Qual princípio de projeto é violado por uma chamada como a mostrada abaixo? Qual mudança de projeto você faria na classe Biblioteca (tipo de bib) para remover essa violação?

bib.getAcervo().getAreaConhecimento(“ES”).

getLivros().find(“ESM”).getNumExemplares();

RESPOSTA: Está com uma alto acoplamento, precisa chamar vários métodos de diversas classes para realizar uma operação. Isso dificulta o entendimento do código e leitura.

Suponha a seguinte classe:

class Tabela {

...

void imprime() {

// código que imprime cabeçalho

// código que imprime cada linha da tabela

// código que imprime rodapé

}

...

}

Esse código não segue o Princípio Aberto/Fechado, pois no nosso sistema é importante poder variar as mensagens de cabeçalho e rodapé. Como você mudaria a implementação dessa classe para atender ao Princípio Aberto/Fechado?

RESPOSTA:

Seria bem melhor se se aplicasse uma interface com os métodos para cabeçalho, tabela e rodapé. Aí em suas respectivas classes se implementam os métodos. Assim estaria aberto para extensão e fechado para modificação.

Suponha uma classe Calculadora com um método que verifica se um número entre 0 e 10000 é primo. Suponha ainda que um algoritmo mais eficiente para essa tarefa foi implementado em uma subclasse CalculadoraRapida. No entanto, ele somente é capaz de trabalhar com números entre 1000 e 9000.

| **class** Calculadora {  boolean isPrimo(n) {  // 0 <= n < 10000  }  } | **class** CalculadoraRapida **extends** Calculadora {  boolean isPrimo(n) {  // 1000 <= n < 9000  }  } |
| --- | --- |

Qual princípio de projeto é violado nessa implementação? Justifique.

RESPOSTAS: O princípio de substituição de Liskov e se forçamos um pouco o princípio da responsabilidade única. Toda subclasse deveria ser substituível pela sua classe mãe, o que não acontece nesse caso, onde o “poder” da subclasse é reduzido.

Suponha que você concluiu um curso de extensão oferecido pelo IFBA e quer receber seu certificado. Para isso, você teve que:

* Mandar uma msg para o coordenador do curso, que te pediu para mandar uma msg para secretaria do DCC.
* Então, você mandou uma msg para a secretaria do DCC, que te pediu para mandar uma mensagem para o CENEX.
* Então, você mandou uma msg para o CENEX, que te pediu para mandar uma msg para a PROEX.
* Então, você mandou uma msg para a PROEX, que finalmente retornou o seu certificado.

Essa sequência de passos ilustra uma violação de qual princípio de projeto? Além de ter que mandar várias mensagens, qual um outro problema dessa “solução”?

RESPOSTA: Semelhante à primeira pergunta, está com uma alto acoplamento, precisa chamar vários métodos de diversas classes para realizar uma operação. Isso dificulta o entendimento do código e leitura. Também ao princípio de responsabilidade única, cada classe deveria ser usada apenas com uma intenção, com uma responsabilidade.

Suponha que você está implementando um sistema S e precisa usar um serviço de processamento de pagamentos X para, por exemplo, aceitar pagamentos via cartões de crédito. No entanto, você não quer acoplar S a X, pois futuramente você pode optar por um outro serviço Y.

1. Explique como você faria para desacoplar S de X.

RESPOSTA: Não entendi muito bem o contexto, mas poderia fazer um método do tipo pagamento, algo assim, e fazer polimorfismo de sobrecarga, ter métodos de nomes igual que recebem parâmetros diferentes com implementações que são de acordo com o tipo de parâmetro recebido.

1. Qual princípio de projeto você usou para garantir esse desacoplamento?

RESPOSTA: Princípio de inversão de dependência, módulos de alto nível não devem depender de módulos de baixo nível. Ambos devem depender de abstrações. As abstrações não devem depender de detalhes. Detalhes devem depender de abstrações.

Em Engenharia de Software, às vezes usamos soluções mais complexas em contextos nos quais isso não é necessário. Esse problema é chamado de over engineering ou otimização prematura.

Então crie e descreva um contexto no qual o uso de um dos princípios de projeto que estudamos é uma otimização prematura.

Se quiser, você pode criar e explicar esse contexto usando o mesmo exemplo que usamos na explicação do princípio.

RESPOSTA: Estou desenvolvendo um sistema de reservas para um restaurante, que vai permitir fazer reservas, cancelar e consultar a disponibilidade. Em vez de complicar o projeto criando várias interfaces separadas para cada funcionalidade desde o início, é melhor manter tudo integrado.Evita complexidade desnecessária, economiza tempo e me permite focar no feedback dos usuários. Em resumo, estou aprendendo que menos é mais no estágio inicial do projeto.