**Documentação do padrão de projeto**

**Mediator**

**Matheus Carvalho Castro1, Rafaela Ferreira2, Lucas Aurélio3, Samuel Hipólito4**

**Tiago Nunes5 ,Kaique Icaro6**

Centro Universitário Una Contagem  
Caixa Postal – Contagem – MG – Brazil

Curso de análise e desenvolvimento de sistemas-Una Contagem

***Abstract.*** *This meta-article describes the mediator design pattern, with its functionalities, characteristics and its objectives within the subject of programming, being very specific in face of its descriptions.*

***Resumo.*** *Este meta-artigo descreve o padrão de projeto mediator, com suas funcionalidades, características e seus objetivos dentro do assunto da programação, sendo bem específico diante de suas descrições.*

**1. Resumo da funcionalidade do Mediator**

O padrão Mediator restringe a comunicação entre objetos usando um objeto mediator por onde ocorre a comunicação, evitando assim um número excessivo de acoplagem de objetos em outros.

**2. Construção do código**

**2.1. Início**

Para dar início à criação do código do projeto, o iniciamos respeitando a seguinte ordem de criação:

* Criação do Pacote PadraoMediator
* Criação Classe Participante
* Criação Classe AbstractChatSala
* Criação Classe ChatSala
* Criação Classe Membro
* Criação Classe NãoMembro
* Criação Classe Module1

**2.2. Meio**

Após a criação das classes, inserimos os complementos necessários.

* Classe Participante

***package padraomediator;***

***public abstract class Participante***

***{***

***private Chatsala \_chatsala;***

***private String \_nome;***

***// Construtor***

***public Participante(String nome)***

***{***

***this.\_nome = nome;***

***}***

***// Pega o nome do participante***

***public final String getNome()***

***{***

***return \_nome;***

***}***

***// Pega a sala de chat***

***public final Chatsala getChatsala()***

***{***

***return \_chatsala;***

***}***

***public final void setChatsala(Chatsala value)***

***{***

***\_chatsala = value;***

***}***

***// Envia mensagem para um participante***

***public final void Enviar(String para, String mensagem)***

***{***

***\_chatsala.Enviar(\_nome, para, mensagem);***

***}***

***// Recebe mensagem de um participante***

***public void Receber(String de, String mensagem)***

***{***

***System.out.printf("%1$s para %2$s: '%3$s'" + "\r\n", de, getNome(), mensagem);***

***}***

***}***

* Classe Chatsala

***package padraomediator;***

***import java.util.\*;***

***public class Chatsala extends AbstractChatSala***

***{***

***private HashMap<String, Participante> \_participantes = new HashMap<String, Participante>();***

***@Override***

***public void Registro(Participante \_participante)***

***{***

***if (!\_participantes.containsValue(\_participante))***

***{***

***\_participantes.put(\_participante.getNome(), \_participante);***

***}***

***\_participante.setChatsala(this);***

***}***

***@Override***

***public void Enviar(String de, String para, String mensagem)***

***{***

***Participante \_participante = \_participantes.get(para);***

***if (\_participante != null)***

***{***

***\_participante.Receber(de, mensagem);***

***}***

***}***

***}***

* Classe AbstractChatSala

***package padraomediator;***

***//Classe abstrata responsável pela mediação***

***public abstract class AbstractChatSala***

***{***

***public abstract void Registro(Participante participante);***

***public abstract void Enviar(String de, String para, String message);***

***}***

* Classe Membro

***package padraomediator;***

***public class Membro extends Participante***

***{***

***// Construtor***

***public Membro(String nome)***

***{***

***super(nome);***

***}***

***//Sobrescreve o método Receber***

***@Override***

***public void Receber(String de, String mensagem)***

***{***

***System.out.print("para Membro : ");***

***super.Receber(de, mensagem);***

***}***

***}***

* Classe NaoMembro

***package padraomediator;***

***public class NaoMembro extends Participante***

***{***

***// Construtor da clasee***

***public NaoMembro(String nome)***

***{***

***super(nome);***

***}***

***//Sobrescreve o método de recebimento***

***@Override***

***public void Receber(String de, String mensagem)***

***{***

***System.out.print("Para NaoMembro : ");***

***super.Receber(de, mensagem);***

***}***

***}***

**2.3. Final**

* Estruturação da Classe Module1.

***pacote padraomediador;***

***aula final pública Módulo1***

***{***

***public static void main(String[] args)***

***{***

***//Inicia um novo chat***

***Chatsala chatsala = new Chatsala();***

***//Cria os Participantes do chat***

***Participante Samuel = new Membro("Samuel");***

***Participante Rafaella = new Membro("Rafaella");***

***Participante Lucas = new Membro("Lucas");***

***Participante Kaique = new Membro("Kaique");***

***Participante Tiago = new NaoMembro("Tiago");***

***Participante Matheus = new NaoMembro("Matheus");***

***//registro de participantes***

***chatsala.Registro(Samuel);***

***chatsala.Registro(Rafaella);***

***chatsala.Registro(Lucas);***

***chatsala.Registro(Kaique);***

***chatsala.Registro(Tiago);***

***chatsala.Registro(Matheus);***

***// Inicia o chat***

***Rafaella.Enviar("Tiago", "Olá, Tiago!");***

***Tiago.Enviar("Rafaella", "Como vai você?");***

***Rafaella.Enviar("Tiago", "Tudo bem!");***

***Tiago.Enviar("Lucas", "Como você esta?");***

***Lucas.Enviar("Tiago", "Tudo tranquilo...");***

***}***

***}***

**3. Vantagens e desvantagens do Mediator**

**3.1. Vantagens**

**3.1.1** Pode inserir novos componentes sem alterar os já existentes

**3.1.2** A comunicação pode ser alterada sem mudar os componentes, uma vez em que ela está centralizada.

**3.1.1** Redução significativa de aclopamentos

**3.2. Desvantagens**

**3.2.1** O mediador tende a ser tornar mais complexo com a evolução do código

**Referências**

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

<https://www.macoratti.net/11/06/pp_med1.htm>