**Princípios em Projeto de Software**

**Atividades de Aprendizagem e Avaliação**

**Aluno: André Luis Quiosi RA: 2369958**

**Use esta cor em seu texto**

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

R: A afirmação de John Ousterhout se refere a quebra de problemas maiores em problemas menores, fazendo com que seja mais fácil trabalhar. Tal método deve ser implementado de forma que todas as partes do processo sejam independentes e possam ser implementadas separadamente, também cada parte deve abstrair as complexidades para as outras.

* 1. Comente o conceito de Integridade Conceitual.

R: Seguindo o que Brooks escreveu, pode-se ter a conclusão que um sistema com uma boa Integridade Conceitual fique mais fácil sua manutenção, também é visto que usando esse conceito temos a diminuição da possibilidade de termos problemas parecidas que sejam resolvidos de formas diferentes.

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

R: Para tornar um sistema mais fácil de se desenvolver e de entender, as classes devem abstrair complexidades e características que podem estar sujeitas a mudanças, para assim poderem ser alteradas e melhoradas sem que afetem as outras partes do sistema que consomem as informações tratadas dentro da classe. Para isso, também é necessário que as interfaces sejam o mais estável possível, diminuindo assim a necessidade de manutenção nos clientes dessa interface.

* 1. Comente a relação entre Coesão e Separação de Interesses.

R: A relação entre eles é que a busca por coesão dentro de uma classe contribui para atingir a separação de interesses em nível micro. Quando as classes são coesas, naturalmente estão implementando uma única responsabilidade, alinhando-se ao princípio da separação de interesses. Isso resulta em um código mais organizado e modular.

* 1. Conceitue Acoplamento no contexto do código fonte de um software.

R: Acoplado no contexto do código fonte de um software se refere à dependência entre diferentes partes do código. Um acoplamento forte traz dependência, onde mudanças em uma parte podem afetar outras. Um acoplamento fraco é mais adequado, onde as interações ocorrem por meio de interfaces bem definidas e estáveis. O objetivo é manter um baixo acoplamento para facilitar a manutenção e evolução do software.

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