Padrões de projeto estruturais

Prof. Hugo de Paula

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Curso de Engenharia de Software

Sumário

Padrões de projeto estruturais

Padrões Estruturais

Estão relacionados com a forma com que os objetos estão estruturados, principalmente em como a herança e as interfaces são aplicadas ao projeto da arquitetura do sistema.

- Em geral, envole empacotamento (*wrapping*), p. ex. uma classe que contém instâncias de outras classes. Normalmente uma classe complexa é empacotada pela por classes mais simples.
- Exemplos mais comuns: Adapter, Bridge, Decorator, Facade.
- Exemplos menos comuns (para iniciantes): Flyweight e Proxy.

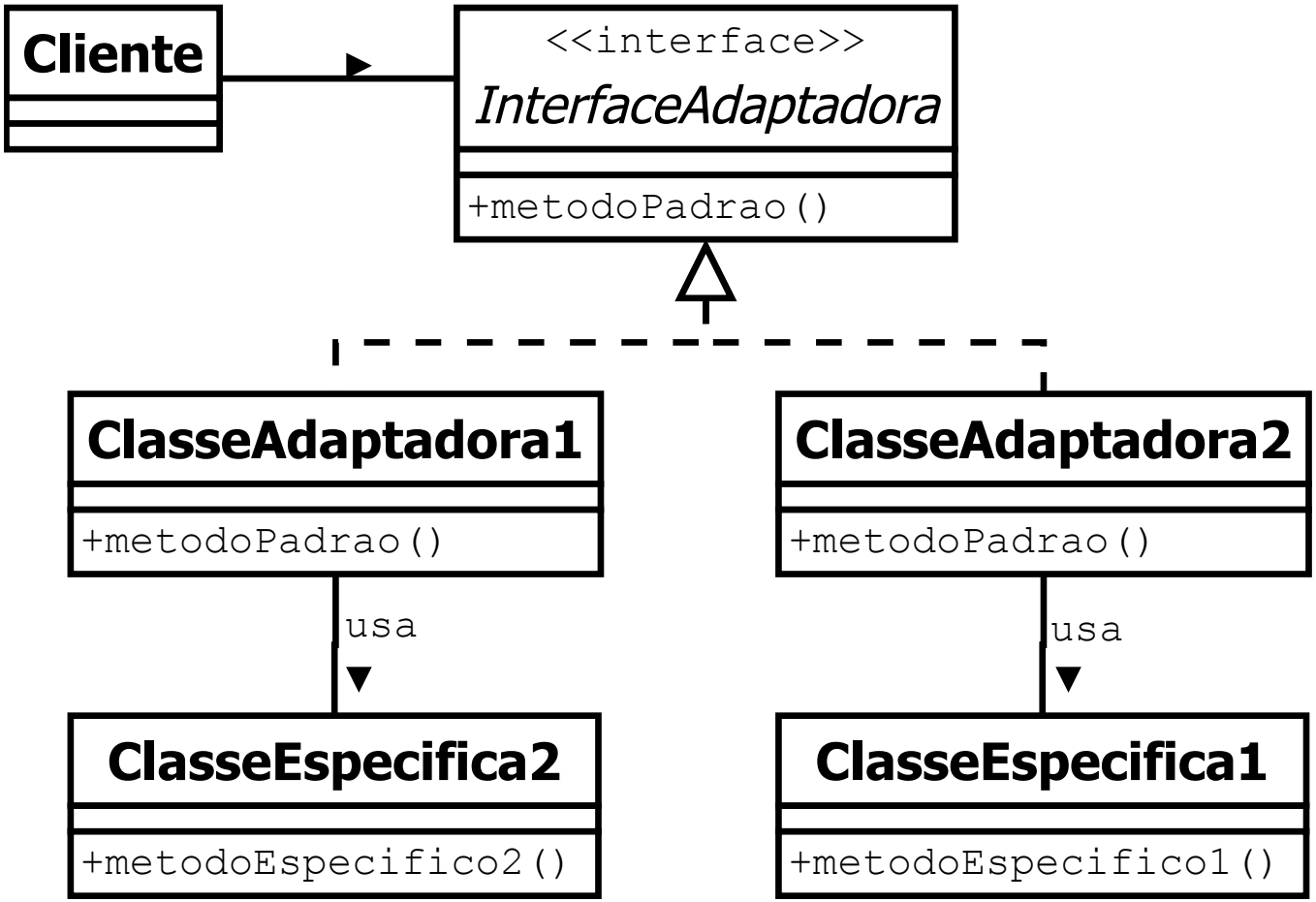
Adapter

Adapter

Fazer classes com interfaces incompatíveis trabalharem juntas.

- Funciona como uma ponte entre duas classes.
- Normalmente usada para refatorar código que teve sua funcionalidade estendida.

Adapter: solução geral



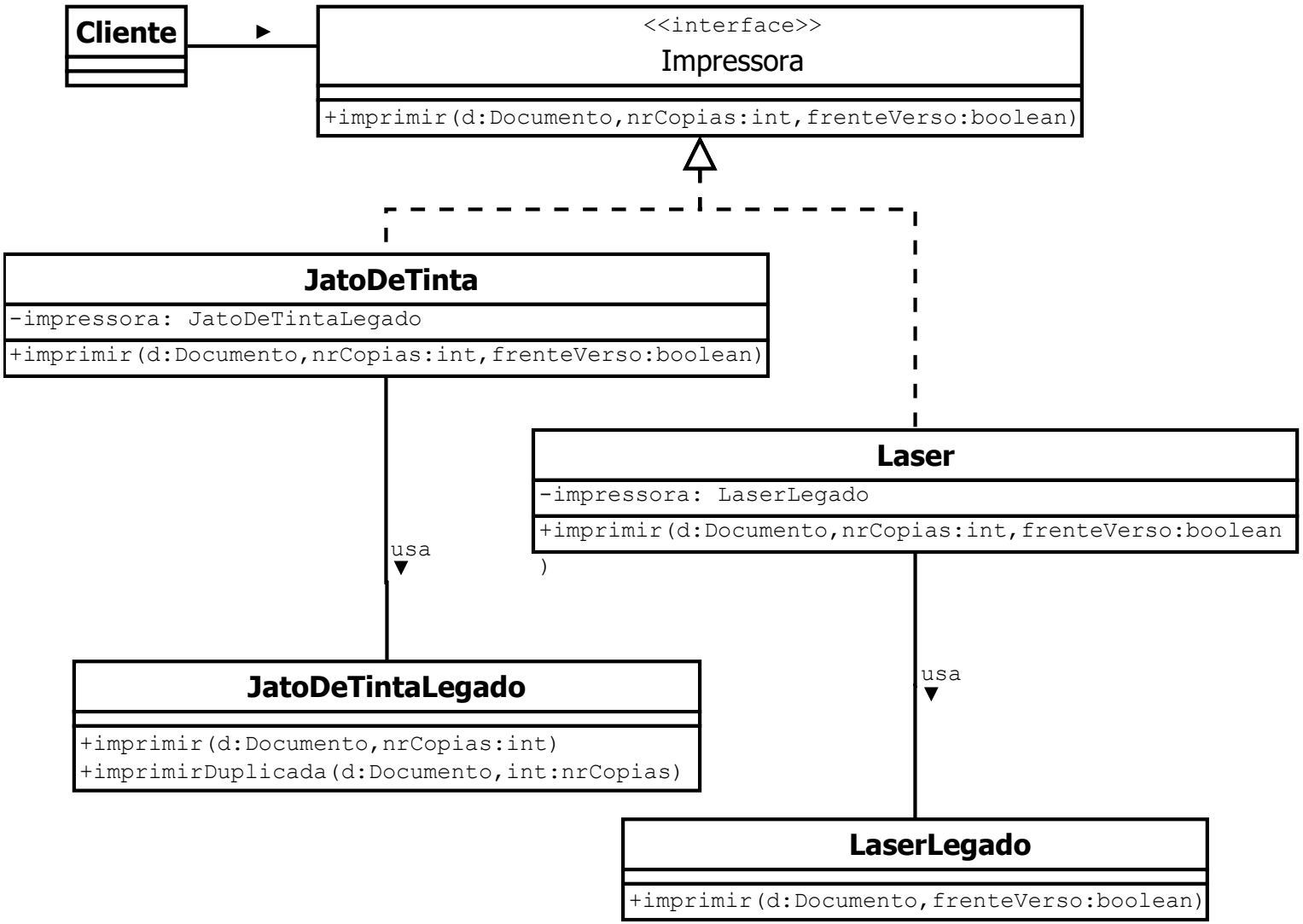
Exemplo: Adapter não polimórfico (solução ruim)

```
public class JatoDeTinta {
    public void imprimir(Documento d, int nrCopias) {
        // Codigo de impressao vai aqui
    }

    public void imprimirFrenteVerso(Documento d, int nrCopias) {
        // Codigo de impressao vai aqui
    }
}

public class Laser {
    public void imprimir(Documento d, boolean frenteVerso) {
        // Codigo de impressao vai aqui
    }
}
```

Exemplo: Adapter não polimórfico (solução ruim)

```
public class ImpressoraUnificada {
   public void imprimir(Documento d, int nrCopias, boolean frenteVerso) {
      if (isLaser()) {
         for (int copia = 1; copia <= nrCopias; copia++)
            imprimirLaser(d, frenteVerso);
      } else {
         if (frenteVerso)
            imprimirJatoDeTintaFrenteVerso(d, nrCopias);
         else
            imprimirJatoDeTinta(d, nrCopias);
      }
   }
   public void imprimirJatoDeTinta(Documento d, int nrCopias) {
      // Codigo de impressao vai aqui
      new JatoDeTinta().imprimir(d, nrCopias);
   }
   public void imprimirJatoDeTintaFrenteVerso(Documento d, int nrCopias) {
      // Codigo de impressao vai aqui
      new JatoDeTinta().imprimirDuplicada(d, nrCopias);
   }
   public void imprimirLaser(Documento d, boolean frenteVerso) {
      // Codigo de impressao vai aqui
      new Laser().imprimir(d, frenteVerso);
   }
   public boolean isLaser() {
      // Descobrir uma forma de identificar qual o tipo de impressora está ativa
      return true;
   }
}
```

Exemplo: Padrão Adapter



Exemplo: Padrão Adapter

```
public class JatoDeTintaLegado {
    public void imprimir(Documento d, int nrCopias) {
        // Codigo de impressao vem aqui.
    }

    public void imprimirFrenteVerso(Documento d, int nrCopias) {
        // Codigo de impressao vem aqui.
    }
}
public class LaserLegado {
    public void imprimir(Documento d, boolean frenteVerso) {
        // Codigo de impressao vem aqui.
    }
}
```

Exemplo: Padrão Adapter

```
public interface Impressora {
    public void imprimir(Documento d, int nrCopias, boolean frenteVerso);
}

public class JatoDeTinta implements Impressora {
    public JatoDeTintaLegado impressora = new JatoDeTintaLegado();

    @Override
    public void imprimir(Documento d, int nrCopias, boolean frenteVerso) {
        if (frenteVerso)
            impressora.imprimirFrenteVerso(d, nrCopias);
        else
            impressora.imprimir(d, nrCopias);
    }
}

public class Laser implements Impressora {
    public LaserLegado impressora = new LaserLegado();

    @Override
    public void imprimir(Documento d, int nrCopias, boolean frenteVerso) {
        for (int copy = 1; copy <= nrCopias; copy++)
            impressora.imprimir(d, frenteVerso);
    }
}
```

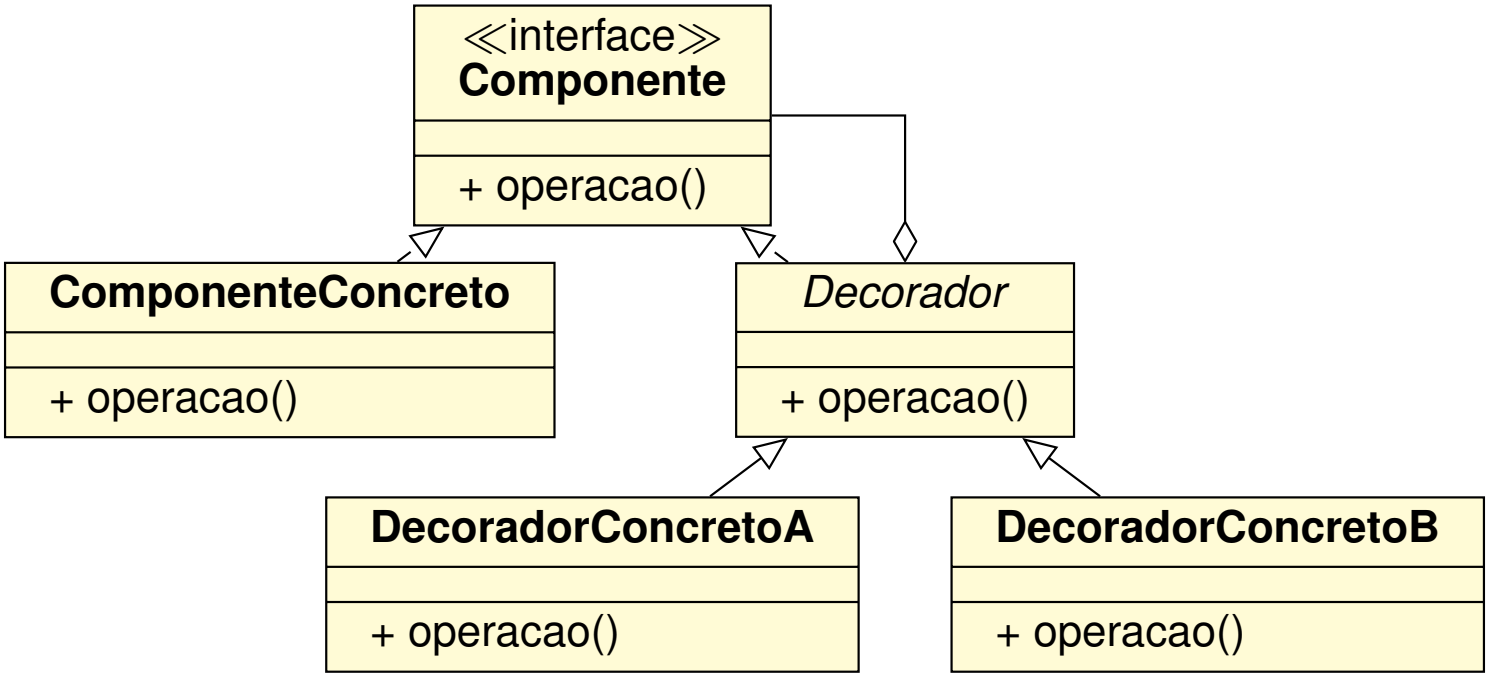
Decorator

Decorator

Adiciona nova funcionalidade a um objeto sem alterar a sua estrutura.

- Funciona como um envelope sobre uma classe.
- Classe decoradora embrulha uma classe existente e adiciona novas funcionalidades.

Decorator: solução geral



- **Componente**: interface do objeto que terá responsabilidades adicionadas dinamicamente.
- **ComponenteConcreto**: onde responsabilidades serão anexadas.
- **Decorador**: mantém referência e mimetiza interface do componente.
- **DecoradorConcreto**: adiciona responsabilidades ao componente.

Exemplo: Cálculo de acréscimos em Pizza

Cálculo de acréscimos em pizza

O objetivo é ajudar uma pizzaria a calcular o valor total de uma pizza com os acréscimos que são colocados. Usuário solicita novos ingredientes que irão gerar um custo adicional. Fonte: *Java Design Patters*, Java Code Geeks.

Exemplo: Cálculo de acréscimos em Pizza