

Trabalho L.P.O.O. (Object Calisthenics)

Nome: Rafael Alves Faria

Resumo

O presente trabalho tem o intuito de apresentar e explicar as nove regras do *Object Calisthenics*, um conjunto de práticas que visam melhorar a qualidade do código em Programação Orientada a Objetos. Essas regras foram introduzidas por Jeff Bay no livro *The ThoughtWorks Anthology* e têm como objetivo promover maior legibilidade, coesão, manutenibilidade e testabilidade do software. Além disso, elas contribuem para a aplicação dos princípios **SOLID**, um acrônimo que reúne cinco diretrizes fundamentais da programação orientada a objetos. Embora não sejam de caráter obrigatório, tais práticas funcionam como um exercício de disciplina que auxilia o programador a desenvolver um estilo de escrita mais claro, organizado e sustentável, permitindo um constante aperfeiçoamento da lógica e da mentalidade de desenvolvimento.

Palavras-chave: Boas práticas; Legibilidade de código; *Object Calisthenics*; Princípios SOLID; Programação Orientada a Objetos.

Introdução

O *Object Calisthenics* é um conjunto de nove regras e boas práticas voltadas para a programação orientada a objetos, com o objetivo de estimular a aplicação dos princípios SOLID. Criado por Jeff Bay, no livro *The ThoughtWorks Anthology*, esse conjunto de orientações busca melhorar a legibilidade, a manutenibilidade, a testabilidade e a compreensão do código, servindo como uma forma de exercício para programadores que desejam aprimorar seu estilo de desenvolvimento.

Regras do *Object Calisthenics*

As regras do *Object Calisthenics* constituem diretrizes que ajudam o desenvolvedor a produzir código mais claro, coeso e de fácil manutenção. São elas:

1. *Only One Level of Indentation Per Method*

Cada método deve possuir apenas um nível de indentação, incentivando a criação de métodos curtos e simples, o que melhora a legibilidade e evita complexidade desnecessária.

2. *Don't Use the ELSE Keyword*

O uso do comando ‘else’ deve ser evitado. Em vez disso, recomenda-se o uso de ‘early return’ ou polimorfismo para simplificar o fluxo de execução e reduzir ramiﬁcações excessivas.

3. *Wrap All Primitives and Strings*

Tipos primitivos e strings devem ser encapsulados em objetos específicos do domínio, prevenindo o problema conhecido como *primitive obsession* e aumentando a expressividade do código.

4. First-Class Collections

Classes que lidam com coleções não devem possuir outros atributos além da própria coleção. Quando for necessário manipular elementos em conjunto, deve-se criar uma classe dedicada, que centralize os comportamentos relacionados a essa coleção.

5. One Dot Per Line

O encadeamento excessivo de chamadas, como '*obj.getX()*', '*.getY()*', '*.getZ()*', deve ser evitado. Em vez disso, os objetos devem expor comportamentos de forma clara, sem obrigar outros objetos a conhecerem sua estrutura interna.

6. Don't Abbreviate

Abreviações em nomes de variáveis, métodos ou classes devem ser evitadas, pois prejudicam a clareza. Recomenda-se utilizar nomes completos e descritivos, seguindo as convenções da linguagem utilizada.

7. Keep All Entities Small

Classes não devem ultrapassar cinquenta linhas de código. Entidades pequenas favorecem a compreensão e a manutenção, tornando o software mais simples de evoluir.

8. No Classes With More Than Two Instance Variables

O número de variáveis de instância em uma classe deve ser limitado a duas. Essa prática promove alta coesão e responsabilidade única, evitando que uma classe desempenhe mais funções do que deveria.

9. No Getters/Setters/Properties

Métodos de acesso como *getters* e *setters* devem ser evitados, uma vez que podem incentivar a exposição desnecessária de dados internos. Os objetos devem priorizar comportamentos, em vez de apenas fornecer ou alterar informações.

Conclusão

Seguir as boas práticas do *Object Calisthenics* auxilia desenvolvedores a produzirem códigos mais coesos, organizados e sustentáveis. Embora não se trate de um conjunto de regras obrigatórias, esse exercício de disciplina incentiva o profissional a aprimorar continuamente sua forma de pensar e escrever código. Dessa forma, além de melhorar a qualidade técnica dos sistemas, também contribui para o desenvolvimento de uma mentalidade mais sólida e madura na programação orientada a objetos.

Referências

CRUZ, Rafael. Desenvolva um código melhor com Object Calisthenics. *Medium*, 6 ago. 2018. Disponível em: https://medium.com/@rafaelcruz_48213/desenvolva-um-c%C3%B3digo-melhor-com-object-calisthenics-d5364767a9ba. Acesso em: 27 ago. 2025.

PAIXÃO, João Roberto da. O que é SOLID: o guia completo para você entender os 5 princípios da POO. *Medium*, 6 jan. 2019. Disponível em: <https://medium.com/desenvolvendo-com-paixao/o-que-%C3%A9-solid-o-guia-completo-para-voc%C3%AA-entender-os-5-princ%C3%ADpios-da-poo-2b937b3fc530>. Acesso em: 27 ago. 2025.