LSP - Liskov Substitution Principle

O LSP afirma que objetos de uma classe derivada devem poder substituir objetos de sua classe base sem alterar a funcionalidade do programa.

Benefícios:

- **Consistência e Confiabilidade:** Classes derivadas devem manter o comportamento esperado da classe base.
- **Polimorfismo Seguro:** Permite que substituições sejam feitas sem quebrar o código.

Exemplo de Má Prática: Classes derivadas que alteram ou invalidam funcionalidades da classe base.

LSP - Mau Exemplo

Problema: A classe `Quadrado` tenta herdar `Retangulo`, mas o comportamento de *Largura* e *Altura* é alterado, quebrando a substituição.

```
public class Retangulo
{
    public virtual double Largura { get; set; }
    public virtual double Altura { get; set; }

    public double Area()
    {
        return Largura * Altura;
    }
}

public class Quadrado : Retangulo
{
    public override double Largura
    {
        set { base.Largura = base.Altura = value; }
    }

        public override double Altura
    {
```

```
        set { base.Largura = base.Altura = value; }
    }
}
```

LSP - Bom Exemplo

Solução: `Retangulo` e `Quadrado` agora herdam da classe `Forma` de maneira independente. Assim, `Quadrado` e `Retangulo` podem ser utilizados substituindo `Forma` sem impactar o comportamento.

```
public abstract class Forma
{
    public abstract double Area()
}

public class Retangulo : Forma
{
    public double Largura { get; set; }
    public double Altura { get; set; }

    public override double Area() => Largura * Altura;
}

public class Quadrado : Forma
{
    public double Lado { get; set; }

    public override double Area() => Lado * Lado;
}
```