Relatório de Refatoração - Gilded Rose

Análise e Refatoração do Sistema Gilded Rose

Problemas Identificados no Código Original

O código original apresentava várias questões de arquitetura e manutenibilidade:

- **Violação do Princípio da Responsabilidade Única**: Toda a lógica de negócio estava concentrada em um único método
- **Alto acoplamento**: Diferentes tipos de itens compartilhavam a mesma lógica de atualização
- **Baixa coesão**: Regras específicas de cada item estavam misturadas
- **Dificuldade de extensão**: Adicionar novos tipos de itens exigia modificações no código principal
- **Complexidade ciclomática alta**: Múltiplos ifs aninhados tornavam o código difícil de analisar

Estratégia de Refatoração Implementada

Foi aplicado o **padrão Strategy** para separar as responsabilidades:

- **Interface** `ItemUpdater` : Define o contrato para atualização de itens
- **Classes concretas**: Cada tipo de item possui sua própria implementação
- **Factory Pattern**: `ItemUpdaterFactory` centraliza a criação dos atualizadores

Arquitetura Resultante

```
ItemUpdater (interface abstrata)
├── RegularItemUpdater
├── AgedBrieUpdater
├── BackstagePassUpdater
├── SulfurasUpdater
└── ConjuredItemUpdater
```

Principais Mudanças Implementadas

Separação de Responsabilidades: Cada tipo de item agora possui sua própria classe de atualização, isolando a lógica específica.

Redução da Complexidade: O método `update_quality` foi reduzido de 30+ linhas para 4 linhas, delegando a responsabilidade para os atualizadores específicos.

Extensibilidade: Novos tipos de itens podem ser adicionados criando uma nova classe que implementa `ItemUpdater` e adicionando uma condição na factory.

Funcionalidade dos Itens Conjurados: Implementação completa seguindo a especificação de degradação 2x mais rápida que itens normais.

Benefícios Técnicos Obtidos

- **Manutenibilidade**: Código mais legível e organizado
- **Testabilidade**: Cada comportamento pode ser testado isoladamente
- **Extensibilidade**: Fácil adição de novos tipos de itens
- **Reutilização**: Lógica comum centralizada no método `_clamp_quality`
- **Preservação do Comportamento**: Funcionalidade original mantida 100%

Validação da Refatoração

A refatoração foi validada através de testes abrangentes que cobrem todos os cenários de negócio, garantindo que o comportamento original foi preservado. O sistema funciona exatamente como antes, mas com uma arquitetura significativamente melhor.

Conclusão

A refatoração resultou em um código mais limpo, manutenível e extensível, seguindo princípios SOLID e padrões de design estabelecidos. A funcionalidade dos itens Conjurados foi implementada com sucesso, e o sistema está preparado para futuras extensões sem impacto no código existente.