

× × × ×

Design Patterns

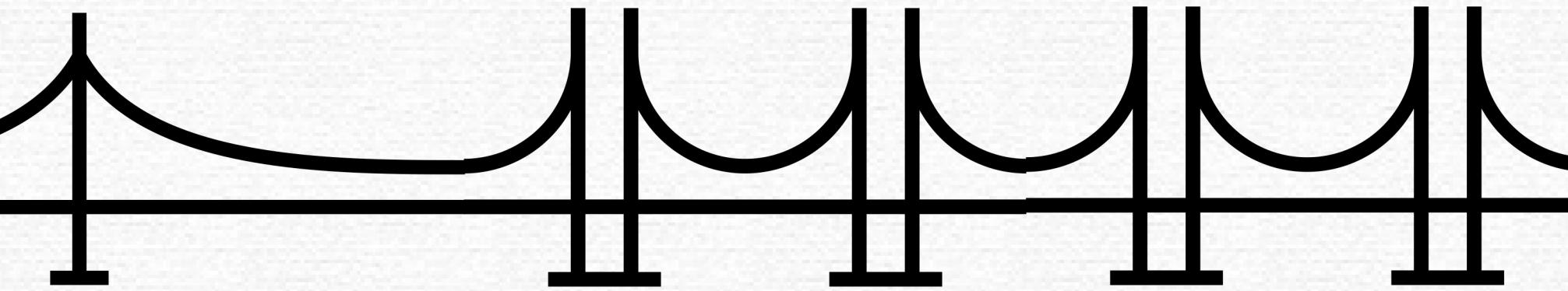
PADRÃO BRIDGE



× × × ×

**O QUE É O PADRÃO
ESTRUTURAL BRIDGE?**

BRIDGE OU PONTE

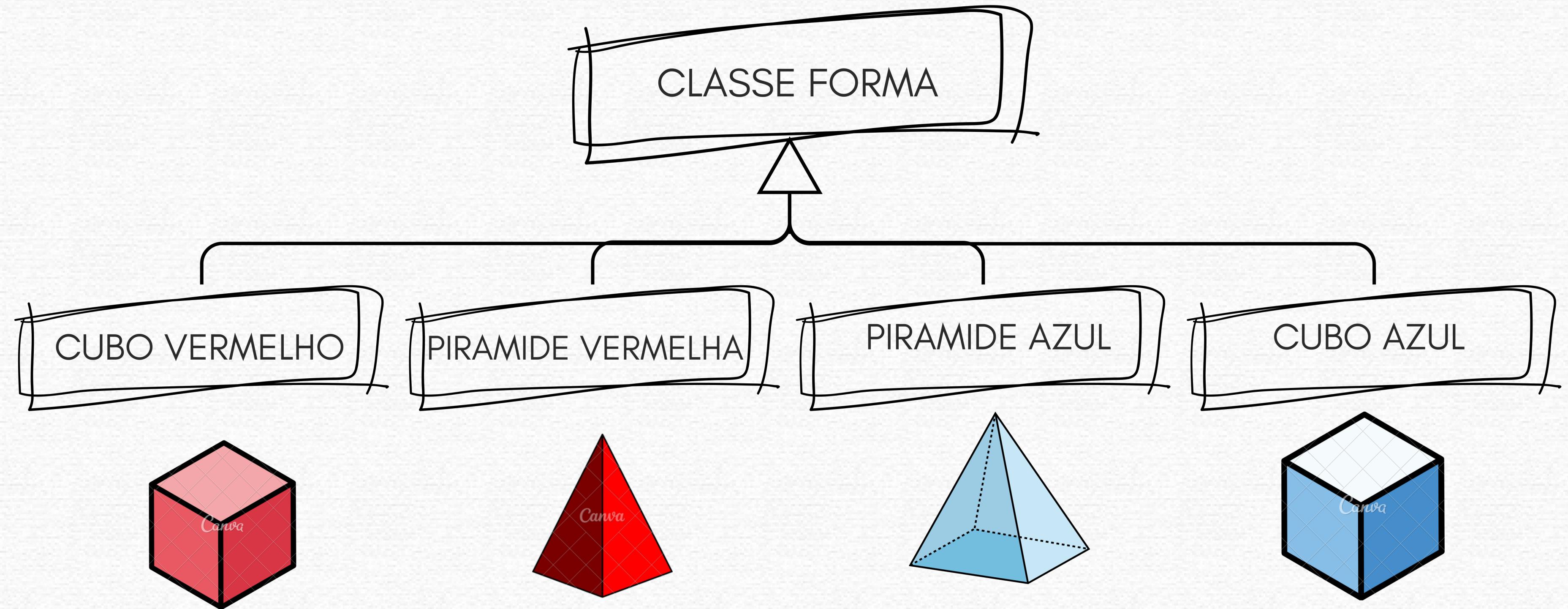


Bridge ou Ponte é um padrão de projeto estrutural, ou seja, um padrão que se preocupa em organizar a estrutura das classes e os relacionamentos entre as mesmas e objetos

Permite que você divida uma classe grande ou um conjunto de classes intimamente ligadas em duas hierarquias separadas

COMO ASSIM?

PROBLEMA !

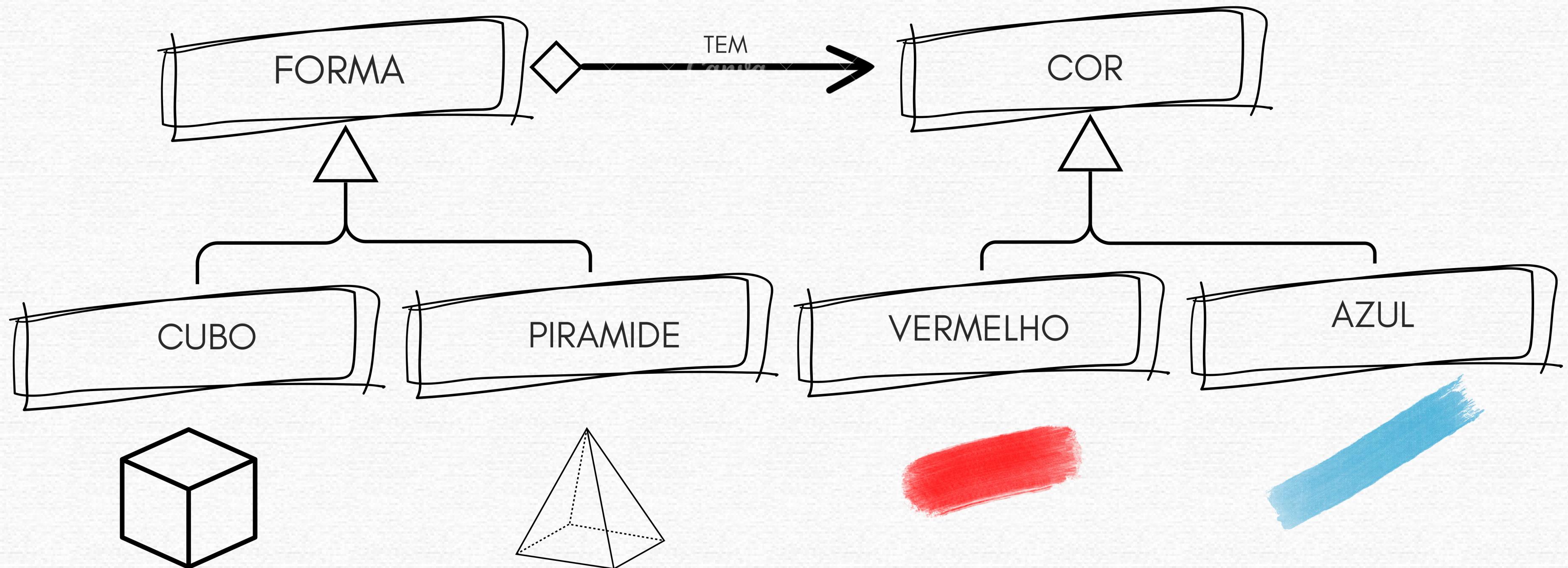


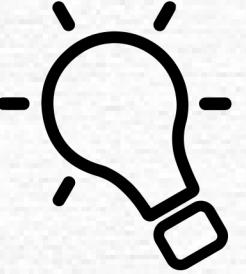
PROBLEMA !

Adicionar novos tipos de forma e cores à hierarquia irá fazê-la crescer exponencialmente. Por exemplo, para adicionar uma forma de esfera vai ser preciso introduzir duas subclasses, uma para cada cor. E depois disso, adicionando uma nova cor será necessário três subclasses, uma para cada tipo de forma. Quanto mais longe vamos, pior isso fica.

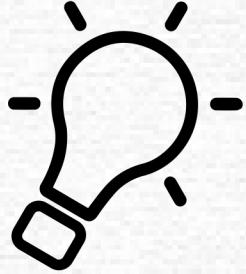
× × × ×

SOLUÇÃO





SOLUÇÃO



Esse problema ocorre porque estamos tentando estender as classes de forma em duas dimensões diferentes. Isso é um problema muito comum com herança de classe.

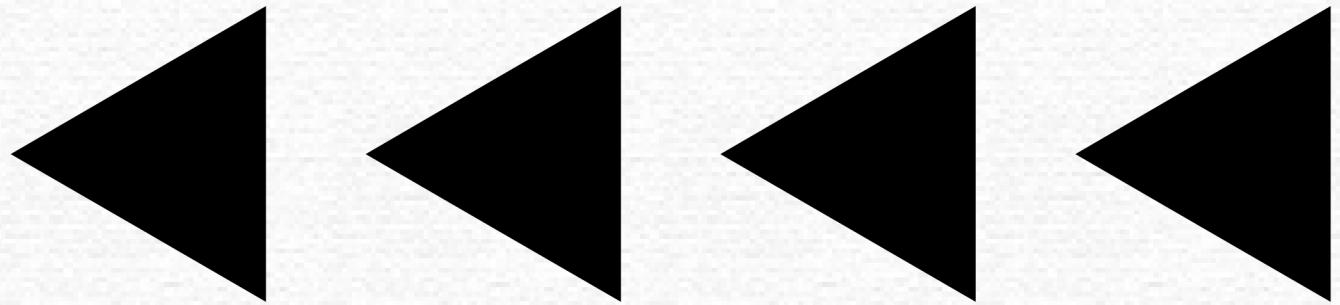
O padrão Bridge tenta resolver esse problema ao trocar de herança para composição do objeto. Isso significa que você extrai uma das dimensões em uma hierarquia de classe separada, para que as classes originais referenciem um objeto da nova hierarquia.

A classe Forma ganha um campo de referência apontando para um dos objetos de cor. Aquela referência vai agir como uma ponte entre as classes Forma e Cor. De agora em diante, para adicionar novas cores não será necessário mudar a hierarquia da forma e vice versa.

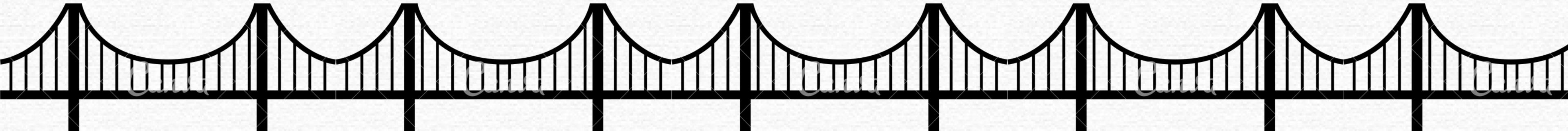


ABSTRAÇÃO E IMPLEMENTAÇÃO PARA A BRIDGE PATTERN

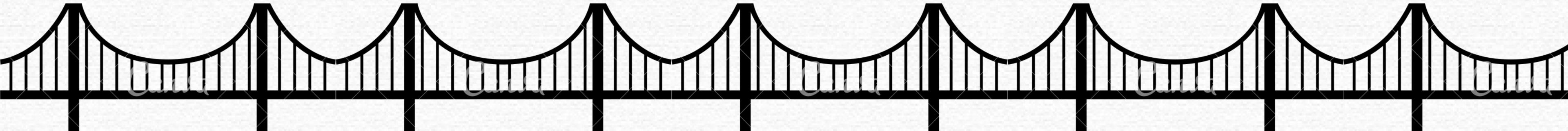
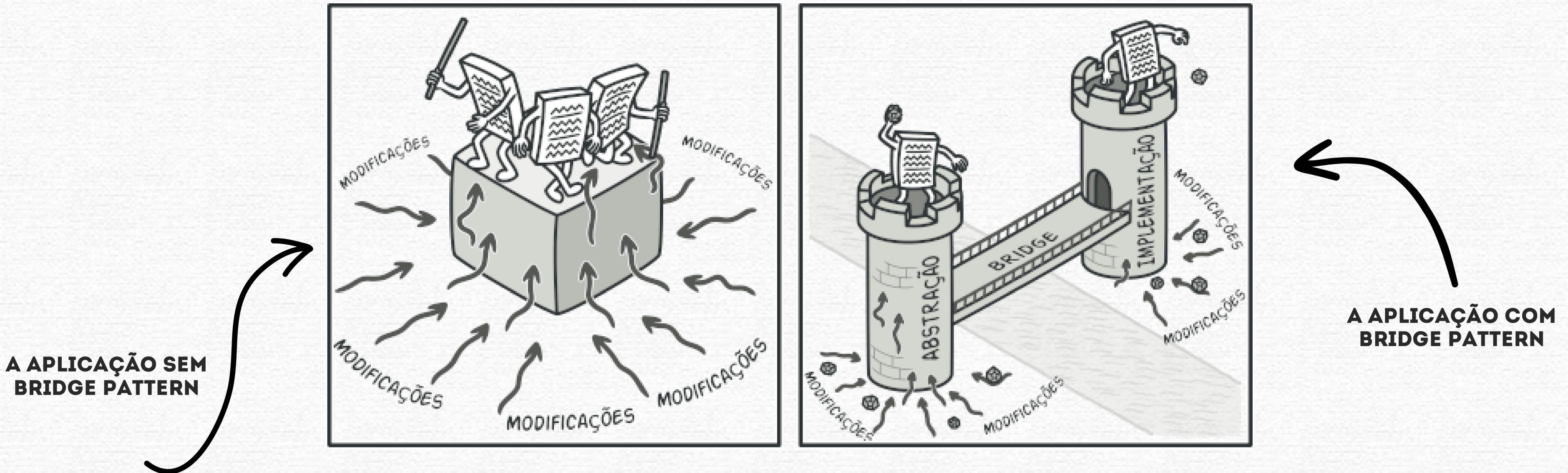
Segundo o livro “Padrões de Projeto – Soluções Reutilizáveis de Software Orientado a Objetos”, os termos Abstração e Implementação fazem parte da definição do Bridge.



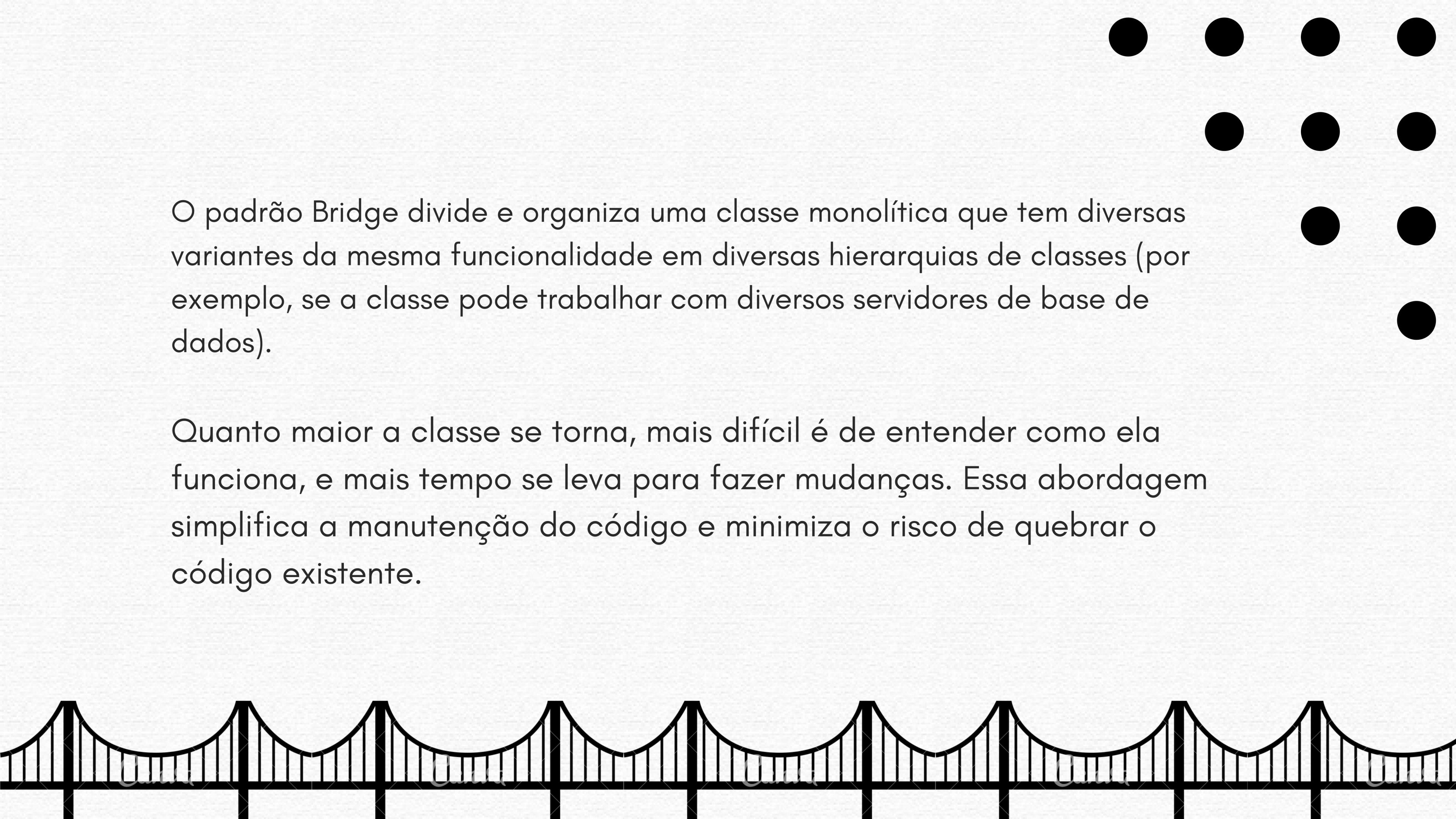
Abstração (também chamado de interface) é uma camada de controle de alto nível para alguma entidade. Essa camada não deve fazer nenhum tipo de trabalho por conta própria. Ela deve delegar o trabalho para a camada de **implementação** (também chamada de plataforma).



A **abstração** pode ser representada por uma interface gráfica de usuário (GUI), e a **implementação** pode ser o código subjacente do sistema operacional (API) a qual a camada GUI chama em resposta às interações do usuário.



APLICABILIDADE



O padrão Bridge divide e organiza uma classe monolítica que tem diversas variantes da mesma funcionalidade em diversas hierarquias de classes (por exemplo, se a classe pode trabalhar com diversos servidores de base de dados).

Quanto maior a classe se torna, mais difícil é de entender como ela funciona, e mais tempo se leva para fazer mudanças. Essa abordagem simplifica a manutenção do código e minimiza o risco de quebrar o código existente.

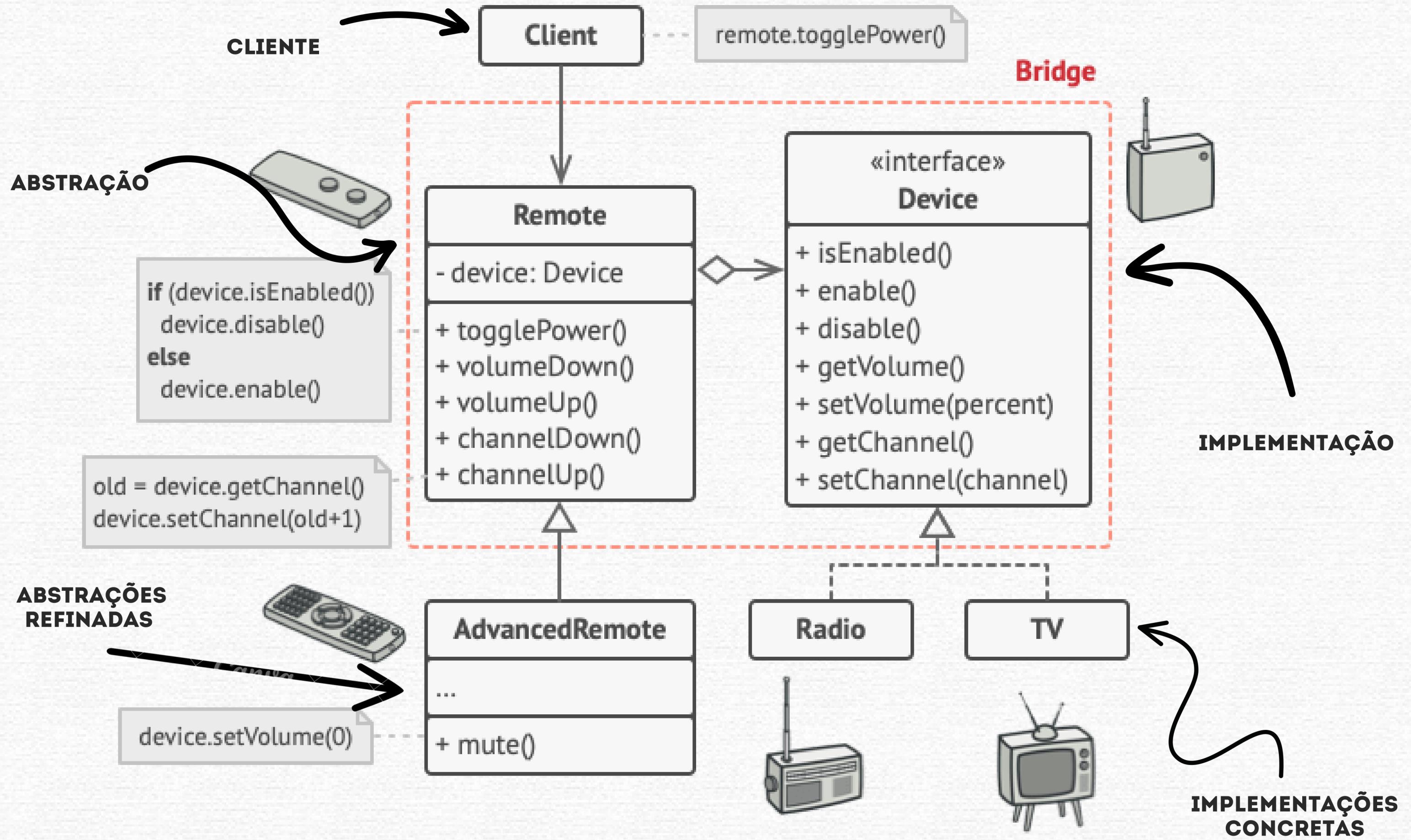
**COMO FAZER A
IMPLEMENTAÇÃO?**

ELEMENTOS

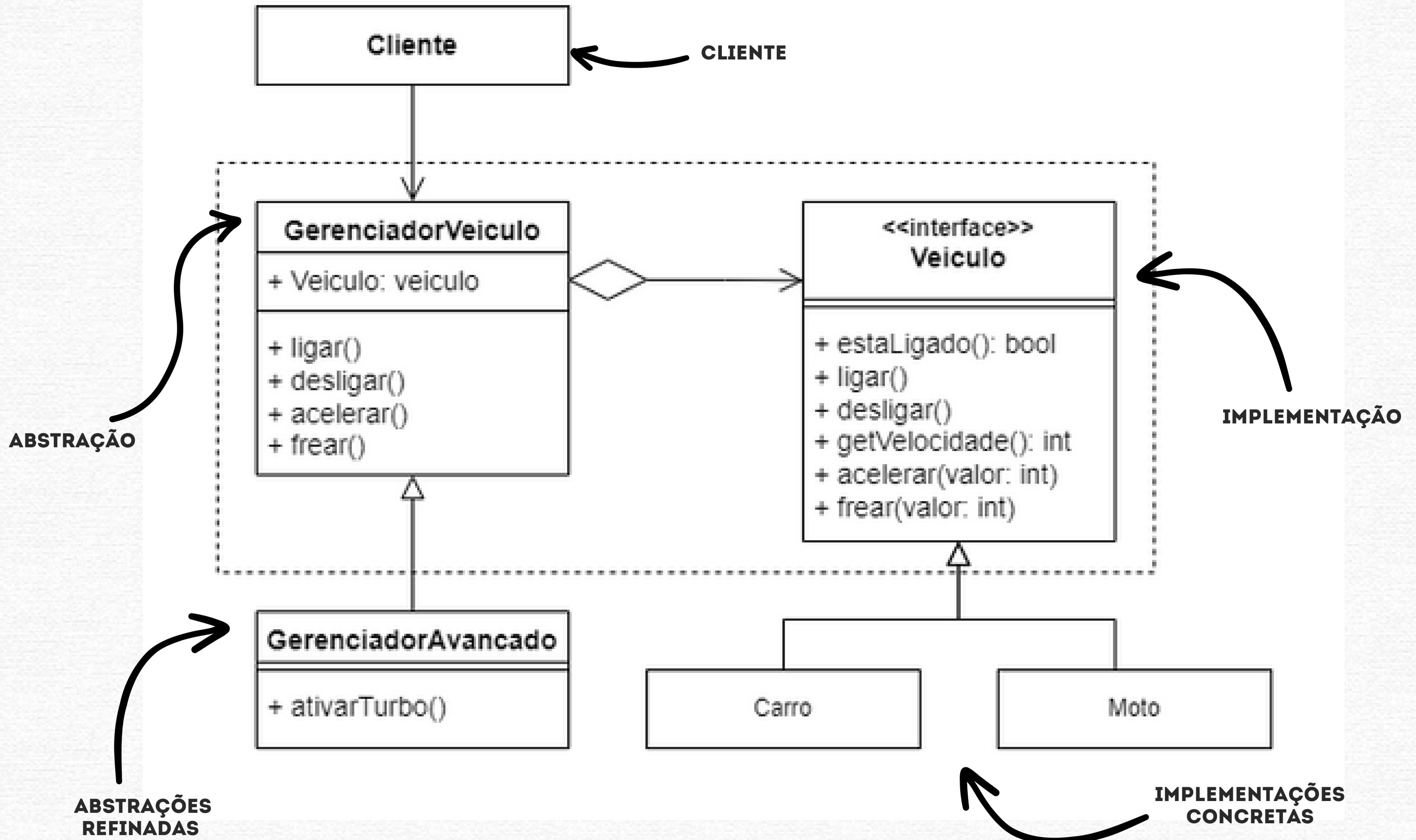


- A **Abstração** fornece a lógica de controle de alto nível. Ela depende do objeto de implementação para fazer o verdadeiro trabalho de baixo nível.
- A **Implementação** declara a interface que é comum para todas as implementações concretas. Uma abstração só pode se comunicar com um objeto de implementação através de métodos que são declarados aqui.
- A **abstração** pode listar os mesmos métodos que a implementação, mas geralmente a **abstração** declara alguns comportamentos complexos que dependem de uma ampla variedade de operações primitivas declaradas pela implementação.
- **Implementações Concretas** contém código plataforma-específicos.
- **Abstrações Refinadas** fornecem variantes para controle da lógica. Como seu superior, trabalham com diferentes implementações através da interface geral de implementação.
- Geralmente o **Cliente** está apenas interessado em trabalhar com a abstração. Contudo, é trabalho do cliente ligar o objeto de abstração com um dos objetos de implementação

ESTRUTURA

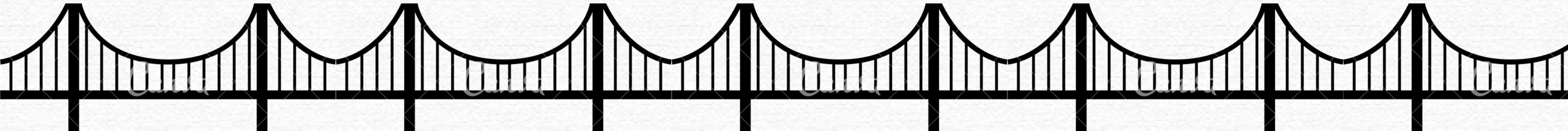


ESTRUTURA



PASSO A PASSO PARA IMPLEMENTAÇÃO

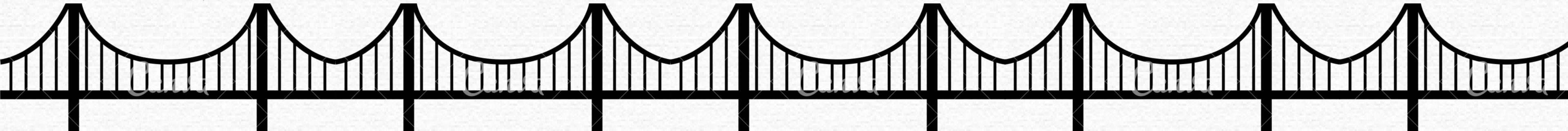
1. Identifique as dimensões ortogonais em suas classes.
2. Veja quais operações o cliente precisa e defina-as na classe abstração base.
3. Determine as operações disponíveis em todas as plataformas. Declare aquelas que a abstração precisa na interface geral de implementação.
4. Crie classes concretas de implementação para todas as plataformas de seu domínio, mas certifique-se que todas elas sigam a interface de implementação.



PASSO A PASSO PARA IMPLEMENTAÇÃO

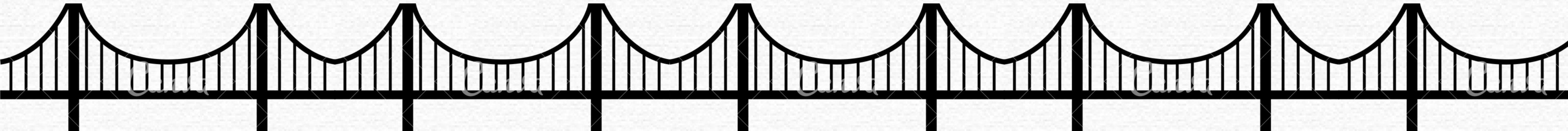
4.

5. Dentro da classe de abstração, adicione um campo de referência para o tipo de implementação. A abstração delega a maior parte do trabalho para o objeto de implementação que foi referenciado neste campo.
6. Se você tem diversas variantes de lógica de alto nível, crie abstrações refinadas para cada variante estendendo a classe de abstração básica.
7. O código cliente deve passar um objeto de implementação para o construtor da abstração para associar um com o outro. Após isso, o cliente pode esquecer sobre a implementação e trabalhar apenas com o objeto de abstração.



PRÓS E CONTRAS

- ✓ Criar classes e aplicações independentes de plataforma.
- ✓ O código cliente trabalha com abstrações em alto nível. Ele não fica exposto a detalhes de plataforma.
- ✓ Princípio aberto/fechado. Você pode introduzir novas abstrações e implementações independentemente uma das outras.
- ✓ Princípio de responsabilidade única. Você pode focar na lógica de alto nível na abstração e em detalhes de plataforma na implementação.
- ✗ Você pode tornar o código mais complicado ao aplicar o padrão em uma classe altamente coesa.



CÓDIGOS

FONTES

<https://refactoring.guru/design-patterns/bridge>

<https://refactoring.guru/design-patterns/bridge/csharp/example>

<http://www.labies.uff.br/padroesdr/questions/1/patterns/bridge>

[https://github.com/eleev/swift-design-patterns/blob/main/
Common%20Design%20Patterns/Structural/Bridge/Bridge.md](https://github.com/eleev/swift-design-patterns/blob/main/Common%20Design%20Patterns/Structural/Bridge/Bridge.md)

https://www.macoratti.net/vb_pd1.html

