

BOAS PRÁTICAS



Gabriel Brazão (ou só Brazão)

Arquiteto de Soluções na
Capgemini/RDI.

Formado em Sistemas para Internet e
Pós-Graduado em Engenharia de
Sistemas pela Faculdade Impacta.

Há 14 anos atuando com análise e
desenvolvimento de sistemas.

<https://www.linkedin.com/in/gbrazao/>

Agenda



Parte I

- ✓ Indentação de código
- ✓ Padrão de Nomenclatura
- ✓ Comentários no Código
- ✓ Números “Mágicos”
- ✓ Princípio D.R.Y.
- ✓ Outras boas práticas

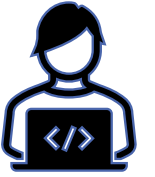


Parte II

- ✓ S.O.L.I.D.
 - ✓ Single Responsibility Principle
 - ✓ Open-Closed Principle
 - ✓ Liskov Substitution Principle
 - ✓ Interface Segregation Principle
 - ✓ Dependency Inversion Principle



Clean Code



“Qualquer pessoa consegue escrever código que um computador entenda. Bons programadores escrevem código que humanos possam entender”

- Martin Fowler



Indentação de código

“Recurar o texto em relação à margem da folha; inserir espaços entre a margem e o começo da linha de um parágrafo: antes de finalizar este trabalho, é preciso indentar.”



Indentação de código

```
if (a) {  
  if (b) {  
    while (c) {  
      d();  
    }  
  } else if (e) {  
    f();  
  } else if (g) {  
    h();  
  }  
} else if (i) {  
  while (j) {  
    k();  
  }  
}
```

- Onde começa?
- Onde termina?
- Como chegar no (f)?

```
if (a) {  
    if (b) {  
        while (c) {  
            d();  
        }  
    } else if (e) {  
        f();  
    } else if (g) {  
        h();  
    }  
} else if (i) {  
    while (j) {  
        k();  
    }  
}
```

- Onde começa?
- Onde termina?
- Como chegar no (f)?



Padrão de Nomenclatura

Como dar nomes?

- Escolha nomes descritivos e inequívocos.
- Nomes pronunciáveis.
- Nomes pesquisáveis.

```
1 string n;  
2 decimal preCombAdtv;
```

```
1 string nomeCliente;  
2 decimal precoCombustivelAditivo;
```

```
1 public class CliPJInterstad()  
2 {  
3  
4 }
```

```
1 public class ClientePJInterestadual()  
2 {  
3  
4 }
```



Padrão de Nomenclatura



PascalCase

Escreve palavras compostas ou frases montadas com palavras onde a primeira letra de cada palavra é iniciada com maiúscula.



CamelCase

CamelCase é a denominação em inglês para a prática de escrever palavras compostas ou frases, onde a primeira letra da primeira palavra é iniciada com minúscula e unidas sem espaços



Padrão de Nomenclatura

PascalCase

- Classe

```
1 public class Cliente
2 {
```

- Método Público e Propriedades

```
1 public string Nome { get; set; }
2
3 public int CalculaFerias()
4 {
5
6 }
```

- Variável Pública
- Interfaces (Começando sempre com I)

```
1 public interface ICliente
2 {
```



Padrão de Nomenclatura

CamelCase

- Método Privado

```
1 private void calcularFerias
2 {
3 }
```

- Variável Privada (também utilizado com "_")

```
1 private int impostoCalculado;
```

```
1 private int _impostoCalculado;
```

- Parâmetros de Métodos

```
1 public void CalcularFerias(Funcionario funcionario, int dias)
2 {
```



Padrão de Nomenclatura

Outros

- Constantes

```
1  const int FATOR_RETENCAO = 5;
```

- Abordagem "Positiva" e utilização do prefixo "is"

```
1  bool isVisible;
```

- Classes e Arquivos devem possuir o mesmo nome

```
► C# Pagamento.cs
```

```
0 references  
public class Pagamento  
{  
    0 references
```

- Evitar anotação Húngara



Padrão de Nomenclatura

Exercício

```
1  public class calculadora ←←←
2  {
3      const int fator = 5; ←←←
4
5      public int mult(int NUMERO, int NUMERO_2) ←←←←
6      {
7          return NUMERO * NUMERO_2; ←←←
8      }
9
10     public int somar(int NUMERO_1, int NUMERO_2)
11     {
12         return NUMERO_1 * NUMERO_2; ←←←
13     }
14
15     ....
16 }
```



Comentários no código

- Evitar comentários explicativos, escolhendo sempre a refatoração.

Exemplo:

```
1 // Verifica se o Cliente está ativo e é maior de idade
2 if (cliente.status == 1 && cliente.idade >= 18)
```

```
1 // Verifica se o Cliente está ativo e é maior de idade
2 if (cliente.status == 1 && cliente.idade >= 18 && cliente.estadoCivil == 2)
```

```
1 if (cliente.elegivelPromocaoCarnaval())
```

- Evitar comentários óbvios.

```
1 //Retorna o nome
2 public string getNome()
```



Comentários no código

- Evitar “narrar” o código.

```
1 public string ValidarCliente( Cliente cliente)
2 {
3     //Verifica se o cliente é maior de idade.
4     if (cliente.idade <= 18)
5     {
6         return "Cliente não é maior de idade";
7     }
8
9     //Verifica se o cliente é motorista.
10    if (cliente.isMotorista)
11    {
12        return "Cliente é motorista";
13    }else //Senão for motorista, então é passageiro.
14    {
15        return "Cliente é passageiro";
16    }
17
18    //Retorna cliente apto para promoção
19    return "Cliente está apto";
20 }
```

CODE COMMENTS BE LIKE



```
//I am not sure why this works but it fixes the problem.
```

```
/* Não sei bem porque isso funciona mas resolve o problema. */
```

```
//When I wrote this, only God and I understood what I was doing  
//Now, God only knows
```

```
/* Quando eu escrevi isso somente Deus e eu sabíamos o que eu estava  
fazendo. Agora só Deus sabe */
```

```
// Magic. Do not touch.
```

```
/*Mágica. Não toque. */
```

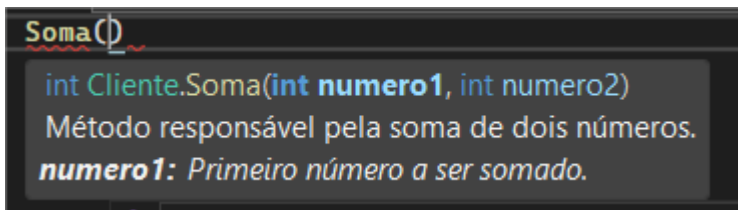


XML Docs

- Funcionalidade geradora de comentários a partir da assinatura do método.

```
1  /// <summary>
2  /// Método responsável pela soma de dois números.
3  /// </summary>
4  /// <param name="numero1">Primeiro número a ser somado.</param>
5  /// <param name="numero2">Segundo número a ser somado</param>
6  /// <returns>Resultado da soma de dois números</returns>
7  public int Soma(int numero1, int numero2)
8  {
9      return (numero1 + numero2);
10 }
```

- Comentário acessível pelo intellisense.



Soma()


int Cliente.Soma(int numero1, int numero2)
Método responsável pela soma de dois números.
numero1: Primeiro número a ser somado.



Números “Mágicos”

- Valores numéricos ou textuais constantes no Código que não são auto descritivos.

```
1 public double aplicaDesconto(double valor)
2 {
3     return valor * 0.50;
4 }
```





Princípio D.R.Y.

- Don't Repeat Yourself. (Não se repita)
- O princípio DRY afirma que todo conhecimento deve ter uma representação única, inequívoca e autoritária dentro de um sistema.
- Evitar a repetição de qualquer parte de um sistema é uma característica desejável.
- O código que é comum a pelo menos duas partes diferentes do seu sistema deve ser fatorado em um único local para que ambas as partes o chamem.
- **Tudo isso significa que você deve parar de copiar e colar imediatamente em seu software.**



Outras boas práticas

- Utilizar blocos Try/Catch.
- Não construir métodos com mais de 7 parâmetros.

```
1 public void GerarRelatorioAnual( List<Cliente> clientes, List<Escritorio> escritorios, List<Vendedor> vendedores,  
2                               int anoBase, bool validaCentroCustos, int fatorCalculoAno,  
3                               bool utilizaBalanceteContabil, TipoSaidaRelatorio tipoSaidaRelatorio, bool imprimeResumo)
```

```
product.GerarRelatorioAnual()  
void Product.GerarRelatorioAnual(List<Cliente> clientes, List<Escritorio> escritorios, List<Vendedor> vendedores, int anoBase,  
                                bool validaCentroCustos, int fatorCalculoAno, bool utilizaBalanceteContabil, TipoSaidaRelatorio tipoSaidaRelatorio,  
                                bool imprimeResumo)
```

- Evitar métodos extensos.



DÚVIDAS





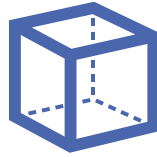
COFFEE TIME



S.O.L.I.D



Single Responsibility Principle



Open-Closed Principle



Liskov Substitution Principle



Interface Segregation Principle

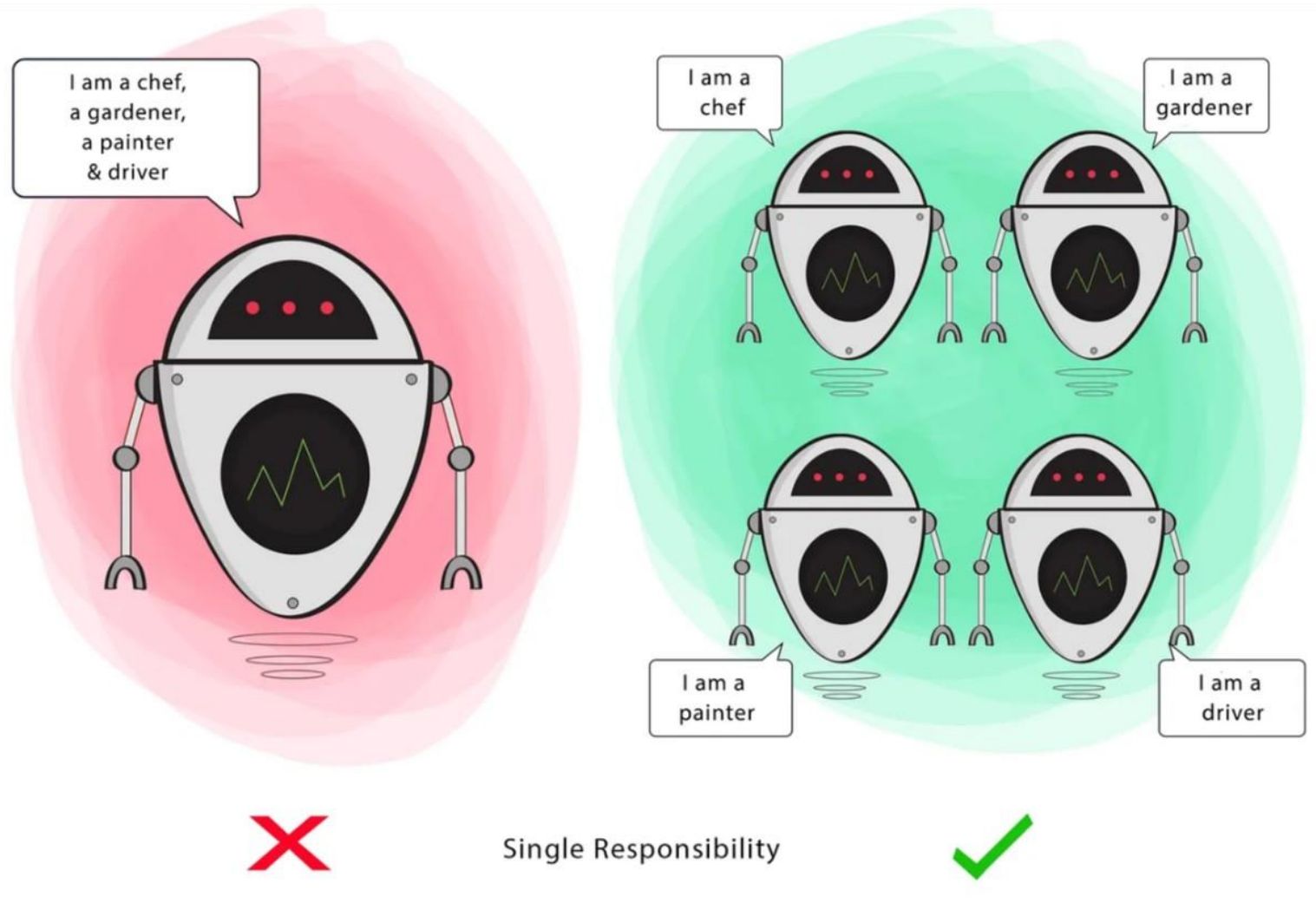


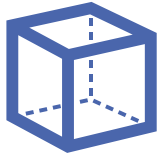
Dependency Inversion Principle



Single Responsibility Principle

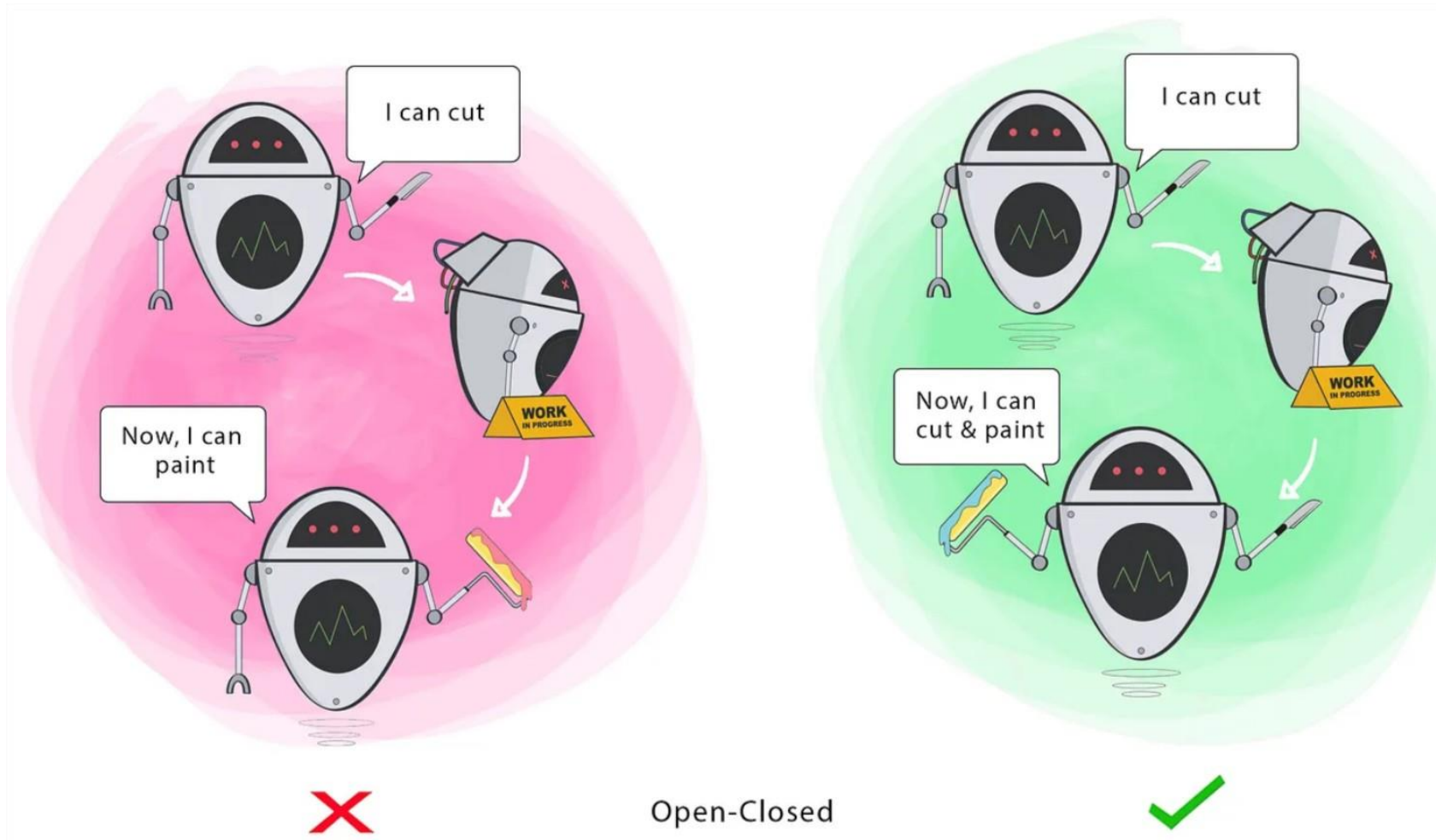
Princípio da Responsabilidade Singular





Open-Closed Principle

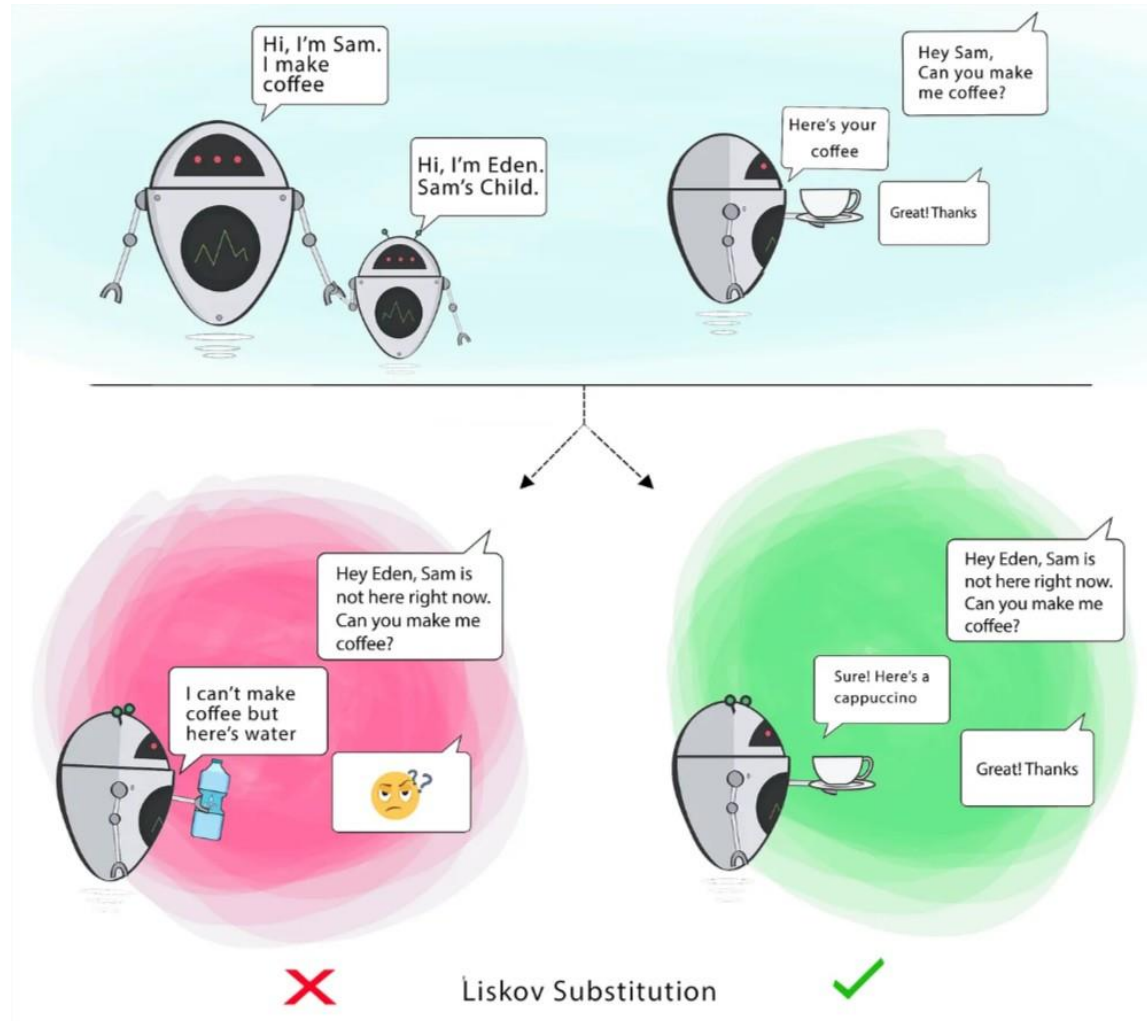
Princípio Aberto-Fechado





Liskov Substitution Principle

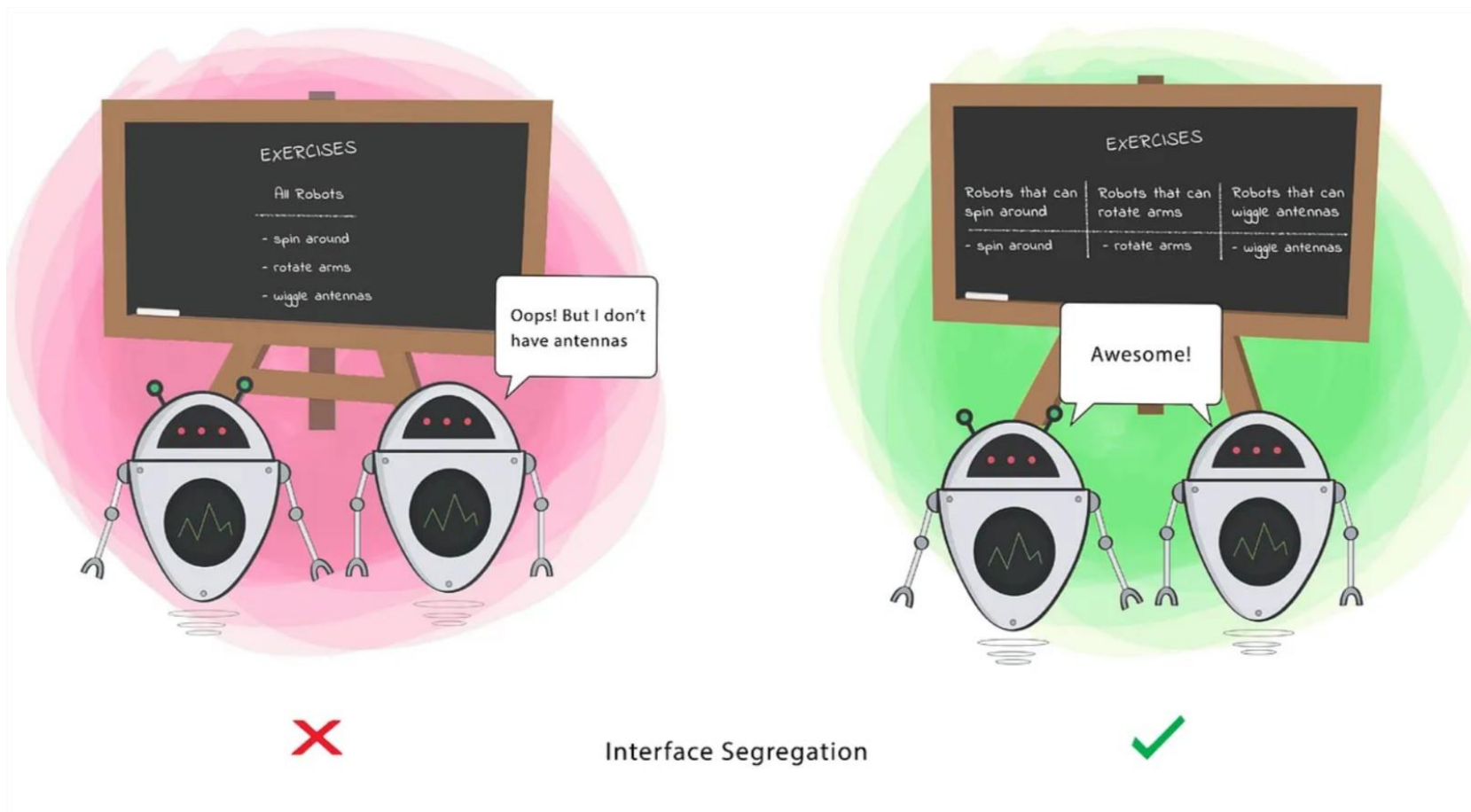
Princípio da substituição de Liskov

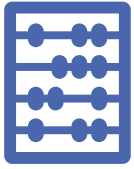




Interface Segregation Principle

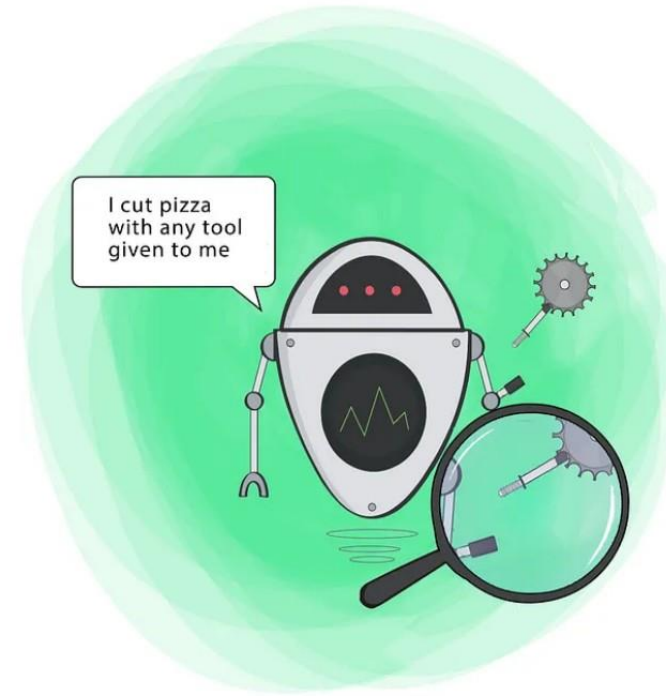
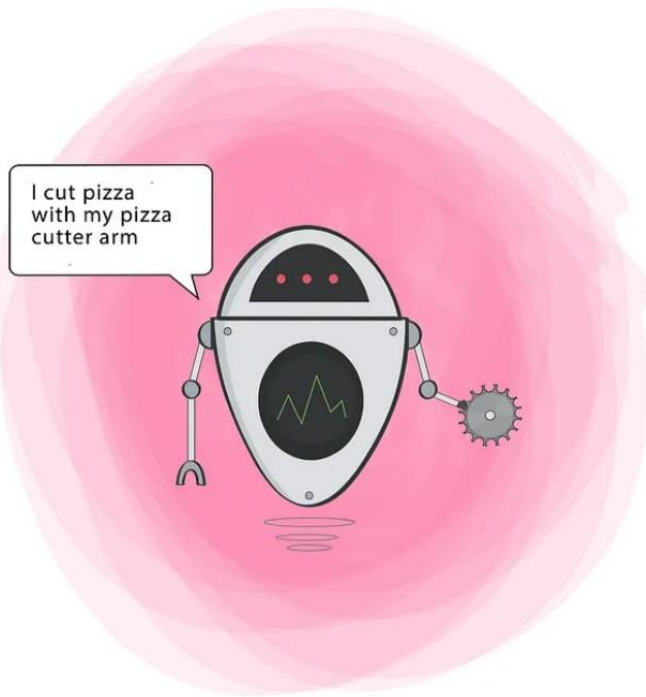
Princípio da Segregação da Interface





Dependency Inversion Principle

Princípio da Inversão de Dependência



Dependency Inversion



DÚVIDAS



OBRIGADO!



Fonte

- <https://learn.microsoft.com/pt-br/dotnet/csharp/fundamentals/coding-style/coding-conventions>
- <https://github.com/dotnet/runtime/blob/main/docs/coding-guidelines/coding-style.md>
- <https://rponte.com.br/2016/02/26/dica-de-programacao-2-numeros-magicos/>
- <https://devinduct.com/blogpost/57/software-design-principles-the-dry-don-t-repeat-yourself-principle>
- <https://www.c-sharpcorner.com/article/dont-repeat-yourselfdry-design-principle/>
- <https://medium.com/backticks-tildes/the-s-o-l-i-d-principles-in-pictures-b34ce2f1e898>
- <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/xml/doc/recommended-tags>