Bien sûr, je peux vous fournir un exemple simple mais complet qui utilise le patron de conception Mediator en UML et une implémentation correspondante en C#. Supposons que nous ayons un système de chat en salle où plusieurs utilisateurs peuvent envoyer et recevoir des messages.

**UML**

En UML, nous aurons quatre classes principales :

1. ChatRoom - Le médiateur
2. User - Le collègue
3. ChatMessage - Un message de chat simple
4. MainApp - Classe de démarrage de l'application

**Relations :**

* ChatRoom connaît tous les User.
* User connaît seulement ChatRoom.

**Méthodes :**

* ChatRoom a SendMessage(User, string).
* User a ReceiveMessage(ChatMessage) et SendMessage(string).

**Code C#**

Voici comment vous pourriez implémenter cela en C#.

**ChatMessage.cs**

csharp

public class ChatMessage

{

public string Sender { get; set; }

public string Message { get; set; }

public ChatMessage(string sender, string message)

{

Sender = sender;

Message = message;

}

}

**ChatRoom.cs (Mediator)**

csharp

using System.Collections.Generic;

public class ChatRoom

{

private readonly List<User> \_users;

public ChatRoom()

{

\_users = new List<User>();

}

public void AddUser(User user)

{

\_users.Add(user);

}

public void SendMessage(User sender, string message)

{

var chatMessage = new ChatMessage(sender.Name, message);

foreach (var user in \_users)

{

if (user != sender)

{

user.ReceiveMessage(chatMessage);

}

}

}

}

**User.cs (Colleague)**

csharp

public class User

{

public string Name { get; }

private readonly ChatRoom \_chatRoom;

public User(string name, ChatRoom chatRoom)

{

Name = name;

\_chatRoom = chatRoom;

\_chatRoom.AddUser(this);

}

public void SendMessage(string message)

{

\_chatRoom.SendMessage(this, message);

}

public void ReceiveMessage(ChatMessage chatMessage)

{

System.Console.WriteLine($"{Name} received a message from {chatMessage.Sender}: {chatMessage.Message}");

}

}

**MainApp.cs**

csharp

class MainApp

{

static void Main()

{

ChatRoom chatRoom = new ChatRoom();

User alice = new User("Alice", chatRoom);

User bob = new User("Bob", chatRoom);

User carol = new User("Carol", chatRoom);

alice.SendMessage("Hi, I am Alice.");

bob.SendMessage("Hi, Alice! I am Bob.");

carol.SendMessage("Hello everyone!");

// Output:

// Bob received a message from Alice: Hi, I am Alice.

// Carol received a message from Alice: Hi, I am Alice.

// Alice received a message from Bob: Hi, Alice! I am Bob.

// Carol received a message from Bob: Hi, Alice! I am Bob.

// Alice received a message from Carol: Hello everyone!

// Bob received a message from Carol: Hello everyone!

}

}

Dans cet exemple, ChatRoom agit comme un médiateur qui facilite la communication entre différents utilisateurs. Chaque utilisateur ne connaît que le ChatRoom, ce qui diminue le couplage entre les utilisateurs.