O Princípio S do SOLID

Single Responsability Principle - SRP

Introdução ao conceito S do SOLID

Imagine que você tem muitos brinquedos diferentes espalhados pelo quarto. Quando você quer encontrar aquele brinquedo específico, fica difícil, não é?

Agora, imagine que você guarda todos os brinquedos em uma caixa, mas mistura carrinhos, bonecas, e quebra-cabeças no mesmo lugar. Isso não ajuda muito, porque, na hora de pegar um brinquedo, você ainda precisa mexer em tudo.

Agora, se você tivesse uma caixa só para os carrinhos, outra para as bonecas e mais uma para os quebra-cabeças, seria muito mais fácil de achar e usar cada brinquedo.

O conceito de S no SOLID é como essa organização: "Single Responsibility Principle" ou Princípio da Responsabilidade Única. Ele diz que cada "caixa" (classe) no código deve ter uma função bem definida e fazer apenas uma coisa. Assim, fica muito mais fácil de encontrar, corrigir ou melhorar o código, exatamente como seria com seus brinquedos organizados.

Agora, vamos ver isso no mundo real com exemplos práticos em Java.

Sem aplicar o princípo SRP:

```java
public class Funcionario {
    private String nome;
    private double salario;

    public Funcionario(String nome, double salario) {
        this.nome = nome;
        this.salario = salario;
    }

    public void calcularSalario() {
        // Lógica para calcular o salário
    }

    public void salvarNoBancoDeDados() {
        // Lógica para salvar no banco de dados
    }

    public void gerarRelatorio() {
        // Lógica para gerar relatório
    }
}
```

Neste exemplo, a classe Funcionario está fazendo muitas coisas: calcula salário, salva no banco de dados e gera relatório. É como se tivéssemos todos os brinquedos misturados em uma caixa só!

Aplicando o princípo SRP:

```java
public class Funcionario {
    private String nome;
    private double salario;

    public Funcionario(String nome, double salario) {
        this.nome = nome;
        this.salario = salario;
    }

    // Métodos getters e setters
}

public class CalculadoraSalario {
    public double calcularSalario(Funcionario funcionario) {
        // Lógica para calcular o salário
    }
}

public class RepositorioFuncionario {
    public void salvar(Funcionario funcionario) {
        // Lógica para salvar no banco de dados
    }
}

public class GeradorRelatorio {
    public void gerarRelatorio(Funcionario funcionario) {
        // Lógica para gerar relatório
    }
}
```

Agora, temos várias classes, cada uma com uma única responsabilidade. É como ter várias caixinhas de brinquedos organizadas!

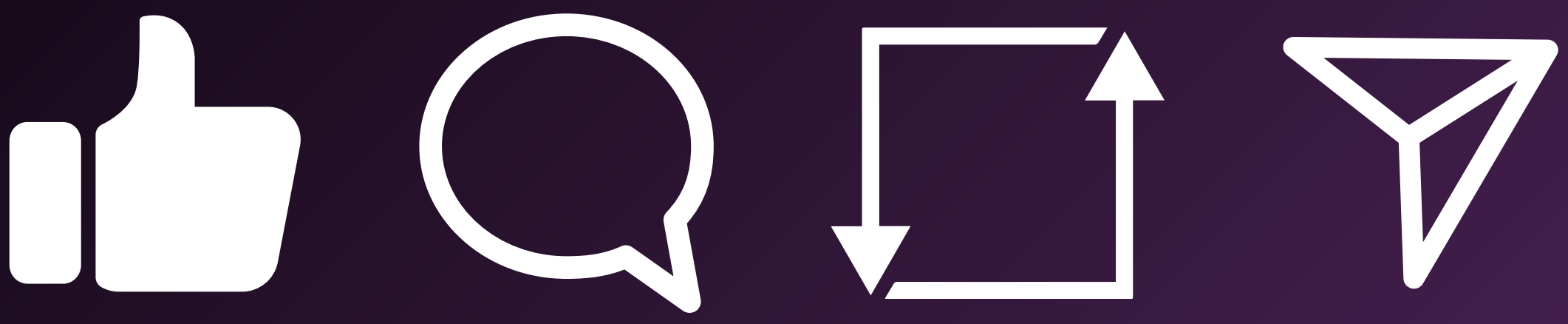
Benefícios de aplicar o SRP:

- **Código mais fácil de entender e manter: Ao separar as responsabilidades, o código fica mais organizado e fácil de entender.**
- **Menos bugs: Ao isolar as responsabilidades, é menos provável que uma mudança em uma parte do código cause problemas em outra.**
- **Reutilização de código: Classes com responsabilidades bem definidas são mais fáceis de reutilizar em outros projetos.**
- **Testes unitários mais simples: Classes com uma única responsabilidade são mais fáceis de testar isoladamente.**

Seguir o Princípio da Responsabilidade Única é como organizar seus brinquedos em caixas separadas: facilita a manutenção e a reutilização. No código, isso significa que cada classe deve ter apenas um motivo para ser alterada, concentrando-se em uma única função.

Ao aplicar esse conceito, você cria um código mais limpo, legível e fácil de manter!

Compartilha com aquele que quer aprender.



curti **comenta** **compartilha** **envia**

Que a força do código esteja com você !