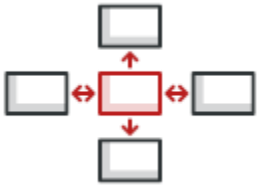


Modelos, Métodos e Técnicas de Engenharia de Software
Visão e análise de projeto Padrões Prática 3 – Mediator (17)

Prof. Osmar de Oliveira Braz Junior

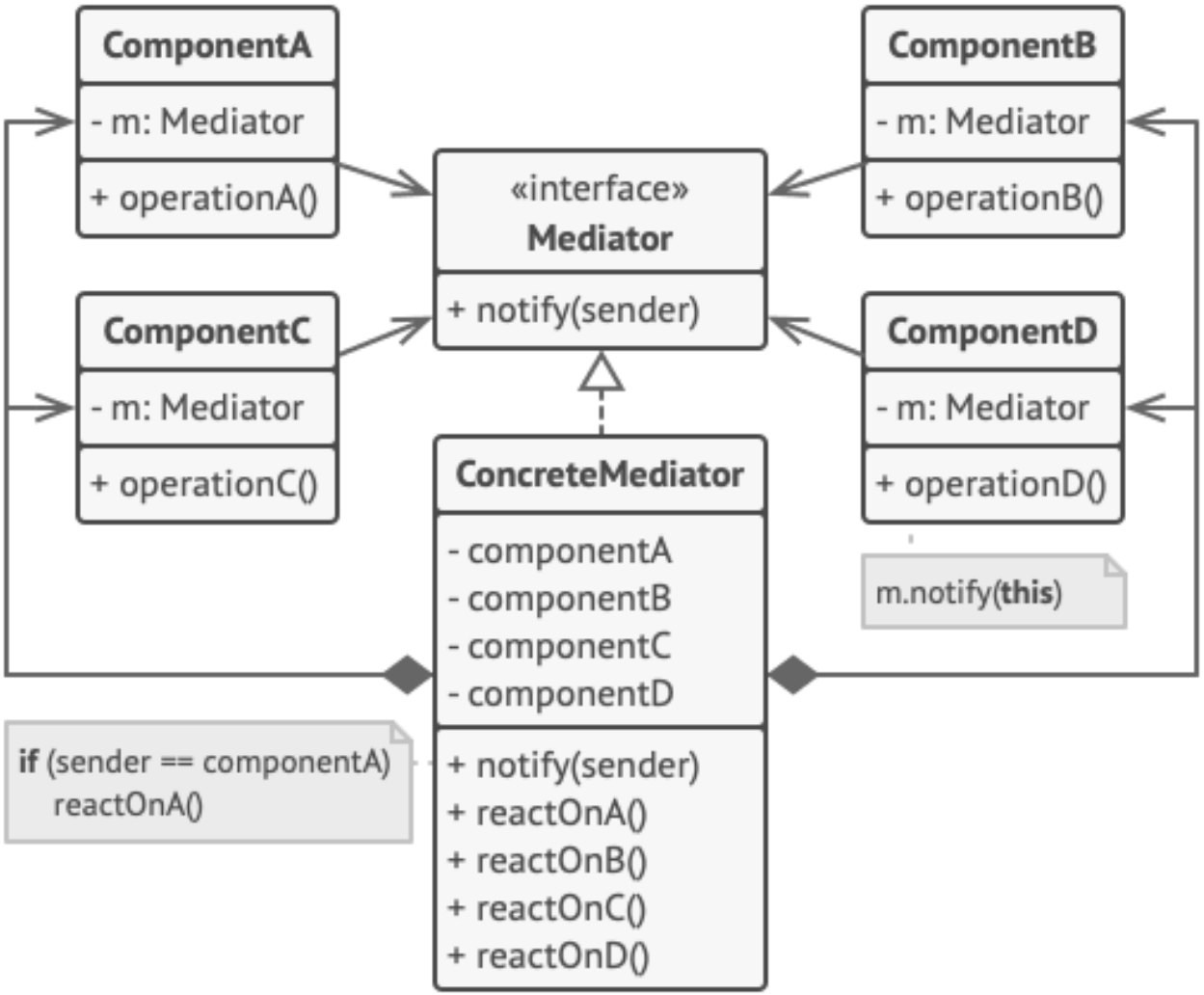
Prof. Richard Henrique de Souza

Objetivos

- Aplicar padrão comportamental ***Interpreter*** em situação problema.

17. Mediator

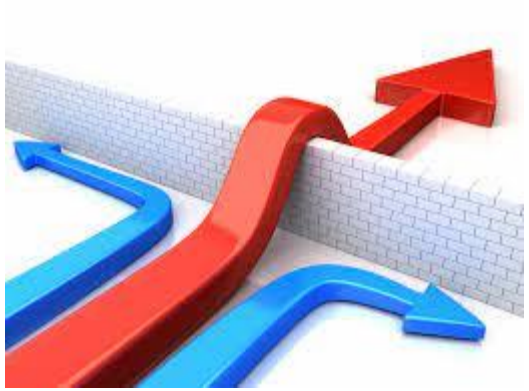
Estrutura



https://refactoring.guru/pt-br/design-patterns/mediator

Importante

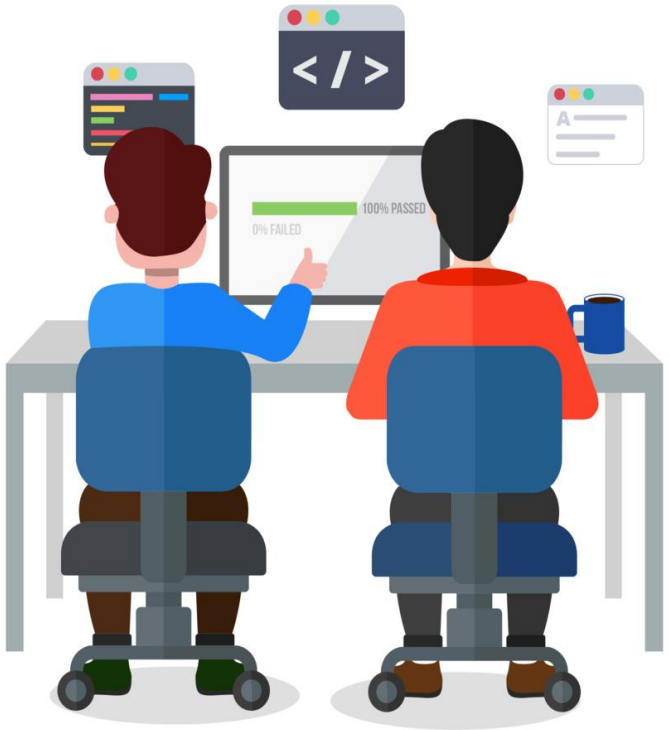
Siga os ROTEIROS !!!



Atividade em Grupo

Para esta atividade crie grupos de 2 alunos, para desenvolver a atividade segundo ***Pair Programming.***

Navegador



Piloto

Extreme Programming, metodologia ágil criada por Kent Beck em 1996.

Pair Programming

- Um é o **piloto**, responsável por escrever o código, o outro o navegador, acompanha a escrita de código e verificar se está de acordo com os **padrões do projeto** e de encontro à solução necessária.
- A intenção desta técnica é **evitar** erros de lógica, e ter um código mais confiável e melhor estruturado, utilizando-se para isso a máxima de que “**duas cabeças pensam melhor do que uma**”.

Preparação do ambiente

- Acesso a ferramenta **draw.io(https://app.diagrams.net/)** para realizar a modelagem.
- Escolha a sua linguagem de programação de preferência
- Escolha uma IDE ou o **git.dev**
- Crie um repositório no github(https://github.com/) para que todos os membros da equipe possam colaborar no desenvolvimento.

Atividade prática

1

17. Mediator
Propósito

- Definir um objeto que encapsula a informação de como um conjunto de outros objetos interagem entre si.
- Promove o acoplamento fraco, permitindo que você altere a forma de interação sem alterar os objetos que interagem.
- Também conhecido como: Mediador, Intermediário, Intermediary, Controlador ou Controller.

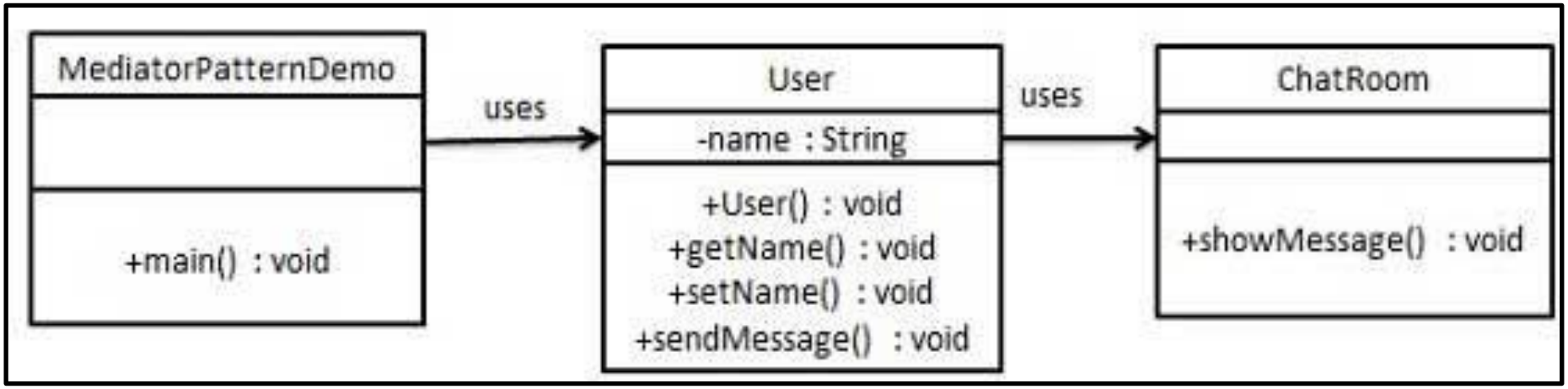
17. Mediator

- Usar este padrão quando…
  - Um conjunto de objetos se comunica de uma forma bem determinada, porém complexa;
  - Reutilizar uma classe é difícil pois ela tem associação com muitas outras;
  - Um comportamento que é distribuído entre várias classes deve ser extensível sem ter que criar muitas subclasses.

17. Mediator

- Vantagens e desvantagens
  - Limita extensão por herança:
    - Para estender ou alterar o comportamento, basta criar uma subclasse do mediador.
  - Desacopla objetos:
    - Desacoplamento promove o reuso.
  - Simplifica o protocolo:
    - Relações Colleagues x Mediator são mais simples de manter do que muitas espalhadas;
  - Fica mais claro como os objetos interagem.
    - Exagero pode levar a sistema monolítico.

17. Mediator


MediatorPatternDemo
+main() : void
uses
User
-name : String
+User() : void
+getName() : void
+setName() : void
+sendMessage() : void
uses
ChatRoom
+showMessage() : void

17. Mediator
Passo 1

Crie uma classe mediadora.

ChatRoom.java

```
import java.util.Date;

public class ChatRoom {
   public static void showMessage(User user, String message){
      System.out.println(new Date().toString() + " [" + user.getName() + "] : " + message);
   }
}
```

17. Mediator
Passo 2

Criar classe de usuário

User.java

```
public class User {
   private String name;

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

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

   public void sendMessage(String message){
      ChatRoom.showMessage(this,message);
   }
}
```

17. Mediator
Passo 3

Use o objeto Usuário para mostrar as comunicações entre eles.

MediatorPatternDemo.java

```
public class MediatorPatternDemo {
   public static void main(String[] args) {
      User robert = new User("Robert");
      User john = new User("John");

      robert.sendMessage("Hi! John!");
      john.sendMessage("Hello! Robert!");
   }
}
```

17. Mediator
Passo 4

- Terminamos
  - Teste sua implementação

Compile e **Mostre** o código para o professor
  - Pense, o que você fez aqui ?

Lembre de salvar no seu github

Conclusão

Os padrões comportamentais tem como principal função designar responsabilidades entre objetos.

Referências

- PRESSMAN, Roger; MAXIM, Bruce. Engenharia de software: uma abordagem profissional. 8.ed. Bookman, 2016. E-book. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788580555349

- SOMMERVILLE, Ian. Engenharia de software. 9. ed. São Paulo: Pearson Prentice Hall, 2011. E-book. Disponível em: https://plataforma.bvirtual.com.br/Leitor/Publicacao/2613/epub/0

- LARMAN, Craig. Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e desenvolvimento iterativo. 3. ed Porto Alegre: Bookman, 2007. E-book. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788577800476

Fim