Grupo:

Lucas Gonçalves da Conceição Nogueira

Daniel Rocha Lopes

Hudson Vieira

Requisito 1: NF não pode ter zero IV. Deve ter 1 ou mais

Este requisito é verificado na construção da nota fiscal, onde verifica-se se se a lista de IV está vazia.

(teste: ThrowsExceptionWhenConstructNFWith0IV)

Requisito 2: *Todo IV deve pertencer a exatamente uma NF*:

Os IVs são criados e Clonados para cada nova NF criada . Assim, não há como existirem IVs em mais de uma NF ou órfãos. A NF também cria (clonando) seus próprios IV (composição).

Requisito 3: *Todo IV se referirá a exatamente um produto ou serviço*

No momento da criação do IV , o mesmo só faz referência a um Produto/Serviço. Não é possível cria IV sem “pai”

(Teste: ThrowsExceptionWhenPSDoesNotExistInDB)

Requisito 4: Um P/S deve sempre pertencer a um IV ou a um outro P/S.No projeto um P/S somente é criado a partir da criação de IV.

Requisito 5: O construtor de NF verifica se PS estão no banco de dados. Apenas BD cria P/S. O BD:P/S contém informação sobre produtos e serviços inclusive a categoria, sendo essas informações retornadas por métodos. BD facade também retorna as mesmas informações .

Requisito 06: Nf mutável possui uma variável estado que denota em elaboração”. A nota fiscal é validada pelo banco de dados, caso validada, o mesmo retorna a NF imutável que contém o novo estado “validada”. A ID gerada é única e a nota imutável não possui métodos de acesso, nem mesmo seus IV são alteráveis.

Requisito 07: As ID`s geradas pelo BD são únicas e geradas exclusivamente caso o BD não possua a atual NF validada e caso a mesma possua todos os requisitos de validação.

Requisito 08: Existe um banco de dados que possuem todos os impostos e cálculos referentes ao mesmo. O BD permite acesso aos impostos para cálculo dos tributos de uma NF.

Requisito 09: Foi utilizado o DP Strategy para estruturar os impostos, de maneira que seja fácil a inclusão de novos impostos. O DP visitor foi utilizado para calculo dos impostos.

Requisito 10: Para que se possa manter o estado durante o calculo do imposto, o banco de dados tem acesso a uma cópia da lista IV do NF.

Requisito 12: Foi criada uma interface de PS de modo a tornar mais fácil a inserção de novos PS, que seriam subclasses da interface existente, não necessitando assim de adequação do algoritmo.

Requisito 13: Foi utilizado protected em getlista de modo que não haja escrita nem alterações inapropriadas. Entretanto, NF imutável possui métodos apropriados para adicionar deletar ou alterar IV.

Requisito 14: Foi criado um BD facade para cumprir este requisito. Além do uso de singleton de modo a permitir apenas uma instanciação.

Requisito 15: O DP strategy aliado ao visitor permitiu que de maneira genérica, mesmo com alteração dos cálculos e inserção de novos impostos fosse possível gerar a nota fiscal sem alterar o código.

Requisito 16: Os objetos pertencentes ao BD foram modelados de modo a possuírem atributos inacessíveis para mudança, apenas para consulta. (Imutable Object)

Requisito 17: O produtoserviço segue o DP composite. Existem 2 contrutores para PS, um que aceita uma lista de PS e um com apenas o nome e o preco. Com isso, se agrega novos produtos servicos aos PS exitstentes. Para o calcculo do preco, se soma o preço de todos os nós, percorrendo a árvore por DFS (vide Teste getPrecofromPSWithVariousLevels())

Requisito 18: Para o cálculo dos impostos é recebida uma lista IV a qual permite acesso as hierarquias de P/S dentro do BD. Assim, permite-se que mesmo com mudança de critérios o código não seja prejudicado.

Requisito 19: Para cálculo dos impostos, há consulta de histórico através do BD.

Requisito 20 : A nota fiscal usa o DP data Object, sendo q a NF principal contem um NF\_Data, que a torna facilmente extensível.

Requisito 21: método toPrint() da NF