UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Programação Orientada a Objetos Avançada

Trabalho 3

Professor

Daniel Lucrédio

Aluno

Pedro Henrique Grespan Carneiro, 761029, Engenharia de Materiais

São Carlos, 07 de janeiro de 2021

1. Introdução

O trabalho tem como objetivo criar uma descrição arquitetural para um sistema de urna, seguindo os requisitos apresentados na atividade de tal modo que atenda aos princípios SOLID (com exceção do último princípio, “D”).
Foi utilizado o software StarUML para apresentar a resolução do problema e a sua visualização foi apresentada junto a esse relatório.

2. Decisões de Projeto e Aplicação dos Princípios

2.1. Princípio da Responsabilidade Única

Esse princípio diz que uma classe deve ter uma, e apenas razão para ser modificada, ou seja, o projeto deve ser feito de modo que cada classe apresente apenas uma funcionalidade bem definida.
No projeto realizado é possível perceber que esse princípio foi respeitado, pois apresenta as seguintes responsabilidades:

- as informações sobre a votação são de responsabilidade da classe Urna;
- a classe Mesário será a responsável por liberar a próxima votação;
- a interface Exibição é responsável apenas por mostrar para o eleitor quais as opções disponíveis;
- a interface Entrada recebe o voto de um dos meios e o retorna;
- a interface Feedback apenas será responsável por enviar o aviso após a realização da votação e
- a gravação dos votos é de responsabilidade da interface Gravação.

2.2. Princípio do Aberto-Fechado

O princípio do aberto-fechado diz que um sistema deve ser aberto para extensão e fechado para modificação, ou seja, deve ser possível adicionar novas funcionalidades sem que seja necessário modificar o que já está presente.
É possível ver a utilização desse princípio no projeto, pois, caso seja necessário adicionar novas funcionalidades, como, um novo meio de exibição, será possível

apenas adicionando uma nova classe que implementa a interface Exibição, também é possível para a entrada de dados, interface Entrada, novos feedbacks após a realização do voto, interface Feedback e novas maneiras de registrar o voto, por meio da interface Gravacao.

2.3. Princípio da Substituição de Liskov

O princípio de Liskov diz que as classes derivadas devem ser substituíveis por suas classes base. O que é respeitado no projeto, pois todas as funcionalidades das classes que implementam as interfaces funcionam independentemente de qual classe utilizada.
É possível ver que o princípio é respeitado na interface Feedback, pois independente de qual classe que a implementa for utilizada, será executada a funcionalidade.

2.4. Princípio da Segregação de Interface

Esse princípio diz que interfaces específicas são melhores que interfaces mais genéricas, ou seja, nenhuma classe deve ser forçada a implementar métodos que não serão utilizados. É possível ver que no projeto todas as interfaces criadas possuem apenas as funcionalidades utilizadas por suas subclasses, respeitando assim este princípio.
Um exemplo do uso do princípio é na interface Gravacao e suas subclasses, ArmInterno terá mais funcionalidades que as outras subclasses, mas não é forçada a implementação dessas funcionalidades nas outras subclasses.

3. Conclusão

O desenvolvimento do projeto não apresentou grandes dificuldades quando pensado sem a aplicação dos princípios, mas junto ao desenvolvimento foi possível entendê-los mais profundamente e a sua aplicação melhorou muito a estrutura do projeto, o muito mais responsivo e de melhor entendimento, evitando possíveis problemas futuros e aumentando a sua longevidade.

4. Referências

- https://docs.google.com/document/d/1R_CbWsOXOfRHdv31F1Not_WrIpjdTQ1S9eodeQ58LyE/edit
- https://www.youtube.com/watch?v=gbgV5jKZfTk&list=PLaPmgS59eMSFYb42BcmYzVcClCh0t-26L
- https://www.venturus.org.br/o-que-e-solid/
- https://ezdevs.com.br/conhecendo-os-principios-do-solid/