

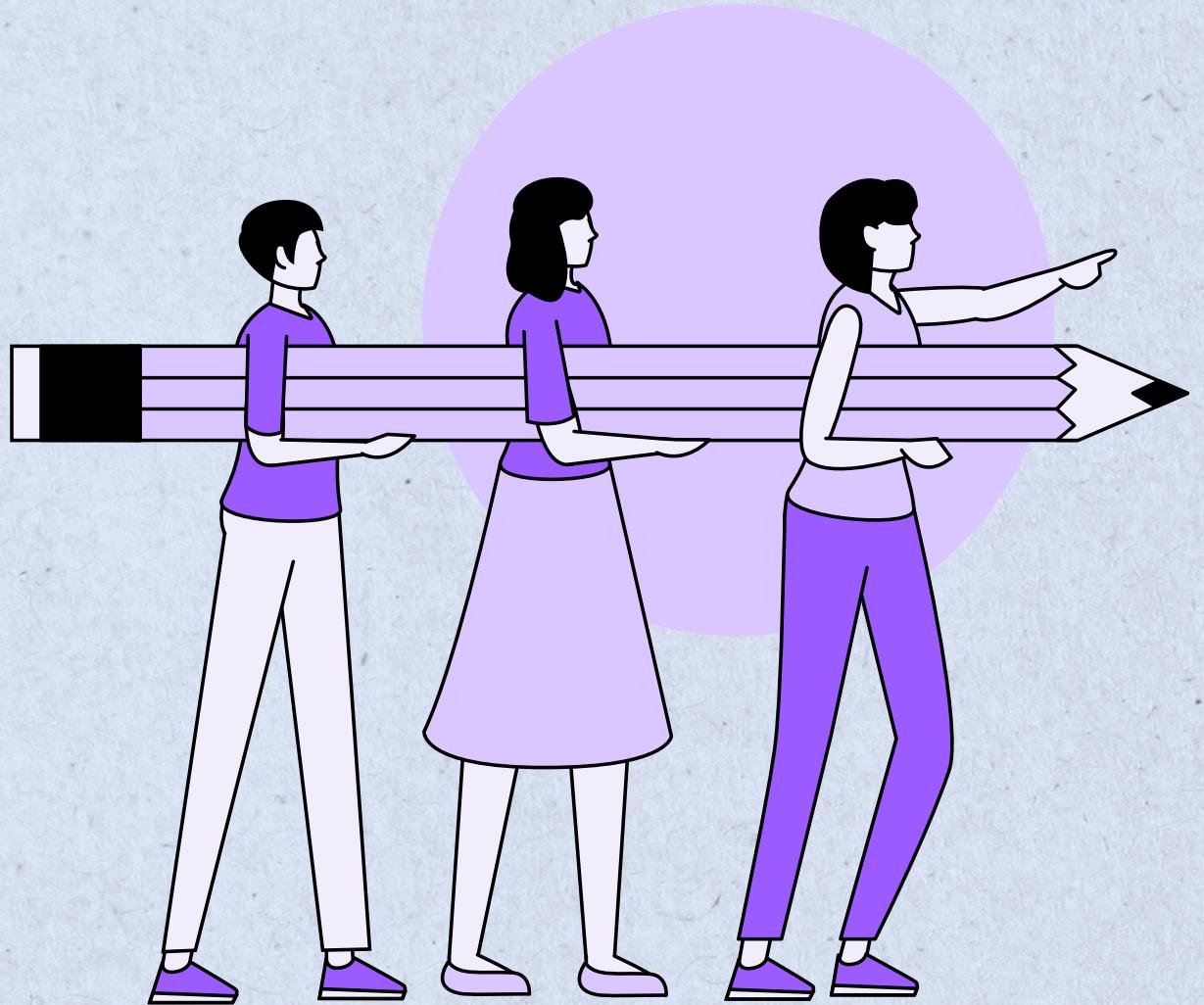
S.O.L.I.D

LISKOV
SUBSTITUTION
PRINCIPLE



O QUE É ?

Parte dos princípios SOLID



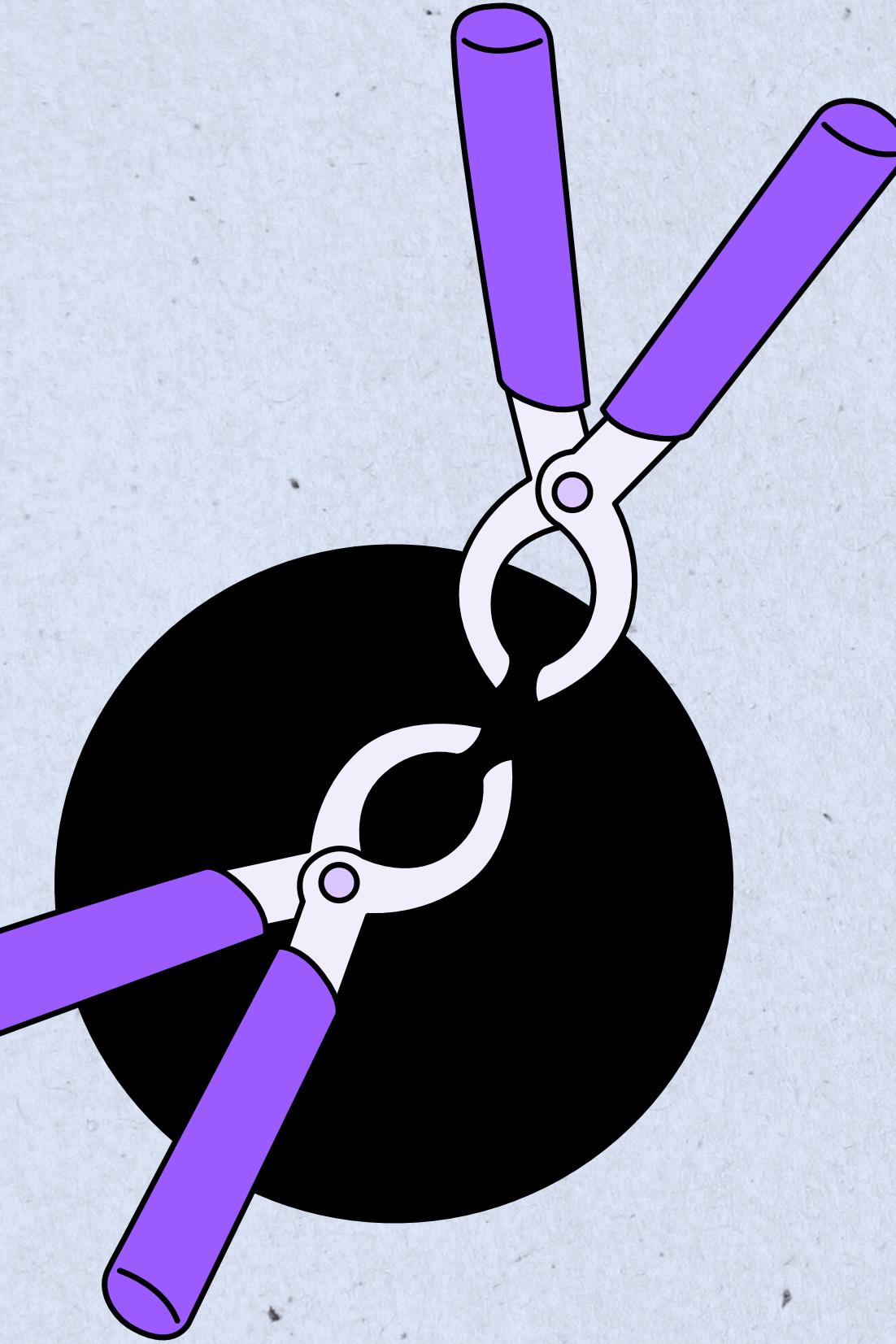
“Se S é um subtipo de T, então objetos do tipo T devem poder ser substituídos por objetos do tipo S sem que o comportamento correto do programa seja alterado.”

SEGUNDA PERGUNTA

Por que isso é importante?

- Consistência: Garante que a hierarquia de classes seja construída de forma que as subclasses possam ser tratadas de forma intercambiável com suas superclasses.
- Reutilização e Extensão: Se o código cliente trabalha com a classe base, ele deve funcionar corretamente ao usar qualquer subtipo, facilitando a extensão do sistema sem quebrar funcionalidades existentes.
- Manutenção: Promove um design mais robusto e facilita a manutenção do código, pois cada classe derivada cumpre o contrato definido pela classe base.





SEM LISKOV

```
public class Retangulo { 5 usages 1 inheritor
    protected int width; 6 usages
    protected int height; 6 usages

    public Retangulo(int width, int height) {
        this.width = width;
        this.height = height;
    }

    public int getArea() { 1 usage
        return width * height;
    }
}
```

```
public class Quadrado extends Retangulo {

    public Quadrado(int side) { no usages
        super(side, side);
    }

    @Override no usages
    public void setWidth(int width) {
        this.width = width;
        this.height = width;
    }

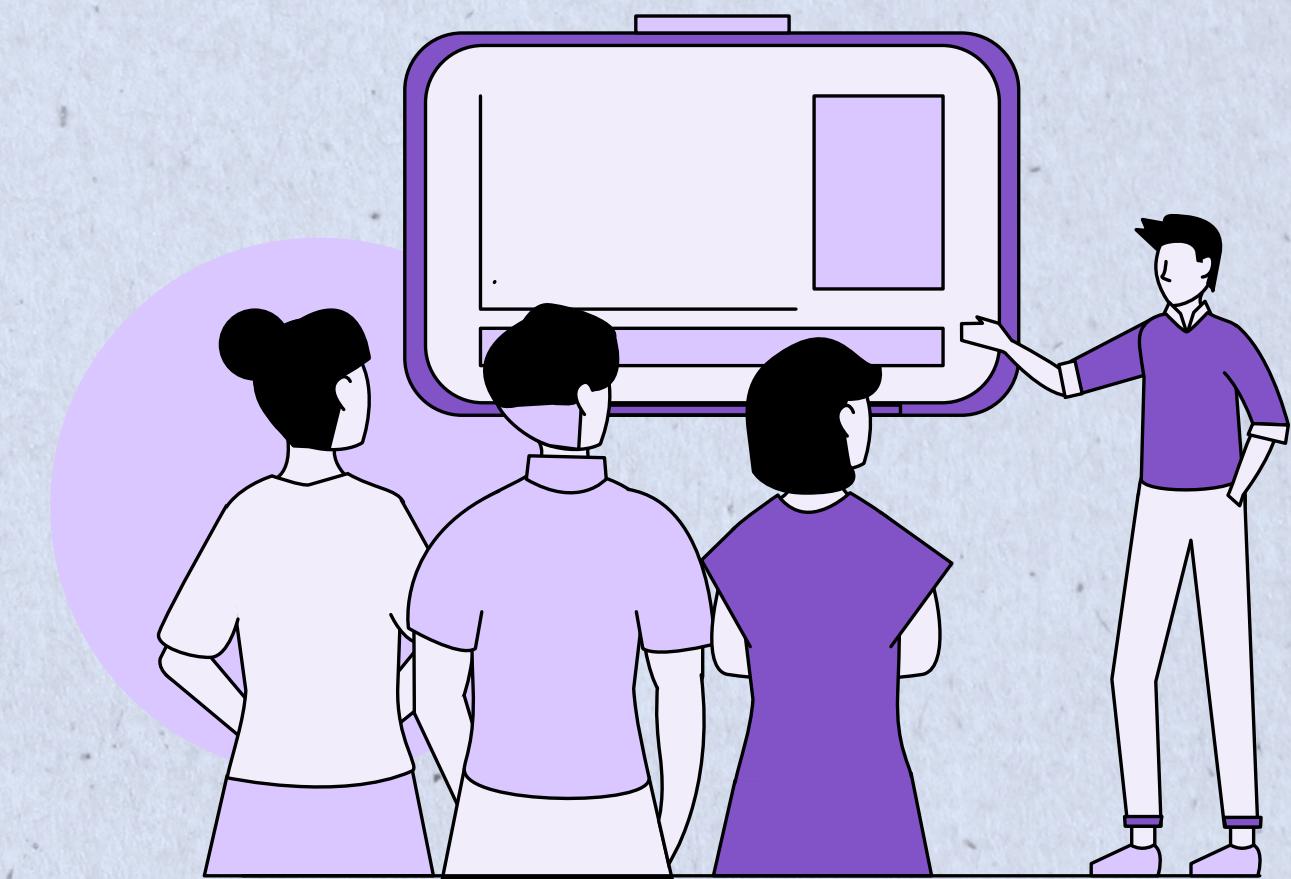
    @Override no usages
    public void setHeight(int height) {
        this.height = height;
        this.width = height;
    }
}
```

Define independentemente

Sobrescreve

Não pode substituir o triângulo sem alterar o comportamento esperado

```
public class Main {  
    public static int testRetanguloArea(Retangulo r) { 2 usages  
        r.setHeight(r.getHeight() + 10);  
        return r.getArea();  
    }  
  
    public static void main(String[] args) {  
  
        Retangulo rect = new Retangulo( width: 5, height: 10);  
        System.out.println("Área do Retângulo após teste: " + testRetanguloArea(rect));  
  
        Retangulo sq = new Quadrado( side: 5);  
        System.out.println("Área do Quadrado após teste: " + testRetanguloArea(sq));  
    }  
}
```



```
Área do Retângulo após teste: 100  
Área do Quadrado após teste: 225
```

COM LISKOV

```
public class Veiculo { 4 usages 1 inheritor
    public void acelerar() { 1 usage 1 override
        System.out.println("Veiculo acelerando");
    }

    public void testarAceleracao(Veiculo v) { now
        v.acelerar();
    }
}
```

```
public class Carro extends Veiculo { 2 usages
    @Override 1 usage
    public void acelerar() {
        System.out.println
            ("Carro acelerando com potência extra");
    }
}
```

Carro respeita o contrato e pode ser usado de forma intercambiável, obedecendo ao comportamento esperado

```
public class Main {  
    public static void main(String[] args) {  
        Veiculo v = new Veiculo();  
        Carro c = new Carro();  
  
        v.testarAceleracao(v); // Imprime: "Veiculo acelerando"  
        v.testarAceleracao(c); // Imprime: "Carro acelerando com potência extra"  
    }  
}
```



REFERENCIAS

O QUE É SOLID: O GUIA COMPLETO
PARA VOCÊ ENTENDER OS 5
PRINCÍPIOS DA POO

SOLID: O QUE É E QUAIS OS 5
PRINCÍPIOS DA PROGRAMAÇÃO
ORIENTADA A OBJETOS (POO)

AGRADECIMENTOS

Enrico Meira
Márcio Torres
Filipe Fogaça
Cauan Ortiz
Beatriz Alamino