

프로그래밍 역량 강화 전문기관, 민코딩

Gilded Rose



목차

1. Gilded Rose KATA 소개
2. GildedRose Refactoring 준비
3. [1] Unit Test 작성하기
4. [2] Method Level Refactoring
5. [3] Class Level Refactoring
6. 참고자료

Refactoring Kata

Gilded Rose KATA 소개

Gilded Rose 시스템 스토리 - 3분간 이해해보기

Gilded Rose 는 Allison 이 운영하는 작은 여관입니다. 이 여관은 유명한 도시의 요지에 자리잡고 있습니다. 우리는 이 여관에서 finest goods를 사고 팝니다. 그런데 상품의 판매가능 기간(sell by date)이 다가올수록 상품들의 품질은 지속적으로 떨어집니다.

- ✓ 모든 아이템에는 판매가능기간 (sellIn)이 있습니다.
 - 판매가능 기간은 아이템을 팔아야 하는 날까지 남아 있는 일 수로 표시합니다.
- ✓ 모든 아이템에는 품질값(quality)이 있습니다.
 - 품질값은 아이템이 얼마나 가치가 있는지를 나타냅니다.
- ✓ 시스템은 매일 자정에 모든 아이템의 값들을 갱신합니다.
- ✓ 판매가능 기간이 지나면, 품질은 두 배씩 빨리 떨어집니다.
- ✓ 아이템의 품질은 음수가 될 수 없습니다.
- ✓ 아이템의 품질은 50보다 클 수 없습니다.
- ✓ "Aged Brie"는 시간이 지날수록 품질이 증가합니다.
- ✓ "Sulfuras"는 전설의 아이템입니다.
 - 절대 팔지도 않고 품질이 떨어지지도 않습니다.
- ✓ "Backstage passes" 는 판매가능기간이 다가올수록 품질이 증가합니다.
 - 판매가능기간이 10일 이하일 때 품질은 2씩 증가하고, 판매가능기간이 5일 이하일때는 3씩 증가합니다.
 - 콘서트가 끝나고 판매가능 기간이 지나면 품질은 0이 됩니다.



GuildRose

✓ 사외 공식 KATA 링크

- <https://github.com/emilybache/GildedRose-Refactoring-Kata/tree/main/cpp>



게임 WOW의 상점(여관)인 Gilded Rose



전설의 아이템 "Sulfuras, Hand of Ragnaros"

요구사항 명세

- ✓우리 상점의 상품들은 판매 기한이 가까워질수록 가격이 변동되곤합니다.
- ✓우리 상품들을 관리하는 Lagacy 시스템이 있습니다.
- ✓시스템에 새로운 기능을 추가하여,
새로운 카테고리의 상품을 판매할 수 있도록 해주세요.

SellIn 과 Quality

✓ 모든 아이템은 SellIn 값과 Quality 값을 가집니다.

✓ SellIn :

- 판매해야하는 남은 기간 입니다. (판매기한)
- 하루에 1씩 줄어듭니다.

✓ Quality

- Quality == 판매금액입니다.
- 시간이 지날수록 가치가 줄어드는 아이템도 있고,
시간이 지날수록 가치가 올라가는 아이템도 있고,
시간이 지나도 가치가 유지되는 아이템이 있습니다.

판매 상품 정보

상품을 총 네 가지로 분류하여 판매중인 시스템이 있는데,
한 가지 상품을 더 추가해야한다.

일반
아이템

<else>

전설의
아이템

Sulfras...

오래된
치즈

Aged Brie

콘서트
입장권

Backstage
passes ...

마법
아이템

Conjured

상품 추가 개발해야함
(추가 기능 구현 내용이지만,
우리는 신경쓰지 않는다.)

판매상품 가격 변동

각 아이템들은 하루가 지날 때 마다
값이 바뀌며, 바뀌는 규칙이 각각 다르다.

일반
아이템

<else>

값 : 0 ~ 50
매일 1씩 떨어짐
최소 0원

전설의
아이템

Sulfras...

값 : 80
매일 가격 변동 없음

오래된
치즈

Aged Brie

값 : 0 ~ 50
매일 1씩 가격이 오름
최대 50원

콘서트
입장권

Backstage
passes ...

값 : 0 ~ 50
콘서트가 가까워지면
가격이 비싸짐
최대 50원

마법
아이템

Conjured

값 : 0 ~ 50
일반 아이템에 2배씩 가격 떨어짐
(추가 기능 구현 내용이지만,
우리는 신경쓰지 않는다.)

아이템별 가치 변화

✓ 일반아이템 (최소 가격 : 0, 음수 불가)

- 매일 하루에 가격 1씩 감소
- 남은 판매기한이 음수가 될 때 부터 가격 2씩 감소

✓ 전설아이템

- 가격 감소 없음
- 하루가 지나도, 남은 판매기한 변경되지 않음

✓ 오래된치즈 (최대 50)

- 하루에 가격 1씩 계속 증가
- 남은 판매기한이 음수가 될 때 부터는, 가격이 2씩 증가

✓ 콘서트 입장권 (최대 50, 최소 0)

- 10일 전 이전에는 하루에 가격 1씩 증가
- 10 ~ 6일에 하루가 지나면 가격이 2씩 증가
- 5 ~ 1일에 하루가 지나면 가격이 3씩 증가
- 남은 판매기한이 음수가 될 때는, 가격은 무조건 0 (콘서트 종료 되었기 때문)

소스코드 구조

✓클래스

- GildedRose : 상점 아이템 관리 시스템
- Item : Item의 정보를 갖는 class

✓하루가 지날때마다 값을 갱신하는 메서드

- Gilded Rose class의 updateQuality() 메서드

Gilded Rose Refactoring 준비

일단 코드부터 살펴보자.

✓다음 링크에서, 수동으로 소스코드 세팅하기

- 사내 링크

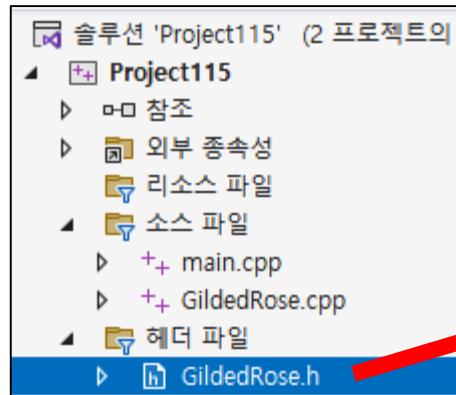
- <https://github.samsungds.net/cra1-sec/GildedRose>

- 사외 링크

- <https://github.com/mincoding1/GildedRose>

파일 수동으로 추가하기 1

✓GildedRose/cpp/GildedRose.h



```
#pragma once
#include <string>
#include <vector>

using namespace std;

class Item
{
public:
    string name;
    int sellIn;
    int quality;
    Item(string name, int sellIn, int quality) : name(name), sellIn(sellIn), quality(quality)
    {}
};

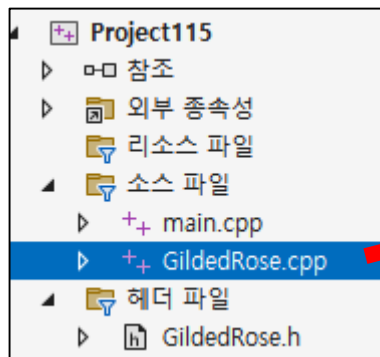
class GildedRose
{
public:
    vector<Item>& items;
    GildedRose(vector<Item>& items);

    void updateQuality();
};
```

#pragma once :
중복 include 방지 (헤더가드)를 넣어주자.

파일 수동으로 추가하기 2

✓GildedRose/cpp/GildedRose.cpp



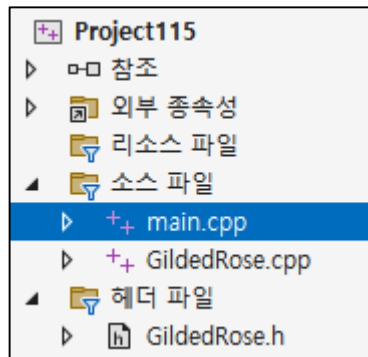
```
#include "GildedRose.h"

GildedRose::GildedRose(vector<Item>& items) : items(items)
{}

void GildedRose::updateQuality()
{
    for (int i = 0; i < items.size(); i++)
    {
        if (items[i].name != "Aged Brie" && items[i].name != "Backstage Passes")
        {
            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;
            }
        }
    }
}
```

파일 수동으로 추가하기 3

✓GildedRose/cpp/main.cpp



```
#include <iostream>
#include "GildedRose.h"

void print_item(Item& item)
{
    std::cout << item.name << ", " << item.sellIn << ", " << item.quality << std::endl;
}

int main()
{
    vector<Item> items;

    items.push_back({ "+5 Dexterity Vest", 10, 20 });
    items.push_back({ "Aged Brie", 2, 0 });
    items.push_back({ "Elixir of the Mongoose", 5, 7 });
    items.push_back({ "Sulfuras, Hand of Ragnaros", 0, 80 });
    items.push_back({ "Sulfuras, Hand of Ragnaros", -1, 80 });
    items.push_back({ "Backstage passes to a TAFKAL80ETC concert", 15, 20 });
    items.push_back({ "Backstage passes to a TAFKAL80ETC concert", 10, 49 });
    items.push_back({ "Backstage passes to a TAFKAL80ETC concert", 5, 49 });

    // this Conjured item doesn't yet work properly
    items.push_back({ "Conjured Mana Cake", 3, 6 });

    std::cout << "OMGHAI!" << std::endl;

    GildedRose app(items);

    for (int day = 0; day <= 30; day++)
    {
        std::cout << "----- day " << day << " -----" << std::endl;
        std::cout << "name, sellIn, quality" << std::endl;
    }
}
```


빌드 테스트

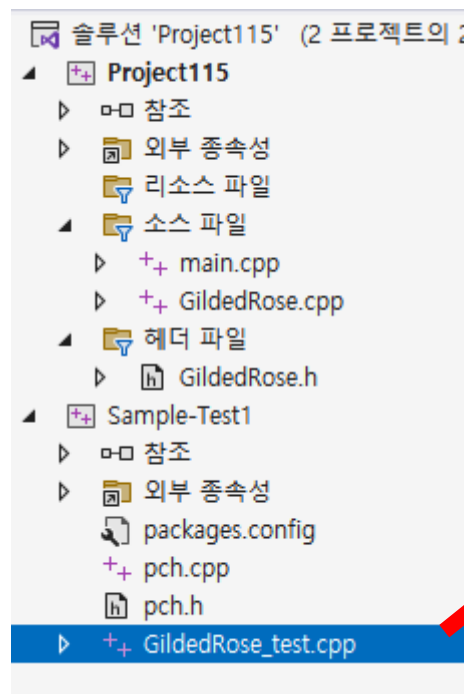
✓day 30까지 로그가 출력되어야한다.

```
Microsoft Visual Studio 디버깅 콘솔
Conjured Mana Cake, -19, 0
----- day 23 -----
name, sellIn, quality
+5 Dexterity Vest, -13, 0
Aged Brie, -21, 44
Elixir of the Mongoose, -18, 0
Sulfuras, Hand of Ragnaros, 0, 80
Sulfuras, Hand of Ragnaros, -1, 80
Backstage passes to a TAFKAL80ETC concert, -8, 0
Backstage passes to a TAFKAL80ETC concert, -13, 0
Backstage passes to a TAFKAL80ETC concert, -18, 0
Conjured Mana Cake, -20, 0
----- day 24 -----
name, sellIn, quality
+5 Dexterity Vest, -14, 0
Aged Brie, -22, 46
Elixir of the Mongoose, -19, 0
Sulfuras, Hand of Ragnaros, 0, 80
Sulfuras, Hand of Ragnaros, -1, 80
Backstage passes to a TAFKAL80ETC concert, -9, 0
Backstage passes to a TAFKAL80ETC concert, -14, 0
Backstage passes to a TAFKAL80ETC concert, -19, 0
Conjured Mana Cake, -21, 0
----- day 25 -----
name, sellIn, quality
+5 Dexterity Vest, -15, 0
Aged Brie, -23, 48
Elixir of the Mongoose, -20, 0
Sulfuras, Hand of Ragnaros, 0, 80
Sulfuras, Hand of Ragnaros, -1, 80
Backstage passes to a TAFKAL80ETC concert, -10, 0
Backstage passes to a TAFKAL80ETC concert, -15, 0
Backstage passes to a TAFKAL80ETC concert, -20, 0
```

유닛 테스트 수동으로 추가하기

✓ GildedRose/cpp/GildedRose_test.cpp

GildedRose.h 가 아닌
GildedRose.cpp 임에 유의하자.

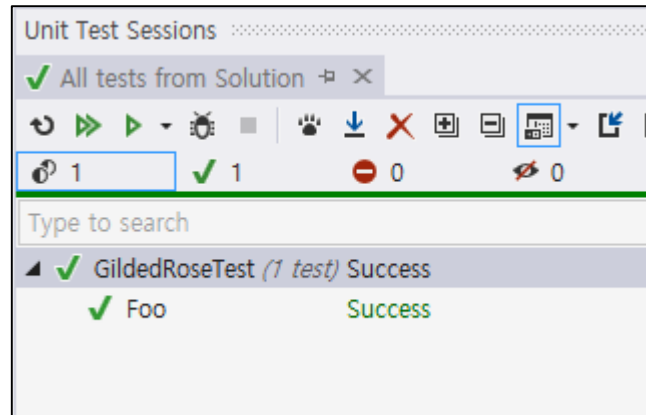


```
#include "pch.h"  
#include "../Project115/GildedRose.cpp"  
  
TEST(GildedRoseTest, Foo) {  
    vector<Item> items;  
    items.push_back(Item("Foo", 0, 0));  
    GildedRose app(items);  
    app.updateQuality();  
    EXPECT_EQ("Foo", app.items[0].name);  
}
```

pch.h 내부에는 #include <gtest/gtest.h> 가 선언됨
일부 코드를 본인에게 맞게 수정한다.

유닛테스트 수행해보기

✓정상동작해야한다.



1. 소스코드 분석

소스코드를 가볍게 분석해본다.

- 완전 분석이 아닌, 가벼운 분석
- main 코드와 unittest 코드를 분석한다.
- GildedRose.cpp 파일을 분석한다.

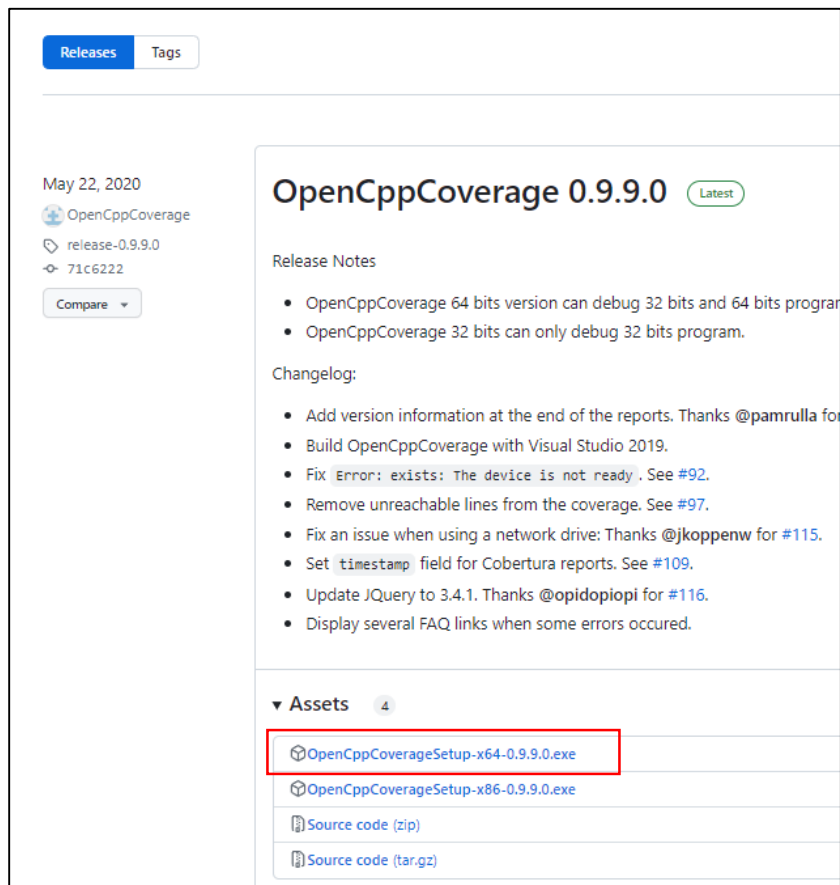
2. 리팩토링 준비

테스트 준비

- 리팩토링 전, 테스트 환경을 준비한다.
- 코드 커버리지가 100% 권장 (우리 실습에서 진행할 예정)

[참고] 코드 커버리지 측정을 위한 준비

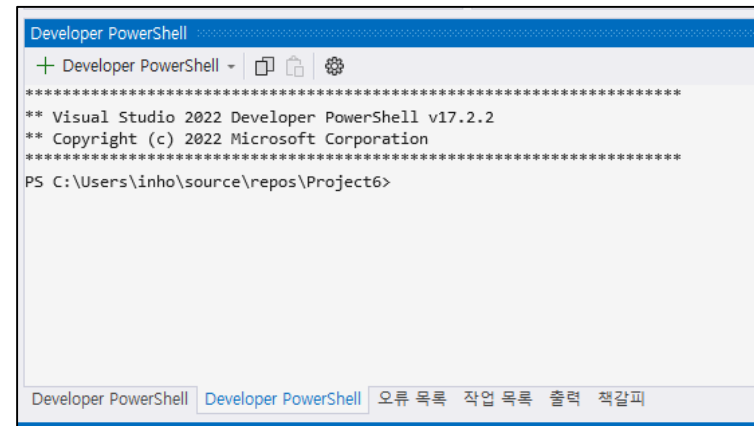
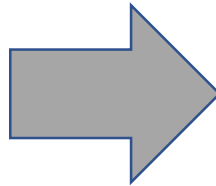
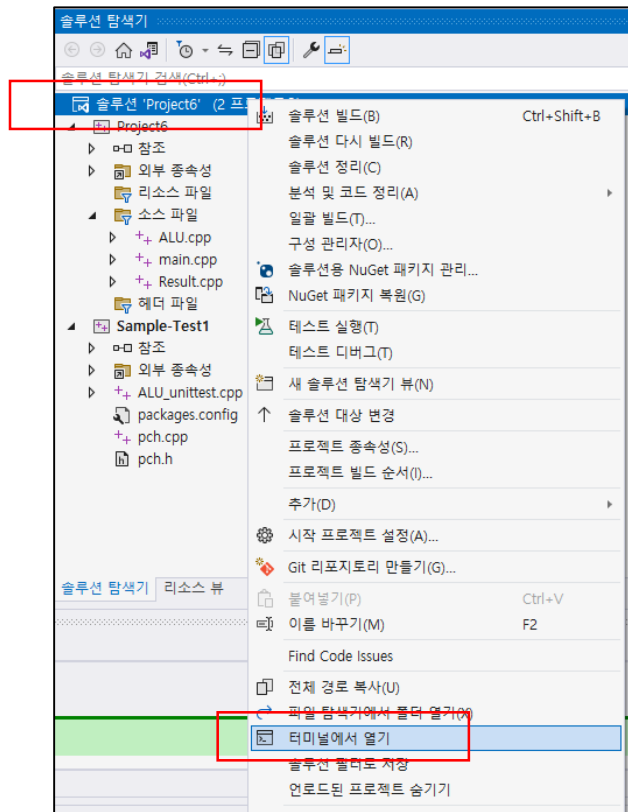
<https://github.com/OpenCppCoverage/OpenCppCoverage/releases>



64bit exe 파일로 다운로드를 받는다.
설치 후, **Visual Studio**를 꺾다가 다시 켜다.

동작 테스트

프로젝트가 아닌 “**솔루션**” 에서 **터미널 열기**



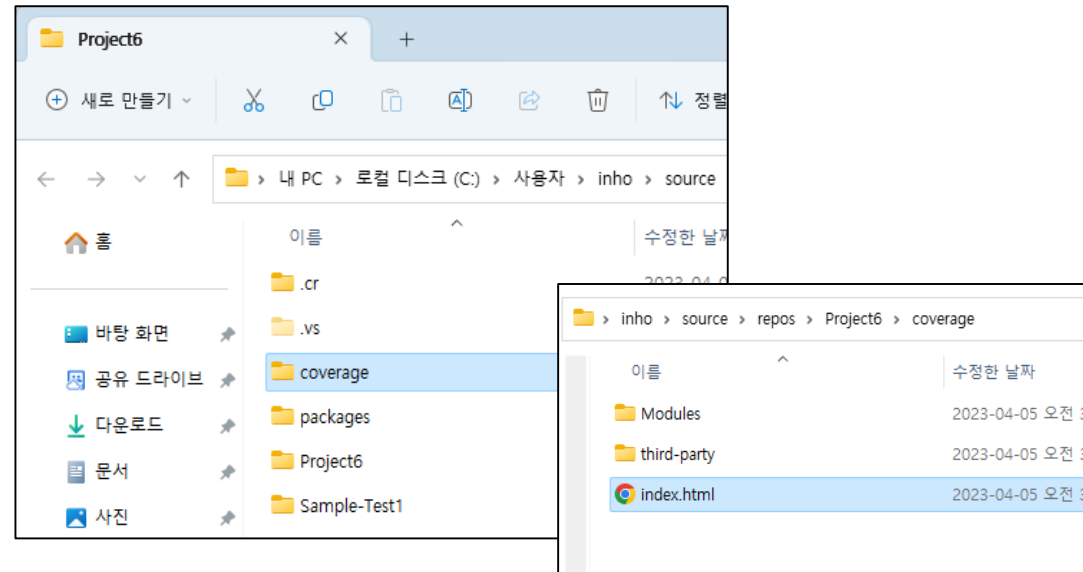
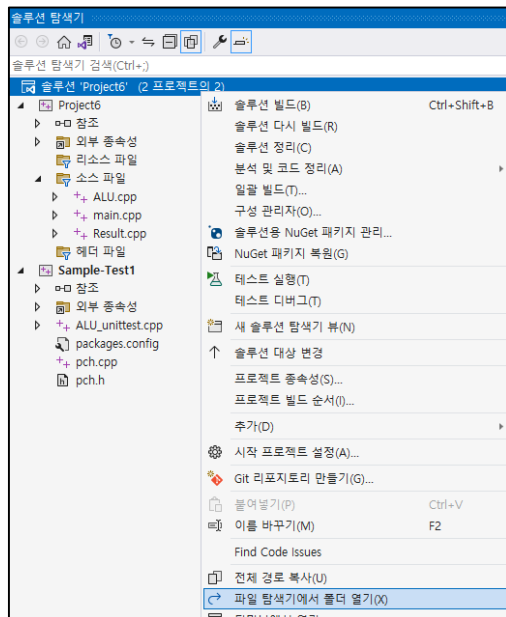
코드 커버리지 확인

OpenCppCoverage.exe --sources C:*.c --export_type=html:coverage -- .\x64\Debug\Sample-Test1.exe

1. 코드 커버리지 측정 시작
2. coverage폴더에서 결과 확인

```
개발자 PowerShell
+ 개발자 PowerShell
[-----] 1 test from GildedRoseTest (3 ms total)

[-----] Global test environment tear-down
[=====] 1 test from 1 test case ran. (5 ms total)
[ PASSED ] 1 test.
[info] Module: C:\Windows\System32\kernel.appcore.dll is selected because it matches selected pattern: *
[info] -----
[info] Coverage generated in Folder C:\Users\minco\source\repos\Project109\coverage
[info] -----
[info] The code coverage report is not what you expect? See the FAQ https://github.com/OpenCppCoverage/OpenCppCoverage/wiki/FAQ.
PS C:\Users\minco\source\repos\Project109> OpenCppCoverage.exe --sources C:*.c --export_type=html:coverage -- .\x64\Debug\Sample-Test1.exe
```



[Trouble Shooting] OpenCppCoverage.exe

```
OpenCppCoverage.exe --sources C:*.c --export_type=html:coverage -- .Wx64WDebugWSample-Test1.exe
```

✓오타율이 매우 높습니다! 다음 항목을 체크해주세요.

1. -- 는 총 3개가 기입이 되었는가?
2. 소스코드가 C:W에 있다면 : --source C:*.c
소스코드가 D:W에 있다면 : --source D:*.c
3. --export_type=html:coverage 가 정확히 입력 되었는가?
4. 띄어쓰기를 정확히 지켰는가? (특히 마지막 -- 부분)
5. 경로에 [GoogleTest 프로젝트명].exe이 정확히 입력 되었는가?
6. 경로지정시 '/' 가 아닌, 'W' or '\'를 사용해야한다.

안된다면
"x64W"를 삭제하여
시도해본다.

Code Coverage 결과

- ✓ 초록색 : 테스트가 닿은 곳
- ✓ 빨간색 : 테스트가 닿지 못한 곳

Sample-Test1.exe

Coverage	Total lines	Items
<div><div>Uncover 40%</div><div>Cover 60%</div></div>	40	C:\Users\minco\source\repos\Project109\x64\Debug\Sample-Test1.exe
<div><div>Uncover 49%</div><div>Cover 51%</div></div>	33	C:\Users\minco\source\repos\Project109\Project109\GildedRose.cc
<div><div>Uncover 0%</div><div>Cover 100%</div></div>	7	C:\Users\minco\source\repos\Project109\Sample-Test1\GildedRose_unittest.cc



```
1. #include "GildedRose.h"
2.
3. GildedRose::GildedRose(vector<Item>& items) : items(items)
4. {}
5.
6. void GildedRose::updateQuality()
7. {
8.     for (int i = 0; i < items.size(); i++)
9.     {
10.         if (items[i].name != "Aged Brie" && items[i].name != "Backstage passes to a TAFKALBOETO concert")
11.         {
12.             if (items[i].quality > 0)
13.             {
14.                 if (items[i].name != "Sulfuras, Hand of Ragnaros")
15.                 {
16.                     items[i].quality = items[i].quality - 1;
17.                 }
18.             }
19.         }
20.         else
21.         {
22.             if (items[i].quality < 50)
23.             {
24.                 items[i].quality = items[i].quality + 1;
25.             }
26.             if (items[i].name == "Backstage passes to a TAFKALBOETO concert")
27.             {
28.                 if (items[i].sellIn < 11)
29.                 {
30.                     if (items[i].quality < 50)
31.                     {
32.                         items[i].quality = items[i].quality + 1;
33.                     }
34.                 }
35.             }
36.             if (items[i].sellIn < 6)
37.             {
38.                 if (items[i].quality < 50)
39.                 {
40.                     items[i].quality = items[i].quality + 1;
41.                 }
42.             }
43.         }
44.     }
45. }
46.
47. if (items[i].name != "Sulfuras, Hand of Ragnaros")
48. {
49.     items[i].sellIn = items[i].sellIn - 1;
50. }
51.
52. if (items[i].sellIn < 0)
53. {
54.     if (items[i].name != "Aged Brie")
55.     {
56.         if (items[i].name != "Backstage passes to a TAFKALBOETO concert")
57.         {
58.             if (items[i].quality > 0)
59.             {
60.                 if (items[i].name != "Sulfuras, Hand of Ragnaros")
61.                 {
62.                     items[i].quality = items[i].quality - 1;
63.                 }
64.             }
65.         }
66.         else
67.         {
68.             items[i].quality = items[i].quality - items[i].quality;
69.         }
70.     }
71.     else
72.     {
73.         if (items[i].quality < 50)
74.         {
75.             items[i].quality = items[i].quality + 1;
76.         }
77.     }
78. }
79.
80. }
```

교안 구성

✓교안은 3개의 챕터로 되어있음

- Unit Test 작성
- Method Level 리팩토링
- Class Level 리팩토링

1. Unit Test 작성하기

[실습] Test case 만들기

✓Step 1. item이 하나도 없는 상태에서 updateQuality 하는 경우

- 기존 작성된 테스트케이스를 지우고, 하기 코드를 작성한다.
- 빨간색으로 뜨는 부분은 Alt + Enter로 문제를 해결한다.

```
TEST(GildedRoseTest, should_be_nothing_when_no_item) {  
    // given (arrange)  
    vector<Item> items;  
    GildedRose app(items);  
  
    // when (act)  
    app.updateQuality();  
  
    // then (assert)  
    EXPECT_EQ(0, items.size());  
}
```

[실습] Test case 만들기

✓ Step 2. name="noname", sellIn=0, quality=0

- 일반아이템, sellIn(기한)은 1 줄어들고
Quality의 최소값은 0이므로, 더 이상 줄어들지 않는다.

```
TEST(GildedRoseTest, noname_sellin_0_quality_0) {  
    // given (arrange)  
    vector<Item> items = { Item("noname", 0, 0) };  
    GildedRose app(items);  
  
    // when (act)  
    app.updateQuality();  
  
    // then (assert)  
    EXPECT_EQ(-1, items[0].sellIn);  
    EXPECT_EQ(0, items[0].quality);  
}
```

[실습] Test case 만들기

✓ Step 3. name="noname", sellin=0, quality=1

- 규칙상, 일반아이템의 남은 기한(sellin)이 음수가 된다면 quality가 2씩 떨어져야 한다.
- 하지만 quality의 최소값은 0 이므로, 하루가 지난 quality 값은 0이 된다.

```
TEST(GildedRoseTest, noname_sellin_0_quality_1) {  
    // given (arrange)  
    vector<Item> items = { Item("noname", 0, 1) };  
    GildedRose app(items);  
  
    // when (act)  
    app.updateQuality();  
  
    // then (assert)  
    EXPECT_EQ(-1, items[0].sellIn);  
    EXPECT_EQ(0, items[0].quality);  
}
```

[실습] Test case 만들기

✓ Step 4. name="Sulfuras...", sellin=0, quality=80

- 전설의 아이템, 규칙상 sellin과 quality는 변하지 않는다.

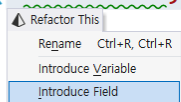
```
TEST(GildedRoseTest, sulfuras_sellin_0_quality_80) {  
    // given (arrange)  
    vector<Item> items = { Item("Sulfuras, Hand of Ragnaros", 0, 80) };  
    GildedRose app(items);  
  
    // when (act)  
    app.updateQuality();  
  
    // then (assert)  
    EXPECT_EQ(0, items[0].sellIn);  
    EXPECT_EQ(80, items[0].quality);  
}
```


[실습] Test case 만들기

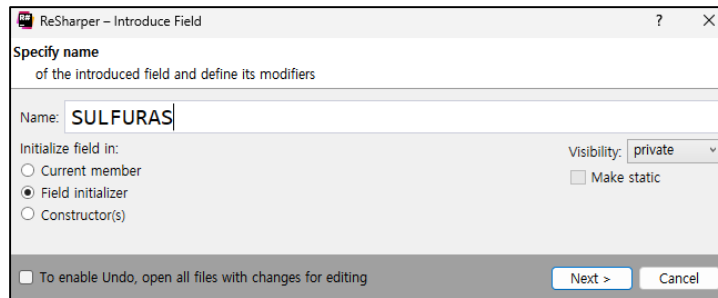
✓ Step 5. 상수 리팩토링 - 1

- IDE의 Refactor (Introduce Field) 기능

```
furas_sellin_0_quality_80) {  
    { Item("Sulfuras, Hand of Ragnaros", 0, 80) };  
};
```

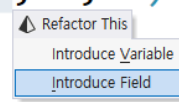


A context menu is shown over the string "Sulfuras, Hand of Ragnaros" in the code. The menu includes options: Refactor This, Rename (Ctrl+R, Ctrl+R), Introduce Variable, and Introduce Field. The 'Introduce Field' option is highlighted.

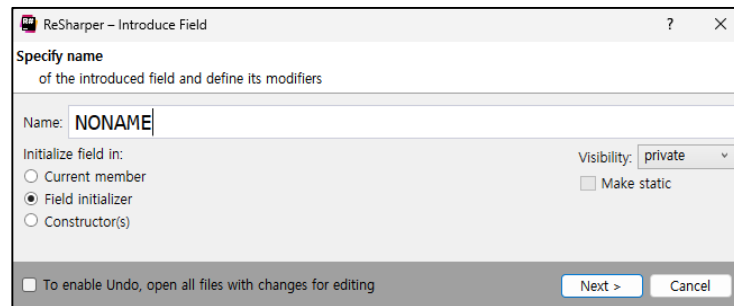


```
1 #include "pch.h"  
2  
3 #include "../Project109/GildedRose.h"  
4 #include "../Project109/GildedRose.cc"  
5  
6 const char* SULFURAS = "Sulfuras, Hand of Ragnaros";  
7 const char* NONAME = "noname";  
8
```

```
em("noname", 0, 1) };
```



A context menu is shown over the string "noname" in the code. The menu includes options: Refactor This, Introduce Variable, and Introduce Field. The 'Introduce Field' option is highlighted.



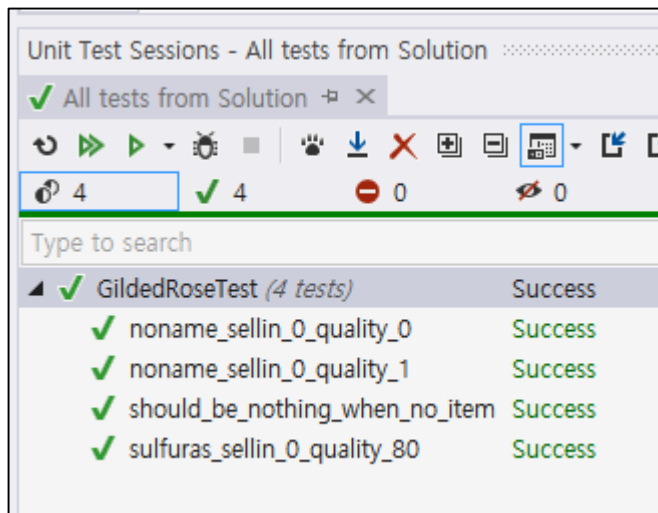
수동으로 최 상단에 이동시켜준다.

Field Initializer는 의미가 없다.

[실습] Test case 만들기

✓Step 5. 상수 리팩토링 - 2

- 상수로 치환한다.
- 테스트 결과가 Pass 이어야한다.



```
6 const char* SULFURAS = "Sulfuras, Hand of Ragnaros";
7 const char* NONAME = "noname";
8
9 TEST(GildedRoseTest, should_be_nothing_when_no_item) {
10     // given (arrange)
11     vector<Item> items;
12     GildedRose app(items);
13
14     // when (act)
15     app.updateQuality();
16
17     // then (assert)
18     EXPECT_EQ(0, items.size());
19 }
20
21 TEST(GildedRoseTest, noname_sellin_0_quality_0) {
22     // given (arrange)
23     vector<Item> items = { Item(NONAME, 0, 0) };
24     GildedRose app(items);
25
26     // when (act)
27     app.updateQuality();
28
29     // then (assert)
30     EXPECT_EQ(-1, items[0].sellIn);
31     EXPECT_EQ(0, items[0].quality);
32 }
33
34 TEST(GildedRoseTest, noname_sellin_0_quality_1) {
35     // given (arrange)
36     vector<Item> items = { Item(NONAME, 0, 1) };
37     GildedRose app(items);
38
39     // when (act)
40     app.updateQuality();
41
42     // then (assert)
43     EXPECT_EQ(-1, items[0].sellIn);
44     EXPECT_EQ(0, items[0].quality);
45 }
46
47 TEST(GildedRoseTest, sulfuras_sellin_0_quality_80)
48 {
49     // given (arrange)
50     vector<Item> items = { Item(SULFURAS, 0, 80) };
51     GildedRose app(items);
```

[실습] Test case 만들기

✓ Step 6. name="Aged Brie", sellin=0, quality=0

- 오래된 치즈는, 남은 기한이 음수가 될 때부터 quality가 2씩 증가

문자열 상수 추가

```
5  
6 const char* SULFURAS = "Sulfuras, Hand of Ragnaros";  
7 const char* AGED_BRIE = "Aged Brie";  
8 const char* NONAME = "noname";  
9
```

Test 메서드 추가

```
TEST(GildedRoseTest, agedBrie_sellin_0_quality_0)  
{  
    // given (arrange)  
    vector<Item> items = { Item(AGED_BRIE, 0, 0) };  
    GildedRose app(items);  
  
    // when (act)  
    app.updateQuality();  
  
    // then (assert)  
    EXPECT_EQ(-1, items[0].sellIn);  
    EXPECT_EQ(2, items[0].quality);  
}
```

[실습] Test case 만들기

✓ Step 7. name="Backstage...", sellin=0, quality=0

- 콘서트 티켓, 남은 기한이 음수가 될때는 무조건 quality는 0가 된다.

문자열 상수 추가

```
const char* SULFURAS = "Sulfuras, Hand of Ragnaros";  
const char* AGED_BRIE = "Aged Brie";  
const char* BACKSTAGE_PASS = "Backstage passes to a TAFKAL80ETC concert";  
const char* NONAME = "noname";
```

Test 메서드 추가

```
TEST(GildedRoseTest, backstage_pass_sellin_0_quality_0)  
{  
    // given (arrange)  
    vector<Item> items = { Item(BACKSTAGE_PASS, 0, 0) };  
    GildedRose app(items);  
  
    // when (act)  
    app.updateQuality();  
  
    // then (assert)  
    EXPECT_EQ(-1, items[0].sellIn);  
    EXPECT_EQ(0, items[0].quality);  
}
```

[실습] Test case 만들기

✓ Step 8. name="Backstage...", sellin=0, quality=49

- 콘서트 티켓, 남은 기한이 음수가 될때는 무조건 quality는 0가 된다.

```
TEST(GildedRoseTest, backstage_pass_sellin_0_quality_49)
{
    // given (arrange)
    vector<Item> items = { Item(BACKSTAGE_PASS, 0, 49) };
    GildedRose app(items);

    // when (act)
    app.updateQuality();

    // then (assert)
    EXPECT_EQ(-1, items[0].sellIn);
    EXPECT_EQ(0, items[0].quality);
}
```

[실습] Test case 만들기

✓ Step 9. name="Backstage...", sellin=12, quality=0

- 콘서트 티켓, 남은 기한이 10일보다 더 남았을 때는, 하루에 quality가 1씩 증가한다.

```
TEST(GildedRoseTest, backstage_pass_sellin_12_quality_0)
{
    // given (arrange)
    vector<Item> items = { Item(BACKSTAGE_PASS, 12, 0) };
    GildedRose app(items);

    // when (act)
    app.updateQuality();

    // then (assert)
    EXPECT_EQ(11, items[0].sellIn);
    EXPECT_EQ(1, items[0].quality);
}
```

[실습] Test case 만들기

✓ Step 10. name="Sulfuras...", sellin=-2, quality=80

- 전설의 아이템은 sellin과 quality는 하루가 지나도 변함이 없다.

m2의 의미 : -2

```
TEST(GildedRoseTest, sulfuras_sellin_m2_quality_80)
{
    // given (arrange)
    vector<Item> items = { Item(SULFURAS, -2, 80) };
    GildedRose app(items);

    // when (act)
    app.updateQuality();

    // then (assert)
    EXPECT_EQ(-2, items[0].sellIn);
    EXPECT_EQ(80, items[0].quality);
}
```

[실습] Test case 만들기

✓ Step 11. name="Aged Brie", sellin=0, quality=50

- 치즈는 남은 기한이 음수가 되는 경우 2씩 증가되어야 한다.
- 하지만 치즈의 최댓값은 50이므로, 하루가 지나도 quality는 50이다.

```
TEST(GildedRoseTest, agedBrie_sellin_0_quality_50)
{
    // given (arrange)
    vector<Item> items = { Item(AGED_BRIE, 0, 50) };
    GildedRose app(items);

    // when (act)
    app.updateQuality();

    // then (assert)
    EXPECT_EQ(-1, items[0].sellIn);
    EXPECT_EQ(50, items[0].quality);
}
```


[실습] Test case 만들기

✓Step 12. char* 을 string class로 변경

- Alt + Shift 마우스 드레그로, 사각형 영역을 잡는다.
- "string" 입력시 한꺼번에 변경된다.

```
1 #include "pch.h"
2
3 #include "../Project109/GildedRose.h"
4 #include "../Project109/GildedRose.cc"
5
6 const char* SULFURAS = "Sulfuras, Hand of Ragnaros";
7 const char* AGED_BRIE = "Aged Brie";
8 const char* BACKSTAGE_PASS = "Backstage passes to a TAFKAL80ETC concert";
9 const char* NONAME = "noname";
10
```



```
const string SULFURAS = "Sulfuras, Hand of Ragnaros";
const string AGED_BRIE = "Aged Brie";
const string BACKSTAGE_PASS = "Backstage passes to a TAFKAL80ETC concert";
const string NONAME = "noname";
```

[도전] Test case 만들기

✓Step 13. 코드 커버리지가 **100% 되도록** 테스트 코드 추가하기

- 직접 코드 커버리지 측정한다.
어떤 소스코드가 부족한지 파악 후
코드 커버리지 100%가 되도록 Test Case를 하나 더 추가한다.

2. Method Level Refactoring

[실습] Refactoring

✓Step 1-1. Constant Refactoring

- 하드코딩된 모든 문자열을 상수로 변경해야한다.
- GildedRose.h 에 다음 코드를 추가한다. (UnitTest에 구현한 코드를 복사하여 편집)

```
#pragma once
#include <string>
#include <vector>

using namespace std;

namespace ITEM {
    const string SULFURAS = "Sulfuras, Hand of Ragnaros";
    const string AGED_BRIE = "Aged Brie";
    const string BACKSTAGE_PASS = "Backstage passes to a TAFKAL80ETC concert";
    const string NONAME = "noname";
}

class Item
```

GildedRose.h

[실습] Refactoring

✓ Step 1-2. Constant Refactoring

- 하드코딩된 모든 문자열을, 문자열 상수로 모두 변경하기 (수작업)

```
1 #include "GildedRose.h"
2
3 GildedRose::GildedRose(vector<Item>& items) : items(items)
4 {}
5
6 void GildedRose::updateQuality()
7 {
8     for (int i = 0; i < items.size(); i++)
9     {
10         if (items[i].name != "Aged Brie" && items[i].name != "Backstage passes to a TAFKAL80ETC concert")
11         {
12             if (items[i].quality > 0)
13             {
14                 if (items[i].name != ITEM::SULFURAS)
15                 {
16                     items[i].quality = items[i].quality - 1;
17                 }
18             }
19         }
20         else
21         {
22             if (items[i].quality < 50)
23             {
24                 items[i].quality = items[i].quality + 1;
25
26                 if (items[i].name == "Backstage passes to a TAFKAL80ETC concert")
27                 {
```

GildedRose.cpp

```
void GildedRose::updateQuality()
{
    for (int i = 0; i < items.size(); i++)
    {
        if (items[i].name != ITEM::AGED_BRIE && items[i].name != ITEM::BACKSTAGE_PASS)
        {
            if (items[i].quality > 0)
            {
                if (items[i].name != ITEM::SULFURAS)
                {
```

위와같이 총 8 곳을 변경해야한다.

[실습] Refactoring

✓ Step 2. 자주 사용되는 “items[i]” 대신 item” 이름 부여하기

- item[i] 블록잡고, Refactor (Introduce Variable) 처리
- 모든 “item[i]”을 item 변수로 변경한다. (총 34개)

```
#include "GildedRose.h"

GildedRose::GildedRose(vector<Item>& items) : items(items)
{}

void GildedRose::updateQuality()
{
    for (int i = 0; i < items.size(); i++)
    {
        if (items[i].name != ITEM::AGED_BRIE && items[i].name != ITEM::SULFURAS)
        {
            if (item
                {
                    if (
                        : SULFURAS)
                    {
                        items[i].quality = items[i].quality - 1;
                    }
                }
            }
        }
        else
        {
            if (items[i].quality < 50)
            {
                items[i].quality = items[i].quality + 1;
            }
        }
    }
}
```



```
3 | GildedRose::GildedRose(vector<Item>
4 | {}
5 |
6 | void GildedRose::updateQuality()
7 | {
8 |     for (int i = 0; i < items.size()
9 |     {
10 |         auto &item = items[i];
11 |         if (item.name != ITEM::AGE
12 |         {
13 |             if (item.quality > 0)
```

[참고] Conditional Complexity Refactoring

✓if문 변경하기

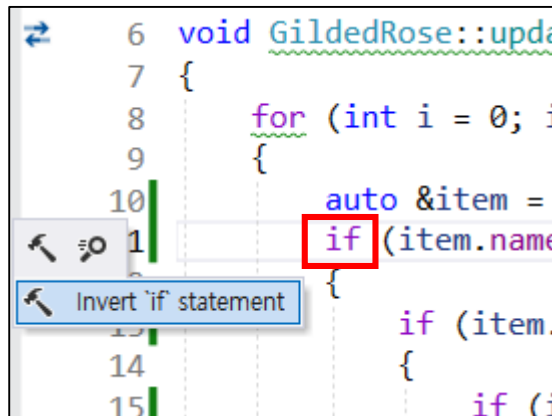
- 실습 Step 3에서는 Invert-if 진행 예정
- 실습 Step 4에서는 Split Condition 진행 예정

Refactoring	Before	After
Invert-if (Step 3 진행예정)	<pre>if (!A) { B } else { C }</pre>	<pre>if (A) { C } else { B }</pre>
Split Condition (Step 4 진행예정)	<pre>if (A B) { C } else { D }</pre>	<pre>if (A) { C } else if (B) { C } else { D }</pre>

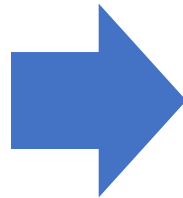
[실습] Refactoring

✓Step 3-1. invert 'if' condition 수행

- Unit Test를 수행하여, 리팩토링에 문제가 없음을 확인한다.



if 클릭 후, Alt + Enter
invert 'if' condition를 수행한다.



```
void GildedRose::updateQuality()
{
    for (int i = 0; i < items.size(); i++)
    {
        auto &item = items[i];
        if (item.name == ITEM::AGED_BRIE || item.name == ITEM::BACKSTAGE_PASS)
        {
            if (item.quality < 50)
            {
                item.quality = item.quality + 1;
            }

            if (item.name == ITEM::BACKSTAGE_PASS)
            {
                if (item.sellIn < 11)
                {
                    if (item.quality < 50)
                    {
                        item.quality = item.quality + 1;
                    }
                }
            }

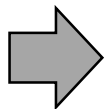
            if (item.sellIn < 6)
            {
                if (item.quality < 50)
                {
                    item.quality = item.quality + 1;
                }
            }
        }
    }
}
```


[실습] Refactoring

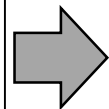
✓Step 3-2. invert 'if' condition 2회 더 수행

- Unit Test를 수행하여, 리팩토링에 문제가 없음을 확인한다.

```
if (item.sellIn < 0)
{
  if (item.name != ITEM::AGED_BRIE)
  {
    if (item.name != ITEM::BACKSTAGE_PASS)
    {
      if (item.quality > 0)
      {
        if (item.name != ITEM::SULFURAS)
        {
          item.quality = item.quality - 1;
        }
      }
    }
    else
    {
      item.quality = item.quality - item.quality;
    }
  }
  else
  {
    if (item.quality < 50)
    {
      item.quality = item.quality + 1;
    }
  }
}
```



```
if (item.sellIn < 0)
{
  if (item.name == ITEM::AGED_BRIE)
  {
    if (item.quality < 50)
    {
      item.quality = item.quality + 1;
    }
  }
  else
  {
    if (item.name != ITEM::BACKSTAGE_PASS)
    {
      if (item.quality > 0)
      {
        if (item.name != ITEM::SULFURAS)
        {
          item.quality = item.quality - 1;
        }
      }
    }
    else
    {
      item.quality = item.quality - item.quality;
    }
  }
}
```



```
if (item.sellIn < 0)
{
  if (item.name == ITEM::AGED_BRIE)
  {
    if (item.quality < 50)
    {
      item.quality = item.quality + 1;
    }
  }
  else
  {
    if (item.name == ITEM::BACKSTAGE_PASS)
    {
      item.quality = item.quality - item.quality;
    }
    else
    {
      if (item.quality > 0)
      {
        if (item.name != ITEM::SULFURAS)
        {
          item.quality = item.quality - 1;
        }
      }
    }
  }
}
```

소스코드 맨 밑쪽 코드

[실습] Refactoring

✓Step 4-1. 이제, 하나의 조건문을 2개의 if 문으로 분할할 것이다.

- 아래 코드는 이해를 위한 코드이다. 코드를 이해해보자.

```
for (int i = 0; i < items.size(); i++)
{
    auto &item = items[i];
    if (item.name == ITEM::AGED_BRIE || item.name == ITEM::BACKSTAGE_PASS)
    {
        //HI
        //...
    }
}
```



```
for (int i = 0; i < items.size(); i++)
{
    auto &item = items[i];
    if (item.name == ITEM::AGED_BRIE)
    {
        //HI
        //...
    }
    else if (item.name == ITEM::BACKSTAGE_PASS)
    {
        //HI
        //...
    }
}
```

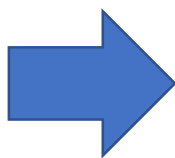
[실습] Refactoring

✓Step 4-2. 수동으로 분할 후 Unit Test를 수행하여 이상없음을 확인한다.

```
void GildedRose::updateQuality()
{
    for (int i = 0; i < items.size(); i++)
    {
        auto &item = items[i];
        if (item.name == ITEM::AGED_BRIE || item.name == ITEM::BACKSTAGE_PASS)
        {
            if (item.quality < 50)
            {
                item.quality = item.quality + 1;

                if (item.name == ITEM::BACKSTAGE_PASS)
                {
                    if (item.sellIn < 11)
                    {
                        if (item.quality < 50)
                        {
                            item.quality = item.quality + 1;
                        }
                    }

                    if (item.sellIn < 6)
                    {
                        if (item.quality < 50)
                        {
                            item.quality = item.quality + 1;
                        }
                    }
                }
            }
        }
    }
}
```



```
6 void GildedRose::updateQuality()
7 {
8     for (int i = 0; i < items.size(); i++)
9     {
10         auto &item = items[i];
11         if (item.name == ITEM::AGED_BRIE)
12         {
13             if (item.quality < 50)
14             {
15                 item.quality = item.quality + 1;
16
17                 if (item.name == ITEM::BACKSTAGE_PASS)
18                 {
19                     if (item.sellIn < 11)
20                     {
21                         if (item.quality < 50)
22                         {
23                             item.quality = item.quality + 1;
24                         }
25                     }
26
27                     if (item.sellIn < 6)
28                     {
29                         if (item.quality < 50)
30                         {
31                             item.quality = item.quality + 1;
32                         }
33                     }
34                 }
35             }
36         }
37         else if (item.name == ITEM::BACKSTAGE_PASS)
38         {
39             //13번 line ~ 35번 line Copy & Paste
40         }
41         else
42         {
43             if (item.quality > 0)
```

코드 추가

13 ~ 35번 라인
내용을 여기에 복사!

[실습] Refactoring

✓Step 5-1. 필요없는 코드 삭제하기

의미 : name이 "AGED_BRIE" 일 때

필요없는 코드이므로 삭제 필요

name이 "AGED_BRIE" 이면서
name이 "BACKSTAGE_PASSES" 일 수 없음.
따라서 진입하지 못하는 **dead 코드**

```
6 void GildedRose::updateQuality()
7 {
8     for (int i = 0; i < items.size(); i++)
9     {
10         auto &item = items[i];
11         if (item.name == ITEM::AGED_BRIE)
12         {
13             if (item.quality < 50)
14             {
15                 item.quality = item.quality + 1;
16             }
17             if (item.name == ITEM::BACKSTAGE_PASS)
18             {
19                 if (item.sellIn < 11)
20                 {
21                     if (item.quality < 50)
22                     {
23                         item.quality = item.quality + 1;
24                     }
25                 }
26                 if (item.sellIn < 6)
27                 {
28                     if (item.quality < 50)
29                     {
30                         item.quality = item.quality + 1;
31                     }
32                 }
33             }
34         }
35     }
```

[실습] Refactoring

✓ Step 5-2. 필요없는 코드 삭제하기

- Unit Test를 수행하여, 리팩토링에 문제가 없음을 확인한다.

같은 내용의 if 문이기에
아래 조건문은 삭제해도 된다

```
6 void GildedRose::updateQuality()
7 {
8     for (int i = 0; i < items.size(); i++)
9     {
10         auto &item = items[i];
11         if (item.name == ITEM::AGED_BRIE)
12         {
13             if (item.quality < 50)
14             {
15                 item.quality = item.quality + 1;
16             }
17         }
18         else if (item.name == ITEM::BACKSTAGE_PASS)
19         {
20             if (item.quality < 50)
21             {
22                 item.quality = item.quality + 1;
23             }
24             if (item.name == ITEM::BACKSTAGE_PASS)
25             {
26                 if (item.sellIn < 11)
27                 {
28                     if (item.quality < 50)
29                     {
30                         item.quality = item.quality + 1;
31                     }
32                 }
33             }
34         }
35     }
36 }
```

[참고] Refactoring

✓ Step 6. Replace Nested Conditional with Guard Clauses

- Step 6부터는 복잡한 조건을 간단한 조건문으로 변경한다.
- 출처 : <https://refactoring.com/catalog/replaceNestedConditionalWithGuardClauses.html>

```
function getPayAmount() {  
  let result;  
  if (isDead)  
    result = deadAmount();  
  else {  
    if (isSeparated)  
      result = separatedAmount();  
    else {  
      if (isRetired)  
        result = retiredAmount();  
      else  
        result = normalPayAmount();  
    }  
  }  
  return result;  
}
```



```
function getPayAmount() {  
  if (isDead) return deadAmount();  
  if (isSeparated) return separatedAmount();  
  if (isRetired) return retiredAmount();  
  return normalPayAmount();  
}
```

[실습] Refactoring

✓Step 6-1. 조건문 순서 변경 (수동으로 변경한다.)

- Unit Test를 수행하여, 리팩토링에 문제가 없음을 확인한다.

```
6 void GildedRose::updateQuality()
7 {
8     for (int i = 0; i < items.size(); i++)
9     {
10         auto &item = items[i];
11         if (item.name == ITEM::AGED_BRIE)
12         {
13             if (item.quality < 50)
14             {
15                 item.quality = item.quality + 1;
16             }
17         }
18         else if (item.name == ITEM::BACKSTAGE_PASS)
19         {
20             if (item.quality < 50)
21             {
22                 item.quality = item.quality + 1;
23             }
24             if (item.sellIn < 11)
25             {
26                 if (item.quality < 50)
27                 {
28                     item.quality = item.quality + 1;
29                 }
30             }
31             if (item.sellIn < 6)
32             {
33                 if (item.quality < 50)
34                 {
35                     item.quality = item.quality + 1;
36                 }
37             }
38         }
39     }
40 }
41 else
42 {
43     if (item.quality > 0)
44     {
45         if (item.name != ITEM::SULFURAS)
46         {
47             item.quality = item.quality - 1;
48         }
49     }
50 }
51 }
```

```
}
else
{
    if (item.name != ITEM::SULFURAS)
    {
        if (item.quality > 0)
        {
            item.quality = item.quality - 1;
        }
    }
}
```

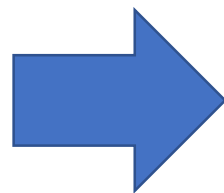
[실습] Refactoring

✓ Step 6-2. 조건문 순서 변경 → if (item.sellIn < 0)

- 순서 변경 후, 테스트 필수

코드 이동

```
57 if (item.sellIn < 0)
58 {
59     if (item.name == ITEM::AGED_BRIE)
60     {
61         if (item.quality < 50)
62         {
63             item.quality = item.quality + 1;
64         }
65     }
66     else
67     {
68         if (item.name == ITEM::BACKSTAGE_PASS)
69         {
70             item.quality = item.quality - item.quality;
71         }
72         else
73         {
74             if (item.quality > 0)
75             {
76                 if (item.name != ITEM::SULFURAS)
77                 {
78                     item.quality = item.quality - 1;
79                 }
80             }
81         }
82     }
83 }
84 }
85 }
```



```
if (item.name == ITEM::AGED_BRIE)
{
    if (item.sellIn < 0)
    {
        if (item.quality < 50)
        {
            item.quality = item.quality + 1;
        }
    }
}
else
{
    if (item.sellIn < 0)
    {
        if (item.name == ITEM::BACKSTAGE_PASS)
        {
            item.quality = item.quality - item.
        }
        else
        {
            if (item.quality > 0)
            {
                if (item.name != ITEM::SULFURAS)
                {
                    item.quality = item.quality - 1;
                }
            }
        }
    }
}
```


[실습] Refactoring

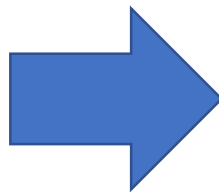
Unit Test 결과
모두 Pass 되어야함

✓ Step 6-3. 조건문 순서 변경 → if (item.sellIn < 0)

- 순서 변경 후, 테스트 필수

코드 이동

```
66 }
67 else
68 {
69     if (item.sellIn < 0)
70     {
71         if (item.name == ITEM::BACKSTAGE_PASS)
72         {
73             item.quality = item.quality - item.quality;
74         }
75         else
76         {
77             if (item.quality > 0)
78             {
79                 if (item.name != ITEM::SULFURAS)
80                 {
81                     item.quality = item.quality - 1;
82                 }
83             }
84         }
85     }
86 }
87 }
88 }
```



```
else
{
    if (item.name == ITEM::BACKSTAGE_PASS)
    {
        if (item.sellIn < 0)
        {
            item.quality = item.quality - item.quality;
        }
    }
    else
    {
        if (item.sellIn < 0)
        {
            if (item.quality > 0)
            {
                if (item.name != ITEM::SULFURAS)
                {
                    item.quality = item.quality - 1;
                }
            }
        }
    }
}
```

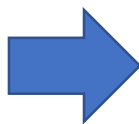
[실습] Refactoring

Unit Test 결과
모두 Pass 되어야함

✓ Step 6-4. invert – if statement

- if 에서 Alt + Enter 후, if state 변경하기

```
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
  
else  
{  
    if (item.sellIn < 0)  
    {  
        if (item.quality > 0)  
        {  
            if (item.name != ITEM::SULFURAS)  
            {  
                item.quality = item.quality - 1;  
            }  
        }  
    }  
}
```



```
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
  
else  
{  
    if (item.sellIn < 0)  
    {  
        if (item.quality > 0)  
        {  
            if (item.name == ITEM::SULFURAS)  
            {  
            }  
            else  
            {  
                item.quality = item.quality - 1;  
            }  
        }  
    }  
}
```

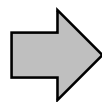
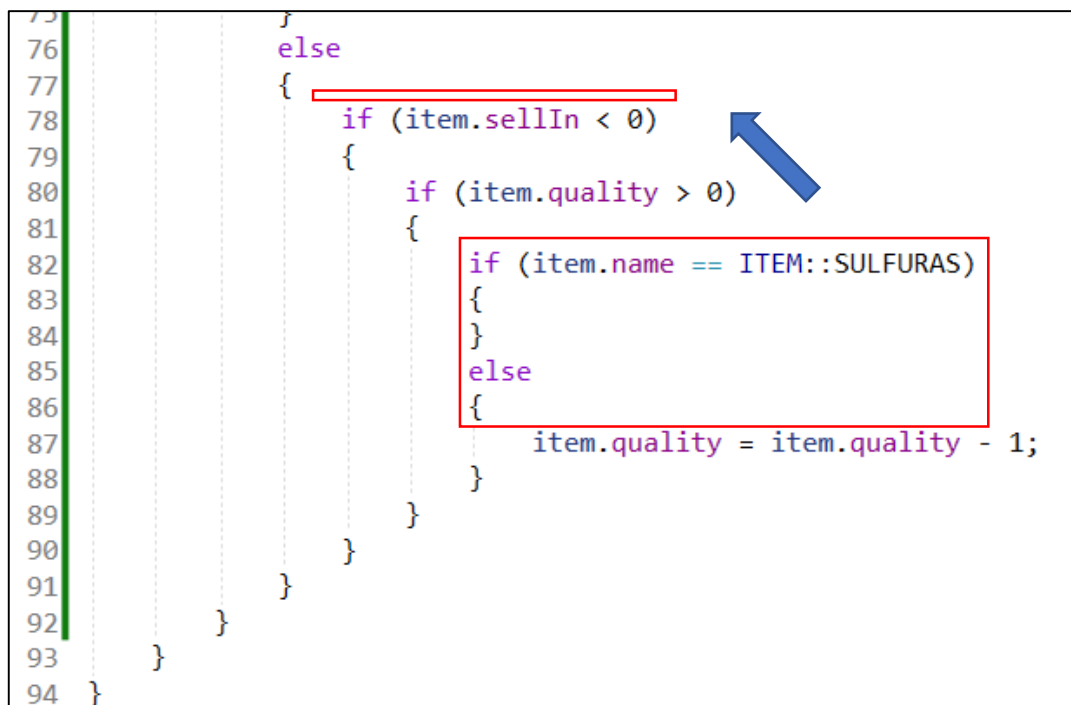
[실습] Refactoring

Unit Test 결과
모두 Pass 되어야함

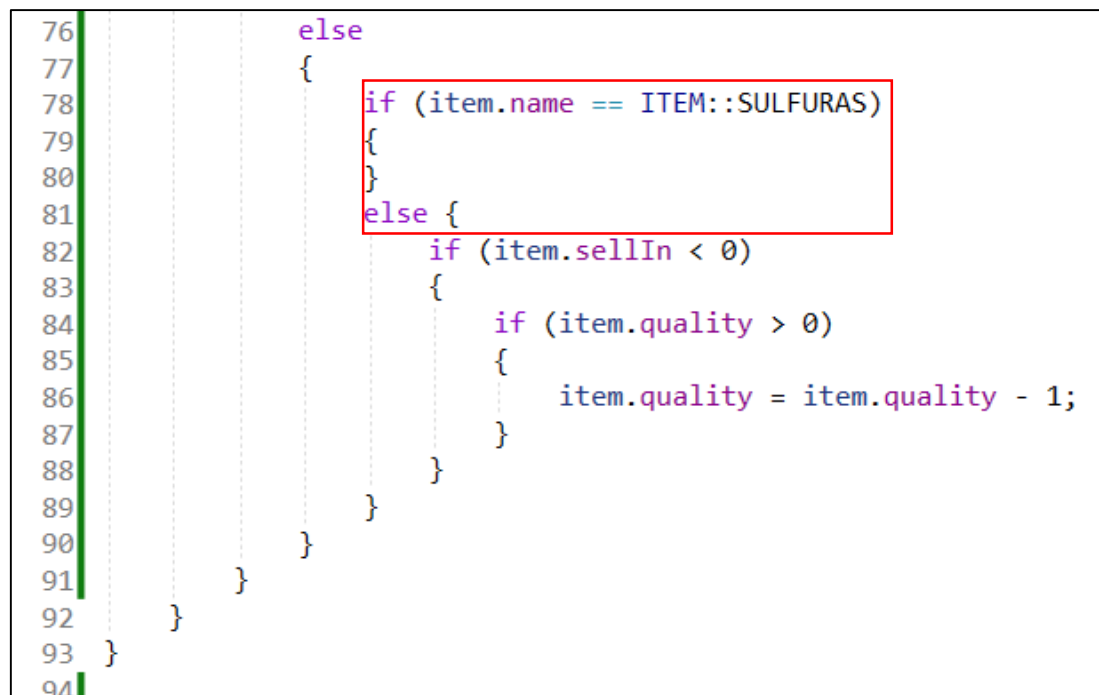
✓ Step 6-5. 이름 비교 조건을 상위 단계로 이동

- 순서 변경 후, 테스트 필수

```
75 }
76 else
77 {
78     if (item.sellIn < 0)
79     {
80         if (item.quality > 0)
81         {
82             if (item.name == ITEM::SULFURAS)
83             {
84             }
85             else
86             {
87                 item.quality = item.quality - 1;
88             }
89         }
90     }
91 }
92 }
93 }
94 }
```



```
76 else
77 {
78     if (item.name == ITEM::SULFURAS)
79     {
80     }
81     else {
82         if (item.sellIn < 0)
83         {
84             if (item.quality > 0)
85             {
86                 item.quality = item.quality - 1;
87             }
88         }
89     }
90 }
91 }
92 }
93 }
94 }
```



[실습] Refactoring

✓Step 7-1. else 와 if 문을 하나로 Merge

```
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94
```

```
    else  
    {  
        if (item.name == ITEM::SULFURAS)  
        {  
        }  
        else {  
            if (item.sellIn < 0)  
            {  
                if (item.quality > 0)  
                {  
                    item.quality = item.quality - 1;  
                }  
            }  
        }  
    }  
}
```



```
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91
```

```
    else if (item.name == ITEM::SULFURAS)  
    {  
    }  
    else {  
        if (item.sellIn < 0)  
        {  
            if (item.quality > 0)  
            {  
                item.quality = item.quality - 1;  
            }  
        }  
    }  
}
```

else, if 를 합쳐 else if 로 변경

[실습] Refactoring

Unit Test 결과
모두 Pass 되어야함

✓ Step 7-2. else 와 if 문을 하나로 Merge

```
67
68
69     else
70     {
71         if (item.name == ITEM::BACKSTAGE_PASS)
72         {
73             if (item.sellIn < 0)
74             {
75                 item.quality = item.quality - item.quality;
76             }
77         }
78         else if (item.name == ITEM::SULFURAS)
79         {
80             if (item.sellIn < 0)
81             {
82                 if (item.quality > 0)
83                 {
84                     item.quality = item.quality - 1;
85                 }
86             }
87         }
88     }
89 }
90 }
```



```
67     else if (item.name == ITEM::BACKSTAGE_PASS)
68     {
69         if (item.sellIn < 0)
70         {
71             item.quality = item.quality - item.quality;
72         }
73     }
74     else if (item.name == ITEM::SULFURAS)
75     {
76         { 를 아래로 내려줌
77         else
78         {
79             if (item.sellIn < 0)
80             {
81                 if (item.quality > 0)
82                 {
83                     item.quality = item.quality - 1;
84                 }
85             }
86         }
87     }
88 }
```

[실습] Refactoring

✓ Step 7-3. else 와 if 문을 하나로 Merge

```
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51
```

```
    }  
    }  
    }  
    else  
    {  
        if (item.name != ITEM::SULFURAS)  
        {  
            if (item.quality > 0)  
            {  
                item.quality = item.quality - 1;  
            }  
        }  
    }  
}
```



```
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51
```

```
    }  
    }  
    }  
    else if (item.name != ITEM::SULFURAS)  
    {  
        if (item.quality > 0)  
        {  
            item.quality = item.quality - 1;  
        }  
    }  
}
```

Refactoring (실습)

Unit Test 결과
모두 Pass 되어야함

✓ Step 7-4. else 문 추가하기

- 부정 조건을 긍정조건으로 바꾸고, else 코드로 처리한다.

```
40 }  
41  
42 else if (item.name != ITEM::SULFURAS)  
43 {  
44     if (item.quality > 0)  
45     {  
46         item.quality = item.quality - 1;  
47     }  
48 }
```



```
40 }  
41  
42 else if (item.name == ITEM::SULFURAS)  
43 {  
44 }  
45 else  
46 {  
47     if (item.quality > 0)  
48     {  
49         item.quality = item.quality - 1;  
50     }  
51 }
```

Refactoring (실습)

✓ Step 8-1. 2개의 if절을 하나로 병합

- AGED_BRIE 조건문이,
위와 아래에 각각 존재하여,
아래에 있는 코드를 위로 이동시키자.

해당 코드들을,
잘라내어,
왼쪽으로 모두 이동시킴

```
6 void GildedRose::updateQuality()
7 {
8     for (int i = 0; i < items.size(); i++)
9     {
10         auto &item = items[i];
11         if (item.name == ITEM::AGED_BRIE)
12         {
13             if (item.quality < 50)
14             {
15                 item.quality = item.quality + 1;
16             }
17         }
18         else if (item.name == ITEM::BACKSTAGE_PASS)
19         {
20             if (item.quality < 50)
21             {
22                 item.quality = item.quality + 1;
23             }
24             if (item.sellIn < 11)
25             {
26                 if (item.quality < 50)
27                 {
28                     item.quality = item.quality + 1;
29                 }
30             }
31             if (item.sellIn < 6)
32             {
33                 if (item.quality < 50)
34                 {
35                     item.quality = item.quality + 1;
36                 }
37             }
38         }
39     }
40 }
41 else if (item.name == ITEM::SULFURAS)
42 {
43 }
44 else
45 {
46     if (item.quality > 0)
47     {
48         item.quality = item.quality - 1;
49     }
50 }
51 if (item.name != ITEM::SULFURAS)
52 {
53     item.sellIn = item.sellIn - 1;
54 }
55 }
56 if (item.name == ITEM::AGED_BRIE)
57 {
58     if (item.sellIn < 0)
59     {
60         if (item.quality < 50)
61         {
62             item.quality = item.quality + 1;
63         }
64     }
65 }
66 }
67 else if (item.name == ITEM::BACKSTAGE_PASS)
68 {
69     if (item.sellIn < 0)
```


Refactoring (실습)

Unit Test 결과
모두 Pass 되어야함

✓Step 8-2. 2개의 if절을 하나로 병합

- 하위 조건(AGED_BRIE)의 내용을 상위 조건(AGED_BRIE)와 묶음
- 버그 방지를 위해, sellIn 조건 변경(0->1), **그리고 테스트 이상없음을 확인하기**

```
6 void GildedRose::updateQuality()  
7 {  
8     for (int i = 0; i < items.size(); i++)  
9     {  
10         auto &item = items[i];  
11         if (item.name == ITEM::AGED_BRIE)  
12         {  
13             if (item.quality < 50)  
14             {  
15                 item.quality = item.quality + 1;  
16             }  
17  
18             if (item.sellIn < 1)  
19             {  
20                 if (item.quality < 50)  
21                 {  
22                     item.quality = item.quality + 1;  
23                 }  
24             }  
25         }  
26     }  
27 }
```

0 -> 1 변

Refactoring (실습)

✓ Step 8-3. 2개의 if절을 하나로 병합

- BACKSTAGE_PASS 조건문이,
위와 아래에 각각 존재하여,
아래에 있는 코드를 위로 이동시키자.

해당 코드들을,
잘라내어,
왼쪽으로 모두 이동시킴

```
24 }
25 else if (item.name == ITEM::BACKSTAGE_PASS)
26 {
27     if (item.quality < 50)
28     {
29         item.quality = item.quality + 1;
30
31         if (item.sellIn < 11)
32         {
33             if (item.quality < 50)
34             {
35                 item.quality = item.quality + 1;
36             }
37         }
38
39         if (item.sellIn < 6)
40         {
41             if (item.quality < 50)
42             {
43                 item.quality = item.quality + 1;
44             }
45         }
46     }
47 }
48 else if (item.name == ITEM::SULFURAS)
49 {
50 }
51 else
52 {
53     if (item.quality > 0)
54     {
55         item.quality = item.quality - 1;
56     }
57 }
58
59 if (item.name != ITEM::SULFURAS)
60 {
61     item.sellIn = item.sellIn - 1;
62 }
63
64 if (item.name == ITEM::AGED_BRIE)
65 {
66 }
67 else if (item.name == ITEM::BACKSTAGE_PASS)
68 {
69     if (item.sellIn < 0)
70     {
71         item.quality = item.quality - item.quality;
72     }
73 }
74 else if (item.name == ITEM::SULFURAS)
```

Refactoring (실습)

Unit Test 결과
모두 Pass 되어야함

✓Step 8-4. 2개의 if절을 하나로 병합

- 하위 조건(**BACKSTAGE_PASS**)의 내용을 상위 조건(**BACKSTAGE_PASS**)와 묶음
- 버그 방지를 위해, sellIn 조건 변경(0->1), **그리고 테스트 이상없음을 확인하기**

```
25     else if (item.name == ITEM::BACKSTAGE_PASS)
26     {
27         if (item.quality < 50)
28         {
29             item.quality = item.quality + 1;
30
31             if (item.sellIn < 11)
32             {
33                 if (item.quality < 50)
34                 {
35                     item.quality = item.quality + 1;
36                 }
37             }
38
39             if (item.sellIn < 6)
40             {
41                 if (item.quality < 50)
42                 {
43                     item.quality = item.quality + 1;
44                 }
45             }
46         }
47         if (item.sellIn < 1)
48         {
49             item.quality = item.quality - item.quality;
50         }
51     }
52     else if (item.name == ITEM::SULFURAS)
53     {
54     }
```

0 -> 1 번

Refactoring (실습)

✓ Step 8-5. 2개의 if절을 하나로 병합

- **else 조건문이,**
위와 아래에 각각 존재하여,
아래에 있는 코드를 위로 이동시키자.

else 문에 있는
내용들 전부
왼쪽 else로 이동시킨다.

```
53 {  
54 }  
55 else  
56 {  
57     if (item.quality > 0)  
58     {  
59         item.quality = item.quality - 1;  
60     }  
61 }  
62  
63 if (item.name != ITEM::SULFURAS)  
64 {  
65     item.sellIn = item.sellIn - 1;  
66 }  
67  
68 if (item.name == ITEM::AGED_BRIE)  
69 {  
70 }  
71 else if (item.name == ITEM::BACKSTAGE_PASS)  
72 {  
73 }  
74 }  
75 else if (item.name == ITEM::SULFURAS)  
76 {  
77 }  
78 else  
79 {  
80     if (item.sellIn < 0)  
81     {  
82         if (item.quality > 0)  
83         {  
84             item.quality = item.quality - 1;  
85         }  
86     }  
87 }  
88 }  
89 }  
90 }
```

Refactoring (실습)

✓Step 8-6. 2개의 if절을 하나로 병합

- 하위 조건(else)의 내용을 상위 조건(else)와 묶음
- 버그방지를 위해, 합치면서 sellIn조건 변경(0->1)

```
52     else if (item.name == ITEM::SULFURAS)
53     {
54     }
55     else
56     {
57         if (item.quality > 0)
58         {
59             item.quality = item.quality - 1;
60         }
61         if (item.sellIn < 1)
62         {
63             if (item.quality > 0)
64             {
65                 item.quality = item.quality - 1;
66             }
67         }
68     }
69 }
```

0 -> 1 변

Refactoring (실습)

Unit Test 결과
모두 Pass 되어야함

✓Step 9. 불필요한 구문 삭제

- 삭제 후, 유닛테스트에 Pass를 확인한다.

```
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90
```

```
    if (item.name != ITEM::SULFURAS)  
    {  
        item.sellIn = item.sellIn - 1;  
    }  
  
    if (item.name == ITEM::AGED_BRIE)  
    {  
    }  
    else if (item.name == ITEM::BACKSTAGE_PASS)  
    {  
    }  
    else if (item.name == ITEM::SULFURAS)  
    {  
    }  
    else  
    {  
    }  
}
```

삭제

Refactoring (실습)

✓Step 10-1. 메서드 추출하기

- 메서드 추출하기 좋게 구분되었다.

✓다음과 같이 메서드 추출을 시도한다.

- 오래된 치즈 : `updateQualityForAgedBrie(item);`
- 콘서트 티켓 : `updateQualityForBackstagePasses(item);`
- 일반아이템 : `updateQualityForNormalItem(item);`

오래된 치즈

콘서트 티켓

전설의아이템

일반아이템

```
6 void GildedRose::updateQuality()
7 {
8     for (int i = 0; i < items.size(); i++)
9     {
10         auto &item = items[i];
11         if (item.name == ITEM::AGED_BRIE)
12         {
13             if (item.quality < 50)
14             {
15                 item.quality = item.quality + 1;
16             }
17             if (item.sellIn < 1)
18             {
19                 if (item.quality < 50)
20                 {
21                     item.quality = item.quality + 1;
22                 }
23             }
24         }
25         else if (item.name == ITEM::BACKSTAGE_PASS)
26         {
27             if (item.quality < 50)
28             {
29                 item.quality = item.quality + 1;
30             }
31             if (item.sellIn < 11)
32             {
33                 if (item.quality < 50)
34                 {
35                     item.quality = item.quality + 1;
36                 }
37             }
38             if (item.sellIn < 6)
39             {
40                 if (item.quality < 50)
41                 {
42                     item.quality = item.quality + 1;
43                 }
44             }
45             if (item.sellIn < 1)
46             {
47                 item.quality = item.quality - item.quality;
48             }
49         }
50         else if (item.name == ITEM::SULFURAS)
51         {
52             if (item.quality > 0)
53             {
54                 item.quality = item.quality - 1;
55             }
56             if (item.sellIn < 1)
57             {
58                 if (item.quality > 0)
59                 {
60                     item.quality = item.quality - 1;
61                 }
62             }
63         }
64         if (item.name != ITEM::SULFURAS)
65         {
66             item.sellIn = item.sellIn - 1;
67         }
68     }
69 }
```

Refactoring (실습)

✓Step 10-2. 메서드 추출하기

✓메서드 추출

- updateSellIn(item);

```
64 void GildedRose::updateQuality()  
65 {  
66     for (int i = 0; i < items.size(); i++)  
67     {  
68         auto &item = items[i];  
69         if (item.name == ITEM::AGED_BRIE)  
70         {  
71             updateQualityForAgedBrie(item);  
72         }  
73         else if (item.name == ITEM::BACKSTAGE_PASS)  
74         {  
75             updateQualityForBackstagePasses(item);  
76         }  
77         else if (item.name == ITEM::SULFURAS)  
78         {  
79             //updateQualityForSulfuras(item);  
80         }  
81         else  
82         {  
83             updateQualityForNormalItem(item);  
84         }  
85  
86         if (item.name != ITEM::SULFURAS)  
87         {  
88             item.sellIn = item.sellIn - 1;  
89         }  
90     }  
91 }
```


[실습] Refactoring

✓Step 11. 비어있는 메서드 하나 만들기

- Unit Test를 수행하여, 리팩토링에 문제가 없음을 확인한다.

```
72 void GildedRose::updateQuality()  
73 {  
74     for (int i = 0; i < items.size(); i++)  
75     {  
76         auto &item = items[i];  
77         if (item.name == ITEM::AGED_BRIE)  
78         {  
79             updateQualityForAgedBrie(item);  
80         }  
81         else if (item.name == ITEM::BACKSTAGE_PASS)  
82         {  
83             updateQualityForBackstagePasses(item);  
84         }  
85         else if (item.name == ITEM::SULFURAS)  
86         {  
87             updateQualityForSulfuras(item);  
88         }  
89         else  
90         {  
91             updateQualityForNormalItem(item);  
92         }  
93     }  
94     updateSellIn(item);  
95 }  
96 }  
97  
98 void GildedRose::updateQualityForSulfuras(const Item& item)  
99 {  
100 }  
101 }
```

3. Class Level Refactoring

[실습] Refactoring - 재설계

✓Step 1. Class 생성

- Step 1-1 부터, 각 아이템 별로, 클래스를 만들 것이다.
 - AgedBriarItem : 오래된치즈 Class
 - BackstagePassesItem : 콘서트티켓 Class
 - SulfurasItem : 전설의아이템 Class
 - NormalItem : 일반아이템 Class

[실습] Refactoring - 재설계

✓Step1. Item.h 파일에 AgedBrieItem 클래스 추가

```
#pragma once
#include "GildedRose.h"

class AgedBrieItem {
private:
    Item* item;
};
```

Items.h 파일을 추가하여 코드 기입

```
#pragma once
#include <string>
#include <vector>

using namespace std;

namespace ITEM
{
    const string SULFURAS = "Sulfuras, Hand of R";
    const string AGED_BRIE = "Aged Brie";
    const string BACKSTAGE_PASS = "Backstage pas";
    const string NONAME = "noname";
}
```

헤더가드 삽입
(중복 Include 방지 코드)

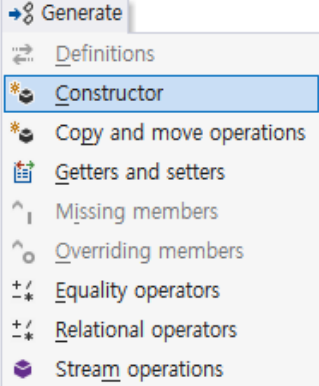

[실습] Refactoring - 재설계

- ✓cpp와 header 역할이 분리되어야 하지만,
실습 안내 편의상, Items.h 파일에 구현까지 진행함.

Items.h

```
#pragma once
#include "GildedRose.h"

class AgedBrieItem {
private:
    Item* item;
};
```



```
#pragma once
#include "GildedRose.h"

class AgedBrieItem {
public:
    explicit AgedBrieItem(Item* item)
        : item_(item) {
    }

private:
    Item* item;
};
```

Alt + Insert 눌러, 생성자 자동 삽입

[실습] Refactoring - 재설계

✓ Step 2. AgedBrie 함수를 클래스로 이동

```
#include "GildedRose.h"

GildedRose::GildedRose(vector<Item>& items) : items(items)
{}

void GildedRose::updateQualityForAgedBrie(Item& item) {
    if (item.quality < 50)
    {
        item.quality = item.quality + 1;
    }
    if (item.sellIn < 1)
    {
        if (item.quality < 50)
        {
            item.quality = item.quality + 1;
        }
    }
}
```

GildedRose.cpp

```
#include "GildedRose.h"

class AgedBrieItem {
public:
    explicit AgedBrieItem(Item* item)
        : item_(item) {
    }

    void GildedRose::updateQualityForAgedBrie(Item& item) {
        if (item.quality < 50)
        {
            item.quality = item.quality + 1;
        }
        if (item.sellIn < 1)
        {
            if (item.quality < 50)
            {
                item.quality = item.quality + 1;
            }
        }
    }

private:
    Item* item_;
};
```

Items.h

위 코드를 Items.h 로 이동시킨다.

[실습] Refactoring - 재설계

✓문법 에러가 나지 않도록, 코드를 수정한다.

```
#include "GildedRose.h"
```

Items.h

```
class AgedBrieItem {  
public:  
    explicit AgedBrieItem(Item* item)  
        : item_(item) {  
삭제  
        void GildedRose::updateQualityForAgedBrie(Item& item) {  
            if (item.quality < 50)  
            {  
                item.quality = item.quality + 1;  
            }  
            if (item.sellIn < 1)  
            {  
                if (item.quality < 50)  
                {  
                    item.quality = item.quality + 1;  
                }  
            }  
        }  
    }  
private:  
    Item* item_;  
};
```



```
void updateQualityForAgedBrie() {  
    Item& item = *item; 코드 추가하기  
    if (item.quality < 50)  
    {  
        item.quality = item.quality + 1;  
    }  
    if (item.sellIn < 1)  
    {  
        if (item.quality < 50)  
        {  
            item.quality = item.quality + 1;  
        }  
    }  
}
```

[실습] Refactoring - 재설계

✓GildedRose.cpp 에 제작한 클래스를 적용한다.

✓ GildedRoseTest (10 tests)	Success
✓ agedBrie_sellin_0_quality_0	Success
✓ agedBrie_sellin_0_quality_50	Success
✓ backstage_pass_sellin_0_quality_0	Success
✓ backstage_pass_sellin_0_quality_49	Success
✓ backstage_pass_sellin_12_quality_0	Success
✓ noname_sellin_0_quality_0	Success
✓ noname_sellin_0_quality_1	Success
✓ should_be_nothing_when_no_item	Success
✓ sulfuras_sellin_0_quality_80	Success
✓ sulfuras_sellin_m2_quality_80	Success

```
#include "GildedRose.h"
#include "Items.h"

GildedRose::GildedRose(vector<Item>
{

}

void GildedRose::updateQualityForAg
    if (item.quality < 50)
    {
        item.quality = item.quality
    }
    if (item.sellIn < 1)
```

헤더 Include 하기

```
void GildedRose::updateQuality()
{
    for (int i = 0; i < items.size(); i++)
    {
        auto& item = items[i];
        if (item.name == ITEM::AGED_BRIE)
        {
            AgedBrieItem* agedBrieItem = new AgedBrieItem(&item);
            agedBrieItem->updateQualityForAgedBrie();
        }
        else if (item.name == ITEM::BACKSTAGE_PASS)
        {
            updateQualityForBackstagePasses(item);
        }
        else if (item.name == ITEM::SULFURAS)
```

updateQuality() 메서드에 제작한 클래스 적용하기

[도전] Refactoring - 재설계

✓ Step 3. 실습 (Step1, Step2 참조)

- Step 1 방법을 참조하여 BackstagePassesItem, SulfurasItem, NormalItem Class를 생성한다.
- Step 2 방법을 참조하여 updateQualityForBackstagePasses(), updateQualityForSulfuras(), updateQualityForNormalItem()를 각 class로 이동시킨다.
- 리팩토링에 이상이 없는지 유닛테스트를 수행해본다.

GildedRose.cpp

```
void GildedRose::updateQuality()
{
    for (int i = 0; i < items.size(); i++)
    {
        auto& item = items[i];
        if (item.name == ITEM::AGED_BRIE)
        {
            AgedBrieItem* agedBrieItem = new AgedBrieItem(&item);
            agedBrieItem->updateQualityForAgedBrie();
        }
        else if (item.name == ITEM::BACKSTAGE_PASS)
        {
            BackstagePassesItem* backstagePassesItem = new BackstagePassesItem(&item);
            backstagePassesItem->updateQualityForBackstagePasses();
        }
        else if (item.name == ITEM::SULFURAS)
        {
            SulfurasItem* sulfurasItem = new SulfurasItem(&item);
            sulfurasItem->updateQualityForSulfuras();
        }
        else
        {
            NormalItem* normalItem = new NormalItem(&item);
            normalItem->updateQualityForNormalItem();
        }

        updateSellIn(item);
    }
}
```

Step 3 수행 완료 후, 소스코드 상태

[실습] Refactoring - 재설계

✓ Step 4. method 이름 변경

- updateQuality***() 메서드 이름을 **updateQuality()**로 통일하기
- 각 메서드마다 이름 바꾸기를 수행한다.

```
void GildedRose::updateQuality()
{
    for (int i = 0; i < items.size(); i++)
    {
        auto& item = items[i];
        if (item.name == ITEM::AGED_BRIE)
        {
            AgedBrieItem* agedBrieItem = new AgedBrieItem(&item);
            agedBrieItem->updateQualityForAgedBrie();
        }
        else if (item.name == ITEM::BACKSTAGE_PASS)
        {
            BackstagePassesItem* backstagePassesItem = new BackstagePassesItem(&item);
            backstagePassesItem->updateQualityForBackstagePasses();
        }
        else if (item.name == ITEM::SULFURAS)
        {
            SulfurasItem* sulfurasItem = new SulfurasItem(&item);
            sulfurasItem->updateQualityForSulfuras();
        }
        else
        {
            NormalItem* normalItem = new NormalItem(&item);
            normalItem->updateQualityForNormalItem();
        }

        updateSellIn(item);
    }
}
```

GildedRose.cpp



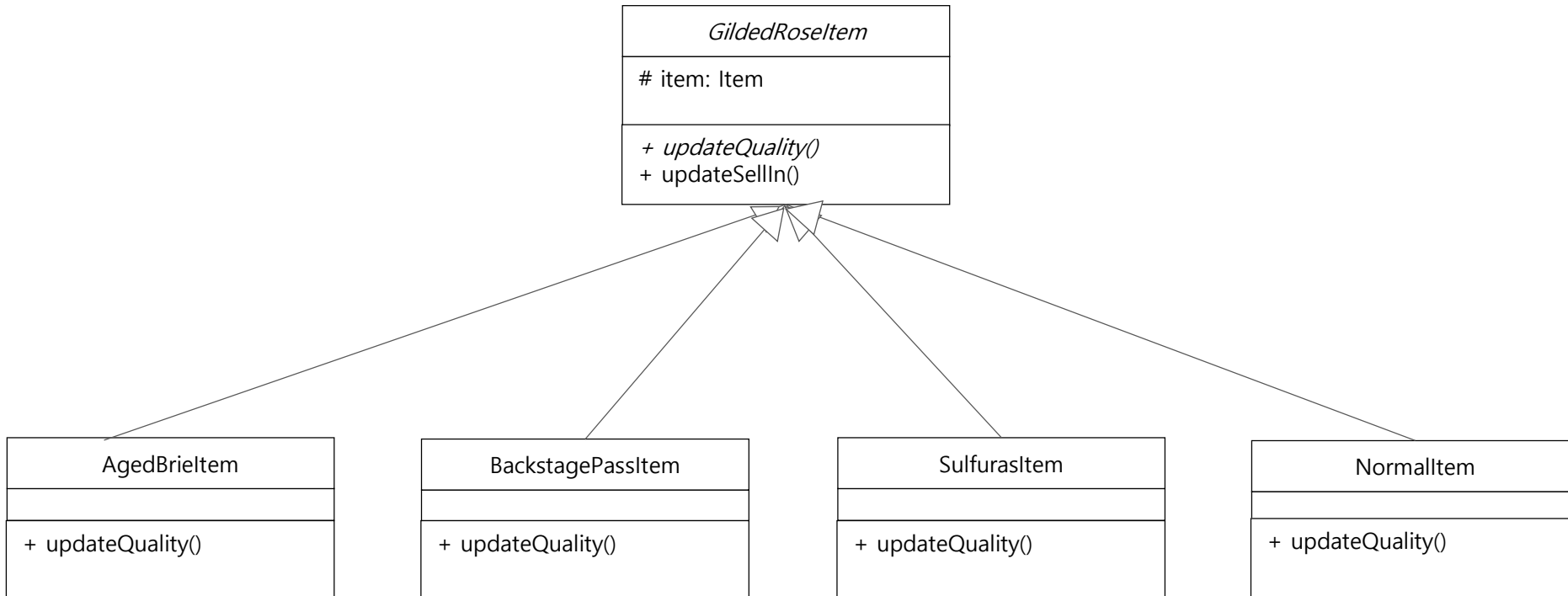
```
auto& item = items[i];
if (item.name == ITEM::AGED_BRIE)
{
    AgedBrieItem* agedBrieItem = new AgedBrieItem(&item);
    agedBrieItem->updateQuality();
}
else if (item.name == ITEM::BACKSTAGE_PASS)
{
    BackstagePassesItem* backstagePassesItem = new BackstagePassesItem(&item);
    backstagePassesItem->updateQuality();
}
else if (item.name == ITEM::SULFURAS)
{
    SulfurasItem* sulfurasItem = new SulfurasItem(&item);
    sulfurasItem->updateQuality();
}
else
{
    NormalItem* normalItem = new NormalItem(&item);
    normalItem->updateQuality();
}

updateSellIn(item);
```

[실습] Refactoring - 재설계

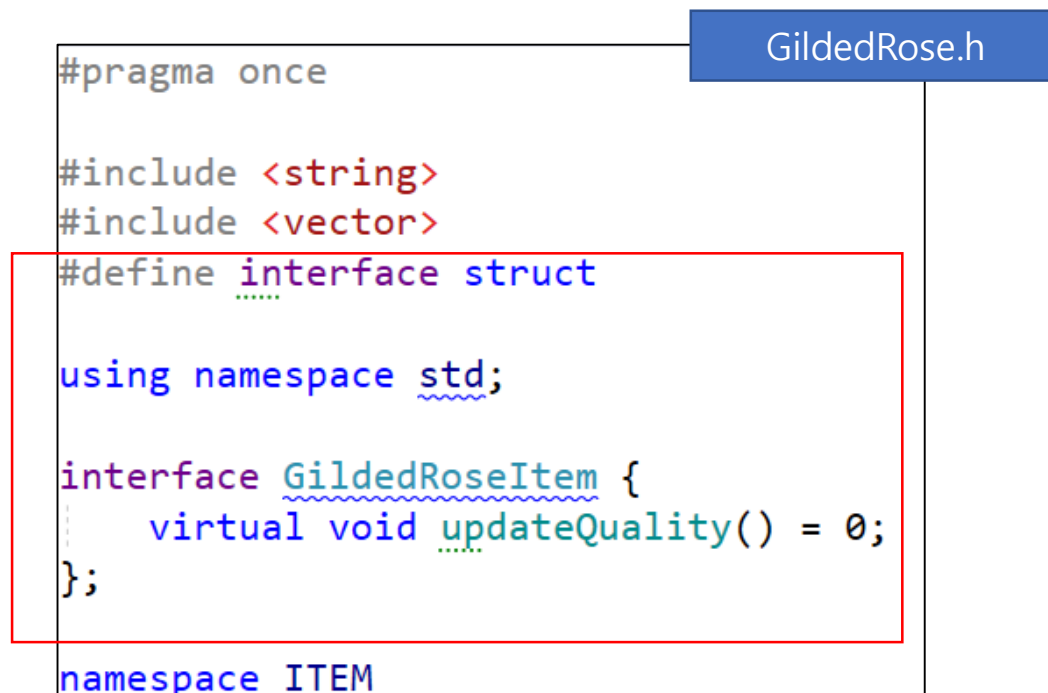
✓목표

- 중복코드를 제거하기 위해, 상위 객체를 만들고, 상속받는 형태로 리팩토링



[실습] Refactoring - 재설계

✓GildedRose.h 에 Interface를 추가한다.



The diagram illustrates the refactoring process. A blue box labeled "GildedRose.h" is connected by a line to a larger box containing C++ code. A red rectangle highlights the new interface definition being added to the file.

```
#pragma once

#include <string>
#include <vector>
#define interface struct

using namespace std;

interface GildedRoseItem {
    virtual void updateQuality() = 0;
};

namespace ITEM
```

[실습] Refactoring - 재설계

- ✓ 기존 제작한 Class에 Interface를 적용한다.
 - 총 4개의 Class에 적용한다.

```
class AgedBrieItem : public GildedRoseItem {  
public:  
    explicit AgedBrieItem(Item* item)  
        : item_(item) {  
    }  
  
    void updateQuality() {
```

```
class BackstagePassesItem : public GildedRoseItem {  
public:  
    explicit BackstagePassesItem(Item* item)  
        : item_(item) {  
    }  
  
    void updateQuality() {  
        Item* item = *item_;
```

```
class SulfurasItem : public GildedRoseItem {  
public:  
    explicit SulfurasItem(Item* item)  
        : item_(item) {  
    }  
  
    void updateQuality() {
```

```
class NormalItem : public GildedRoseItem {  
public:  
    explicit NormalItem(Item* item)  
        : item_(item) {  
    }  
  
    void updateQuality() {
```

[실습] Refactoring - 재설계

✓ 각 Class의 Instance 이름을 모두 같은 이름으로 변경한다.

- 모두 **gildedRoseItem** 이라는 이름으로 변경

```
if (item.name == ITEM::AGED_BRIE)
{
    AgedBrieItem* agedBrieItem = new AgedBrieItem(&item);
    agedBrieItem->updateQuality();
}
else if (item.name == ITEM::BACKSTAGE_PASS)
{
    BackstagePassesItem* backstagePassesItem = new BackstagePassesItem(&item);
    backstagePassesItem->updateQuality();
}
else if (item.name == ITEM::SULFURAS)
{
    SulfurasItem* sulfurasItem = new SulfurasItem(&item);
    sulfurasItem->updateQuality();
}
else
{
    NormalItem* normalItem = new NormalItem(&item);
    normalItem->updateQuality();
}
```

GildedRose.cpp



```
auto& item = items[i];
if (item.name == ITEM::AGED_BRIE)
{
    AgedBrieItem* gildedRoseItem = new AgedBrieItem(&item);
    gildedRoseItem->updateQuality();
}
else if (item.name == ITEM::BACKSTAGE_PASS)
{
    BackstagePassesItem* gildedRoseItem = new BackstagePassesItem(&item);
    gildedRoseItem->updateQuality();
}
else if (item.name == ITEM::SULFURAS)
{
    SulfurasItem* gildedRoseItem = new SulfurasItem(&item);
    gildedRoseItem->updateQuality();
}
else
{
    NormalItem* gildedRoseItem = new NormalItem(&item);
    gildedRoseItem->updateQuality();
}
```

[실습] Refactoring - 재설계


✓다형성 코드로 변환 후, 중복코드를 제거한다.

GildedRose.cpp



```
auto& item = items[i];
if (item.name == ITEM::AGED_BRIE)
{
    GildedRoseItem* gildedRoseItem = new AgedBrieItem(&item);
    gildedRoseItem->updateQuality();
}
else if (item.name == ITEM::BACKSTAGE_PASS)
{
    GildedRoseItem* gildedRoseItem = new BackstagePassesItem(&item);
    gildedRoseItem->updateQuality();
}
else if (item.name == ITEM::SULFURAS)
{
    GildedRoseItem* gildedRoseItem = new SulfurasItem(&item);
    gildedRoseItem->updateQuality();
}
else
{
    GildedRoseItem* gildedRoseItem = new NormalItem(&item);
    gildedRoseItem->updateQuality();
}
```

1. 각 클래스를 다형성 코드로 변환한다.



```
auto& item = items[i];
GildedRoseItem* gildedRoseItem = nullptr;

if (item.name == ITEM::AGED_BRIE)
{
    gildedRoseItem = new AgedBrieItem(&item);
}
else if (item.name == ITEM::BACKSTAGE_PASS)
{
    gildedRoseItem = new BackstagePassesItem(&item);
}
else if (item.name == ITEM::SULFURAS)
{
    gildedRoseItem = new SulfurasItem(&item);
}
else
{
    gildedRoseItem = new NormalItem(&item);
}
gildedRoseItem->updateQuality();
updateSellIn(item);
```

2. 중복코드를 제거한다.

[실습] Refactoring - 재설계

✓메서드 추상화

GildedRose.cpp

```
auto& item = items[i];
GildedRoseItem* gildedRoseItem = nullptr;

if (item.name == ITEM::AGED_BRIE)
{
    gildedRoseItem = new AgedBrieItem(&item);
}
else if (item.name == ITEM::BACKSTAGE_PASS)
{
    gildedRoseItem = new BackstagePassesItem(&item);
}
else if (item.name == ITEM::SULFURAS)
{
    gildedRoseItem = new SulfurasItem(&item);
}
else
{
    gildedRoseItem = new NormalItem(&item);
}
gildedRoseItem->updateQuality();
updateSellIn(item);
```

메서드추출

```
void GildedRose::updateQuality()
{
    for (int i = 0; i < items.size(); i++)
    {
        auto& item = items[i];
        updateQuality(item);
        updateSellIn(item);
    }
}
```


[실습] Refactoring - 재설계

- ✓클래스를 생성해주는 Factory 역할을 하는 메서드 선언부를 추가한다.

```
class GildedRose
{
public:
    vector<Item>& items;
    GildedRose(vector<Item>& items);
    void updateSellIn(Item& item);
    void updateQuality(Item& item);
    GildedRoseItem* getGildedRoseItem(Item& item);

    void updateQuality();
};
```

GildedRose.h

메서드 추출을 위해,
미리 Header에 위 코드를 추가한다.

[실습] Refactoring - 재설계

✓한번 더 메서드 추출하기

```
void GildedRose::updateQuality(Item& item) {  
    GildedRoseItem* gildedRoseItem = nullptr;  
  
    if (item.name == ITEM::AGED_BRIE)  
    {  
        gildedRoseItem = new AgedBrieItem(&item);  
    }  
    else if (item.name == ITEM::BACKSTAGE_PASS)  
    {  
        gildedRoseItem = new BackstagePassesItem(&item);  
    }  
    else if (item.name == ITEM::SULFURAS)  
    {  
        gildedRoseItem = new SulfurasItem(&item);  
    }  
    else  
    {  
        gildedRoseItem = new NormalItem(&item);  
    }  
    gildedRoseItem->updateQuality();  
}
```

```
GildedRoseItem* GildedRose::getGildedRoseItem(Item& item) { 코드 정리하기  
    GildedRoseItem* gildedRoseItem = nullptr;  
    if (item.name == ITEM::AGED_BRIE) return new AgedBrieItem(&item);  
    if (item.name == ITEM::BACKSTAGE_PASS) return new BackstagePassesItem(&item);  
    if (item.name == ITEM::SULFURAS) return new SulfurasItem(&item);  
    return new NormalItem(&item);  
}  
  
void GildedRose::updateQuality(Item& item) {  
    GildedRoseItem* gildedRoseItem = getGildedRoseItem(item);  
    gildedRoseItem->updateQuality();  
}
```

리팩토링 완료

✓ 완성된 소스코드

- 메서드 순서를 읽기 편리하도록 배치를 변경한다.

```
#include "GildedRose.h"
#include "Items.h"

GildedRose::GildedRose(vector<Item>& items) : items(items)
{}

void GildedRose::updateQuality()
{
    for (int i = 0; i < items.size(); i++)
    {
        auto& item = items[i];
        updateQuality(item);
        updateSellIn(item);
    }
}

void GildedRose::updateQuality(Item& item) {
    GildedRoseItem* gildedRoseItem = getGildedRoseItem(item);
    gildedRoseItem->updateQuality();
}

GildedRoseItem* GildedRose::getGildedRoseItem(Item& item) {
    GildedRoseItem* gildedRoseItem = nullptr;
    if (item.name == ITEM::AGED_BRIE) return new AgedBrieItem(&item);
    if (item.name == ITEM::BACKSTAGE_PASS) return new BackstagePassesItem(&item);
    if (item.name == ITEM::SULFURAS) return new SulfurasItem(&item);
    return new NormalItem(&item);
}

void GildedRose::updateSellIn(Item& item) {
    if (item.name != ITEM::SULFURAS)
    {
        item.sellIn = item.sellIn - 1;
    }
}
```

Quality와 SellIn을 각각 update한다.

Factory 역할