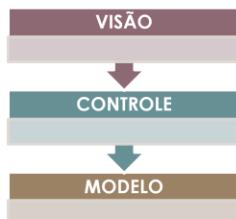
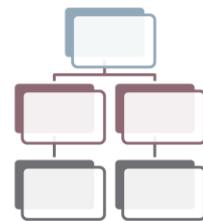


Arquitetura e Desenho de Software

AULA 14



Profa. Milene Serrano



Agenda



Considerações Iniciais

Padrões GoF

* Categoria: Estruturais

Considerações Finais

Considerações Iniciais

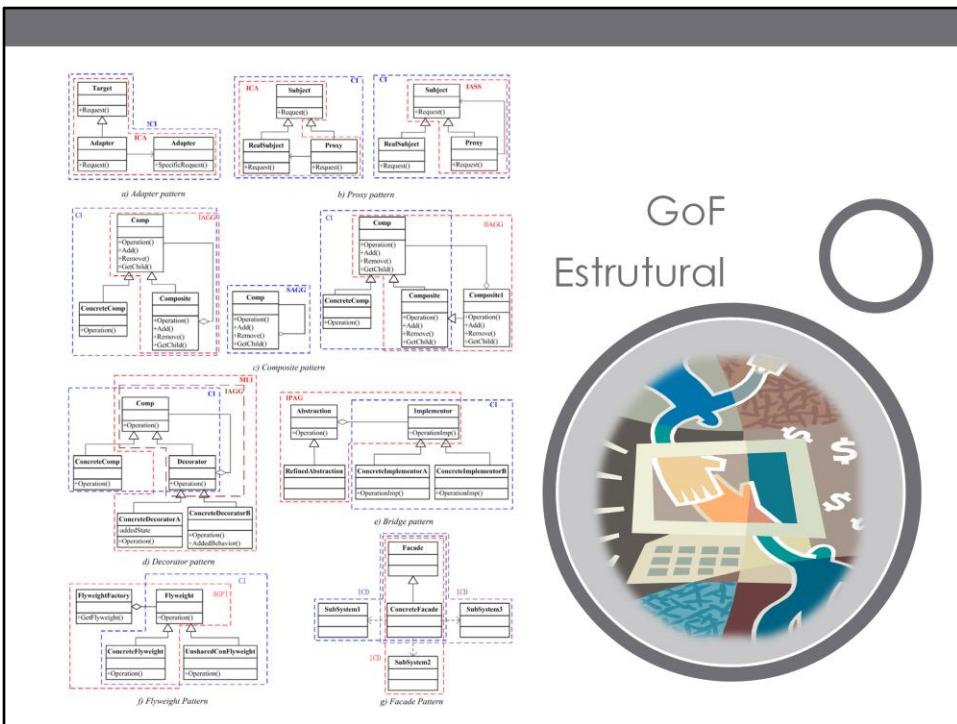


Padrões GoF

Uma solução consolidada para um problema recorrente no desenvolvimento e manutenção de software orientado a objetos.

- **Referência Relevante:** Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides. Design Patterns: Elements os Reusable Object-Oriented Software. Addison-Wesley, 1995.
- **Gang of Four - GoF**

GoF Estructural



Padrões de Projeto – GoFs Estruturais



As iterações entre os objetos de um sistema podem gerar fortes dependências entre esses elementos.



Essas dependências aumentam a complexidade das eventuais alterações no funcionamento do sistema. Consequentemente, o custo da manutenção aumenta.



Os padrões de projeto estruturais procuram diminuir o acoplamento entre os objetos de um sistema orientado a objetos. Portanto, esses padrões alteram a estrutura, principalmente, no nível de classes.

Padrões de Projeto – GoFs Estruturais

Principais Gofs Estruturais

Adapter

Bridge

Composite

Decorator

Permitir que um objeto seja substituído por outro que, apesar de realizar a mesma tarefa, possui uma interface diferente.

Separar uma abstração de sua representação, de forma que ambos possam variar e produzir tipos de objetos diferentes.

Agrupar objetos que fazem parte de uma relação parte-todo de forma a tratá-los sem distinção.

Adicionar funcionalidade a um objeto dinamicamente.

Ao final dessa aula, em **Complementar**, têm detalhes sobre vários desses padrões...

Padrões de Projeto – GoFs Estruturais

Principais Gofs Estruturais

Facade

Flyweight

Proxy

Prover uma interface simplificada capaz de centralizar a utilização de várias interfaces de um sistema.

Compartilhar, de forma eficiente, objetos que são usados em grande quantidade.

Controlar as chamadas a um objeto através de outro objeto de mesma interface.

Alguns padrões são vistos como anti-patterns (ex. Facade)...

Ao final dessa aula, em **Complementar**, têm detalhes sobre vários desses padrões...

GoF Estrutural Adapter

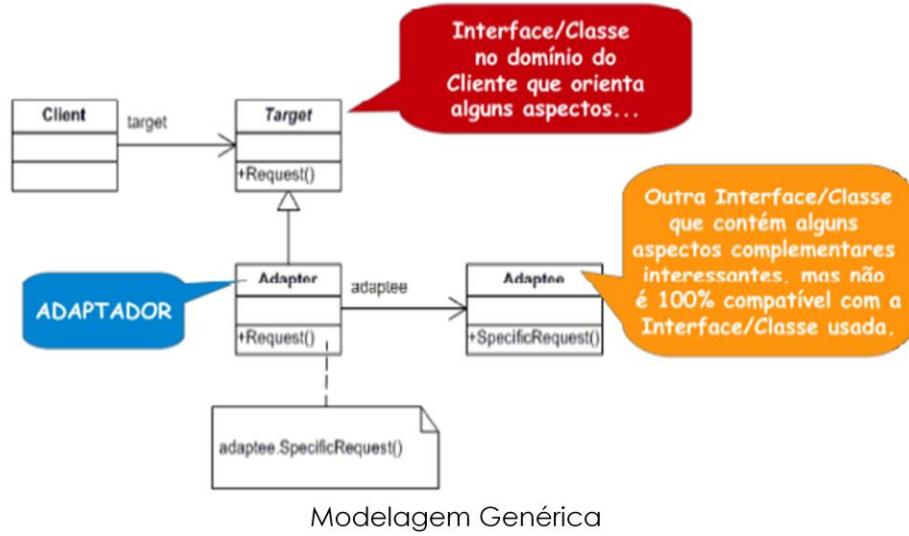


GoF Estrutural – Adapter

Objetivo

- Permitir que um objeto seja substituído por outro que, apesar de realizar a mesma tarefa, possui uma interface diferente...

GoF Estrutural – Adapter



GoF Estrutural – Adapter

Participantes

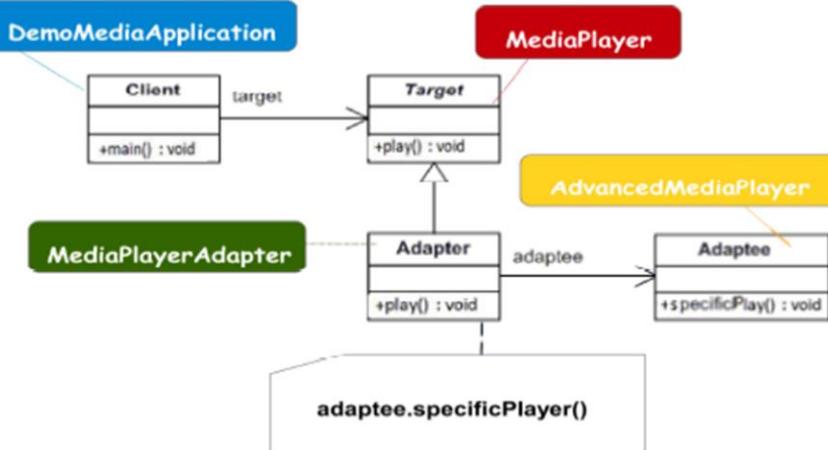
- **Target** (ex. *MediaPlayer*) – define uma interface/classe específica de domínio, a qual o *Client* usa.
- **Adapter** (ex. *MediaAdapter*) – adapta a interface/classe *Target* para que seja possível fazer uso da interface/classe *Adaptee*.

GoF Estrutural – Adapter

Participantes

- **Adaptee** (ex. *AdvancedMediaPlayer*) – define uma interface/classe existente, a qual é interessante ser usada como um complemento, por exemplo, à interface/classe *Target*.
- **Client** (ex. *DemoMediaApplication*) – Colabora com outros objetos de acordo com a interface/classe *Target*.

GoF Estrutural – Adapter



Modelagem Aplicada/Instanciada

GoF Estrutural – Adapter



GoF Estrutural – Adapter

MOTIVAÇÃO

Estamos realizando manutenção no sistema de gerenciamento de uma determinada empresa.

O controle de ponto desse sistema possui diversas limitações.

Essas limitações causam muitos prejuízos, principalmente, financeiros.

Uma empresa parceira implementou uma biblioteca Java para controlar a entrada e a saída de funcionários. Essa biblioteca não possui as limitações que existem hoje no sistema que estamos realizando manutenção.

GoF Estrutural – Adapter

MOTIVAÇÃO

Diretores decidiram que a melhor estratégia seria adquirir essa biblioteca e implantá-la no sistema.

Para implantar essa biblioteca, teremos que substituir as classes que atualmente cuidam do controle de ponto pelas classes dessa biblioteca.

A complexidade dessa substituição é alta, pois os métodos das classes antigas não são compatíveis com os métodos das classes novas.

Em outras palavras, as interfaces são diferentes.

GoF Estrutural – Adapter

MOTIVAÇÃO

Para minimizar o impacto dessa substituição no código do sistema, podemos definir classes intermediárias para adaptar as chamadas às classes da biblioteca que foi adquirida.

Adapter Pattern

(Padrão Adaptador)



GoF Estrutural – Adapter

Registrando a Entrada de um Funcionário no Sistema (atualmente)

```
PointControl pointControl = new PointControl();
```

```
Employee employee = ...
```

```
pointControl.registerAnEntry(employee);
```

```
pointControl.registerAnExit(employee);
```

Registro quanto
à saída do
funcionário!

Registro quanto
à entrada do
funcionário!

Alguns detalhes do contexto...

GoF Estrutural – Adapter

Registrando a Entrada de um Funcionário no Sistema (nova biblioteca)

```
NewPointControl newPointControl = ...  
Employee employee = ...
```

Único método
para lidar com registro
de **entrada** do funcionário
e de **saída** do mesmo!

```
// true indicates entry and false indicates exit  
newPointControl.register(employee.getCode(), true);
```

Alguns detalhes do contexto...

GoF Estrutural – Adapter

ADAPTADOR

```
public class PointControlAdapter extends PointControl {  
    private NewPointControl newPointControl;  
  
    public void registerAnEntry(Employee employee) {  
        this.newPointControl.register(employee.getCode(), true);  
    }  
}
```

Pequena adaptação e
tudo está resolvido,
compatível.

Alguns detalhes do contexto...

GoF Estrutural – Adapter

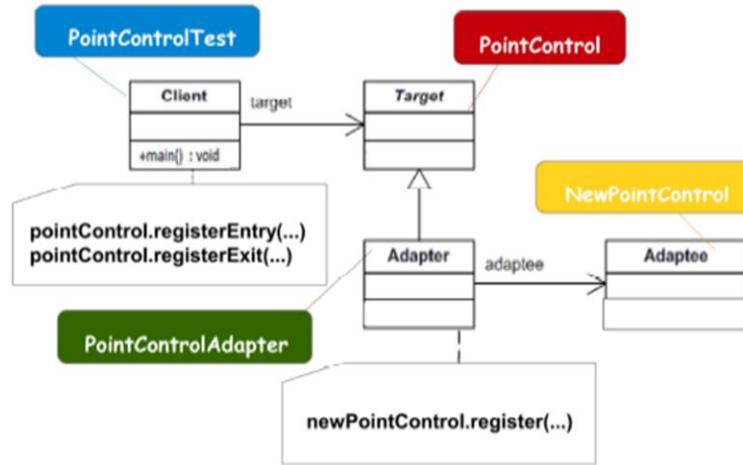
Continuamos Registrando a Entrada de um Funcionário da MESMA Forma...

```
PointControl pointControl = new PointControlAdapter();
Employee employee = ...
pointControl.registerAnEntry(employee);
```



Alguns detalhes do contexto...

GoF Estrutural – Adapter



Modelagem Aplicada/Instanciada

GoF Estrutural – Adapter

SOLUÇÃO

Podemos empregar o padrão de projeto GoF, estrutural, Adapter.

Como seria a implementação em Java?



GoF Estrutural – Adapter

```
Employee.java
package actualPointControl;

public class Employee {
    private String name;

    public Employee(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

Apenas para representar os funcionários.

Possível Implementação

GoF Estrutural – Adapter

```
PointControl.java 21
package actualPointControl;
import java.text.SimpleDateFormat;
import java.util.Calendar;

public class PointControl {
    int pointControlResponsibleID;

    public PointControl(int pointControlResponsibleID) {
        this.pointControlResponsibleID = pointControlResponsibleID;
    }

    public void registerEntry(Employee employee) {
        Calendar calendar = Calendar.getInstance();
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd/MM/yyyy H:m:s");
        String format = simpleDateFormat.format(calendar.getTime());
        System.out.println("Entrada: " + employee.getName() + " às " + format);
    }

    public void registerExit(Employee employee) {
        Calendar calendar = Calendar.getInstance();
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd/MM/yyyy H:m:s");
        String format = simpleDateFormat.format(calendar.getTime());
        System.out.println("Saída: " + employee.getName() + " às " + format);
    }
}
```

Possível Implementação

GoF Estrutural – Adapter

PointControlTest.java ATUAL

```
/*PointControlTest.java */
public class PointControlTest {
    public static void main(String[] args) throws InterruptedException {
        PointControl pointControl = new PointControl(001);
        Employee employee = new Employee("Milene Serrano");
        pointControl.registerEntry(employee);
        Thread.sleep (3000);
        pointControl.registerExit(employee);
    }
}
```

Possível Implementação

GoF Estrutural – Adapter

```
NewPointControl.java
package newLibraries;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import actualPointControl.Employee;

public class NewPointControl{
    int pointControlResponsibleID;

    public NewPointControl(int pointControlResponsibleID){
        this.pointControlResponsibleID = pointControlResponsibleID;
    }

    public void register(Employee employee, boolean isEntry){
        Calendar calendar = Calendar.getInstance();
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd/MM/yyyy HH:mm");
        String format = simpleDateFormat.format(calendar.getTime());

        if (isEntry == true){
            System.out.println("Entrada: " + employee.getName() + " às " + format);
        } else{
            System.out.println("Saída: " + employee.getName() + " às " + format);
        }
    }
}
```

Mesmo método
cuida da
entrada e
da saída.

Possível Implementação

GoF Estrutural – Adapter

```
PointControlAdapter.java
package pointControlAdapter;

import actualPointControl.Employee;
import actualPointControl.PointControl;
import newLibraries.NewPointControl;

public class PointControlAdapter extends PointControl {

    private NewPointControl newPointControl;

    public PointControlAdapter(int PointControlResponsibleID) {
        super(PointControlResponsibleID);
        this.newPointControl = new NewPointControl(001);
    }

    public void registerEntry(Employee employee) {
        this.newPointControl.register(employee, true);
    }

    public void registerExit(Employee employee) {
        this.newPointControl.register(employee, false);
    }
}
```

Adaptação

Adaptação

Possível Implementação

GoF Estrutural – Adapter

MODIFICADA!

PointControlTest.java

```
PointControlTest.java X
package actualPointControl;

import pointControlAdapter.PointControlAdapter;

public class PointControlTest {

    public static void main(String[] args) throws InterruptedException{
        PointControl pointControl = new PointControlAdapter(001);
        Employee employee = new Employee("Milene Serrano");
        pointControl.registerEntry(employee);
        Thread.sleep (3000);
        pointControl.registerExit(employee);
    }
}
```

Pequeno Impacto...

Lembrando:
um padrão sempre
opta por minimizar os
impactos na cliente,
não necessariamente
zerar esses impactos...

Possível Implementação

Zerar os impactos na cliente pode incorrer em muitas novas dependências (ex. uso de uma combinação de padrões).

Dessa forma, torna-se uma solução bem mais complexa, o que pode não ser interessante.

GoF Estrutural – Adapter

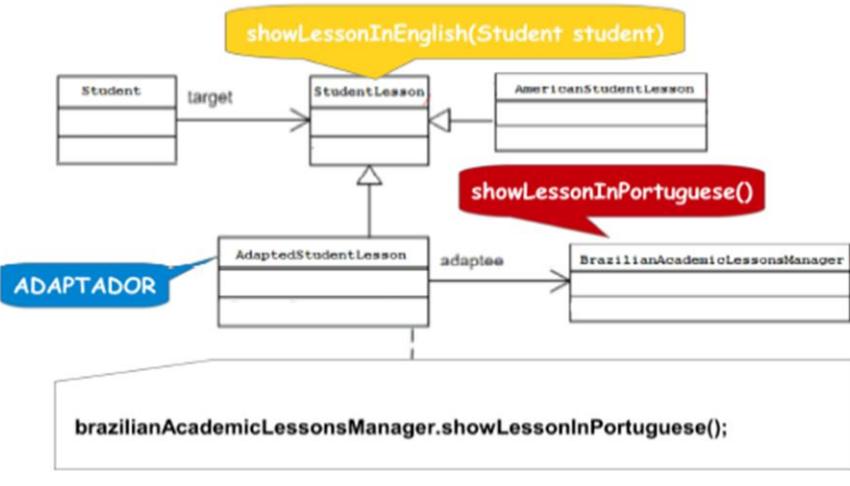
Executando...

```
Problems @ Javadoc Declaration Console X
<terminated> PointControlTest [Java Application] C:\Program Files\Java\
Entrada: Milene Serrano às 03/05/2014 22:50:4
Saida: Milene Serrano às 03/05/2014 22:50:7
```

O uso da Thread.sleep(3000) permite
observarmos mudanças nos registros
dos horários de entrada e saída. Ok?

Possível Implementação

GoF Estrutural – Adapter



Mais um exemplo de aplicação de *adapter*...

GoF Estrutural
Composite

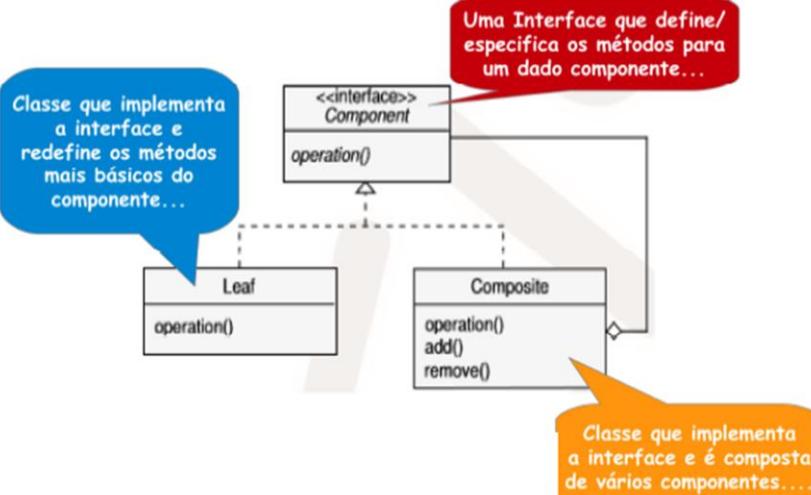


GoF Estrutural – Composite

Objetivo

- Agrupar objetos que fazem parte de uma relação parte-todo de forma a tratá-los sem distinção.

GoF Estrutural – Composite



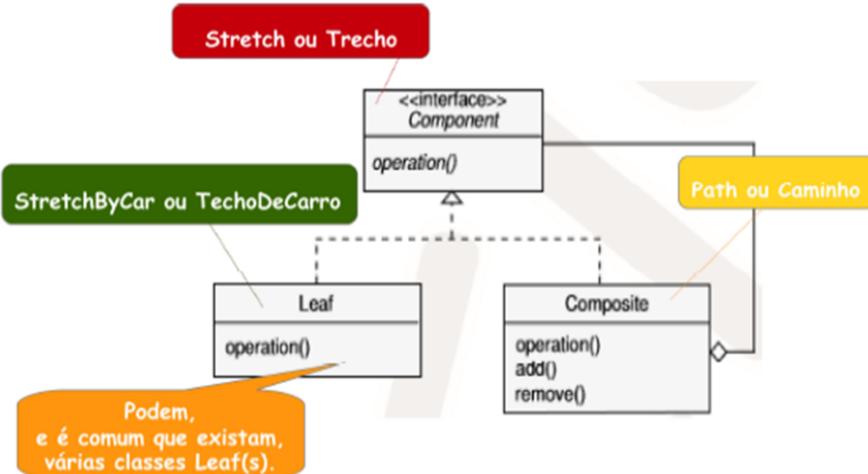
Modelagem Genérica

GoF Estrutural – Composite

Participantes

- **Component** (ex. *Trecho*) – interface que define os elementos da composição.
- **Leaf** (ex. *TrechoDeCarro*, *TrechoAndando*) – define os elementos básicos da composição. Isto é, aqueles que não são formados por outros Components.
- **Composite** (ex. *Caminho*) – define os Components que são formados por outros Components.

GoF Estrutural – Composite



Modelagem Aplicada/Instaciada

GoF Estrutural – Composite

MOTIVAÇÃO

Suponha que estejamos desenvolvendo um sistema para calcular um caminho entre quaisquer dois pontos do mundo.

Um caminho pode ser percorrido de diversas maneiras: a pé, de carro, de ônibus, de trem, de avião, de navio, dentre outras formas.

GoF Estrutural – Composite

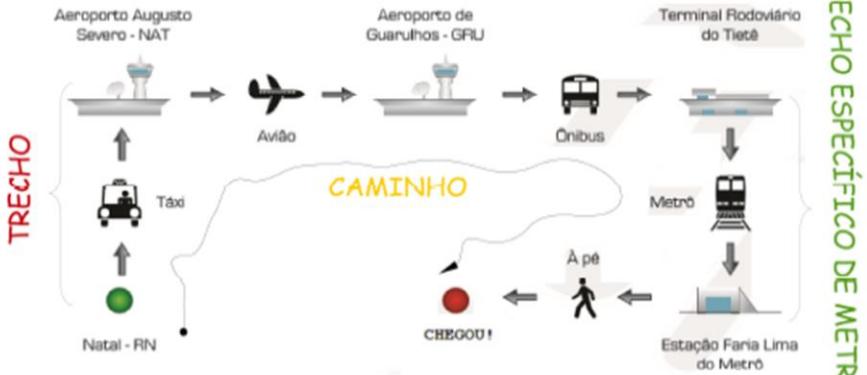


Ilustração: Um TRECHO ESPECÍFICO é um TRECHO.
Um único TRECHO é um CAMINHO,
ou seja, um CAMINHO com um única PARTE (i.e. TRECHO)

GoF Estrutural – Composite

MOTIVAÇÃO

O sistema deve apresentar graficamente para os usuários os caminhos que forem calculados.

Cada tipo de trecho deve ser apresentado de uma maneira específica. Por exemplo, se o trecho for de caminhada, então, deve aparecer na impressão do caminho a ilustração de uma pessoa andando.

Cada tipo de trecho pode ser implementado por uma classe, e seria interessante definir uma interface para padronizá-las.

GoF Estrutural – Composite

SOLUÇÃO

Podemos empregar o padrão de projeto GoF, estrutural, *Composite*.

Como seria a implementação em Java?



GoF Estrutural – Composite

```
Stretch.java
public interface Stretch {
    public void printStretch();
}
```

Possível Implementação

GoF Estrutural – Composite

```
J StretchByFoot.java X
public class StretchByFoot implements Stretch {

    private String direction;
    private double distance;

    public StretchByFoot(String direction , double distance){
        this.direction = direction;
        this.distance = distance;
    }

    public void printStretch(){
        System.out.println("Go by Foot: ");
        System.out.println(this.direction);
        System.out.println("The completed distance was: " + this.distance + " meters. <_ _");
    }
}
```



Imagen relacionada
ao fato de
ser andando...
:)

Possível Implementação

GoF Estrutural – Composite

StretchByCar.java

```
public class StretchByCar implements Stretch {  
  
    private String direction;  
    private double distance;  
  
    public StretchByCar(String direction , double distance){  
        this.direction = direction;  
        this.distance = distance;  
    }  
  
    public void printStretch(){  
        System.out.println("Go by Car: ");  
        System.out.println(this.direction);  
        System.out.println("The completed distance was: " + this.distance + " meters. <- O-O");  
    }  
}
```



Imagen relacionada
ao fato de
ser de carro...
:)

Possível Implementação

GoF Estrutural – Composite

StretchBySubway.java

```
public class StretchBySubway implements Stretch {  
  
    private String direction;  
    private double distance;  
  
    public StretchBySubway(String direction , double distance){  
        this.direction = direction;  
        this.distance = distance;  
    }  
  
    public void printStretch(){  
        System.out.println("Go by Subway: ");  
        System.out.println(this.direction);  
        System.out.println("The completed distance was: " + this.distance + " meters. <- =====");  
    }  
}
```

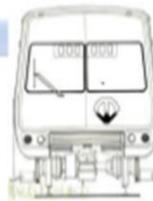


Imagen relacionada
ao fato de
ser de metrô...
:)

Possível Implementação

GoF Estrutural – Composite

```
Path.java 33  
import java.util.ArrayList;  
import java.util.List;  
  
public class Path implements Stretch{  
    private List <Stretch> stretches = new ArrayList <Stretch>();  
  
    public void addStretch(Stretch stretch){  
        this.stretches.add(stretch);  
    }  
  
    public void removeStretch(Stretch stretch){  
        this.stretches.remove(stretch);  
    }  
  
    @Override  
    public void printStretch(){  
        for (Stretch stretch : this.stretches){  
            stretch.printStretch();  
        }  
    }  
}
```

Possível Implementação

GoF Estrutural – Composite

```
*CompositeTest.java */
public class CompositeTest {

    public static void main(String[] args) {
        Stretch stretch01 = new StretchByFoot("Go to the corner of Sherbourne Street and Dundas Street.", 500);
        Stretch stretch02 = new StretchByCar("Continue straight on Dundas Street.", 1500);
        Stretch stretch03 = new StretchByCar("Turn right when you see a big Supermarket.", 500);
        Stretch stretch04 = new StretchBySubway("Try to park at the supermarket, corner of Dundas Street and Oliver Street." +
            "In Continue by using the subway, from Oliver's station to University's station.", 1000);

        Path path1 = new Path();
        path1.addStretch(stretch01);
        path1.addStretch(stretch02);
        System.out.println("First Path: ");
        System.out.println("Printing the path, composed by different stretches...");
        path1.printStretch();
        ...
    }
}
```

Possível Implementação

GoF Estrutural – Composite

```
J "CompositeTest.java" 32
    ...
    Path path2 = new Path();
    path2.addStretch(path1);
    path2.addStretch(stretch03);
    System.out.println("-----");
    System.out.println("Second Path: ");
    System.out.println("Printing the path, composed by different stretches...");
    path2.printStretch();

    Path path3 = new Path();
    path3.addStretch(path2);
    path3.addStretch(stretch04);
    System.out.println("-----");
    System.out.println("Third Path: ");
    System.out.println("Printing the path, composed by different stretches...");
    path3.printStretch();
}
}
```

Possível Implementação

GoF Estrutural – Composite

```
Problems Javadoc Declaration Console
<terminated> CompositeTest [Java Application] C:\Program Files\Java\jre7\bin\javaw
First Path:
Printing the path, composed by different stretches...
Go by Foot:
Go to the corner of Sherbourne Street and Dundas Street.
The completed distance was: 500.0 meters. <- _ _
Go by Car:
Continue straight on Dundas Street.
The completed distance was: 1500.0 meters. <- O-O
-----
Second Path:
Printing the path, composed by different stretches...
Go by Foot:
Go to the corner of Sherbourne Street and Dundas Street.
The completed distance was: 500.0 meters. <- _ _
Go by Car:
Continue straight on Dundas Street.
The completed distance was: 1500.0 meters. <- O-O
Go by Car:
Turn right when you see a big Supermarket.
The completed distance was: 500.0 meters. <- U-U
-----
```

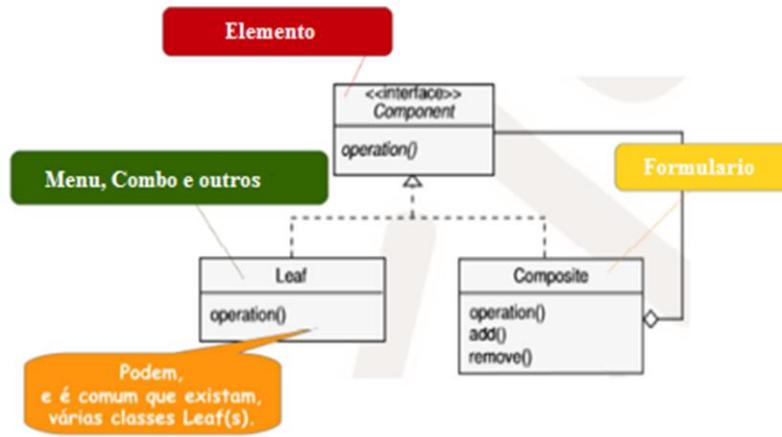
Possível Implementação

GoF Estrutural – Composite

```
Problems Javadoc Declaration Console 
<terminated> CompositeTest [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe
...
Third Path:
Printing the path, composed by different stretches...
Go by Foot:
Go to the corner of Sherbourne Street and Dundas Street.
The completed distance was: 500.0 meters. <- _ _
Go by Car:
Continue straight on Dundas Street.
The completed distance was: 1500.0 meters. <- O-O
Go by Car:
Turn right when you see a big Supermarket.
The completed distance was: 500.0 meters. <- O-O
Go by Subway:
Try to park at the supermarket, corner of Dundas Street and Oliver Street.
Continue by using the subway, from Oliver's station to University's station.
The completed distance was: 1000.0 meters. <- ===
```

Possível Implementação

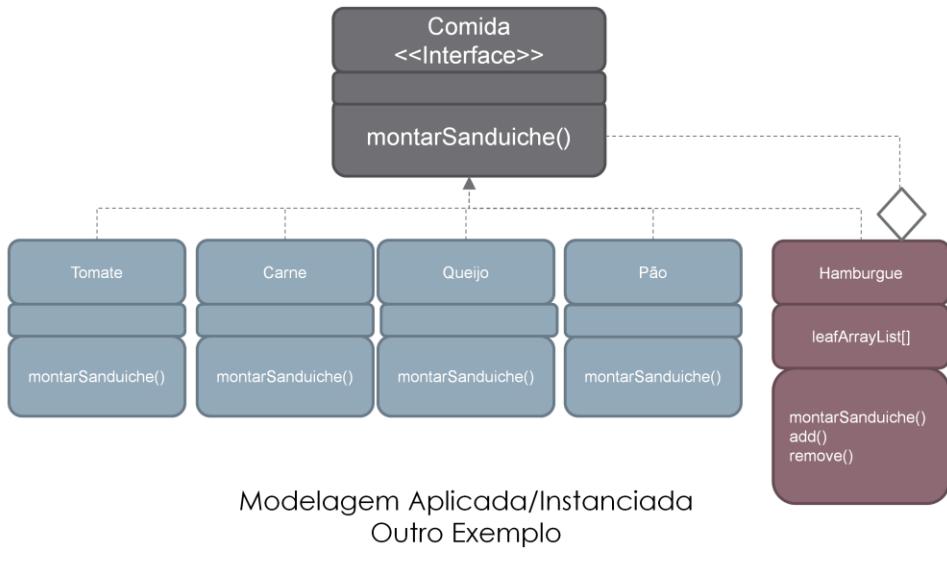
GoF Estrutural – Composite



Modelagem Aplicada/Instaciada
Outro Exemplo

Mais um exemplo de aplicação de *composite*...

GoF Estrutural – Composite



Mais um exemplo de aplicação de *composite*...

Extra Classe



Extra Classe

Construam um catálogo de padrões, disponibilizando-o em um repositório. Assim, irão praticar a implementação dos demais padrões criacionais e estruturais. Logo, poderão acrescentar ao repositório os padrões comportamentais. :)

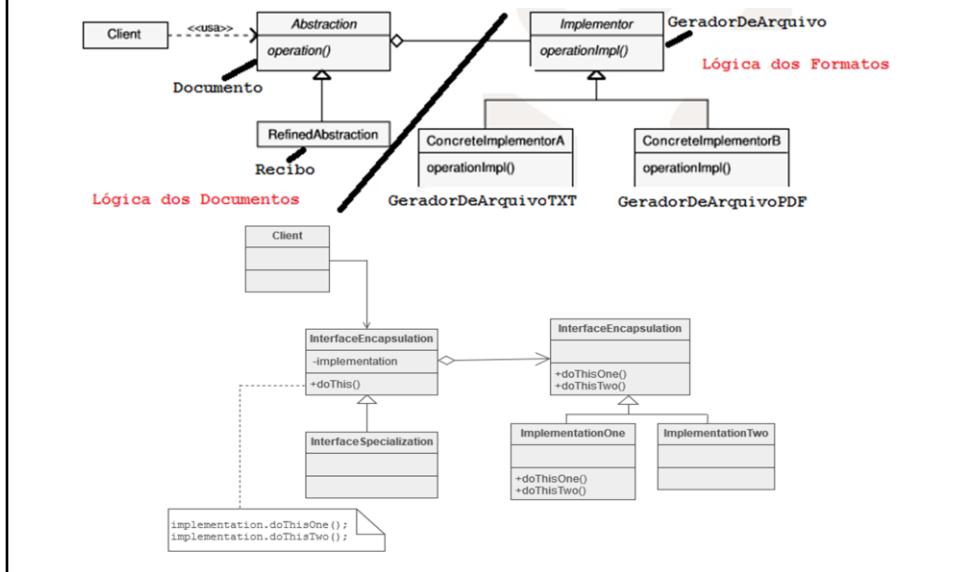
Complementar



Complementar

- Seguem *slides* com os demais padrões GoFs estruturais.
- Procurem implementá-los.
- Qualquer dúvida, entrem em contato comigo ou com os nossos monitores.

Bridge - Modelagem



Possíveis modelagens...

Ilustrado com *Client*, *Documento*, *Recibo*, *GeradorDeArquivo*, *GeradorDeArquivoTXT* e *GeradorDeArquivoPDF*.

Poderia ser, conforme colocado nas implementações disponíveis no próximo slide: *BridgeDisc*, *Node*, *StackArray*, *StackList*, *StackFIFO* e *StackHanoi*.

Ou:

BridgeDisk, *Node*, *Stack*, *StackHanoi*, *StackFIFO*, *StackImpl* e *StackList*.

Ou:

BridgeDemo, *Stack*, *StackHanoi*, *StackImpl* e *StackMine*.

Ou outro uso, em outro domínio, aplicado de forma similar. Ok?

Bridge - Implementação

- Por gentileza, consultem:

https://sourcemaking.com/design_patterns/bridge/java/1

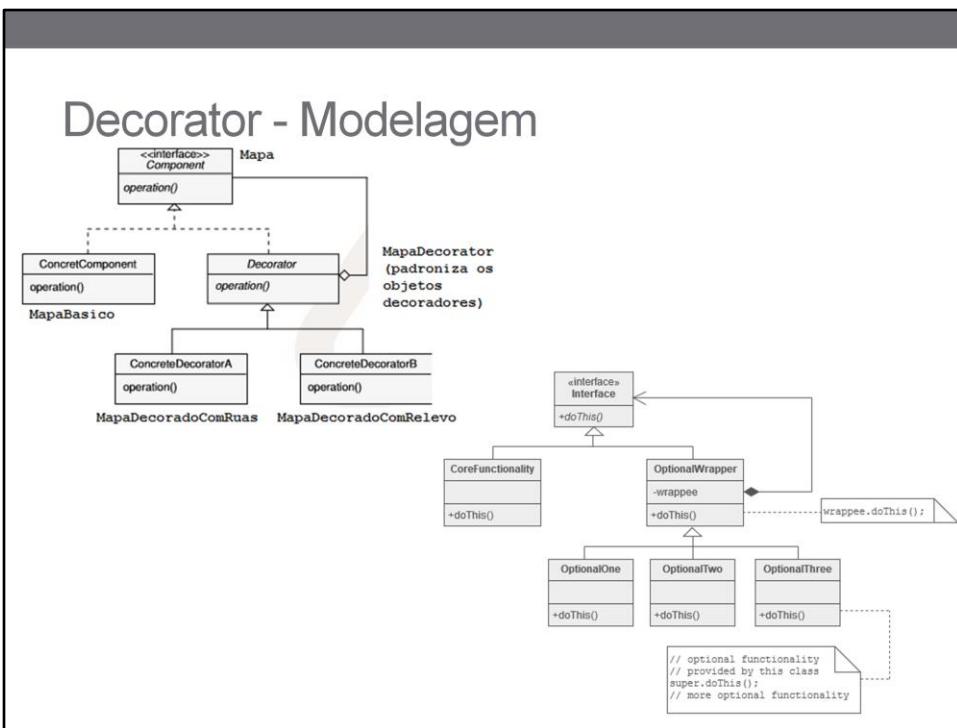
https://sourcemaking.com/design_patterns/bridge/java/2

https://sourcemaking.com/design_patterns/bridge/java/3

- Reparem que têm, ao final do artigo, outros *links* para implementações desse padrão em outras linguagens.

Possível implementação...

Naveguem pelo *site*, pois tem a implementação desse padrão em outras linguagens...



Possíveis modelagens...

Ilustrado com *Mapa*, *MapaBasico*, *MapaDecorator*, *MapaDecoradoComRuas* e *MapaDecoradoComRelevo*.

Poderia ser, conforme colocado nas implementações disponíveis no próximo slide...

Ou outro uso, em outro domínio, aplicado de forma similar. Ok?

Decorator - Implementação

- Por gentileza, consultem:

https://sourcemaking.com/design_patterns/decorator/java/1

https://sourcemaking.com/design_patterns/decorator/java/2

https://sourcemaking.com/design_patterns/decorator/java/3

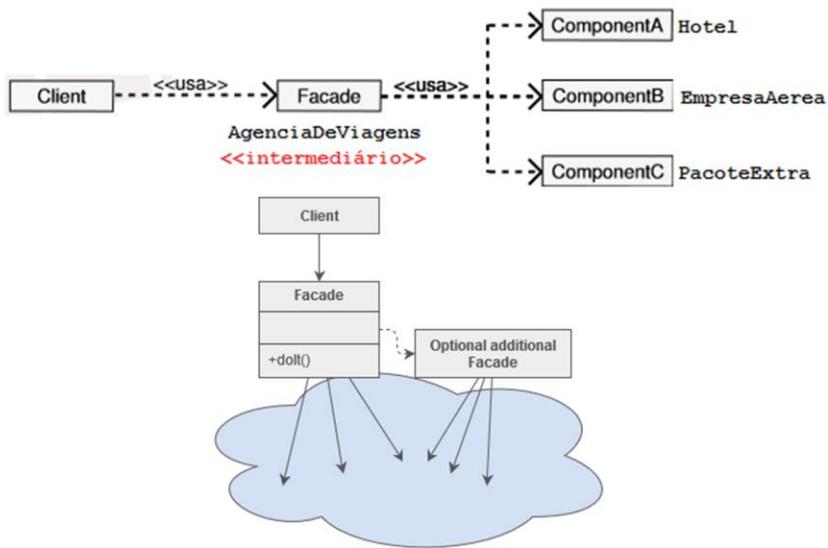
https://sourcemaking.com/design_patterns/decorator/java/4

- Reparem que têm, ao final do artigo, outros *links* para implementações desse padrão em outras linguagens.

Possível implementação...

Naveguem pelo *site*, pois tem a implementação desse padrão em outras linguagens...

Facade - Modelagem



Possíveis modelagens...

Ilustrado com *AgenciaDeViagens*, *Hotel*, *EmpresaAerea* e *PacoteExtra*.

Poderia ser, conforme colocado na implementação disponível no próximo slide:
ComputadorFacade, *Cpu*, *Memoria* e *HardDrive*.

Ou outro uso, em outro domínio, aplicado de forma similar. Ok?

Facade - Implementação

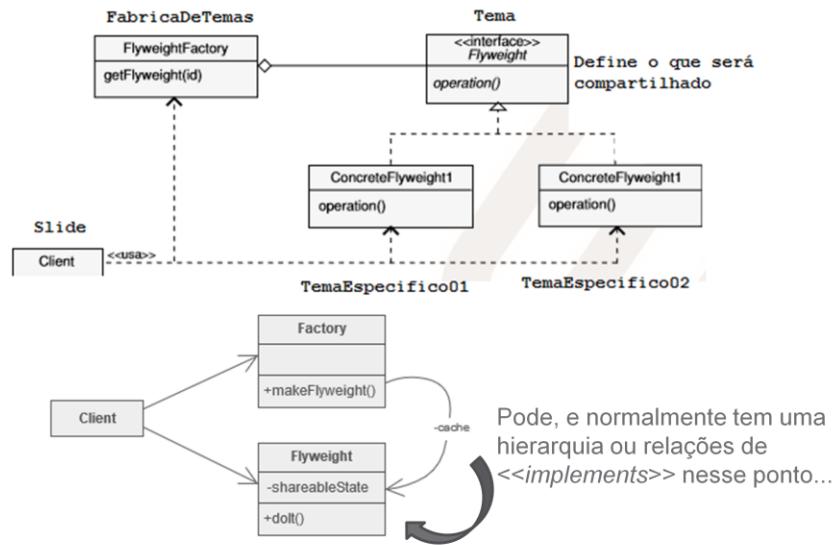
- Por gentileza, consultem:

<http://www.devmmedia.com.br/padrao-de-projeto-facade-em-java/26476>

- Reparem que têm, ao final do artigo, outros *links* para implementações desse padrão em outras linguagens.

Possível implementação...

Flyweight - Modelagem



Possíveis modelagens...

Ilustrado com *Client*, *FabricaDeTemas* <<Flyweight Factory>>, *Tema* <<Flyweight>>, *TemaEspecifico01* e *TemaEspecifico02*.

Poderia ser, conforme colocado na implementação disponível no próximo slide: *FlyweightDemo*, *FlyweightFactory* e *ButtonListener*.

Ou outro uso, em outro domínio, aplicado de forma similar. Ok?

Flyweight - Implementação

- Por gentileza, consultem:

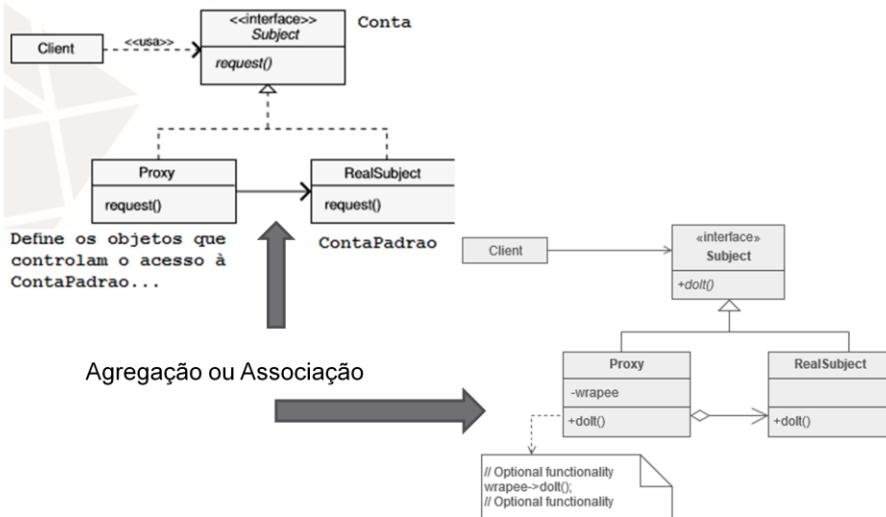
https://sourcemaking.com/design_patterns/flyweight/java/2

- Reparem que têm, ao final do artigo, outros *links* para implementações desse padrão em outras linguagens.

Possível implementação...

Naveguem pelo *site*, pois tem a implementação desse padrão em outras linguagens...

Proxy - Modelagem



Possíveis modelagens...

Ilustrado com *Client*, *Conta*, *Proxy* e *ContaPadrao*.

Poderia ser, conforme colocado na implementação disponível no próximo slide:
ProxyDemo, *SocketInterface* e *SocketProxy*.

Ou outro uso, em outro domínio, aplicado de forma similar. Ok?

Proxy - Implementação

- Por gentileza, consultem:

https://sourcemaking.com/design_patterns/proxy/java/1

- Reparem que têm, ao final do artigo, outros *links* para implementações desse padrão em outras linguagens.

Possível implementação...

Naveguem pelo *site*, pois tem a implementação desse padrão em outras linguagens...

Considerações Finais



Considerações Finais

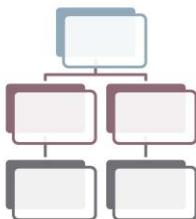
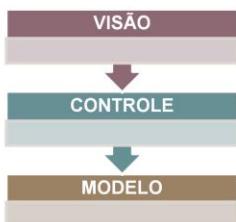
- Nessa aula, conhecemos os padrões GoFs estruturais.
- Continuem os estudos! Só se aprende praticando!
- Nas referências, têm vários materiais complementares! :)

Referências



Referências

- LARMAN, Craig. Utilizando UML e Padrões: Uma Introdução a Análise e ao Projeto
- Orientado a Objetos. 3a. edição. Bookman, 2007.
- COCKBURN, Alistair. Escrevendo Casos de Uso Eficazes. Bookman, 2005.
- SILVA, Ricardo Pereira. UML 2 em Modelagem Orientada a Objetos. Visual Books, 2007.
- PRESSMAN, Roger S. Engenharia de Software. 6a. edição . McGraw-Hill, 2006.
- IEEE. SWEBOK-Guide to the Software Engineering Body of Knowledge, 2004.
- SOMMERVILLE, Ian. Engenharia de Software. 8a. edição. Pearson, 2007.



FIM

Dúvidas?

CONTATO:
mileneserrano@unb.br
ou
mileneserrano@gmail.com

Sugestões?

