Santander Coders | Automação de Testes

BE-JV-003 PROGRAMAÇÃO ORIENTADA A OBJETOS II

Professor: Thalles Trevizan
Aluno: Wandemberg S. Santos

Princípios do SOLID

Popularizado no início dos anos 2000 pelo Engenheiro de Software Robert C. Martin, e trata de princípios que visam garantir boas práticas no desenvolvimento de software que utilizan programação orientada a objetos. Abaixo, escrevo a definição desses princípios, junto com exemplos de casos que violam esse princípio.

S - Single Responsability

Estabelece que cada classe deve ter uma única responsabilidade, de forma a ser facilmente compreendida e mantida.
Exemplo: Uma classe que pretende modelar simultaneamente cachorros e gatos, possuindo atributos exclusivos e compartilhados a ambos. Nesse caso deve-se explorar a herança, um dos pilares da POO.

O - Open-Closed

Trata da flexibilidade das classes. Uma classe deve ser aberta apenas a expansão, de forma que novas funcionalidades possam ser acrescentadas sem que seu código seja modificado.
Exemplo: Uma classe Pessoa com atributos e métodos característicos de uma classe que modela um funcionário. Nesse caso, torna-se dificultoso herdar essa classe pela classe Cliente, por exemplo.

L - Liskov Substitution

Fala que todo objeto filho deve ser capaz de substituir uma superclasse. Dessa forma, todo método da superclasse continua sendo utilizável, não limitando o programa e reduzindo a ocorrência de erros caso haja uma substituição.
Exemplo: Se o objeto Gato, em uma aplicação de Petshop, não herda o método get para o atributo Porte, isso pode atrapalhar tanto desenvolvedor na hora de estabelecer alguma classificação para os dados no banco de dados, como pode reduzir a usabilidade para o usuário.

I - Interface Segregation

Discute sobre a necessidade dos métodos implementados em uma classe. Assim, estabelece que é importante não implementar métodos que não irão ser usados, apoiando o uso de interfaces específicas para a implementação.
Exemplo: Suponha que a Classe Animal implemente previamente métodos relacionados a variados animais. Dessa forma, ao herdar ou implementar métodos de uma interface, se têm métodos descartáveis e fora de contexto.

D - Dependency Inversion

Se preocupa com a redução de complexidade dos módulos de um sistema, focado em diminuir o acoplamento entre as classes através do uso de classes abstratas, aumentando a flexibilidade do sistema e facilitando seu aprimoramento.
Exemplo: Se o objeto Pessoa faz uso de muitos métodos provenientes de um acoplamento com a classe animal, isso torna essas classes fortemente acopladas e dependentes, proporcionando mais erros e inconvenientes após alterações.