

Faça

Prof. Igor Avila Pereira
igor.pereira@riogrande.ifrs.edu.br

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS)
Campus Rio Grande
Divisão de Computação

Agenda

- 1 Introdução
 - Propósito
 - Motivação
- 2 Aplicabilidade
- 3 Exemplo
 - Exemplo: Façade
- 4 Um pouco de teoria
- 5 Consequências
- 6 UML
- 7 Façade vs Proxy
- 8 Trabalho

Introdução

Vamos aprender sobre um padrão de projeto (design pattern) que você provavelmente já implementou sem dar um nome para isso.

Trata-se de do padrão de projeto **Façade** (traduzido ao pé da letra fica como: **Fachada**)

Introdução

A intenção do padrão Façade é:

Fornecer uma interface unificada para um conjunto de interfaces em um subsistema. Façade define uma interface de nível mais alto que torna o subsistema mais fácil de utilizar.

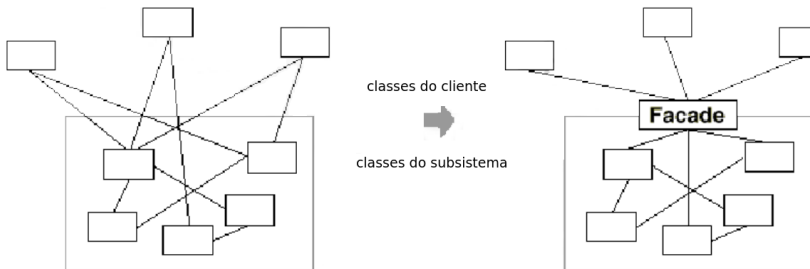
Introdução

- **Propósito:**
 - Disponibilizar uma interface unificada para um conjunto de interfaces de um subsistema
- **Motivação:**
 - Estruturar um sistema em subsistemas ajuda a reduzir complexidade
 - Geralmente deseja-se minimizar as comunicações e dependências entre subsistemas
 - O Façade pode ser utilizado para este objetivo

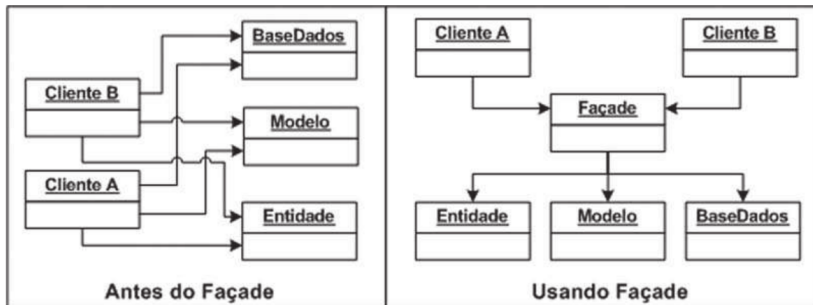
Introdução
Aplicabilidade
Exemplo
Um pouco de teoria
Consequências
UML
Façade vs Proxy
Trabalho

Próposito
Motivação

Motivação



Motivação



Aplicabilidade

- **Aplicabilidade:**

- Deseja-se disponibilizar uma forma de acesso (interface) simples a um subsistema complexo:
 - À medida em que evoluem e utilizam mais padrões de projeto, os sistemas passam a ser formados por um número maior de classes, geralmente pequenas
 - Isso torna o sistema mais reutilizável e fácil de configurar mas também o torna mais difícil de ser utilizado por clientes que não necessitam configurá-lo
 - O Façade disponibiliza uma visão default simples do sistema, suficiente para a maioria dos clientes
 - Somente aqueles clientes que precisam de uma maior capacidade de configuração irão acessar o subsistema sem utilizar o Façade

Colaborações

- **Colaborações:**

- Clientes se comunicam com o subsistema através de requisições enviadas ao Façade, que as repassa para os objetos do subsistema apropriados
- Embora sejam os objetos do subsistema que realizem o trabalho o Façade pode precisar realizar algum processamento próprio para traduzir sua interface nas interfaces dos objetos do subsistema
- Clientes que utilizam o Façade não têm acesso direto aos objetos do subsistema

Exemplo

- Para reproduzir um determinado som é utilizado o subsistema de Audio, que normalmente provê funcionalidades desde a configuração da reprodução de áudio, até a reprodução de um determinado arquivo.
- Antes de iniciar um filme, o é necessário realizar ajustes em todos os subsistemas que serão utilizados, por exemplo, é necessário configurar a resolução do subsistema de Video para que este possa renderizar imagens corretamente.

Exemplo

Para exemplificar veja as interfaces da seguintes classes, que representa o subsistema de Audio:

```
1  public class SistemaDeAudio {  
2  
3      public void configurarFrequencia() {  
4          System.out.println("Frequencia configurada");  
5      }  
6  
7      public void configurarVolume() {  
8          System.out.println("Volume configurado");  
9      }  
10  
11     public void configurarCanais() {  
12         System.out.println("Canais configurados");  
13     }  
14  
15     public void reproduzirAudio(String arquivo) {  
16         System.out.println("Reproduzindo: " + arquivo);  
17     }  
18 }
```

Exemplo

Como falado ela fornece os métodos para configuração e reprodução de arquivos de áudio.

- Para reproduzir um arquivo de áudio, por exemplo, seria necessário realizar as seguintes operações:

```
1  public static void main(String[] args) {  
2      System.out.println("##### Configurando subsistemas #####");  
3      SistemaDeAudio audio = new SistemaDeAudio();  
4      audio.configurarCanais();  
5      audio.configurarFrequencia();  
6      audio.configurarVolume();  
7  
8      System.out.println("##### Utilizando subsistemas #####");  
9      audio.reproduzirAudio("teste.mp3");  
10 }
```

Exemplo

Neste exemplo de código cliente, o próprio cliente deve instanciar e configurar o subsistema para que só depois seja possível a utilização dos mesmos.

- Além disso, existe um comportamento padrão que é executado antes de reproduzir um som: sempre deve ser configurado o canal, a frequência e o volume.

Agora pense como seria caso fosse necessário utilizar vários subsistemas? O código cliente ficaria muito sobrecarregado com responsabilidades que não são dele.

Exemplo

```
1 public static void main(String[] args) {
2     System.out.println("##### Configurando subsistemas #####");
3     SistemaDeAudio audio = new SistemaDeAudio();
4     audio.configurarCanais();
5     audio.configurarFrequencia();
6     audio.configurarVolume();
7
8     SistemaDeInput input = new SistemaDeInput();
9     input.configurarTeclado();
10    input.configurarJoystick();
11
12    SistemaDeVideo video = new SistemaDeVideo();
13    video.configurarCores();
14    video.configurarResolucao();
15
16    System.out.println("##### Utilizando subsistemas #####");
17    audio.reproduzirAudio("teste.mp3");
18    input.lerInput();
19    video.renderizarImagem("imagem.png");
20 }
```

Exemplo: Façade

A intenção do padrão:

Fornecer uma interface unificada para um conjunto de interfaces em um subsistema. Façade define uma interface de nível mais alto que torna o subsistema mais fácil de ser usado.

Pela intenção é possível notar que o padrão pode ajudar bastante na resolução do nosso problema. O conjunto de interfaces seria exatamente o conjunto de subsistemas.

Exemplo: Façade

- Nesse sentido o Façade vai definir operações a serem realizadas com estes subsistemas.
- Assim, é possível definir uma operação padrão para configurar o subsistema de áudio, evitando a necessidade de chamar os métodos de configuração de áudio a cada novo arquivo de áudio que precise ser reproduzido.
- A utilização do padrão Façade é bem simples: apenas é necessário criar a classe fachada que irá se comunicar com os subsistemas no lugar no cliente

Exemplo: Façade

```
1 public class SistemasFacade {
2     protected SistemaDeAudio audio;
3     protected SistemaDeInput input;
4     protected SistemaDeVideo video;
5
6     public void inicializarSubsistemas() {
7         video = new SistemaDeVideo();
8         video.configurarCores();
9         video.configurarResolucao();
10
11         input = new SistemaDeInput();
12         input.configurarJoystick();
13         input.configurarTeclado();
14
15         audio = new SistemaDeAudio();
16         audio.configurarCanais();
17         audio.configurarFrequencia();
18         audio.configurarVolume();
19     }
20
21     public void reproduzirAudio(String arquivo) {
22         audio.reproduzirAudio(arquivo);
23     }
24
25     public void renderizarImagem(String imagem) {
26         video.renderizarImagem(imagem);
27     }
28 }
```

Exemplo: Façade

- A classe fachada realiza a inicialização de todos os subsistemas e oferece acesso aos métodos necessários, por exemplo o método de renderização de uma imagem, a reprodução de um áudio.
- Com esta mudança, tiramos toda a responsabilidade do cliente, que agora precisa se preocupar apenas em utilizar os subsistemas que desejar.

Exemplo: Façade

```
1 public static void main(String[] args) {  
2     System.out.println("##### Configurando subsistemas #####");  
3     SistemasFacade fachada = new SistemasFacade();  
4     fachada.inicializarSubsistemas();  
5  
6     System.out.println("##### Utilizando subsistemas #####");  
7     fachada.renderizarImagem("imagem.png");  
8     fachada.reproduzirAudio("teste.mp3");  
9     fachada.lerInput();  
10 }
```

Um pouco de teoria

- Note que as classes do subsistema continuam sendo visíveis em todo o projeto.
 - Portanto, caso seja necessário, o cliente pode definir suas configurações sem sequer utilizar a classe fachada.
- Ainda mais, o cliente pode criar uma fachada própria, que define suas operações customizadas.
 - Por exemplo, se não serão utilizados joysticks no projeto, não há necessidade de inicializá-los
- O problema com essa centralização da complexidade é que a classe fachada pode crescer descontroladamente para abrigar um conjunto grande de possibilidades.

Um pouco de teoria

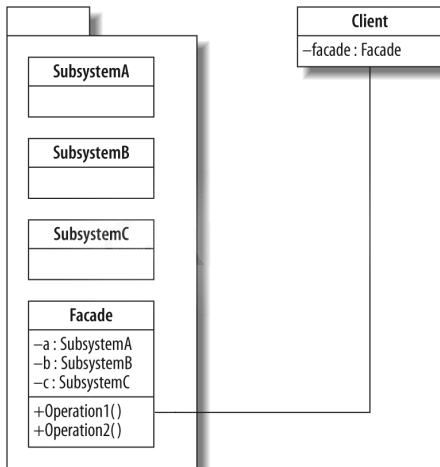
- Existem algumas semelhanças, fáceis de serem notadas, deste padrão com outros já discutidos.
 - Por exemplo, já que o Façade define uma interface, qual a sua diferença em relação ao padrão Adapter, já que ambos definem uma nova interface?
 - A diferença básica é que o Adapter adapta uma interface antiga para uma outra interface enquanto que o Façade cria uma interface completamente nova, que é mais simples.

Consequências

● Consequências:

- Isola os clientes dos componentes do subsistema, reduzindo o número de objetos com os quais o cliente precisa lidar e tornando o subsistema mais fácil de usar
- Promove fraco acoplamento entre o subsistema e seus clientes:
 - Componentes de um subsistema geralmente são fortemente acoplados
 - Com o Façade pode-se variar os componentes do subsistema sem afetar seus clientes
 - Pode eliminar dependências complexas ou circulares
 - Minimiza recompilações quando as classes do subsistema mudarem
- Não impede que aplicações utilizem diretamente as classes do subsistema se assim desejarem
 - Pode-se escolher entre facilidade de uso ou generalidade

UML



Façaade vs Proxy

Proxy

- Representa um único objeto.
- O objeto do cliente não acessa o objeto alvo diretamente.
- O objeto Proxy fornece controle de acesso ao objeto alvo único.

Façaade

- Representa um subsistema de objetos.
- O objeto do cliente tem habilidade de acessar o subsistema diretamente, se necessário.
- Um objeto Façaade fornece uma interface de alto nível simplificada para os componentes do subsistema.

Trabalho

Descrição

Suponha que temos que cadastrar um objeto Cliente que tem que lidar com o BD com instruções de consulta de um sistema.

- 1 A aplicação deve primeiro se conectar na API e verificar se o CPF do cliente já não possui um cadastro no sistema.
- 2 Em seguida, ele deve enviar os dados para o banco de dados para efetivar o cadastro, caso o CPF não exista.

Trabalho

Teríamos que definir uma interface para passar como parâmetros os dados pertinentes:

- Conexão
- API de consulta
- objeto de Cliente
- método de verificação de existe ou não o cpf cadastrado
- método que efetiva o cadastramento do usuário no sistema (caso não esteja ainda cadastrado)

Descrição

Nesse caso o padrão de projeto pode também ser utilizado para ocultar ou encapsular o acesso as dados.

Faça

Prof. Igor Avila Pereira
igor.pereira@riogrande.ifrs.edu.br

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS)
Campus Rio Grande
Divisão de Computação