie ze Factory method pattern oraz stworzenie fabryki produkującej przedmioty:

```

public class ItemFactory {
    public DefaultItem createItem(String type, int sellIn, int quality){
        switch(type){
            case "Sulfuras, Hand of Ragnaros":

```

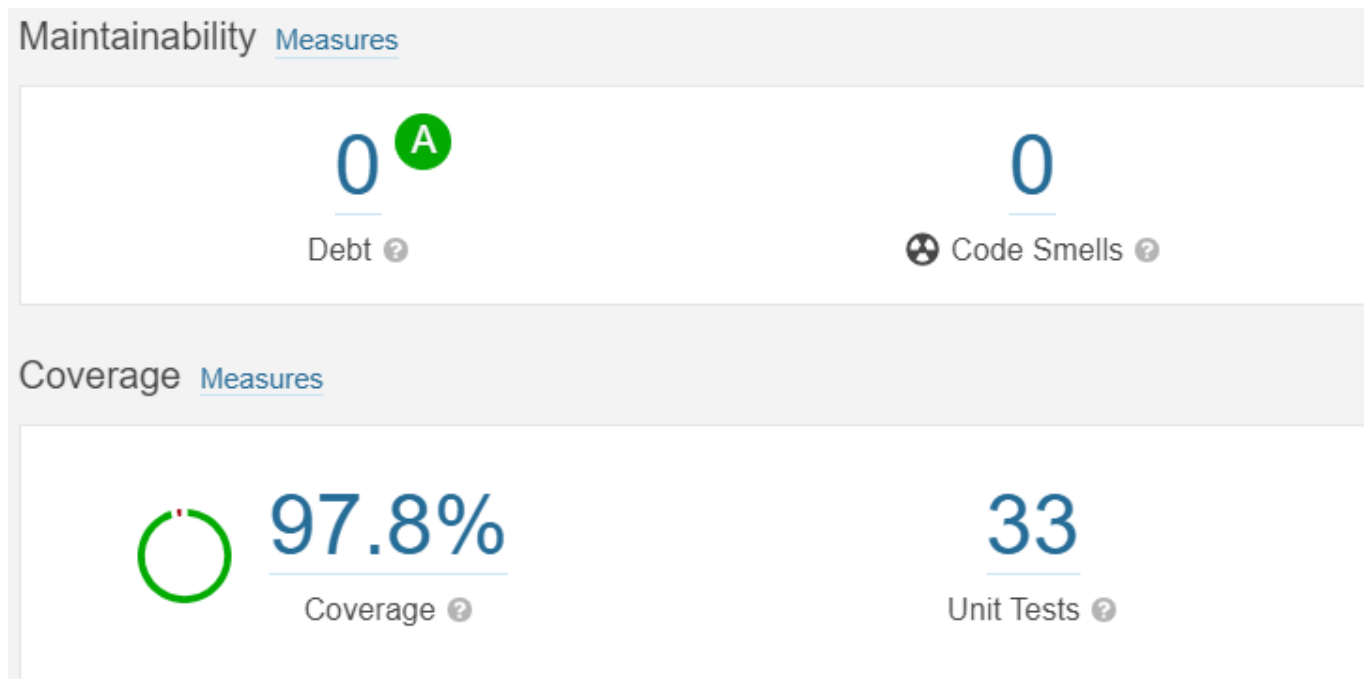
```

        return new Sulfuras(type, sellIn, quality);
    case "Conjured Mana Cake":
        return new ConjuredItem(type, sellIn, quality);
    case "Aged Brie":
        return new AgedBrie(type, sellIn, quality);
    case "Backstage passes to a TAFKAL80ETC concert":
        return new BackstagePass(type, sellIn, quality);
    default:
        return new DefaultItem(type, sellIn, quality);
    }
}
}

```










4. Po refaktoryzacji

Wyniki z Sonarclouda po refaktoryzacji:



Jak widać na załączonym zrzucie ograniczyłam dług technologiczny do zera oraz pozbyłam się wszystkich code smelli, a moje testy pokryły prawie 98% kodu.

Złożoność poznawcza moich klas jest również zdecydowanie niższa:

| | |
|--|---|
|  src/main/java/com/gildedrose/item/DefaultItem.java | 5 |
|  src/main/java/com/gildedrose/item/BackstagePass.java | 4 |
|  src/main/java/com/gildedrose/GildedRose.java | 2 |
|  src/main/java/com/gildedrose/item/ItemFactory.java | 1 |
|  src/main/java/com/gildedrose/item/AgedBrie.java | 0 |
|  src/test/java/com/gildedrose/AgedBrieTest.java | 0 |
|  src/test/java/com/gildedrose/BackstagePassTest.java | 0 |
|  src/main/java/com/gildedrose/item/ConjuredItem.java | 0 |
|  src/test/java/com/gildedrose/ConjuredItemTest.java | 0 |
|  src/test/java/com/gildedrose/DefaultItemTest.java | 0 |
|  src/main/java/com/gildedrose/Item.java | 0 |
|  pom.xml | 0 |
|  src/main/java/com/gildedrose/item/Sulfuras.java | 0 |
|  src/test/java/com/gildedrose/SulfurasTest.java | 0 |
|  src/test/java/com/gildedrose/TextTestFixture.java | 0 |

Do wyliczenia złożoności cykloatycznej skorzystałam dodatkowo z Codacy.

Przed:

| GRADE ^ | FILENAME ^ | ISSUES v | DUPPLICATION ^ | COMPLEXITY ^ |
|---------|---|----------|----------------|--------------|
| B | src/main/java/com/gildedrose/GildedRose.java | 2 | 0 | 21 |
| C | src/test/java/com/gildedrose/GildedRoseTest.java | 1 | 0 | 1 |
| A | src/test/java/com/gildedrose/TextTestFixture.java | 0 | 0 | 4 |
| A | src/main/java/com/gildedrose/Item.java | 0 | 0 | 1 |

Po:

| GRADE ^ | FILENAME ^ | ISSUES v | DUPPLICATION ^ | COMPLEXITY ^ |
|---------|--|----------|----------------|--------------|
| A | src/main/java/com/gildedrose/item/ItemFactory.java | 0 | 0 | 5 |
| A | src/main/java/com/gildedrose/item/DefaultItem.java | 0 | 0 | 4 |
| A | src/main/java/com/gildedrose/item/BackstagePass.java | 0 | 0 | 4 |
| A | src/main/java/com/gildedrose/item/Sulfuras.java | 0 | 0 | 1 |
| A | src/main/java/com/gildedrose/GildedRose.java | 0 | 0 | 2 |
| A | src/main/java/com/gildedrose/item/AgedBrie.java | 0 | 0 | 1 |
| A | src/main/java/com/gildedrose/item/ConjuredItem.java | 0 | 0 | 1 |
| A | src/main/java/com/gildedrose/Item.java | 0 | 0 | 1 |