Princípios SOLID

Aplicando os Princípios SOLID na Prática

Gustavo Gurgel Medeiros

14 de janeiro de 2025

UNIVERSIDADE FEDERAL DO CEARÁ

O que são os Princípios SOLID

Os princípios SOLID são um conjunto de **cinco diretrizes** de design de software definidas por Robert C. Martin. O objetivo dessas diretrizes é ajudar a **criar sistemas mais compreensíveis, flexíveis e fáceis de manter.**

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

UNIVERSIDADE FEDERAL DO CEARÁ

Single Responsibility

Uma classe deve ter apenas uma única responsabilidade

UNIVERSIDADE FEDERAL DO CEARÁ

Exemplo Ruim

```python
class User:
    def __init__(self, name, email):
        self.name = name
        self.email = email

    def get_info(self):
        return f"Usuário: {self.name}, Email: {self.email}"

    def save_to_db(self):
        print(f"Salvando {self.name} no banco de dados.")

    def send_welcome_email(self):
        print(f"Enviando e-mail de boas-vindas para {self.email}.")

```

Exemplo Bom

```python
class User:
    def __init__(self, name, email):
        self.name = name
        self.email = email

    def get_info(self):
        return f"Usuário: {self.name}, Email: {self.email}"

class UserRepository:
    def save_to_db(self, user: User):
        print(f"Salvando {user.name} no banco de dados.")

class EmailService:
    def send_welcome_email(self, user: User):
        print(f"Enviando e-mail de boas-vindas para {user.email}.")
```

Open-Closed

Uma classe deve estar aberta para extensão, mas fechada para modificação.

UNIVERSIDADE FEDERAL DO CEARÁ

Exemplo Ruim

```python
class AreaCalculator:
    def calculate(self, shape):
        if shape["type"] == "circle":
            return 3.14 * (shape["radius"] ** 2)
        elif shape["type"] == "rectangle":
            return shape["width"] * shape["height"]

# Uso
circle = {"type": "circle", "radius": 5}
rectangle = {"type": "rectangle", "width": 4, "height": 6}

calculator = AreaCalculator()
print(calculator.calculate(circle))  # Saída: 78.5
print(calculator.calculate(rectangle))  # Saída: 24

```

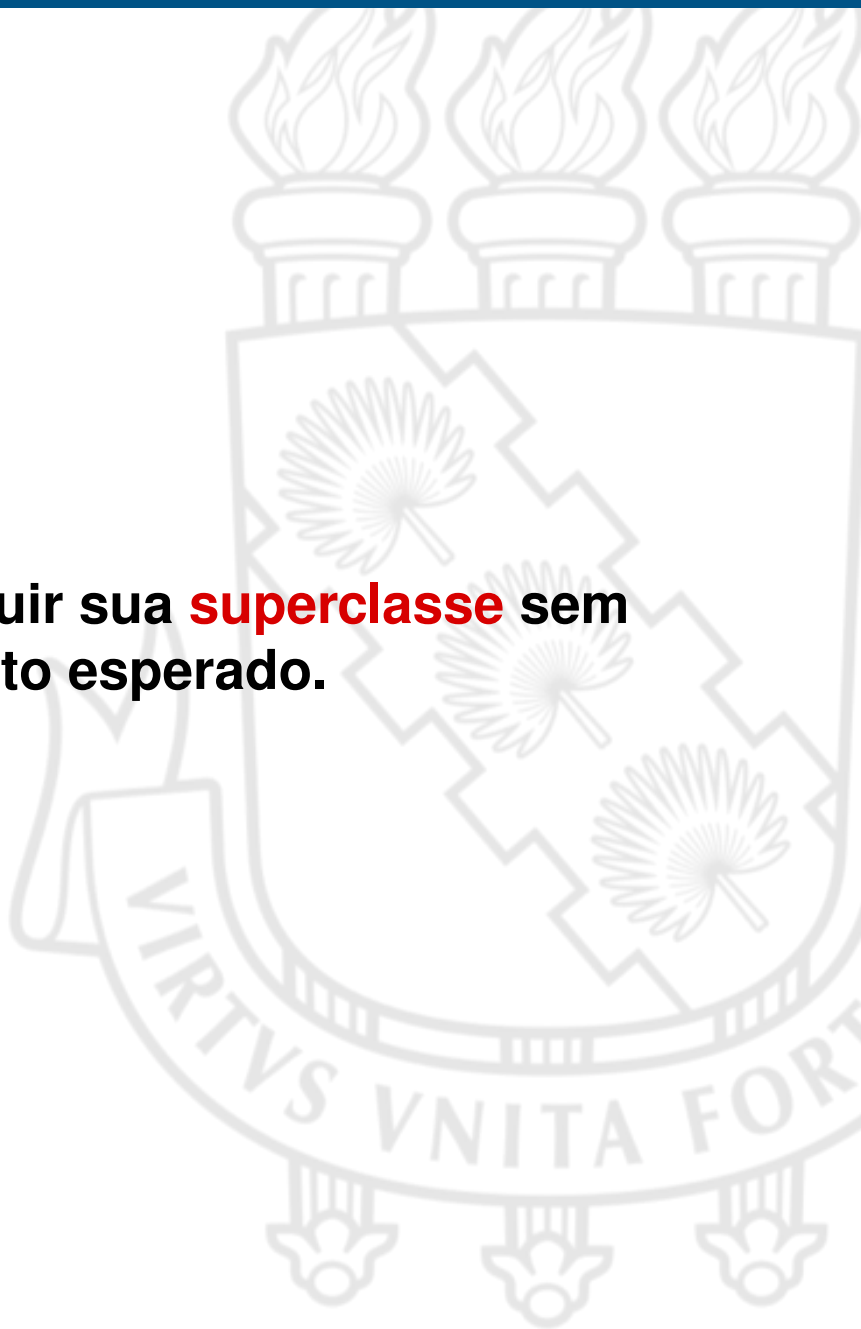
Exemplo Bom

```python
from abc import ABC, abstractmethod

class Shape(ABC):
    @abstractmethod
    def area(self):
        pass

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14 * (self.radius ** 2)

class Rectangle(Shape):
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

# Uso
shapes = [Circle(5), Rectangle(4, 6)]

for shape in shapes:
    print(shape.area())  # Saída: 78.5 e 24

```

UNIVERSIDADE FEDERAL DO CEARÁ

Liskov Substitution

Uma subclasse deve poder substituir sua superclasse sem alterar o comportamento esperado.



Exemplo Ruim

```python
class Bird:
    def fly(self):
        print("Voando")

class Penguin(Bird):
    def fly(self):
        raise NotImplementedError("Pinguins não voam!")

# Uso
bird : Bird = Bird()
bird.fly()  # Voando
bird : Bird  = Penguin()
bird.fly()  # Erro: NotImplementedError: Pinguins não voam!

```

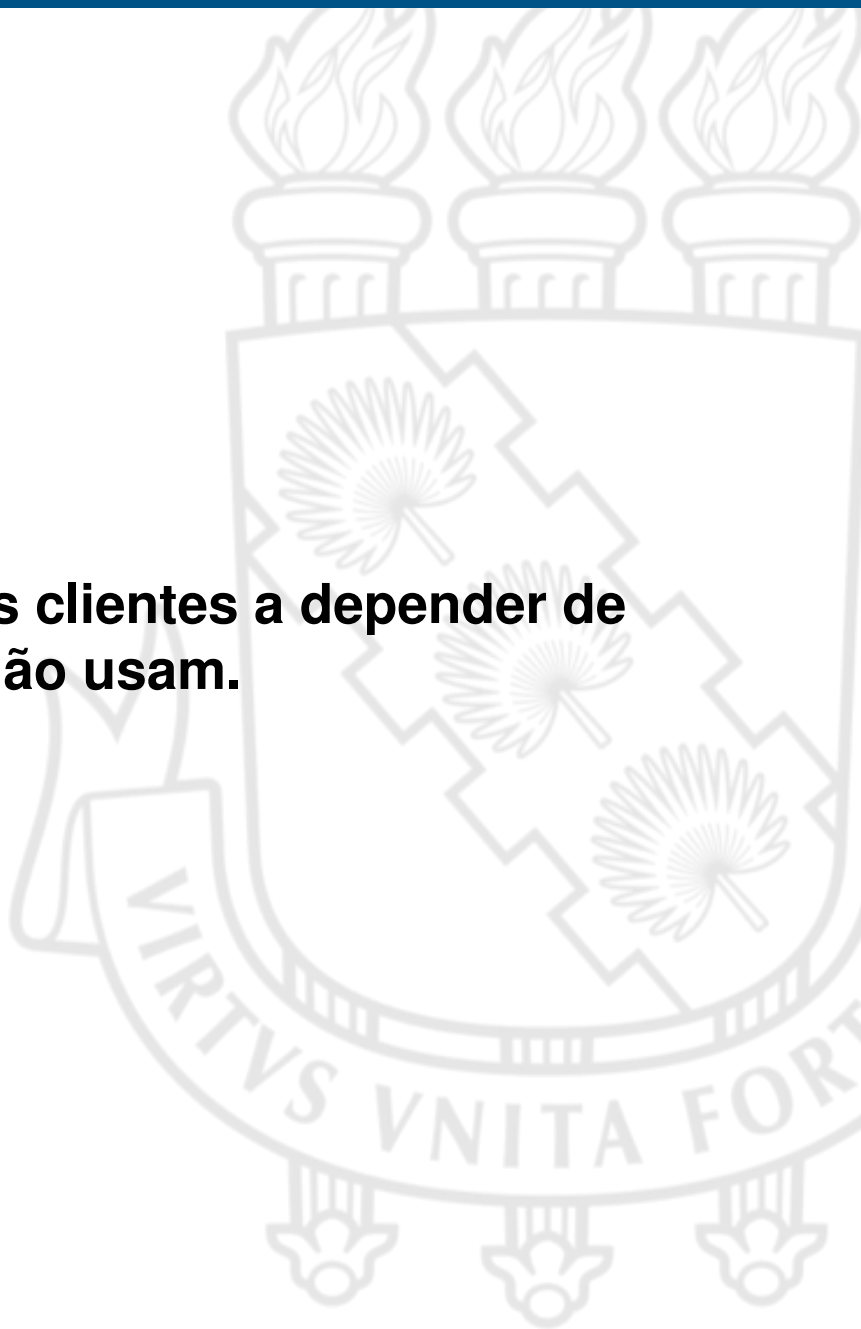
UNIVERSIDADE FEDERAL DO CEARÁ

Exemplo Bom

```python
class Animal:
    def make_sound(self):
        print("Faz algum som")

class Dog(Animal):
    def make_sound(self):
        print("Latindo")

my_pet: Animal = Animal()
my_pet.make_sound() # Faz algum som
my_pet: Animal = Dog()
my_pet.make_sound() # Latindo

```

Interface Segregation

Uma interface não deve forçar seus clientes a depender de métodos que eles não usam.



Exemplo Ruim

```python
from abc import ABC, abstractmethod

class Machine(ABC):
    @abstractmethod
    def print(self):
        pass

    @abstractmethod
    def scan(self):
        pass

class BasicPrinter(Machine):
    def print(self):
        print("Imprimindo...")

    def scan(self):
        raise NotImplementedError("Não sei escanear!")  # Problema!

printer = BasicPrinter()
printer.print()  # Imprimindo...
printer.scan()   # Erro: NotImplementedError: Não sei escanear!

```

Exemplo Bom

```python
from abc import ABC, abstractmethod

class Printer(ABC):
    @abstractmethod
    def print(self):
        pass

class Scanner(ABC):
    @abstractmethod
    def scan(self):
        pass

class BasicPrinter(Printer):
    def print(self):
        print("Imprimindo...")

class AdvancedPrinter(Printer, Scanner):
    def print(self):
        print("Imprimindo...")

    def scan(self):
        print("Escaneando...")

# Uso
basic_printer = BasicPrinter()
basic_printer.print()  # Imprimindo...

advanced_printer = AdvancedPrinter()
advanced_printer.print()  # Imprimindo...
advanced_printer.scan()   # Escaneando...
```

UNIVERSIDADE FEDERAL DO CEARÁ

Dependency Inversion

Módulos de alto nível não devem depender de módulos de baixo nível. Ambos devem depender de abstrações.

Abstrações não devem depender de detalhes. Detalhes devem depender de abstrações.

UNIVERSIDADE FEDERAL DO CEARÁ

Exemplo Ruim

```python
class MySQLDatabase:
    def save(self, data):
        print(f"Salvando '{data}' no MySQL")

class DataService:
    def __init__(self):
        self.db = MySQLDatabase() # Ruim

    def save_data(self, data):
        self.db.save(data)
```

Exemplo Bom

```python
from abc import ABC, abstractmethod

class Database(ABC):
    @abstractmethod
    def save(self, data):
        pass

class MySQLDatabase(Database):
    def save(self, data):
        print(f"Salvando '{data}' no MySQL")

class PostgreSQLDatabase(Database):
    def save(self, data):
        print(f"Salvando '{data}' no PostgreSQL")

class DataService:
    def __init__(self, db: Database):  # Injeção de dependência
        self.db = db

    def save_data(self, data):
        self.db.save(data)
```

UNIVERSIDADE FEDERAL DO CEARÁ

Obrigado(a) pela Atenção!

UNIVERSIDADE FEDERAL DO CEARÁ