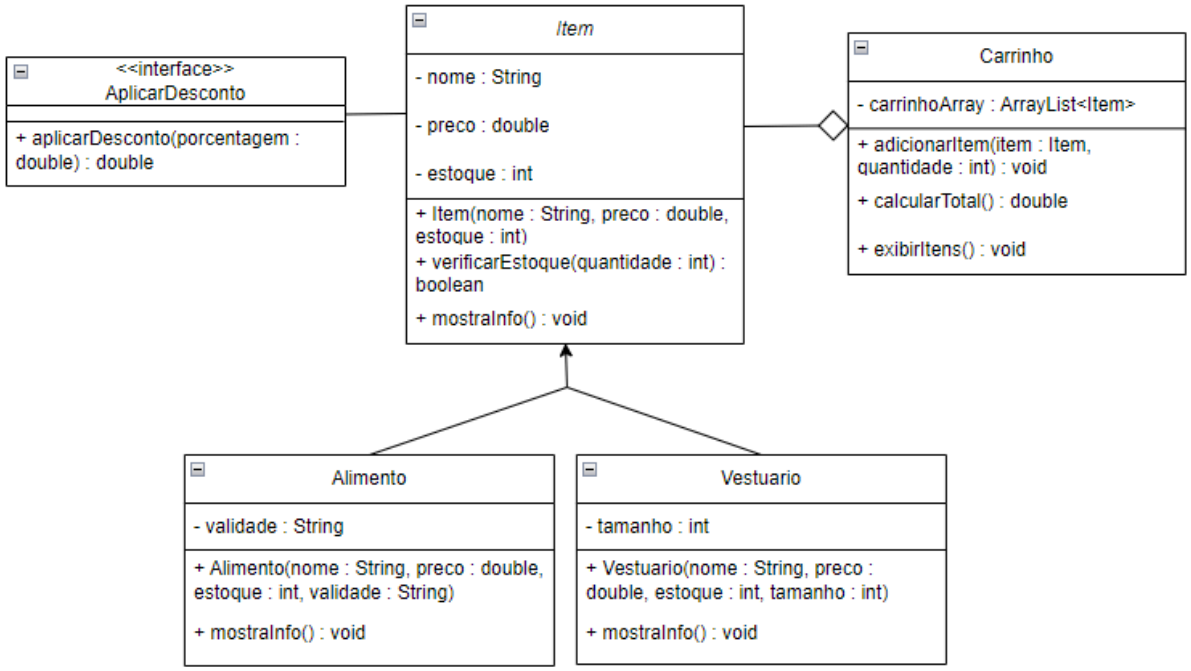
Implemente o sistema a seguir de um carrinho de compra de itens



NOTAS IMPORTANTES:

- Se necessário, crie apenas os getters e setters que for utilizar. A criação de um getter ou setter desnecessário irá acarretar na perda de pontos;
- Siga a UML, não segui-la irá acarretar na perda de pontos.

Classes e interfaces:

- **Item:**
  - verificarEstoque() deve receber uma quantidade de itens e verificar se o estoque disponível para aquele item é suficiente e retornar true ou false com base no resultado
- **AplicarDesconto:**
  - AplicarDesconto() deve receber um double de porcentagem (0<porcentagem<1) e atualizar o preço do item que chamou o método, independente do tipo do item
- **Alimento e Vestuário:**
  - Não devem sobrescrever o método verificarEstoque()
- **Carrinho**:
  - adicionarItem() deve lançar uma Exception **EstoqueInsuficienteException** com a mensagem que você achar melhor, caso, ao verificar o estoque do item, o retorno seja falso. Do contrário, deve adicionar o item ao array
  - calcularTotal() deve lançar uma Exception **CarrinhoVazioException** com a mensagem que você achar melhor, caso o carrinho esteja vazio. Do contrário deve-se somar o preço de cada item no carrinho e retornar o total (ignore o parâmetro “quantidade” que foi passado ao verificar o estoque, considere que há apenas 1 no carrinho)

    - exibirItens() deve verificar se o carrinho está vazio e caso não, mostrar as informações de cada item no carrinho.

Na main, crie todos os objetos necessários e em seguida crie 4 try-catches com os seguintes cenários:

- Adicionar itens no carrinho com sucesso(pelo menos um item com desconto)
- Adicionar itens no carrinho com erro (forçar a exception)
- Calcular o total do carrinho de sucesso
- Calcular o total do carrinho com erro (forçar a exception)

Por fim, chame todos os métodos do carrinho que ainda não foram chamados