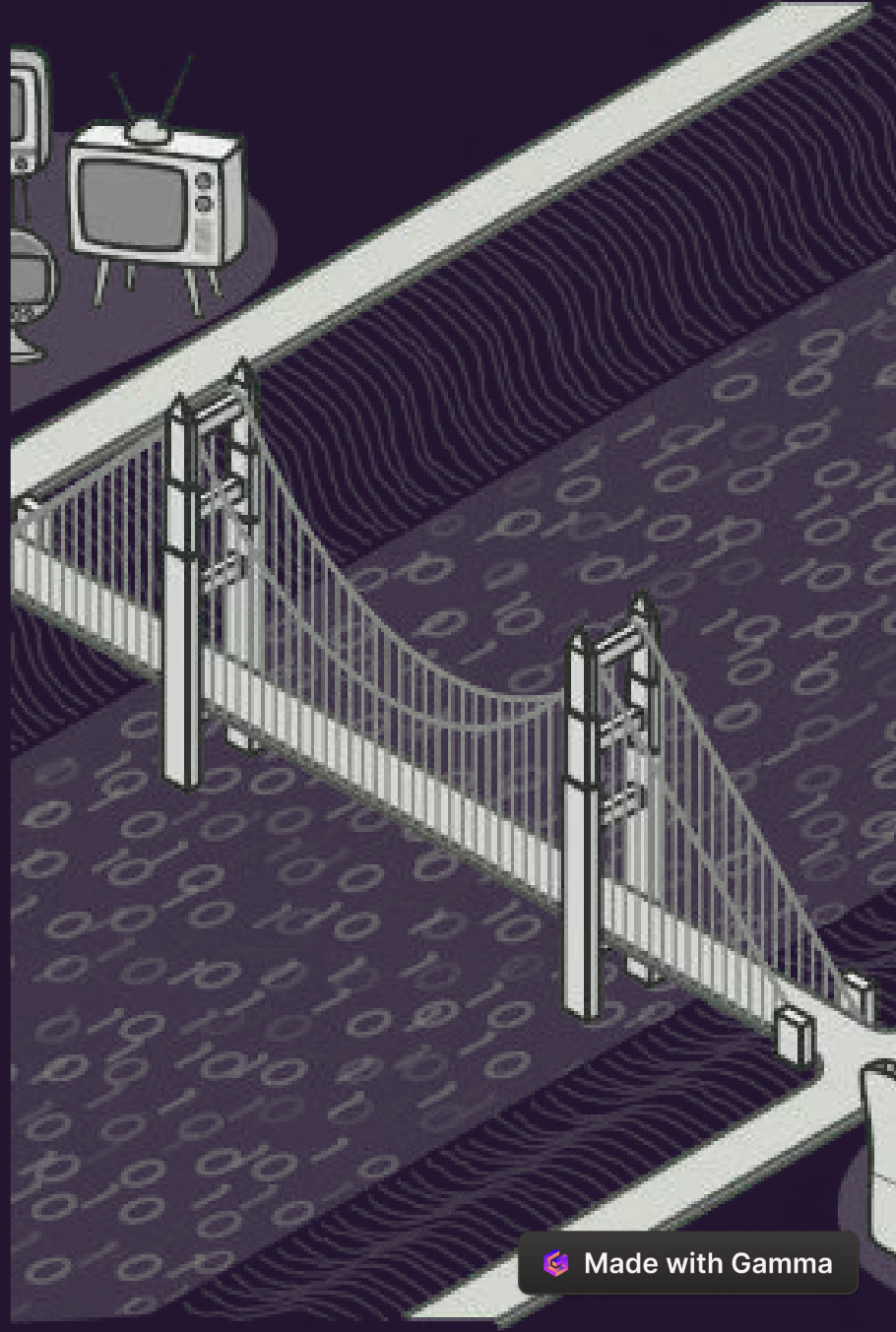
Bridge - Design Pattern

Padrões de Projeto são soluções reutilizáveis para problemas comuns no desenvolvimento de software, visando código mais robusto e flexível.



Bridge

História

O design pattern Bridge foi apresentado pela primeira vez no livro "Design Patterns: Elements of Reusable Object-Oriented Software", escrito pela "Gang of Four" (GoF). O livro foi publicado em 1994 e rapidamente se tornou uma referência fundamental no mundo da programação orientada a objetos.

Motivação

Em sistemas complexos, frequentemente há múltiplas dimensões de hierarquia que precisam ser estendidas sem interferir umas nas outras. O Bridge facilita a manutenção e a extensão do código, promovendo maior flexibilidade e reutilização, especialmente em situações onde as mudanças frequentes são comuns.

Objetivo do Padrão Bridge

Desacoplamento

O padrão Bridge é utilizado para desacoplar uma abstração de sua implementação, permitindo que ambas possam variar independentemente.

Evitar Explosão de Subclasses

O padrão Bridge também ajuda a evitar a explosão de subclasses, que ocorre quando há múltiplas dimensões de variação.

Estrutura do Padrão Bridge

Abstração

Define a interface abstrata para as classes que representam a abstração. Mantém uma referência a um objeto da hierarquia de implementação e delega a ele todas as chamadas.

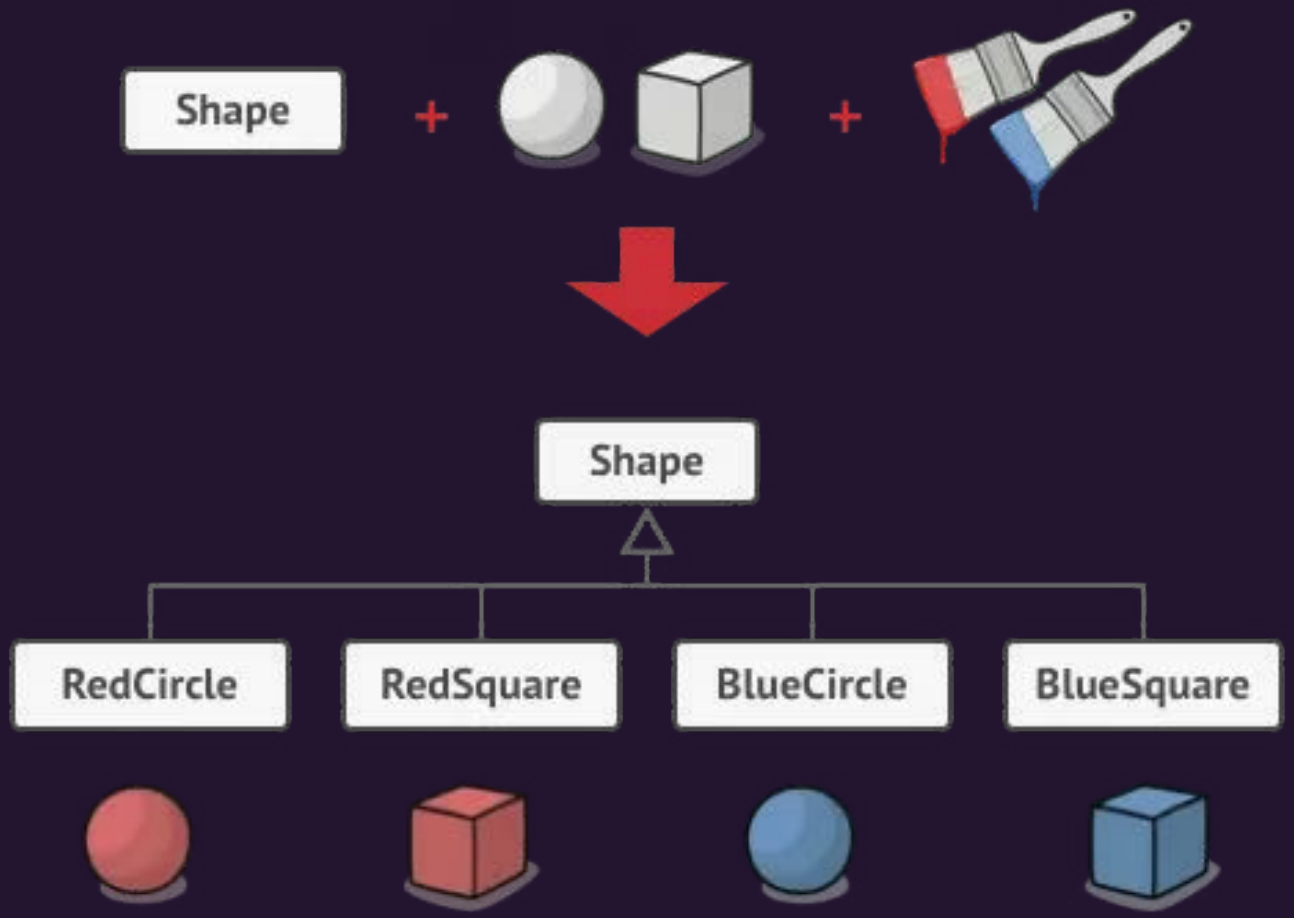
Implementação

Define a interface para as classes de implementação. Esta interface não precisa corresponder exatamente à interface da abstração e permite variações independentes das abstrações.

Refinamento Abstrato e Implementação Concreta

O Refinamento Abstrato estende a interface da abstração e pode incluir métodos adicionais que utilizam a implementação. A Implementação Concreta fornece implementações específicas para as interfaces de implementação.

Exemplo Prático: Problema



Exemplo Prático: Solução

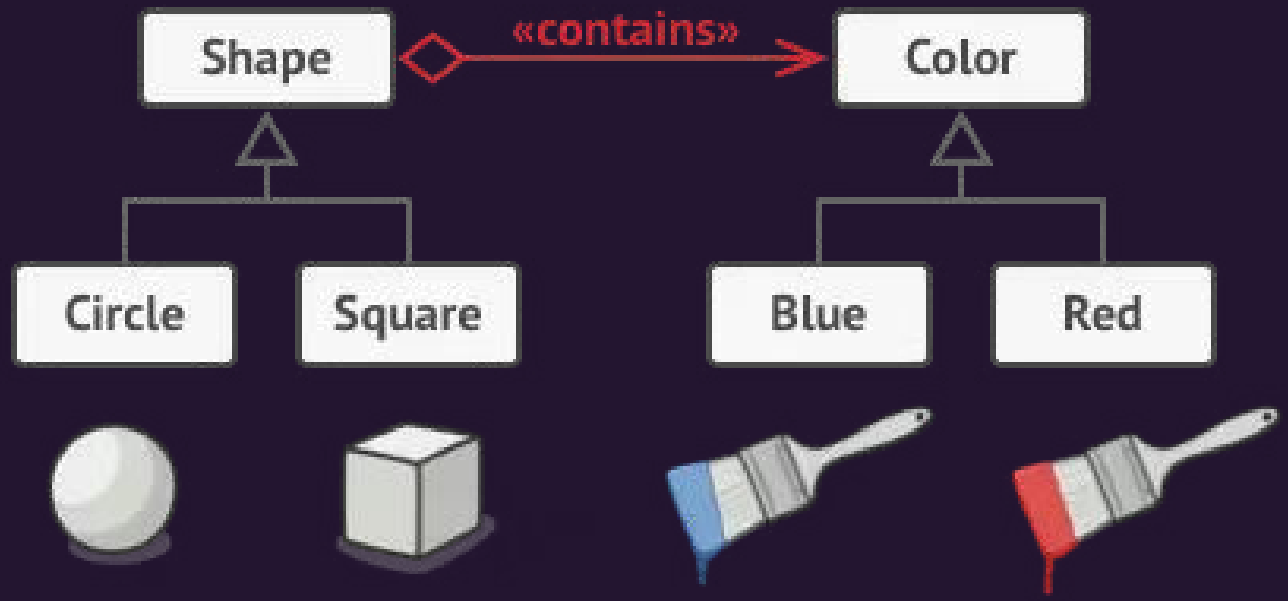
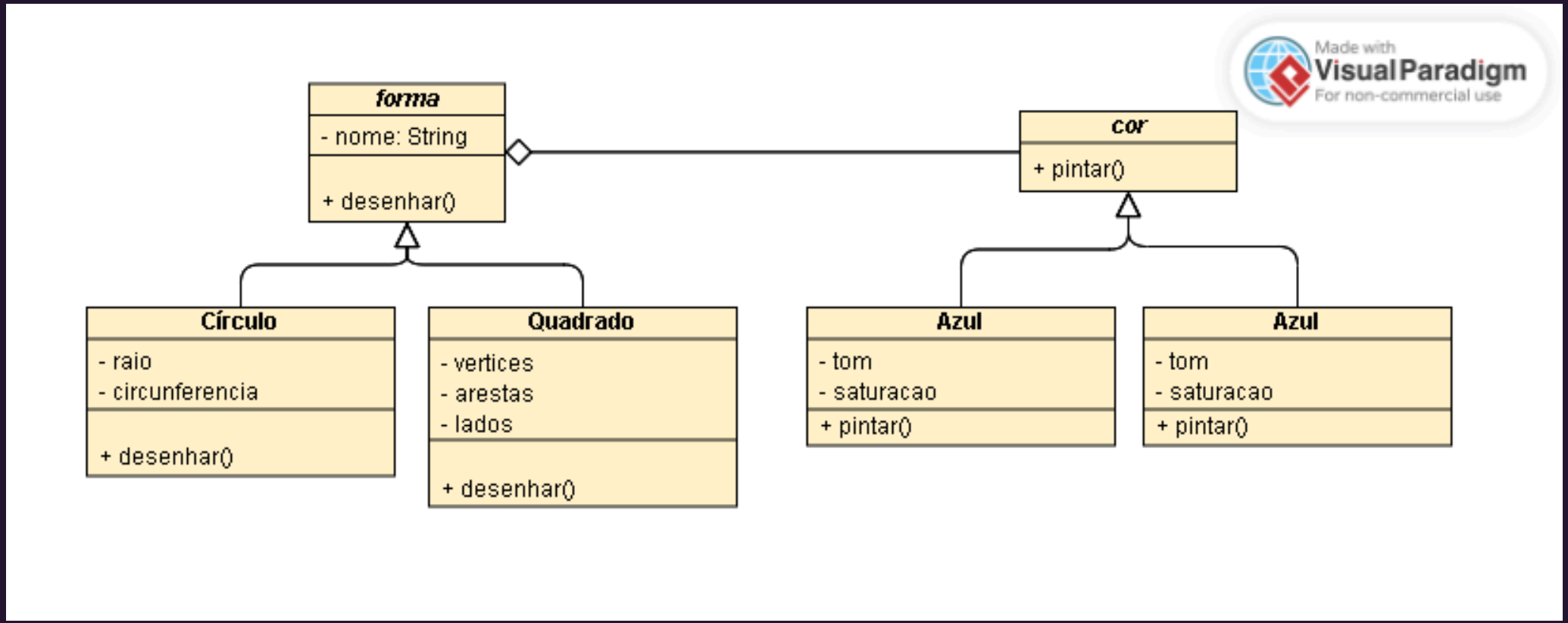


Diagrama de Classe



Vantagens do Padrão Bridge

1

Manutenção

Reduz o impacto das mudanças ao dividir em partes menores e mais gerenciáveis.

2

Flexibilidade

Facilita a criação de novas funcionalidades combinando diferentes abstrações e implementações.

3

OPC

Entidades abertas a extensão e fechadas para modificações.

4

SRP

Classes com responsabilidades únicas.

"Uma classe deve ter somente uma razão para mudar".