ACH2003 — Computação Orientada a Objetos

Adryelli Reis dos Santos
(14714019)

Relatório de Refatoração e Modernização do Código "Gerador de Relatórios"

Objetivo

Este relatório descreve a refatoração e modernização do código do "Gerador de Relatórios" aplicando os padrões de projeto Strategy e Decorator e utilizando o arcabouço de coleções do Java. O objetivo foi melhorar a legibilidade, facilitar a manutenção e extensão do sistema, e adicionar novas funcionalidades conforme especificado no enunciado do exercício-programa.

Estrutura Original do Código

O código original era composto pelas classes `GeradorDeRelatorios`, `Produto`, e `ProdutoPadrao`, com a seguinte estrutura:

1. **GeradorDeRelatorios**: Principal classe do sistema, responsável por ordenar, filtrar e formatar um vetor de produtos, e gerar um relatório HTML.
2. **Produto**: Interface que define os métodos a serem implementados por qualquer classe de produto.
3. **ProdutoPadrao**: Implementação concreta da interface `Produto`.

Problemas Identificados

- **Baixa coesão e alta acoplamento**: A classe `GeradorDeRelatorios` tinha múltiplas responsabilidades, o que dificultava a manutenção e extensão do código.
- **Uso de vetores ao invés de coleções**: O código utilizava vetores de tipos primitivos, o que não seguia as convenções modernas do Java.
- **Código redundante e não orientado a objetos**: Algoritmos de ordenação e critérios de filtragem estavam misturados na mesma classe, com código repetitivo.
- **Falta de flexibilidade na formatação**: A formatação era aplicada uniformemente a todos os produtos, sem a possibilidade de personalização individual.

Refatoração e Modernização

Aplicação dos Padrões de Projeto

1. **Padrão Strategy**:
    - Criamos interfaces para os algoritmos de ordenação (`OrdenacaoStrategy`) e critérios de filtragem (`FiltroStrategy`).

    - Implementamos classes concretas para cada algoritmo de ordenação (`InsertionSortStrategy` e `QuickSortStrategy`) e critérios de filtragem (`FiltroTodos`, `FiltroEstoqueMenorOuIgual`, `FiltroCategoriaIgual`).
2. **Padrão Decorator**:
    - Criamos uma interface `ProdutoDecorator` que estende `Produto` e implementamos decoradores concretos para formatação (`NegritoDecorator`, `ItalicoDecorator`, `CorDecorator`).

Uso do Arcabouço de Coleções

- Substituímos o vetor de produtos por uma coleção (`List<Produto>`) para facilitar a manipulação e extensão dos dados.

Nova Estrutura do Código

- **GeradorDeRelatorios**:
    - A classe `GeradorDeRelatorios` foi simplificada para delegar as responsabilidades de ordenação, filtragem e formatação às estratégias e decoradores.
    - O código foi modularizado, movendo cada responsabilidade para classes separadas, seguindo os princípios de responsabilidade única.
- **Produto** e **ProdutoPadrao**:
    - A interface `Produto` e a classe `ProdutoPadrao` foram mantidas inalteradas conforme a restrição do enunciado.
- **Novas Funcionalidades**:
    - Implementamos novos critérios de ordenação (ordem decrescente), novos critérios de filtragem (preço dentro de um intervalo e descrição contendo uma substring), e novos decoradores de formatação (definir a cor do texto).
    - Adicionamos a funcionalidade de carregar produtos de um arquivo CSV.
- **Padrão de Nomenclatura:**
    - O código foi completamente revisado para seguir um padrão de nomenclatura consistente em inglês, eliminando o uso misto de inglês e português nas variáveis e nos comentários. Essa mudança visa aumentar a clareza e a coesão do código, seguindo as melhores práticas de desenvolvimento de software.

Conclusão

A refatoração e modernização do código do "Gerador de Relatórios" resultou em um sistema mais modular, flexível e fácil de manter. A aplicação dos padrões de projeto Strategy e Decorator, juntamente com o uso do arcabouço de coleções do Java, eliminou redundâncias, melhorou a legibilidade e facilitou a adição de novas funcionalidades. O código agora segue as melhores práticas de orientação a objetos e está preparado para futuras extensões e manutenções.