EP2 – Gerador de Relatórios

Prof. Flávio Luiz Coutinho

ACH2003 — Computação Orientada a Objetos

1º semestre de 2023

1 Objetivo

Este Exercício-Programa consiste em refatorar o código de um "Gerador de Relatórios" com o emprego de dois padrões de projeto: o padrão *Strategy* e o *Decorator* e modernizar o código para que ele use o arcabouço de coleções de Java.

O projeto do sistema é composto por uma interface (`Produto`) e duas classes (`ProdutoPadrao` e `GeradorDeRelatorios`). A classe `GeradorDeRelatorios` é a classe principal do sistema. Na versão fornecida com o enunciado, o papel dessa classe é processar um vetor (*array*) de `Produtos` e gerar uma listagem de produtos, seguindo certos critérios de **ordenação** e **filtragem**. A listagem gerada é gravada em um arquivo HTML que pode ser visualizado em qualquer navegador.

Além de definir o critério de ordenação e o critério de filtragem (para definir quais produtos devem fazer parte da listagem gerada), é possível escolher também qual o **algoritmo de ordenação** a ser utilizado. A classe `GeradorDeRelatorios` incorpora dois algoritmos de ordenação velhos conhecidos de vocês: o *Quicksort* e o *Insertion Sort* (pode parecer sem sentido oferecer a opção de escolha pelo *Insertion Sort*, mas lembrem-se de que tal algoritmo pode ter desempenho linear se o vetor a ser ordenado tiver poucos elementos fora do lugar). Uma quarta opção de personalização na geração dos relatórios permite definir a **formatação do texto** apresentado no arquivo HTML resultante.

De forma resumida, o "Gerador de Relatórios" em sua versão atual recebe um vetor de `Produtos` e:

1. os ordena usando o **algoritmo de ordenação** escolhido,
2. segundo um determinado **critério de ordenação**;
3. seleciona os produtos que satisfazem o **critério de filtragem**;
4. aplica **opções de formatação** para gerar o arquivo HTML resultante.

Há, portanto, quatro aspectos comportamentais do "Gerador de Relatórios" que podem ser personalizados. O conjunto completo de opções de configurações existentes, na versão atual do código, para cada um dos quatro aspectos listados acima, são:

1. **algoritmo de ordenação**: *Quicksort* e *Insertion Sort*.

2. **critério de ordenação**: ordem crescente pelo atributo **descrição** de um produto; ordem crescente pelo atributo **preço** de um produto; e ordem crescente pelo atributo **quantidade em estoque** de um produto.

3. **critério de filtragem**: todos (ou seja, todos os produtos entram na listagem gerada); produtos cujo estoque seja menor ou igual a uma certa quantidade; e produtos de uma determinada categoria.

4. **opções de formatação**: padrão (nenhuma opção aplicada); itálico; e negrito. As formatações são implementadas usando *tags* HTML que aplicam o efeito desejado a um texto.

Em relação aos três primeiros aspectos que podem ser configuráveis, cada opção é mutuamente exclusiva. Já em relação às opções de formatação, elas podem ser combinadas. É possível, por exemplo, definir que o texto de saída deve ser formatado em itálico e negrito ao mesmo tempo. Uma **limitação relevante** deste esquema de formatação, na atual versão do código, é que a formatação escolhida é aplicada a todos os produtos que entram na listagem resultante. Esse comportamento **deverá ser corrigido** com a aplicação de um dos padrões de projeto, de modo que as opções de formatação possam ser especificadas individualmente para cada instância de produto (desta forma, será possível destacar apenas um ou outro produto, e de várias formas distintas).

Apesar da diversidade de opções de personalização, devido ao fato de o código não estar bem estruturado e pouco orientado a objetos, ele apresenta diversos problemas: poderia ser mais legível e fácil de entender; o conceito de coleções de elementos não segue as convenções da linguagem Java; há código redundante; e eventuais extensões de funcionalidade demandam modificações em partes prontas do sistema e em diversos pontos do código. Com a refatoração e aplicação dos padrões, todos estes problemas devem ser resolvidos. Estender o sistema (acrescentar novos algoritmos de ordenação, novos critérios de ordenação e filtragem, e novas opções de formatação) ficará bem mais fácil e prático.

2 O que deve ser feito

O código deve ser modernizado para ficar mais orientado a objetos, usar as convenções da linguagem de programação Java e ficar mais fácil de ser lido e estendido.

Devem ser aplicados os padrões de projeto *Strategy* e *Decorator* [1]. O *Strategy* deve ser aplicado à classe `GeradorDeRelatorios` para melhorar a personalização desta em relação aos aspectos comportamentais 1, 2 e 3 listados. O aspecto comportamental 4, referente à formatação, deve deixar de ser responsabilidade da classe `GeradorDeRelatorios` e ser incorporado ao tipo `Produto` através da aplicação do padrão *Decorator*.

Além disso, o código deve ser modificado para que as classes e convenções do arcabouço de coleções sejam usados ao invés de vetores de tipos primitivos.

Após a adaptação para uso de coleções e das aplicações dos padrões, as seguintes **novas** funcionalidades devem ser implementadas:

- critérios de ordenação em ordem decrescente para os atributos descrição, preço e estoque de um produto (são 3 critérios no total);
- critério de filtragem para selecionar produtos com preço dentro de um intervalo determinado;
- critério de filtragem para selecionar produtos cuja descrição contenha uma determinada substring;
- decorador de formatação para definir a cor do texto referente a um produto;
- carregar produtos de um arquivo CSV (exemplo acompanha enunciado) e salvá-los em uma coleção de produtos.

Algumas restrições devem ser respeitadas. A primeira é que a interface `Produto` e a classe `ProdutoPadrao` não podem ser alteradas. A segunda é que os mesmos algoritmos de ordenação fornecidos no código orginal devem ser utilizados no novo código, adaptados para o uso com o arcabouço de coleções. Por fim, os decoradores implementados devem funcionar corretamente com qualquer instância de `Produto`.

Além da aplicação dos padrões e implementação das funcionalidades extras, também deve ser entregue um relatório sucinto que deve explicar resumidamente as mudanças que foram realizadas no código, bem como justificá-las.

3 Prazos e Entrega

Este EP pode ser feito em **grupos de 2 a 3 pessoas**. Este EP deve ser entregue na tarefa disponível no portal eDisciplinas **até o dia 24/07/2023** (último dia do semestre letivo "estendido"). Para a entrega, crie um diretório nomeado com o número USP de um dos componentes do grupo e coloque nele os arquivos fonte(s) de sistema refatorado. Adicione também o relatório (em formato `PDF`) no diretório e, em seguida, compacte o diretório no formato `.tar.xz`, `.tar.gz` ou `.zip`. Faça, em seguida, o envio deste arquivo. Deve ser feito apenas um envio por grupo.

Referências

[1] Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides. **Design Patterns**: *Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1995. ISBN: 0201633612.