

INF011 – Padrões de Projeto

00 - Apresentação

Sandro Santos Andrade
sandroandrade@ifba.edu.br

Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Departamento de Tecnologia Eletro-Eletrônica
Graduação Tecnológica em Análise e Desenvolvimento de Sistemas



Roteiro

- Contexto e Motivação
- Padrões de Projeto
- Princípios de Projeto OO
- Estudo de Caso: *Bridge*
- Padrões Arquiteturais e Idiomas
- Conclusão

Contexto e Motivação

- Uma rápida estória:
 - Você é um professor com a responsabilidade de informar aos estudantes, ao final da sua aula, o local onde ocorrerá a aula seguinte
 - Uma primeira implementação:
 1. Obtenha a lista de estudantes da classe
 2. Para cada estudante, faça o seguinte:
 1. Encontre a próxima aula que ele assistirá
 2. Localize a sala onde ocorrerá esta aula
 3. Encontre um caminho da sua sala para a sala localizada
 4. Informe ao estudante o caminho a ser seguido

Contexto e Motivação

- Uma rápida estória:
 - Você é um professor com a responsabilidade de informar aos estudantes, ao final da sua aula, o local onde ocorrerá a aula seguinte
 - Uma primeira implementação:
 - Programa de Controle {
 1. Obtenha a lista de estudantes da classe
 2. Para cada estudante, faça o seguinte:
 1. Encontre a próxima aula que ele assistirá
 2. Localize a sala onde ocorrerá esta aula
 3. Encontre um caminho da sua sala para a sala localizada
 4. Informe ao estudante o caminho a ser seguido

Contexto e Motivação

- Uma rápida estória:
 - Você é um professor com a responsabilidade de informar aos estudantes, ao final da sua aula, o local onde ocorrerá a aula seguinte
 - Uma outra implementação:
 1. Publique no mural os caminhos da sua sala para todas as outras
 2. Informe para toda a turma: “O mural contém os caminhos para as outras salas. Utilizem-o e dirijam-se para as suas próximas aulas”

Contexto e Motivação

- O que mudou entre as soluções ?
 - Na primeira, você presta atenção a uma série de detalhes e é responsável por tudo
 - Na segunda, instruções básicas são fornecidas e espera-se que cada pessoa saiba fazer o seu trabalho

Contexto e Motivação

- O que mudou entre as soluções ?
 - Na primeira, você presta atenção a uma série de detalhes e é responsável por tudo
 - Na segunda, instruções básicas são fornecidas e espera-se que cada pessoa saiba fazer o seu trabalho

DESVIO DE RESPONSABILIDADES

Contexto e Motivação

- O que mudou entre as soluções ?
 - Na primeira, você presta atenção a uma série de detalhes e é responsável por tudo
 - Na segunda, instruções básicas são fornecidas e espera-se que cada pessoa saiba fazer o seu trabalho

DESVIO DE RESPONSABILIDADES

ESTUDANTES RESPONSÁVEIS PELO SEU PRÓPRIO COMPORTAMENTO

Contexto e Motivação

- Porque esta reorganização de responsabilidades é importante ?
 - Suponha que alunos bolsistas tenham que preencher um formulário de avaliação da aula antes de se dirigir à aula seguinte
 - Na primeira solução o programa de controle deve ser modificado de modo a distinguir os tipos de alunos e dar instruções específicas a cada tipo
 - Na segunda solução o comportamento dos alunos bolsistas seria modificado e o programa de controle continuaria dizendo “Vá para a sua próxima aula”

DESIGN FOR CHANGE

Contexto e Motivação

- Porque esta reorganização de responsabilidades é importante ?
 - Suponha que alunos bolsistas tenham que preencher um formulário de avaliação da aula antes de se dirigir à aula seguinte
 - Na primeira solução o programa de controle deve ser modificado de modo a distinguir os tipos de instruções específicas a cada tipo de operação
 - Na segunda solução o comportamento do programa de controle seria modificado e o programa de controle continuaria dizendo “Vá para a sua próxima aula”

DESIGN FOR CHANGE

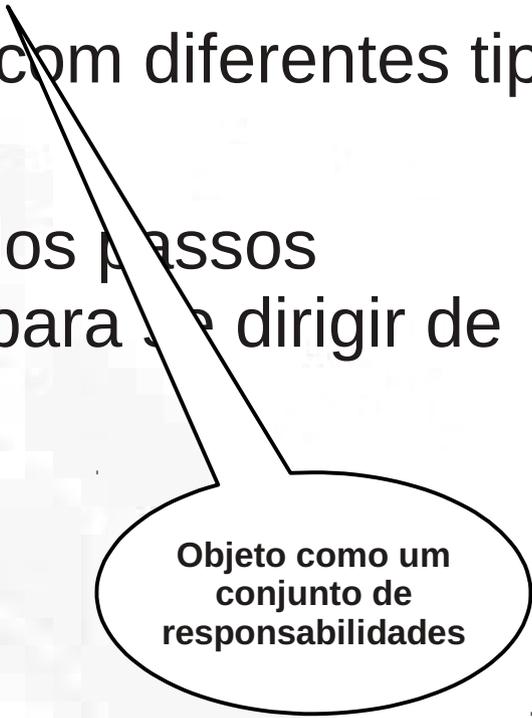
Quero saber mais !
*Lehman, Metrics and
Laws of Software
Evolution*

Contexto e Motivação

- Para que isso funcione é necessário que:
 - Os alunos conheçam o seu tipo (bolsista / não bolsista) e sejam responsáveis pelo seu próprio comportamento
 - O programa de controle se comunique com diferentes tipos de alunos de maneira uniforme
 - O programa de controle não dependa dos passos particulares que estudantes executam para se dirigir de uma sala à outra

Contexto e Motivação

- Para que isso funcione é necessário que:
 - Os alunos conheçam o seu tipo (bolsista / não bolsista) e sejam responsáveis pelo seu próprio comportamento
 - O programa de controle se comunique com diferentes tipos de alunos de maneira uniforme
 - O programa de controle não dependa dos passos particulares que estudantes executam para se dirigir de uma sala à outra



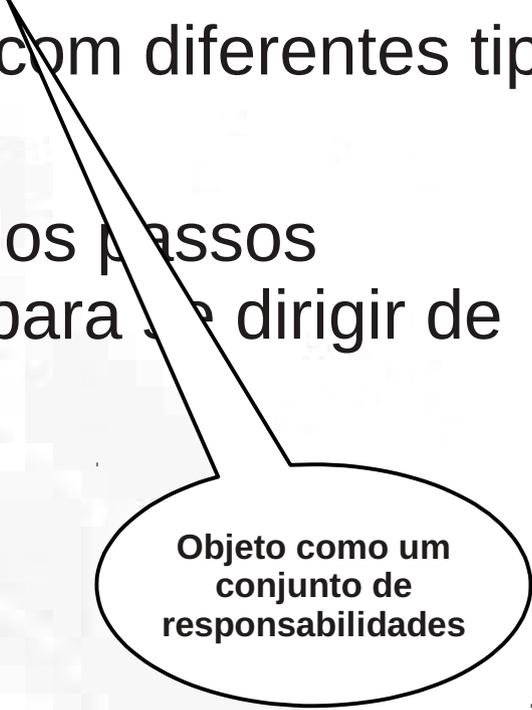
Objeto como um conjunto de responsabilidades

Contexto e Motivação

- Para que isso funcione é necessário que:
 - Os alunos conheçam o seu tipo (bolsista / não bolsista) e sejam responsáveis pelo seu próprio comportamento
 - O programa de controle se comunique com diferentes tipos de alunos de maneira uniforme
 - O programa de controle não dependa dos passos particulares que estudantes executam para se dirigir de uma sala à outra



Interfaces como um mecanismo para representar conceitos



Objeto como um conjunto de responsabilidades

Contexto e Motivação

- Para que isso funcione é necessário que:
 - Os alunos conheçam o seu tipo (bolsista / não bolsista) e sejam responsáveis pelo seu próprio comportamento
 - O programa de controle se comunique com diferentes tipos de alunos de maneira uniforme
 - O programa de controle não dependa dos passos particulares que estudantes executam para se dirigir de uma sala à outra



Encapsulamento
para prover vários
tipos de
ocultamentos



Interfaces como
um mecanismo
para representar
conceitos



Objeto como um
conjunto de
responsabilidades

Contexto e Motivação

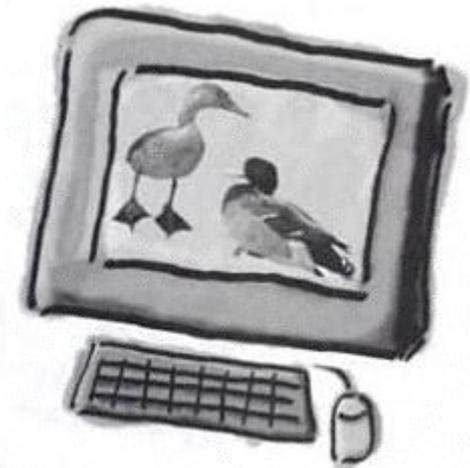
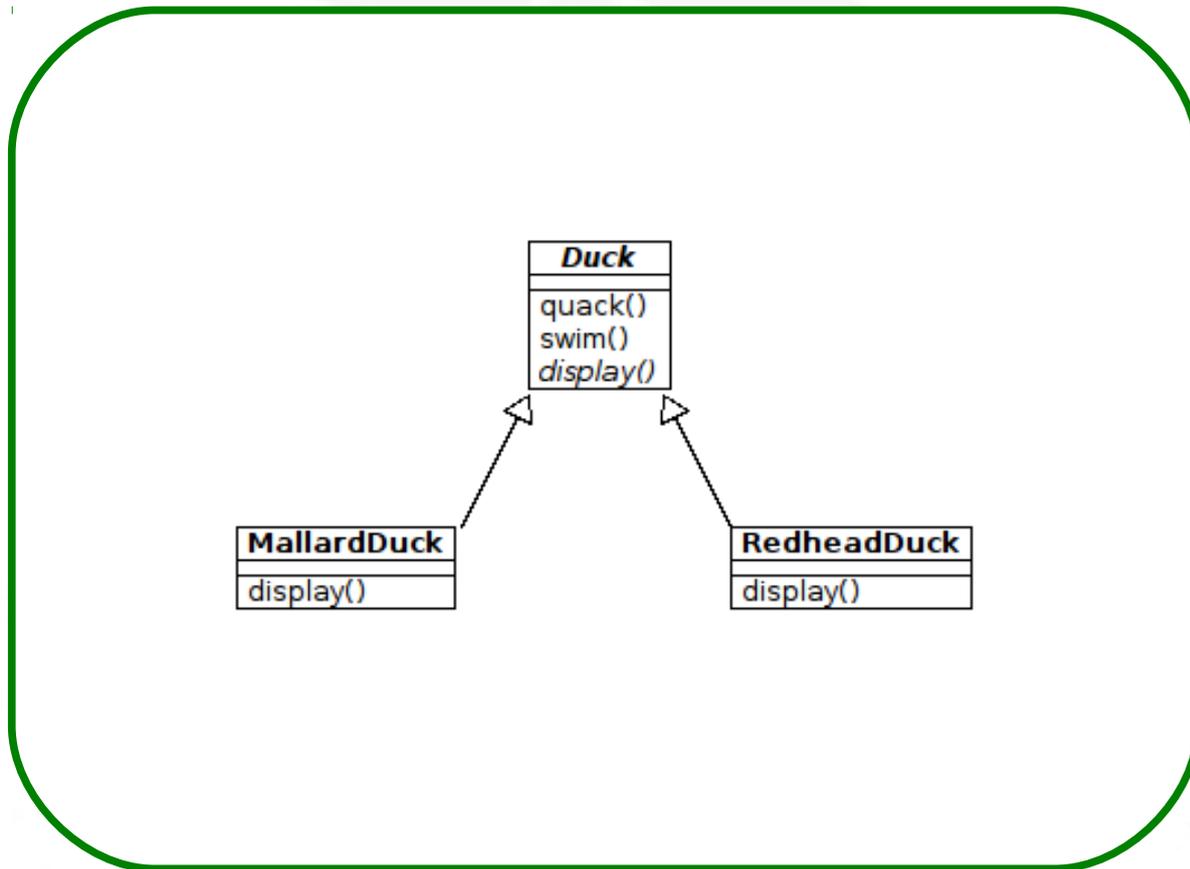
- Perspectivas no processo de desenvolvimento:

PERSPECTIVA	DESCRIÇÃO
CONCEITUAL	Representa os conceitos no domínio estudado Qual a minha responsabilidade ?
ESPECIFICAÇÃO	Foco nas interfaces, não na implementação Como eu sou utilizado ?
IMPLEMENTAÇÃO	Foco na implementação Como eu cumpro minhas responsabilidades ?

Fowler, Martin. UML Distilled

Contexto e Motivação

- Porque precisamos de padrões de projeto ?
 - Exemplo: jogo de patos



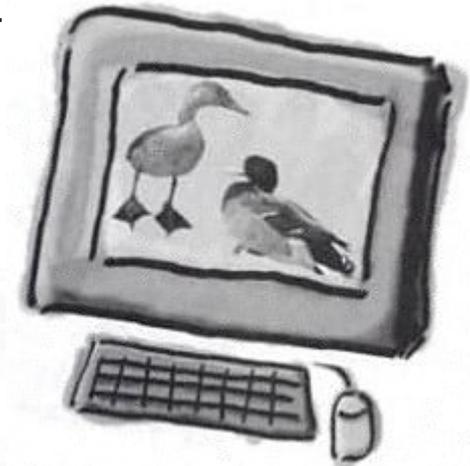
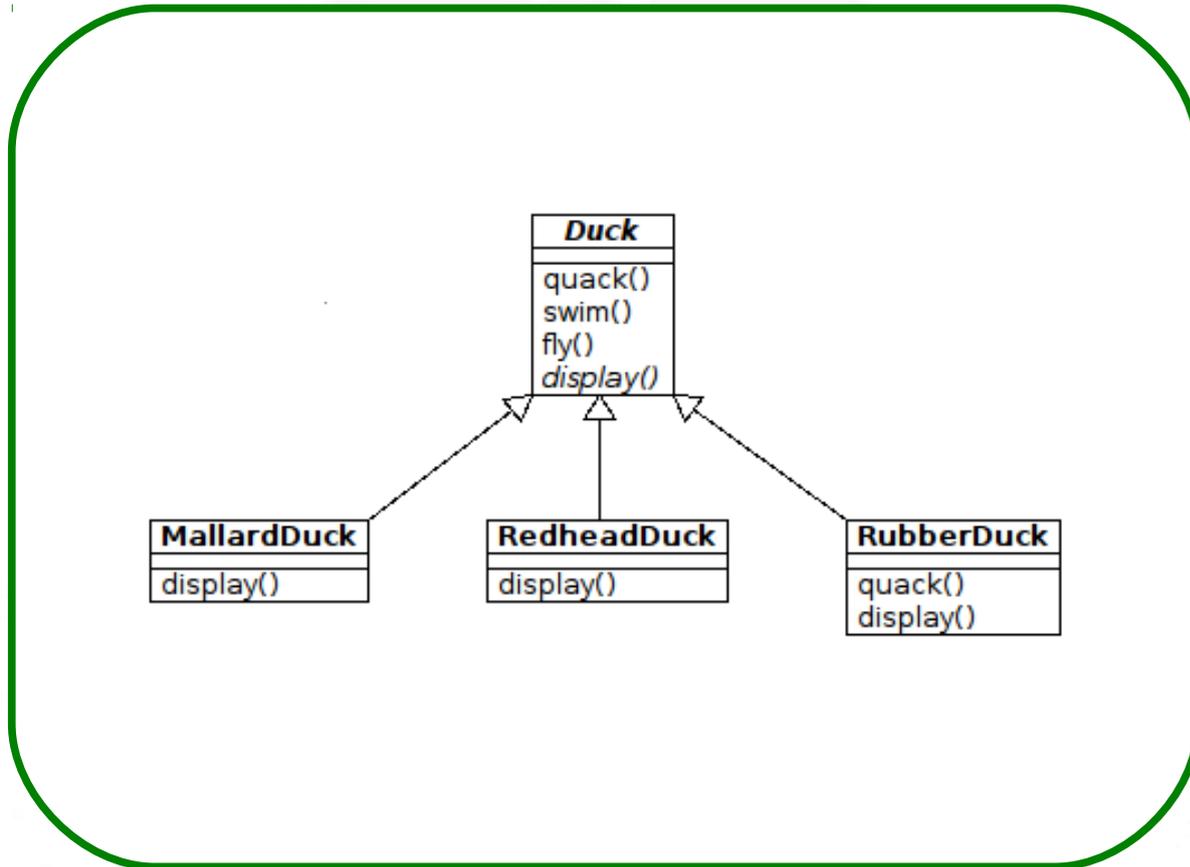
Contexto e Motivação

- Porque precisamos de padrões de projeto ?
 - 1^a



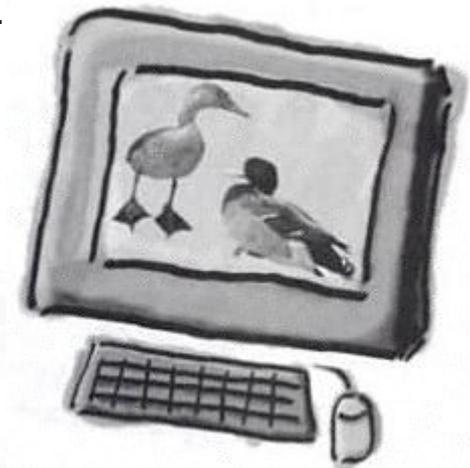
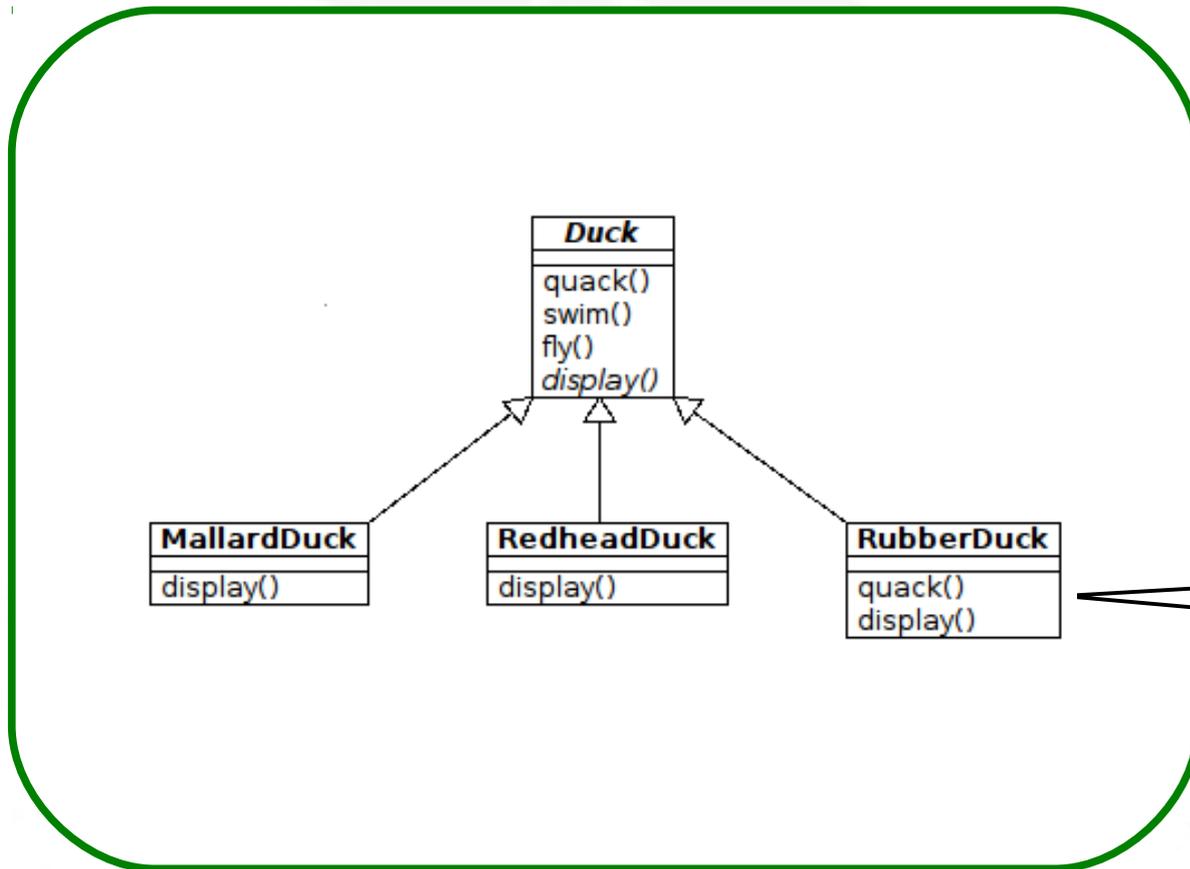
Contexto e Motivação

- Porque precisamos de padrões de projeto ?
 - 2ª mudança: suportar patos de borracha



Contexto e Motivação

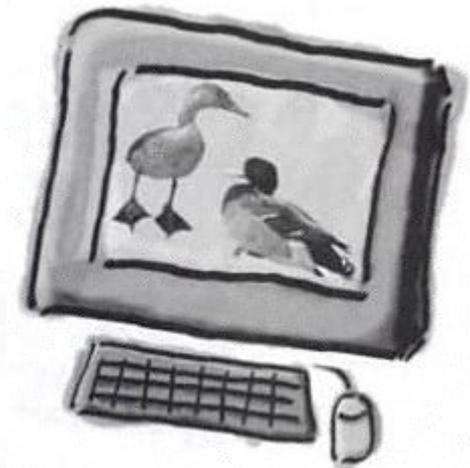
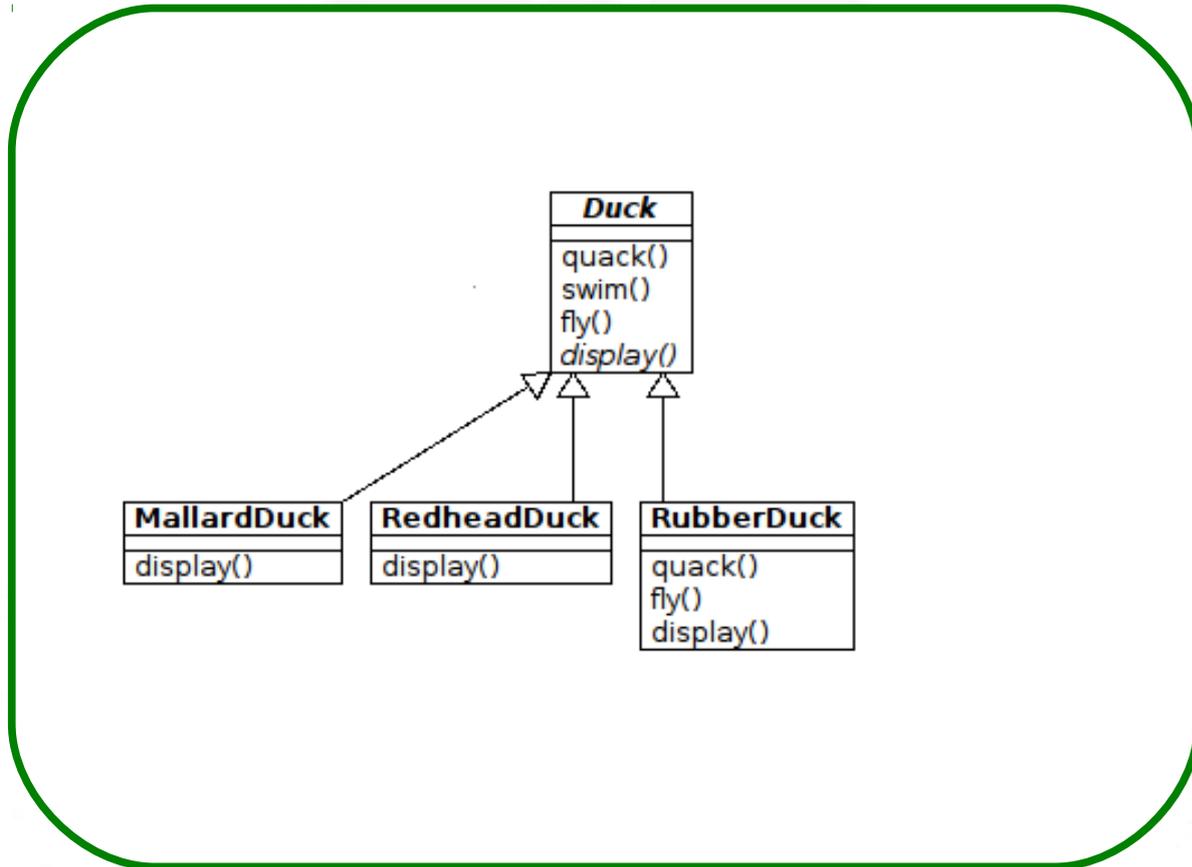
- Porque precisamos de padrões de projeto ?
 - 2ª mudança: suportar patos de borracha



Hmm, patos de borracha não devem voar. O que fazer ?

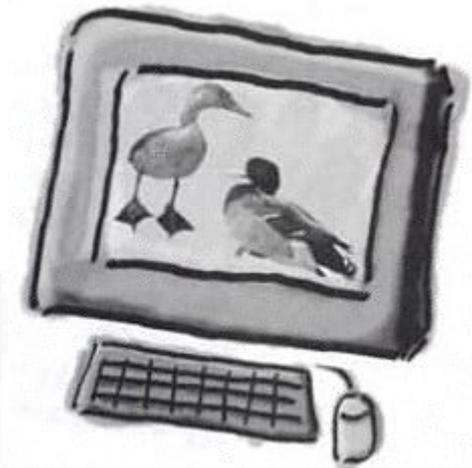
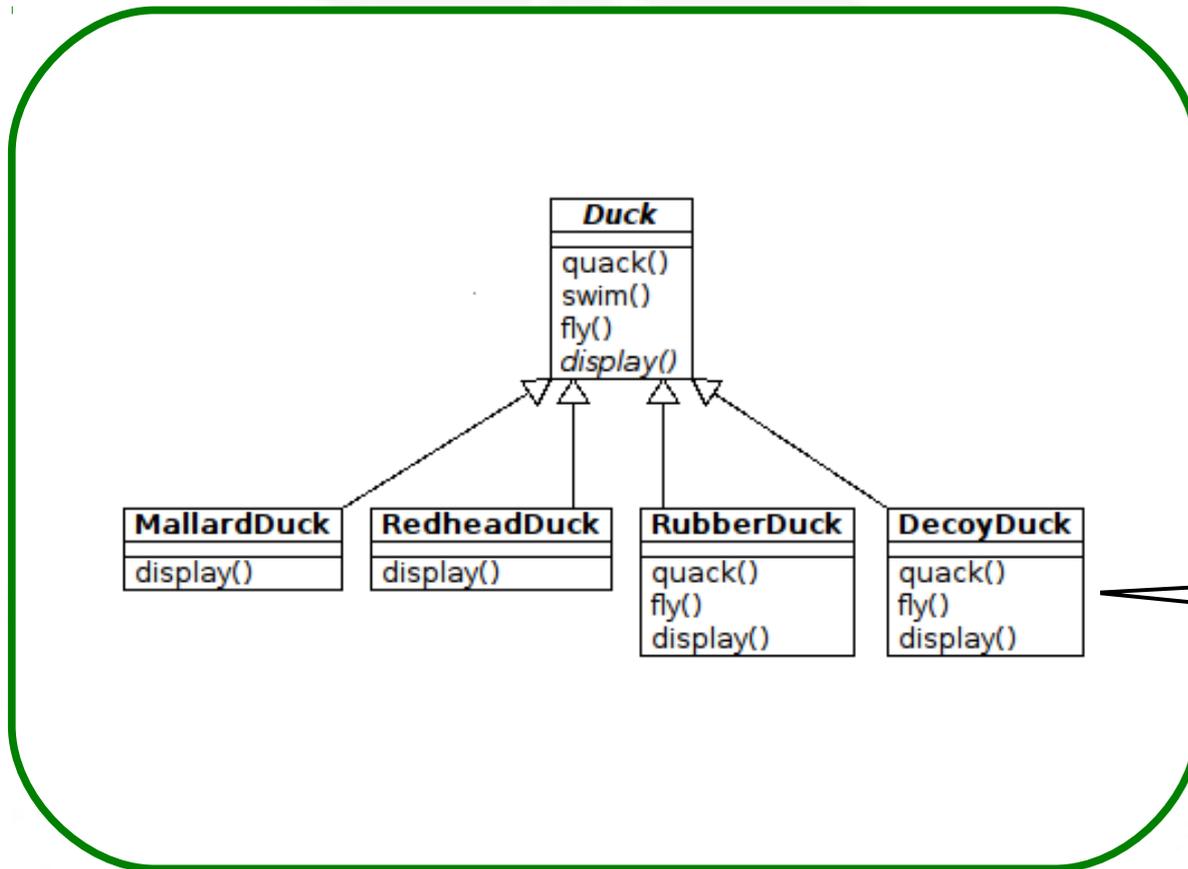
Contexto e Motivação

- Porque precisamos de padrões de projeto ?
 - 1ª tentativa: sobreposições vazias ?



Contexto e Motivação

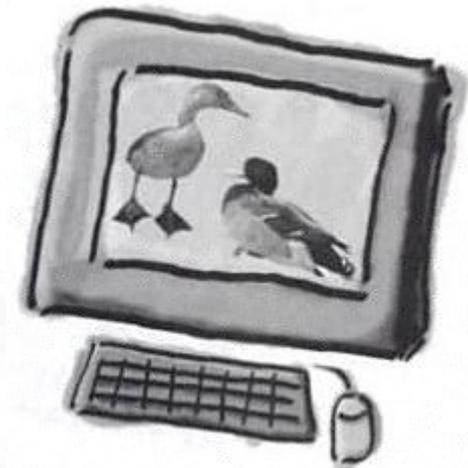
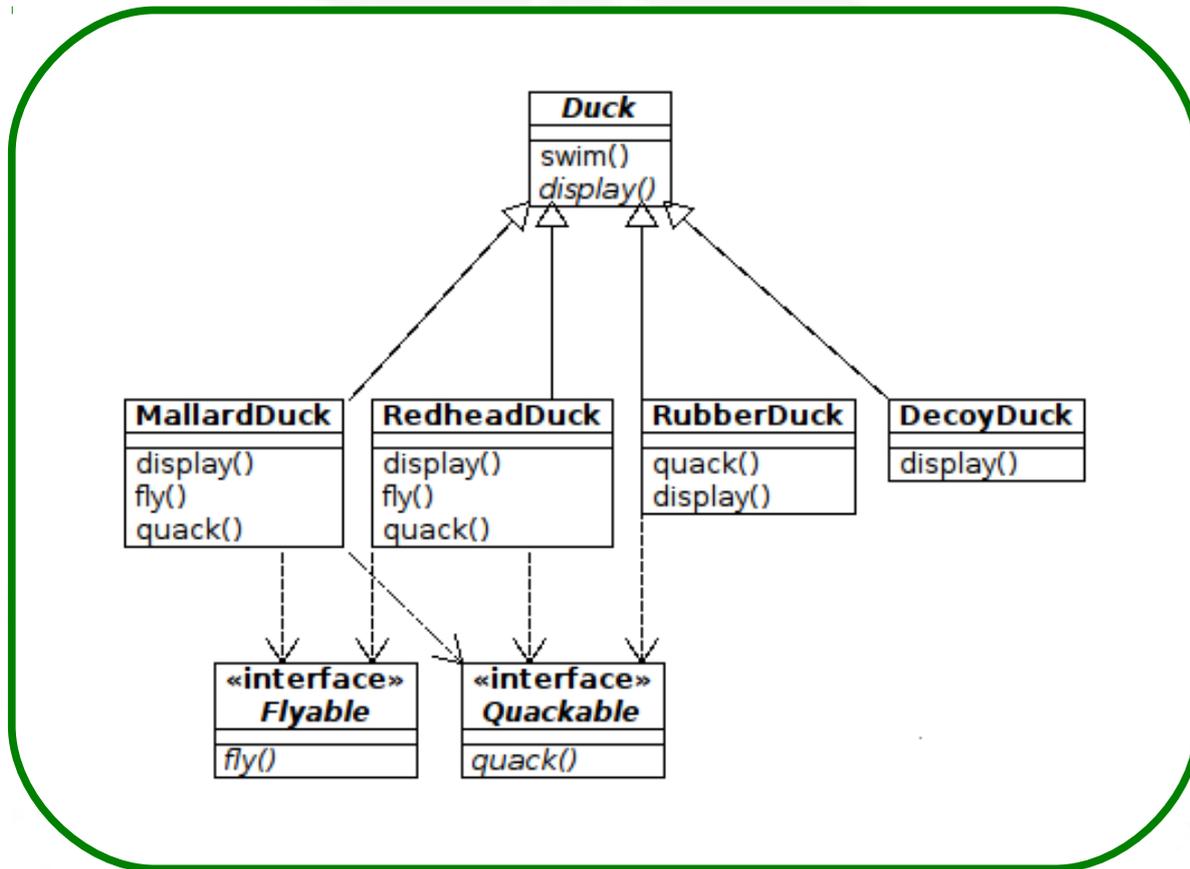
- Porque precisamos de padrões de projeto ?
 - 1ª tentativa: sobreposições vazias ?



E se tivéssemos
que suportar
patos decorativos
?

Contexto e Motivação

- Porque precisamos de padrões de projeto ?
 - 2ª tentativa: interfaces ?



Padrões de Projeto

- “A beleza está nos olhos de quem vê ou pessoas concordariam que certas coisas são bonitas e outras não ?” [Christopher Alexander]
- Culturas tecem julgamentos sobre a beleza, transcendendo crenças individuais [Ruth Benedict]

Padrões de Projeto

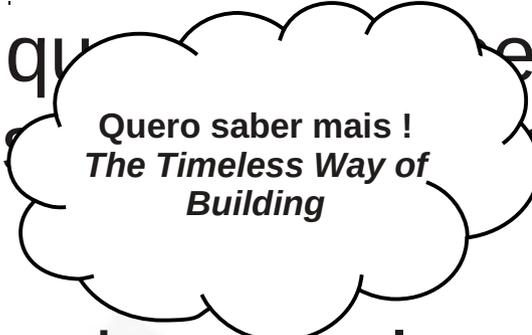
- “A beleza está nos olhos de quem vê. Pessoas concordariam que certas coisas são belas e outras não ?” [Christopher Alexander]
- Culturas tecem julgamentos sobre a beleza, transcendendo crenças individuais [Ruth Benedict]



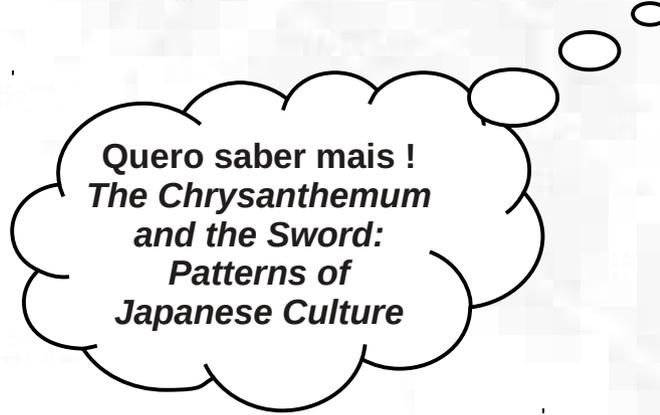
Quero saber mais!
*The Timeless Way of
Building*

Padrões de Projeto

- “A beleza está nos olhos de quem vê. Pessoas concordariam que certas coisas são belas e outras não ?” [Christopher Alexander]
- Culturas tecem julgamentos sobre a beleza, transcendendo crenças individuais [Ruth Benedict]



Quero saber mais !
The Timeless Way of Building



Quero saber mais !
The Chrysanthemum and the Sword: Patterns of Japanese Culture

Padrões de Projeto

- E o que isso tem a ver com *software* ?
 - Muitos problemas encontrados no desenvolvimento de *software* são recorrentes e podem ser resolvidos seguindo uma mesma estratégia (padrão de projeto)
 - Pode-se projetar *software* através da identificação antecipada destes padrões
 - Padrões de projeto proporcionam soluções flexíveis, que acomodam mudanças futuras
- *Design Patterns: Elements of Reusable Object-Oriented Software (Gang of Four - 1994)*

Padrões de Projeto

Padrões de Projeto são descrições de objetos e classes que se comunicam e que são configurados para resolver um problema genérico de projeto OO em um contexto particular

Gang of Four - 1994

- **Características:**
 - Promovem reuso de boas soluções
 - Estabelecem uma terminologia comum
 - Mantêm a discussão no âmbito de projeto, não de implementação
 - São descobertos e não inventados

Princípios de Projeto OO

- Os cinco mandamentos do bom projetista OO:
 1. Objetos são definidos por um “contrato” (interface) e este contrato não é violado
 2. Todos os dados são privados, sem exceções
 3. Deve ser possível modificar a forma com que um objeto é implementado fazendo alterações em uma única classe
 4. Métodos *get* e *set* são, geralmente, indícios de um projeto ruim
 5. Heranças com muitos níveis também são um indício de projeto ruim

Princípios de Projeto OO

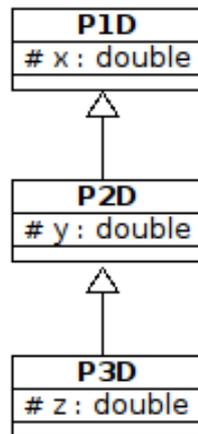
- 1 Encontre o que varia e o encapsule
- 2 Programe pensando em interfaces
- 3 Prefira agregação a herança
- 4 Princípio do Aberto-Fechado (*Open-Closed Principle*)
- 5 Princípio da Substituição de Bárbara Liskov
- 6 Peça por ajuda, não por informação
- 7 *One Rule, One Place*

Princípios de Projeto OO

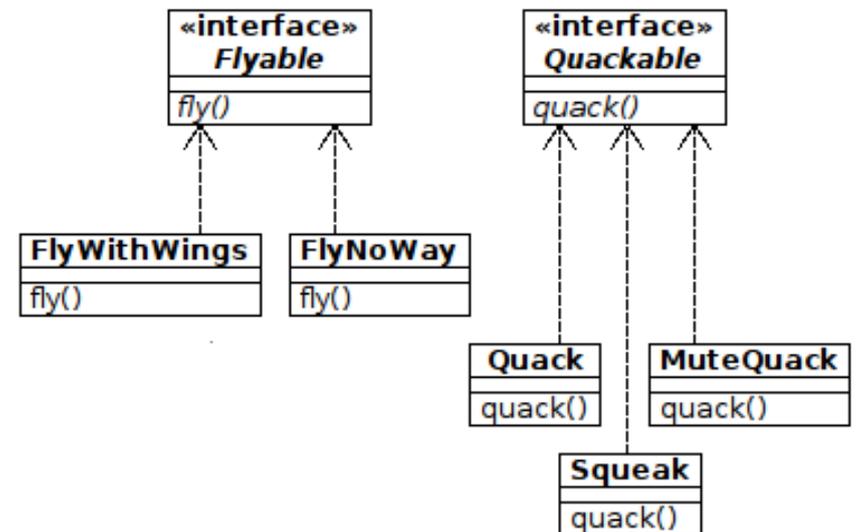
1

ENCONTRE O QUE VARIA E O ENCAPSULE

HERANÇA DE IMPLEMENTAÇÃO (sub-classing)



HERANÇA DE INTERFACE (sub-typing)



Princípios de Projeto OO

1

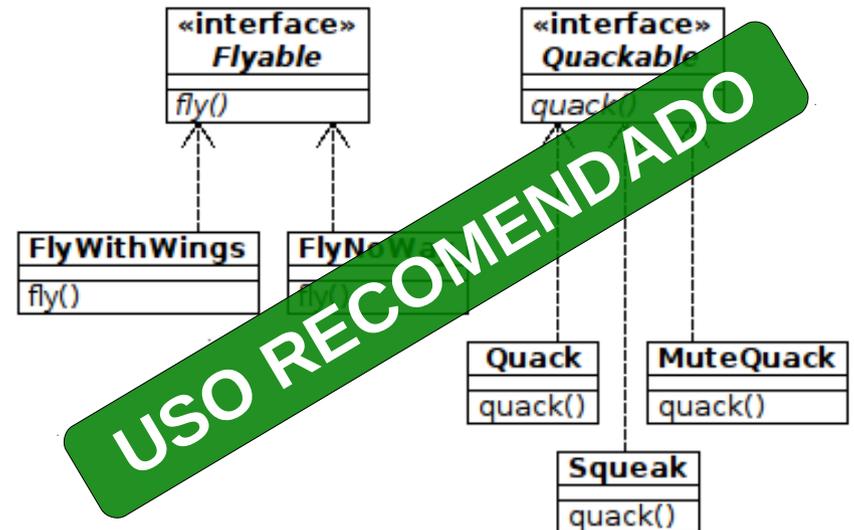
ENCONTRE O QUE VARIA E O ENCAPSULE

HERANÇA DE IMPLEMENTAÇÃO
(sub-classing)



USE COM CUIDADO

HERANÇA DE INTERFACE
(sub-typing)



USO RECOMENDADO

Princípios de Projeto OO

1

ENCONTRE O QUE VARIA E

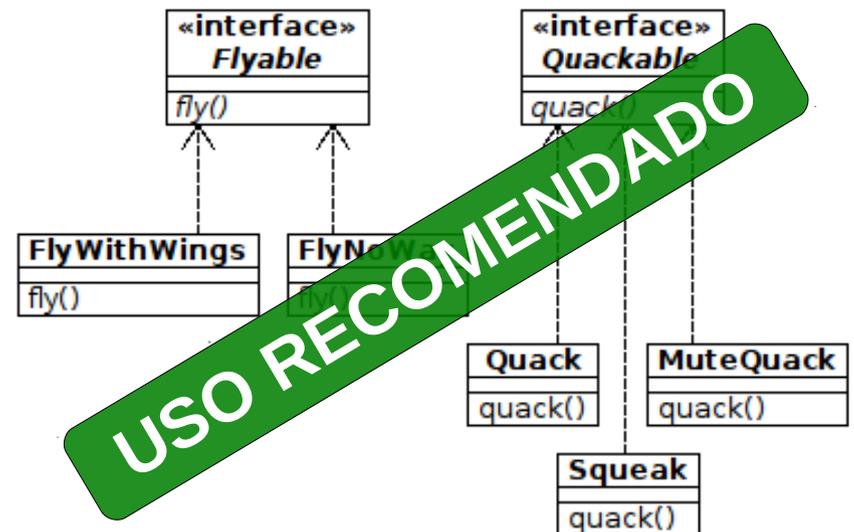
Quero saber mais!
*Taivalsaari, On The
Notion of
Inheritance*

HERANÇA DE IMPLEMENTAÇÃO (sub-classing)



USE COM CUIDADO

HERANÇA DE INTERFACE (sub-typing)

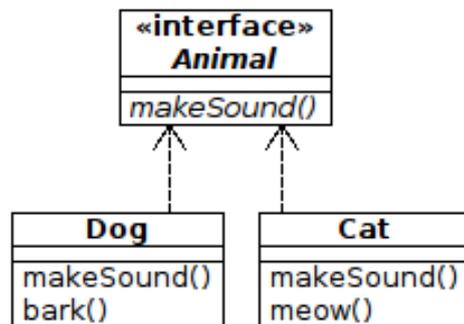


USO RECOMENDADO

Princípios de Projeto OO

2

PROGRAME PENSANDO EM INTERFACES



PROGRAMAÇÃO DIRIGIDA A IMPLEMENTAÇÕES

```
Dog dog = new Dog();
dog.bark();
```

PROGRAMAÇÃO DIRIGIDA A INTERFACES

```
Animal animal = new Dog();
animal.makeSound();
```

OU (AINDA MELHOR)

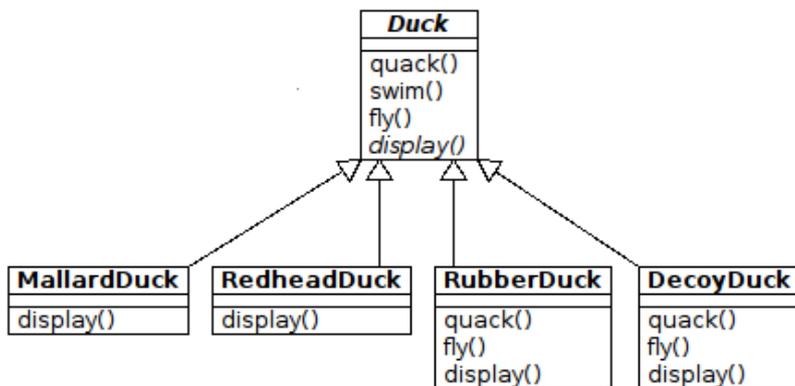
```
Animal animal = createAnimal();
animal.makeSound();
```

Princípios de Projeto OO

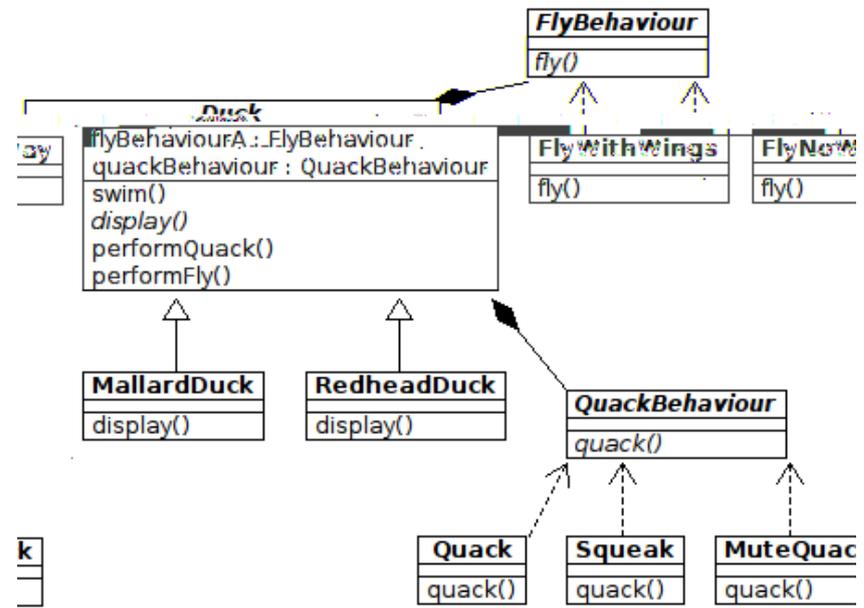
3

PREFIRA AGREGAÇÃO A HERANÇA

COM HERANÇA



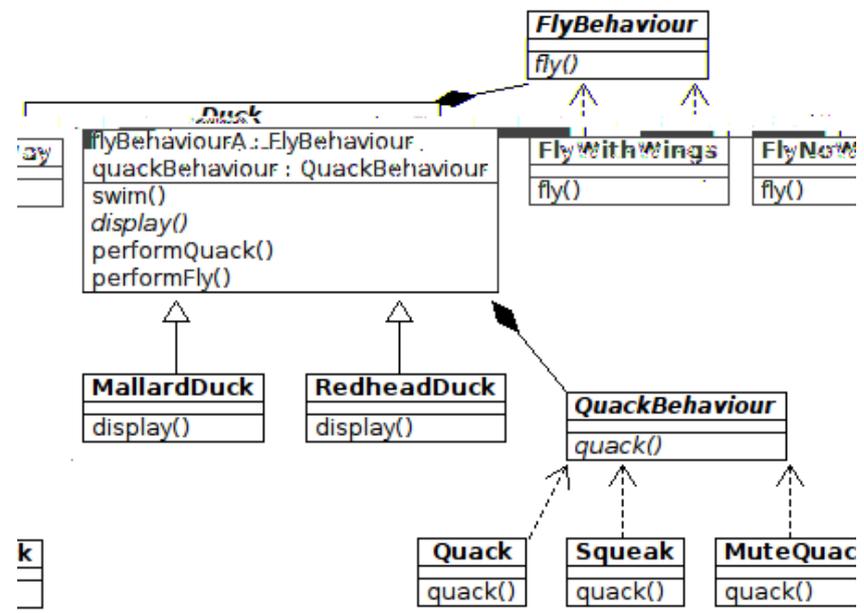
COM AGREGAÇÃO



Princípios de Projeto OO

4 PRINCÍPIO DO ABERTO-FECHADO: módulos (classes) devem ser fechados para modificação e abertos para extensão

- Novas funcionalidades são criadas com a introdução de novas classes
- Evita a introdução de novos *bugs*
- Fundamenta uma arquitetura de *plugins*



Princípios de Projeto OO

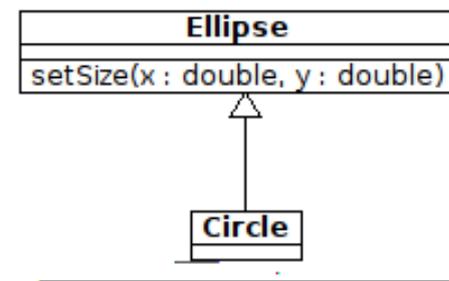
5

PRINCÍPIO DA SUBSTITUIÇÃO DE BÁRBARA LISKOV

Seja $q(x)$ uma propriedade definida para objetos x do tipo T .
Então $q(y)$ deve ser verdade para objetos y do tipo S , onde S
é um sub-tipo de T

- Define o conceito de *substitutability*
- Formaliza a metodologia de *design by contract*

VIOLAÇÕES DO PRINCÍPIO



Princípios de Projeto OO

5

PRINCÍPIO DA SUBSTITUIÇÃO DE BÁRBARA LISKOV

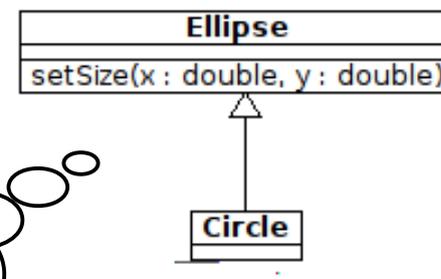
Seja $q(x)$ uma propriedade definida para objetos x do tipo T .
Então $q(y)$ deve ser verdade para objetos y do tipo S , onde S
é um sub-tipo de T

Quero saber mais !
*Liskov, Behavioral
Subtyping Using
Invariants and
Constraints*

- Demonstra o conceito de *substitutability*.
- Formaliza a metodologia de *design by contract*.

Quero saber mais !
C++ FAQ Lite

VIOLAÇÕES DO PRINCÍPIO



Princípios de Projeto OO

6

PEÇA POR AJUDA, NÃO POR INFORMAÇÃO

- “A manutenibilidade é inversamente proporcional à quantidade de dados que trafega entre os objetos” [James Gosling]
- Em poucos casos *gets* e *sets* são justificados

COM GETS E SETS

```
Money a, b;  
a.setValue(a.getValue() +  
           b.getValue());
```

SEM GETS E SETS

```
Money a, b;  
a.increaseBy(b);
```

Princípios de Projeto OO

6

PEÇA POR AJUDA, NÃO POR INFORMAÇÃO

- “A manutenibilidade é inversamente proporcional à quantidade de dados que trafega entre os objetos” [James Holub, *Holub, Holub on Patterns – Getters and Setters are Evil*]
- Em poucos casos, e sets são justificados

Quero saber mais!
*Holub, Holub on
Patterns – Getters
and Setters are Evil*

COM GETS E SETS

```
Money a, b;  
a.setValue(a.getValue() +  
           b.getValue());
```

SEM GETS E SETS

```
Money a, b;  
a.increaseBy(b);
```

Princípios de Projeto OO

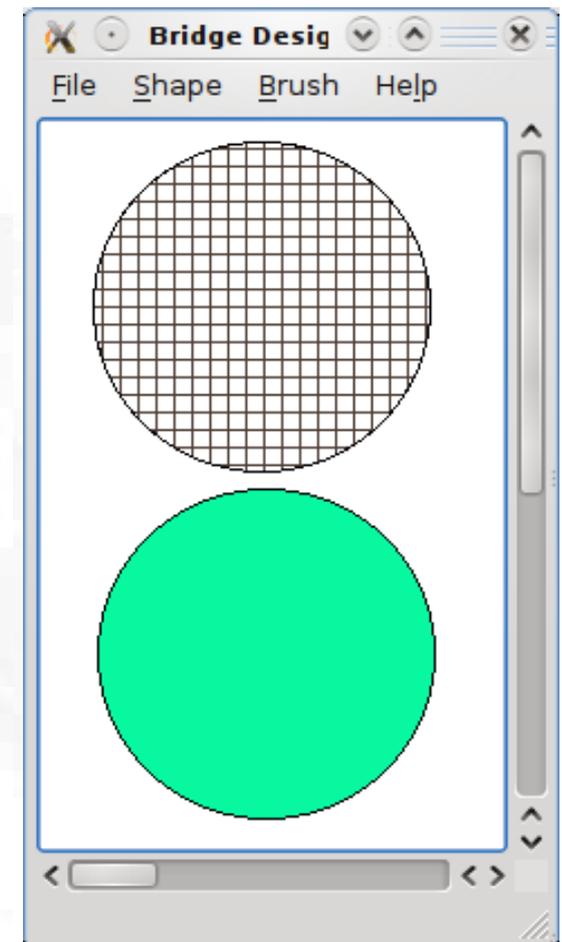
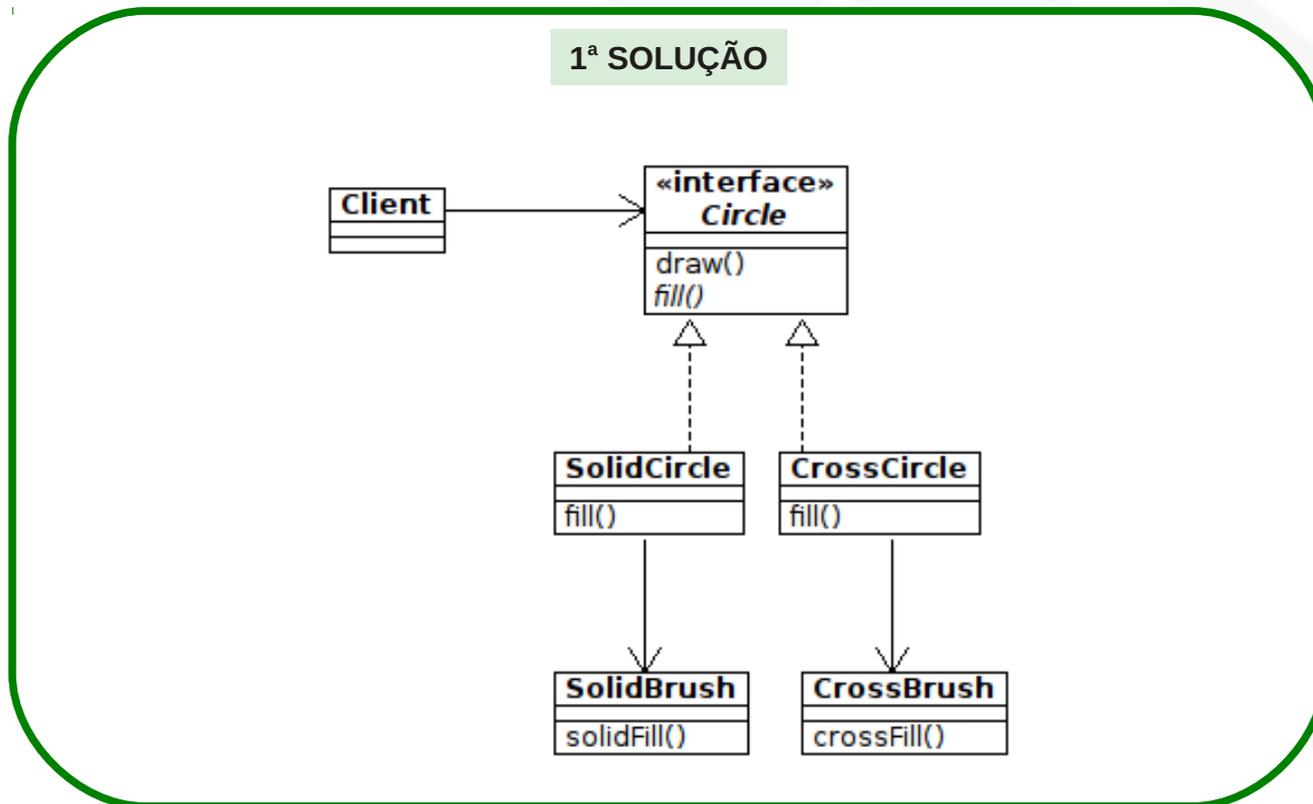
7

ONE RULE, ONE PLACE

- Cada regra de negócio deve ser implementada em somente um lugar, sem código redundante

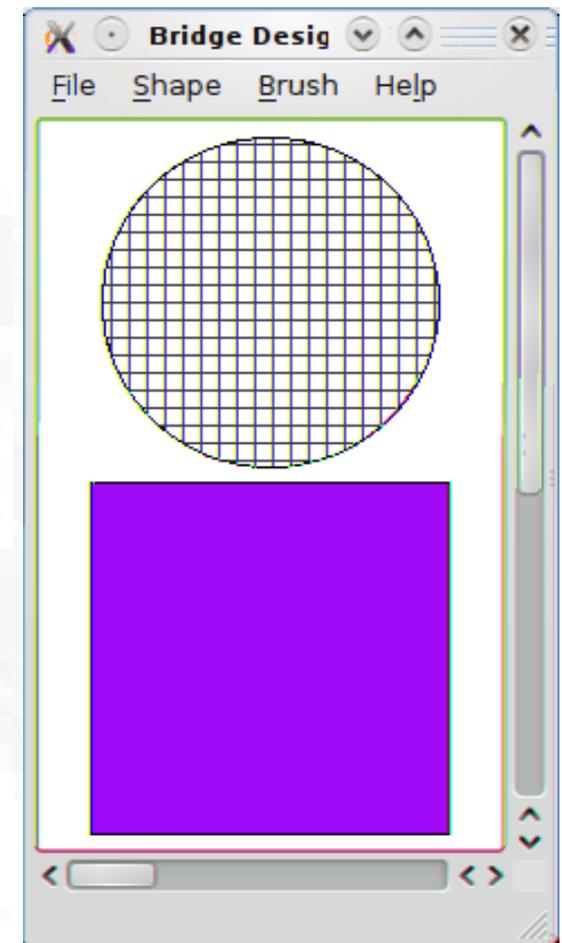
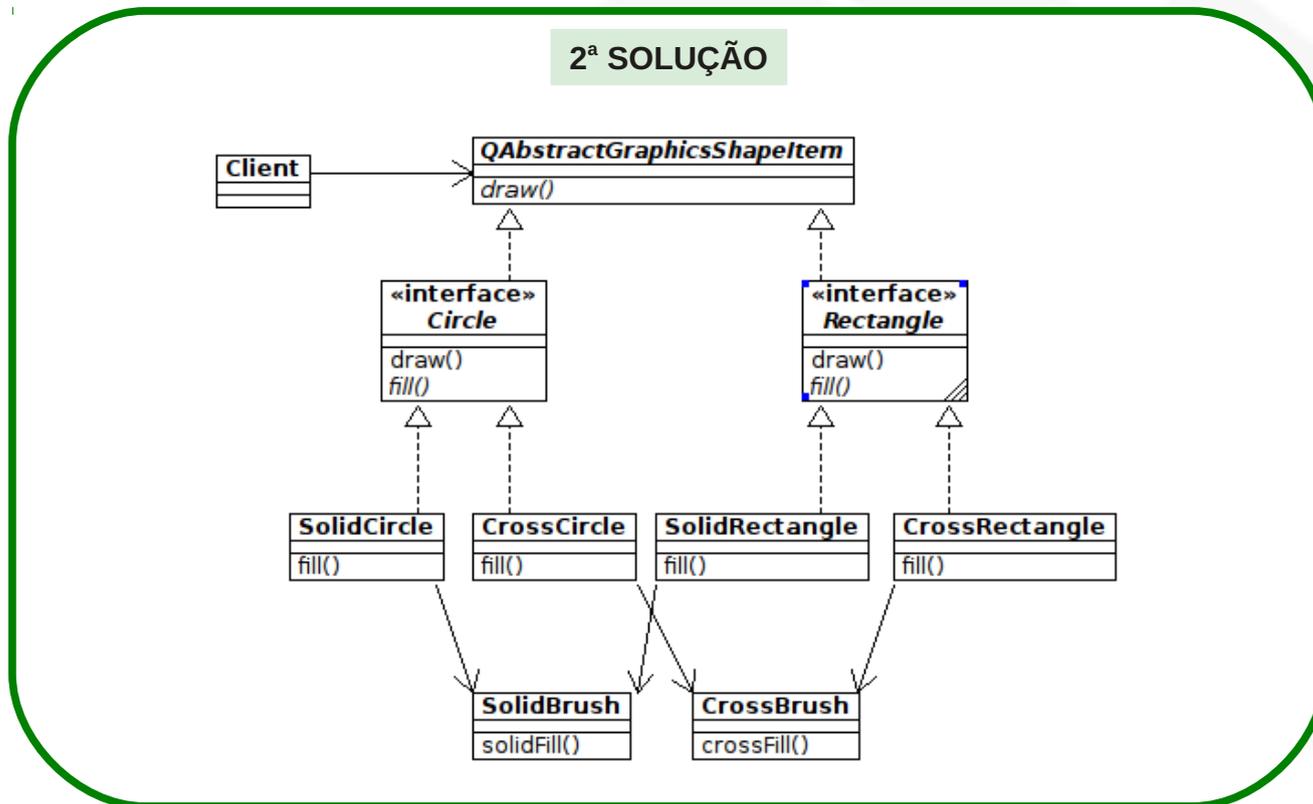
Estudo de Caso: *Bridge*

- Contexto: você deve implementar um sistema que desenhe círculos com dois tipos diferentes de preenchimento - sólido e cruzado



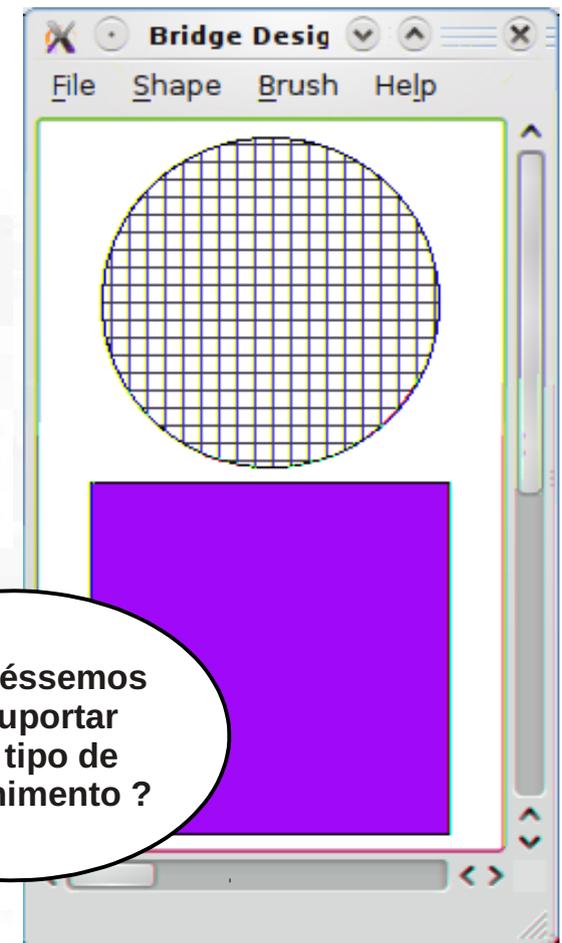
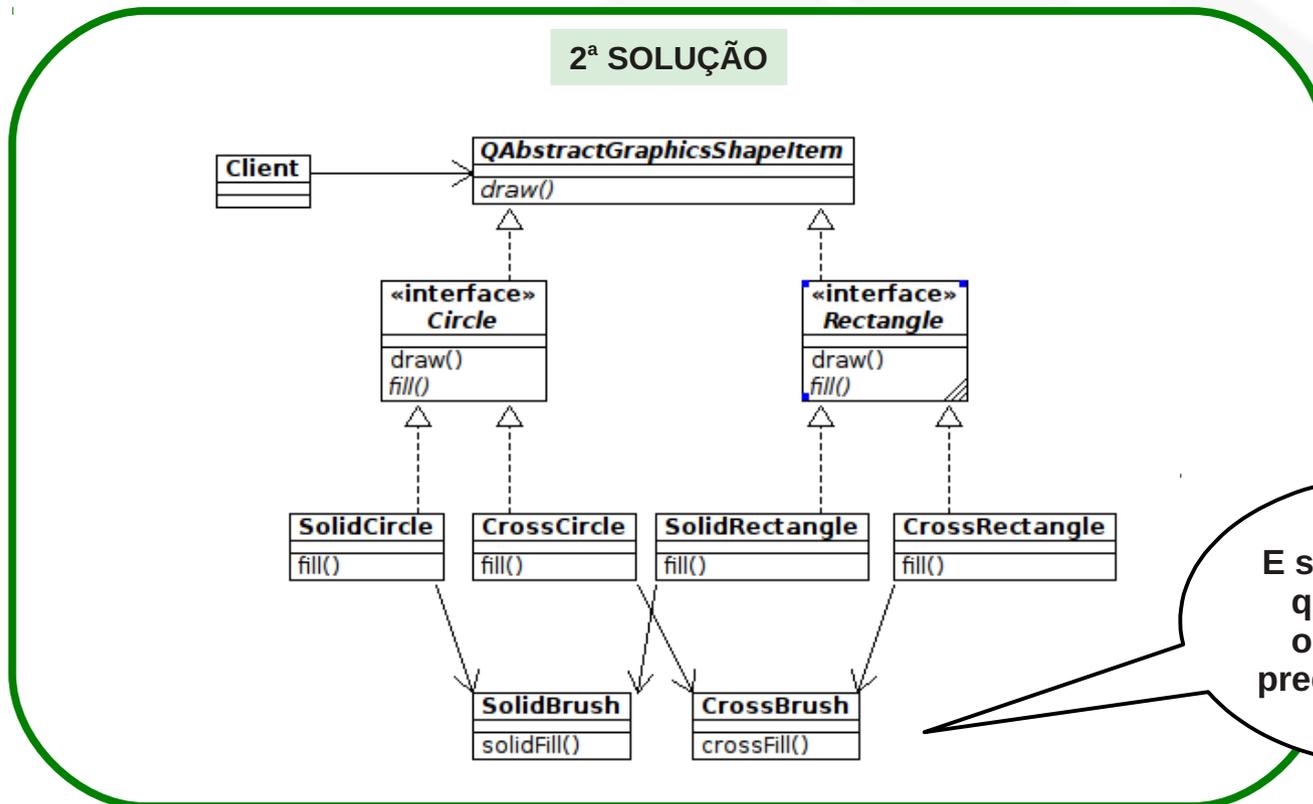
Estudo de Caso: *Bridge*

- 1ª mudança: o sistema deve agora também desenhar retângulos com as duas possibilidades de preenchimento



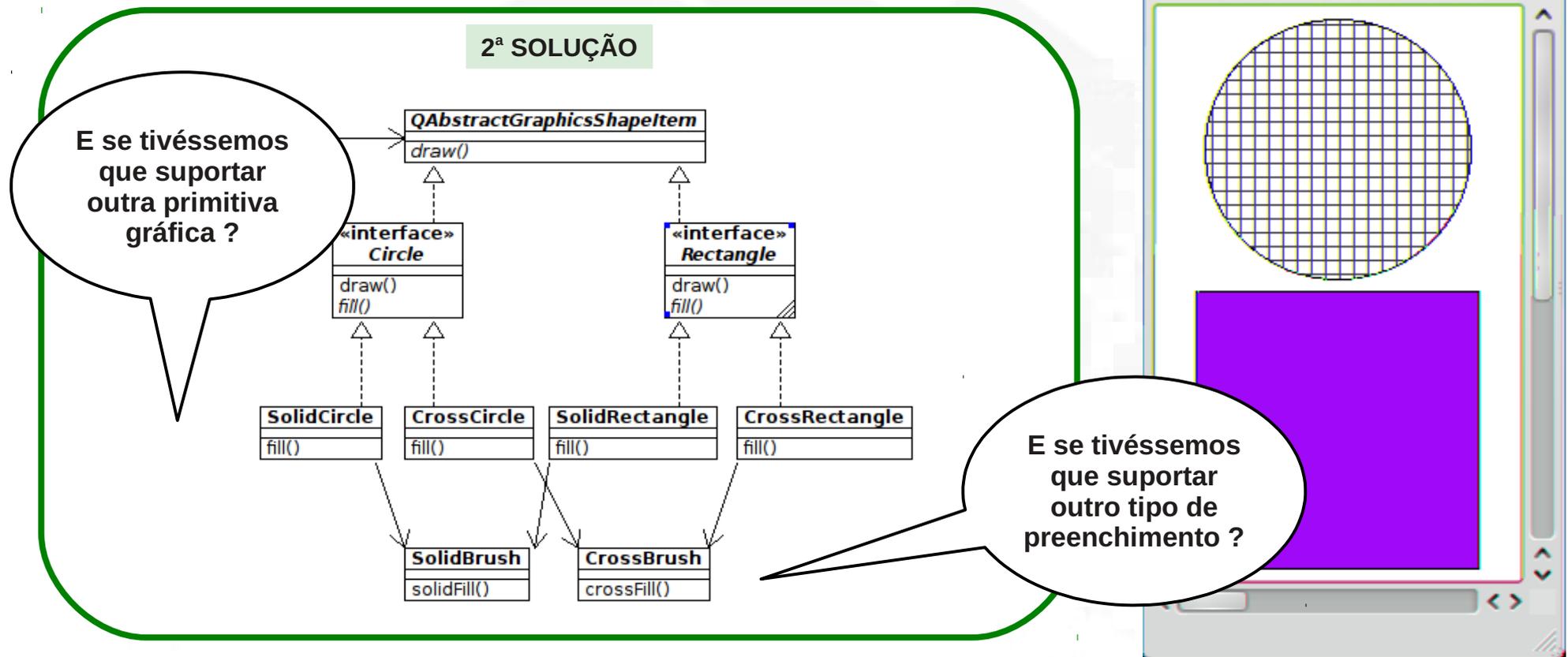
Estudo de Caso: *Bridge*

- 1ª mudança: o sistema deve agora também desenhar retângulos com as duas possibilidades de preenchimento



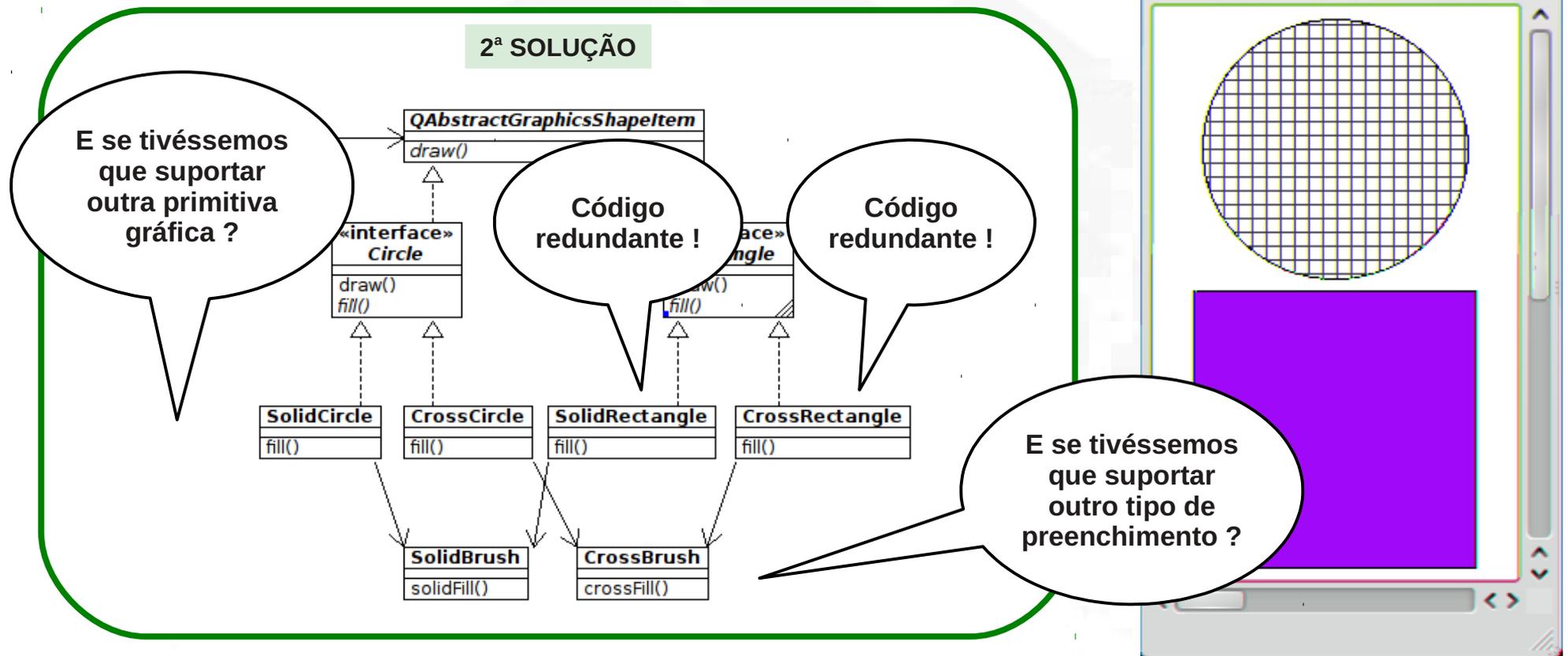
Estudo de Caso: *Bridge*

- 1ª mudança: o sistema deve agora também desenhar retângulos com as duas possibilidades de preenchimento



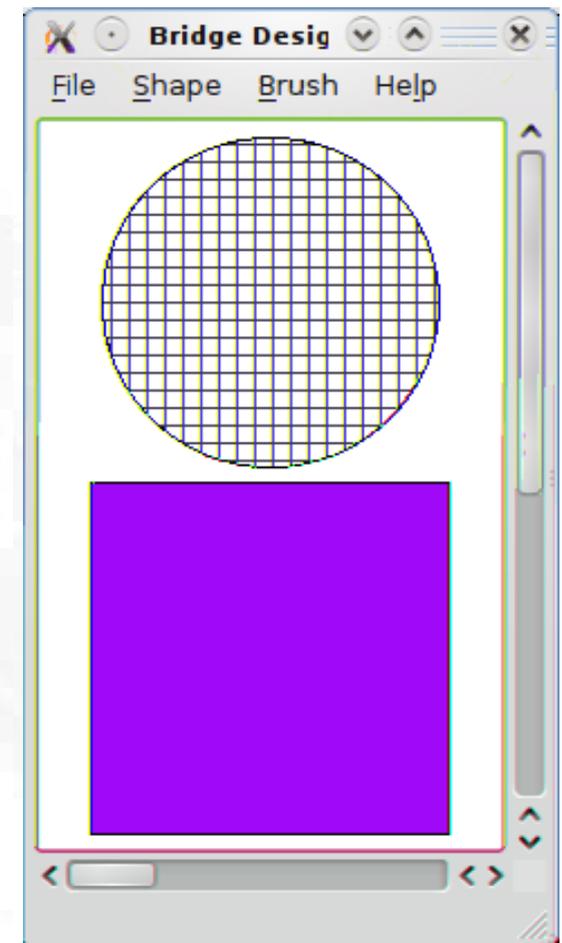
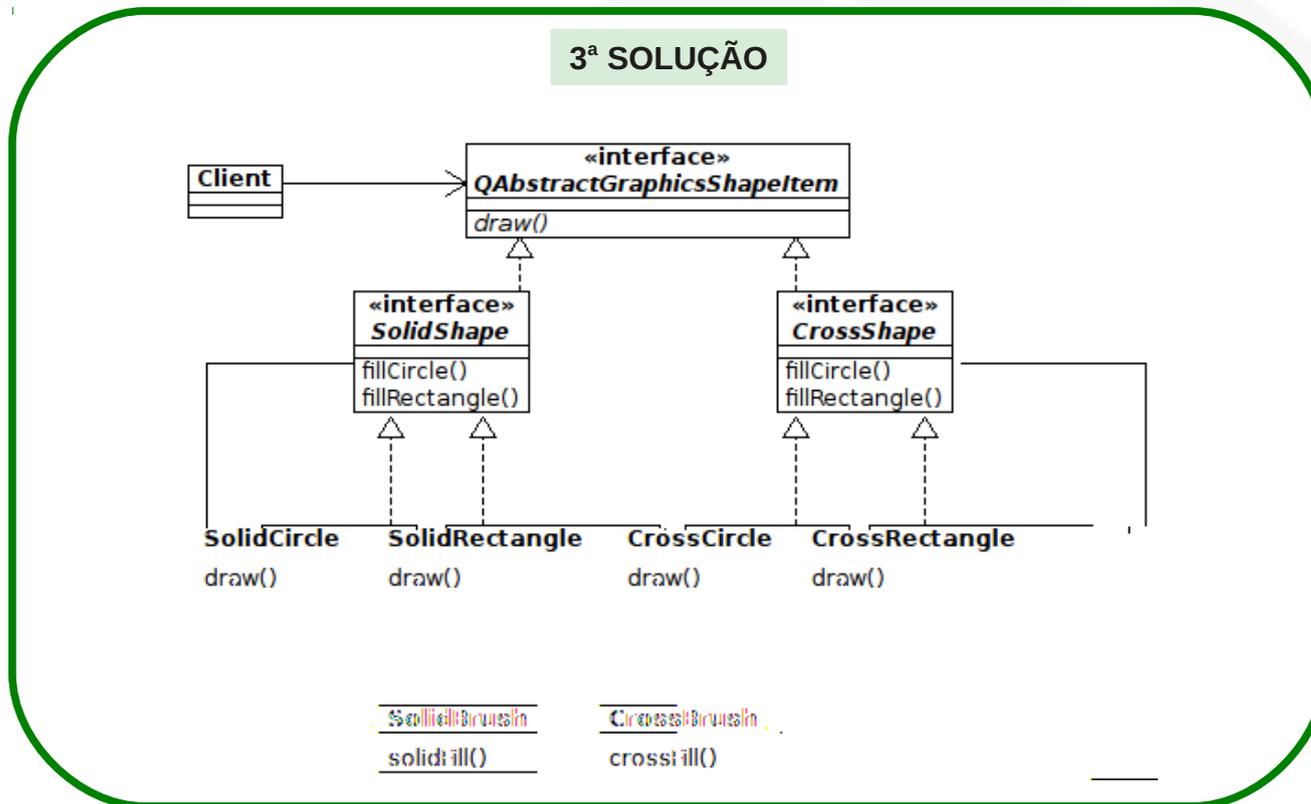
Estudo de Caso: *Bridge*

- 1ª mudança: o sistema deve agora também desenhar retângulos com as duas possibilidades de preenchimento



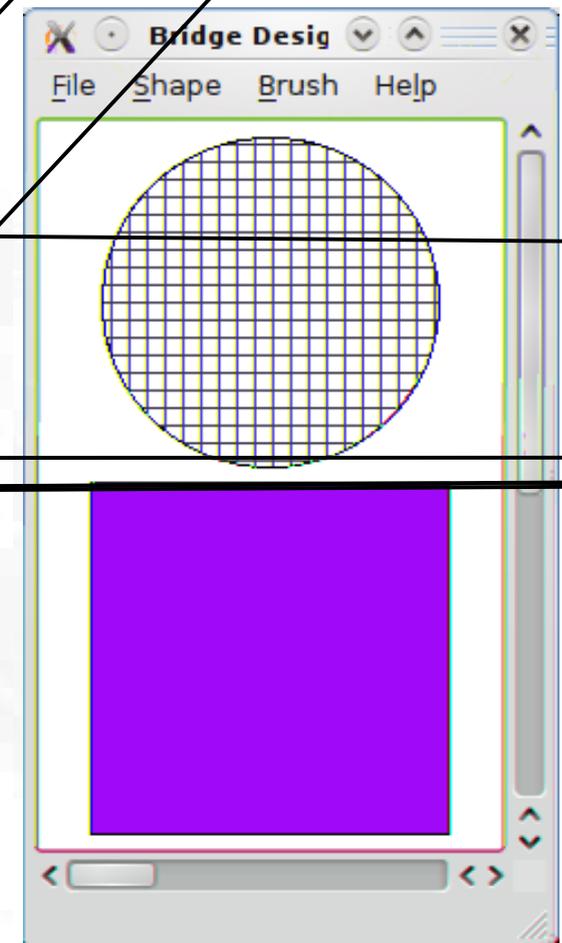
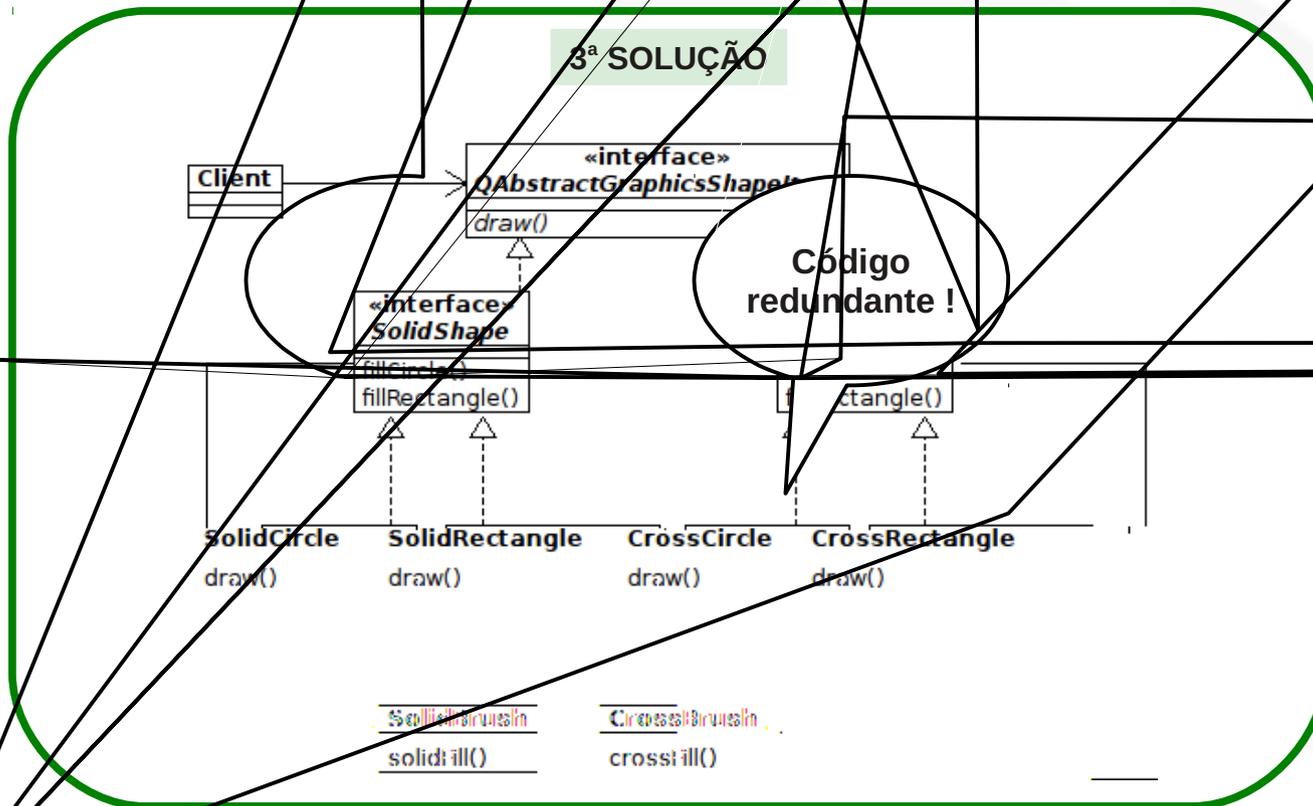
Estudo de Caso: *Bridge*

- 1ª mudança: o sistema deve agora também desenhar retângulos com as duas possibilidades de preenchimento



Estudo de Caso: *Bridge*

- 1ª mudança: o sistema deve agora também desenhar retângulos com as duas possibilidades de preenchimento

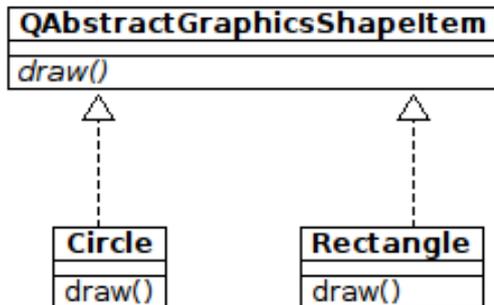


Estudo de Caso: *Bridge*

- O que há de errado com estes projetos ?
 - Violam o princípio ① “Encontre o que varia e o encapsule” e o princípio ③ “Prefira agregação a herança”
 - Quais aspectos variam ?
 - As primitivas gráficas: círculo, retângulo etc
 - As formas de preenchimento: sólido, cruzado etc
 - Deve-se conceber tais aspectos como conceitos que podem possuir diferentes implementações
 - Heranças de implementação acoplam eternamente a classe-filha com a classe-pai

Estudo de Caso: *Bridge*

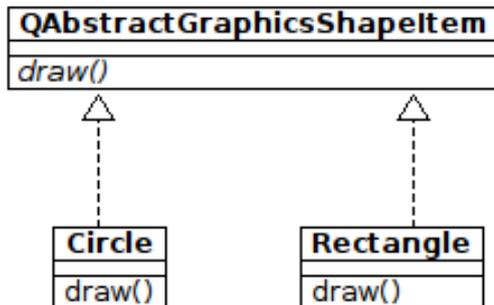
- Um projeto melhorado:



Estudo de Caso: *Bridge*

- Um projeto melhorado:

1º conceito que varia



Estudo de Caso: *Bridge*

- Um projeto melhorado:



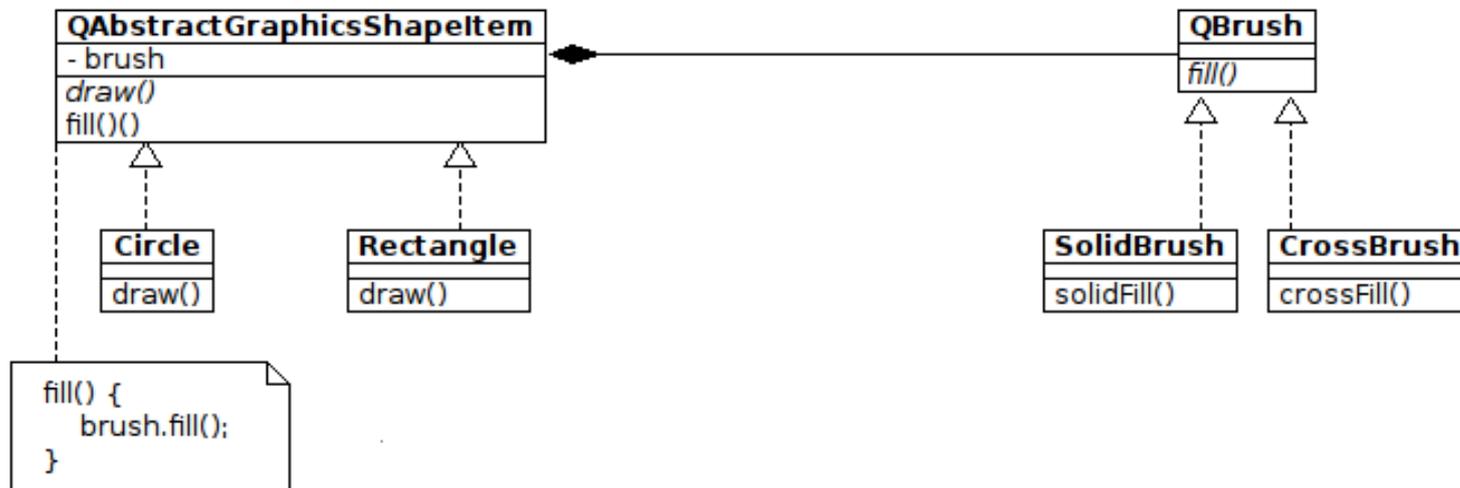
Estudo de Caso: *Bridge*

- Um projeto melhorado:



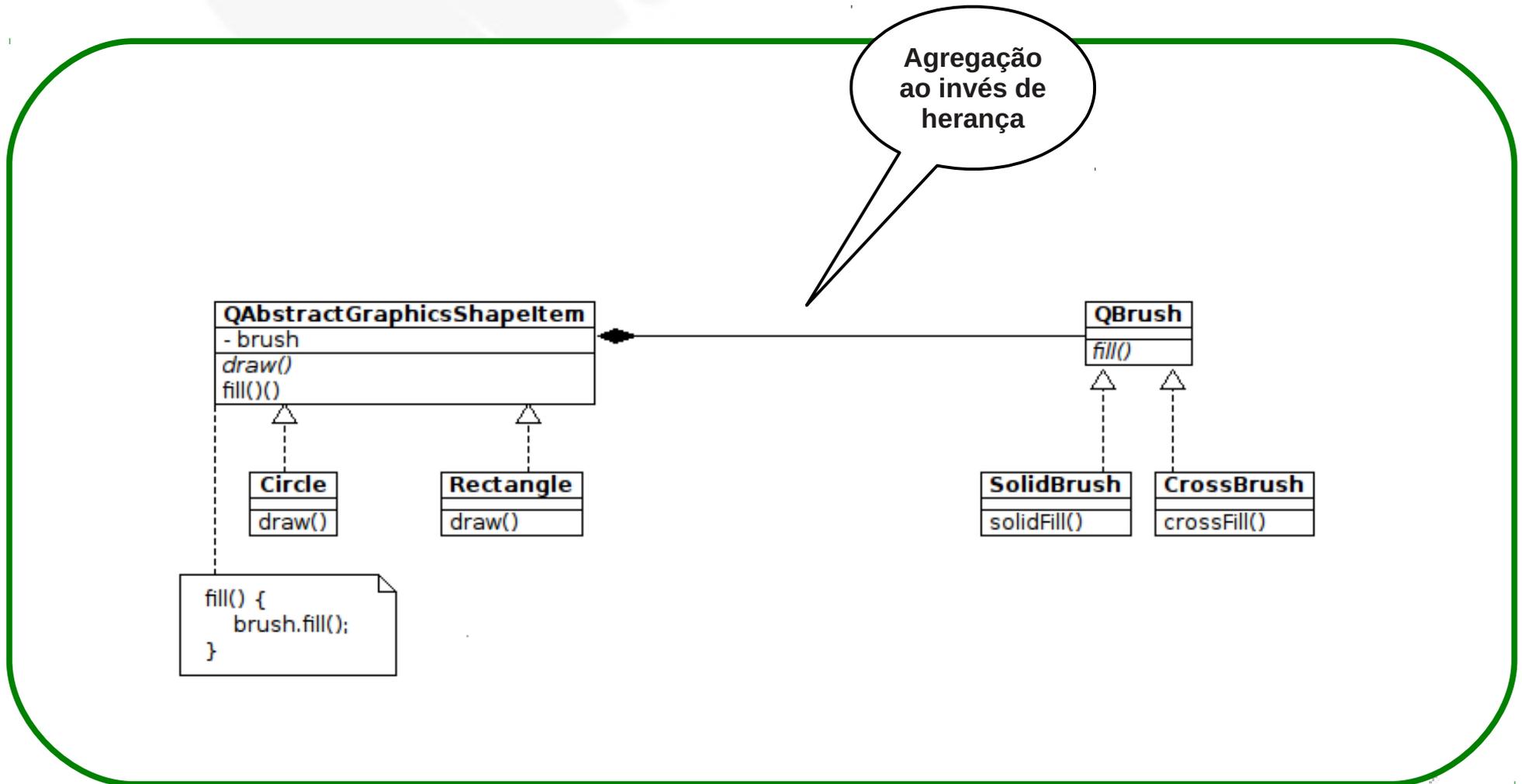
Estudo de Caso: *Bridge*

- Um projeto melhorado:



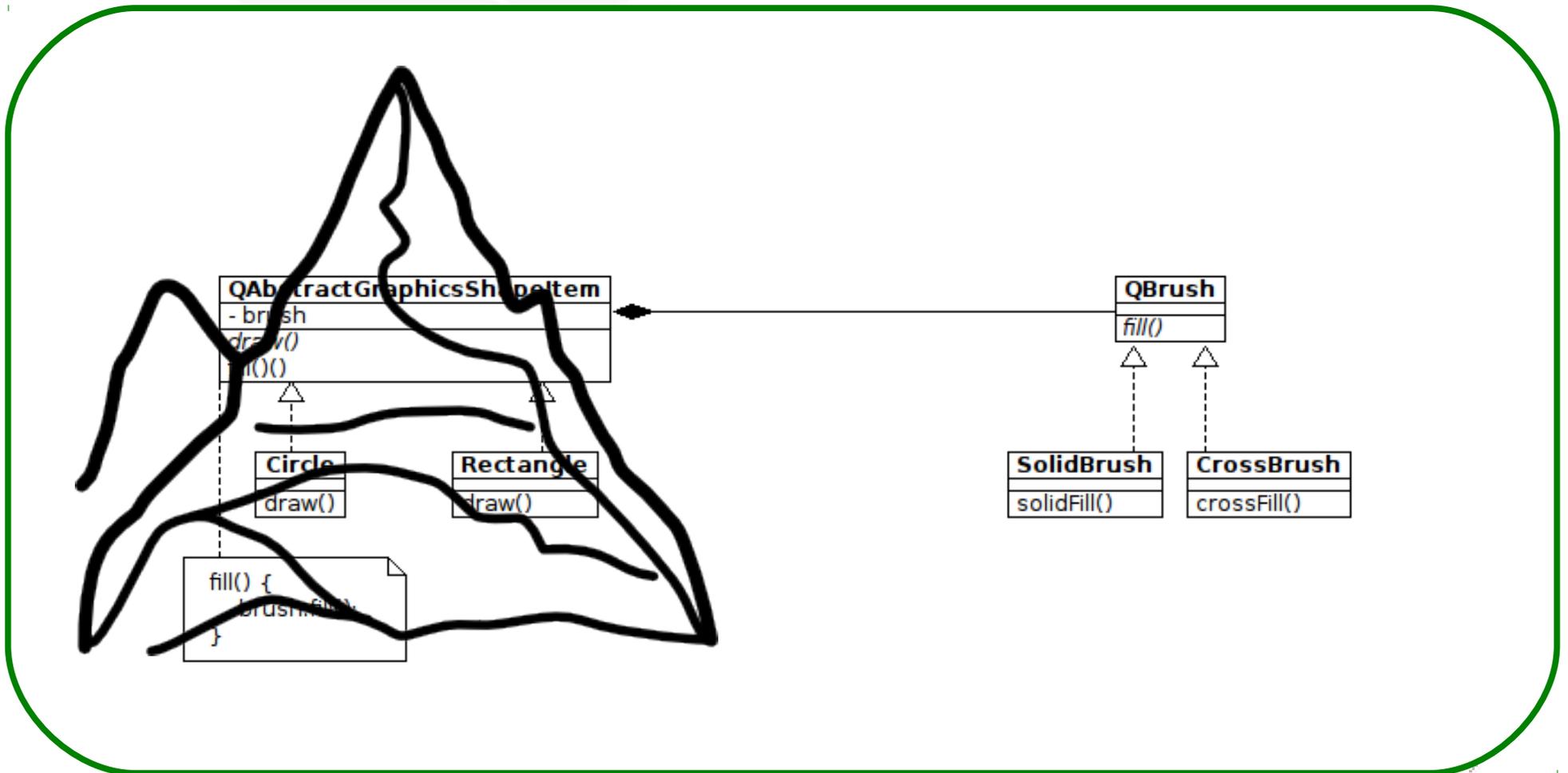
Estudo de Caso: *Bridge*

- Um projeto melhorado:



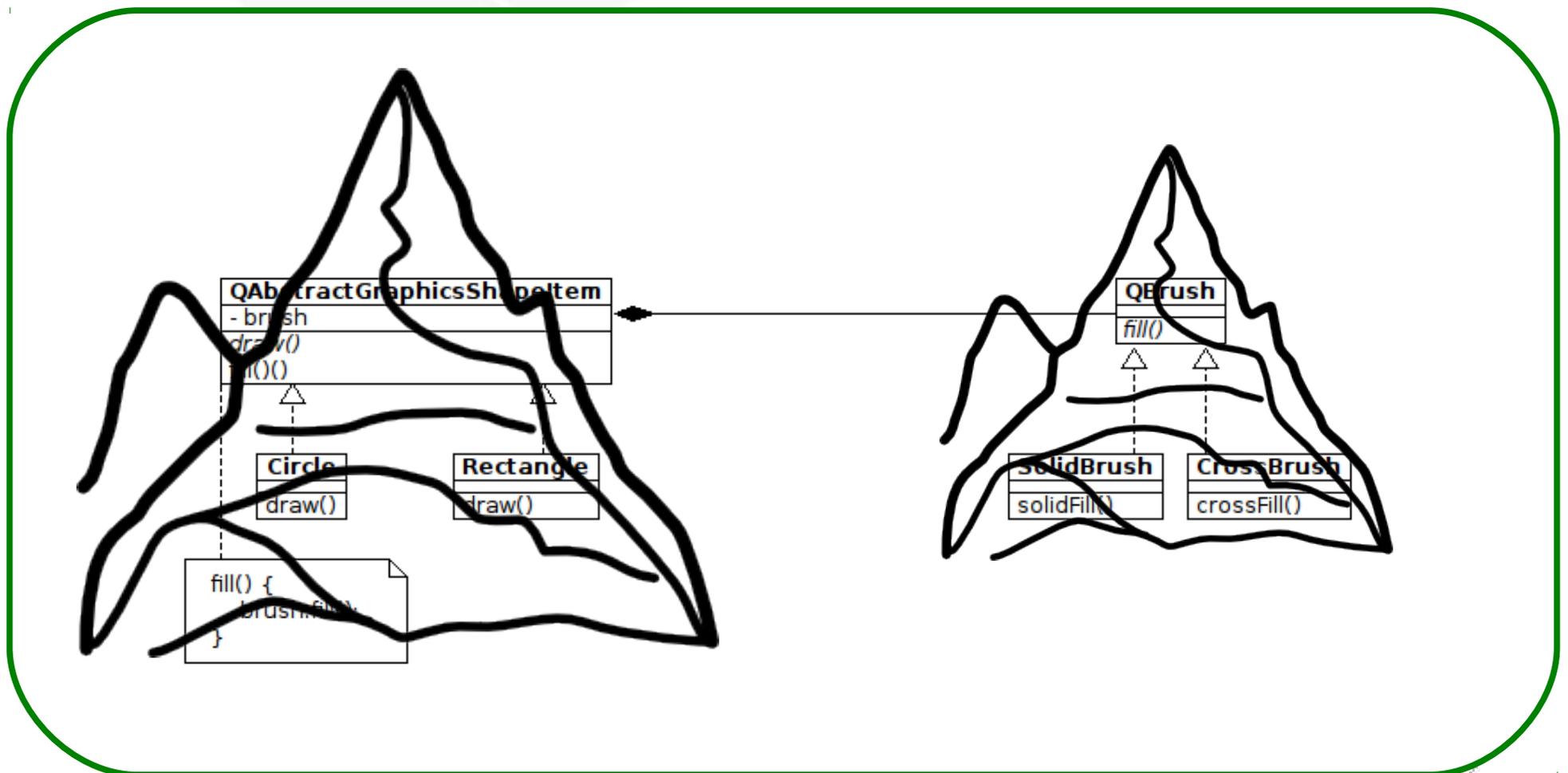
Estudo de Caso: *Bridge*

- Um projeto melhorado:



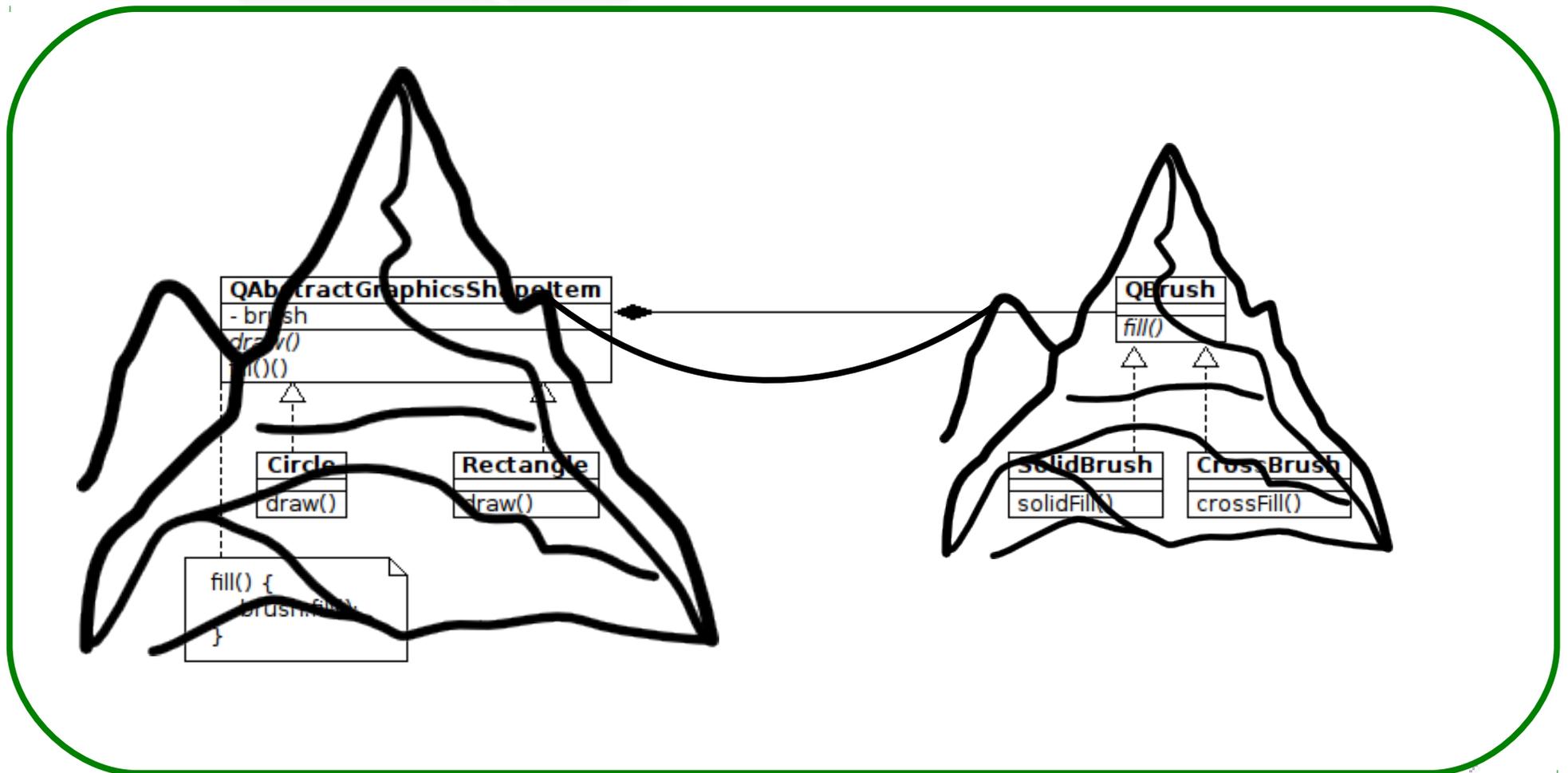
Estudo de Caso: *Bridge*

- Um projeto melhorado:



Estudo de Caso: *Bridge*

- Um projeto melhorado:



Estudo de Caso: *Bridge*

BRIDGE

Desacopla a abstração da sua implementação, de modo que os dois possam variar de forma independente

Padrões Arquiteturais e Idiomas

- Muitos outros padrões existem e são amplamente utilizados:
 - *Creational Patterns*: desacoplam o código que cria as instâncias de classes do código que efetivamente utiliza estes objetos. Ex: *Abstract Factory*, *Builder* e *Prototype*
 - *Structural Patterns*: formam estruturas maiores através da composição de objetos, sem comprometer a flexibilidade da solução. Ex: *Adapter*, *Bridge*, *Decorator* e *Flyweight*
 - *Behavioral Patterns*: descrevem padrões de comunicação entre objetos, permitindo a variação de algoritmos e de responsabilidades. Ex: *Strategy*, *Observer* e *Iterator*

Padrões Arquiteturais e Idiomas

- Padrões são aplicados em diversos níveis de abstração:

PADRÕES E ESTILOS ARQUITETURAIS

PADRÕES DE PROJETO

IDIOMAS DE PROGRAMAÇÃO

Padrões Arquiteturais e Idiomas

- Padrões são aplicados em diversos níveis de abstração:

PADRÕES E ESTILOS ARQUITETURAIS

PADRÕES DE PROJETO

IDIOMAS DE PROGRAMAÇÃO

Boas práticas para desenvolvimento em uma determinada linguagem

Padrões Arquiteturais e Idiomas

- Padrões são aplicados em diversos níveis de abstração:

PADRÕES E ESTILOS ARQUITETURAIS

PADRÕES DE PROJETO

IDIOMAS DE PROGRAMAÇÃO

Soluções flexíveis
para problemas de
projeto OO
recorrentes

Boas práticas para
desenvolvimento em
uma determinada
linguagem

Padrões Arquiteturais e Idiomas

- Padrões são aplicados em diversos níveis de abstração:

PADRÕES E ESTILOS ARQUITETURAIS

Soluções recorrentes para modularização em alto nível

PADRÕES DE PROJETO

Soluções flexíveis para problemas de projeto OO recorrentes

IDIOMAS DE PROGRAMAÇÃO

Boas práticas para desenvolvimento em uma determinada linguagem

Conclusões

- Estudar padrões de projeto é uma excelente oportunidade para efetivamente entender os princípios fundamentais da orientação a objetos
- Existe uma solução de compromisso entre o uso de padrões de projeto e simplicidade (*paralysis by analysis*)
- Catálogos de padrões para domínios específicos estão disponíveis (J2EE, real-time, sistemas distribuídos, etc)
- Boas bibliotecas, tais como o Qt 4 ou as APIs do Java fazem uso intensivo de padrões e merecem ser estudadas
- Um padrão de projeto geralmente aparece em conjunto com (ou como parte de) outro padrão

INF011 – Padrões de Projeto

00 - Apresentação

Sandro Santos Andrade
sandroandrade@ifba.edu.br

Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Departamento de Tecnologia Eletro-Eletrônica
Graduação Tecnológica em Análise e Desenvolvimento de Sistemas

