**Roteiro 04**

Seguindo a sequência com mais um roteiro no nosso projeto, utilizaremos na evolução deste roteiro o padrão adapter.

**Padrão Adapter** – Permite converter a interface de uma classe em outra interface esperada pelo cliente.Adapter , fazendo com que classes com interfaces incompatíveis trabalhem juntas.

Existem basicamente dois tipos de Adapters.

- Adapter de classes

- Adapter de objetos

Ambos possuem o mesmo objetivo que é adaptar interfaces que a priori são incompatíveis. A imagem a seguir ilustra de forma bem lúdica o padrão adapter

Gráfico

O conteúdo gerado por IA pode estar incorreto.

Abaixo temos o diagrama formal e genérico do padrão Adapter. No desenvolvimento deste roteiro vamos tentar associar este modelo genérico a um exemplo prático.

Diagrama

O conteúdo gerado por IA pode estar incorreto.

**Cenário:**

Novamente vamos um jogo, só que desta vez em um cenário de guerra, com atiradores e suas armas. Inicialmente vamos criar apenas um único personagem principal que seria o nosso atirador e os diversos tipos de armas que ele possa utilizar. Temos a princípio as seguintes armas disponíveis: Pistola, Rifle, Fuzil.

**Início do projeto – Pacote : roteiro4.parte1**

1 – Dentro do projeto criar um pacote chamado **roteiro4.parte1**

2 – Inicialmente crie a interface **Arma** dentro do pacote **roteiro4.parte1.**

Neste momento já sabemos como criar uma interface em java. Dentro dela devemos apenas declarar os métodos. A implementação destes métodos será feita posteriormente nas classes concretas. Segue os métodos :

- void carregar() – permite carregar a arma.

- void atirar() – permite realizar o tiro.

- void mirar() – permite acionar a mira da arma.

3 – Devemos agora criar as classes que implementam a interface **Arma**. São elas : **Pistola, Rifle, Fuzil**.

Na implementação de cada método nas respectivas classes iremos colocar as seguintes mensagens

Para o método carregar() :

System.out.println("Carregando a Pistola");

System.out.println("Carregando o Rifle");

System.out.println("Carregando o Fuzil");

Para o método atirar() :

System.out.println("Pistola com tiro certo");

System.out.println("Tiro com rifle não falha");

System.out.println("Esse tiro faz um estrago !");

Para o método mirar() :

System.out.println("Você está na mira");

System.out.println("Zoom no alvo");

System.out.println("Alvo certeiro");

4 – Agora podemos criar a classe **Atirador**. Ela deve ter as seguintes características:

- Atributos :

private String nome;

private Arma arma;

- Construtor recebendo como parâmetro estes 2 atributos:

this.nome = nome;

this.arma = arma;

- Gets e Sets para estes 2 atributos

5 – Ainda na classe **Atirador** acrescente os métodos abaixo.

public void carregarArma(){

System.out.println("Ação - " + this.nome + " : ");

this.arma.carregar();

}

public void fazerMira(){

System.out.println("Ação - " + this.nome + " : ");

this.arma.mirar();

}

public void usarArma(){

System.out.println("Ação - " + this.nome + " : ");

this.arma.atirar();

}

6 – Para testarmos as classes do jogo crie a classe **Teste** conforme segue abaixo

package roteiro4.parte1;

public class Teste {

public static void main(String[] args) {

Arma pistola = new Pistola();

Atirador atirador01 = new Atirador("Sniper Joaquim", pistola);

System.out.println("\n" + atirador01.getNome() + " brincando com a Pistola");

atirador01.carregarArma();

atirador01.fazerMira();

atirador01.usarArma();

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Arma fuzil = new Fuzil();

atirador01.setArma(fuzil);

System.out.println("\n" + atirador01.getNome() + " brincando com o Fuzil");

atirador01.carregarArma();

atirador01.fazerMira();

atirador01.usarArma();

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Arma rifle = new Rifle();

atirador01.setArma(rifle);

System.out.println("\n" + atirador01.getNome() + " brincando com o Rifle");

atirador01.carregarArma();

atirador01.fazerMira();

atirador01.usarArma();

}

}

7 – Utilize uma ferramenta de software qualquer para geração do diagrama de classes para esta etapa do projeto (Sugestão : Astah Community). Obs.: Adicione aqui o diagrama para que seja disponibilizado no teams

**Pacote : roteiro4.parte2**

Imagine agora que no nosso projeto possamos utilizar uma nova arma de um outro fornecedor. Esta nova arma foi implementada por uma outra equipe e possivelmente não temos nem o código fonte. Isso é uma prática muito comum em projetos de software, com a utilização de bibliotecas, DLLs, plugins, em que não se tem o código fonte da implementação desejada. Até mesmo quando temos o código fonte, devemos considerar que se trata de 2 projetos diferentes e independentes, mas que precisam ser integrados para compartilhar recursos.

Pensando neste cenário, deixamos disponível na nossa pasta de trabalho o documento (.Doc) deste roteiro, os arquivos **“Roteiro4\_nova\_arma-1.0-SNAPSHOT.jar”** e **“Roteiro4\_nova\_arma.zip”**.

Para trabalhar na nossa simulação podemos seguir 3 abordagens :

1. O arquivo **.jar** é o **projeto compilado** da nova arma. Podemos utilizá-lo como biblioteca em nosso projeto. Neste caso, não temos o código fonte, e por isso é necessário importar a biblioteca .jar no Netbeans.
2. O arquivo .zip contém o **projeto completo** da nova arma e com o código fonte. Neste caso é preciso descompactar este arquivo em nosso workspace e abrir este projeto no Netbeans.
3. Criar um pacote novo dentro do nosso projeto com as classes existentes no arquivo **“Roteiro4\_nova\_arma.zip”**.

Nesta etapa do projeto podemos utilizar o recurso de uma forma ou de outra (itens a ,b ou c). O importante é que iremos utilizar o padrão Adapter, utilizando de forma integrada um novo recurso no nosso projeto.

Você está livre para adotar qualquer uma das abordagens, mas faremos aqui algumas observações

* É importante conhecer bem a IDE que está trabalhando, pois existem pequenas diferenças ao executar ações como : importar projeto (jar), incluir dependências e etc.
* Nem sempre importar o arquivo .jar tem o comportamento esperado, pois podem existir incompatibilidades entre a versões do java (seu projeto x projeto a ser importado)
* A abordagem “c” sem dúvida é a mais fácil, mas numa situação real isso seria bem difícil de acontecer. Estamos falando a integração de 2 projetos e na abordagem “c” estamos fazendo uma cópia das classes para um pacote pertencente ao nosso projeto. Então na prática não seriam 2 projetos de verdade
* Na abordagem “b” temos de fato 2 projetos e devemos criar uma relação de dependência entre eles

Caso queira seguir a abordagem “b”, apenas a título de curiosidade, mostraremos aqui com fazer isso no Netbeans, lembrando que em outras IDEs o processo pode ser diferente

* Primeiro descompacte o arquivo . zip na sua pasta de trabalho
* Em seguida abra os 2 projetos no NetBeans (PadroesRoteiros e Roteiro4\_nova\_arma)

Interface gráfica do usuário, Aplicativo, Word

O conteúdo gerado por IA pode estar incorreto.

* Devemos ter os 2 projetos abertos no Netbeans

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

* Vamos criar agora uma relação de dependência entre os 2 projetos, adicionando uma dependência dentro do projeto PADROESroteiros

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

* Na tela abaixo na aba “Open Projects”, selecione o projeto Roteiro4\_nova\_arma e todos os campos são preenchidos automaticamente. Em seguida confirme a adição da dependência.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

* Abaixo temos a tela após adicionar a dependência. Agora um projeto pode utilizar recursos do outro projeto.

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

* Abaixo temos um exemplo de código onde uma variável do tipo ShotGun. Esta é a nova arma que estamos importando . Observe que no nosso código temos apenas 3 armas (Pistola, Fuzil e Rifle). Agora podermos utilizar a ShotGun

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

**Vamos ao passo a passo deste pacote**

1 – No mesmo projeto crie o pacote **roteiro4.parte2**

2 – Copie todas as classes criadas na parte1 para o novo pacote.

3 – Conforme descrito no cenário acima, escolha a abordagem de integração entre os projetos a ser adotada (itens a, b ou c).

4 – Devemos criar agora a classe **ShotGunAdapter**. A nova arma do outro fornecedor se chama **ShotGun**, e por isso estamos criando a classe ShotGunAdapter. Esta classe será responsável por adaptar um projeto de software ao outro. Em outras palavras, esta classe é responsável por compatibilizar a nossa implementação de arma a implementação do outro fornecedor.

A arma ShotGun possui os métodos diferentes das nossas armas. Segue abaixo os métodos da ShotGun:

- loadGun() – permite que a ShotGun seja carregada. Adaptável ao nosso método carregar().

- shootKill() – Tiro de alta destruição. Adaptável ao nosso método atirar().

- targetEnemy () – permite uma mira fácil. Adaptável ao nosso método mirar().

Atenção : Vejam abaixo que a classe ShotGunAdpater herda de ShotGun, mas também implementa a nossa classe Arma. E assim vamos implementando métodos a método, compatibilizando uma classe a outra.

package roteiro4.parte2;

import armas.artilharia.ShotGun;

public class ShotGunAdapter extends ShotGun implements Arma{

@Override

public void carregar() {

this.loadGun();

}

@Override

public void atirar() {

this.shotKill();

}

@Override

public void mirar() {

this.targetEnemy();

}

}

Seguindo para os testes !

5 – Para testarmos as classes do jogo vamos adicionar o trecho de código referente a ShotGun.

package roteiro4.parte2;

public class Teste {

public static void main(String[] args) {

Arma pistola = new Pistola();

Atirador atirador01 = new Atirador("Sniper Joaquim", pistola);

System.out.println("\n" + atirador01.getNome() + " brincando com a Pistola");

atirador01.carregarArma();

atirador01.fazerMira();

atirador01.usarArma();

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Arma fuzil = new Fuzil();

atirador01.setArma(fuzil);

System.out.println("\n" + atirador01.getNome() + " brincando com o Fuzil");

atirador01.carregarArma();

atirador01.fazerMira();

atirador01.usarArma();

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Arma rifle = new Rifle();

atirador01.setArma(rifle);

System.out.println("\n" + atirador01.getNome() + " brincando com o Rifle");

atirador01.carregarArma();

atirador01.fazerMira();

atirador01.usarArma();

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Arma shotgun = new ShotGunAdapter();

atirador01.setArma(shotgun);

System.out.println("\n" + atirador01.getNome() + " brincando com a ShotGun");

atirador01.carregarArma();

atirador01.fazerMira();

atirador01.usarArma();

}

}

6 – Utilize uma ferramenta de software qualquer para geração do diagrama de classes para esta etapa do projeto (Sugestão : Astah Community). Obs.: Adicione aqui o diagrama para que seja disponibilizado no teams

Obs: Outra possibilidade para geração dos diagramas é o StarUML : <https://staruml.io/> . Para fazer a engenharia reversa do código siga as orientações conforme o vídeo a seguir : <https://www.youtube.com/watch?v=z6DDuZQArro>

7 – Associe o papel de cada classe do diagrama formal padrão Adapter indicado no início deste documento as classes criadas no nosso roteiro. Ou seja :

- Quem representa a classe Adapter ?

- Quem representa a classe Adaptee ?

- Quem representa a classe Target ?

- Quem representa a classe Client ?