	INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ **Curso: ADS** **Disciplina: Engenharia de Software III** **Professor: Ely**

Exercícios 04 – Parte 1

Pesquise a respeito do Liskov Substitution Principle e responda as questões abaixo:

1. Por que o uso do nome próprio Liskov?

O princípio de Liskov recebe esse nome em homenagem a Barbara Liskov, cientista da computação, foi a primeira mulher dos EUA a receber um doutorado na área, tendo contribuições significativas para o meio.

2. Qual a principal imagem relacionada ao princípio e qual a explicação sobre ela?

A principal imagem ou característica que podemos ressaltar sobre esse princípio é a de contrato e contrato de comportamento, a ideia é que uma classe filha deve se comportar de maneira condizente com sua classe base, ou seja, os métodos implementados na classe filha devem respeitar os métodos já estabelecidos na classe base, respeitando os mesmo contratos de entrada(parâmetros) e saída (retorno), por exemplo, em uma classe base foi definido que os parâmetros sejam do tipo *number,* as operações na classe filha não podem alterar esse tipo.

3. Cite um exemplo onde a herança pode ser usada de forma conveniente, porém deixa uma impressão de que está sendo mal aplicada.

Podemos citar um exemplo onde uma hierarquia de classes parece seguir um modelo lógico, porém acaba ficando inadequada ou muito complexa, considerando uma classe base *Animal* , com os métodos *mover() e emitirsom()*, dela são geradas duas sub-classes: *cachorro* e *gato*, porém com o decorrer da hierarquia podem surgir sub-classes cuja o comportamento se distancie bastante do comportamento padrão das suas classes pai, por exemplo as classes *cachorrovoador* e *gatovoador.*

4. Cite um exemplo onde a herança pode ser usada de forma conveniente, porém deixa futuras expansões comprometidas ou com problemas de design

Quando por exemplo há uma hierarquia de classes que inicialmente parece adequada para representar diferentes tipos de veículos, por exemplo, carros, bicicletas e aviões. existe uma classe base Veiculo e diferentes subclasses como Carro, Bicicleta e Aviao. Cada uma dessas subclasses tem seus próprios métodos e características específicas.Todos os veículos possuem o método *mover(),* o problema surge quando se deseja adicionar novos tipos de veículos que não se encaixam facilmente nessa hierarquia. Por exemplo, se quisermos adicionar um Hovercraft que se move tanto na água quanto na terra, a estrutura existente pode não ser flexível o suficiente para acomodar isso sem comprometer a lógica existente.

5. Nos exemplos que você citou, a composição seria mais aplicável? Refaça-os.

RESPOSTA NA PASTA DO REPOSITÓRIO

Sugestão de links para estudo:

1. https://www.youtube.com/watch?v=tIcfvP9jf9k
2. https://web.archive.org/web/20151128004108/http:/www.objectmentor.com/resources/articles/lsp.pdf
3. https://www.engr.mun.ca/~theo/Courses/sd/5895-downloads/sd-principles3.ppt.pdf
4. https://medium.com/@wrong.about/liskov-substitution-principle a982551d584a
5. https://www.tomdalling.com/blog/software-design/solid-class-design-the liskov-substitution-principle/
6. https://springframework.guru/principles-of-object-oriented-design/liskov substitution-principle/
7. https://medium.com/swlh/lsp-liskov-substitution-principle-24311d1f854
8. https://www.youtube.com/watch?v=-Z-17h3jG0A