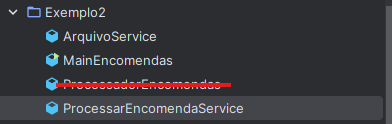
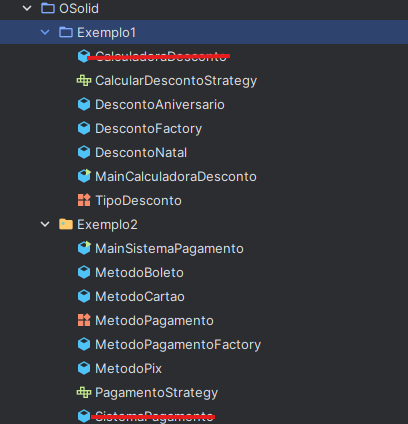
# Single Responsability Principle

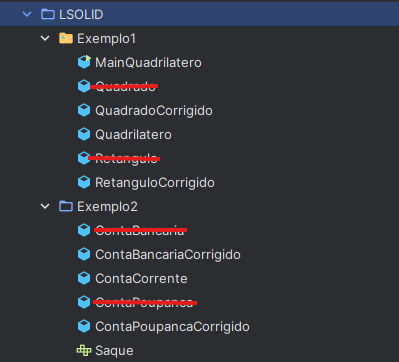
Exemplo 2:  
Foi realizada a separação das responsabilidades dividindo cada uma em uma nova classe:  
  
  
  
Open-Closed Principle

Aplicação do Design Pattern Strategy para conseguirmos criar novos comportamentos sem precisar alterar as já existentes.  
Aplicação de ENUMs e de uma classe Factory para servirem como gerenciadores de comportamentos, onde serão os únicos que precisarão de atualizações em caso de inserção ou remoção de um comportamento que implementa o nosso Strategy.



Liskov Substitution Principle

Aqui para não quebrarmos o princípio da Substituição de Liskov, criamos uma super classe em cada um dos exemplos onde contém apenas campos e métodos que realmente é utilizado por todos os seus filhos, dessa forma, podemos substituir um objeto da classe pai por qualquer outro objeto das classes filhas sem nenhum problema.



# Interface Segregation Principle

Nesse princípio, o que fizemos foi a separação de funcionalidades no qual nem todos que implementam a interface irão utilizar, em outras interfaces. Dessa forma, cada classe que for utilizar uma funcionalidade, ela irá implementar as interfaces que deseja, assim podendo implementar apenas métodos que irão ser utilizados.  
