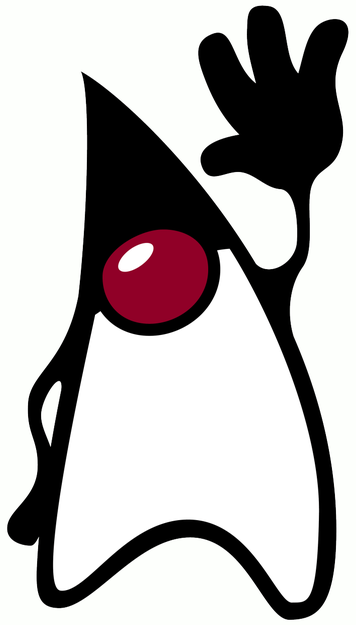

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

# Princípios S.O.L.I.D

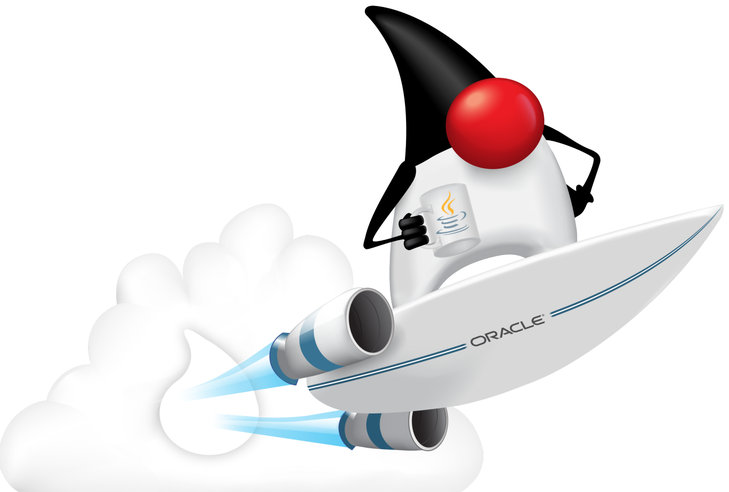
## QXD0007 - Programação Orientada a Objetos



Prof. Bruno Góis Mateus (brunomateus@ufc.br)

# Conteúdo

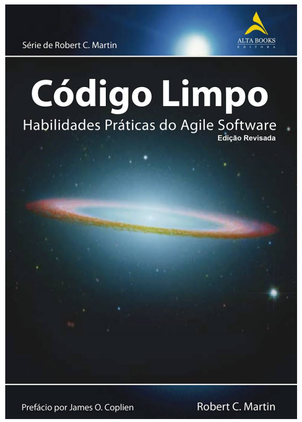
- Introdução
- Single Responsibility Principle
- Open Closed Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion Principle


ORACLE

# Introdução

# Introdução

- Os princípios S.O.L.I.D são fundamentais para garantir que a orientação a objetos foi utilizada corretamente
- Foi inicialmente promovida como um conjunto de princípios por Robert C Martin (Uncle Bob),
- Tornou-se um padrão do campo de engenharia de software




# Introdução

| S.O.L.I.D |
| --- |
| • Single Responsibility Principle - Responsabilidade Única<br>• Open Closed Principle - Aberto e Fechado<br>• Liskov Substitution Principle - Substituição de Liskov<br>• Interface Segregation Principle - Segregação de Interface<br>• Dependency Inversion Principle - Inversão de Dependência |

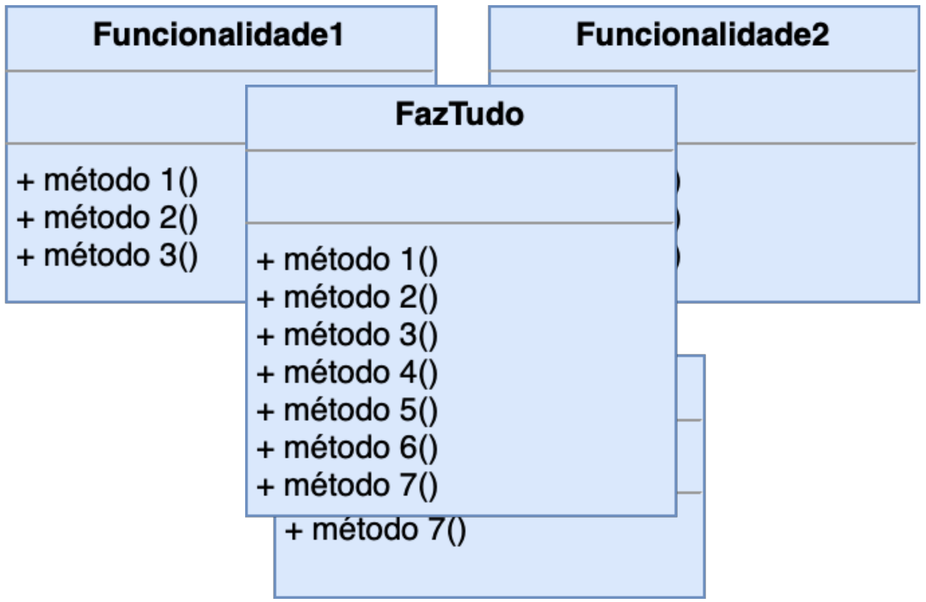
# Single Responsibility Principle

# Single Responsibility Principle

“A class should only have a single responsibility, that is, only changes to one part of the software’s specification should be able to affect the specification of the class.”

“a class should have only one reason to change”

# Single Responsibility Principle



# Single Responsibility Principle

- Cada classe deve ser responsável por apenas uma funcionalidade
- Devem conter apenas métodos e atributos relevantes para a implementação da funcionalidade
  - Tornam a classe mais fácil de ler e entender
  - Favorece a legibilidade já que seria necessário ler apenas uma classe para entender/determinar a sua funcionalidade

# Single Responsibility Principle

- Favorece o uso de outro princípio básico de POO, o Encapsulamento
  - É mais fácil esconder os detalhes (*data hiding*) quando todos atributos e método relacionados a uma tarefa estão em uma única classe
- Se getters and setters forem adicionados ao uma classe que segue o SRP, automaticamente a classe atenderá o princípio do encapsulamento

# Open Closed Principle

# Open Closed Principle

“Software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification.”

# Open Closed Principle

- Aberto para extensão
  - Novas implementações podem ser adicionadas para extender funcionalidades específicas
- Fechadas para modificação
  - Componentes que implementam funcionalidades fundamentais permanecem inalterados
- A ideia deste princípio é tornar os componentes resilientes a mudanças de requisitos

# Open Closed Principle

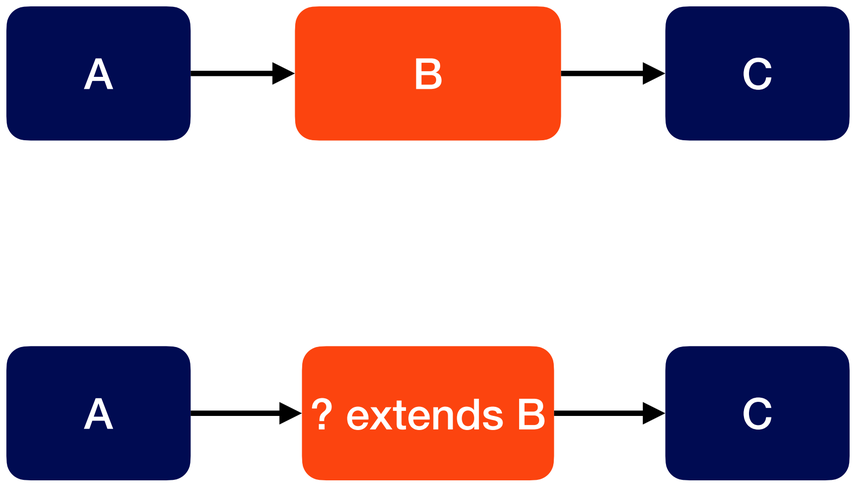
- Por meio do polimorfismo, é possível adicionar funcionalidades nas classes filhas sem alterar as classes mães
- Ao invés de alterar a classe base para acomodar uma funcionalidade nova, é criada uma classe derivada
- Evitando que uma modificação ocasione mudanças em todas as classes dependentes

# Liskov Substitution Principle

# Liskov Substitution Principle

“Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.”

# Liskov Substitution Principle



# Liskov Substitution Principle

- Objetos das subclasses devem se comportar da mesma maneira que os objetos da superclasse
- O comportamento do objeto deve funcionar como um contrato em que o cliente pode confiar
- Este contrato é definido por meio dos métodos públicos, suas restrições e as consequentes mudanças de estado devido a execução dos seus métodos

# Liskov Substitution Principle

- Método original vs Método sobrescrito
  - Implementações diferentes, porém de acordo com o comportamento da super classe
  - O tipo dos argumentos do método sobrescrito podem ser idênticos ou mais amplo
  - O tipo do retorno do método sobrescrito pode ser mais restrito
  - O método sobrescrito pode lançar menos exceções ou exceções mais restritas
- A subclasse não devem permitir mudanças de estados que a super classe não permite

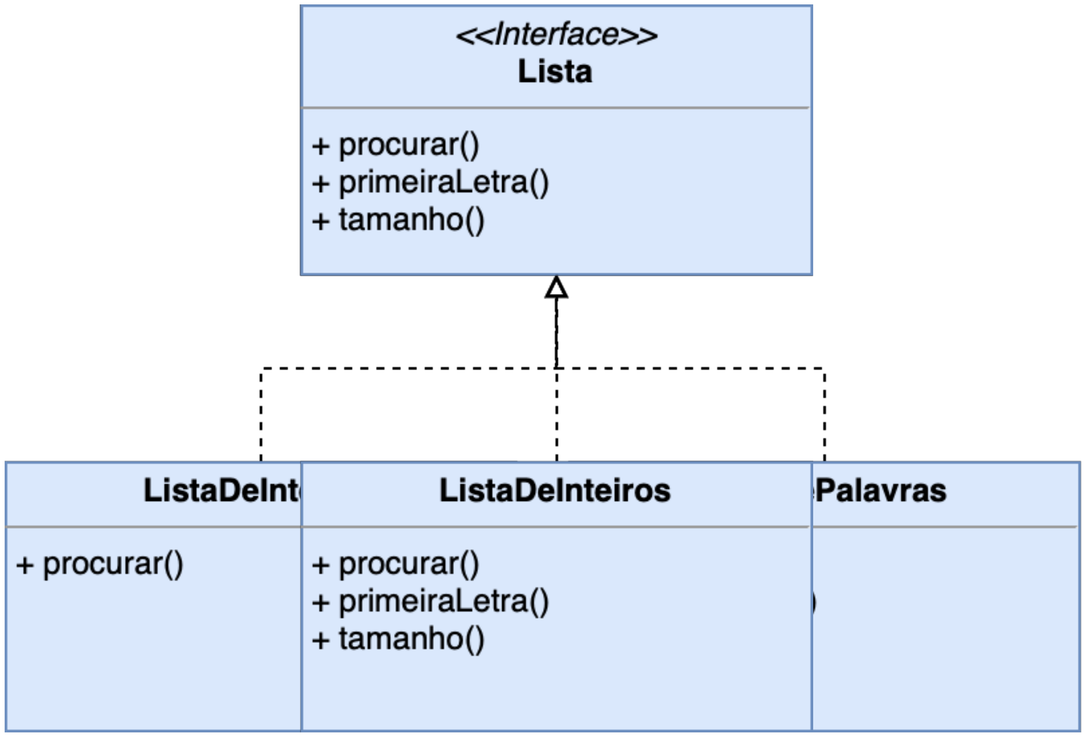
# Interface Segregation Principle

# Interface Segregation Principle

“Clients should not be forced to depend upon interfaces that they do not use.”

Many client-specific interfaces are better than one general-purpose interface.”

# Interface Segregation Principle



# Interface Segregation Principle

- Classes devem possuir apenas comportamentos necessários para entregar a funcionalidade sob sua responsabilidade
- Interfaces não devem ser pensadas para capturar todos os cenários
  - Uma classe não deve ser forçada a implementar um método que não é útil
  - É melhor possuir vários Interfaces

# Interface Segregation Principle

- Interfaces menores
  - São mais de implementar
  - Maior flexibilidade
  - Maior possibilidade de reuso
- Menos classes dependente dessas interfaces
  - Menores as chances de precisar alterá-las devido a uma mudança na Interface

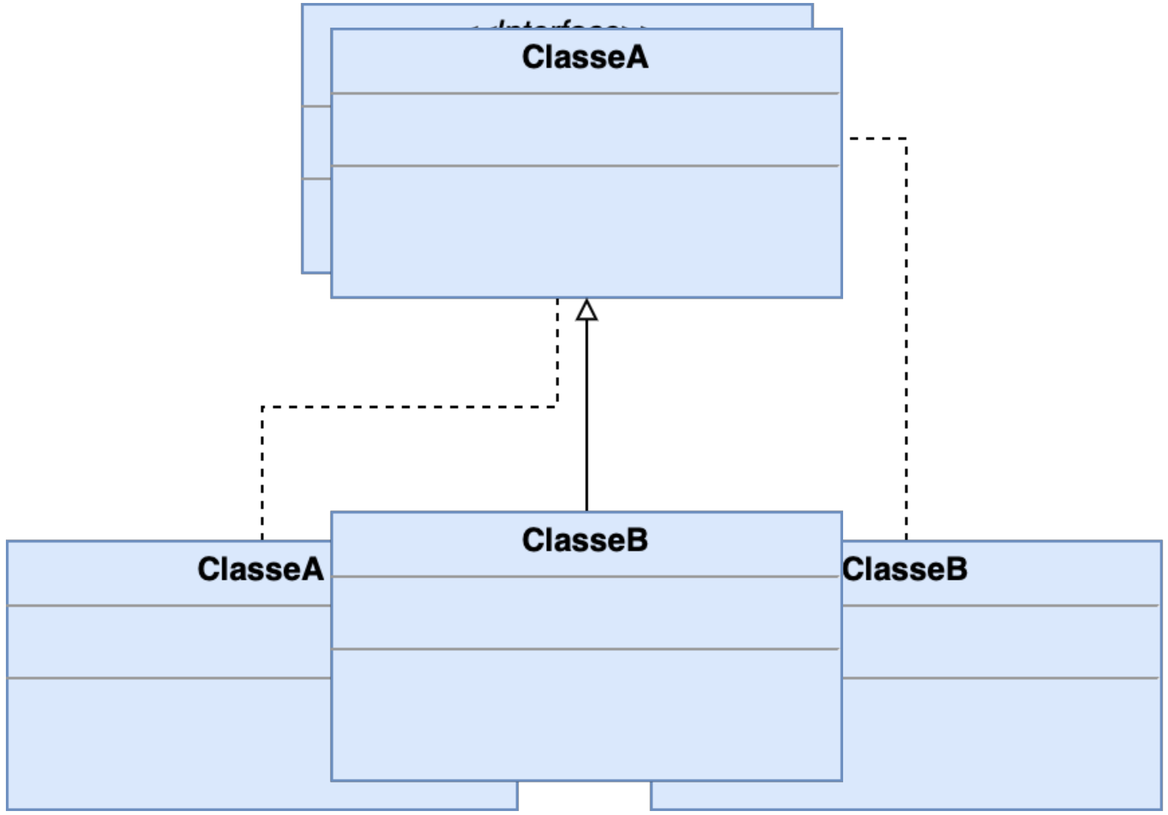
# Dependency inversion principle

# Dependency inversion principle

“High-level modules should not depend on low-level modules. Both should depend on abstractions (e.g. interfaces).”

“Abstractions should not depend on details. Details (concrete implementations) should depend on abstractions.”

# Dependency inversion principle



# Dependency inversion principle

- Módulo de alto nível, que provêm uma lógica complexa, devem ser facilmente reusáveis
  - Não devem ser afetados por mudanças em módulos de baixo nível
- Para isso é necessário introduzir uma camada de abstração
  - Prover o desacoplamento entre módulos de alto e baixo nível
  - Conectando-os através da camada de abstração
  - Mudanças em um não afetam diretamente o outro

# Dependency inversion principle

- A direção da dependência não muda de sentido como o nome sugere
  - Módulos de alto nível passam a depender das abstrações
  - Módulos de baixo nível também a depender das mesmas abstrações
- Facilitando a realização de teste isolados
- Menor acoplamento implica em menos esforço para acomodar mudanças

# Mais informações

- SOLID Principles of OOP
- SOLID Design Principles Explained
- S.O.L.I.D. Principles of Object-Oriented Programming in C#
- SOLID Design Principles in Java Application Development
- The SOLID Principles of Object-Oriented Programming Explained in Plain English
- Liskov Substitution Principle in Java

# Por hoje é só