Princípios em Projeto de Software

Atividades de Aprendizagem e Avaliação

Aluno: João Pedro Rodrigues Leite RA: A2487055

Use esta cor em seu texto

1. Considerando o conteúdo do texto do link “Princípios em Projeto de Software”
   a) Comente a afirmação de John Ousterhout.

   R: Quando falamos de projeto isso indica que estamos buscando a solução para um problema. Esse problema consiste na implementação de um sistema que atenda aos requisitos funcionais e não-funcionais. Para isso, devemos decompor, isto é, quebrar o problema que pode ser bastante complexo e partes menores. Além disso, há uma restrição na afirmação de Oustehout, a decomposição deve permitir que cada uma das partes do projeto possa ser resolvida (ou implementada) de forma independente. Uma das estratégias importantes para combater a complexidade de sistemas de software passa pela criação de abstrações que é uma representação simplificada de uma entidade.

   b) Comente o conceito de Integridade Conceitual.

   R: É uma propriedade de projeto proposta por Frederick Brooks. Um sistema não pode ser um amontoado de funciondalidades, ele deve ter coesão. Integridade conceitual é importante pois facilida o uso e entendimento de um sistema por parte de seus usuários. Um contra-exemplo de integridade conceitual pode ser um sistema que exiba informações em formato de tabela, mas em algumas tabelas é possível ordenar as informações clicando no título, outras não está disponível, algumas tabelas monstram informações em negrito ou na moeda real, e outras mostram em dólares. Essa falta de padrão é a ausência de integridade conceitual. O principio também se aplica ao design e código de um sistema. Por exemplo parte do sistema está utilizando o padrão de nomes camel case, como em notaTotal e outra parte snake case, como em nota_total. Ou frameworks diferentes partes do sistema onde seria necessário apenas um deles. Toda essa ausência de padronização caracteriza a falta de integridade conceitual.

   c) Cite e comente a(s) estratégia(s) para tornar sistemas de software mais flexíveis e fáceis de entender.

   R: O ocultadmento de informação pode treazer beneficios como, desenvolvimento em paralelo, flexibilidade a mudanças e facilidade no entendimento. No entanto para que issso ocorrra, as classes devem esconder decisões de projeto que são sujeitas a mudanças. Para isso os atributos e métodos que uma classe pretende encapsular são declarados com o modificador de

visibilidade privado. Mas alguns dos métodos devem ser públicos pois senão a classe não seria útli, para isso utilizamos a interface. Interfaces devem ser estáveis, pois mudanças na interface de uma classe podem demandar atualizações nas classes que as implementa,

d) Comente a relação entre Coesão e Separação de Interesses.

R: Coesão nos dá as seguintes vantagens, facilita a implementação de uma classe, bem como o seu entendimento e manutenção, facilita a alocação de um único responsável por manter uma classe, facilita o reúso e teste de uma classe, pois é mais simples de reusar e testar uma classe coesa do que uma classe com várias responsabilidades. Já a separação de interesses defende que uma classe deve implementar apenas um interesse(concern), o termo interesse se refere a qualquer funcionalidade, requisito ou responsabilidade da classe. Com isso podemos ver que essas duas propriedades possuem uma relação, pois suas recomendações são equivalentes. (1) uma classe deve ter uma única responsabilidade; (2) uma classe deve implementar um único interesse; (3) uma classe deve ser coesa.

e) Conceitue Acoplamento no contexto do código fonte de um software.

R: Acoplamento é uma forte conexão entre duas classes. Há dois tipos de acoplamento, o aceitável, quanto por exemplo a classe A utiliza apenas métodos públicos da classe B; A interface provida por B é estável do ponto de vista sintático e semântico. Por outro lado temos o acoplamento ruim, onde por exemplo quando a classe A realiza um acesso direto a um arquivo ou banco de dados da classe B; Quanto a classe A e B compartilham uma variável ou estrutura de dados global. Por exempo, a classe B altera o valor de uma variável global que a classe A usa; Quando a interface da classe B não é estavel. Os métodos de B são renomeados com frequência. Ou seja, o acomplamento pode ser muito útil, mas o acomplamento ruim deve ser evitado, para isso deve haver uma interface estável mediadora.

f) Considere a tabela dada no início da seção 5.6 relacionando os **Princípios SOLID** com as **Propriedades de Projeto.** Estude o conteúdo que vem em sequência. Reproduza a tabela aqui, adicionando uma breve explicação (síntese) de cada relacionamento. (máximo de 5 linhas por item)

Responsibilidade única	Coesão	O princípío de responsabilidade única garante que uma classe tenha uma única razão para mudar, aumentando a coesão, pois todas as funções da classe estão relacionadas a uma única tarefa.
Segregação de Interfaces	Coesão	A segregação de interfaces evita a implementação de métodos desnecessários, o que melhora a coesão ao manter interfaces específicas e alinhadas ao que realmente precisa ser usado.

Inversão de Dependências	Acoplamento	Inversão de dependências diminui o acomplamento, promove a utilização de abstrações em vez de dependências diretas, tornando o sistema mais flexível a mudanças.
Prefira Composição a Herança	Acoplamento	Composição, ao invés de herança, reduz o acomplamento entre classes, pois as dependências são passadas como componentes, facilitando a modificação e substituição de partes do sistema.
Demeter	Ocultamento de Informação	A lei de Demeter promove o ocultamento de informações ao limitar o acesso a objetos distantes, garantindo que as classes conheçam apenas seus colaboradores diretos.
Aberto/Fechado	Extensibilidade	O princípio Aberto/Fechado incentiva a criação de sitemas que são abertos para extensão, mas fechados para modificação, facilitando a adição de novas funcionalidades sem alterar o código existente.
Substituição de Liskov	Extensibilidade	A substituição de Liskov garante que subclasses possam substituir suas classes base sem quebrar o comportamento do sistema, favorecendo a extensibilidade de forma segura.

Dica: Enriqueça seu aprendizado com os exercícios de fixação que estão ao final do capítulo.