**Exercice : Conception d'un système de chat**

**Description** : Vous êtes amené à concevoir un système de chat simple qui permet à plusieurs utilisateurs de discuter en temps réel. Le système doit permettre à un utilisateur de rejoindre une salle de chat, de quitter une salle de chat, et d'envoyer un message à tous les autres utilisateurs de la salle.

**Contraintes** :

* Utilisez le design pattern Médiateur pour coordonner la communication entre les utilisateurs.
* Implémentez le système en C#.

**UML** :

1. **Interface IUser** :
   * +SendMessage(message: string): void
   * +ReceiveMessage(message: string): void
2. **Classe ConcreteUser** implémente IUser :
   * -mediator: IChatMediator
   * +ConcreteUser(mediator: IChatMediator)
   * +SendMessage(message: string): void
   * +ReceiveMessage(message: string): void
3. **Interface IChatMediator** :
   * +AddUser(user: IUser): void
   * +SendMessage(message: string, user: IUser): void
4. **Classe ConcreteChatMediator** implémente IChatMediator :
   * -users: List<IUser>
   * +AddUser(user: IUser): void
   * +SendMessage(message: string, user: IUser): void

**Implémentation C#**:

csharp

using System;

using System.Collections.Generic;

// Médiateur

public interface IChatMediator

{

void AddUser(IUser user);

void SendMessage(string message, IUser user);

}

public class ConcreteChatMediator : IChatMediator

{

private List<IUser> users = new List<IUser>();

public void AddUser(IUser user)

{

users.Add(user);

}

public void SendMessage(string message, IUser user)

{

foreach (var u in users)

{

if (u != user)

{

u.ReceiveMessage(message);

}

}

}

}

// Utilisateur

public interface IUser

{

void SendMessage(string message);

void ReceiveMessage(string message);

}

public class ConcreteUser : IUser

{

private IChatMediator mediator;

public ConcreteUser(IChatMediator mediator)

{

this.mediator = mediator;

mediator.AddUser(this);

}

public void SendMessage(string message)

{

Console.WriteLine("Envoi du message : " + message);

mediator.SendMessage(message, this);

}

public void ReceiveMessage(string message)

{

Console.WriteLine("Message reçu : " + message);

}

}

// Test

public class Program

{

public static void Main()

{

IChatMediator mediator = new ConcreteChatMediator();

IUser user1 = new ConcreteUser(mediator);

IUser user2 = new ConcreteUser(mediator);

user1.SendMessage("Salut tout le monde !");

}

}

**Correction** :

Dans cette solution, le pattern Médiateur est utilisé pour coordonner la communication entre différents utilisateurs. La classe ConcreteChatMediator joue le rôle de médiateur et maintient une liste de tous les utilisateurs. Quand un utilisateur envoie un message via SendMessage, le médiateur s'assure que tous les autres utilisateurs reçoivent ce message via leur méthode ReceiveMessage.

Haut du formulaire

Bas du formulaire