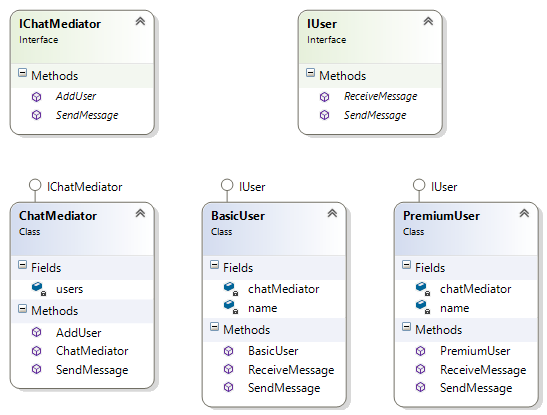
**Mediator Pattern**

Consider a chat application having several participants. Each participant cannot be connected to all others due to firewalls, proxies or other problems. Instead, providing a hub, which is the mediator class for them to connect, will serve the purpose.

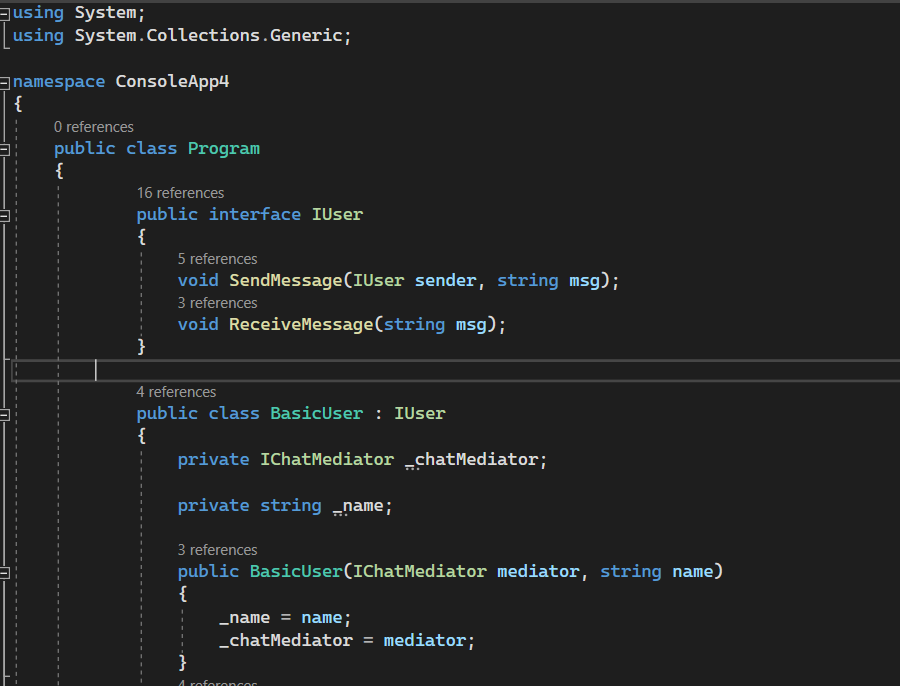
Whenever a user will send a message then it will be delivered to all other participants in the chat. Any online user of the chat group can send and receive the message, by instantiating the chat mediator. Below is a UML of a basic simulation of the scenario.

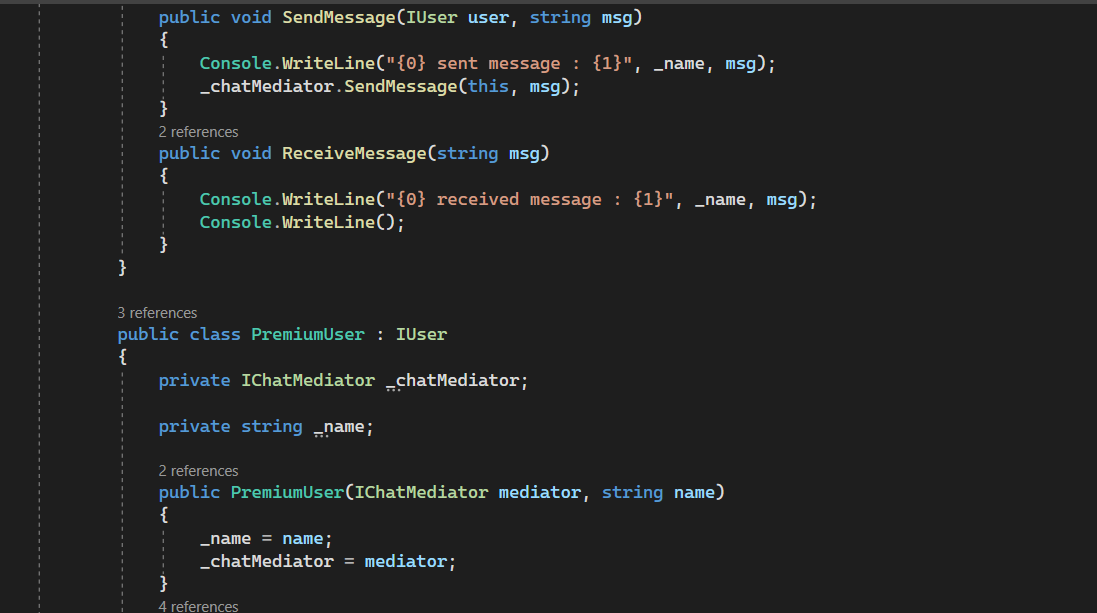


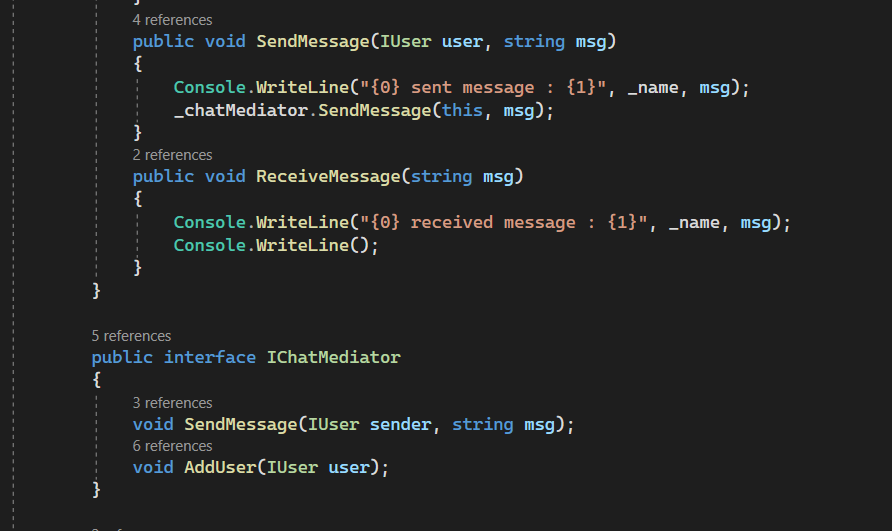
Steps:

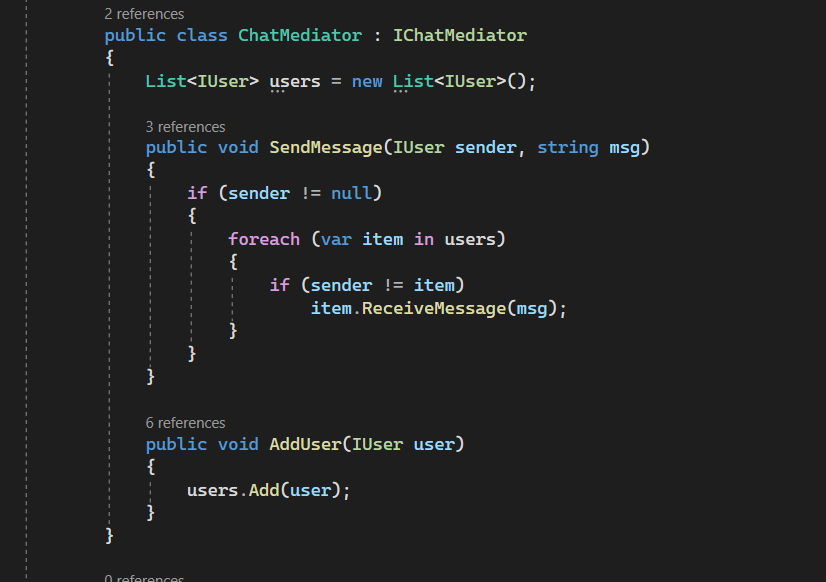
1. Create interfaces for User and ChatMediator as shown in the UML
2. Implement ChatMediator class which will have a list or array to hold the users. Implement Add user method which will add the user object to the ‘users’ list. Implement ‘SendMessage’ method to send the messages to all the users. Please note that this method should invoke ‘ReceiveMessage’ method of each user object ‘users’ list.
3. Implement two user classes BasicUser and PremiumUser from the IUser interface
4. Implement methods to send and receive message. The user classes will hold a reference to the ChatMediator class. Please note that the ‘send message’ of this class will invoke ‘send message of the ChatMediator , which will broadcast the message. Implement the ‘ReceiveMessage’ method to print the name of the user and the message.
5. From the client invoke create objects for ChatMediator. Add couple of users to the ChatMediator. Instantiate a separate user who will be the source of the messages. Do not add this object to the ChatMediator class. Invoke the send message of ChatMediator to send a message.
6. Run the program and validate if the all the users have been broadcasted the message except the source user.

**Implementation ->**

****

****

****

****

****

Program.cs

using System;

using System.Collections.Generic;

namespace ConsoleApp4

{

public class Program

{

public interface IUser

{

void SendMessage(IUser sender, string msg);

void ReceiveMessage(string msg);

}

public class BasicUser : IUser

{

private IChatMediator \_chatMediator;

private string \_name;

public BasicUser(IChatMediator mediator, string name)

{

\_name = name;

\_chatMediator = mediator;

}

public void SendMessage(IUser user, string msg)

{

Console.WriteLine("{0} sent message : {1}", \_name, msg);

\_chatMediator.SendMessage(this, msg);

}

public void ReceiveMessage(string msg)

{

Console.WriteLine("{0} received message : {1}", \_name, msg);

Console.WriteLine();

}

}

public class PremiumUser : IUser

{

private IChatMediator \_chatMediator;

private string \_name;

public PremiumUser(IChatMediator mediator, string name)

{

\_name = name;

\_chatMediator = mediator;

}

public void SendMessage(IUser user, string msg)

{

Console.WriteLine("{0} sent message : {1}", \_name, msg);

\_chatMediator.SendMessage(this, msg);

}

public void ReceiveMessage(string msg)

{

Console.WriteLine("{0} received message : {1}", \_name, msg);

Console.WriteLine();

}

}

public interface IChatMediator

{

void SendMessage(IUser sender, string msg);

void AddUser(IUser user);

}

public class ChatMediator : IChatMediator

{

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

public void SendMessage(IUser sender, string msg)

{

if (sender != null)

{

foreach (var item in users)

{

if (sender != item)

item.ReceiveMessage(msg);

}

}

}

public void AddUser(IUser user)

{

users.Add(user);

}

}

static void Main()

{

ChatMediator mediator = new ChatMediator();

IUser John = new BasicUser(mediator, "John");

IUser Mike = new BasicUser(mediator, "Mike");

IUser Kane = new BasicUser(mediator, "Kane");

IUser Jenny = new PremiumUser(mediator, "Jenny");

IUser Chris = new PremiumUser(mediator, "Chris");

mediator.AddUser(John);

mediator.AddUser(Mike);

mediator.AddUser(Kane);

mediator.AddUser(Jenny);

mediator.AddUser(Chris);

John.SendMessage(John, "Hello everyone");

Console.WriteLine();

Mike.SendMessage(Mike, "Hi John");

Console.WriteLine();

Chris.SendMessage(Chris, "Merry Christmas");

Console.ReadLine();

}

}

}

**Output ->**

