ATIVIDADE - PADRÕES FAÇADE E STRATEGY

Instruções:

- Esta atividade deve ser realizada considerando o mesmo grupo de Seminário de Sistemas Distribuídos;
- A atividade contará como a segunda nota da etapa 1 e valerá 100 pontos (50 pts para cada padrão);
- Deverá ser entregue até o dia 09/12/2020.

Atividades:

1. Pense em como estes dois padrões podem ser aplicados no seu projeto de Sistemas Distribuídos;
   a. Discorra sobre o problema que o padrão está resolvendo no seu projeto;
   b. Desenvolva o diagrama de classes representando a aplicação de cada padrão;
   c. Implemente na linguagem definida para o seu projeto, mesmo que de forma simplista, os dois padrões;
2. Envie o projeto para o Github e envie o link para esta atividade no Sala de Aula, juntamente com os arquivos dos diagramas em .pdf ou .png.

Observação:

Caso o padrão não se aplique no seu projeto: i) Explique com os fundamentos do padrão o porque do mesmo não se aplicar ao seu projeto; ii) Realize a atividade considerando um ou dois exemplos de projeto (implementado(s) em java); iii) Envie na atividade a descrição do(s) exemplo(s).

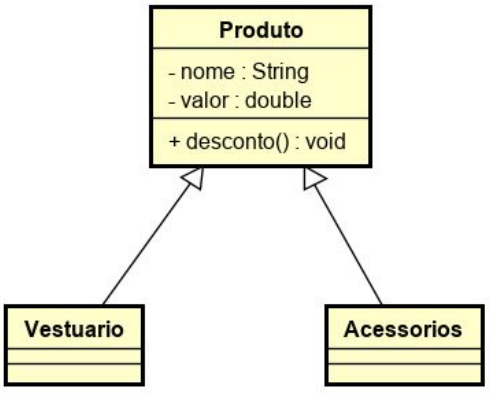
Padrão Strategy

1. **Por que não se aplica no projeto?**

Porque esse tipo de padrão deve ser utilizado quando existem muitas classes relacionadas que se diferenciam somente no seu comportamento, ou quando há a necessidade de variantes de um algoritmo, ou quando um algoritmo usa dados dos quais os clientes não deveriam ter conhecimento ou ainda quando uma classe define muitos comportamentos, e estes aparecem em suas operações como múltiplos comandos condicionais da linguagem e no nosso projeto não foi identificado nenhuma dessas situações.

2. **Que problema que o padrão está resolvendo neste projeto?**

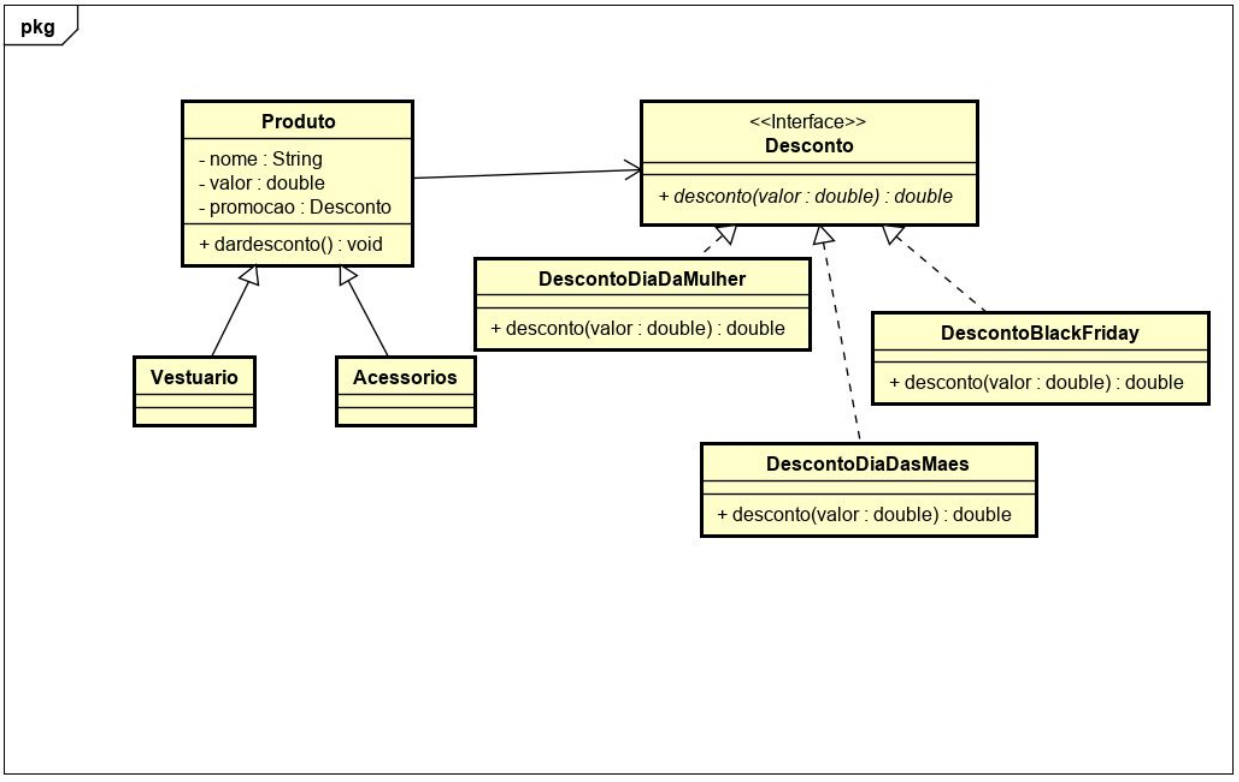
Existe uma loja virtual onde vende-se artigos femininos e em certos períodos do ano são oferecidos descontos com o objetivo de atrair clientes, abaixo é apresentado o diagrama de classe para exemplificar o problema (sem o padrão Strategy).



De acordo com o diagrama acima é possível fornecer desconto as mercadorias que estendem de Produto já que o método está na classe mais genérica e todas as suas subclasses herdam este comportamento. Neste exemplo, utiliza-se herança para expressar entidades que possuem características e comportamentos incomuns preservando suas particularidades. Ao mudar o presente contexto, onde o administrador da loja resolve dar descontos a determinados produtos e em datas comemorativas, provindo do mesmo objetivo que é atrair o consumidor, contendo uma observação em que alguns produtos não receberão descontos pela grande saída na loja.

Conforme a estrutura atual, ao incluirmos o desconto na classe mais genérica, será atribuído desconto a todos os produtos e esse não é o objetivo, uma das formas que poderá ser realizada é sobrescrever o método desconto() e atribuir o desconto necessário em cada subclasse, desta forma irá funcionar, mas assim teremos um cenário muito inflexível e assim difícil de ser mantido, já que a finalidade é atribuir os descontos com uma certa periodicidade deve-se criar uma estrutura favorável.

Tendo em vista que o método desconto() tem possibilidade de mudança ao longo do tempo, será utilizado alguns princípios de projeto, um deles seria dar preferência ao uso de composição ao invés de herança, onde dessa forma dará maior flexibilidade ao objeto tendo a possibilidade de mudar o seu comportamento em tempo de execução, outro princípio programar para interfaces e não para classes concretas tornando as classes independentes umas das outras, e tornando ainda um designer de projeto consideravelmente bom para ser fechado para alterações e aberto para extensões, abaixo é apresentado o diagrama de classe para exemplificar a solução proposta para o problema (com o padrão Strategy).



Padrão Façade

1. **Discorra sobre o problema que o padrão está resolvendo no seu projeto**

O padrão FACADE tem como objetivo disponibilizar uma interface unificada de nível mais alto e simplificada para os recursos e facilidades mais gerais de um subsistema.

Devendo ser utilizada quando se deseja fornecer uma interface simples para um subsistema complexo, ou quando existirem muitas dependências entre clientes e classes de implementação de uma abstração, ou ainda quando desejar estruturar seus subsistemas em camadas.

Tendo em vista que para o desenvolvimento do projeto de sistema distribuído o mesmo foi estruturado em camadas e que é desejável fornecer uma interface mais simples para o sistema, foi desenvolvida uma classe que funcionará como uma fachada, onde a mesma é responsável por gerenciar as chamadas ao banco de dados, e ainda é responsável por chamar os métodos que possuem as regras de negócios para cálculos que serão utilizados para a exibição de dados no front-end. Neste sentido, a classe funcionará como uma interface de acesso aos dados e lógicas executadas em cima dos mesmos.

.

2. **Diagrama**
   OBS: Diagrama em formato de imagem

3. **Link para o diretório do projeto**
   https://github.com/Gileno29/COVID-SYS