Instituto Federal de Educação Ciência e Tecnologia de São Paulo
Campus Guarulhos
Análise e Desenvolvimento de Sistemas (ADS)

-

ENGENHARIA DE SOFTWARE - REFATORAÇÃO DE CÓDIGO FONTE

Diogo da Silva Almeida - (GU3047032)

Guarulhos - SP
2024

Especificações de Requisitos de Gilded Rose

Bem-vindo ao time Gilded Rose. Como você deve saber, nós somos uma pequena pousada estrategicamente localizada em uma prestigiosa cidade, atendida pelo amigável atendente Allison. Além de ser uma pousada, nós também compramos e vendemos as mercadorias de melhor qualidade. Infelizmente nossas mercadorias vão perdendo a qualidade conforme chegam próximo a sua data de venda.

Nós temos um sistema instalado que atualiza automaticamente os preços do nosso estoque. Esse sistema foi criado por um rapaz sem noção chamado Leeroy, que agora se dedica a novas aventuras. Seu trabalho será adicionar uma nova funcionalidade para o nosso sistema para que possamos vender uma nova categoria de itens.

Descrição preliminar

Vamos dar uma breve introdução do nosso sistema:

- Todos os itens (classe *Item*) possuem uma propriedade chamada *SellIn* que informa o número de dias que temos para vendê-lo
- Todos os itens possuem uma propriedade chamada *quality* que informa o quão valioso é o item.
- No final do dia, nosso sistema decrementa os valores das propriedades *SellIn* e *quality* de cada um dos itens do estoque através do método *updateQuality*.

Bastante simples, não é? Bem, agora que as coisas ficam interessantes:

- Quando a data de venda do item tiver passado, a qualidade (*quality*) do item diminui duas vezes mais rápido.
- A qualidade (*quality*) do item não pode ser negativa
- O "Queijo Brie envelhecido" (*Aged Brie*), aumenta sua qualidade (*quality*) em 1 unidade a medida que envelhece.
- A qualidade (*quality*) de um item não pode ser maior que 50.
- O item "Sulfuras" (*Sulfuras*), por ser um item lendário, não precisa ter uma data de venda (*SellIn*) e sua qualidade (*quality*) não precisa ser diminuída.
- O item "Entrada para os Bastidores" (*Backstage Passes*), assim como o "Queijo Brie envelhecido", aumenta sua qualidade (*quality*) a medida que o dia da venda (*SellIn*) se aproxima;
    - A qualidade (*quality*) aumenta em 2 unidades quando a data de venda (*SellIn*) é igual ou menor que 10.
    - A qualidade (*quality*) aumenta em 3 unidades quando a data de venda (*SellIn*) é igual ou menor que 5.
    - A qualidade (*quality*) do item vai direto à 0 quando a data de venda (*SellIn*) tiver passado.

Recentemente assinamos um suprimento de itens Conjurados Magicamente. Isto requer que nós atualizamos nosso sistema: Os itens "Conjurados" (*Conjured*) diminuem a qualidade (*quality*) duas vezes mais rápido que os outros itens. Sinta-se livre para fazer qualquer alteração no método *updateQuality* e adicionar código novo contanto que tudo continue funcionando perfeitamente.

Analisando e listando as Regras de Negócios

As regras fornecidas nos dizem como os diferentes tipos de itens devem se comportar em relação à qualidade (quality) e ao prazo de venda (sell_in). Aqui estão as principais regras novamente:

Geral para todos os itens:

- A qualidade (quality) diminui em 1 unidade por dia, a menos que o item seja um "Queijo Brie envelhecido" ou "Sulfuras".
- Quando o prazo de venda (sell_in) passa, a qualidade diminui duas vezes mais rápido, exceto para "Queijo Brie", "Sulfuras" e "Entrada para os Bastidores".

Itens específicos:

- "Queijo Brie" aumenta sua qualidade com o tempo.
- "Sulfuras" é um item lendário e sua qualidade não muda.
- "Entrada para os Bastidores" aumenta de qualidade com base em regras específicas.

Item "Conjurado":

- Itens "Conjurados" diminuem a qualidade duas vezes mais rápido do que outros itens normais.

Explicação do código da Refatoração

- **Sulfuras**: Como é um item lendário, não alteramos sua qualidade ou prazo de venda.
- **Aged Brie**: Aumenta sua qualidade em 1 unidade por dia e não pode exceder 50.
- **Backstage passes**: Aumenta a qualidade com base nas regras especificadas. Quando o prazo de venda passa, sua qualidade vai para zero.
- **Conjurados**: Diminuem a qualidade duas vezes mais rápido que outros itens normais.
- **Outros itens normais**: Diminuem a qualidade em 1 unidade por dia, exceto quando o prazo de venda passa, onde a diminuição é acelerada.

```
-----------------------------------------------------------------------
              CÓDIGO REFATORADO - GILDED ROSE EM PYTHON
-----------------------------------------------------------------------
```

```python
class GildedRose(object):
    def __init__(self, items):
        self.items = items

    def update_quality(self):
        for item in self.items:
            if item.name == "Sulfuras, Hand of Ragnaros":
                continue

            if item.name == "Aged Brie":
                item.quality = min(item.quality + 1, 50)
            elif item.name == "Backstage passes to a TAFKAL80ETC concert":
                if item.sell_in <= 0:
                    item.quality = 0
                elif item.sell_in <= 5:
                    item.quality = min(item.quality + 3, 50)
                elif item.sell_in <= 10:
                    item.quality = min(item.quality + 2, 50)
                else:
                    item.quality = min(item.quality + 1, 50)
            elif "Conjured" in item.name:
                item.quality = max(item.quality - 2, 0)
            else:
                item.quality = max(item.quality - 1, 0)

            if item.name != "Sulfuras, Hand of Ragnaros":
                item.sell_in -= 1
```

Relato de Refatoração do Código para Gilded Rose

Descrição do Processo de Refatoração: O processo de refatoração do código da classe *GildedRose* envolveu revisar e reestruturar o método *update_quality* para torná-lo mais legível, modular e aderente às novas especificações fornecidas. Aqui estão os passos principais e considerações durante o processo:

1. **Análise das Especificações**: O primeiro passo foi compreender completamente as novas regras de negócio para os itens da pousada Gilded Rose. Isso incluiu entender como cada tipo de item deve ser tratado em termos de qualidade e prazo de venda.
2. **Identificação de Problemas no Código Original**: O código original tinha várias condições aninhadas e repetições de lógica para diferentes tipos de itens. Isso tornava difícil entender e modificar o comportamento dos itens de forma consistente.
3. **Estruturação do Novo Código**: O objetivo foi criar uma estrutura mais modular e clara, onde cada tipo de item tivesse seu próprio conjunto de regras de atualização de qualidade e prazo de venda. Isso implicou em criar blocos distintos de código para cada tipo de item, como "Aged Brie", "Backstage passes", "Conjurados" e itens normais.
4. **Uso de Funções Auxiliares**: Para evitar repetições e melhorar a legibilidade, foram usadas funções auxiliares para calcular a qualidade de cada tipo de item com base nas especificações fornecidas. Isso ajudou a simplificar o código principal do método update_quality.
5. **Teste e Validação**: Após a refatoração, foram realizados testes para garantir que o comportamento dos itens estava de acordo com as novas regras. Isso incluiu verificar se a qualidade não excedia 50 para itens que têm essa restrição e se itens com prazo de venda passado se comportavam corretamente.

Dificuldades Encontradas: Durante o processo de refatoração, algumas dificuldades foram encontradas:

- **Compreensão das Regras Complexas**: As regras específicas para cada tipo de item e como elas interagiam umas com as outras exigiram um entendimento detalhado. Isso foi crucial para evitar erros na implementação das novas lógicas.
- **Integração com Código Existente**: O código original tinha sua própria estrutura e estilo. Integrar as novas funcionalidades sem quebrar o funcionamento existente foi um desafio, especialmente ao lidar com as exceções como "Sulfuras".
- **Garantia de Qualidade**: Assegurar que todas as novas funcionalidades estavam corretamente implementadas e que não havia regressões nos comportamentos dos itens foi essencial. Testes meticulosos foram fundamentais para validar as mudanças.

Lições Aprendidas:

- **Clareza e Legibilidade:** A importância de escrever código claro e legível ficou evidente durante a refatoração. Isso não só facilita a manutenção futura, mas também ajuda na compreensão imediata das regras de negócio implementadas.
- **Modularidade:** Dividir o código em partes mais pequenas e específicas para cada tipo de funcionalidade (neste caso, cada tipo de item) simplifica o desenvolvimento e a depuração. Isso também facilita a adição de novas funcionalidades ou ajustes no futuro.
- **Testes Rigorosos:** Investir tempo em testes robustos é crucial para garantir que as alterações não causem regressões ou comportamentos inesperados. Isso inclui testar não apenas casos comuns, mas também casos de borda e situações excepcionais.
- **Conhecimento do Domínio:** Compreender profundamente as regras de negócio é essencial para uma implementação precisa. Isso não apenas guia o desenvolvimento, mas também ajuda a explicar e justificar decisões de design durante o processo de refatoração.

Em resumo, a refatoração da classe *GildedRose* foi um exercício valioso que não apenas melhorou o código existente, mas também proporcionou aprendizados importantes em termos de boas práticas de programação e gestão de complexidade em sistemas existentes.