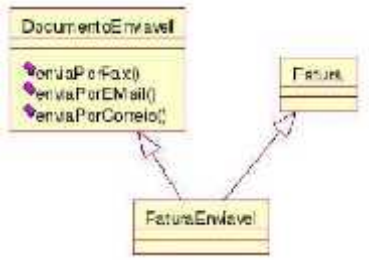
Resposta dos exercícios de Arquitetura e Projeto de Sistemas II

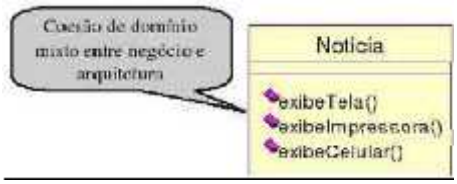
Aula 07 – Princípios de Projetos OO – Parte I

Slide 19 – A classe FaturaEnviavel tem, sem dúvida, coesão de domínio misto. Este fato representa algum problema para o sistema? Tente fornecer argumentos sobre o seu ponto de vista.

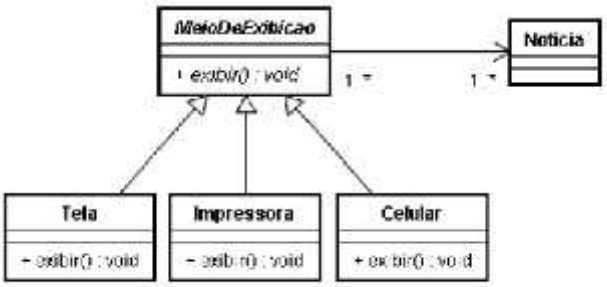


Resposta: D

Slide 30 – Como é possível permitir que a classe `Noticia` (Figura abaixo) possa ser exibida na tela, na impressora ou no celular sem criar coesão de domínio misto e que facilite a criação de novos métodos de exibição? Tente encontrar uma solução e argumentar as vantagens e desvantagens da sua solução.



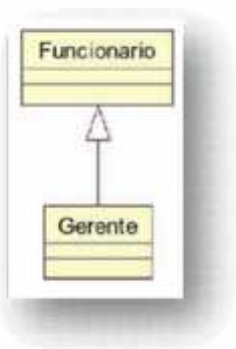
Resposta: É possível permitir que a classe `Noticia` possa ser exibida na tela, na impressora ou no celular sem criar coesão de domínio misto e de forma a facilitar a criação de novos métodos de exibição. Uma possível solução seria manter a classe `Noticia` no domínio de negócio, retirando dela métodos que se refiram ao domínio de arquitetura e criando um relacionamento de associação com uma nova classe abstrata `MeioDeExibicao`, a qual possuirá o método `exibir()` responsável por exibir uma notícia em algum meio de exibição disponível no sistema em questão. Esses meios seriam providos por meio de classes concretas do domínio de arquitetura, que possuem um relacionamento de herança com a classe `MeioDeExibicao` (do domínio de arquitetura) e que possuem implementações para o método `exibir()`: `Tela`, `Impressora` e `Celular`. Dessa forma, uma das vantagens é a possibilidade de exibição de notícias em diversos meios, bastando a implementação de uma nova classe que implemente a classe `MeioDeExibicao`, sem a necessidade de ser invasivo ao domínio de negócio, o que facilita a realização da evolução do sistema e a manutenção de sua organização estrutural. Uma possível desvantagem está no relacionamento direto entre as classes `Noticia` e `MeioDeExibicao`, o qual amarra uma classe de domínio de negócio a uma de domínio de arquitetura, sem a utilização de uma classe do domínio de sistema, por exemplo, que faça o relacionamento ("classe cola") e não impacte o sistema, caso esse relacionamento se torne desnecessário o futuro.



Solução elaborada para o problema de coesão de domínio misto

Aula 08 – Princípios de Projetos OO – Parte II

Slide 11 - Qual atitude pode ser tomada na estrutura abaixo caso não seja possível projetar as pré-condições dos métodos de Gerente com restrições iguais ou menores que as restrições das pré-condições dos métodos de Funcionário?



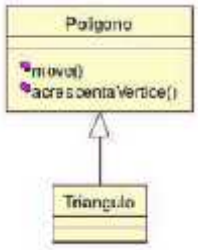
Resposta: Dado que não seja possível projetar as pré-condições dos métodos de *Gerente* com restrições iguais ou menores que as restrições das pré-condições dos métodos de *Funcionário,* pode-se pensar que essa é uma herança questionável. Dessa forma, uma das atitudes que podem ser tomadas se refere à criação de uma nova classe (e.g., *Pessoa*), que reúna as características (atributos) e comportamentos (métodos) básicos e comuns ao pessoal empregado na organização na qual o sistema é utilizado, de forma que essa nova classe seja herdada pelas classes *Gerente* e *Funcionário*. Assim, cada subclasse definiria e implementaria os métodos que, devido aos problemas de contrato, inviabilizaram a herança em questão.

Por outro lado, observando-se as entidades envolvidas na estrutura acima, caso os métodos de *Gerente* necessitem de restrições maiores quanto às suas pré-condições, dada a herança a partir de *Funcionário*, pode ser preciso reavaliar o projeto da classe *Funcionário*: fazendo um paralelo com o domínio de aplicação que abriga essas entidades, um "gerente" tem mais privilégios do que um "funcionário" comum, inclusive os destes. Considere um método *int calculaBonus (int avaliacao),* em que um funcionário pode ser avaliado numa escala de 1 a 10 (espaço-estado da variável *avaliação*), ao passo que um gerente poderia ser avaliado numa escala de -5 a 15. Caso a escala de avaliação para um gerente fosse reduzida para entre 3 e 9, provavelmente essa redução teria que ser considerada para um funcionário, visando manter a coerência e validade do domínio com relação à sua modelagem da aplicação; caso contrário, incorrer-se-ia em problemas com contratos. Dessa forma, uma atitude a ser tomada corresponderia à verificação da classe *Funcionário* e de seus impactos na estrutura do sistema, a fim de realizar uma manutenção evolutiva e avaliar a existência de herança entre *Gerente* e *Funcionário*, caso Gerente cumpra as cláusulas contratuais.

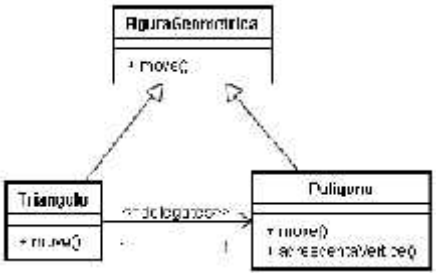
Slide 14 - De acordo com os princípios de Projeto OO, você considera aceitável a criação de pré e pós-condições em métodos abstratos de uma classe? E quanto aos métodos de uma interface?

Resposta: Sim, uma vez que a criação de pré e pós-condições em métodos abstratos de uma classe representa a sua documentação e qualquer classe que implemente esse método também deve respeitar a essa documentação, mantendo-se uma padronização na especificação das variações de espaço-estado possíveis para o método. Da mesma forma, essa padronização é entendida como aceitável para métodos de uma interface, garantindo, dessa forma, a metodologia do projeto por contrato.

Slide 17 - Qual modificação na estrutura abaixo poderia ser feita para possibilitar a manutenção do comportamento fechado global?



Resposta: Existem alguns possíveis caminhos visando possibilitar a manutenção do comportamento fechado global: (i) fazer polimorfismo sobre o método `acrescentaVertice()`, com lançamento de exceção, o que representa uma solução ruim; (ii) evitar a herança de `acrescentaVertice()`, alterando a estrutura hierárquica; e (iii) preparar o projeto para possível reclassificação do objeto da classe `Triangulo` para outra classe (e.g., `Quadrilatero`, `Pentagono` etc.). Por exemplo, a solução exibida a seguir é relativa ao caminho (ii).



A solução exibida abaixo corresponde ao caminho (iii). Um objeto da classe `FiguraGeometrica` está associado a um objeto de alguma subclasse de *`Poligono`*. A depender da execução do método `acrescentaVertice()`de `FiguraGeometrica`, o objeto associado ao objeto destas pode ter suas propriedades alteradas, devendo-se instanciar um outro objeto de uma subclasse mais adequada de *`Poligono`*. Ou seja, deve-se permitir que um objeto modifique o seu comportamento em função do seu estado interno, gerando o efeito de uma troca de tipo de objeto em tempo de execução. Isso equivale a criar um novo polígono, mantendo as características anteriores (antigos vértices) e adicionando uma nova característica (novo vértice).