UNIVERSIDADE DE SÃO PAULO

Escola de Artes, Ciências e Humanidades

Relatório de pesquisa e desenvolvimento

ACH2003 – Computação Orientada a Objetos

Laís da Silva Moreira – 13838482

Marcos Paulo Tomás Ferreira – 13747950

Rafael Francisco de Freitas Timóteo – 12924740

Rafael Moura de Almeida – 11225505

São Paulo

2023

1. Introdução

Para este trabalho desenvolvemos a refatoração do código de um “Gerador de Relatórios”, essa é uma importante prática no desenvolvimento de software e consiste em melhorar a estrutura interna do código-fonte sem alterar seu comportamento externo. Em suma, trata-se da reorganização do código para torná-lo mais limpo e fácil de entender, facilitando assim a manutenção futura e a adição de novas funcionalidades.

No código original dado, podemos encontrar diversas falhas que podem facilmente ser corrigidas com a refatoração, a partir da aplicação dos padrões de projeto *Strategy* e *Decorator* como também da utilização convencional do conceito de coleções de elementos e de estruturação orientada a objetos. Ademais, além de consertar tais erros, também implementamos novas funcionalidades e critérios de ordenação e novas opções de formatação.

2. Padrões de Projeto utilizados

Como podemos observar no código dado, a classe *GeradorDeRelatorios* fica encarregada dos comportamentos de ordenação e filtragem, fato este que implica em uma inflexibilidade no que se trata de eventuais alterações ou adições em tais funcionalidades. Da necessidade de permitir a seleção e o uso de um algoritmo específico em tempo de execução sem depender de sua implementação concreta, utilizamos o padrão ***Strategy***, que possibilita a criação de uma família de algoritmos, encapsule cada um deles em uma classe separada e os torne intercambiáveis.

Para que fosse possível estender as funcionalidades de um objeto de maneira mais dinâmica e granular, evitando a criação de muitas subclasses para diferentes combinações de comportamentos, utilizamos o padrão de projeto ***Decorator***. Ele funciona como uma espécie de "invólucro" em torno do objeto original, adicionando comportamentos adicionais a ele sem alterar sua interface ou código existente. Assim sendo, depois da refatoração feita, a classe *GeradorDeRelatorios* deixa de ser responsável pelas opções de formatação de texto, sendo que agora a classe Produto passa a incorporar tal função.

3. O que foi feito

→ Utilização de tipos genéricos em conjunto com a interface List - que define uma coleção ordenada de elementos, permitindo o acesso aos elementos por meio de um índice - como também com a interface Map - que é responsável por mapear chaves únicas para elementos não ordenados

→ Criação das interfaces

IAlgoritmo - implementada pelas classes *InsertionSort* e *QuickSort*

ICriterio - implementada pelas classes *QuantidadeEstoque*, *Preco* e *Descricao*

IFiltragem - implementada pelas classes *Substring*, *Todos*, *Intervalo*, *CategoriaIgual* e *EstoqueMenorIgual*

IOrdenação - implementada pelas classes *Crescente* e *Decrescente*

IGeradorDeRelatorios - implementada *GeradorDeRelatórios*

→ Modificação e adaptação das implementações dos algoritmos de ordenação *quicksort* e *insertionsort*

→ Criação da classe *ProdutoCompleto*, que representa uma coleção de objetos *ProdutoPadrao* com seus objetos *Formatacao* correspondentes

4. Como utilizar o código na prática

→ As instruções para utilização do código estão presentes na seção *Main()* do código fornecido, caso a primeira condição seja cumprida, isto é, se o usuário fornecer uma quantidade de argumentos menor do que a requisitada