BRIDGE

PADRÃO DE PROJETOS ESTRUTURAL

Gustavo H. Pinto - 5º DSM

MOTIVAÇÃO

SIMPLIFICAÇÃO DO CÓDIGO

O uso do padrão também facilita e simplifica o desenvolvimento e a manutenção de muitas hierarquias de classes independentes

DESACOPLAMENTO

O padrão Bridge atua especialmente na separação entre **abstração** e **implementação**, promovendo um maior desacoplamento entre os objetos de um projeto

SOLID

Bridge promove a adesão de várias práticas do SOLID no projeto, como o princípio da responsabilidade única e o princípio da inversão de dependência

O PADRÃO

TIPO

Bridge é um padrão de projeto estrutural, especificando como objetos devem ser criados e organizados em estruturas maiores durante seu ciclo de vida

PROPOSTA

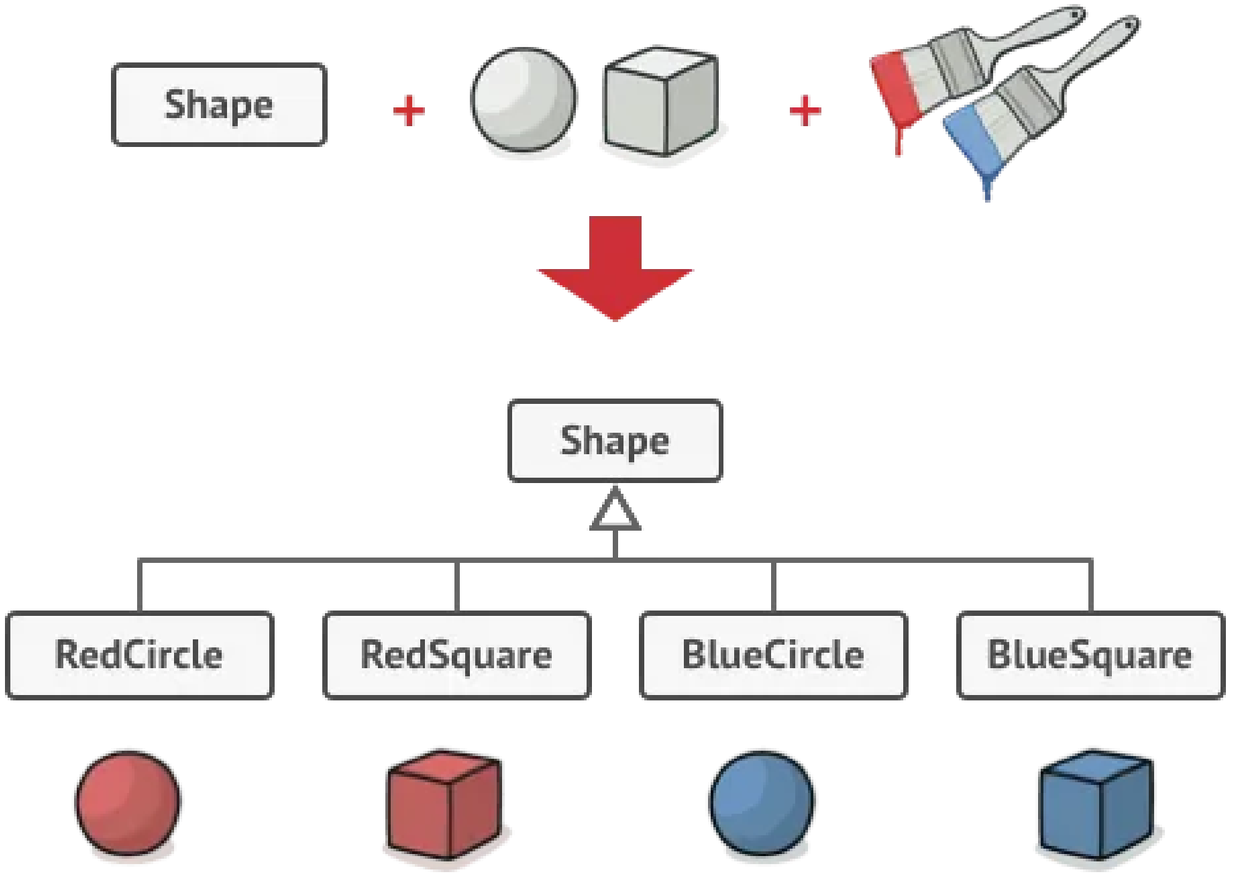
O padrão **Bridge** propõe o desenvolvimento de uma interface que atuará como **ponte** entre dois ou mais objetos, em conjunto com a utilização da **composição** para a ligação entre essa interface **ponte** e a implementação

LIGAÇÕES

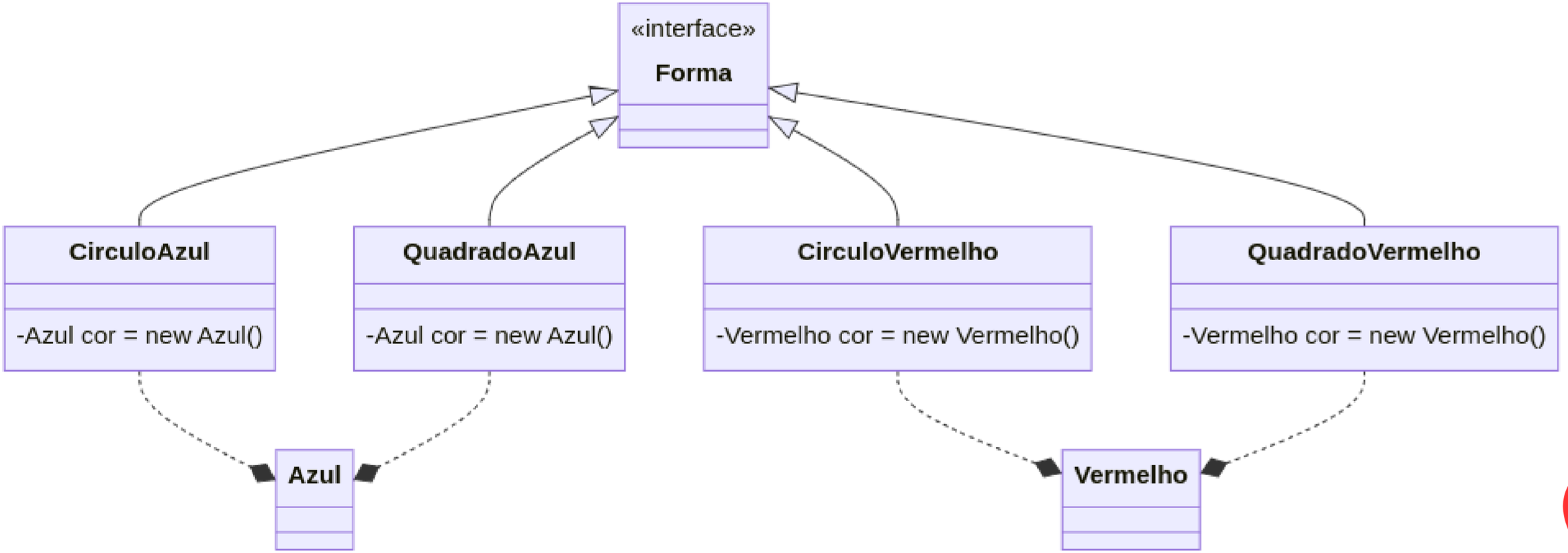
Trabalha em conjunto com a técnica de **injeção de dependência** para abstrair totalmente a implementação utilizada da interface requisitada em um objeto ou método

ARQUITETURA SEM USO DE BRIDGE...

Em uma arquitetura sem uso de Bridge e com alto uso de herança temos o desenvolvimento de vários objetos que devem sempre ser atualizados e criados em lote quando qualquer nova forma (*shape*) ou cor for adicionada

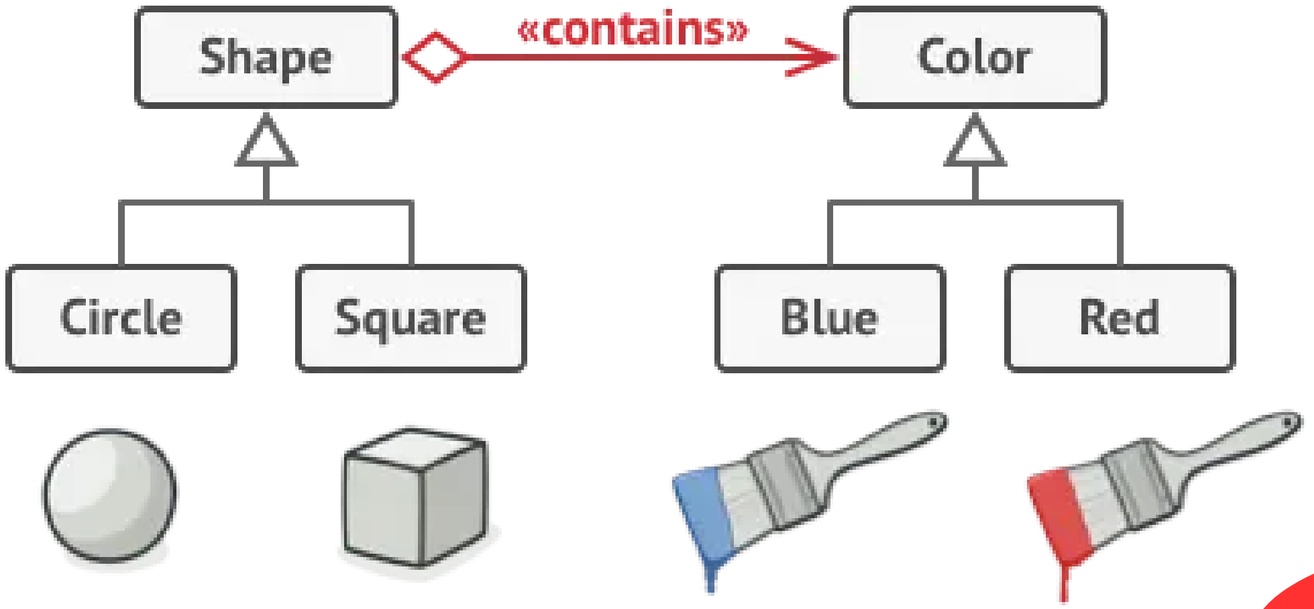


... E A UML

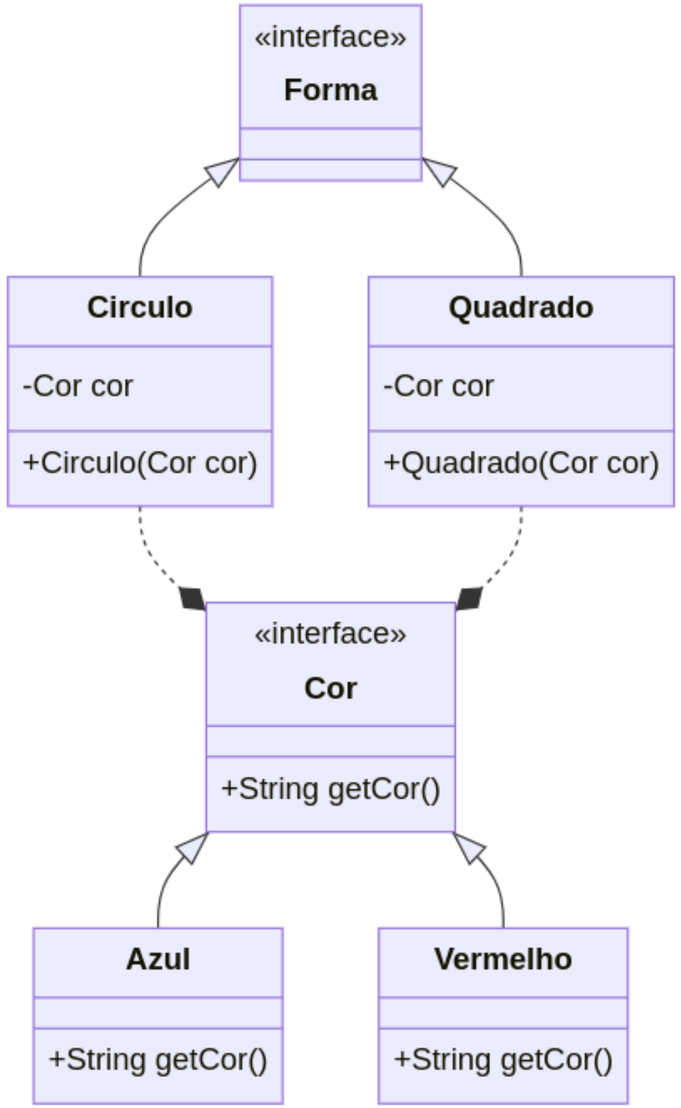

«interface»
Forma
CirculoAzul
-Azul cor = new Azul()
QuadradoAzul
-Azul cor = new Azul()
CirculoVermelho
-Vermelho cor = new Vermelho()
QuadradoVermelho
-Vermelho cor = new Vermelho()
Azul
Vermelho

ARQUITETURA APLICANDO BRIDGE...

Já quando aplicamos o padrão Bridge, temos o desenvolvimento de menos objetos, com as especializações das formas (*shape*) criadas dinamicamente a partir da injeção de cores



... E A UML


«interface»
Forma
Circulo
-Cor cor
+Circulo(Cor cor)
Quadrado
-Cor cor
+Quadrado(Cor cor)
«interface»
Cor
+String getCor()
Azul
+String getCor()
Vermelho
+String getCor()

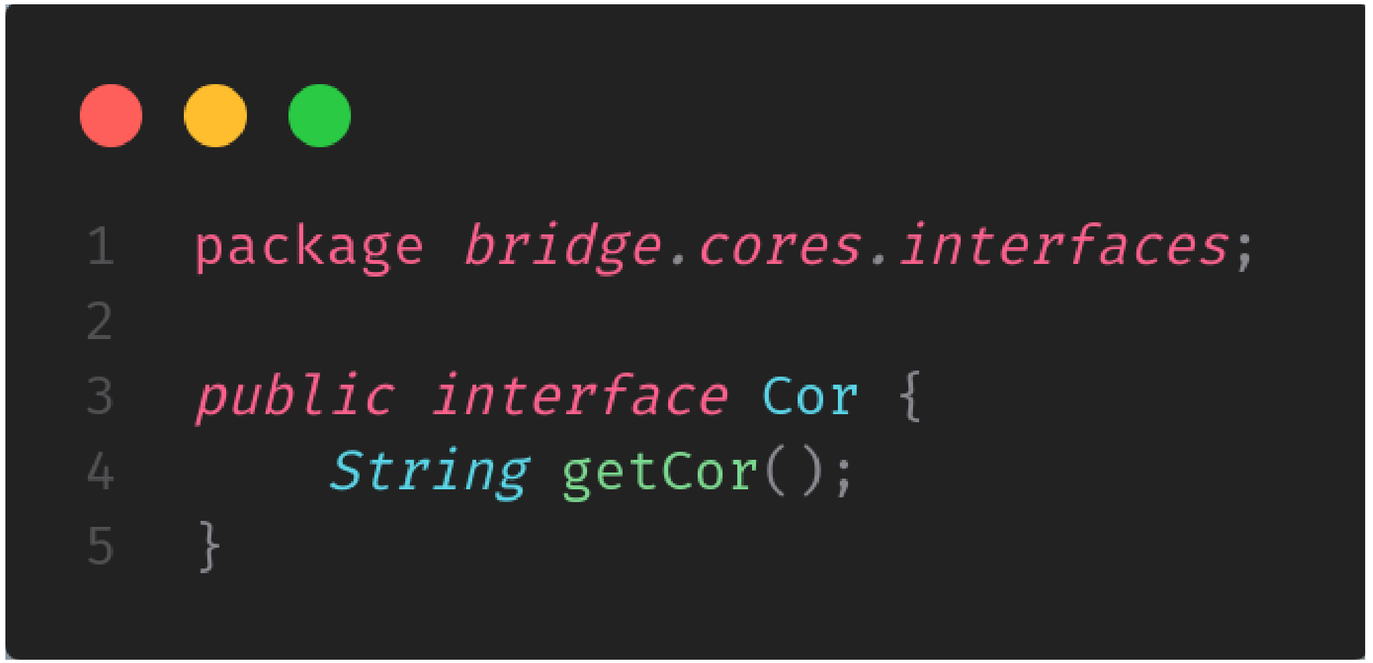
CÓDIGO - MAIN

```java
package bridge;

import bridge.cores.Azul;
import bridge.cores.Vermelho;
import bridge.formas.Circulo;
import bridge.formas.Quadrado;
import bridge.formas.interfaces.Forma;

public class App {
    public static void main(String[] args) {
        Forma quadradoAzul = new Quadrado(new Azul());
        Forma circuloAzul = new Circulo(new Azul());
        Forma quadradoVermelho = new Quadrado(new Vermelho());
        Forma circuloVermelho = new Circulo(new Vermelho());

        System.out.println(quadradoAzul);
        System.out.println(circuloAzul);
        System.out.println(quadradoVermelho);
        System.out.println(circuloVermelho);
    }
}
```

CÓDIGO - QUADRADO

```java
package bridge.formas;

import bridge.cores.interfaces.Cor;
import bridge.formas.interfaces.Forma;

public class Quadrado implements Forma {
    private Cor cor;

    public Quadrado(Cor cor) {
        this.cor = cor;
    }

    @Override
    public Cor getCor() {
        return this.cor;
    }

    @Override
    public void setCor(Cor cor) {
        this.cor = cor;
    }

    @Override
    public String toString() {
        return String.format("Quadrado[cor = %s]", this.cor.getClass().getSimpleName());
    }
}
```

CÓDIGO - COR



```java
package bridge.cores.interfaces;

public interface Cor {
    String getCor();
}
```

CÓDIGO - AZUL

```java
package bridge.cores;

import bridge.cores.interfaces.Cor;

public class Azul implements Cor {
    @Override
    public String getCor() {
        return "azul";
    }
}
```

CÓDIGO - SAÍDA

```
[130] % gradle clean run
Starting a Gradle Daemon, 2 incompatible and 1 stopped Daemons could not be reused, use --status for details

> Task :app:run
Quadrado[cor = Azul]
Circulo[cor = Azul]
Quadrado[cor = Vermelho]
Circulo[cor = Vermelho]

BUILD SUCCESSFUL in 8s
4 actionable tasks: 4 executed
```

OBRIGADO PELA ATENÇÃO!