# Relatório - EP2 COO

---

Ana Julia Silva de Oliveira - nro.usp 14557202

Rebecka Bocci Domingues - nro.usp 15486608

## 1. Algoritmo e Critérios de Ordenação

O projeto foi refatorado com o objetivo de torná-lo mais flexível quanto aos critérios de ordenação dos produtos. Duas dimensões configuráveis foram extraídas do código original: o **algoritmo de ordenação** (**como** os produtos são ordenados) e o **critério de ordenação** (**por qual atributo** os produtos são comparados e **em que ordem** eles devem ser dispostos). Para isso, foi utilizado o **padrão de projeto Strategy**, que permite definir uma família de algoritmos e torná-los intercambiáveis em tempo de execução.

### Estratégias implementadas

1. **Algoritmos de ordenação** (SortStrategy): Essas estratégias definem como a lista de produtos deve ser ordenada.

- **SortStrategy (interface)**: define o método sort(List<Produto>, Comparator<Produto>).
- **QuickSortStrategy**: implementação do algoritmo Quicksort.
- **InsertionSortStrategy**: implementação do algoritmo Insertion Sort.

2.1 **Critérios de ordenação** (Comparator<Produto>): Essas estratégias definem por qual atributo a ordenação será realizada.

- **DescricaoComparator**: compara os produtos com base na descrição.
- **PrecoComparator**: compara os produtos com base no preço.
- **QuantidadeComparator**: compara os produtos com base na quantidade em estoque.

2.2 **Ordem dos elementos após ordenação**: Para permitir ordenação em ordem crescente ou decrescente sem duplicar código ou criar múltiplas versões dos mesmos comparadores, foi introduzida uma nova estratégia: a direção da ordenação. Essa solução respeita o princípio de responsabilidade única (SRP) e torna o código

mais modular e extensível. A implementação também utiliza o padrão de projeto Strategy para encapsular o comportamento de direção.

- **OrdemStrategy (interface)**: define o método aplicar(Comparator<T>) que recebe um comparador base e devolve o comparador final com a ordem desejada (natural ou invertida).
- **OrdemCrescente:** implementação que retorna o comparador original, mantendo a ordenação crescente (padrão).
- **OrdemDecrescente:** implementação que retorna o comparador invertido usando o método Comparator.reversed(), aplicando ordenação decrescente.

Enquanto isso, a classe principal (GeradorDeRelatorios), foi modificada para aceitar estratégias de ordenação e critérios de comparação via injeção. O **método Ordenar** aplica a ordenação configurada antes de gerar o relatório. O comportamento da ordenação pode ser **trocado dinamicamente** usando os **métodos setSortStrategy** e **setComparator**.

As classes foram separadas em pacotes de acordo com sua função:

- **Estrategia_Ordenação/**: contém as estratégias de ordenação (QuickSortStrategy, InsertionSortStrategy, SortStrategy).
- **Criterio_Ordenação/**: contém os critérios de comparação (DescricaoComparator, PrecoComparator, QuantidadeComparator).
- **Direcao_Ordenacao/**: onde ficam as estratégias OrdemStrategy, OrdemCrescente e OrdemDecrescente

## 2. Critérios de Filtragem

A filtragem também foi refatorada usando o padrão de projeto **Strategy**, permitindo novamente que seja criado uma família de algoritmos intercambiáveis em tempo de execução. Para isso, foi adicionada uma nova dimensão: **Filtragem**, que abraça a interface **Filtros** e as classes estendidas por ele.

### Estratégias implementadas

1. **Filtragem** (Filtros): Essa estratégia define qual algoritmo de filtragem deve ser usado.

- **Filtros (interface)**: define o método selecionar(Produto p), que retorna um booleano se o produto deve ser selecionado ou não, com base na filtragem escolhida, definida pelas classes estendidas por ele.

- **CategoriaIgual**: implementação da filtragem na qual é passada uma categoria para o construtor da classe e o método selecionar retorna se a categoria do produto é a mesma da passada anteriormente.
- **DescricaoContem**: implementação da filtragem na qual é passada um valor de String para o construtor da classe e o método selecionar retorna se esse valor existe na descrição do produto.
- **EstoqueMenorOuIgual**: implementação da filtragem na qual é passada um valor de inteiro para o construtor da classe e o método selecionar retorna se o estoque do produto é menor ou igual a esse valor.
- **PrecoIntervalo**: implementação da filtragem na qual é passada um valor de String para o construtor da classe com os limites do intervalo representados como **"0.1,1.0"** (por exemplo), esses valores são separados e convertidos para double e, dessa forma, o método selecionar retorna se o valor está contido nesse intervalo.

Enquanto isso, a classe principal foi modificada para selecionar o critério de filtragem através de um **switch case** antes de executá-lo, sendo que cada produto é verificado iterativamente usando a coleção de Lista.

As classes foram separadas no pacote **Filtragem**.

## 3. Formatação

A formatação do relatório também foi um fator a ser considerado como mudança, dessa vez, implementamos a estratégia **Decorator**, permitindo que o objeto **Produto** possa ser decorado no relatório de forma que ele possa ser **negrito**, *itálico* ou colorido.

### Estratégias implementadas

1. **ProdutoDecorator**: Na pasta de **Produtos**, foi adicionada mais uma classe abstrata: a **ProdutoDecorator**. Nela, além de já ter as funcionalidades básicas de um produto, temos também funcionalidades adicionais para abordar a formatação, como cores, negrito e itálico.

2. **Formatadores**: Essa estratégia define qual formatador será utilizado. Todas as classes são filhas de **ProdutoDecorator**.

- **Cor**: define a cor na qual o produto será escrito no relatório.
- **Itálico**: define se o produto será escrito em itálico.
- **Negrito**: define se o produto será escrito em negrito.

Enquanto isso, a classe principal foi modificada para selecionar a formatação já na leitura do csv, passando o produto decorado diretamente para a lista de produtos. A decoração será feita ao programa imprimir o produto, onde ele verificará se ele contém alguma decoração e agirá de acordo.

As classes foram separadas no pacote **Formatadores**.