Módulo III
Padrões GOF: Bridge

Professores
Eduardo Bezerra – edubezerra@gmail.com
Ismael H F Santos – ismael@tecgraf.puc-rio.br

Ementa

- Padrões GOF
  - Bridge

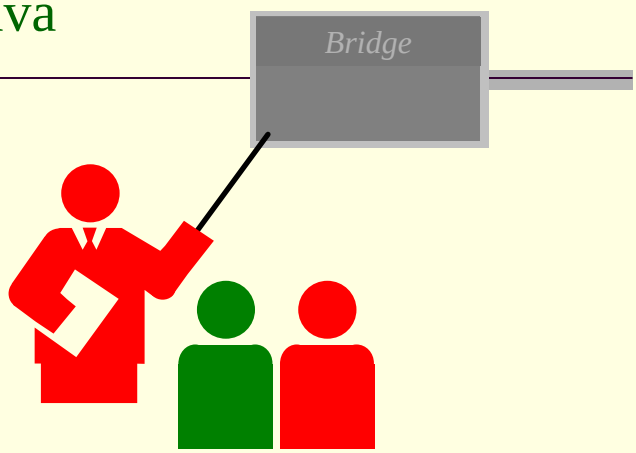
Bibliografia

- *Eric Gamma, et ali, Padrões de Projeto, Ed Bookman*
- *Martin Fowler, Analysis Patterns - Reusable Object Models, Addison-Wesley,1997*
- *Martin Fowler, Refatoração - Aperfeiçoando o projeto de código existente, Ed Bookman*

Livros

- **Core Java 2**, Cay S. Horstmann, Gary Cornell
  - Volume 1 (Fundamentos)
  - Volume 2 (Características Avançadas)
- **Java: Como Programar**, Deitel & Deitel
- **Thinking in Patterns with JAVA**, Bruce Eckel
  - **Gratuito**. http://www.mindview.net/Books/TIJ/

POO-Java

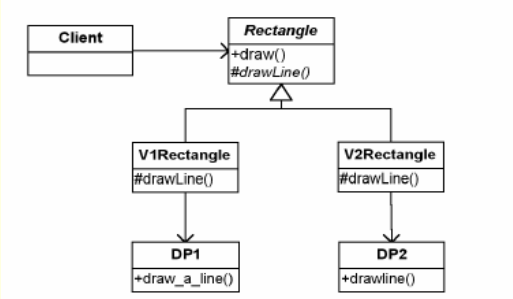


Bridge

- **Intenção:** desacoplar uma abstração de sua implementação de tal forma que a implementação possa ser facilmente trocada.
- **Solução:** encapsular os detalhes de implementação em um objeto que é um componente da abstração

Bridge

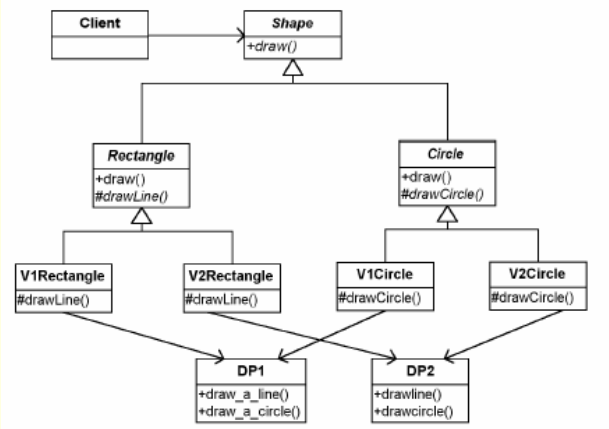
- Considere a construção de um módulo para desenhar figuras.
- Considere que há duas *classes externas* de desenho a serem utilizadas: DP1 e DP2
- Primeira versão: “somente retângulos devem ser desenhados”.
- Solução:



Bridge

- Suponha agora, o seguintes novos requisitos:
  - “As classes externas agora desenham círculos. Portanto, nosso módulo de desenho deve também ter a possibilidade de desenhar círculos”.
  - “Além disso, o cliente do nosso módulo de desenho não precisa saber a diferença entre um retângulo e um círculo.”
- Solução:
  - Criamos uma classe abstrata Shape, e fazemos com que ela seja superclasse tanto de Rectangle quanto de Circle.
  - O cliente agora se comunica com objetos Shape.
- Sendo assim, temos uma nova solução, apresentada a seguir...

Bridge



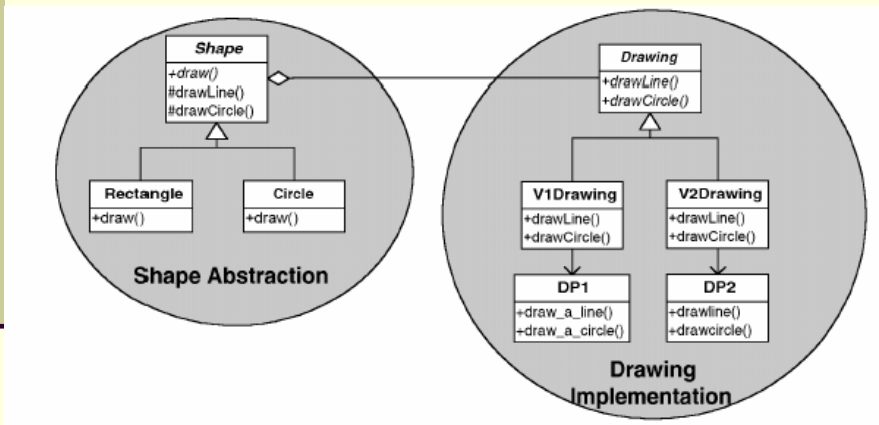
Bridge

- Infelizmente, a solução da figura anterior introduz alguns problemas:
  - O que acontece se preciso dar suporte a um novo programa de desenho?
  - E se um novo tipo de Figura (Shape) tiver que ser adicionado?
- O que aconteceria com a complexidade de manutenção neste módulo quando a quantidade de tipos de figura e de módulos externos de desenho chegasse à cada das dezenas?
- Resposta: *explosão de classes*.

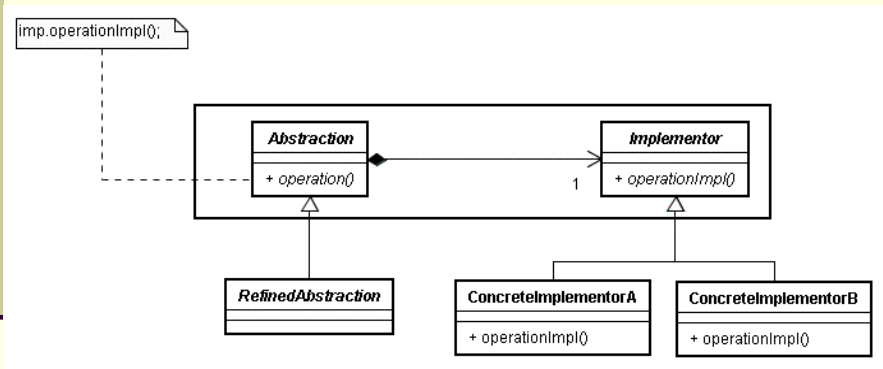
Bridge

- A explosão de classes surge porque, nesta solução, a abstração (ou seja, os tipos de Shape) e a implementação (os programas de desenho) estão fortemente acoplados.
  - i.e., cada tipo de figura (abstração) deve saber que tipo de módulo externo (implementação) ele deve utilizar.
- Precisamos de um modo de separar (desacoplar) as variações na abstração das variações na implementação, de tal forma que o número de classes cresça somente linearmente.
  - Esta é exatamente a intenção (objetivo) do padrão **Bridge**.

Bridge



Bridge (estrutura)



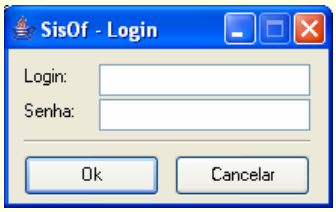
Bridge (exemplo)

- Java Swing possui diversos “look and feels”.
- Objetos JFrame, JButton, etc. possuem um componente interno (peer) que é construído para com um look and feel particular.
- No entanto, clientes (programadores) somente precisam interagir com JFrame, JButton, etc.

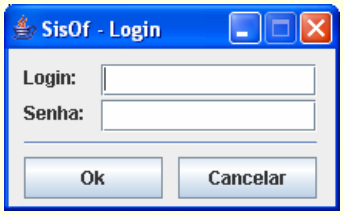
Bridge (exemplo)

- O método UIManager.setLookAndFeel define a fábrica concreta para “produtos” do Windows.

WindowsLookAndFeel

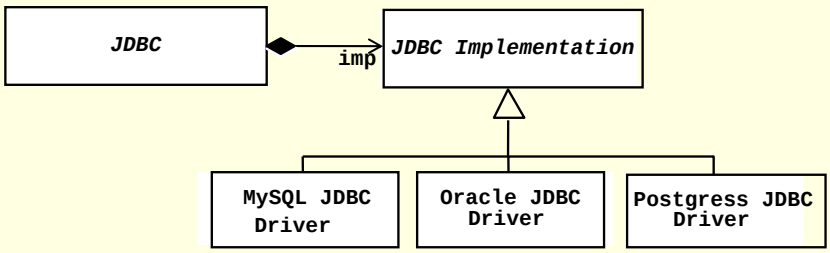


DefaultLookAndFeel



Bridge (exemplo)

- Exemplo: usando o padrão Bridge para abstrair o driver específico de banco de dados.



Bridge (aplicabilidade)

- Aplicabilidade
  - O padrão Bridge é útil quando se tem uma abstração que tem diferentes implementações.
  - Este padrão permite que a abstração e a sua implementação variem independentemente.